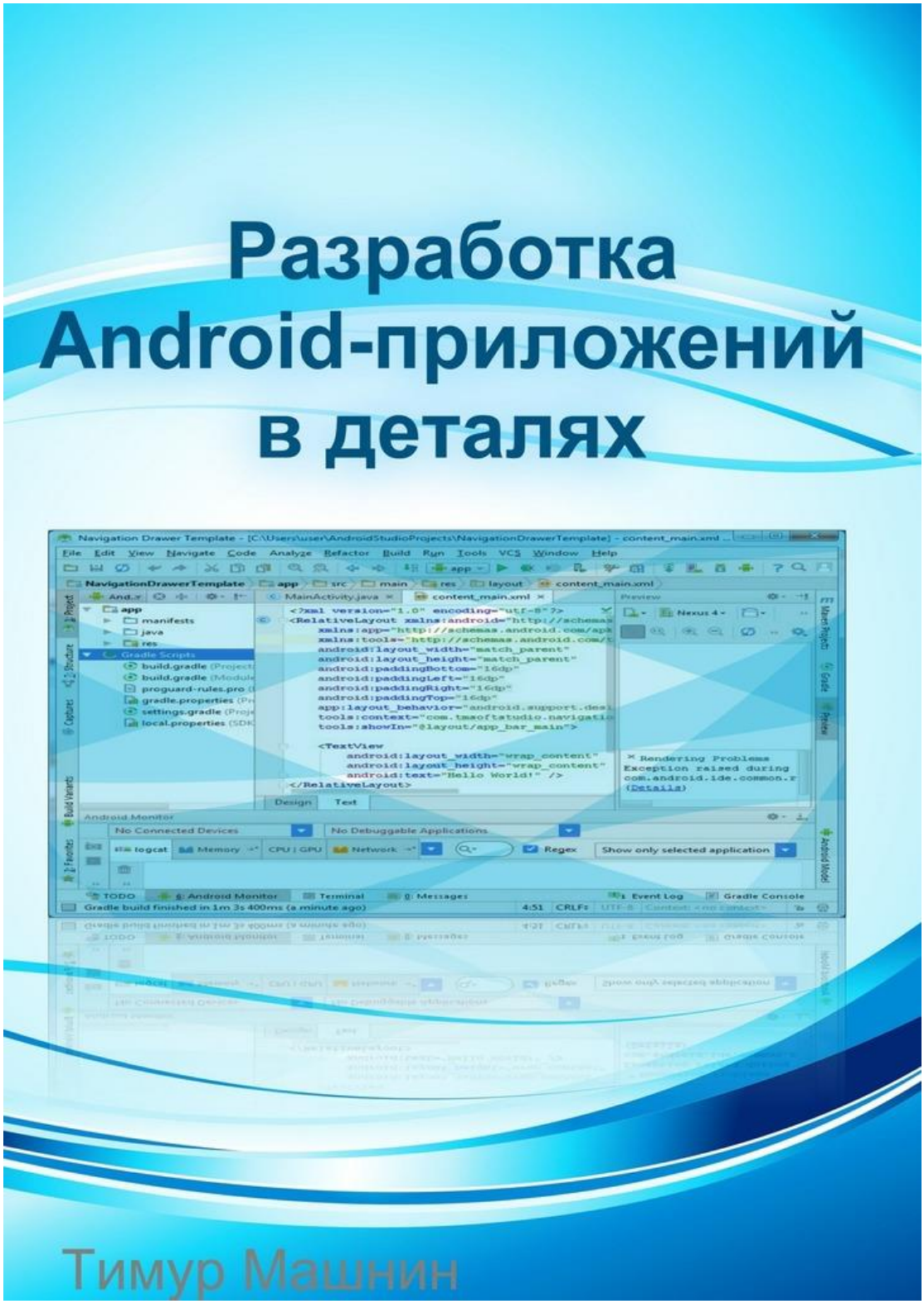
Тимур Сергеевич Машнин
Разработка Android-приложений в деталях



ISBN 9785448304507

Аннотация

В книге приведены некоторые рецепты разработки Android-приложений и их примеры,

рассмотрена работа в среде Eclipse и Android Studio, разработка мобильных сайтов и гибридных мобильных приложений.

Разработка Android-приложений в деталях
Тимур Сергеевич Машнин

Создано в интеллектуальной издательской системе Ridero

Введение

Пригласить автора в проект [битая ссылка] admin@tmsoftstudio.com

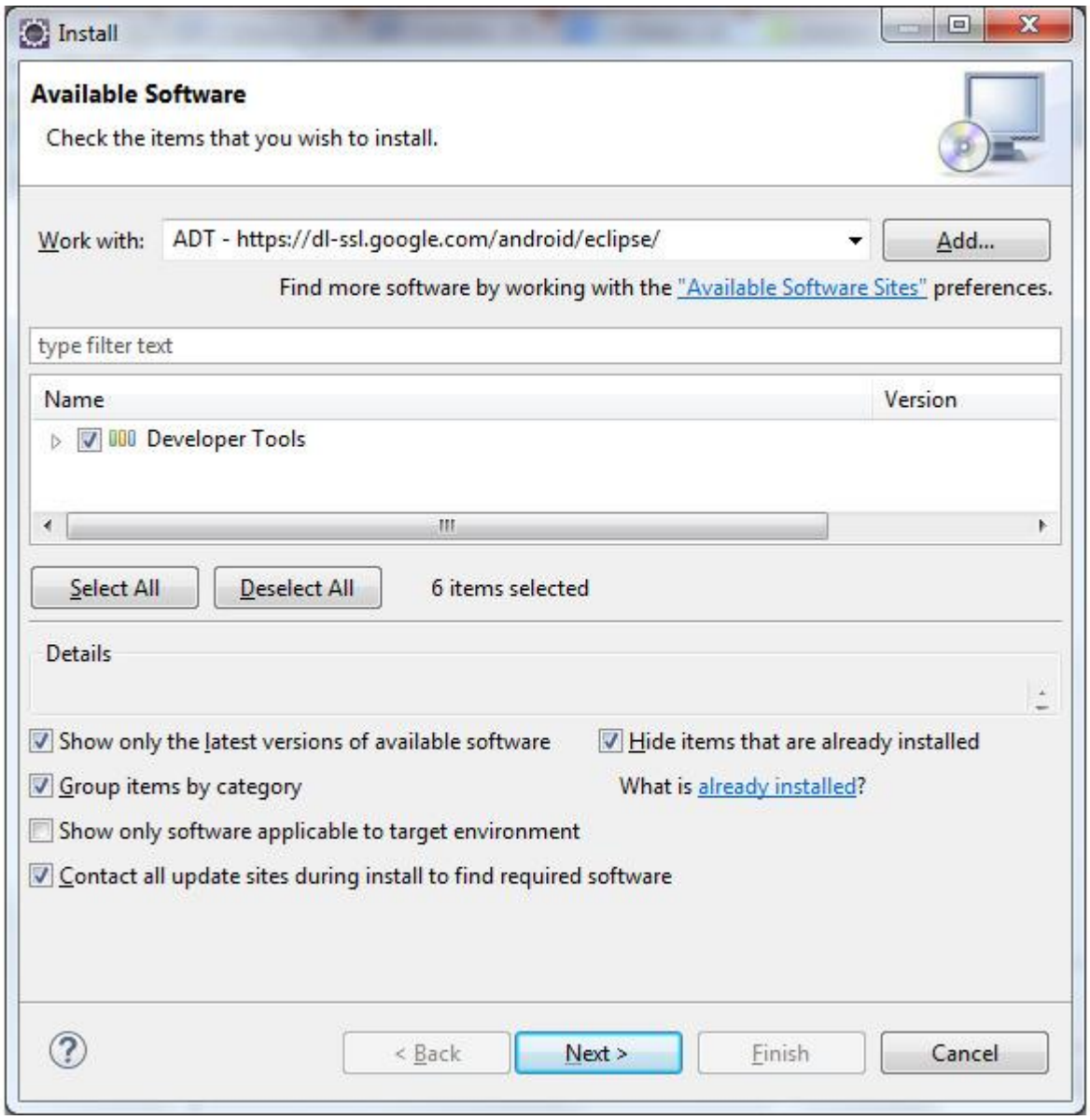
Разработку Android-приложений можно вести как в среде Eclipse, так и в среде Android Studio. Хотя среда Android Studio является официально поддерживаемой средой разработки, среда Eclipse не теряет актуальности из-за своей универсальности в разработке Java-приложений самого широкого спектра применений.

Поддержку разработки Android-приложений в среде Eclipse обеспечивает Eclipse-плагин Android Development Tools (ADT) (http://developer.android.com/sdk/eclipse-adt.html).

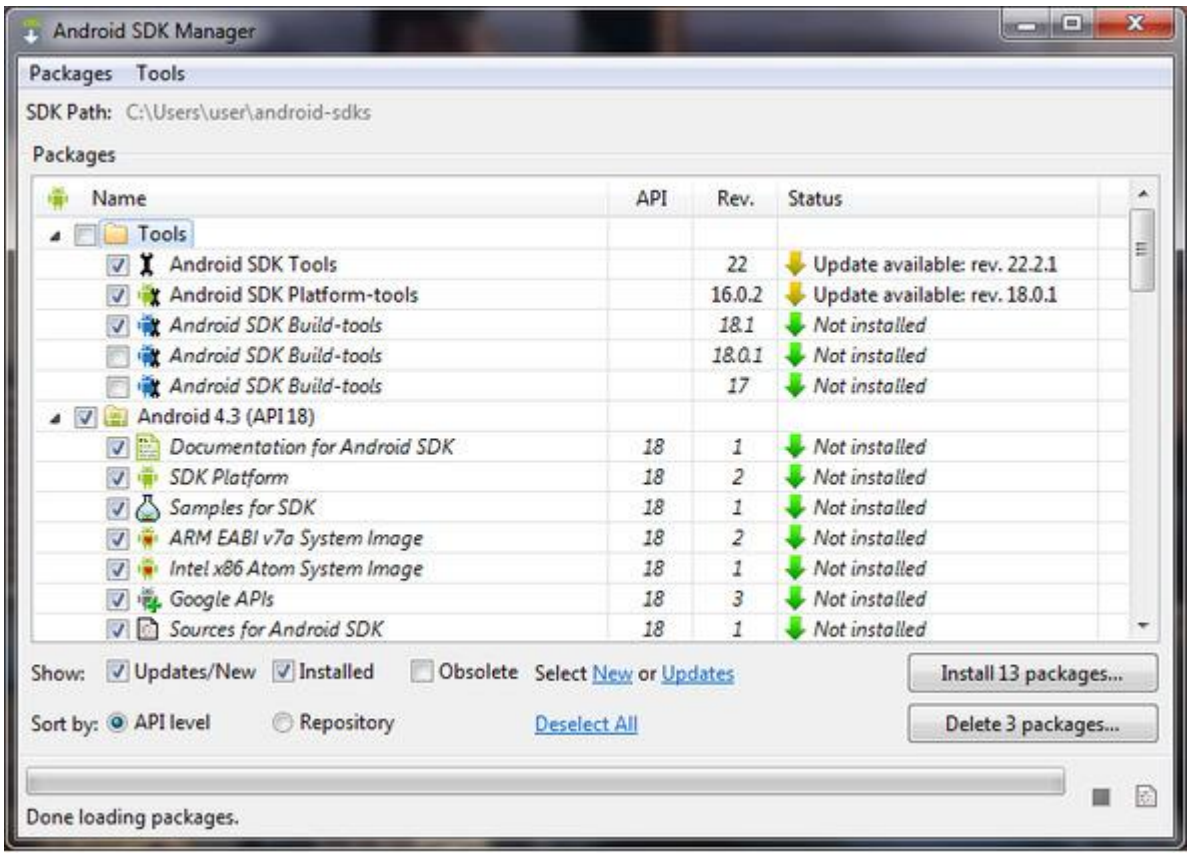
ADT-плагин помогает создать Android-проект, разработать UI-интерфейс приложения на основе программного интерфейса Android Framework API, отладить Android-приложение и подготовить подписанный файл. apk к публикации.

Установка ADT плагина

Для установки ADT-плагина откроем среду Eclipse IDE for Java EE Developers и в меню **Help** выберем команду **Install New Software** . В списке **Work with:** нажмем кнопку **Add** , в поле **Name:** введем имя плагина ADT, а в поле **Location:** – адрес https://dl-ssl.google.com/android/eclipse/ хранилища плагина, нажмем кнопку **OK** , в мастере **Install** отметим флажок **Developer Tools** и нажмем кнопку **Next** .



После установки ADT-плагина и перезапуска среды Eclipse может открыться окно приложения Android SDK Manager набора разработчика Android SDK.



Сам по себе дистрибутив набора Android SDK (http://developer.android.com/sdk/index.html) включает в себя набор инструментов SDK Tools, SDK Platform-tools, а также приложения AVD Manager и SDK Manager.

Приложение SDK Manager дает возможность устанавливать и обновлять компоненты набора Android SDK, а также запускать приложение AVD Manager и управлять URL-адресами дополнений.

Приложение SDK Manager можно также запустить из среды Eclipse с помощью команды **Android SDK Manager** меню **Window** перспективы **Java** .

Набор инструментов SDK Tools обеспечивает отладку и тестирование Android-приложений. Набор инструментов SDK Platform-tools обеспечивает поддержку самой последней версии Android-платформы и включает в себя инструмент Android Debug Bridge (adb), позволяющий взаимодействовать с эмулятором или Android-устройством. Приложение AVD Manager предоставляет GUI-интерфейс для моделирования различных конфигураций Android-устройств, используемых Android-эмулятором запуска приложений в среде выполнения Android. Набор Android Build Tools обеспечивает сборку кода Android-приложения.

Для разработки Android-приложений требуется установка конкретной Android-платформы, включающей в себя библиотеки платформы, системные изображения, образцы кода, оболочки эмуляции.

Поэтому, используя приложение SDK Manager, установим последнюю возможную версию Android-платформы и наиболее распространенную или минимальную версию Android-платформы. Обратная совместимость между максимальной и минимальной версиями осуществляется с помощью библиотеки Android Support Library каталога Extras.

Дополнительно можно загрузить другие версии Android-платформы, документацию, примеры и различные дополнения набора Android SDK.

Различные версии API Android-платформы отличаются друг от друга наличием новых пакетов, а также изменениями в существующих пакетах.

Помимо изменений программного интерфейса API, от версии к версии

Android-платформы изменялись предустановленные приложения, добавлялась поддержка новых технологий и улучшалась производительность.

Описание ADT-плагина

В результате установки ADT-плагина в команде **New** меню **File** среды Eclipse появится раздел **Android** , содержащий следующие мастера:

Android Activity – создает класс, расширяющий класс android.app.Activity и представляющий экран приложения.

Android Application Project – обеспечивает создание проекта Android-приложения.

Android Icon Set – позволяет создать набор значков приложения:

Launcher Icons – значок, представляющий приложение.

Action Bar and Tab Icons (Android 3.0+) – значки элементов панели действий пользователя для платформы версии 3.0 и выше.

Notification Icons – значки уведомлений панели состояния.

Pre-Android 3.0 Tab Icons – значки элементов панели действий пользователя для платформы версии ниже 3.0.

Pre-Android 3.0 Menu Icons – значки меню для платформы версии ниже 3.0.

Android Object – создает различные компоненты, такие как Activity, Widget, Fragment, Receiver, Provider, Service и др.

Android Project from Existing Code – импорт проекта приложения.

Android Sample Project – при условии установки с помощью SDK Manager пакета примеров Samples for SDK, позволяет создать проект выбранного примера Android-приложения.

Android Test Project – для выбранного Android-проекта создает основу набора тестов на базе каркаса Android testing framework, являющегося расширением платформы тестирования JUnit.

Android XML File – обеспечивает создание таких ресурсов приложения как:

Layout – XML-описание GUI-интерфейса Activity-компонента.

Values – XML-файл, содержащий набор текстовых строк, стилей, различного рода значений, используемых приложением.

Drawable – XML-файл, формирующий отображаемую на экране графику.

Menu – XML-файл, определяющий меню приложения.

Color List – XML-файл, определяющий набор цветов для различных состояний GUI-компонента.

Property Animation – XML-файл, задающий анимацию свойств объекта.

Tween Animation – XML-файл, задающий анимацию View-компонента (вращение, исчезновение, перемещение и масштабирование).

AppWidgetProvider – XML-файл, содержащий метаданные для миниатюрного приложения App Widget, как правило размещаемого на главном экране Home Screen.

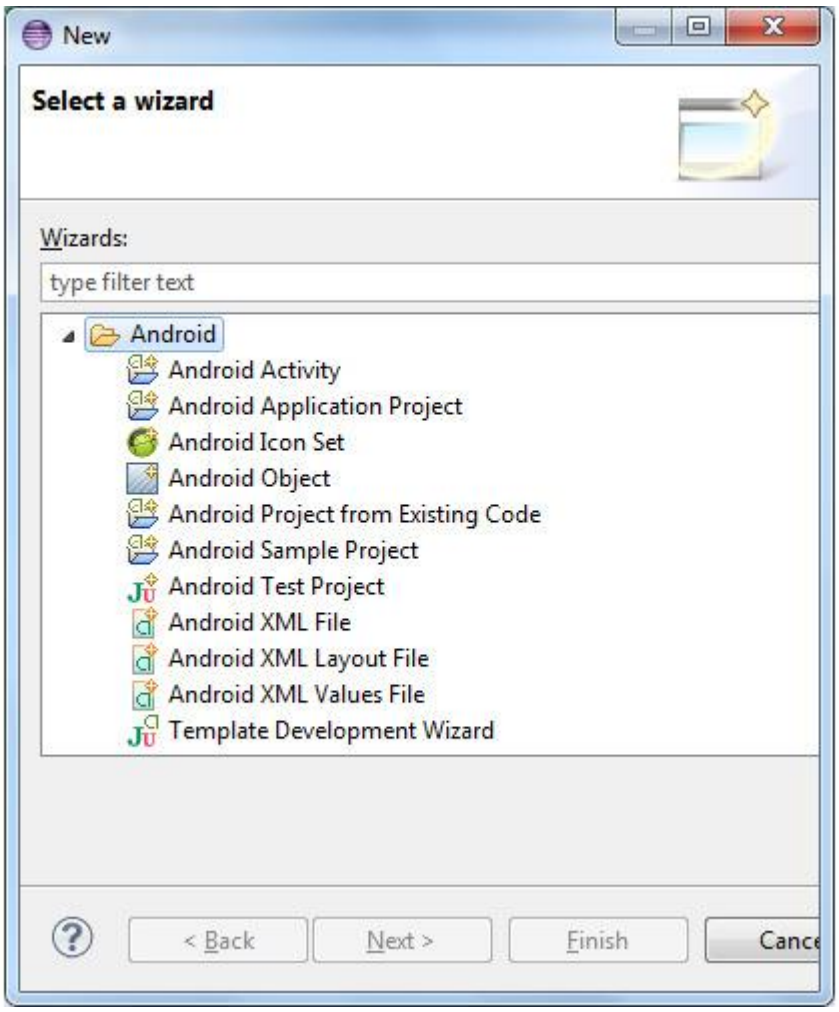
Preference – XML-описание GUI-интерфейса PreferenceActivity-операции, позволяющей пользователю персонифицировать приложение.

Searchable – XML-файл, определяющий настройки GUI-компонента поиска.

Android XML Layout File – аналог мастера **Android XML File | Layout** .

Android XML Values File – аналог мастера **Android XML File | Values** .

Template Development Wizard – генерирует код на основе файла шаблона template. xml.



После установки ADT-плагина, в контекстном меню окна **Package Explorer** появятся следующие команды:

Run As | Android Application – запускает Android-приложение в виртуальном мобильном устройстве, созданным с помощью AVD Manager.

Run As | Android JUnit Test – запускает набор тестов для Android-приложения с использованием виртуального мобильного устройства.

Android Tools | New Test Project – открывает мастер **Android Test Project** создания набора тестов для Android-приложения.

Android Tools | New Resource File – открывает мастер **Android XML File** создания ресурсов приложения.

Android Tools | Export Signed Application Package – открывает мастер **Export Android Application** экспорта пописанного цифровой подписью и готового к публикации Android-приложения.

Android Tools | Export Unsigned Application Package – экспортирует неподписанный для релиза APK-файл Android-приложения.

Android Tools | Display dex bytecode – в окне Eclipse-редактора отображает инструкции байткода, дизассемблированные из DEX-файла, который создается в процессе сборки приложения путем конвертации из Java класс-файлов для выполнения виртуальной машиной Dalvik среды выполнения Android.

Android Tools | Rename Application Package – переименовывает пакет приложения.

Android Tools | Add Support Library – запускает приложение SDK Manager для добавления в путь приложения библиотеки Android Support Package, предоставляющей дополнительный API, не являющийся частью API версии Android-платформы. Другой способ добавления библиотеки Android Support Package – установить библиотеку с помощью

раздела **Extras** приложения SDK Manager, создать папку libs в каталоге проекта, скопировать в нее библиотеку из папки extrasкаталога Android SDK и добавить библиотеку в путь приложения используя команду **Build Path | Configure Build Path** контекстного меню окна **Package Explorer** .

Android Tools | Fix Project Properties – в случае импорта готового Android-проекта гарантирует правильную его сборку, в частности добавляет в путь приложения необходимые библиотеки.

Android Tools | Run Lint: Check for Common Errors – сканирует Android-проект для поиска потенциальных багов с выводом сообщений о них в окно **Lint Warnings** .

Android Tools | Clear Lint Markers – очищает окно **Lint Warnings** .

Android Tools | Add Native Support – добавление поддержки Android NDK.

В меню **Windows** Workbench-окна появятся команды **Android SDK Manager** , **AVD Manager** и **Run Android Lint** , с помощью которых можно запустить приложения набора SDK Tools и сканирование Android-проекта для поиска потенциальных багов. Данные команды дублируются соответствующими кнопками панели инструментов Workbench-окна.

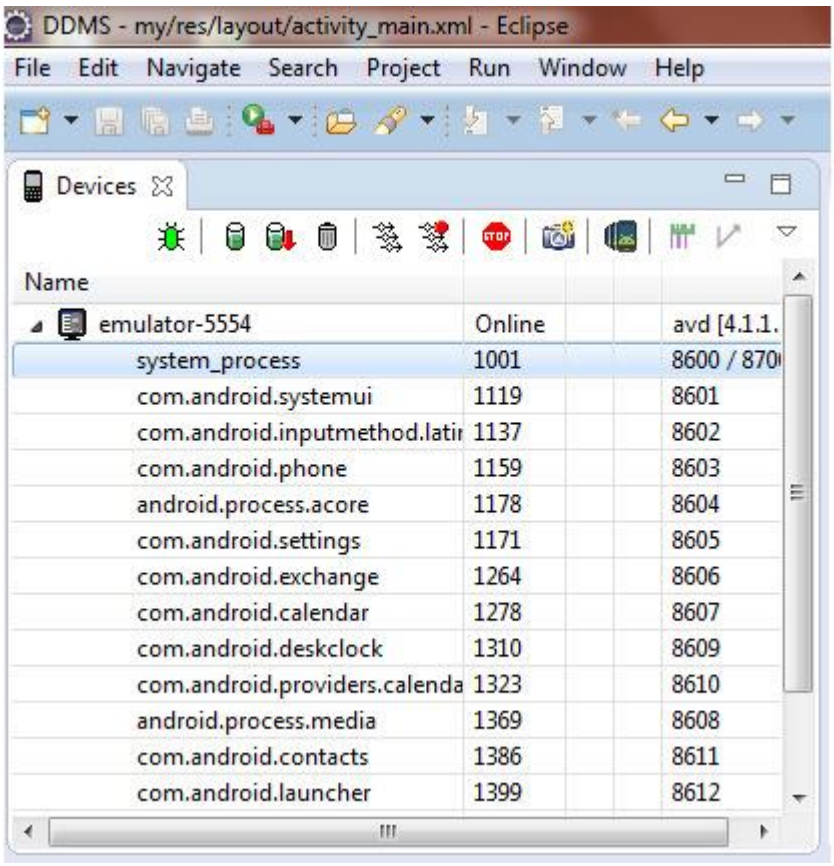
ADT-плагин добавляет в среду Eclipse перспективы **DDMS** , **Hierarchy View** , **Pixel Perfect** , **Tracer for OpenGL ES** и **XML** .

Перспектива DDMS

Перспектива **DDMS** запускает инструмент отладки Dalvik Debug Monitor Server набора SDK Tools и отображает его GUI-интерфейс в виде набора Eclipse-представлений, обеспечивая информацию о работе эмулятора или подсоединенного Android-устройства.

Перспектива **DDMS** содержит представления **Devices** , **Emulator Control** , **LogCat** , **Threads** , **Heap** , **Allocation Tracker** , **Network Statistics** , **System Information** и **File Explorer** .

Представление **Devices** отображает подключенные Android-устройства. Для каждого подключенного Android-устройства Devices-представление показывает все запущенные на нем процессы, каждый из которых работает в своем экземпляре виртуальной машины Dalvik. Каждый отображаемый процесс представляет инсталлированное и запущенное на Android-устройстве приложение, поэтому идентификация процесса производится по имени пакета приложения. Так как виртуальная машина Dalvik работает поверх ядра Linux, каждый процесс имеет свой Linux-идентификатор, отображаемый в окне **Devices** после имени пакета. Крайний правый столбец окна **Devices** показывает номер порта, который DDMS-инструмент назначает для подсоединения Eclipse-отладчика к экземпляру Dalvik-машины с использованием протокола JDWP (Java Debug Wire Protocol). По умолчанию Eclipse-отладчик подсоединяется к статическому порту 8700, на который перенаправляются трафики экземпляров Dalvik-машины от всех портов. DDMS-инструмент взаимодействует с подключенным Android-устройством с помощью инструмента Android Debug Bridge (adb), имеющего клиент-серверную архитектуру. DDMS-инструмент создает adb-клиента, который взаимодействует с adb-демоном (фоновый процесс, работающий в Android-устройстве) через adb-сервер.



Панель инструментов представления **Devices** содержит следующие кнопки:

(Debug the selected process, …) – подсоединяет процесс, представляющий Android-приложение с открытым в среде Eclipse проектом, к Eclipse-отладчику, для работы с которым используется перспектива **Debug** .

(Update Heap) – включает информацию об использовании динамической памяти для процесса.

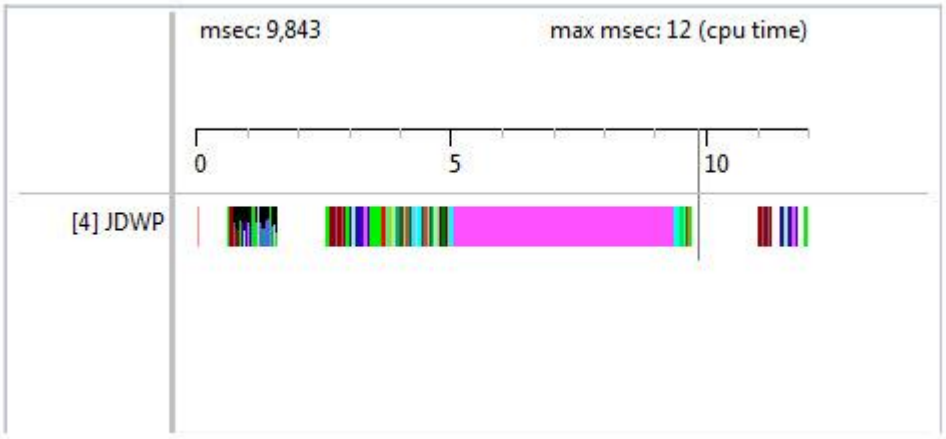
(Dump HPROF file) – создает снимок динамической памяти в виде HPROF-файла. В случае Android-устройств версии 2.1 и ранее для создания HPROF-файла требуется наличие SD-карты памяти, а также разрешения <uses-permission http://www.eclipse.org/mat/).rmission. WRITE_EXTERNAL_STORAGE»/> в файле манифеста AndroidManifest. xml Android-приложения. Анализ HPROF-файла можно выполнить с помощью Eclipse-плагина Memory Analyzer (MAT) (android:name="android.pe

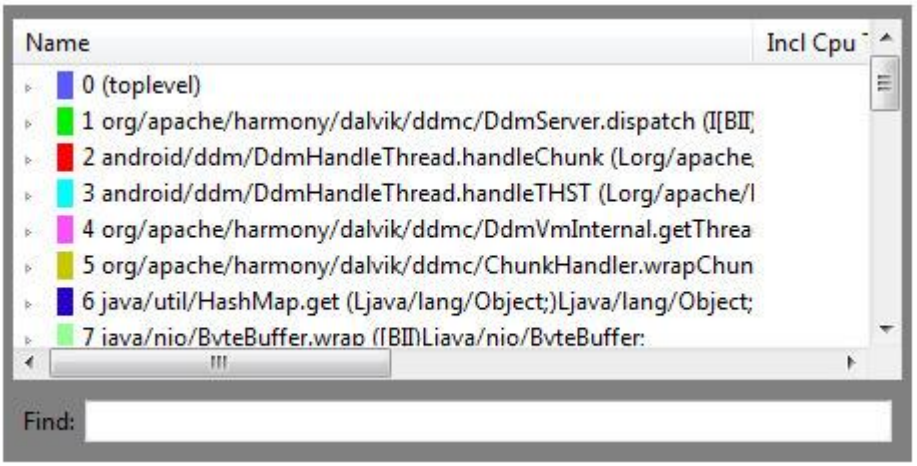
(Cause GC) – вызывает сборщика мусора, что влечет за собой сборку данных о динамической памяти.

(Update Threads) – включает информацию о запущенных потоках для выбранного

процесса.

(Start Method Profiling) и (Stop Method Profiling) – запускает и останавливает запись информации о выполнении методов приложения в Trace-файл, который после остановки записи открывается в Traceview-окне, отображающем журнал выполнения в виде двух панелей: Timeline Panel – с помощью цветовой гаммы и шкалы времени описывает старт и остановку выполнения метода в потоке, Profile Panel – показывает детали выполнения методов. В случае Android-устройств версии 2.1 и ранее для создания Trace-файла требуется наличие SD-карты памяти, а также разрешения <uses-permission android:name="android.permission. WRITE_EXTERNAL_STORAGE»/> в файле манифеста AndroidManifest. xml Android-приложения. За отображение окна Traceview отвечает инструмент traceview набора SDK Tools.





(Stop Process) – останавливает выбранный процесс.

(Screen Capture) – открывает окно **Device Screen Capture** , которое позволяет создавать скриншоты экрана Android-устройства.

(Dump View Hierarchy for UI Automator) – обеспечивает тестирование GUI-интерфейса приложения путем получения снимка экрана Tablet-устройства API 16 и выше, предоставляя визуальный интерфейс для проверки GUI-иерархии и просмотра свойств отдельных компонентов GUI-интерфейса. Работа команды обеспечивается инструментом uiautomatorviewer набора Android SDK.

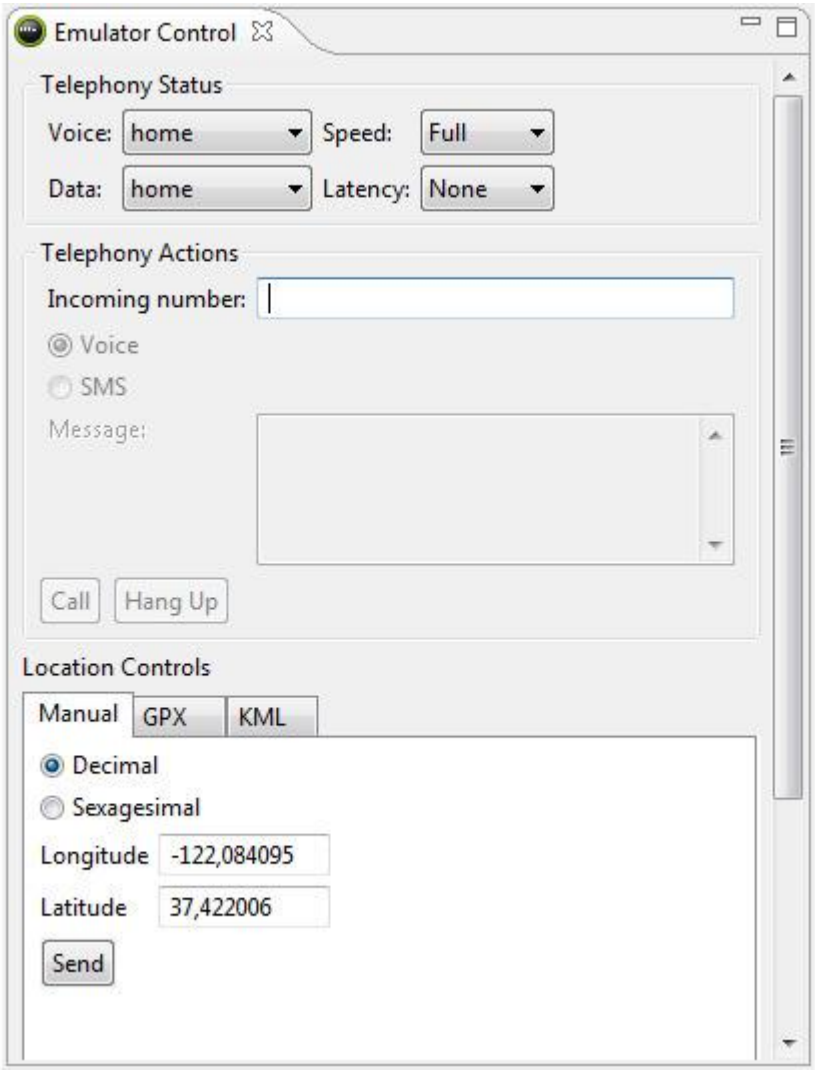
(Capture system wide trace using Android systrace) – для устройства Android 4.1 (API Level 16) помогает анализировать производительность приложения, формируя журнал

событий системы и приложения в виде HTML-файла.

(Start OpenGL Trace) – для устройства Android 4.1 (API Level 16) – помогает анализировать выполнение графических OpenGL ES команд.

Меню панели инструментов представления **Devices** , помимо вышеперечисленных команд, содержит команду **Reset adb** , обеспечивающую перезапуск adb-инструмента.

Представление **Emulator Control** дает возможность имитировать для экземпляра Android-эмулятора входящий звонок, SMS-сообщение и локализацию.



Списки **Voice:** и **Data:** раздела **Telephony Status** представления **Emulator Control** позволяют установить состояние GPRS-соединения:

unregistered – сеть отсутствует.

home – локальная сеть.

roaming – телефон в роуминге.

searching – поиск сети.

denied – только звонки экстренных служб.

Список **Speed:** раздела **Telephony Status** представления **Emulator Control** позволяет установить скорость передачи данных сети:

GSM – 14.4 килобит\uc1сек.

HSCSD – от 14.4 до 43.2 килобит\uc1сек.

GPRS – от 40.0 до 80.0 килобит\uc1сек.

EDGE – от 118.4 до 236.8 килобит\uc1сек.

UMTS – от 128.0 до 1920.0 килобит\uc1сек.

HSDPA – от 348.0 до 14400.0 килобит\uc1сек.

Full – без ограничений.

Список **Latency:** раздела **Telephony Status** представления **Emulator Control** позволяет имитировать уровень задержки сети:

GPRS – от 150 до 550 миллисекунд.

EDGE – от 80 до 400 миллисекунд.

UMTS – от 35 до 200 миллисекунд.

None – задержка отсутствует.

Раздел **Telephony Actions** представления **Emulator Control** дает возможность имитировать входящий звонок и SMS-сообщение.

Раздел **Location Controls** представления **Emulator Control** обеспечивает определение локализации Android-устройства вручную (вкладка **Manual**) или с помощью файлов GPS eXchange (вкладка **GPX**) и Keyhole Markup Language (вкладка **KML**).

Представление **LogCat** обеспечивает отображение всех системных сообщений от Android-устройства, в то время как представление **Console** показывает только сообщения, относящиеся к изменениям состояния Android-устройства и его приложений.

LogCat-окно отображает системные сообщения в таблице, содержащей столбцы Level (приоритет сообщения), Time (время создания сообщения), PID (Linux-идентификатор процесса), Application (имя пакета приложения), Tag (идентификатор системного компонента, от которого получено сообщение), Text (текст сообщения). Соответственно панель инструментов представления **LogCat** обеспечивает фильтрацию отображаемых сообщений по приоритету, тэгу, по идентификатору и имени пакета приложения.

Представление **Threads** показывает запущенные потоки для выбранного процесса. Для просмотра потоков необходимо в окне **Devices** выбрать процесс и нажать кнопку **Update Threads** панели инструментов окна **Devices** .

Threads-окно отображает информацию о потоках в виде двух таблиц. Верхняя таблица показывает все запущенные потоки для выбранного процесса и имеет следующие столбцы:

ID – Dalvik-идентификатор потока – нечетные числа, начиная с 3. Демоны помечаются «*».

TID – Linux-идентификатор потока.

Status – статус потока:

Wait – вызван метод Object. wait ().

Native – выполняет системный код.

Vmwait – ожидает Dalvik-ресурс.

Runnable – может быть запущен.

TimedWait – ожидает в течение определенного количества времени.

utime – общее время выполнения пользовательского кода (единица 10 мс.).

stime – общее время выполнения системного кода (единица 10 мс.).

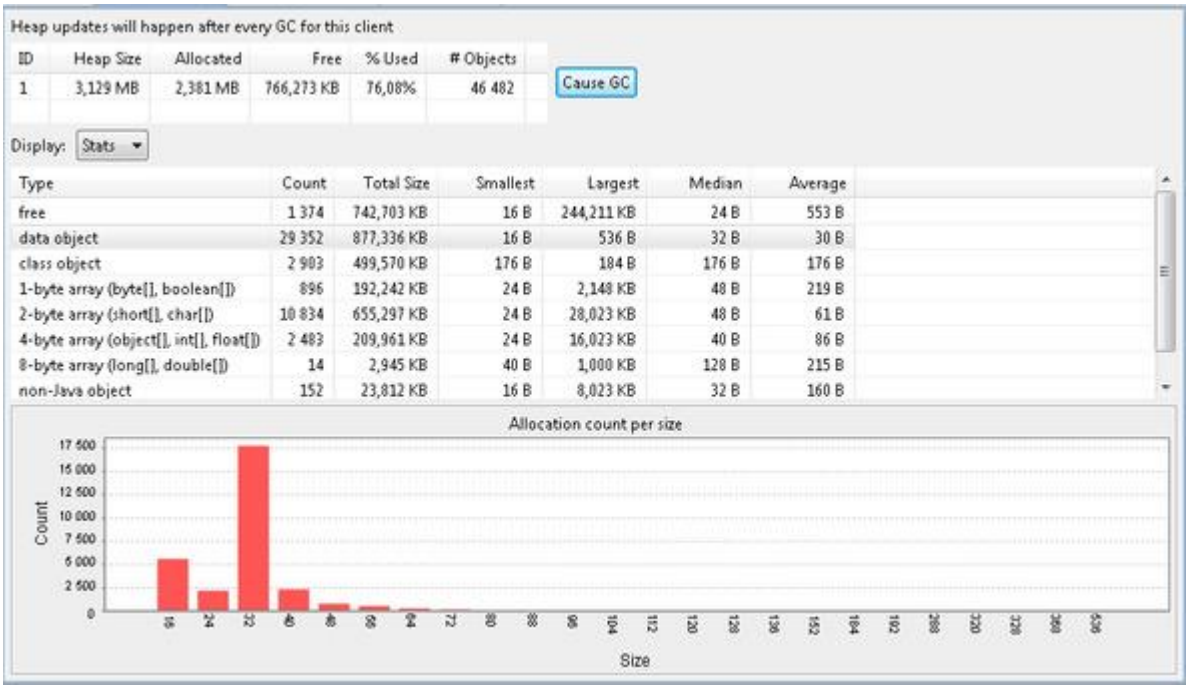
Name – имя потока.

Нижняя таблица для выбранного потока показывает выполняемый потоком код, указывая класс, метод, файл, строку и признак кода.

Представление **Heap** отображает информацию об использовании динамической памяти выбранным процессом. Для просмотра кучи процесса в Heap-окне необходимо в окне **Devices** выбрать процесс и нажать кнопку **Update Heap** , затем кнопку **Cause GC** панели инструментов окна **Devices** .

Представление **Heap** содержит три области. Самая верхняя область показывает таблицу структуры кучи процесса со столбцами ID (идентификатор кучи), Heap Size (общее количество памяти кучи), Allocated (количество занятой памяти кучи), Free (количество свободной памяти кучи), %Used (процент занятости кучи) и #Objects (количество объектов кучи), а также имеет кнопку **Cause GC** обновления информации о куче.

Далее расположена область с таблицей распределения объектов кучи по типам. Самая нижняя область отображает гистограмму распределения выбранного типа объектов по размерам занимаемой памяти.



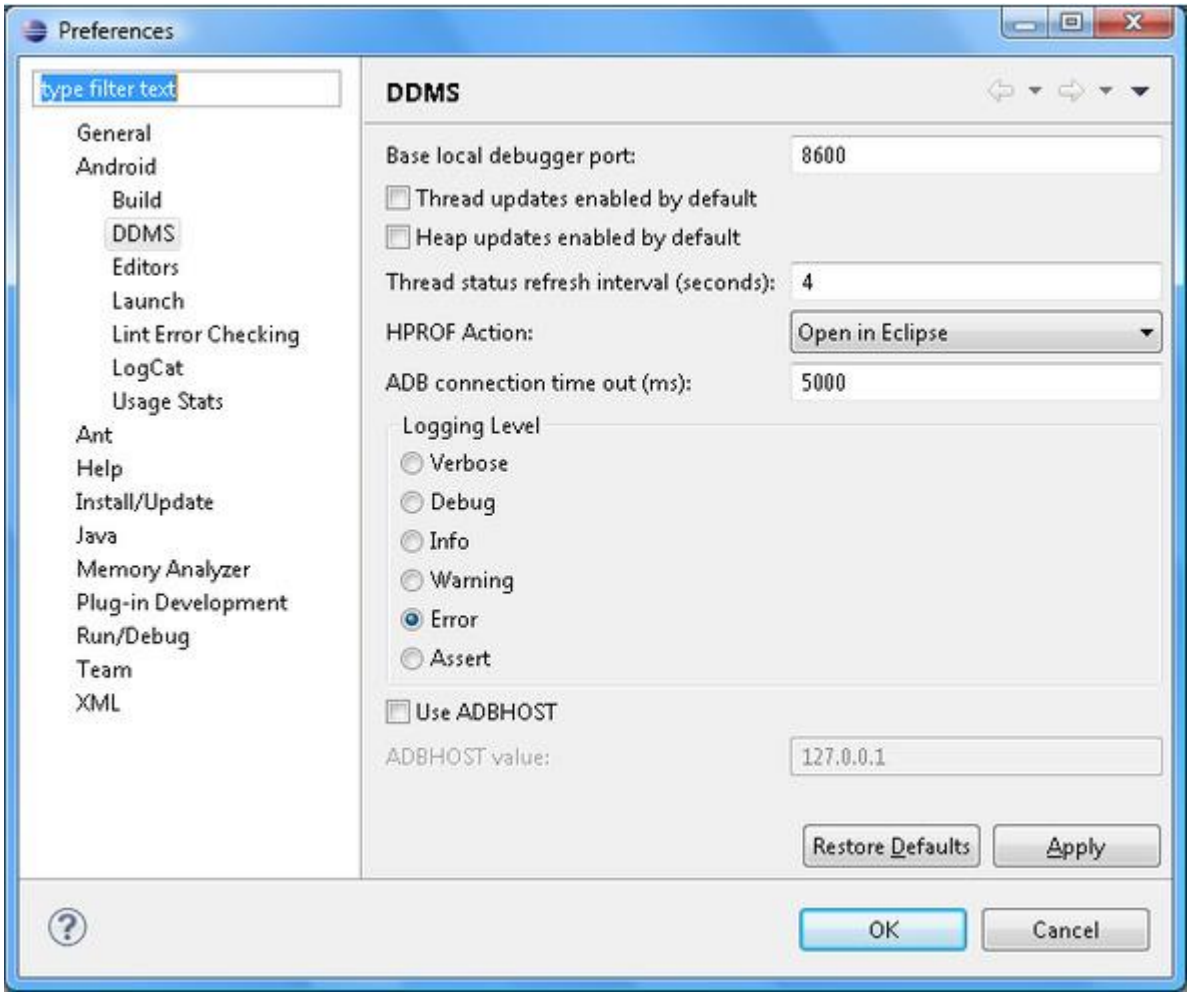
Представление **Allocation Tracker** позволяет в реальном времени отследить объекты, для которых выделяется память. Для начала просмотра журнала выделения памяти для объектов необходимо в окне **Devices** выбрать процесс и нажать кнопку **Start Tracking** в окне **Allocation Tracker** , затем кнопку **Get Allocations** . В результате верхняя область окна **Allocation Tracker** покажет список объектов, созданных с момента нажатия кнопки **Start Tracking** до момента нажатия кнопки **Get Allocations** , с указанием выделенной памяти, идентификатора потока, класса и метода, а нижняя область – более детальную информацию для выбранного объекта, с указанием класса, метода, файла, строки и признака кода.

Представление **Network Statistics** позволяет сформировать и проанализировать журнал передачи данных по сети.

Представление **System Information** отображает диаграммы использования системных ресурсов.

Представление **File Explorer** показывает файловую систему Android-устройства с возможностью экспорта и импорта файлов, удаления файлов и создания новых папок.

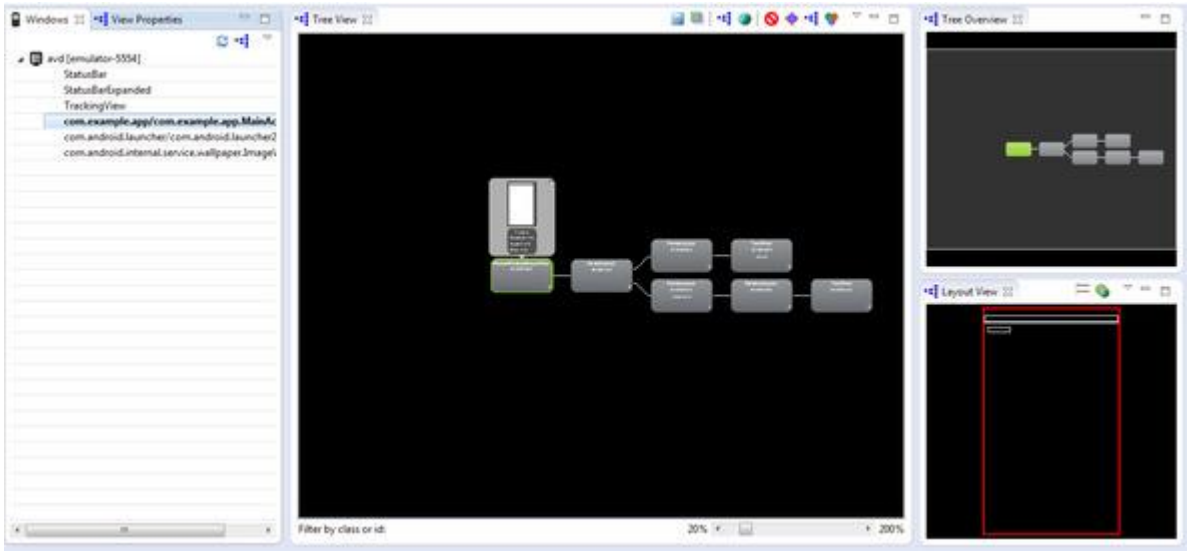
Общая настройка DDMS-инструмента осуществляется с помощью раздела **Android | DDMS** команды **Preferences** меню **Window** , где можно определить номер порта, с которого DDMS-инструмент начинает назначать порт для подсоединения Eclipse-отладчика к экземпляру Dalvik-машины по протоколу JDWP, обновление по умолчанию информации о куче и потоках с указанным интервалом, сохранение HPROF-файла или открытие его в среде Eclipse, время ожидания adb-инструмента, adb-хост для связи с Android-устройством по сети.



Перспективы Hierarchy View и Pixel Perfect

Перспективы **Hierarchy View** и **Pixel Perfect** запускают инструмент hierarchyviewer набора SDK Tools и отображают его GUI-интерфейс в виде набора Eclipse-представлений, помогая отладить и оптимизировать GUI-интерфейс Android-приложения.

Перспектива **Hierarchy View** содержит представления **Windows** , **View Properties** , **Tree View** , **Tree Overview** , **Layout View** .



Представление **Windows** отображает список подключенных Android-устройств, для каждого из которых показывает список Activity-объектов, включающий в себя системные объекты и объект приложения, GUI-интерфейс которых отображается на экране Android-устройства.

Представление **Tree View** показывает иерархию GUI-компонентов графического интерфейса выбранного Activity-объекта. Для просмотра диаграммы иерархии GUI-интерфейса приложения в представлении **Tree View** , в окне **Windows** необходимо выбрать Activity-объект приложения и нажать кнопку **Load the view hierarchy into the tree view** панели инструментов Windows-окна. В результате в окне **Tree View** отобразится иерархия View-объектов приложения.

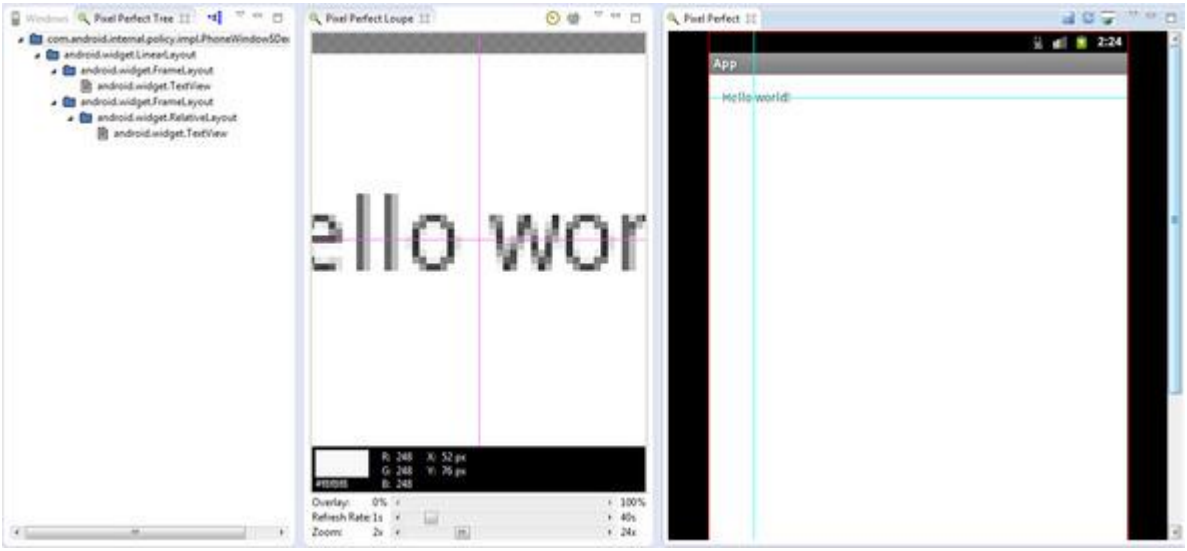
Диаграмму окна **Tree View** можно увеличивать с помощью нижнего ползунка, фильтровать, используя поле **Filter by class or id:** . Панель инструментов представления **Tree View** позволяет сохранить отображаемую диаграмму как PNG-изображение, как PSD-документ, открыть выбранный View-объект в отдельном окне и др.

Представление **Tree Overview** обеспечивает перемещение по диаграмме окна **Tree View** с помощью перетаскивания выделенного прямоугольника окна **Tree Overview** .

Окно **Layout View** является блочным представлением GUI-интерфейса. При выборе компонента диаграммы окна **Tree View** его расположение в GUI-интерфейсе подсвечивается красным цветом в окне **Layout View** . Также при выборе View-объекта диаграммы окна **Tree View** , выше его узла появляется небольшое окно с реальным изображением GUI-компонента и информацией о количестве View-объектов, представляющих компонент, и о времени отображения компонента в миллисекундах. При этом свойства выбранного View-объекта диаграммы отображаются в представлении **View Properties** .

Информация о времени визуализации компонентов GUI-интерфейса приложения помогает найти причину его медленной работы.

Перспектива **Pixel Perfect** содержит представления **Windows** , **Pixel Perfect Tree** , **Pixel Perfect Loup** и **Pixel Perfect** .



Представление **Windows** отображает список подключенных Android-устройств без детализации. При выборе устройства и нажатии кнопки **Inspect a screenshot in the pixel perfect view** панели инструментов окна **Windows** снимок экрана выбранного Android-устройства открывается в представлениях **Pixel Perfect Loup** и **Pixel Perfect** . В представлении **Pixel Perfect Tree** отображается дерево View-объектов GUI-интерфейса приложения, формирующего снимок экрана. При выборе View-объекта в окне **Pixel Perfect Tree** , его расположение обозначается красной рамкой в окне **Pixel Perfect** .

Представление **Pixel Perfect Loup** содержит перекрестье, которое дает информацию о пикселе, находящемся в центре пересечения, включающую в себя HTML-код цвета пикселя, его RGB-значение и координаты. Изображение окна **Pixel Perfect Loup** можно перемещать мышкой относительно перекрестья. Слайдер **Zoom** позволяет регулировать увеличение снимка экрана Android-устройства.

Представление **Pixel Perfect** также содержит перекрестье, расположение которого относительно снимка экрана совпадает с расположением перекрестья окна **Pixel Perfect Loup** и наоборот. Перекрестье окна **Pixel Perfect** можно передвигать мышкой, а панель инструментов окна **Pixel Perfect** дает возможность сохранить снимок экрана как PNG-изображение, а также загрузить поверх снимка экрана другое изображение, представляющее макет GUI-интерфейса приложения, при этом прозрачность загруженного изображения можно регулировать с помощью слайдера **Overlay:** окна **Pixel Perfect Loup** .

Возможность загрузки изображений поверх снимка экрана Android-устройства помогает в работе над дизайном GUI-интерфейса разрабатываемого Android-приложения.

Wizard мастера ADT плагина

Мастер Android Project

Для создания Android-приложения откроем среду Eclipse с инсталлированным ADT-плагином и в меню **File** выберем команду **New | Other | Android | Android Application Project** и нажмем кнопку **Next** .

Введем имя приложения, отображаемое в устройстве, имя проекта, имя пакета. Выберем минимальную версию SDK, предпочтительную версию SDK, версию SDK относительно которой приложение будет компилироваться, тему приложения и нажмем кнопку **Next** . Оставим отмеченными флажки **Create custom launcher icon** и **Create activity** и нажмем кнопку **Next** . Определим значок приложения и нажмем кнопку **Next** . Выберем создаваемый Activity-компонент и нажмем кнопку **Next** :

Blank Activity – экран с надписью «Hello world!».

Blank Activity with Fragment – экран с фрагментом с надписью «Hello world!».

Empty Activity – все равно экран с надписью «Hello world!».

Fullscreen Activity – экран, нажатие на который вызывает переключение между обычным и полноэкранным режимами.

Master/Detail Flow – экран с боковой панелью меню.

Navigation Drawer Activity – экран с двумя фрагментами, панелью навигации и контентом.

Tabbed Activity – экран с вкладками и типом навигации: с помощью жеста Swipe Views (ViewPager), с помощью панели закладок Action Bar Tabs (with ViewPager), с помощью выпадающего списка Action Bar Spinner.

Определим имя Activity-компонента, имя компоновочного файла res/layout/activity_main. xml и нажмем кнопку **Finish** – в результате будет сгенерирована основа проекта Android-приложения.

Модель программирования Android-приложений основывается не на конструкции с главным классом приложения, имеющим точку входа – статический метод main (), а является компонентной моделью. Android-приложение может состоять из одного или нескольких компонентов, объявленных в файле манифеста приложения AndroidManifest. xml и относящихся к четырем типам:

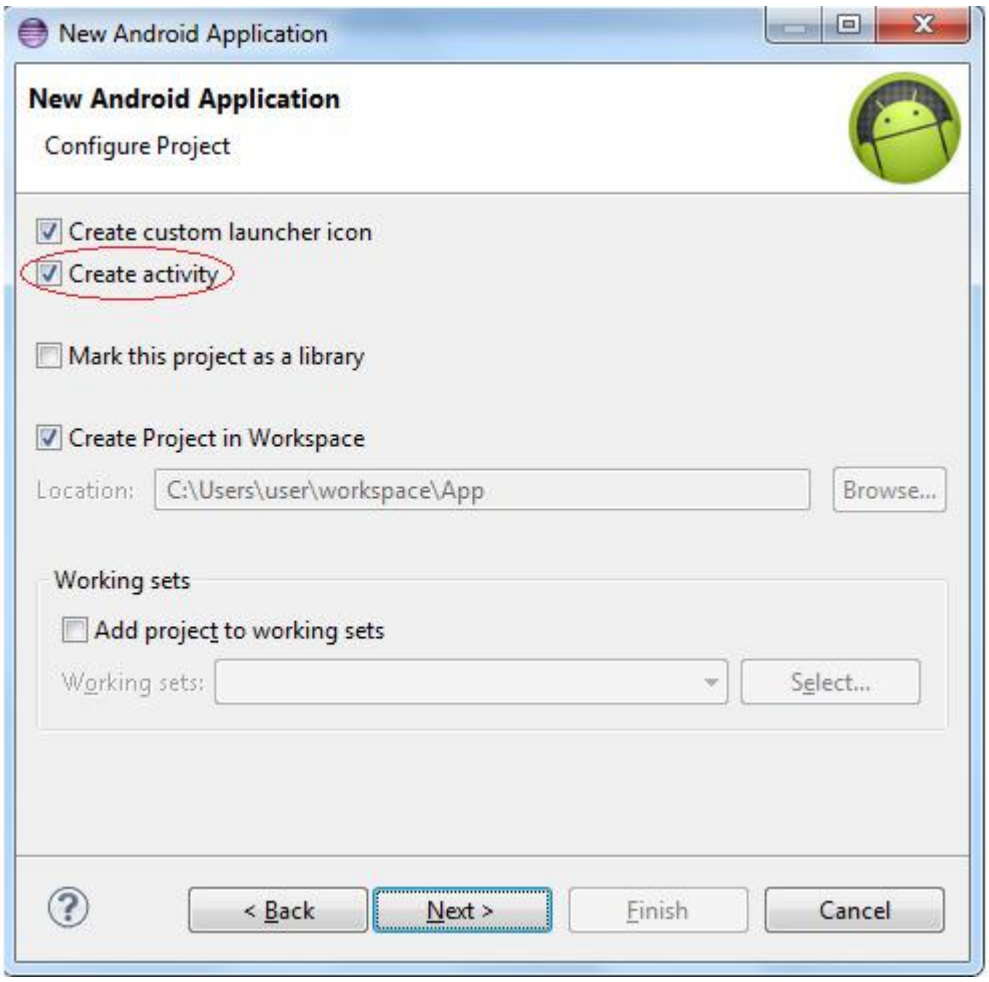
Activity – расширение класса android.app.Activity, обеспечивающее создание одного окна на экране Android-устройства с формированием в нем GUI-интерфейса.

Service – расширение класса android.app.Service, обеспечивающее выполнение операций без предоставления GUI-интерфейса.

BroadcastReceiver – расширение класса android.content.BroadcastReceiver, отвечающее за прослушивание широковещательных сообщений с запуском других компонентов Android-приложения или выводом уведомлений пользователю в строку статуса.

ContentProvider – расширение класса android.content.ContentProvider, обеспечивающее хранение и извлечение общих данных.

Существующая версия ADT-плагина при создании Android-проекта предлагает формирование основы только Activity-компонента.



Основа самого простого Android-проекта, сгенерированная средой Eclipse, состоит из следующих узлов окна **Package Explorer** :

src – содержит пакет класса, расширяющего класс android.app.Activity.

gen – содержит R-класс, автоматически генерируемый инструментом aapt набора SDK Platform-tools из существующих ресурсов проекта для программного к ним доступа, а также класс BuildConfig, содержащий константу DEBUG, которая со значением true определяет запуск приложения в режиме отладки. При экспорте подписанного приложения значение константы DEBUG автоматически становится false.

Android x.x – библиотека Android-платформы, на основе которой создается приложение.

Android Private Libraries – дополнительная библиотека android-support, обеспечивающая обратную совместимость с предыдущими версиями Android API.

assets – каталог предназначен для хранения данных приложения, доступ к которым осуществляется с помощью класса android.content.res.AssetManager. Отличие данного каталога от каталога res заключается в том, что он не должен иметь строго преопределенной структуры, которая для каталога res обеспечивает автоматическую генерацию R-класса.

bin – каталог сборки приложения.

libs – содержит JAR-файл библиотеки android-support.

res – содержит ресурсы приложения, доступ к которым осуществляется с помощью R-класса, и имеет строго предопределенную структуру:

animator – XML-файлы для создания объектов анимации.

color – XML-файлы, определяющие цветовую гамму View-объектов.

drawable – PNG, JPEG, GIF, 9-PNG и XML-файлы, формирующие графику.

layout – XML-файлы для формирования структуры GUI-интерфейса Activity-объектов.

menu – XML-файлы, описывающие меню приложения.

raw – каталог предназначен для хранения таких данных приложения как файлов в формате MP3 или Ogg.

values – XML-файлы для хранения строк, стилей, чисел, размеров и др., используемых приложением, в виде пар имя-значение.

xml – различные конфигурационные и ресурсные XML-файлы.

AndroidManifest. xml – файл манифеста приложения, определяющий запуск Android-приложения средой выполнения Android и описывающий Android-компоненты приложения, права пользователя, минимальный уровень API Android-платформы, необходимый для запуска приложения, требуемые опции Android-устройства и др.

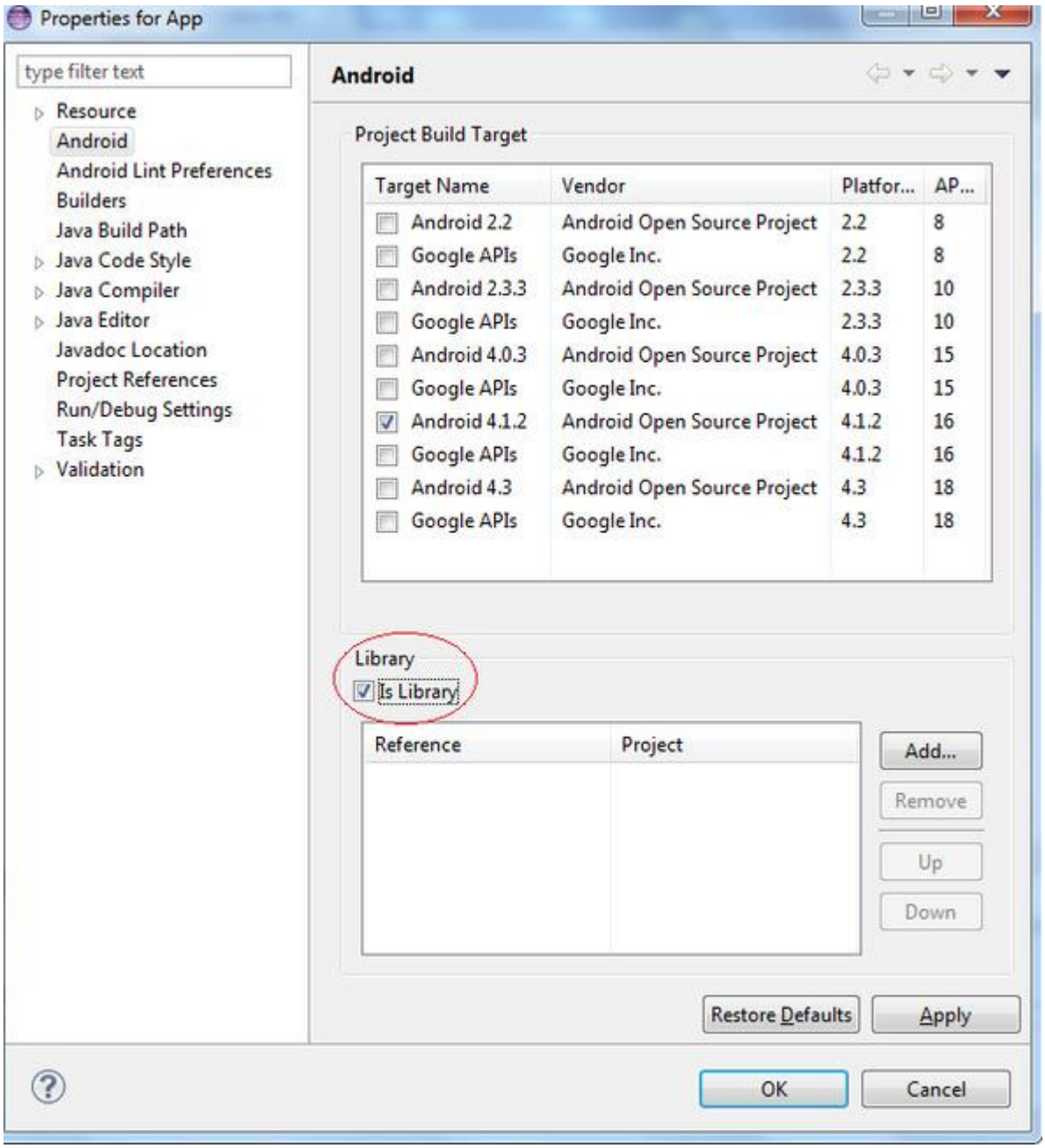
ic_launcher-web.png – значок приложения для магазина Google Play Store.

proguard.cfg – файл инструмента proguard набора SDK Tools, обеспечивающего сокращение, оптимизацию и обфускацию кода.

project.properties – содержит установки проекта.

Созданный Android-проект можно перевести в статус библиотеки, предоставляющей исходный код и ресурсы для других Android-проектов. При этом Android-библиотека не может содержать ресурсы в каталоге assets и версия Android-платформы библиотеки должна быть меньше или равна версии Android-платформы проекта, использующего библиотеку.

Для создания Android-библиотеки нужно в окне **Package Explorer** нажать правой кнопкой мышки на узле Android-проекта и в контекстном меню выбрать команду **Properties** . Далее в разделе **Android** отметить флажок **Is Library** и нажать кнопку **OK** .



Для использования созданной Android-библиотеки другим Android-проектом необходимо в окне **Package Explorer** нажать правой кнопкой мышки на узле Android-проекта и в контекстном меню выбрать команду **Properties** . Далее в разделе **Android** нажать кнопку **Add** и выбрать Android-библиотеку.

В результате в окне **Package Explorer** в Android-проект добавится узел **Library Projects** , содержащий временный JAR-файл Android-библиотеки, код и ресурсы которой можно использовать в проекте.

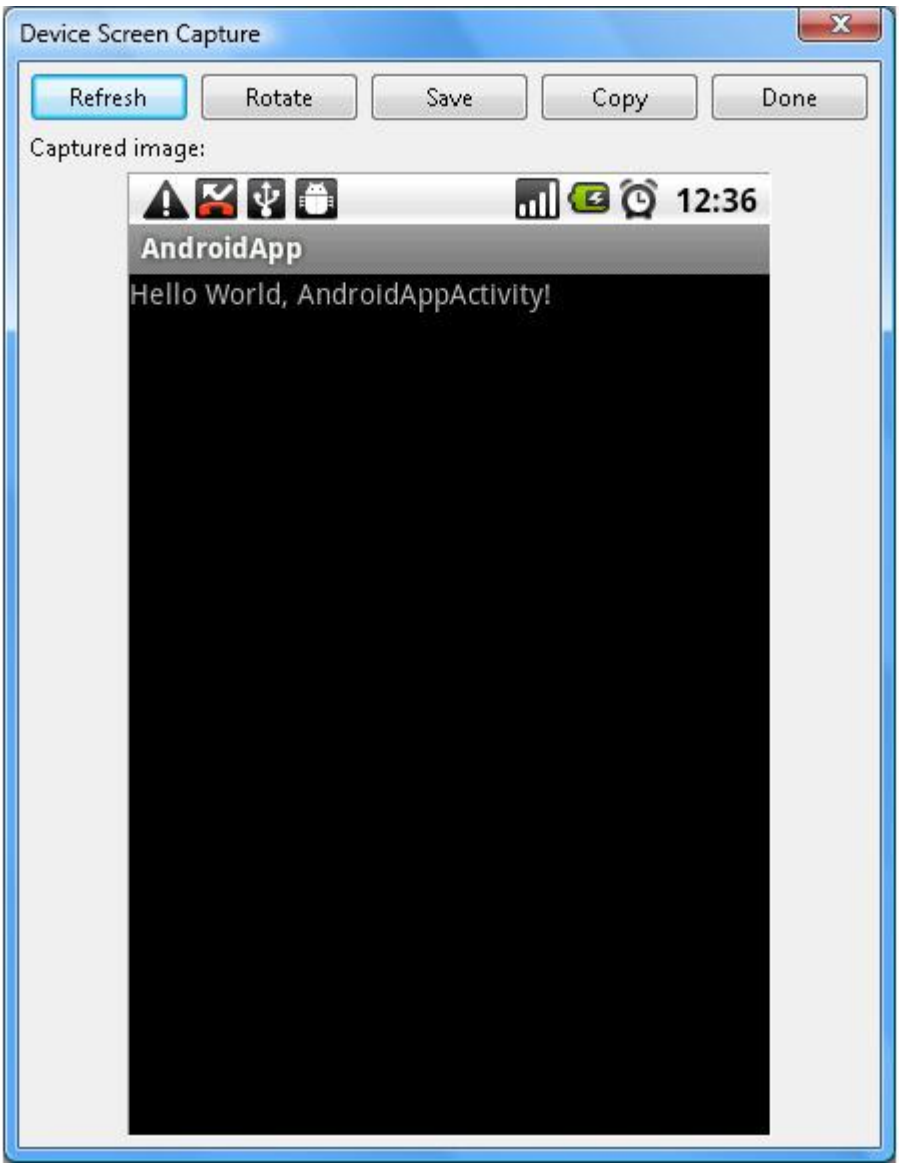
Запуск Android-приложения из среды Eclipse

Перед тем как воспользоваться командой **Run As | Android Application** контекстного меню окна **Package Explorer** для тестирования Android-приложения в реальной среде выполнения, необходимо либо подсоединить к компьютеру реальное Android-устройство, либо создать экземпляр Android-эмулятора.

Для тестирования Android-приложения в реальном Android-устройстве нужно зайти в настройки устройства и открыть раздел **Приложения** . В разделе **Приложения** отметить

флажок **Неизвестные источники** , затем открыть раздел **Разработка** и отметить флажок **Отладка USB** . После чего инсталлировать драйвер устройства на компьютер и подсоединить устройство к компьютеру. В результате среда Eclipse произведет опознание устройства, которое отобразится в окне **Devices** (команда **Window | Show View | Android | Devices**).

Для запуска Android-приложения выберем команду **Run As | Run Configurations** контекстного меню окна **Package Explorer** и в разделе приложения во вкладке **Target** отметим флажок **Always prompt to pick device** . Нажмем кнопку **Run** , в окне **Android Device Chooser** выберем устройство и нажмем кнопку **OK** . В результате Android-приложение будет инсталлировано и запущено в реальном Android-устройстве. Нажав кнопку **Screen Capture** панели инструментов окна **Devices** можно сделать снимок экрана реального Android-устройства.



Для тестирования Android-приложения в Android-эмуляторе нужно запустить приложение AVD Manager и для создания виртуального Android-устройства во вкладке **Android Virtual Devices** нажать кнопку **Create** :

В поле **AVD Name:** ввести имя устройства.

В списке **Device:** выбрать тип устройства.

В списке **Target:** выбрать версию Android-платформы устройства.

В списке **CPU/ABI:** выбрать тип процессора.

В списке **Skin:** выбрать оболочку эмулятора – установка аппаратной клавиатуры и отображение кнопок устройства в оболочке Android-эмулятора.

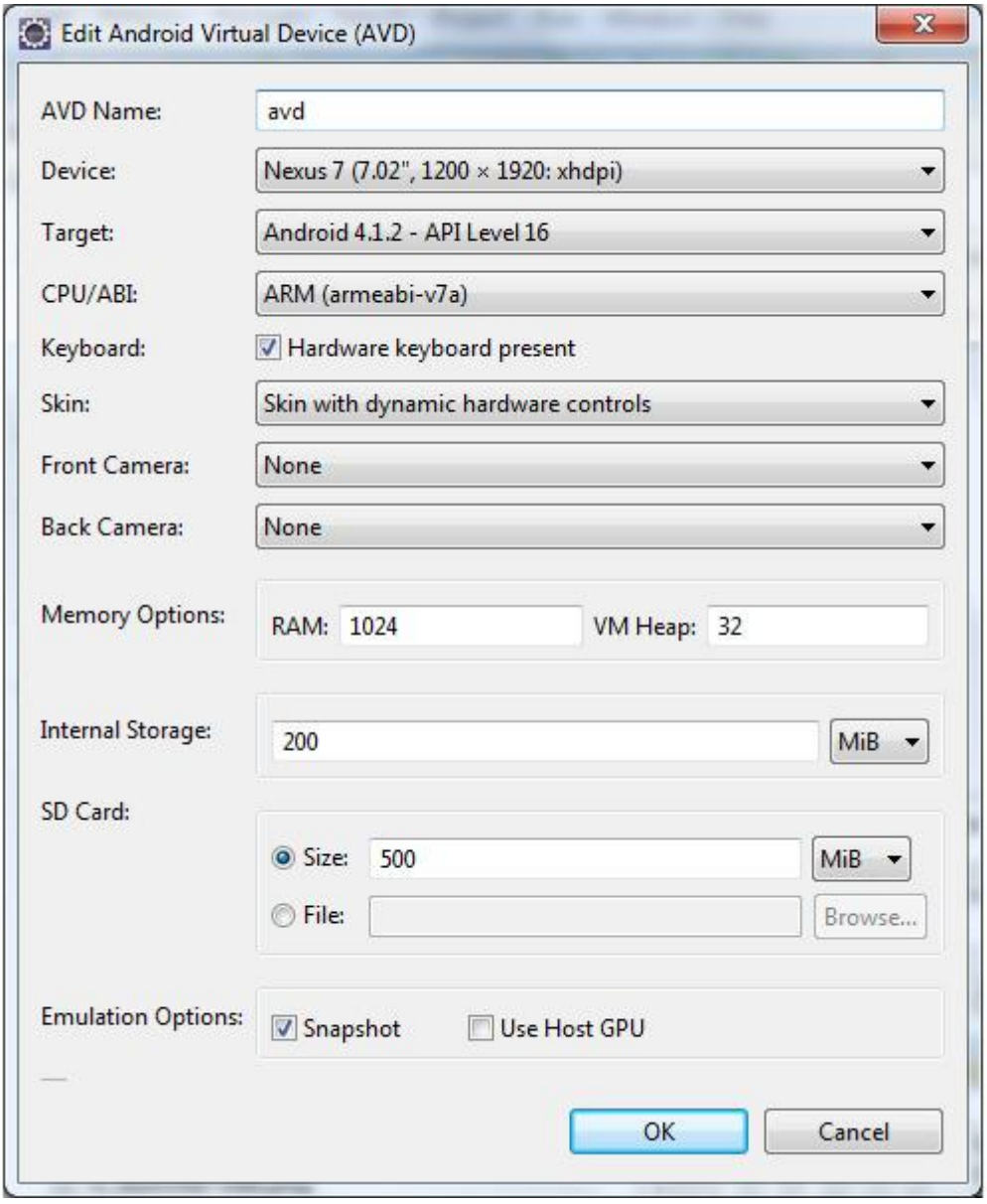
В списках **Front Camera:** и **Back Camera:** определить камеры устройства.

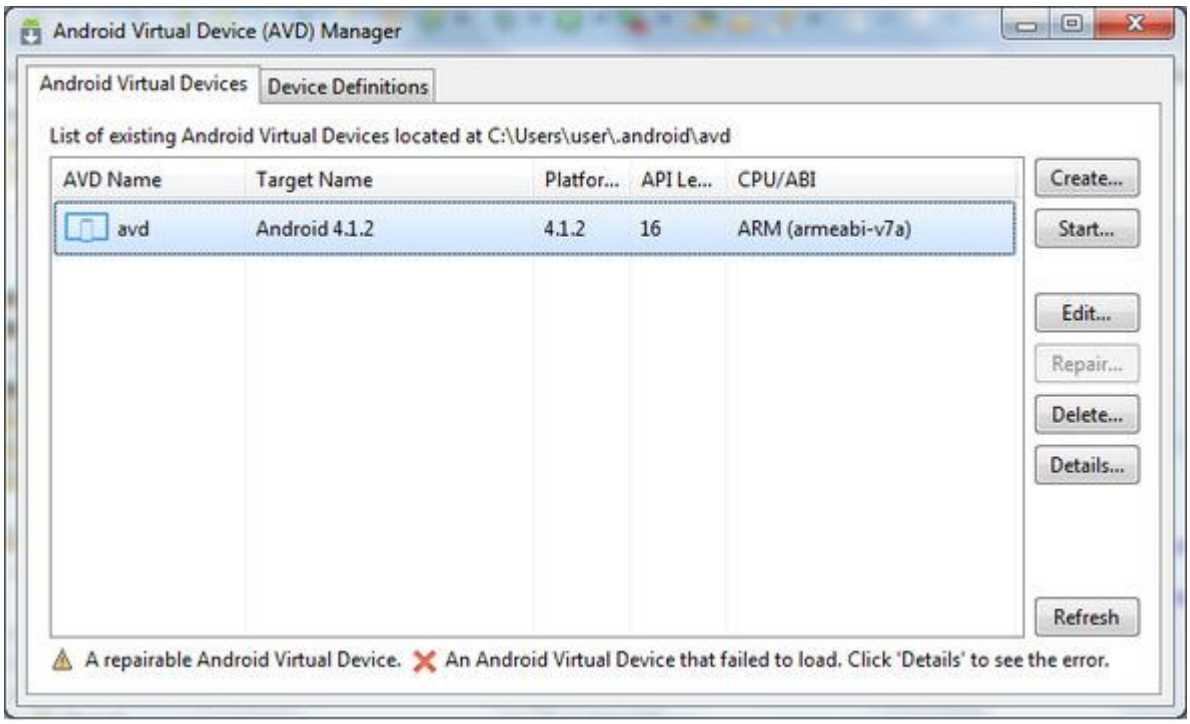
В разделе **Memory Options:** определить размер оперативной памяти и максимальный размер кучи, выделяемый для работы одного Android-приложения.

В разделах **Internal Storage:** и **SD Card:** определить размер внутренней памяти и тип/размер карты памяти устройства. Поле **File:** раздела **SD Card:** предназначено для определения образа карты памяти, созданного с использованием инструмента mksdcard набора SDK Tools.

В разделе **Emulation Options:** с помощью флажка **Snapshot** определить ускорение повторного запуска виртуального устройства, так как его состояние будет сохраняться при закрытии, с помощью флажка **Use Host CPU** определить ускорение работы эмулятора, так как он будет использовать CPU компьютера.

И нажать кнопку **OK** . В результате в окне **Android Virtual Device Manager** появится созданное виртуальное устройство, которое нужно запустить кнопкой **Start** .





Кнопка **Create Device** вкладки **Device Definitions** позволяет создать новое устройство списка **Device** .

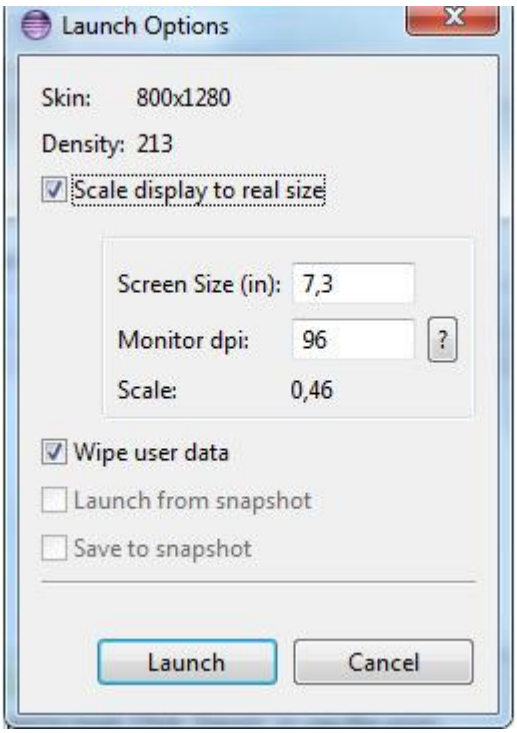
При запуске кнопкой **Start** виртуального устройства появляется окно **Launch Options** определения опций запуска, в котором:

Флажки **Launch from snapshot** и **Save to snapshot** работают в случае отмеченного флажка **Snapshot** раздела **Emulation Options:** и определяют загрузку и сохранение состояния виртуального устройства при закрытии.

Флажок **Scale display to real size** определяет масштабирование виртуального устройства путем установки диагонали и плотности экрана.

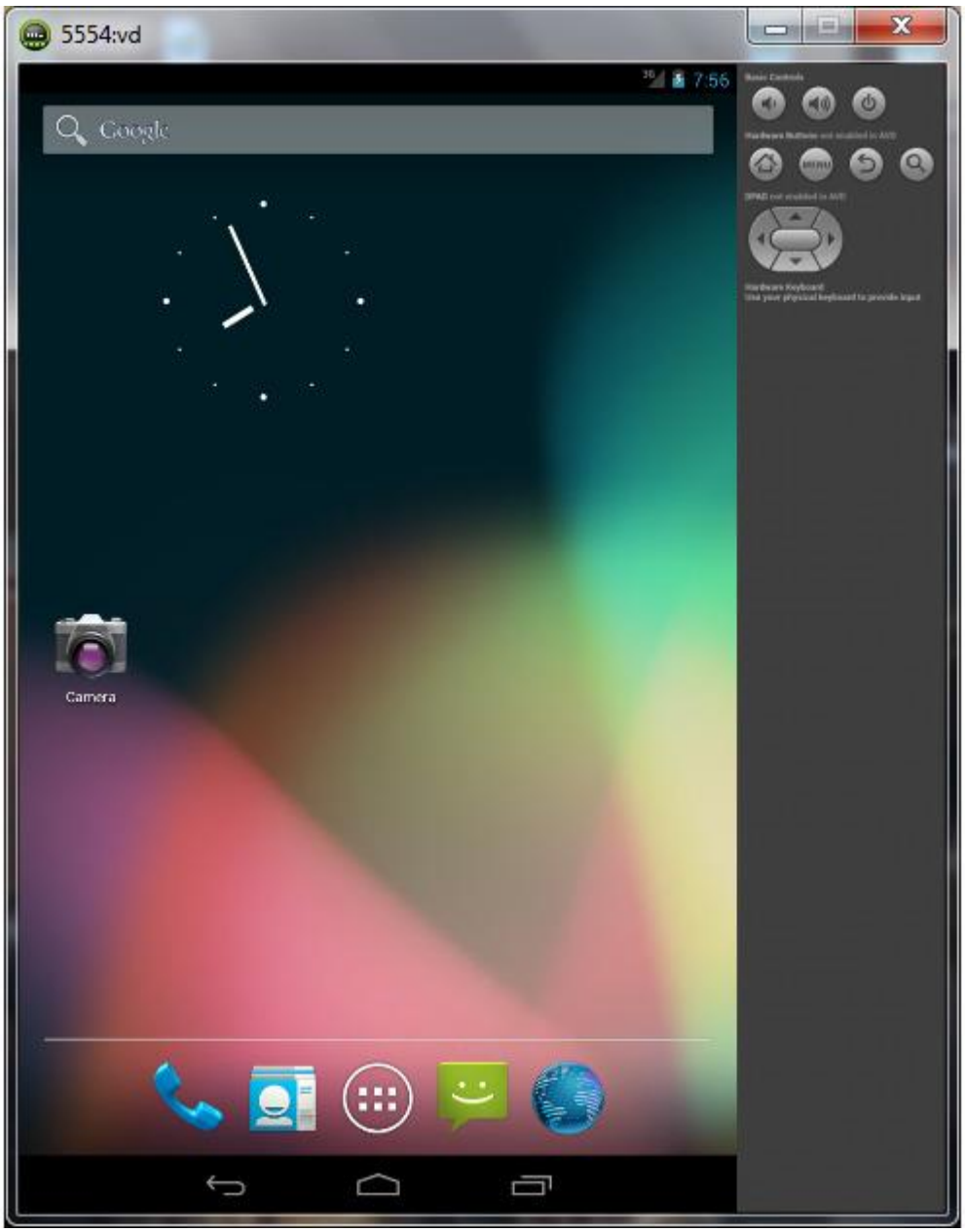
Флажок **Wipe user data** позволяет при запуске виртуального устройства стереть его сохраненное состояние, при этом будет отключена опция **Launch from snapshot** .

Виртуальное устройство окончательно запускается кнопкой **Launch** окна **Launch Options** .



По умолчанию приложение AVD Manager создает конфигурацию виртуального устройства в каталоге [user]androidфайловой системы компьютера. Если в пути данного каталога будут присутствовать русские буквы, виртуальное устройство не запустится. Кроме того, запуск виртуального устройства занимает значительное время, поэтому рекомендуется как можно реже его закрывать.

После запуска виртуальное устройство отобразится на экране компьютера.



Для запуска Android-приложения в виртуальном Android-устройстве воспользуемся командой **Run As | Android Application** контекстного меню окна **Package Explorer** .

В результате запуска Android-приложения каталог bin Android-проекта заполнится файлами и папками:

Папка classes – откомпилированные Java класс-файлы, включая классы приложения и R-классы.

Папка dexedLibs – конвертированные в DEX-формат виртуальной машины Dalvik дополнительные библиотеки.

Папка res – ресурсы приложения.

Файл манифеста AndroidManifest. xml.

Бинарный файл resources.arsc строк приложения.

Файл classes.dex – конвертированные в DEX-формат виртуальной машины Dalvik Java класс-файлы.

APK-файл – ZIP-архив Android-приложения для инсталляции в отладочном режиме, содержащий папку META-INF с Java-манифестом MANIFEST. MF и сертификатами, папку res с ресурсами, файл манифеста AndroidManifest. xml, файл resources.arsc и файл classes.dex.

Подготовка к публикации Android-приложения

Среда выполнения Android при инсталляции приложений требует, чтобы все Android-приложения были подписаны цифровой подписью с помощью сертификата, закрытый ключ которого имеется в распоряжении разработчиков приложений. Однако в отладочном режиме инструменты сборки приложения набора Android SDK автоматически подписывают приложение специальным отладочным ключом, который генерируется и по умолчанию хранится в файле debug.keystore каталога [user]android. По умолчанию отладочный сертификат имеет срок действия 365 дней и по истечении этого периода необходимо удалить файл debug.keystore для повторной автоматической генерации сертификата.

Для того чтобы подготовить Android-приложение к реальной инсталляции в Android-устройстве, так как среда выполнения Android не позволит инсталлировать приложение, подписанное отладочным ключом, можно воспользоваться командой **Android Tools | Export Signed Application Package** контекстного меню окна **Package Explorer** – в результате откроется мастер **Export Android Application** , в поле **Project:** которого будет отображено имя Android-проекта и в окне которого надо нажать кнопку **Next** .

Далее необходимо создать хранилище своего закрытого ключа, которым будут подписываться все версии данного Android-приложения. Важно использовать один постоянный ключ для одного Android-приложения, так как Android-система позволит обновление приложения только в том случае, если сертификаты старой и новой версии будут совпадать – иначе придется изменять имя пакета приложения, и новая версия будет уже устанавливаться как новое приложение. Кроме того, если приложение имеет модульную структуру и все модули подписаны одним сертификатом, тогда все модули будут запускаться в одном процессе, и смогут обновляться независимо друг от друга.

Для создания хранилища закрытого ключа в окне **Export Android Application** выберем переключатель **Create new keystore** , в поле **Location:** введем путь файла хранилища, в полях **Password:** и **Confirm:** введем пароль хранилища и нажмем кнопку **Next** .

Примечание

Для поля **Location:** необязательно использовать кнопку **Browse** – можно вручную ввести путь несуществующего файла, например, C:_keystore, где файл my_keystore будет сгенерирован.

В поле **Alias:** введем имя ключа, в полях **Password:** и **Confirm:** введем пароль ключа, в поле **Validity (years):** введем срок действия сертификата более 25 лет, в поле **First and Last Name:** введем имя разработчика и нажмем кнопку **Next** , в поле **Destination APK file:** введем путь APK-файла Android-приложения и нажмем кнопку **Finish** .

В результате будет создан подписанный и готовый к публикации файл приложения, обработанный при этом оптимизирующим инструментом zipalign набора SDK Tools.

Activity-компонент

Сгенерированная мастером **Android Project** основа Android-проекта Blank Activity в узле **src** окна **Package Explorer** содержит файл исходного кода Activity-компонента, в котором его класс, расширяющий класс android.app.Activity, переопределяет метод onCreate ().

Данный метод является одним из методов обратного вызова Activity-компонента, которые среда выполнения Android вызывает при переходе Activity-компонента между различными состояниями его жизненного цикла. Переопределение метода onCreate () является важным, так как он вызывается при запуске Activity-компонента и предназначен для инициализации GUI-интерфейса.

Помимо метода onCreate () класс android.app.Activity предоставляет следующие методы обратного вызова для их переопределения:

onRestart – метод жизненного цикла, вызывается после того как Activity-компонент был остановлен, перед вызовом метода onStart.

onStart – метод жизненного цикла, вызывается, когда Activity-компонент становится видимым.

onResume – метод жизненного цикла, вызывается, когда Activity-компонент помещается на передний план для взаимодействия с пользователем.

onPause – метод жизненного цикла, вызывается, когда Activity-компонент помещается на задний план. После данного метода может вызываться метод onResume, если Activity-компонент помещается снова на передний план, или метод onStop, если Activity-компонент становится невидимым.

onStop – метод жизненного цикла, вызывается, когда Activity-компонент становится невидимым. После данного метода может вызываться метод onRestart или метод onDestroy.

onDestroy – метод жизненного цикла, вызывается перед уничтожением Activity-компонента программным способом методом finish класса android.app.Activity или Android-системой для освобождения ресурсов.

onActionModeFinished – вызывается при окончании работы режима контекстного меню ActionMode.

onActionModeStarted – вызывается при запуске режима контекстного меню ActionMode для Activity-компонента.

onActivityResult – при запуске другого Activity-компонента методом startActivityForResult вызывается после закрытия запущенного Activity-компонента для обработки возвращаемых им результатов.

onAttachFragment – вызывается при присоединении объекта Fragment к объекту Activity между вызовами методов жизненного цикла Fragment. onAttach и Fragment. onCreate.

onAttachedToWindow – вызывается при присоединении окна Activity-компонента к Window-менеджеру, метод может быть использован вместо метода onCreate.

onBackPressed – вызывается при нажатии пользователем клавиши Back.

onConfigurationChanged – вызывается при изменении конфигурации устройства во время работы Activity-компонента, при этом информацию о новой конфигурации предоставляет объект android.content.res.Configuration.

onContentChanged – вызывается при изменении GUI-интерфейса Activity-компонента при вызове метода setContentView.

onContextItemSelected – вызывается при выборе элемента контекстного меню.

onContextMenuClosed – вызывается при закрытии контекстного меню.

onCreateContextMenu – вызывается при создании контекстного меню – меню, которое открывается при долгом нажатии на GUI-элементе.

onCreateDescription – вызывается перед вызовом метода onPause.

onCreateNavigateUpTaskStack – вызывается при создании стека задач.

onCreateOptionsMenu – вызывается при создании меню опций – меню, которое связано с Activity-компонентом.

onCreatePanelMenu – вызывается для инициализации содержимого меню (меню опций или контекстного меню).

onCreatePanelView – вызывается при создании панели меню.

onCreateThumbnail – вызывается перед вызовом метода onPause и позволяет определить для Activity-компонента значок, а не скриншот.

onCreateView – вызывается для создания фрагментом GUI-интерфейса.

onDetachedFromWindow – вызывается при отсоединении окна Activity-компонента от Window-менеджера.

onGenericMotionEvent – вызывается для необработанного события MotionEvent.

onKeyDown – вызывается для необработанного события KeyEvent при нажатии клавиши.

onKeyLongPress – вызывается для необработанного события KeyEvent при долгом нажатии.

onKeyMultiple – вызывается для необработанного события KeyEvent при многократном нажатии одной клавиши.

onKeyShortcut – вызывается для необработанного события KeyEvent при нажатии комбинации клавиш.

onKeyUp – вызывается для необработанного события KeyEvent при освобождении клавиши.

onLowMemory – вызывается при уменьшении оперативной памяти, когда система будет вынуждена остановить все фоновые процессы, используется для освобождения всех ненужных ресурсов.

onMenuItemSelected – вызывается при выборе элемента меню.

onMenuOpened – вызывается при открытии меню.

onNavigateUp – вызывается при нажатии кнопки Up.

onNavigateUpFromChild – вызывается, если дочерний Activity-компонент использует Up-навигацию.

onNewIntent – при запуске данного Activity-компонента другим Android-компонентом вызывается для уже существующего экземпляра Activity-компонента переднего плана своей задачи, имеющего атрибут android: launchMode=«singleTop» файла манифеста, или если вызывающий Android-компонент использует метод startActivity с флагом

FLAG_ACTIVITY_SINGLE_TOP Intent-объекта, вместо создания нового экземпляра Activity-компонента.

onOptionsItemSelected – вызывается при выборе элемента меню опций.

onOptionsMenuClosed – вызывается при закрытии меню опций.

onPanelClosed – вызывается при закрытии панели меню.

onPostCreate – вызывается после вызова метода onRestoreInstanceState.

onPostResume – вызывается после вызова метода onResume.

onPrepareNavigateUpTaskStack – вызывается перед созданием стека задач.

onPrepareOptionsMenu – вызывается перед открытием меню опций.

onPreparePanel – вызывается перед открытием панели меню.

onProvideAssistData – вызывается, когда пользователь запрашивает помощь.

onRestoreInstanceState – вызывается после метода onStart для восстановления состояния Activity-компонента из объекта android. os. Bundle.

onSaveInstanceState – вызывается перед уничтожением Activity-компонента, перемещенного с переднего плана, Android-системой для освобождения ресурсов памяти. Данный метод предназначен для сохранения состояния Activity-компонента в объекте android. os. Bundle в виде пар имя-значение. Измененный объект Bundle передается Android-системой в методы onCreate (Bundle) и onRestoreInstanceState (Bundle).

onSearchRequested – вызывается при запуске поиска.

onTouchEvent – вызывается для необработанного события MotionEvent при прикосновении к экрану.

onTrackballEvent – вызывается для необработанного события MotionEvent при перемещении указателя.

onTrimMemory – вызывается при сокращении ненужной памяти у процесса.

onUserInteraction – вызывается при взаимодействии с пользователем.

onUserLeaveHint – вызывается, когда Activity-компонент перемещается на задний план в результате действий пользователя.

onWindowAttributesChanged – вызывается при изменении атрибутов окна.

onWindowFocusChanged – вызывается при потере или получении фокуса окном.

onWindowStartingActionMode – вызывается при запуске режима ActionMode для окна.

Другой метод обратного вызова класса android.app.Activity, который рекомендуется переопределять – это метод onPause (), вызываемый при потере фокуса Activity-компонентом и который предназначен для сохранения состояния Activity-компонента, так как Android-приложение не контролирует полностью жизненный цикл своих компонентов – Android-система может уничтожать приостановленные Activity-компоненты для освобождения ресурсов памяти.

В методе onPause () производится сохранение данных, общих для приложения или для использования другими приложениями, с помощью ContentProvider-компонента, или прямое сохранение измененных данных с помощью объекта SharedPreferences (сохранение пар имя-значение примитивных типов данных), метода openFileOutput () класса android.content.Context (сохранение данных во внутреннем хранилище устройства), метода getCacheDir () класса android.content.Context (кэширование данных), метода getExternalStorageDirectory () класса android. os. Environment (сохранение данных в карте памяти), сохранение данных в базе данных SQLite, в Web-сервисах с использованием пакетов java.net.* и android.net.*.

Использование метода onPause () для сохранения состояния Activity-компонента имеет свои преимущества, по сравнению с применением метода onSaveInstanceState (), так как метод onSaveInstanceState () не будет вызываться Android-системой, если Activity-компонент был уничтожен пользователем, например, нажатием клавиши BACK.

Переопределение методов onCreate (), onStart (), onRestart (), onResume (), onPause (), onStop (), onDestroy () и др. должно сопровождаться вызовом суперкласса с помощью ключевого слова super.

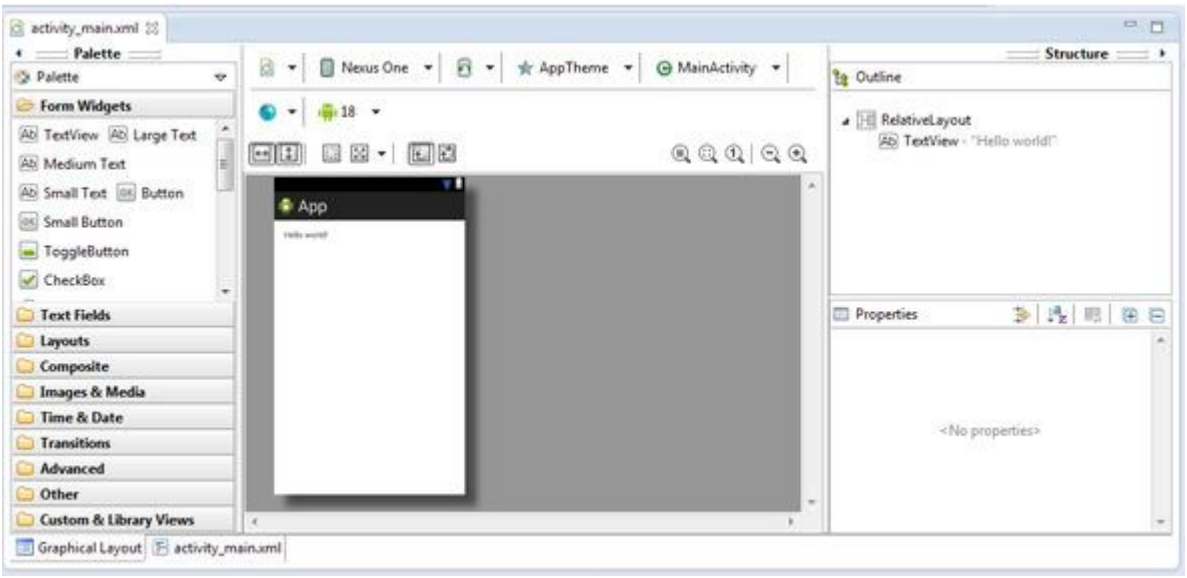
В переопределенном методе onCreate () класса Activity-компонента сгенерированной основы Android-проекта вызывается метод setContentView () класса android.app.Activity, устанавливающий GUI-интерфейс Activity-компонента на основе XML-файла activity_main. xml каталога ресурсов res/layout проекта.

В переопределенном методе onCreateOptionsMenu () класса Activity-компонента сгенерированной основы Android-проекта методом getMenuInflater () получается объект android.view.MenuInflater, отвечающий за создание объектов меню из XML-описания.

Переопределенный метод onOptionsItemSelected () класса Activity-компонента сгенерированной основы Android-проекта представляет каркас обработки выбора элементов меню.

Layout-редактор ADT-плагина

Для работы с XML-описанием GUI-интерфейса Activity-компонента ADT-плагин предлагает визуальный графический редактор.



Layout-редактор ADT-плагина имеет вкладку **Graphical Layout** для визуального редактирования GUI-интерфейса и XML-вкладку, отображающую код Layout-файла.

XML-код Layout-файла сгенерированной основы Android-проекта Blank Activity определяет GUI-интерфейс, состоящий из RelativeLayout-контейнера, содержащего TextView-компонент.

RelativeLayout-контейнер представлен классом android.widget.RelativeLayout, обеспечивающим компоновку дочерних компонентов android.view.View друг относительно друга и относительно родительского компонента.

Компоновку View-компонентов определяют XML-атрибуты.

Атрибуты android: layout_above и android: layout_below располагают компонент выше или ниже компонента с указанным идентификатором.

Атрибуты android: layout_toLeftOf, android: layout_toStartOf и android: layout_toRightOf, android: layout_toEndOf располагают компонент слева или справа компонента с указанным идентификатором.

Атрибуты android: layout_alignLeft, android: layout_alignStart, android: layout_alignRight, android: layout_alignEnd, android: layout_alignBottom, android: layout_alignTop выравнивают стороны компонента по сторонам компонента с указанным идентификатором.

Атрибут android: layout_alignBaseline выравнивает компонент по базовой линии компонента с указанным идентификатором.

Атрибуты android: layout_alignParentBottom, android: layout_alignParentEnd, android:

layout_alignParentTop, android: layout_alignParentStart, android: layout_alignParentLeft, android: layout_alignParentRight располагают компонент внизу, вверху, в левой и в правой части родительского компонента.

Атрибут android: layout_alignWithParentIfMissing со значением true определяет расположение компонента по умолчанию относительно родительского компонента.

Атрибуты android: layout_centerHorizontal, android: layout_centerInParent, android: layout_centerVertical располагают компонент по центральной горизонтальной линии, по центру и по центральной вертикальной линии родительского компонента.

Атрибуты android: layout_marginBottom, android: layout_marginEnd, android: layout_marginLeft, android: layout_marginRight, android: layout_marginStart, android: layout_marginTop определяют отступы компонента.

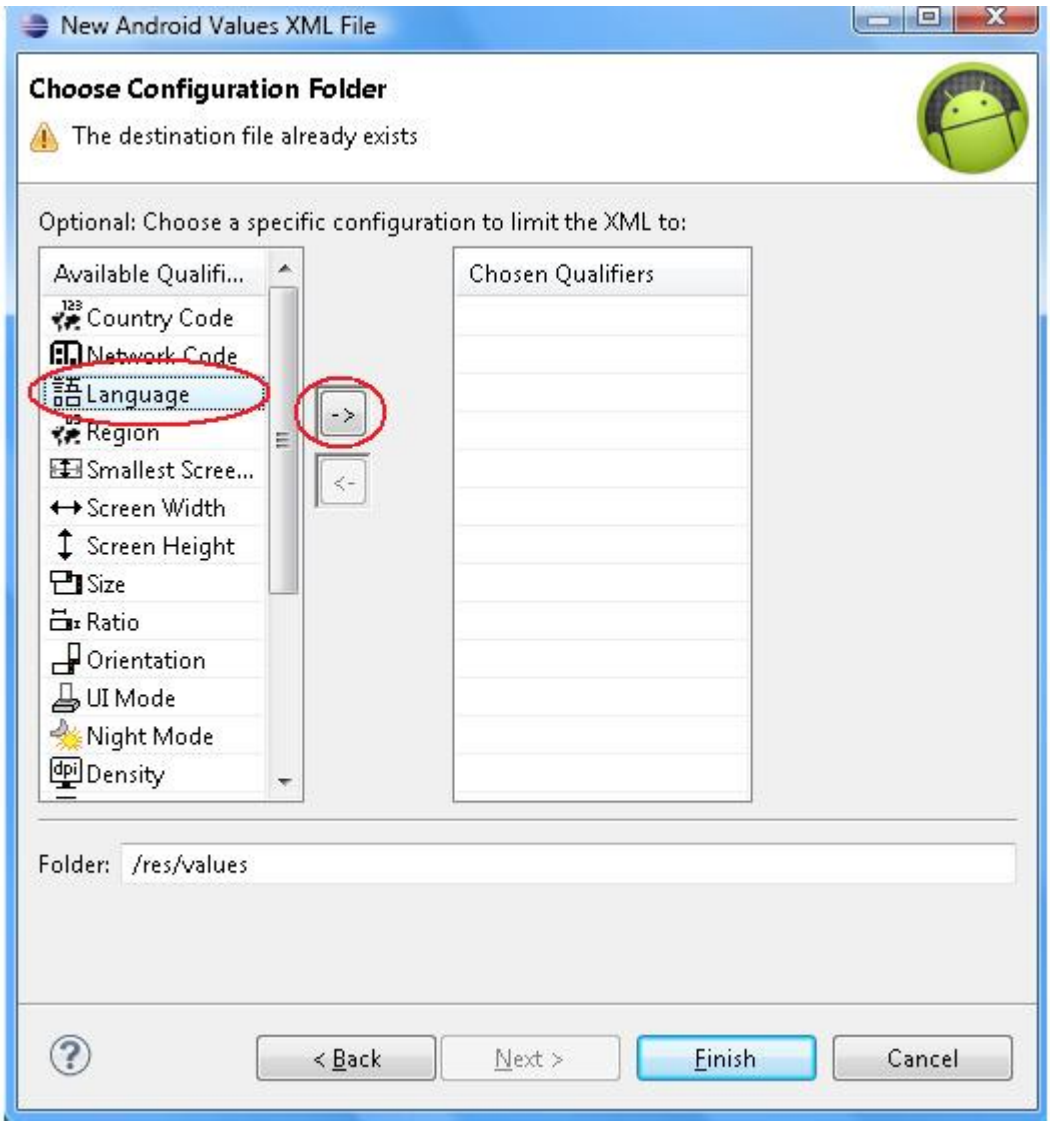
Атрибуты android: layout_height и android: layout_width указывают размеры компонента, при этом константы FILL_PARENT, MATCH_PARENT и WRAP_CONTENT определяют заполнение родительского компонента, соответствие размерам родительского компонента и соответствие содержимому. Данные атрибуты могут принимать значения в виде px (пиксели), dp (виртуальные пиксели, px = dp * (dpi / 160)), sp (масштабируемые пиксели, основанные на предпочтительном размере шрифта), in (дюймы), mm (миллиметры).

TextView-компонент представлен классом android. widget. TextView, обеспечивающий отображение текста пользователю. XML-атрибуты android: layout_height и android: layout_width со значением «wrap_content» устанавливает высоту и ширину компонента, определяемые размером его содержимого. XML-атрибут android: text со значением @string/hello_world устанавливает текстовое содержимое компонента в виде значения строкового ресурса файла strings. xml Android-проекта с именем hello_world.

Интернационализация

Кнопка **Any** вкладки **Graphical Layout** указывает, что данный Android-проект не обеспечивает интернационализацию и локализацию приложения.

Для интернационализации Android-приложения в окне **Package Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML Values File** и нажмем кнопку **Next** . В окне мастера в поле **File:** введем имя файла strings. xml и нажмем кнопку **Next** , в списке **Optional: Choose a specific configuration to limit the XML to:** выберем **Language** и нажмем кнопку **->** . В поле **Language** введем ru и нажмем кнопку **Finish** . В результате в каталоге res проекта будет создана папка values-ru с файлом strings. xml.



Для работы с Values-файлами Android-проекта ADT-плагин также предлагает визуальный графический редактор, имеющий вкладку **Resources** для визуального редактирования и XML-вкладку, отображающую код Values-файла.

Для создания локализованных строк Android-приложения откроем файл strings. xml каталога res/values-ru в редакторе и нажмем кнопку **Add** вкладки **Resources** , в предложенном списке выберем элемент **String** и нажмем кнопку **OK** . В поле **Name** введем имя элемента «hello_world», а в поле **Value** введем строку «Привет!». Еще раз нажмем кнопку **Add** вкладки **Resources** , в предложенном списке выберем элемент **String** и нажмем кнопку **OK** . В поле **Name** введем имя элемента «app_name», а в поле **Value** введем строку «Приложение Андроид». Таким образом, файл strings. xml каталога res/values будет локализован для России.

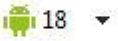
Откроем файл activity_main. xml каталога ресурсов res/layout Android-проекта и увидим, что кнопка **Locale…** вкладки **Graphical Layout** изменилась на список с элементами **Russian (ru)** (русская версия) b **Other** (английская версия), при выборе которых в окне конечного вида GUI-интерфейса вкладки **Graphical Layout** будет отображаться соответствующий текст TextView-компонента.

После инсталляции и запуска Android-приложения в виртуальном устройстве с помощью выбора команды **Run As | Android Application** контекстного меню окна **Package Explorer** , нажмем кнопки **Home** и **Settings** устройства и выберем настройки **Language & keyboard** , в настройке **Select locale** выберем **Русский** – в результате Android-приложение будет отображать GUI-интерфейс в русской версии.

Другой, более быстрый способ интернационализации Android-приложения – это использование команды **Add New Translation** кнопки **Locale…** , открывающей диалоговое окно, в котором список **Language** позволяет выбрать язык локализации, а поля **New Translation** – ввести локализованные значения строковых ключей.

Панель инструментов Graphical Layout

Кнопка **Android…** вкладки **Graphical Layout** позволяет посмотреть конечный вид GUI-интерфейса относительно установленных версий Android-платформы.



Меню кнопки **Configuration…** вкладки **Graphical Layout** позволяет посмотреть конечный вид GUI-интерфейса для различных типов устройств, различных размеров экрана, различных локализаций, фрагментов и версий. Команда **Manual Previews** в сочетании с командой **Add As Thumbnail** дает возможность сформировать свой список просмотра.

Команда **Create New** кнопки **Configuration…** вкладки **Graphical Layout** обеспечивает создание альтернативных версий файла activity_main. xml описания GUI-интерфейса Activity-компонента для различных конфигураций Android-устройства. При запуске Android-приложения среда выполнения Android-устройства будет загружать подходящий ее конфигурации Layout-файл. Команда **Create New** предлагает следующие спецификаторы Android-конфигураций:

Country Code и Network Code – альтернатива языковой и региональной локализации.

LTR – layout-direction-left-to-right (определитель ldltr) направление письменности слева направо.

sw [n] dp – создает Layout-файл каталога res/layout-sw [n] dp для наименьшего размера из высоты и ширины ndp.

w [n] dp – создает Layout-файл каталога res/layout-w [n] dp для минимальной ширины экрана ndp.

h [n] dp – создает Layout-файл каталога res/layout-h [n] dp для минимальной высоты экрана ndp.

Small, Normal, Large, Xlarge – создает Layout-файл каталога res/layout- [small, normal, large, xlarge] для различных разрешений экрана 320x426, 320x470, 480x640, 720x960.

Long, Not Long – создает Layout-файл каталога res/layout-long и res/layout-notlong для широких экранов WQVGA, WVGA, FWVGA и для экранов QVGA, HVGA, VGA.

Portrait, Landscape – создает Layout-файл каталога res/layout-port и res/layout-land для вертикальной и горизонтальной ориентации экрана.

Not Night, Night – создает Layout-файл каталога res/layout-notnight и res/layout-night для работы в дневное и ночное время.

Low Density, Medium Density, High Density, Extra High Density, TV Density – создает

Layout-файл каталога res/layout-ldpi, res/layout-mdpi, res/layout-hdpi, res/layout-xhdpi, res/layout-tvdpi для плотности экрана 120dpi, 160dpi, 240dpi, 320dpi, 213dpi.

Finger – создает Layout-файл каталога res/layout-finger для сенсорного экрана.

Soft – создает Layout-файл каталога res/layout-keyssoft для устройства с виртуальной клавиатурой.

No Keys – создает Layout-файл каталога res/layout-nokeys для устройства без аппаратной клавиатуры.

Hidden, Exposed – создает Layout-файл каталога res/layout-navhidden, res/layout-navexposed для устройства без и с кнопками навигации.

None, Trackball – создает Layout-файл каталога res/layout-nonav, res/layout-trackball для устройства, предоставляющим навигацию только с помощью сенсорного экрана, и для устройства с трекболом.

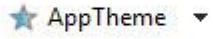
800x480 – создает Layout-файл каталога res/layout-1280x800 для экрана с разрешением 1280x800.

API 18 – создает Layout-файл каталога res/layout-v18 для устройства с Android-платформой 4.3 и выше.

Меню кнопки обеспечивает просмотр конечного вида GUI-интерфейса для различных типа экранов.

Меню кнопки дает возможность посмотреть конечный вид GUI-интерфейса для вертикальной и горизонтальной ориентации экрана (Portrait и Landscape), в нормальном состоянии, в настольном и автомобильном держателях, при соединении с телевизором, без экрана (Normal, Car Dock, Desk Dock, Television, Appliance), для Android-устройства, работающего в дневное и ночное время (Day Time и Night Time).

Меню кнопки обеспечивает просмотр конечного вида GUI-интерфейса с применением различных стилей для приложения.



Для всего приложения стиль устанавливается с помощью атрибута android: theme="@style/AppTheme» тэга <application> файла манифеста AndroidManifest. xml и ресурса res/values/styles. xml.

Для Activity-компонента стиль устанавливается с помощью атрибута android: theme="@style/ActivityTheme» тэга <activity> файла манифеста AndroidManifest. xml и ресурса res/values/styles. xml.

Применение стиля к Activity-компоненту может существенно менять отображение его GUI-интерфейса на экране Android-устройства. Например, при установке стиля Theme. Dialog, Activity-компонент отображается в виде диалогового окна, не заполняя полностью весь экран.

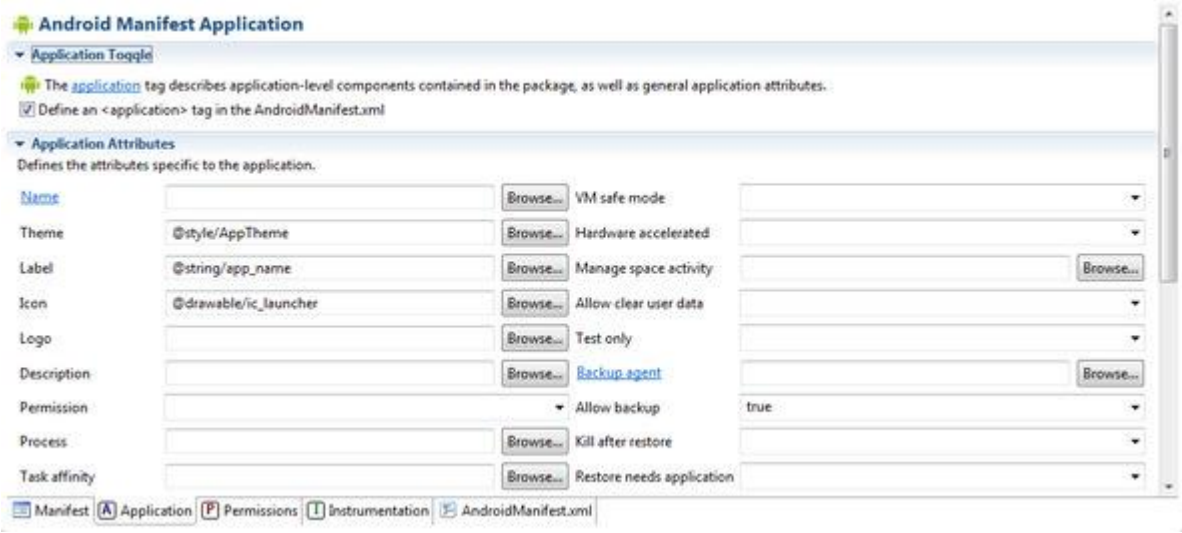
Кнопки вкладки **Graphical Layout** , расположенные ниже панели кнопок с меню, обеспечивают регулировку значений атрибутов android: layout_width, android: layout_height и др. корневого контейнера, а также эмуляцию размера экрана и увеличение-уменьшение изображения экрана.

Palette-палитра вкладки **Graphical Layout** позволяет визуально заполнить GUI-интерфейс Activity-компонента View-компонентами с помощью перетаскивания элементов Palette-палитры в область просмотра конечного вида GUI-интерфейса.

Кроме того, вкладка **Graphical Layout** имеет контекстное меню, открывающееся при нажатии правой кнопкой мышки на View-компоненте в окне просмотра конечного вида GUI-интерфейса, с помощью опций которого можно изменять свойства выбранного View-компонента.

Редактор файла AndroidManifest. xml ADT-плагина

Для файла манифеста AndroidManifest. xml ADT-плагин также предоставляет визуальный графический редактор.



Редактор файла AndroidManifest. xml ADT-плагина имеет вкладки **Manifest** , **Application** , **Permissions** , **Instrumentation** и **AndroidManifest. xml** .

Набор опций вкладок **Manifest** и **Application** зависит от версии Android-платформы, на основе которой создан Android-проект.

Вкладка **Manifest** ADT-редактора файла AndroidManifest. xml содержит следующие поля и ссылки:

Package – редактирование имени пакета Android-приложения, значение атрибута package элемента <manifest>.

Version Code – редактирование версии Android-приложения, значение атрибута android: versionCode элемента <manifest>.

Version name – редактирование строки, представляющей пользователю версию Android-приложения, значение атрибута android: versionName элемента <manifest>.

Shared user id – если данное приложение является одним из модулей большого Android-приложения, установка данного идентификатора одинаковым для всех модулей с подписанием их одним сертификатом дает взаимный доступ к данным, значение атрибута android: sharedUserId элемента <manifest>.

Shared user label – отображаемая пользователю метка sharedUserId-идентификатора, значение атрибута android: sharedUserLabel элемента <manifest>.

Раздел Manifest Extras – с помощью кнопки **Add** обеспечивает добавление в манифест следующих тэгов:

<uses-sdk> (элемент **Uses Sdk**) – указывает совместимость с версиями Android-платформы.

<supports-screens> (элемент **Supports Screens**) – указывает поддержку Android-приложением различных экранов.

<uses-configuration> (элемент **Uses Configuration**) – указывает, какие опции устройства требуются для работы Android-приложения.

<uses-feature> (элемент **Uses Feature**) – указывает для других Android-приложений, от какой опции устройства зависит работа данного Android-приложения.

<protected-broadcast> (элемент **Protected Broadcast**) – указывает Broadcasts-сообщения, которые может посылать только Android-система.

<compatible-screens> (элемент **Compatible Screens**) – указывает для Android Market совместимость приложения с конфигурациями экрана, используя тэги <screen> (элемент **Screen** кнопки **Add**).

<original-package> (элемент **Original Package**) – предназначен только для системных приложений.

<package-verifier> (элемент **Package Verifier**) – указывает имя пакета

приложения, которое вызывается PackageManager-сервисом при инсталляции данного приложения. PackageManager-сервис посылает Broadcast-сообщение ACTION_PACKAGE_NEEDS_VERIFICATION указываемому пакету, который должен содержать BroadcastReceiver-компонент для верификации инсталляции.

Exporting – содержит ссылки **Use the Export Wizard** и **Export an unsigned APK** , запускающие опции экспорта подписанного и неподписанного для публикации Android-приложения.

Links – содержит ссылки Application (открывает вкладку **Application** редактора), Permission (открывает вкладку **Permission** редактора), Instrumentation (открывает вкладку **Instrumentation** редактора), XML Source (открывает вкладку **AndroidManifest. xml** редактора), Documentation (пытается открыть локализованную страницу документации).

Вкладка **Application** ADT-редактора файла AndroidManifest. xml помогает редактировать тэг <application> файла манифеста с помощью разделов **Application Toggle** , **Application Attributes** и **Application Nodes** .

Раздел **Application Toggle** вкладки **Application** содержит ссылку **application** – открывает страницу документации элемента <application> и флажок **Define an <application> tag in the AndroidManifest. xml** – включает элемент <application> в файл манифеста.

Раздел **Application Attributes** вкладки **Application** определяет атрибуты элемента <application> с помощью следующих полей и списков:

Name – при нажатии открывает мастер создания Java-класса, расширяющего класс android. app. Application. Созданный Application-класс указывается в качестве значения атрибута android: name тэга <application>. Если приложение содержит несколько Activity-компонентов, решить проблему обеспечения для них общих глобальных в рамках приложения данных и сервисов поможет Application-класс. При запуске приложения Android-система создаст единственный экземпляр Application-класса и будет вызывать его методы жизненного цикла. Рекомендуется реализовать Application-класс как Singleton-класс со статическим доступом к глобальным данным и сервисам.

Theme – общий для Activity-компонентов стиль, указываемый как значение атрибута android: theme тэга <application>. Предварительно необходимо создать ресурсный файл каталога res/values со стилем, используя команду **New | Other | Android | Android XML Values File** контекстного меню окна **Package Explorer** , дополнить его тэгом <style>, нажать кнопку **Browse** поля **Theme** и выбрать созданный ресурс – в результате у тэга <application> появится атрибут android: theme.

Label – отображаемое пользователю имя приложения, указываемое значением атрибута android: label тэга <application>. Кнопка **Browse** поля **Label** позволяет выбрать значение атрибута в ресурсном файле каталога res/values, содержащем тэги <string>.

Icon – значок приложения, определяемый значением атрибута android: icon тэга <application>. Кнопка **Browse** поля **Icon** позволяет выбрать значение атрибута как имя файла изображения, расположенного в каталоге res/drawable. Папки drawable могут иметь спецификаторы ldpi, mdpi, hdpi, xhdpi, nodpi и tvdpi, обеспечивающие отображение значка на экранах с различной плотностью.

Logo – определяет значение атрибута android: logo тэга <application>, указывающего логотип приложения для отображения в панели ActionBar.

Description – краткое описание приложения, которое указывается значением атрибута android: description тэга <application> и должно определяться ссылкой на строковый ресурс. Кнопка **Browse** поля **Description** позволяет выбрать значение атрибута в ресурсном файле каталога res/values, содержащем тэги <string>.

Permission – список позволяет выбрать разрешение, которое должно иметь стороннее Android-приложение для взаимодействия с данным Android-приложением в целом, указывается значением атрибута android: permission тэга <application>.

Process – определяет значение атрибута android: process тэга <application>,

указывающего имя процесса приложения. Если данное приложение является одним из модулей большого Android-приложения, которые имеют одинаковый sharedUserId-идентификатор и подписаны одним сертификатом, тогда установка значения атрибута android: process одинаковым для всех модулей обеспечивает их запуск в одном процессе.

Task affinity – определяет значение атрибута android: taskAffinity тэга <application>, указывающего имя задачи для всех Activity-компонентов приложения, по умолчанию – имя пакета приложения. Task-задача представляет собой набор Activity-компонентов, с которыми пользователь взаимодействует для выполнения своей задачи, при этом Activity-компоненты задачи организуются в обратный стек, в порядке, в котором каждый Activity-компонент был запущен другим Activity-компонентом.

Allow task reparenting – определяет значение атрибута android: allowTaskReparenting тэга <application> – если true, тогда Activity-компоненты приложения могут перемещаться из задачи, которая их запустила, в задачу переднего плана, с которой Activity-компоненты имеют общее taskAffinity-значение, по умолчанию false.

Has code – определяет значение атрибута android: hasCode тэга <application> – если false, тогда приложение не содержит Java-кода, а полностью реализовано на основе программного интерфейса NDK API, по умолчанию true.

Persistent – определяет значение атрибута android: persistent тэга <application> – если true, тогда приложение работает до тех пор, пока работает устройство, обычно используется системными приложениями, по умолчанию false.

Enabled – определяет значение атрибута android: enabled тэга <application> – если false, тогда Android-система не может создавать экземпляры компонентов приложения, по умолчанию true.

Debuggable – определяет значение атрибута android: debuggable тэга <application>. Android-инструменты сборки ADT-плагина автоматически добавляют значение атрибута true в отладочном режиме и удаляют данный атрибут, имеющий по умолчанию значение false, при экспорте релиза приложения.

Vm safe mode – определяет значение атрибута android: vmSafeMode тэга <application> – если true, тогда JIT-оптимизация отключается.

Hardware accelerated – определяет значение атрибута android: hardwareAccelerated тэга <application> – если true, тогда включается аппаратное ускорение визуализации, по умолчанию false.

Manage space activity – определяет значение атрибута android: manageSpaceActivity тэга <application>, указывает имя Activity-компонента, который запускается дополнительной кнопкой **Управление местом** в разделе настроек **Приложения | Управление приложениями** Android-устройства.

Allow clear user data – определяет значение атрибута android: allowClearUserData тэга <application> – применимо только для системных приложений, для обычных приложений игнорируется.

Test only – определяет значение атрибута android: testOnly тэга <application> – если true, тогда приложение находится в стадии тестирования и не может быть инсталлировано в Android-устройстве.

Backup agent – определяет значение атрибута android: backupAgent тэга <application>, указывает имя класса, расширяющего класс android.app.backup.BackupAgent, который вызывается сервисом Backup Manager для определения настроек приложения, сохраняемых в облачном хранилище, и их восстановления при реинсталляции приложения в случае обновления Android-системы устройства.

Allow backup – определяет значение атрибута android: allowBackup тэга <application> – если false, тогда приложение не обслуживается сервисом Backup Manager, по умолчанию true.

Kill after restore – определяет значение атрибута android: killAfterRestore тэга <application> – используется системными приложениями.

Restore needs application – определяет значение атрибута android: restoreNeedsApplication тэга <application> – используется системными приложениями.

Restore any version – определяет значение атрибута android: restoreAnyVersion тэга <application> – если true, тогда сервис Backup Manager будет восстанавливать приложение даже в том случае, если версии облачного хранилища и текущей инсталляции не совпадают, по умолчанию false.

Never encrypt – определяет значение атрибута android: neverEncrypt тэга <application> – если true, тогда приложение отказывается от защиты хранения своих данных.

Large heap – определяет значение атрибута android: largeHeap тэга <application> – если true, тогда приложению может понадобиться расширение размера кучи.

Cant save state – определяет значение атрибута android: cantSaveState тэга <application> – если true, тогда приложение является ресурсоемким и отказывается участвовать в сохранении-восстановлении Android-системой своего состояния. При таком работающем приложении, если пользователь пытается загрузить другое приложение, он запрашивается на выход из первого приложения.

Ui options – определяет значение атрибута android: uiOptions тэга <application>, указывающее дополнительные опции GUI-интерфейса Activity-компонентов приложения с помощью двух значений: none (по умолчанию, нет дополнительных опций) и splitActionBarWhenNarrow (добавляет панель ActionBar, разделенную на секцию навигации и панель действий).

Supports rtl – определяет значение атрибута android: supportsRtl тэга <application> – если true, тогда приложение поддерживает right-to-left (RTL) письменность справа налево.

Раздел **Application Nodes** вкладки **Application** кнопкой **Add** обеспечивает добавление в тэг <application> тэгов <activity> (элемент **Activity**), <activity-alias> (элемент **Activity Alias**), <meta-data> (элемент **Meta Data**), <provider> (элемент **Provider**), <receiver> (элемент **Receiver**), <service> (элемент **Service**), <uses-library> (элемент **Uses Library**).

Кнопка **Add** позволяет добавлять в тэги <activity>, <receiver> и <service> тэги <intent-filter> (элемент **Intent Filter**) и <meta-data> (элемент **Meta Data**), при этом в тэг <intent-filter> могут добавляться кнопкой **Add** тэги <action> (элемент **Action**), <category> (элемент **Category**), <data> (элемент **Data**).

В тэг <provider> кнопка **Add** добавляет тэги <grant-uri-permission> (элемент **Grant Uri Permission**), <meta-data> (элемент **Meta Data**), <path-permission> (элемент **Path Permission**).

Тэг <activity> (элемент **Activity**) описывает Activity-компонент приложения (класс, расширяющий класс android.app.Activity). При выборе элемента **Activity** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Activity** , позволяющий определить атрибуты тэга <activity> с помощью следующих полей и списков:

Name – при нажатии открывает мастер создания Java-класса, расширяющего класс android.app.Activity. Созданный Activity-компонент указывается в качестве значения атрибута android: name.

Theme – определяет для Activity-компонента стиль, указываемый как значение атрибута android: theme.

Label – отображаемая пользователю метка Activity-компонента, указываемая значением атрибута android: label.

Icon – значок Activity-компонента, определяемый значением атрибута android: icon.

Logo – определяет значение атрибута android: logo, указывающего логотип приложения

для отображения в панели ActionBar.

Launch mode – список позволяет выбрать значение атрибута android: launchMode, определяющего загрузку Activity-компонента при получении вызывающего Intent-объекта:

standart (по умолчанию) – Android-система всегда создает новый экземпляр Activity-компонента в целевой задаче и передает ему Intent-объект.

singleTop – если экземпляр Activity-компонента уже существует на переднем плане целевой задачи, вызывается метод onNewIntent () уже существующего экземпляра, вместо создания нового экземпляра Activity-компонента.

singleTask – Android-система создает новый экземпляр Activity-компонента в новой задаче и передает ему Intent-объект. Если экземпляр Activity-компонента уже существует, тогда вызывается его метод onNewIntent (), вместо создания нового экземпляра Activity-компонента.

singleInstance – работает аналогично singleTask, за исключением того, что задача может содержать только один Activity-компонент.

Screen orientation – список позволяет выбрать значение атрибута android: screenOrientation, определяющего ориентацию отображения Activity-компонента на экране:

unspecified (по умолчанию) – ориентацию выбирает Android-система.

user – ориентация определяется пользовательскими предпочтениями.

behind – ориентация такая же, как и у предыдущего Activity-компонента.

landscape – альбомная (горизонтальная) ориентация.

portrait – портретная (вертикальная) ориентация.

reverseLandscape – альбомная (горизонтальная) ориентация в противоположном направлении.

reversePortrait – портретная (вертикальная) ориентация в противоположном направлении.

sensorLandscape – альбомная (горизонтальная) ориентация, направление которой определяется Android-системой на основе сенсора.

sensorPortrait – портретная (вертикальная) ориентация, направление которой определяется Android-системой на основе сенсора.

sensor – ориентация определяется Android-системой на основе сенсора.

fullSensor – ориентация определяется Android-системой на основе сенсора с возможностью ориентаций landscape, portrait, reverseLandscape и reversePortrait.

nosensor – сенсор устройства игнорируется.

Config changes – кнопка **Select** позволяет выбрать значение атрибута android: configChanges, определяющего изменения конфигурации, при которых Activity-компонент не перезапускается, а вызывается его метод onConfigurationChanged ():

mcc – изменение MCC-кода страны.

mnc – изменение MNC-кода сети.

locale – изменение локализации устройства.

touchscreen – изменение сенсорного экрана.

keyboard – изменение типа клавиатуры устройства.

keyboardHidden – изменение доступности клавиатуры.

navigation – изменение механизма навигации устройства.

screenLayout – изменение компоновки экрана.

fontScale – изменение размера шрифта.

uiMode – изменение состояния устройства (устройство помещено в держатель).

orientation – изменилась ориентация экрана.

screenSize – при изменении ориентации экрана изменились пропорции экрана.

smallestScreenSize – при подключении устройства к внешнему дисплею изменился размер экрана.

Permission – список позволяет выбрать разрешение, которое должно иметь стороннее Android-приложение для вызова Activity-компонента, указывается значением атрибута

android: permission.

Multiprocess – определяет значение атрибута android: multiprocess – если true, тогда Activity-компонент запускается в том же процессе, что и вызвавший его Android-компонент.

Process – определяет значение атрибута android: process, указывающего имя процесса, в котором запускается Activity-компонент.

Task affinity – определяет значение атрибута android: taskAffinity, указывающего имя задачи, в которой запускается Activity-компонент с флагом FLAG_ACTIVITY_NEW_TASK.

Allow task reparenting – определяет значение атрибута android: allowTaskReparenting – если true, тогда Activity-компонент может перемещаться из задачи, которая его запустила, в задачу переднего плана, с которой Activity-компонент имеет общее taskAffinity-значение, по умолчанию false.

Finish on task launch – определяет значение атрибута android: finishOnTaskLaunch – если true, тогда существующий экземпляр Activity-компонента уничтожается, если пользователь снова запускает его задачу, по умолчанию false.

Finish on close system dialogs – определяет значение атрибута android: finishOnCloseSystemDialogs – если true, тогда Activity-компонент уничтожается при закрытии текущего окна, например при нажатии кнопки HOME или при блокировке устройства.

Clear task on launch – определяет значение атрибута android: clearTaskOnLaunch – если true, тогда при перезапуске задачи из домашнего экрана, задача очищается от всех Activity-компонентов до данного корневого Activity-компонента, по умолчанию false.

No history – определяет значение атрибута android: noHistory – если true, тогда Activity-компонент удаляется из стека задачи и уничтожается, когда становится невидимым на экране, по умолчанию false.

Always retain task state – определяет значение атрибута android: alwaysRetainTaskState – если true, тогда Android-система не очищает задачу данного корневого Activity-компонента, а сохраняет ее последнее состояние, по умолчанию false.

State not need – определяет значение атрибута android: stateNotNeeded – если true, тогда метод onSaveInstanceState () Activity-компонента не вызывается, а его метод onCreate () в качестве аргумента всегда получает null, по умолчанию false.

Exclude from recents – определяет значение атрибута android: excludeFromRecents – если true, тогда Activity-компонент не появляется в списке недавно запущенных Activity-компонентов, который отображается при долгом нажатии на кнопку HOME устройства, по умолчанию false.

Enabled – определяет значение атрибута android: enabled – если false, тогда Android-система не может создавать экземпляры Activity-компонента, по умолчанию true.

Exported – определяет значение атрибута android: exported – если true, тогда Activity-компонент может запускаться другими Android-приложениями, если false, тогда Activity-компонент может запускаться только Android-компонентами своего приложения или другими модулями с общим sharedUserId-идентификатором.

Window soft input mode – кнопка **Select** позволяет выбрать значение атрибута android: windowSoftInputMode, определяющего как окно Activity-компонента взаимодействует с окном экранной клавиатуры:

stateUnspecified (по умолчанию) – состояние видимости или нет экранной клавиатуры выбирает Android-система.

stateUnchanged – экранная клавиатура сохраняет свое последнее состояние.

stateHidden – экранная клавиатура скрыта когда пользователь переходит вперед к Activity-компоненту.

stateAlwaysHidden – экранная клавиатура всегда скрыта.

stateVisible – экранная клавиатура появляется когда пользователь переходит вперед к Activity-компоненту.

stateAlwaysVisible – экранная клавиатура всегда появляется.

adjustUnspecified (по умолчанию) – будет окно Activity-компонента изменять свои

размеры и включать в себя окно экранной клавиатуры или экранная клавиатура будет накладываться на окно Activity-компонента с его панорамированием определяет Android-система.

adjustResize – окно Activity-компонента изменяет свои размеры и включает в себя окно экранной клавиатуры.

adjustPan – экранная клавиатура накладывается на окно Activity-компонента, которое панорамируется на ввод.

adjustNothing – окно Activity-компонента не изменяет свои размеры и не панорамируется.

Immersive – определяет значение атрибута android: immersive – если true, тогда Activity-компонент не прерывается другими Activity-компонентами и уведомлениями.

Hardware accelerated – определяет значение атрибута android: hardwareAccelerated – если true, тогда включается аппаратное ускорение визуализации, по умолчанию false.

Ui options – определяет значение атрибута android: uiOptions, указывающее дополнительные опции GUI-интерфейса Activity-компонента с помощью двух значений: none (по умолчанию, нет дополнительных опций) и splitActionBarWhenNarrow (добавляет панель ActionBar, разделенную на секцию навигации и панель действий).

Parent activity name – определяет значение атрибута android: parentActivityName, указывающее имя класса Activity-компонента, являющегося логическим родителем данному Activity-компоненту и к которому будет осуществляться переход с помощью кнопки Up.

Тэг <intent-filter> (элемент **Intent Filter**) обеспечивает создание объекта android.content.IntentFilter, который указывает Android-системе какие неявные (не указывающие целевой класс) объекты android.content.Intent может обрабатывать Android-компонент. При выборе элемента **Intent Filter** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Intent Filter** , позволяющий определить атрибуты тэга <intent-filter> с помощью следующих полей:

Label – определяет значение атрибута android: label, указывающего отображаемую пользователю метку Android-компонента, запущенного соответствующим фильтру Intent-объектом.

Icon – определяет значение атрибута android: icon, указывающего значок Android-компонента, запущенного соответствующим фильтру Intent-объектом.

Logo – определяет значение атрибута android: logo, указывающего логотип панели ActionBar Android-компонента, запущенного соответствующим фильтру Intent-объектом.

Priority – определяет значение атрибута android: priority, указывающего приоритет обработки соответствующих фильтру Intent-объектов для случая, когда несколько Android-компонентов соответствуют Intent-объекту.

Дочерний тэг <action> (элемент **Action**) тэга <intent-filter> указывает действие Intent-объекта, поддерживаемое Android-компонентом. При выборе элемента **Action** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Action** , позволяющий определить атрибут тэга <action> с помощью списка **Name** , обеспечивающего выбор действия android.intent.action.* как значения атрибута android: name.

Дочерний тэг <category> (элемент **Category**) тэга <intent-filter> указывает, к какому типу принадлежит Android-компонент, чтобы соответствовать категории Intent-объекта. При выборе элемента **Category** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Category** , позволяющий определить атрибут тэга <category> с помощью списка **Name** , обеспечивающего выбор категории android.intent.category.* как значения атрибута android: name.

Дочерний тэг <data> (элемент **Data**) тэга <intent-filter> описывает, какие данные могут быть переданы Intent-объектом Android-компоненту. При выборе элемента **Data** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Data** , позволяющий определить атрибуты тэга <data> с помощью полей **Mime type** (атрибут android: mimeType указывает MIME-тип данных Intent-объекта), **Scheme** , **Host** , **Port** , **Path** ,

Path prefix , **Path pattern** (URI-адрес данных в формате scheme://host: port/path, атрибуты android: scheme, android: host, android: port, android: path, android: pathPrefix, android: pathPattern).

Тэг <meta-data> (элемент **Meta Data**) позволяет добавить дополнительные данные к Android-компоненту, доступ к которым можно получить программным способом:

ApplicationInfo ai = getPackageManager().getApplicationInfo(activity.getPackageName (), PackageManager.GET_META_DATA);

Bundle bundle = ai.metaData;

String myValue = bundle.getString («myKey»);

При выборе элемента **Meta Data** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Meta Data** , позволяющий определить атрибуты тэга <meta-data> с помощью полей **Name** (атрибут android: name определяет имя элемента метаданных), **Value** (атрибут android: value определяет значение элемента метаданных), **Resource** (атрибут android: resource указывает ссылку на ресурс).

Тэг <activity-alias> (элемент **Activity Alias**) обеспечивает запуск целевого Activity-компонента под другим именем, меткой, с другим Intent-фильтром. При выборе элемента **Activity Alias** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Activity Alias** , позволяющий определить атрибуты тэга <activity-alias> с помощью полей и списков:

Name (атрибут android: name указывает псевдоним для целевого Activity-компонента),

Target activity (атрибут android: targetActivity указывает имя целевого Activity-компонента),

Label (атрибут android: label определяет метку псевдонима),

Description (атрибут android: description определяет описание псевдонима),

Icon (атрибут android: icon указывает значок псевдонима),

Logo (атрибут android: logo определяет логотип панели ActionBar),

Permission (атрибут android: permission указывает разрешение, которое должно иметь стороннее Android-приложение для вызова Activity-компонента через псевдоним),

Enabled (атрибут android: enabled указывает возможность создания экземпляра целевого Activity-компонента через псевдоним),

Exported (атрибут android: exported указывает возможность запуска целевого Activity-компонента сторонними Android-приложениями через псевдоним).

Parent activity name – определяет значение атрибута android: parentActivityName, указывающее имя класса Activity-компонента, являющегося логическим родителем данному Activity-компоненту и к которому будет осуществляться переход с помощью кнопки Up.

Тэг <provider> (элемент **Provider**) описывает ContentProvider-компонент приложения (класс, расширяющий класс android.content.ContentProvider), обеспечивающий управление данными приложения. При выборе элемента **Provider** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Provider** с полями и списками, позволяющий определить атрибуты тэга <provider>.

Поле со ссылкой **Name** при нажатии открывает мастер создания Java-класса, расширяющего класс android.content.ContentProvider. Созданный ContentProvider-компонент указывается в качестве значения атрибута android: name.

Поля **Label** , **Description** , **Icon** , **Logo** , **Process** , **Permission** , **Multiprocess** , **Enabled** , **Exported** элемента **Provider** работают аналогично соответствующим полям элемента **Activity** раздела **Application Nodes** вкладки **Application** .

Поле **Authorities** элемента **Provider** определяет значение атрибута android: authorities тэга <provider>, указывающего один или несколько URI-адресов, идентифицирующих для Android-системы ContentProvider-компонент.

Список **Syncable** определяет значение атрибута android: syncable тэга <provider> – если true, тогда данные ContentProvider-компонента синхронизированы с данными сервера.

Поле **Read permission** и поле **Write permission** определяют значения атрибутов android: readPermission и android: writePermission, указывающих разрешения, необходимые для чтения и изменения данных ContentProvider-компонента.

Поле **Grand URI permissions** определяет значение атрибута android: grantUriPermissions – если true, тогда приложению, вызывающему ContentProvider-компонент Intent-объектом с флагами FLAG_GRANT_READ_URI_PERMISSION и FLAG_GRANT_WRITE_URI_PERMISSION, предоставляется одноразовый доступ к данным.

Поле **Init order** определяет значение атрибута android: initOrder, указывающего номер в очереди инициализации ContentProvider-компонентов приложения.

Дочерний тэг <grant-uri-permission> (элемент **Grant Uri Permission**) тэга <provider> указывает URI-адрес ContentProvider-компонента, к которому может быть дан одноразовый доступ стороннему приложению, с помощью полей **Path** , **Path prefix** и **Path pattern** , определяющих значения атрибутов android: path, android: pathPrefix и android: pathPattern.

Дочерний тэг <path-permission> (элемент **Path Permission**) тэга <provider> указывает для URI-адреса ContentProvider-компонента разрешения доступа к его данным сторонним приложениям, используя поля **Path** , **Path prefix, Path pattern** , **Permission** , **Read permission** , **Write permission** , определяющих значения атрибутов android: path, android: pathPrefix, android: pathPattern, android: permission, android: readPermission и android: writePermission.

Тэг <receiver> (элемент **Receiver**) описывает BroadcastReceiver-компонент приложения (класс, расширяющий класс android.content.BroadcastReceiver), позволяющий обрабатывать Intent-объекты, посылаемые широковещательным способом Android-системой или другими приложениями. При выборе элемента **Receiver** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Receiver** с полями и списками, позволяющий определить атрибуты тэга <receiver>. Поле со ссылкой **Name** при нажатии открывает мастер создания Java-класса, расширяющего класс android.content.BroadcastReceiver. Созданный BroadcastReceiver-компонент указывается в качестве значения атрибута android: name.

Поля **Label** , **Description** , **Icon** , **Logo** , **Process** , **Permission** , **Enabled** , **Exported** элемента **Receiver** работают аналогично соответствующим полям элемента **Activity** раздела **Application Nodes** вкладки **Application** .

Тэг <service> (элемент **Service**) описывает Service-компонент приложения (класс, расширяющий класс android.app.Service), предназначенный для выполнения продолжительных операций без предоставления GUI-интерфейса. При выборе элемента **Service** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Service** с полями и списками, позволяющий определить атрибуты тэга <service>. Поле со ссылкой **Name** при нажатии открывает мастер создания Java-класса, расширяющего класс android.app.Service. Созданный Service-компонент указывается в качестве значения атрибута android: name. Поля **Label** , **Description** , **Icon** , **Logo** , **Process** , **Permission** , **Enabled** , **Exported** элемента **Service** работают аналогично соответствующим полям элемента **Activity** раздела **Application Nodes** вкладки **Application** .

Список **Stop with task** элемента **Service** определяет значение атрибута android: stopWithTask тэга <service> – если true, тогда сервис автоматически завершит свою работу при удалении пользователем задачи приложения, по умолчанию false.

Список **Isolated process** элемента **Service** определяет значение атрибута android: isolatedProcess – если true, тогда сервис будет работать в изолированном процессе, не имеющим те разрешения, которые даны остальной части приложения.

Тэг <uses-library> (элемент **Uses Library**) указывает Android-библиотеку, которая требуется для работы приложения. При выборе элемента **Uses Library** кнопкой **Add** , во вкладке **Application** появляется раздел **Attributes for Uses Library** с полями

и списками, позволяющий определить атрибуты тэга <uses-library>. Поле Name определяет значение атрибута android: name, указывающего имя Android-библиотеки, с которой связано приложение, а список Required – значение атрибута android: required – если true (по умолчанию), тогда приложение не может работать и быть инсталлированным без наличия указанной библиотеки в устройстве.

Вкладка **Permissions** ADT-редактора файла AndroidManifest. xml с помощью кнопки **Add** обеспечивает добавление в тэг <manifest> тэгов <permission> (элемент **Permission**), <permission-group> (элемент **Permission Group**), <permission-tree> (элемент **Permission Tree**), <uses-permission> (элемент **Uses Permission**).

Тэг <permission> (элемент **Permission**) позволяет объявить пользовательское разрешение, которое должно получить стороннее приложение для доступа к Android-компонентам данного приложения. При выборе элемента **Permission** кнопкой **Add** , во вкладке **Permissions** появляется раздел **Attributes for Permission** с полями и списками, позволяющими определить атрибуты тэга <permission>.

Поля **Name** , **Label** , **Description** , **Icon** и **Logo** определяют значения атрибутов android: name, android: label, android: description, android: icon и android: logo, указывающих имя, метку, описание, значок и логотип пользовательского разрешения.

Поле **Permission group** определяет значение атрибута android: permissionGroup, указывающего группу разрешений, к которой относится данное разрешение.

Список **Protection level** определяет значение атрибута android: protectionLevel, указывающего уровень риска, который несет данное разрешение:

normal – минимальный риск для других приложений, Android-системы, пользователя.

dangerous – может причинить вред пользователю, например, разрешает доступ к данным пользователя.

signature – Android-система даст данное разрешение запрашивающему его приложению, только если запрашивающее разрешение приложение подписано тем же сертификатом, что и данное приложение, которое объявило пользовательское разрешение.

signatureOrSystem – используется только для системных приложений или приложений, подписанных тем же сертификатом, что и приложение, которое объявило пользовательское разрешение.

system – используется только для системных приложений.

development – разрешения даются только при разработке, но не при инсталляции.

Тэг <permission-group> (элемент **Permission Group**) объявляет группу пользовательских разрешений. При выборе элемента **Permission Group** кнопкой **Add** , во вкладке **Permissions** появляется раздел **Attributes for Permission Group** с полями и списками, позволяющими определить атрибуты тэга <permission-group>.

Поля **Name** , **Label** , **Description** , **Icon** и **Logo** определяют значения атрибутов android: name, android: label, android: description, android: icon и android: logo, указывающих имя, метку, описание, значок и логотип группы пользовательских разрешений.

Поле **Priority** определяет значение атрибута android: priority, указывающего приоритет обработки Intent-объекта.

Тэг <permission-tree> (элемент **Permission Tree**) объявляет базовое имя дерева разрешений, которые могут быть добавлены программным способом с помощью метода addPermission () класса android.content.pm.PackageManager. При выборе элемента **Permission Tree** кнопкой **Add** , во вкладке **Permissions** появляется раздел **Attributes for Permission Tree** с полями, позволяющими определить атрибуты тэга <permission-tree>. Поля **Name** , **Label** , **Icon** и **Logo** определяют значения атрибутов android: name, android: label, android: icon и android: logo, указывающих базовое имя, метку, значок и логотип дерева динамически добавляемых разрешений.

Тэг <uses-permission> (элемент **Uses Permission**) обеспечивает при инсталляции приложения запрос на предоставление ему определенного разрешения, которое указывается атрибутом android: name и может быть выбрано с помощью списка **Name** раздела **Attributes**

for Uses Permission вкладки **Permissions** .

Вкладка **Instrumentation** ADT-редактора файла AndroidManifest. xml с помощью кнопки **Add** обеспечивает добавление в тэг <manifest> тэга <instrumentation>, который используется в файле манифеста проекта Android-тестирования (основа проекта Android-тестирования создается с помощью мастера **Android Test Project**).

При открытии в ADT-редакторе специфических для Android-разработки файлов, таких как activity_main. xml, strings. xml и AndroidManifest. xml, в меню **Refactor** Workbench-окна появляется подменю **Android** , содержащее опции Android-рефакторинга.

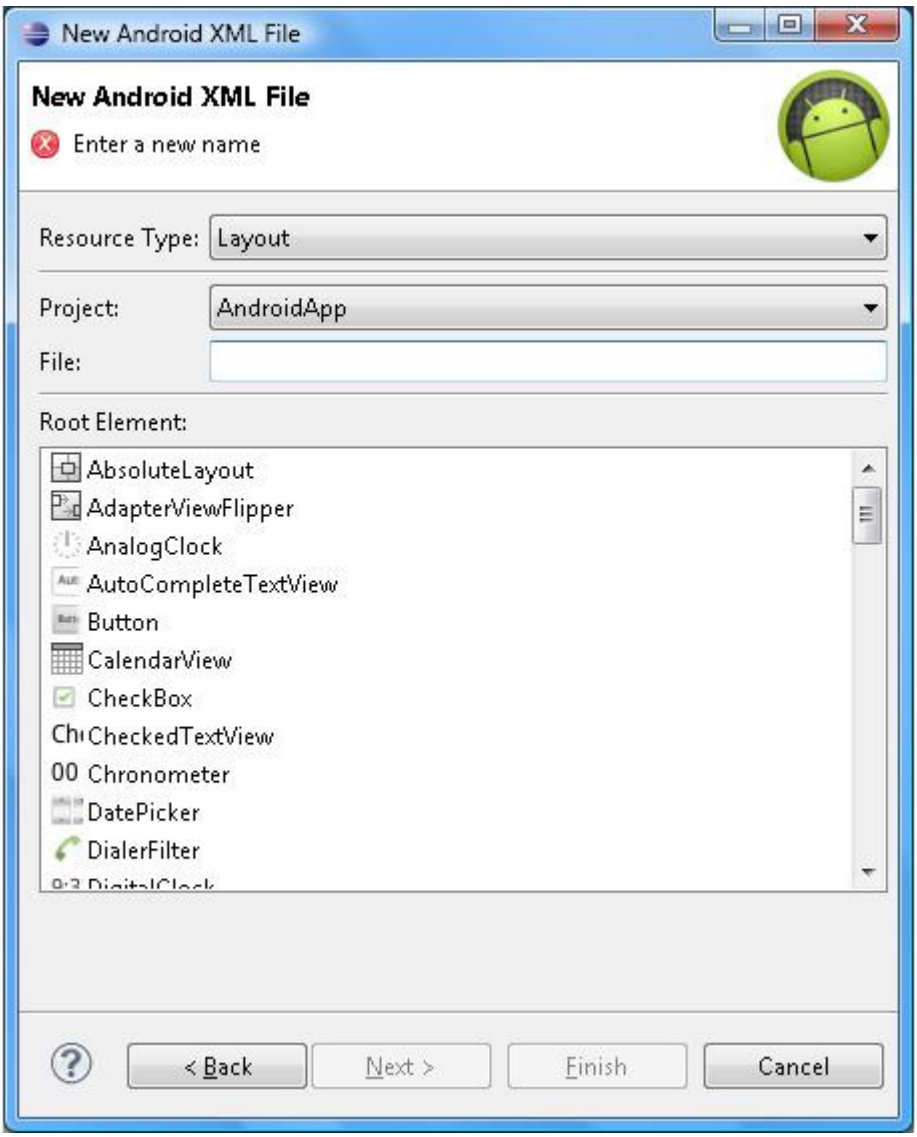
Мастер Android XML File

Мастер **Android XML File** , доступный в разделе **Android** команды **New** , обеспечивает создание набора ресурсов Android-приложения, состоящего из:

XML-описаний GUI-интерфейса Activity-компонентов (тип ресурса Layout),
различного рода значений, используемых приложением (тип ресурса Values),
графики (тип ресурса Drawable),
меню приложения (тип ресурса Menu),
наборов цветов (тип ресурса Color List),
анимации (тип ресурса Property Animation и Tween Animation),
метаданных приложения App Widgets (тип ресурса AppWidget Provider),
GUI-интерфейса PreferenceActivity-операции (тип ресурса Preference),
настроек поиска (тип ресурса Searchable).

Тип ресурса Layout

Для создания Layout-файла Android-приложения в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File** или **Android XML Layout File** , нажмем кнопку **Next** – в результате откроется окно мастера создания Layout-файла, в списке **Resource Type** которого выбран тип **Layout** .



Поле **File:** мастера создания Layout-файла предлагает ввести имя нового файла XML-описания GUI-интерфейса, который затем с расширением. xml появится в каталоге res/layout Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.layout. [имя Layout-файла] или в XML-коде с помощью ссылки @ [package:] layout/ [имя Layout-файла].

Раздел **Root Element:** мастера создания Layout-файла предлагает выбрать корневой View-компонент GUI-интерфейса, который может быть как контейнером для других GUI-компонентов, так и отдельным GUI-компонентом.

В качестве контейнера обычно используются ViewGroup-компоненты LinearLayout (компоновка в столбец или строку), RelativeLayout (якорная компоновка) и FrameLayout (стековая компоновка), а индивидуальные GUI-компоненты представлены такими View-компонентами как кнопка, флажок, переключатель, текстовая область и др.

Помимо контейнера и индивидуального GUI-компонента корневым элементом Layout-файла может служить элемент <merge>, который предназначен для создания Layout-файла, включаемого в другой Layout-файл с помощью тэга <include>. Тэг индивидуального GUI-компонента может также содержать тэг <requestFocus>, дающий первоначальный фокус View-компоненту.

Тэг <fragment>, начиная с версии Android 3.0 (API level 11), обеспечивает модульность GUI-интерфейса Activity-компонента, описывая его часть, которая имеет свой жизненный цикл, свое взаимодействие с пользователем и с другими компонентами приложения и которая может добавляться или удаляться во время работы родительского

Activity-компонента. Тэг <fragment> не может быть корневым элементом Layout-файла, а включается в основной Layout-файл в качестве дочернего тэга контейнера LinearLayout, RelativeLayout или FrameLayout. При этом атрибут android: name указывает имя класса фрагмента, расширяющего класс android.app.Fragment или android.support.v4.app.Fragment.

После ввода имени нового Layout-файла, выбора его корневого элемента и нажатия кнопки **Next** , появляется окно **Choose Configuration Folder** , позволяющее выбрать спецификатор папки layout, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка layout с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового Layout-файла он будет открыт в Layout-редакторе ADT-плагина, обеспечивающим визуальное редактирование GUI-интерфейса.

Тип ресурса Values

Для создания ресурсного файла Android-приложения в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File** или **Android XML Values File** , нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Values** .

Поле **File:** мастера создания ресурсного файла предлагает ввести имя нового файла XML-описания значений приложения, который затем с расширением. xml появится в каталоге res/values Android-проекта и будет доступен в Java-коде с помощью сгенерированного R-класса или в XML-коде с помощью ссылки на имя ресурса.

Раздел **Root Element:** мастера создания ресурсного файла показывает, что корневым элементом XML-файла служит тэг <resources>.

После ввода имени нового ресурсного файла и нажатия кнопки **Next** , появляется окно **Choose Configuration Folder** , позволяющее выбрать спецификатор папки values, обеспечивающий поддержку специфической конфигурации Android-устройства,

в соответствии с которой папка values с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового ресурсного файла он будет открыт в редакторе Values-файлов ADT-плагина. Кнопка **Add** вкладки **Resources** редактора Values-файлов обеспечивает добавление в корневой тэг <resources> ресурсного файла тэгов <color> (элемент **Color**), <dimen> (элемент **Dimension**), <drawable> (элемент **Drawable**), <integer-array> (элемент **Integer Array**), <item> (элемент **Item**), <plurals> (элемент **Quantity Strings (Plurals)**), <string> (элемент **String**), <string-array> (элемент **String Array**), <style> (элемент **Style/Theme**).

Тэг <color> (элемент **Color**) определяет цвет, используя синтаксис <color name=«color_name»> hex_color </color>, где hex_color – значение цвета в формате #RGB, #ARGB, #RRGGBB, #AARRGGBB. При добавлении элемента **Color** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Color** с полем **Name** (определяет значение атрибута name) и полем **Value** (определяет значение цвета). Тэг <color> как правило используется в ресурсном файле с именем colors. xml каталога res/values. Именованный ресурс цвета может использоваться для определения цвета различных объектов, таких как Drawable или TextView. Созданный ресурс доступен в

Java-коде с помощью сгенерированного класса R.color.color_name, или в XML-коде – с помощью ссылки @ [package:] color/color_name.

Тэг <dimen> (элемент **Dimension**) определяет величину измерения, используя синтаксис <dimen name=«dimension_name»> dimension </dimen>, где dimension – значение в формате dp (независимый от плотности пиксель), sp (независимый от масштаба пиксель), pt (точка, 1/72 дюйма), px (пиксель), mm (миллиметр), in (дюйм). При добавлении элемента **Dimension** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Dimension** с полем **Name** (определяет значение атрибута name) и полем **Value** (определяет значение). Созданный ресурс доступен в

Java-коде с помощью сгенерированного класса R. dimen. dimension_name, или в XML-коде – с помощью ссылки @ [package:] dimen/dimension_name.

Тэг <drawable> (элемент **Drawable**) обеспечивает создание объекта android.graphics.drawable.PaintDrawable, представляющего прямоугольник, заполненный цветом, используя синтаксис <drawable name=color_name> color_value </drawable>, где color_value – значение цвета в формате #RGB, #ARGB, #RRGGBB, #AARRGGBB. При добавлении элемента **Drawable** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Drawable** с полем **Name** (определяет значение атрибута name) и полем **Value** (определяет значение). Созданный ресурс доступен в Java-коде с помощью сгенерированного класса R. drawable. drawable_name, или в XML-коде – с помощью ссылки @ [package:] drawable/drawable_name.

Тэг <integer-array> (элемент **Integer Array**) определяет массив целых чисел, используя синтаксис <integer-array name=«integer_array_name»> <item> integer </item> </integer-array>. При добавлении элемента **Integer Array** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Integer Array** с полем **Name** – определяет значение атрибута name. Созданный ресурс доступен в Java-коде с помощью сгенерированного класса R.array.integer_array_name или в XML-коде с помощью ссылки @ [package:] array/integer_array_name.

Тэг <item> (элемент **Item**) позволяет определить различного типа константы, для их последующего использования в Java-коде с помощью сгенерированного класса R. [тип константы]. [имя константы]. При добавлении элемента **Item** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Item** с полями и списками:

Поле Name – определяет значение атрибута name, указывающего имя константы.

Список Type – определяет значение атрибута type, указывающего тип константы.

Поле Format – определяет значение атрибута format, указывающего формат значения константы.

Поле Value – определяет значение константы.

Тэг <plurals> (элемент **Quantity Strings (Plurals)**) обеспечивает строки для правильного склонения разного количества ключа ресурса. При добавлении элемента **Quantity Strings (Plurals)** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Quantity Strings (Plurals)** с полем **Name** , определяющим значение атрибута name – ключ ресурса. Для созданного элемента кнопка **Add** позволяет добавлять дочерние элементы **Item** – тэги <item> содержащие строки ресурса. При этом раздел **Attributes for Item** содержит список **Quantity** , с помощью которого выбирается количество ресурса, которому должна соответствовать строка (атрибут quantity). Созданный ресурс доступен в Java-коде с помощью метода getQuantityString (int id, int quantity) класса android.content.res.Resources.

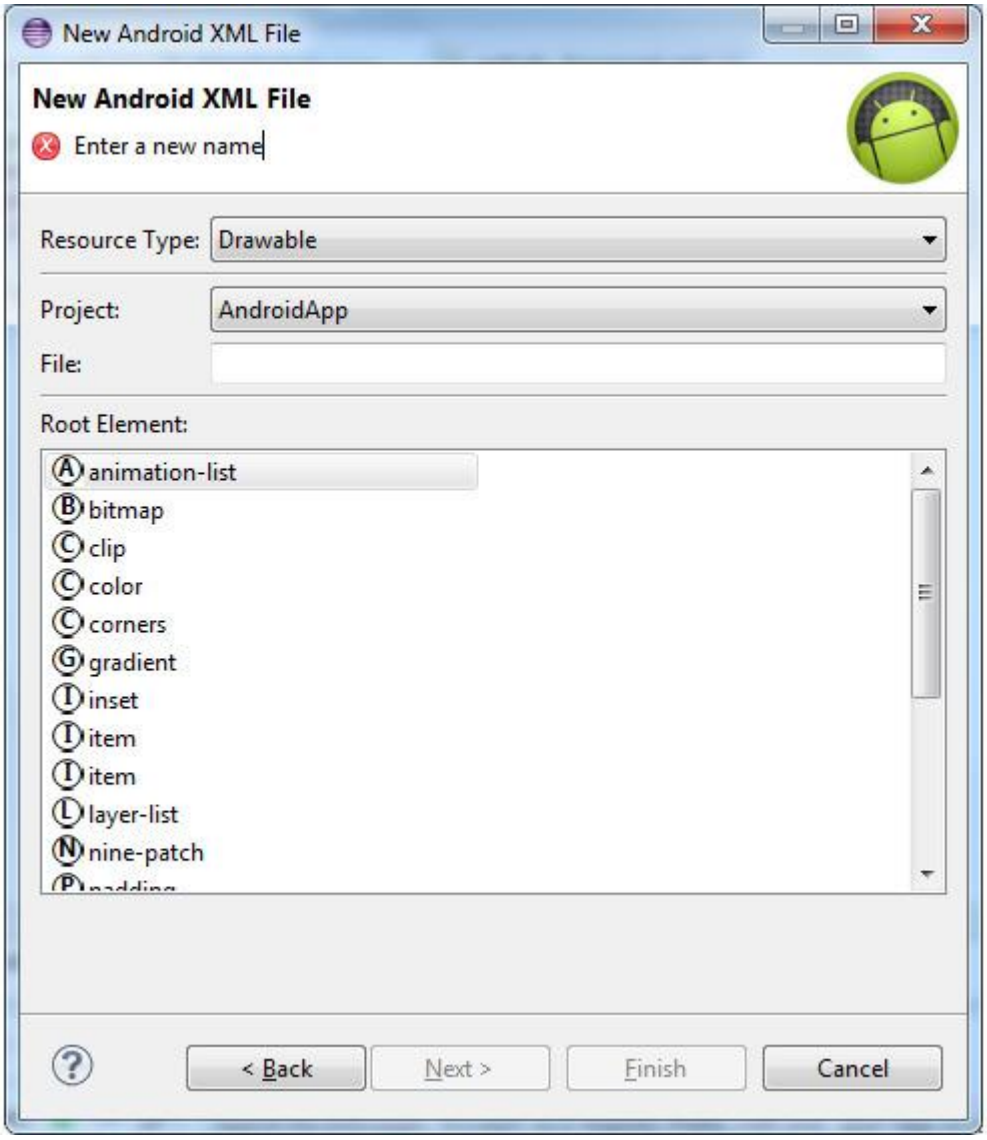
Тэг <string> (элемент **String**) определяет именованную строку, используя синтаксис <string name=«string_name»> text_string </string>. При добавлении элемента **String** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for String** с полем **Name** (определяет значение атрибута name) и полем **Value** (определяет строку). Созданный строковый ресурс доступен в Java-коде с помощью сгенерированного класса R.string.string_name, или в XML-коде – с помощью ссылки @string/string_name.

Тэг <string-array> (элемент **String Array**) определяет массив строк, используя синтаксис <string-array name=«string_array_name»> <item> text_string </item> </string-array>. При добавлении элемента **String Array** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for String Array** с полем **Name** – определяет значение атрибута name. Созданный ресурс доступен в Java-коде с помощью сгенерированного класса R.array.string_array_name или в XML-коде с помощью ссылки @ [package:] array/string_array_name.

Тэг <style> (элемент **Style/Theme**) позволяет определить стиль для индивидуального View-компонента, для Activity-компонента и для приложения в целом, используя синтаксис <style name=«style_name» parent="@ [package:] style/style_to_inherit»> <item name=» [package:] style_property_name»> style_value </item> </style>. При добавлении элемента **Style/Theme** кнопкой **Add** , во вкладке **Resources** появляется раздел **Attributes for Style/Theme** с полем **Name** – определяет значение атрибута name и полем Parent – определяет значение атрибута parent. Созданный стиль доступен в Java-коде с помощью сгенерированного класса R.style.style_name или в XML-коде с помощью ссылки @ [package:] style/style_name.

Тип ресурса Drawable

Для создания графического ресурса Android-приложения в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File** , нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Drawable** .



Поле **File:** мастера создания графического ресурса предлагает ввести имя нового XML-файла, который затем появится в каталоге res/drawable Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.drawable.filename или в XML-коде с помощью ссылки @ [package:] drawable/filename.

Раздел **Root Element:** мастера создания графического ресурса предлагает выбрать корневой XML-элемент ресурса:

<animation-list> – обеспечивает покадровую анимацию, каждый кадр которой представлен Drawable-объектом, определяемым дочерним тэгом <item>. Тэг <animation-list> имеет атрибуты android: visible (true/false, определяет видимость объекта), android: variablePadding (true/false, определяет изменяемость отступов), android: oneshot (true/false, определяет одноразовую или повторяющуюся анимацию). Тэг <item> имеет атрибуты android: drawable (ссылка на Drawable-объект кадра) и android: duration (продолжительность кадра в миллисекундах).

<bitmap> – обертывает PNG, JPG, GIF изображение, имеет атрибуты android: src (ссылка на обертываемое изображение), android: antialias (true/false, сглаживание изображения), android: filter (true/false, сглаживание при масштабировании изображения), android: dither (true/false, сглаживание переходов при несовпадении конфигураций изображения и экрана), android: gravity (выравнивание изображения, возможные значения top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill, clip_vertical, clip_horizontal), android: tileMode (режим повторения изображения для заполнения им контейнера, возможные значения disabled, clamp, repeat, mirror).

<clip> – накладывает маску на Drawable-объект, основываясь на Level-значении и используя атрибуты android: clipOrientation (ориентация маски, возможные значения horizontal, vertical), android: gravity (выравнивание маски, возможные значения top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center, fill, clip_vertical, clip_horizontal), android: drawable (ссылка на исходный Drawable-объект).

<color> – создает прямоугольник, заполненный цветом, используя атрибут android: color (цвет заполнения).

<corners> – дочерний тэг тэга <shape>, определяет закругленные углы прямоугольника с помощью атрибутов android: radius (радиус всех 4 углов как ресурс <dimen>), android: topLeftRadius (радиус верхнего левого угла как ресурс <dimen>), android: topRightRadius (радиус верхнего правого угла как ресурс <dimen>), android: bottomLeftRadius (радиус нижнего левого угла как ресурс <dimen>), android: bottomRightRadius (радиус нижнего правого угла как ресурс <dimen>).

<gradient> – дочерний тэг тэга <shape>, определяет градиентную заливку геометрической формы с помощью атрибутов android: angle (угол градиента в градусах), android: centerX (относительный центр градиента по оси X, от 0 до 1.0), android: centerY (относительный центр градиента по оси Y, от 0 до 1.0), android: centerColor (промежуточный цвет градиента), android: endColor (конечный цвет градиента), android: gradientRadius (радиус для радиального градиента), android: startColor (начальный цвет градиента), android: type (тип градиента, возможные значения linear, radial, sweep), android: useLevel (true/false, если геометрическая форма участвует в <level-list>, тогда если true – количество отображений градиента зависит от уровня формы).

<inset> – вставляет Drawable-объект с отступами, используя атрибуты android: drawable (ссылка на вставляемый Drawable-объект), android: insetTop (верхний отступ как ресурс <dimen>), android: insetRight (правый отступ как ресурс <dimen>), android: insetBottom (нижний отступ как ресурс <dimen>), android: insetLeft (левый отступ как ресурс <dimen>).

<item> – дочерний тэг тэгов <animation-list>, <layer-list>, <level-list>, <selector>.

<layer-list> – стек Drawable-объектов, определяемых дочерними элементами <item> с атрибутами android: drawable (ссылка на Drawable-объект), android: id (идентификатор в форме @+id/name), android: top (верхний отступ в пикселях), android: right (правый отступ в пикселях), android: bottom (нижний отступ в пикселях), android: left (левый отступ в пикселях).

<nine-patch> – обертывает 9PNG-изображение с изменяющимися размерами, создаваемое инструментом draw9patch SDK Tools из PNG-изображения, используя атрибуты android: src (ссылка на 9PNG-изображение), android: dither (true/false, сглаживание переходов при несовпадении конфигураций изображения и экрана).

<padding> – дочерний тэг тэга <shape>, определяет отступы для содержимого формы с помощью атрибутов android: top (верхний отступ как ресурс <dimen>), android: right (правый отступ как ресурс <dimen>), android: bottom (нижний отступ как ресурс <dimen>), android: left (левый отступ как ресурс <dimen>).

<rotate> – поворачивает Drawable-объект, основываясь на Level-значении и используя атрибуты android: visible (true/false, определяет видимость объекта), android: fromDegrees (первоначальный угол вращения), android: toDegrees (конечный угол вращения), android: pivotX (центр вращения по оси X в процентном соотношении к ширине объекта), android: pivotY (центр вращения по оси Y в процентном соотношении к высоте объекта), android: drawable (ссылка на вращаемый объект).

<scale> – масштабирует Drawable-объект, основываясь на Level-значении и используя атрибуты android: scaleWidth (масштабирование ширины в процентах), android:

scaleHeight (масштабирование высоты в процентах), android: scaleGravity (выравнивание после масштабирования), android: drawable (ссылка на первоначальный Drawable-объект), android: useIntrinsicSizeAsMinimum (true/false, определяет использование собственных размеров объекта как минимальных).

<selector> – содержит набор Drawable-объектов для различных состояний View-компонента. Набор Drawable-объектов описывается дочерними тэгами <item>, которые связываются с определенными состояниями с помощью атрибутов android: drawable (ссылка на Drawable-объект), android: state_pressed (true/false), android: state_focused (true/false), android: state_hovered (true/false), android: state_selected (true/false), android: state_checkable (true/false), android: state_checked (true/false), android: state_enabled (true/false), android: state_activated (true/false), android: state_window_focused (true/false).

<shape> – описывает геометрическую форму, используя атрибут android: shape (возможные значения rectangle, oval, line, ring) и дочерние тэги <corners>, <gradient>, <padding>, <size>, <solid>, <stroke>.

<size> – дочерний тэг тэга <shape>, определяет размеры геометрической формы, используя атрибуты android: height (высота как ресурс <dimen>), android: width (ширина как ресурс <dimen>).

<solid> – дочерний тэг тэга <shape>, определяет цвет заполнения формы с помощью атрибута android: color.

<stroke> – дочерний тэг тэга <shape>, определяет контур геометрической формы, используя атрибуты android: width (ширина контура как ресурс <dimen>), android: color (цвет контура), android: dashGap (расстояние между пунктирами как ресурс <dimen>), android: dashWidth (ширина пунктира как ресурс <dimen>).

После ввода имени нового графического ресурса, выбора его корневого элемента и нажатия кнопки **Next** , появляется окно **Choose Configuration Folder** , позволяющее выбрать спецификатор папки drawable, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка drawable с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового графического ресурса он будет открыт в XML-редакторе кода.

Тип ресурса Menu

Android-платформа обеспечивает создание трех видов меню для Android-приложения – меню опций, которое открывается при нажатии кнопки **MENU** устройства или для Android-версии 3.0 и выше элементы которого могут быть помещены в ActionBar-панель, контекстное меню View-компонента и подменю элемента меню.

Все три вида меню могут быть созданы программным способом или на основе XML-описания. Создание меню на основе XML-описания является предпочтительным способом, так как позволяет разделить содержимое меню и его бизнес-логику. После создания XML-описания меню для меню опций необходимо в классе Activity-компонента переопределить метод onCreateOptionsMenu (), в котором необходимо создать программный объект из XML-описания, используя метод MenuInflater.inflate (), а также переопределить метод onOptionsItemSelected (), обрабатывающий выбор элемента меню.

Для контекстного меню необходимо в методе onCreate () Activity-компонента зарегистрировать View-компонент как имеющий контекстное меню с помощью метода registerForContextMenu (), переопределить метод onCreateContextMenu (), в котором необходимо создать программный объект из XML-описания, используя метод MenuInflater.inflate (), а также переопределить метод onContextItemSelected (), обрабатывающий выбор элемента меню. Подменю элемента меню определяется простым вложением его XML-описания в тэг элемента меню.

Для создания XML-описания меню Android-приложения в окне **Project Explorer**

нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File** , нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Menu** .

Поле **File:** мастера создания Menu-файла предлагает ввести имя нового файла XML-описания меню, который затем с расширением. xml появится в каталоге res/menu Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.menu. [имя Menu-файла] или в XML-коде с помощью ссылки @ [package:] menu. [имя Menu-файла].

Раздел **Root Element:** мастера создания Menu-файла показывает, что корневым элементом XML-файла служит тэг <menu>.

После ввода имени нового Menu-файла и нажатия кнопки **Next** , появляется окно **Choose Configuration Folder** , позволяющее выбрать спецификатор папки menu, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка menu с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового Menu-файла он будет открыт в редакторе ADT-плагина, обеспечивающим визуальное редактирование XML-описания меню. Кнопка **Add** вкладки **Layout** редактора Menu-файла обеспечивает добавление в корневой тэг <menu> тэги <group> (элемент **Group**) и <item> (элемент **Item**).

Тэг <item> описывает элемент меню, может быть дочерним тэгом тэга <menu> и <group> и иметь в качестве дочернего тэг <menu>, представляющий подменю (элемент **Sub-Menu**). Тэг <item> имеет следующие атрибуты:

android: id – идентификатор элемента в виде @+id/name.

android: menuCategory – категория элемента меню, определяющая его приоритет (номер в списке) при отображении, возможные значения container, system, secondary, alternative.

android: orderInCategory – номер элемента в списке отображения в пределах категории.

android: title – текстовая метка элемента.

android: titleCondensed – укороченная текстовая метка элемента.

android: icon – ссылка на Drawable-ресурс, представляющий значок элемента, который отображается для первых 6 элементов меню опций.

android: alphabeticShortcut – символ быстрого вызова элемента.

android: numericShortcut – цифра быстрого вызова элемента.

android: checkable – если true, тогда элемент содержит флажок выбора.

android: checked – если true, тогда флажок элемента отмечен по умолчанию.

android: visible – если true, тогда элемент видим.

android: enabled – если true, тогда элемент доступен.

android: onClick – имя метода, вызываемого при нажатии элемента.

android: showAsAction – определяет как элемент отображается в ActionBar-панели, возможные значения ifRoom (отображается при наличии места в панели), never (не отображается), withText (отображается с меткой), always (всегда отображается), collapseActionView (с элементом связан разворачивающийся View-компонент).

android: actionLayout – ссылка на Layout-файл, описывающий View-компонент элемента ActionBar-панели.

android: actionViewClass – имя класса View-компонента элемента ActionBar-панели.

android: actionProviderClass – имя ActionProvider-класса, связанного с элементом ActionBar-панели.

Тэг <group> позволяет сгруппировать элементы меню так, что для них всех одновременно можно регулировать видимость, доступность и отображение флажка выбора. Тэг <group> имеет следующие атрибуты:

android: id – идентификатор группы в виде @+id/name.

android: menuCategory – категория группы элементов меню, определяющая ее приоритет (номер в списке) при отображении, возможные значения container, system, secondary, alternative.

android: orderInCategory – номер группы в списке отображения в пределах категории.

android: checkableBehavior – тип группировки элементов, возможные значения none (элементы не отображают флажок выбора), all (элементы группируются как флажки checkbox), single (элементы группируются как переключатели radio button).

android: visible – видимость элементов группы, true/false.

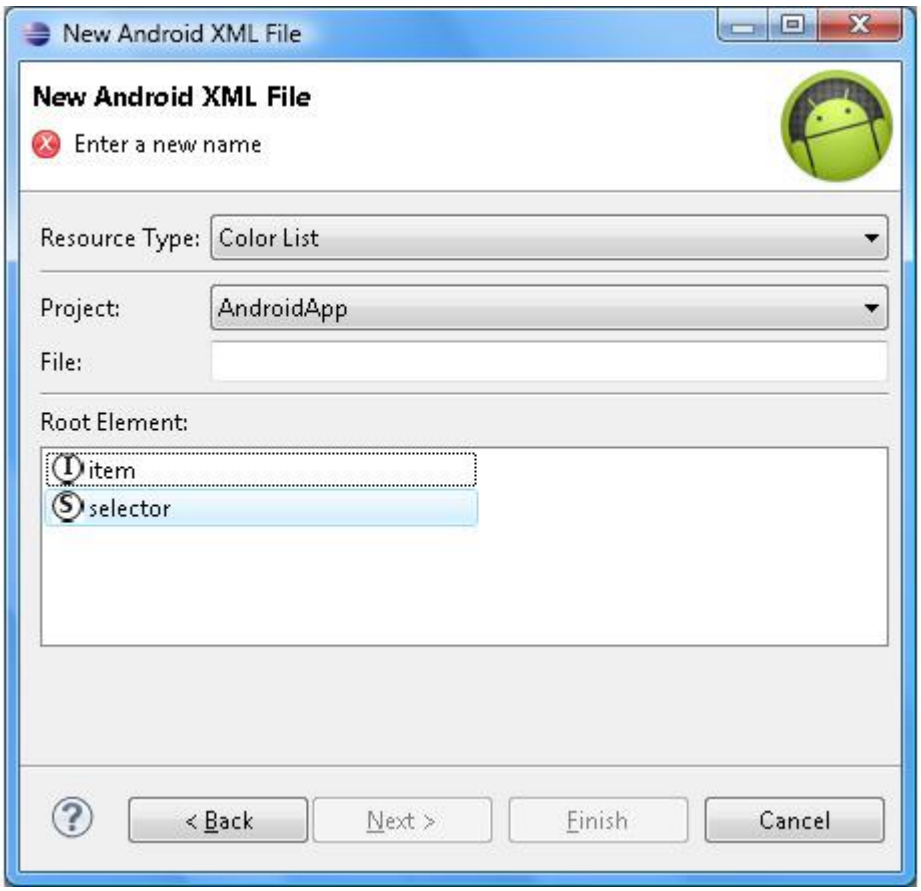
android: enabled – доступность элементов группы, true/false.

При выборе элементов **Group** и **Item** кнопкой **Add** , во вкладке **Layout** появляются разделы **Attributes for Group** и **Attributes for Item** с полями, позволяющими определить атрибуты тэгов <group> и <item>.

Тип ресурса Color List

Данный тип ресурса является аналогом Drawable-ресурса State List с корневым элементом <selector>. Разница состоит в том, что ресурс Drawable State List определяет набор изображений для представления различных состояний View-компонента, а ресурс Color State List определяет набор цветов для представления различных состояний View-компонента.

Для создания ресурса Color State List в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File** , нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Color List** .



Поле **File:** мастера создания ресурса Color State List предлагает ввести имя нового файла XML-описания набора цветов различных состояний View-компонента, который затем с расширением. xml появится в каталоге res/color Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.color.filename или в XML-коде с помощью ссылки @ [package:] color/filename.

Раздел **Root Element:** мастера отображает элементы **item** и **selector** , представляющие тэги <item> и <selector> соответственно, при этом тэг <selector> является корневым тэгом XML-файла ресурса Color State List и поэтому в разделе **Root Element:** необходимо выбрать элемент **selector** .

Тэг <selector> может содержать один или несколько тэгов <item>, определяющих цвета для различных состояний View-объекта, используя атрибуты:

android: color – цвет состояния в формате #RGB, #ARGB, #RRGGBB, #AARRGGBB.

android: state_pressed – состояние нажатия, true/false.

android: state_focused – компонент в фокусе, true/false.

android: state_selected – компонент выбран, true/false.

android: state_checkable – компонент содержит флажок выбора, true/false.

android: state_checked – флажок компонента отмечен, true/false.

android: state_enabled – компонент доступен, true/false.

android: state_window_focused – окно приложения на переднем плане, true/false.

После ввода имени нового XML-файла ресурса Color State List, выбора элемента **selector** и нажатия кнопки **Next** , появляется окно **Choose Configuration Folder** , позволяющее выбрать спецификатор папки color, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка color с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового XML-файла ресурса Color State List он будет открыт в XML-редакторе кода.

Тип ресурса Property Animation и Tween Animation

Ресурс Property Animation описывает изменение свойства объекта в течение определенного промежутка времени. Анимация свойств объектов представлена в версиях Android-платформы, начиная с версии 3.0.

Для запуска анимации свойства объекта на основе XML-описания необходимо создать из XML-ресурса Property Animation объект android.animation.AnimatorSet, android.animation. ObjectAnimator или android.animation.ValueAnimator, используя статический метод android.animation.AnimatorInflater. loadAnimator (), и связать анимацию с объектом с помощью метода setTarget () суперкласса android.animation.Animator, после чего запустить анимацию методом start () суперкласса Animator.

Для создания ресурса Property Animation в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File**, нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Property Animation**.

Поле **File:** мастера создания ресурса Property Animation предлагает ввести имя нового файла XML-описания анимации, который затем с расширением. xml появится в каталоге res/animator Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.animator.filename или в XML-коде с помощью ссылки @ [package:] animator/filename.

Раздел **Root Element:** мастера отображает элементы **animator**, **objectAnimator** и **set**, представляющие тэги <animator>, <objectAnimator> и <set> соответственно, при этом каждый из них может служить единственным корневым тэгом XML-файла ресурса Property Animation.

Тэг <animator> представляет класс ValueAnimator и описывает анимацию значения типа float, int или color в течение определенного промежутка времени, используя атрибуты:

android: duration – время анимации в миллисекундах, по умолчанию 300ms.

android: valueFrom – начальное значение.

android: valueTo – конечное значение.

android: startOffset – задержка анимации в миллисекундах.

android: repeatCount – количество циклов анимации, значение -1 означает бесконечную анимацию.

android: repeatMode – режим повторения анимации, возможные значения repeat и reverse.

android: valueType – тип значения для анимации, для значения типа color не указывается, возможные значения intType и floatType (по умолчанию).

Тэг <objectAnimator> представляет класс ObjectAnimator и описывает анимацию значения свойства объекта в течение определенного промежутка времени, используя атрибуты:

android: propertyName – имя свойства объекта, например android: propertyName=«alpha».

android: duration – время анимации в миллисекундах, по умолчанию 300ms.

android: valueFrom – начальное значение свойства.

android: valueTo – конечное значение свойства.

android: startOffset – задержка анимации в миллисекундах.

android: repeatCount – количество циклов анимации, значение -1 означает бесконечную анимацию.

android: repeatMode – режим повторения анимации, возможные значения repeat и reverse.

android: valueType – тип значения для анимации, для значения типа color не указывается, возможные значения intType и floatType (по умолчанию).

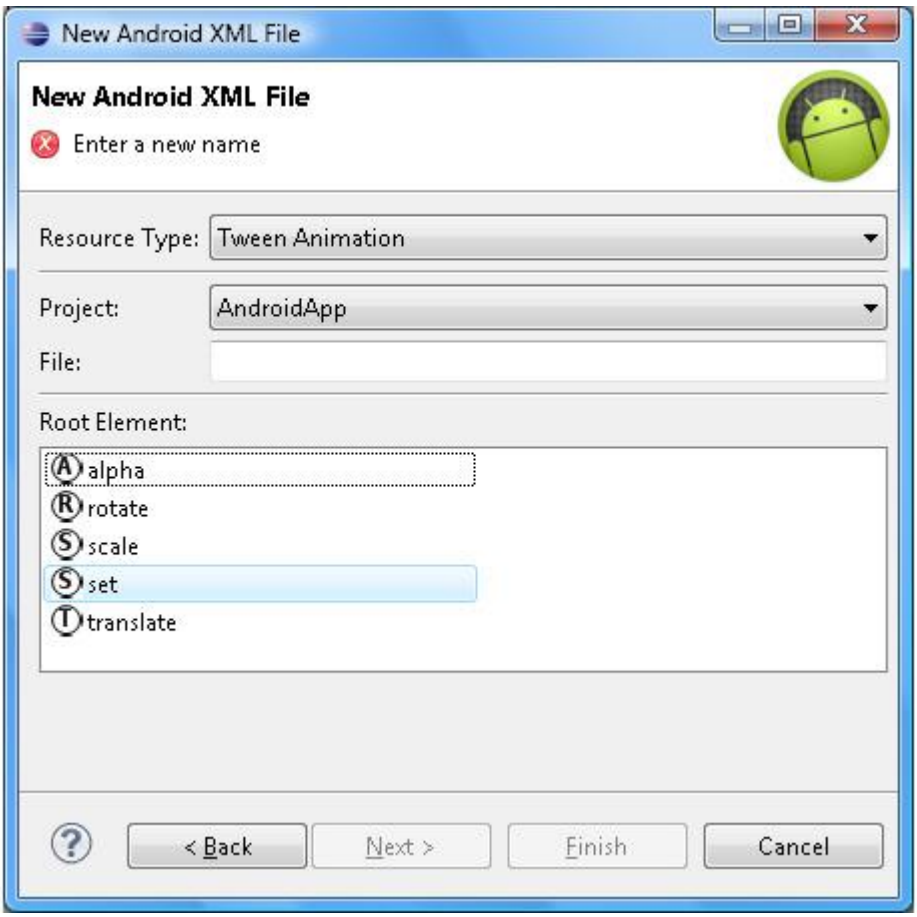
Тэг <set> представляет класс AnimatorSet и обеспечивает группировку анимаций, используя атрибут android: ordering с возможными значениями together (анимации проигрываются параллельно) и sequentially (анимации проигрываются последовательно).

После ввода имени нового XML-файла ресурса Property Animation, выбора корневого элемента и нажатия кнопки **Next**, появляется окно **Choose Configuration Folder**, позволяющее выбрать спецификатор папки animator, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка animator с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового XML-файла ресурса Property Animation он будет открыт в XML-редакторе кода.

Ресурс Tween Animation описывает анимацию вращения, исчезновения, перемещения и масштабирования View-компонента. Для запуска анимации View-компонента на основе XML-описания необходимо создать из XML-ресурса Tween Animation объект android.view.animation.Animation, используя статический метод android.view.animation.AnimationUtils. loadAnimation () и запустить анимацию методом startAnimation (Animation animation) суперкласса android.view.View.

Для создания ресурса Tween Animation в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File**, нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Tween Animation.**



Поле **File:** мастера создания ресурса Tween Animation предлагает ввести имя нового файла XML-описания анимации, который затем с расширением. xml появится в каталоге res/anim Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.anim.filename или в XML-коде с помощью ссылки @ [package:] anim/filename.

Раздел **Root Element:** мастера отображает элементы **alpha**, **rotate**, **scale**, **set** и **translate**, представляющие тэги <alpha>, <rotate>, <scale>, <set> и <translate> соответственно, при этом каждый из них может служить единственным корневым тэгом XML-файла ресурса Tween Animation.

Вышеупомянутые тэги имеют общие атрибуты, унаследованные от суперкласса android.view.animation.Animation:

android: detachWallpaper – если true, тогда обои не анимируются вместе с окном.

android: duration – продолжительность анимации в миллисекундах.

android: fillAfter – если true, тогда преобразование применяется после окончания анимации.

android: fillBefore – если true, тогда преобразование применяется перед началом анимации.

android: fillEnabled – если true, тогда значение fillBefore учитывается.

android: interpolator – указывает объект android.view.animation.Interpolator, отвечающий за определение скорости анимации.

android: repeatCount – количество циклов анимации.

android: repeatMode – режим повторения анимации, возможные значения repeat и reverse.

android: startOffset – задержка анимации в миллисекундах.

android: zAdjustment – определяет поведение компонента по оси Z, возможные значения normal (позиция в стеке сохраняется), top (компонент в течение анимации находится на вершине стека), bottom (компонент в течение анимации находится внизу стека).

Тэг <alpha> представляет класс AlphaAnimation и описывает анимацию значения прозрачности в течение определенного промежутка времени, используя атрибуты:

android: fromAlpha – начальное значение прозрачности.

android: toAlpha – конечное значение прозрачности.

Тэг <rotate> представляет класс RotateAnimation и описывает вращение вокруг оси, используя атрибуты:

android: fromDegrees – начальный угол вращения.

android: toDegrees – конечный угол вращения.

android: pivotX и android: pivotY – координаты оси вращения от левого края компонента в пикселях или процентах.

Тэг <scale> представляет класс ScaleAnimation и описывает масштабирование компонента, используя атрибуты:

android: fromXScale – начальный коэффициент масштабирования по оси X.

android: toXScale – конечный коэффициент масштабирования по оси X.

android: fromYScale – начальный коэффициент масштабирования по оси Y.

android: toYScale – конечный коэффициент масштабирования по оси Y.

android: pivotX и android: pivotY – координаты центра масштабирования.

Тэг <translate> представляет класс TranslateAnimation и описывает движение компонента, используя атрибуты:

android: fromXDelta – начальная позиция по оси X в пикселях или процентах.

android: toXDelta – конечная позиция по оси X в пикселях или процентах.

android: fromYDelta – начальная позиция по оси Y в пикселях или процентах.

android: toYDelta – конечная позиция по оси Y в пикселях или процентах.

Тэг <set> представляет класс AnimationSet и обеспечивает группировку анимаций в параллельную анимацию, используя атрибуты android: interpolator – ссылка на объект android.view.animation.Interpolator, отвечающий за определение скорости анимации, и android: shareInterpolator – если true, тогда интерполятор является общим для дочерних анимаций.

Начиная с версии 4.0 Android-платформы атрибуты duration, repeatMode, fillBefore, fillAfter, определенные в тэге <set> применяются к дочерним элементам, атрибуты repeatCount и fillEnabled игнорируются, атрибуты startOffset и shareInterpolator применяются к объекту AnimationSet, до версии 4.0 Android-платформы данные атрибуты игнорируются.

После ввода имени нового XML-файла ресурса Tween Animation, выбора корневого элемента и нажатия кнопки **Next**, появляется окно **Choose Configuration Folder**, позволяющее выбрать спецификатор папки anim, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка anim с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового XML-файла ресурса Tween Animation он будет открыт в XML-редакторе кода.

Тип ресурса AppWidgetProvider

Ресурс AppWidgetProvider содержит метаданные для приложения App Widget, представляющего собой мини-приложение, которое можно разместить на рабочем экране Home Screen устройства, используя опцию **Выбор виджета** рабочего экрана.

Для создания приложения App Widget необходимо:

Создать ресурс AppWidgetProvider.

Создать Layout-файл приложения App Widget.

Создать класс, расширяющий класс android. appwidget. AppWidgetProvider, который обеспечивает взаимодействие с приложением App Widget.

Зарегистрировать AppWidgetProvider-класс и AppWidgetProvider-ресурс в файле манифеста AndroidManifest. xml Android-приложения.

Для создания ресурса AppWidgetProvider в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File**, нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **AppWidgetProvider**.

Поле **File:** мастера создания ресурса AppWidgetProvider предлагает ввести имя нового XML-файла метаданных приложения App Widget, который затем с расширением. xml появится в каталоге res/xml Android-проекта и будет доступен в XML-коде с помощью ссылки @ [package:] xml/filename.

Раздел **Root Element:** мастера создания AppWidgetProvider-ресурса показывает, что корневым элементом XML-файла служит тэг <appwidget-provider>.

После ввода имени нового XML-файла AppWidgetProvider-ресурса и нажатия кнопки **Next**, появляется окно **Choose Configuration Folder**, позволяющее выбрать спецификатор папки xml, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка xml с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового AppWidgetProvider-ресурса, он будет открыт в редакторе ADT-плагина, обеспечивающим визуальное редактирование атрибутов тэга <appwidget-provider>.

Для определения атрибутов тэга <appwidget-provider> раздел **Attributes for AppWidget Provider**вкладки **Structure** ADT-редактора содержит следующие поля и списки:

Min width – определяет атрибут android: minWidth, указывающий минимальную ширину окна приложения App Widget в формате px, dp, sp, in, mm.

Min height – определяет атрибут android: minHeight, указывающий минимальную высоту окна приложения App Widget в формате px, dp, sp, in, mm.

Min resize width – определяет атрибут android: minResizeWidth, указывающий для Android-платформы версии 3.1 и выше минимальную ширину в формате px, dp, sp, in, mm, до которой пользователь может уменьшить ширину окна приложения App Widget.

Min resize height – определяет атрибут android: minResizeHeight, указывающий для Android-платформы версии 3.1 и выше минимальную высоту в формате px, dp, sp, in, mm, до которой пользователь может уменьшить высоту окна приложения App Widget.

Update period millis – определяет атрибут android: updatePeriodMillis, указывающий период в миллисекундах (но не чаше чем раз в час) между автоматическими вызовами метода onUpdate () AppWidgetProvider-класса.

Initial layout – определяет атрибут android: initialLayout, указывающий ссылку на Layout-файл приложения App Widget.

Configure – определяет атрибут android: configure, указывающий Activity-компонент, который загружается при добавлении пользователем приложения App Widget и который позволяет пользователю настроить приложение App Widget.

Preview image – определяет атрибут android: previewImage, указывающий для Android-платформы версии 3.0 и выше ссылку на значок, который дает пользователю представление о том, как выглядит приложение App Widget.

Auto advance view id – определяет атрибут android: autoAdvanceViewId, указывающий для Android-платформы версии 3.0 и выше идентификатор View-компонента коллекции, элементы которой автоматически прокручиваются в слайд-шоу.

Resize mode – определяет атрибут android: resizeMode, указывающий для Android-платформы версии 3.1 и выше режим изменения пользователем размеров окна приложения App Widget, возможные значения horizontal, vertical, none.

Так как приложение App Widget работает не в своем собственном процессе, а в процессе компонента, в который приложение App Widget встроено (компонент Home Screen), иерархия View-компонентов приложения App Widget определяется объектом android.widget.RemoteViews, поддерживающим ограниченный набор View-компонентов. Поэтому Layout-файл приложения App Widget может содержать только компоненты FrameLayout, LinearLayout, RelativeLayout, AnalogClock, Button, Chronometer, ImageButton, ImageView, ProgressBar, TextView, ViewFlipper, ListView, GridView, StackView, AdapterViewFlipper.

В классе расширения android. appwidget. AppWidgetProvider как правило переопределяется метод onUpdate (), автоматически вызываемый всякий раз, когда приложение App Widget требует обновления. В методе onUpdate () на основе Layout-файла создается объект RemoteViews, позволяющий обновлять текст TextView-компонента и загружать Activity-компонент в ответ на щелчок пользователя на компоненте приложения App Widget. После работы с объектом RemoteViews приложение App Widget обновляется в методе onUpdate () с помощью метода updateAppWidget () класса android. appwidget. AppWidgetManager.

Так как класс AppWidgetProvider расширяет класс android.content.BroadcastReceiver, в файле манифеста AndroidManifest. xml приложение App Widget описывается тэгом <receiver>, который при этом содержит Intent-фильтр, указывающий действие обновления, и тэг <meta-data> с ссылкой на AppWidgetProvider-ресурс:

<receiver android: name=«MyAppWidgetProvider»>
<intent-filter>
<action android:name="android.appwidget.action. APPWIDGET_UPDATE» />
</intent-filter>
<meta-data android:name="android.appwidget.provider»
android: resource="@xml/my_appwidget_info» />
</receiver>

Тип ресурса Preference

Система Android предоставляет механизм хранения и редактирования настроек (предпочтений) Android-приложения в виде пар имя-значение файла [имя пакета] _preferences. xml, расположенного в папке приложения каталога /data/data/ устройства.

Для работы с предпочтениями Android-приложение должно иметь:

Preference-ресурс, определяющий GUI-интерфейс редактирования предпочтений.

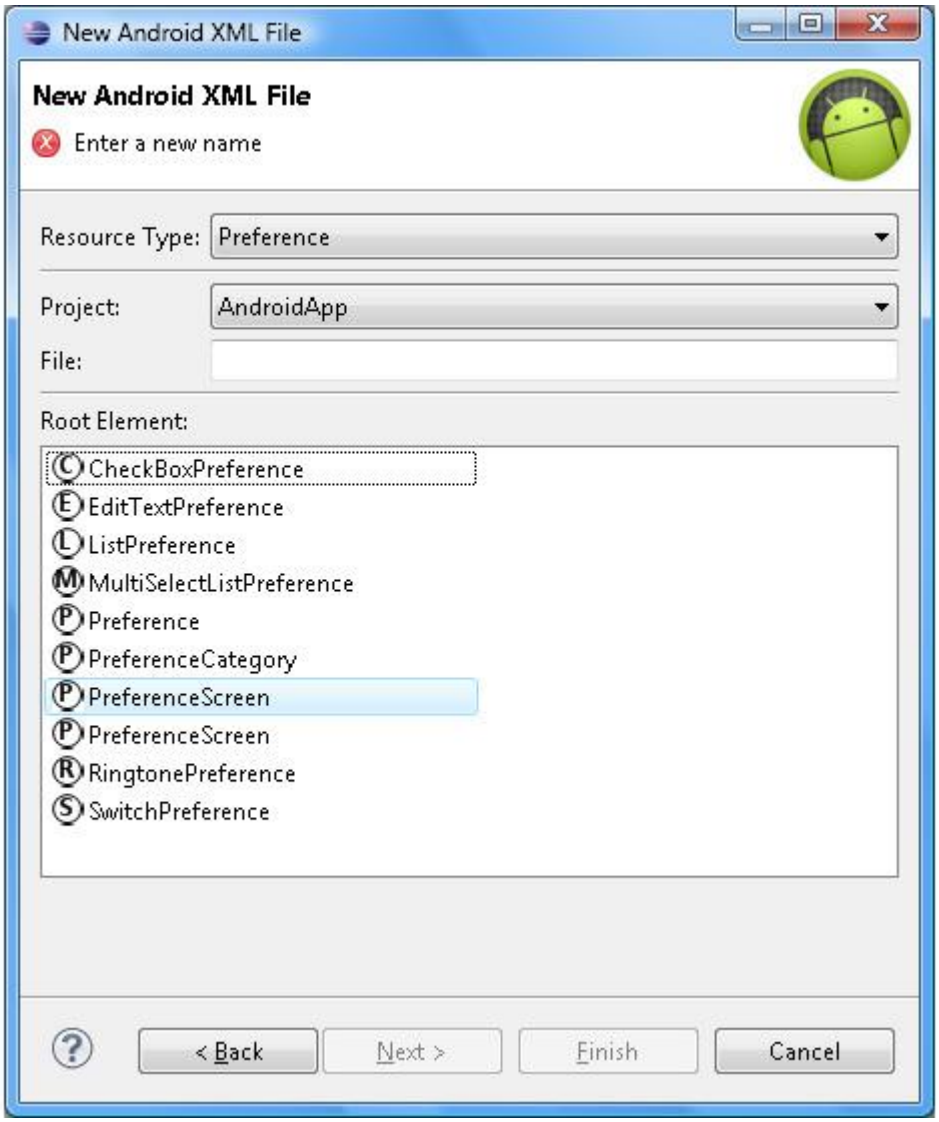
Activity-компонент, класс которого расширяет класс android.preference.PreferenceActivity и отвечает за загрузку Preference-ресурса, используя метод addPreferencesFromResource ().

Файл манифеста AndroidManifest. xml, в котором PreferenceActivity-компонент объявляется с помощью тэга <activity>.

Activity-компонент приложения, вызывающий PreferenceActivity-компонент методом startActivity ().

Activity-компонент приложения, считывающий предпочтения из XML-файла в объект android.content.SharedPreferences методом getSharedPreferences (), при этом методы класса SharedPreferences обеспечивают доступ к парам имя-значение предпочтений.

Для создания Preference-ресурса в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File**, нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Preference**.



Поле **File:** мастера создания ресурса Preference предлагает ввести имя нового XML-файла описания GUI-интерфейса предпочтений, который затем с расширением. xml появится в каталоге res/xml Android-проекта и будет доступен в Java-коде с помощью сгенерированного класса R.xml.filename.

Раздел **Root Element:** мастера отображает элементы **CheckBoxPreference**, **EditTextPreference**, **ListPreference**, **MultiSelectListPreference**, **Preference**, **PreferenceCategory**, **PreferenceScreen**, **RingtonePreference** и **SwitchPreference**, представляющие тэги <CheckBoxPreference>, <EditTextPreference>, <ListPreference>, <MultiSelectListPreference>, <Preference>, <PreferenceCategory>, <PreferenceScreen>, <RingtonePreference> и <SwitchPreference> соответственно, при этом тэг <PreferenceScreen> служит корневым тэгом XML-файла ресурса Preference.

После ввода имени нового XML-файла Preference-ресурса, выбора элемента **PreferenceScreen**и нажатия кнопки **Next**, появляется окно **Choose Configuration Folder**, позволяющее выбрать спецификатор папки xml, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка xml с нужным спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового файла Preference-ресурса он будет открыт в визуальном редакторе ADT-плагина.

Кнопка **Add** вкладки **Structure** визуального редактора обеспечивает добавление в корневой тэг <PreferenceScreen> файла Preference-ресурса тэгов <CheckBoxPreference> (элемент **CheckBoxPreference**), <EditTextPreference> (элемент **EditTextPreference**), <ListPreference> (элемент **ListPreference**), <MultiSelectListPreference> (элемент **MultiSelectListPreference**), <Preference> (элемент **Preference**), <PreferenceCategory> (элемент **PreferenceCategory**), <PreferenceScreen> (элемент **PreferenceScreen**), <RingtonePreference> (элемент **RingtonePreference**) и <SwitchPreference> (элемент **SwitchPreference**).

Раздел **Attributes from Preference** вкладки **Structure** редактора обеспечивает определение атрибутов, общих для тэгов <CheckBoxPreference>, <EditTextPreference>, <ListPreference>, <MultiSelectListPreference>, <Preference>, <PreferenceCategory>, <PreferenceScreen>, <RingtonePreference> и <SwitchPreference> с помощью следующих полей и списков:

Icon – определяет атрибут android: icon, указывающий ссылку на значок предпочтения.

Key – определяет атрибут android: key, указывающий имя для хранения значения предпочтения.

Title – определяет атрибут android: title, указывающий отображаемый заголовок предпочтения.

Summary – определяет атрибут android: summary, указывающий краткое описание предпочтения, отображаемое ниже заголовка.

Order – определяет атрибут android: order, указывающий порядок отображения предпочтения.

Fragment – определяет атрибут android: fragment, указывающий класс расширения android.preference.PreferenceFragment, который отвечает за отображение дополнительного списка предпочтений при выборе пользователем данного предпочтения (для версии Android 3.0 и выше).

Layout – определяет атрибут android: layout, указывающий ссылку на Layout-файл View-компонентов, отображаемых в предпочтении.

Widget layout – определяет атрибут android: widgetLayout, указывающий ссылку на Layout-файл компонента контроля предпочтения.

Enabled – атрибут android: enabled – если true (по умолчанию), тогда предпочтение доступно.

Selectable – атрибут android: selectable – если true (по умолчанию), тогда предпочтение доступно для выбора.

Dependency – определяет атрибут android: dependency, указывающий ключ android: key другого предпочтения, при этом если другое предпочтение недоступно, тогда недоступно и данное предпочтение.

Persistent – атрибут android: persistent – если true (по умолчанию), тогда значение предпочтения сохраняется в файле на устройстве.

Default value – определяет атрибут android: defaultValue, указывающий значение предпочтения по умолчанию.

Should disable view – атрибут android: shouldDisableView – если false, тогда при установке недоступности предпочтение отображается обычным способом, по умолчанию true.

Раздел **Attributes from PreferenceGroup** вкладки **Structure** редактора обеспечивает с помощью списка **Ordering from xml** определение атрибута android: orderingFromXml тэга <PreferenceScreen> – если true (по умолчанию), тогда при отсутствии атрибутов android: order дочерние предпочтения отображаются в том порядке, в каком они определены в XML-файле, если false, тогда дочерние предпочтения отображаются в алфавитном порядке их заголовков или согласно значений атрибутов android: order.

Тэг <CheckBoxPreference> (элемент **CheckBoxPreference**) обеспечивает отображение флажка в предпочтениях, при этом в файле устройства сохраняется значение true или false в зависимости

от выбранного состояния флажка. Раздел **Attributes from CheckBoxPreference** вкладки **Structure** редактора определяет атрибуты тэга <CheckBoxPreference> с помощью следующих полей и списков:

Summary on – определяет атрибут android: summaryOn, указывающий краткое описание отмеченного флажка, отображаемое ниже заголовка.

Summary off – определяет атрибут android: summaryOff, указывающий краткое описание не отмеченного флажка, отображаемое ниже заголовка.

Disable dependents state – атрибут android: disableDependentsState – если true, тогда предпочтение, в атрибуте android: dependency которого указан ключ флажка, становится недоступным при отмеченном флажке, если false – тогда при не отмеченном флажке.

Тэг <EditTextPreference> (элемент **EditTextPreference**) обеспечивает отображение поля для ввода строки, которая сохранится в файле устройства. Раздел **Attributes from DialogPreference** вкладки **Structure** редактора определяет атрибуты тэга <EditTextPreference> с помощью следующих полей:

Dialog title – определяет атрибут android: dialogTitle, указывающий заголовок диалогового окна с полем ввода и кнопками **OK** и **Отмена**, которое появляется при выборе предпочтения.

Dialog message – определяет атрибут android: dialogMessage, указывающий текст сообщения диалогового окна, который отображается ниже заголовка.

Dialog icon – определяет атрибут android: dialogIcon, указывающий ссылку на значок диалогового окна.

Positive button text – определяет атрибут android: positiveButtonText, указывающий текст кнопки OK диалогового окна.

Negative button text – определяет атрибут android: negativeButtonText, указывающий текст кнопки **Отмена** диалогового окна/

Dialog layout – определяет атрибут android: dialogLayout, указывающий ссылку на Layout-файл, определяющий View-компонент диалогового окна.

Тэг <ListPreference> (элемент **ListPreference**) обеспечивает отображение списка переключателей, при этом в файле устройства сохраняется значение выбранного переключателя. Раздел **Attributes from ListPreference** вкладки **Structure** редактора определяет атрибуты тэга <ListPreference> с помощью следующих полей:

Entries – определяет атрибут android: entries, указывающий ресурс @array/ [name] XML-файла каталога res/values, который определяет заголовки переключателей с помощью тэгов <string-array> и <item>.

Entry values – определяет атрибут android: entryValues, указывающий ресурс @array/ [name] XML-файла каталога res/values, который определяет значения переключателей с помощью тэгов <string-array> и <item>.

Тэг <MultiSelectListPreference> (элемент **MultiSelectListPreference**, версия Android 3.0 и выше) обеспечивает отображение списка флажков. Раздел **Attributes from MultiSelectListPreference** вкладки **Structure** редактора определяет атрибуты тэга <MultiSelectListPreference> с помощью следующих полей:

Entries – определяет атрибут android: entries, указывающий ресурс @array/ [name] XML-файла каталога res/values, который определяет заголовки флажков с помощью тэгов <string-array> и <item>.

Entry values – определяет атрибут android: entryValues, указывающий ресурс @array/ [name] XML-файла каталога res/values, который определяет значения флажков с помощью тэгов <string-array> и <item>.

Тэг <Preference> (элемент **Preference**) обеспечивает отображение предпочтения, содержимое которого определяется с помощью раздела **Attributes from Preference** вкладки **Structure** редактора.

Тэг <PreferenceCategory> (элемент **PreferenceCategory**) обеспечивает группировку предпочтений под определенным заголовком.

Тэг <RingtonePreference> (элемент **RingtonePreference**) обеспечивает отображение списка переключателей, позволяющих выбрать рингтон из мелодий устройства, при этом в файле устройства сохраняется URI-адрес мелодии. Раздел **Attributes from**

RingtonePreference вкладки **Structure** редактора определяет атрибуты тэга <RingtonePreference> с помощью следующих списков:

Ringtone type – определяет атрибут android: ringtoneType, указывающий тип мелодии для отображения в списке, возможные значения ringtone, notification, alarm, all.

Show default – определяет атрибут android: showDefault – если true (по умолчанию), тогда в списке отображается переключатель **Мелодия по умолчанию**, позволяющий выбрать рингтон устройства по умолчанию для данного типа мелодий.

Show silent – определяет атрибут android: showSilent – если true (по умолчанию), тогда в списке отображается переключатель **Без звука**, при выборе которого в файле устройства сохраняется пустая строка.

Тэг <SwitchPreference> (элемент **SwitchPreference**, для версии Android 4.0 и выше) обеспечивает отображение предпочтения, которое может иметь два состояния – нажатое и отжатое, при этом в файле устройства сохраняется значение true или false в зависимости от выбранного состояния предпочтения. Раздел **Attributes from SwitchPreference** вкладки **Structure** редактора определяет атрибуты тэга <SwitchPreference> с помощью следующих полей и списков:

Summary on – определяет атрибут android: summaryOn, указывающий краткое описание нажатого предпочтения, отображаемое ниже заголовка.

Summary off – определяет атрибут android: summaryOff, указывающий краткое описание отжатого предпочтения, отображаемое ниже заголовка.

Switch text on – определяет атрибут android: switchTextOn, указывающий текст переключателя в нажатом состоянии.

Switch text off – определяет атрибут android: switchTextOff, указывающий текст переключателя в отжатом состоянии.

Disable dependents state – атрибут android: disableDependentsState – если true, тогда предпочтение, в атрибуте android: dependency которого указан ключ данного предпочтения, становится недоступным при нажатом предпочтении, если false – тогда при отжатом предпочтении.

Тип ресурса Searchable

Android-система обеспечивает механизм, помогающий Android-приложениям предоставлять пользователю возможность поиска различного рода данных. При этом для пользователя в Android-приложении в верхней части экрана устройства активируется диалоговое окно поиска, содержащее поле ввода запроса, или пользователь взаимодействует с SearchView-компонентом (для версии Android 3.0 и выше).

Для реализации функций поиска на основе поискового каркаса Android-платформы Android-приложение должно иметь:

Ресурс Searchable – XML-файл с настройками диалогового окна поиска или SearchView-компонента.

Activity-компонент, который получает строку запроса пользователя, обрабатывает ее и отображает результаты запроса пользователю. Строка запроса пользователя содержится в дополнительных данных Intent-объекта, который Android-система автоматически формирует и отправляет Activity-компоненту при завершении пользователем работы в диалоговом окне поиска или SearchView-компоненте. Строка запроса извлекается Activity-компонентом из Intent-объекта методом getIntent().getStringExtra (SearchManager. QUERY).

Файл манифеста AndroidManifest. xml, в котором Activity-компонент, обрабатывающий запрос, при своем объявлении содержит Intent-фильтр, указывающий действие поиска, и тэг <meta-data> со ссылкой на Searchable-ресурс:

```
<intent-filter>
<action android:name="android.intent.action.SEARCH» />
</intent-filter>
<meta-data android:name="android.app.searchable»
android: resource="@xml/ [filename]»/>
```

Для включения диалогового окна поиска Activity-компонент, предоставляющий его пользователю, при своем объявлении в файле манифеста AndroidManifest. xml содержит тэг <meta-data> со ссылкой на Activity-компонент, получающий запрос:

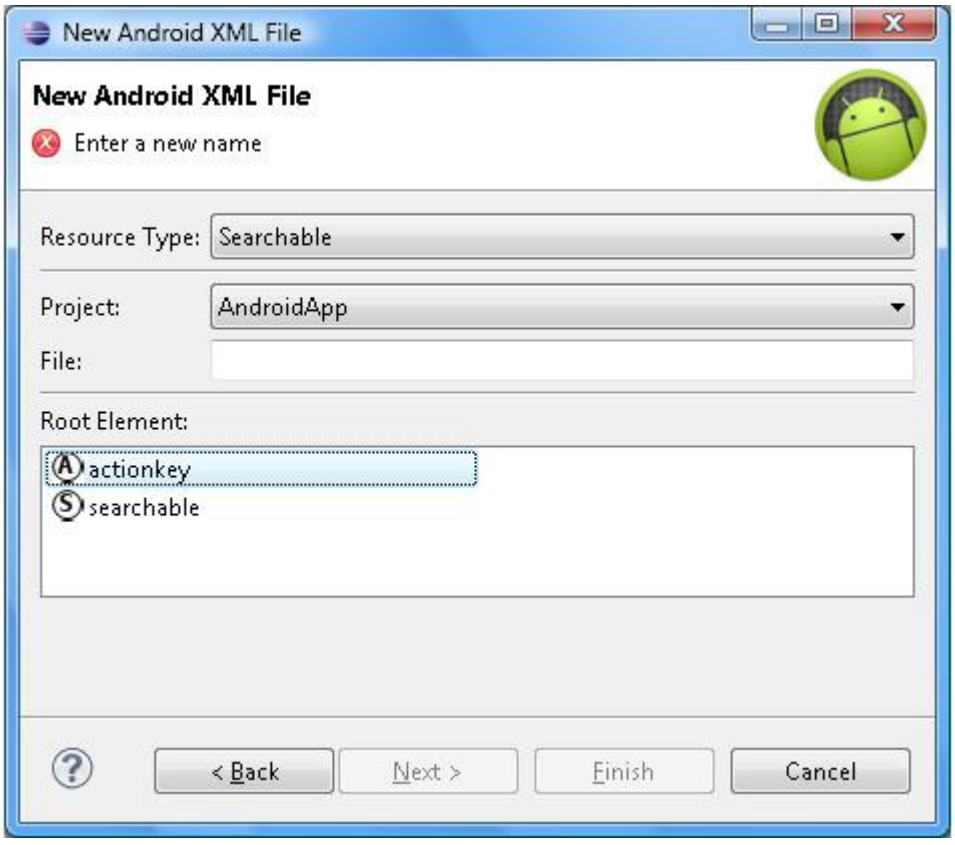
<meta-data android:name="android.app.default_searchable»

android: value=». [имя класса Activity-компонента, получающего запрос]» />

Диалоговое окно поиска активируется в Activity-компоненте нажатием кнопки **SEARCH** устройства или вызовом метода onSearchRequested ().

Кроме того, поисковый каркас Android-платформы позволяет Android-приложению обеспечить для пользователя поисковые предложения для облегчения ввода запроса. При этом поисковые предложения могут быть двух типов – основанные на ранее введенных запросах или пользовательские предложения, хранящиеся в базе данных. Для реализации поисковых предложений Android-приложение должно иметь SearchRecentSuggestionsProvider-компонент для предложений, основанных на ранее введенных запросах, или ContentProvider-компонент для пользовательских предложений.

Для создания Searchable-ресурса в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android XML File**, нажмем кнопку **Next** – в результате откроется окно мастера, в списке **Resource Type** которого выберем тип **Searchable**.



Поле **File:** мастера создания ресурса Searchable предлагает ввести имя нового XML-файла, который затем появится в каталоге res/xml Android-проекта и будет доступен в XML-коде с помощью ссылки @ [package:] xml/filename.

Раздел **Root Element:** мастера отображает элементы **actionkey** и **searchable**, представляющие тэги <actionkey> и <searchable> соответственно, при этом тэг <searchable> служит корневым тэгом XML-файла ресурса Searchable.

После ввода имени нового XML-файла Searchable-ресурса, выбора элемента **searchable** и нажатия кнопки **Next**, появляется окно **Choose Configuration Folder**, позволяющее выбрать спецификатор папки xml, обеспечивающий поддержку специфической конфигурации Android-устройства, в соответствии с которой папка xml с нужным

спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

После создания нового файла Searchable-ресурса он будет открыт в визуальном редакторе ADT-плагина.

Кнопка **Add** вкладки **Structure** визуального редактора обеспечивает добавление в корневой тэг <searchable> файла Searchable-ресурса тэгов <actionkey> (элемент **Action Key**).

Для тэга <searchable> раздел **Attributes for Searchable** вкладки **Structure** редактора обеспечивает определение атрибутов с помощью следующих полей и списков:

Icon – определяет атрибут android: icon (больше не используется), указывающий значок поиска.

Label – определяет атрибут android: label, указывающий имя приложения для отображения в настройках поиска устройства.

Hint – определяет атрибут android: hint, указывающий строку, которая первоначальное отображается в поле ввода для подсказки пользователю.

Search button text – определяет атрибут android: searchButtonText (больше не используется), указывающий текст кнопки поиска.

Input Type – определяет атрибут android: inputType, указывающий тип вводимого текста, возможные значения none, text, textCapCharacters, textCapWords, textCapSentences, textAutoCorrect, textAutoComplete, textMultiLine, textImeMultiLine, textNoSuggestions, textUri, textEmailAddress, textEmailSubject, textShortMessage, textLongMessage, textPersonName, textPostalAddress, textPassword, textVisiblePassword, textWebEditText, textFilter, textPhonetic, textWebEmailAddress, textWebPassword, number, numberSigned, numberDecimal, numberPassword, phone, datetime, date, time.

IME options – определяет атрибут android: imeOptions, указывающий дополнительные опции ввода, возможные значения normal, actionUnspecified, actionNone, actionGo, actionSearch, actionSend, actionNext, actionDone, actionPrevious, flagNoFullscreen, flagNavigatePrevious, flagNavigateNext, flagNoExtractUi, flagNoAccessoryAction, flagNoEnterAction.

Search mode – определяет атрибут android: searchMode, указывающий способ замещения запроса поисковым предложением и отображения значка поиска, возможные значения showSearchLabelAsBadge, showSearchIconAsBadge, queryRewriteFromData, queryRewriteFromText.

Voice search mode – определяет атрибут android: voiceSearchMode, указывающий отображение кнопки голосового поиска, переключение на Activity-компонент Web-поиска, переключение на Activity-компонент перевода речи в текст. Возможные значения showVoiceSearchButton, launchWebSearch, launchRecognizer.

Voice language model – определяет атрибут android: voiceLanguageModel, указывающий модель распознавания речи, возможные значения free_form и web_search.

Voice prompt text – определяет атрибут android: voicePromptText, указывающий дополнительное сообщение пользователю при голосовом запросе.

Voice language – определяет атрибут android: voiceLanguage, указывающий язык голосового запроса.

Voice max results – определяет атрибут android: voiceMaxResults, указывающий максимальное возвращаемое количество вариантов перевода речи в текст.

Search suggest authority – определяет атрибут android: searchSuggestAuthority, указывающий класс SearchRecentSuggestionsProvider-компонента или ContentProvider-компонента, обеспечивающего поисковые предложения.

Search suggest path – определяет атрибут android: searchSuggestPath, указывающий дополнительный путь предложения для разрешения неоднозначностей. Дополнительный путь включается в URI-запрос предложений Android-системы к ContentProvider-компоненту: content://[suggest_authority]/[optional.suggest.path] /SUGGEST_URI_PATH_QUERY.

Search suggest selection – определяет атрибут android: searchSuggestSelection, указывающий параметр selection, передаваемый методу query () ContentProvider-компонента.

Search suggest intent action – определяет атрибут android: searchSuggestIntentAction, указывающий Intent-действие по умолчанию, используемое при выборе пользовательского предложения поиска.

Search suggest intent data – определяет атрибут android: searchSuggestIntentData, указывающий Intent-данные по умолчанию, используемые при выборе пользовательского предложения поиска.

Search suggest threshold – определяет атрибут android: searchSuggestThreshold, указывающий минимальное количество введенных пользователем символов для активации поисковых предложений.

Include in global search – определяет атрибут android: includeInGlobalSearch – если true, тогда поисковые предложения данного Android-приложения включаются в Quick Search Box.

Query after zero results – определяет атрибут android: queryAfterZeroResults – если true, тогда при получении нулевого результата ContentProvider-компонент вызывается еще раз с расширенным набором символов, по умолчанию false.

Search settings description – определяет атрибут android: searchSettingsDescription, указывающий описание поисковых предложений для Quick Search Box.

Auto url detect – определяет атрибут android: autoUrlDetect – если true, тогда вводимый запрос обрабатывается как URL-адрес с вызовом браузера.

Тэг <actionkey> (элемент **Action Key**) определяет клавишу устройства, которую пользователь может нажать вместо кнопки поиска. Для тэга <actionkey> раздел **Attributes for Action Key** вкладки **Structure**редактора обеспечивает определение атрибутов с помощью следующих полей и списков:

Keycode – определяет атрибут android: keycode, указывающий код клавиши устройства.

Query action msg – определяет атрибут android: queryActionMsg, указывающий сообщение действия ACTION_SEARCH, посылаемое при нажатии клавиши.

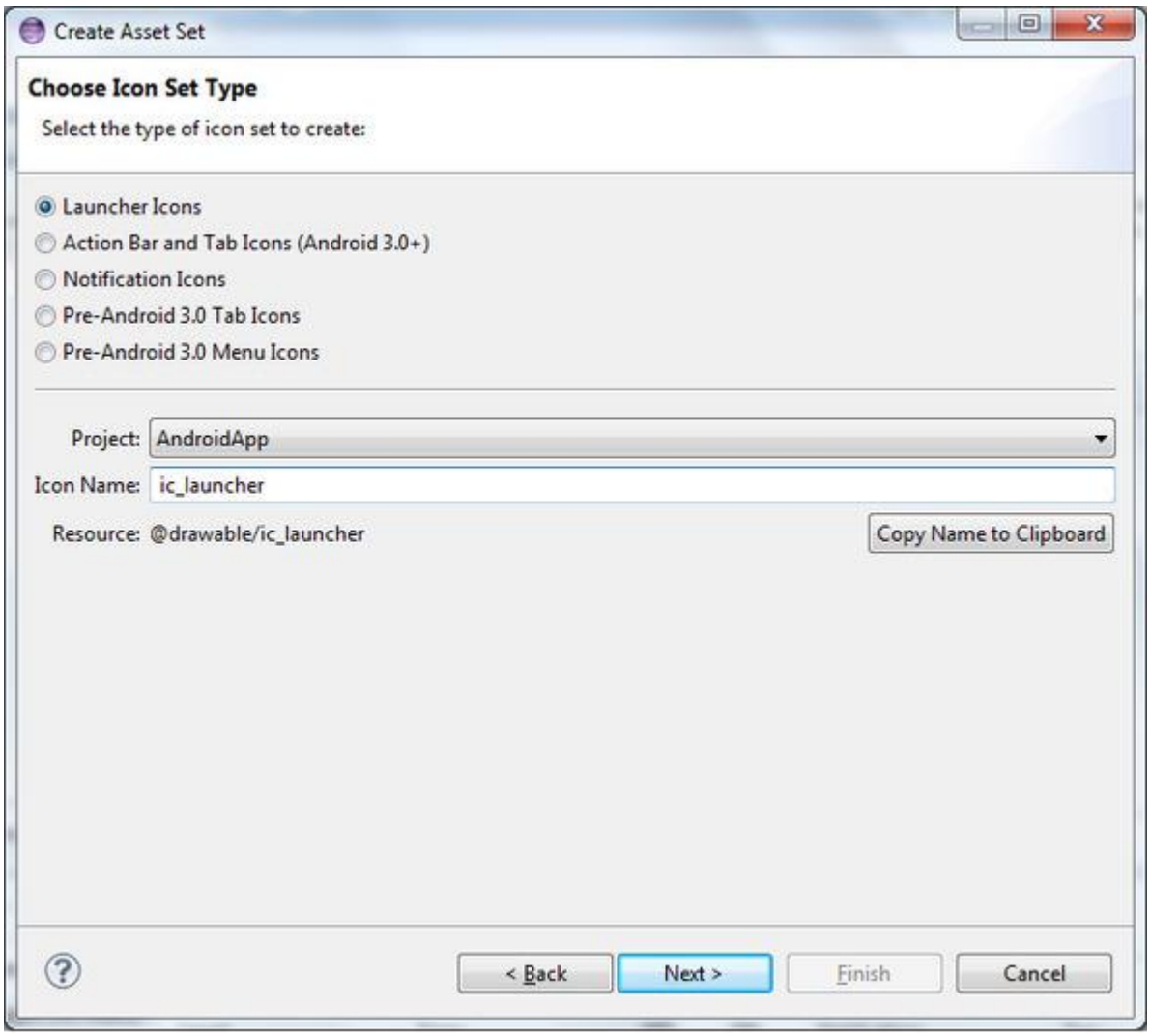
Suggest action msg – определяет атрибут android: suggestActionMsg, указывающий сообщение действия, посылаемое при выборе поискового предложения.

Suggest action msg column – определяет атрибут android: suggestActionMsgColumn, указывающий имя столбца ContentProvider-компонента для установки сообщения действия, посылаемого при выборе поискового предложения.

Мастер Android Icon Set

Мастер **Android Icon Set** ADT-плагина помогает создать значок Launcher Icons, представляющий приложение, значки Menu Icons опций меню, значки Action Bar Icons элементов панели действий, значки Tab Icons вкладок и значки Notification Icons уведомлений панели состояния.

Для создания значков приложения в окне **Project Explorer** нажмем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду **New | Other | Android | Android Icon Set**, нажмем кнопку **Next** – в результате откроется окно мастера.



Окно мастера **Android Icon Set** позволяет выбрать с помощью переключателя тип значка и ввести имя его файла, после чего нажатием кнопки **Next** перейти к созданию значка.

Кнопка **Text** мастера позволяет создать значок в виде текста на цветном фоне. При этом поле **Text:**определяет текст значка, а кнопка **Font:** дает возможность выбрать шрифт текста.

Кнопка **Image** мастера с помощью кнопки **Browse** поля **Image File:** позволяет выбрать в качестве значка изображение.

Кнопка **Clipart** мастера с помощью кнопки Choose позволяет выбрать из предоставляемой коллекции изображение.

Ползунок **Additional Padding** мастера устанавливает отступ текста от краев значка путем масштабирования текста.

Для Launcher-значков опция **Foreground Scaling:** с помощью кнопок **Crop** и **Center** определяет способ масштабирования текста, опция **Shape** с помощью кнопок **Square** и **Circle** позволяет установить форму значка, а опции **Background Color:** и **Foreground Color:** обеспечивают выбор цвета фона и текста значка.

Для значков Action Bar Icons кнопки **Holo Light** и **Holo Dark** опции **Theme** определяют стиль отображения текста.

Окно **Preview** мастера показывает конечный вид значка для экранов устройства с различной плотностью.

После нажатия кнопки **Finish** мастера будет создан набор файлов значка в каталогах res/drawable- [ldpi, mdpi, hdpi, xhdpi] со спецификаторами, обеспечивающими поддержку экранов Android-устройства с различной плотностью, в соответствии с которой папка drawable с нужным

спецификатором будет выбрана Android-системой для загрузки при выполнении кода приложения.

Мастер Android Test Project

Мастер **Android Test Project** помогает создать для выбранного Android-проекта набор тестов на базе Android-расширения платформы тестирования JUnit.

Android-расширение платформы тестирования JUnit представлено библиотекой android. test платформы Android.

ADT-плагин обеспечивает сборку проекта Android-тестов в пакет и его загрузку вместе с пакетом тестируемого Android-приложения в Android-устройство, в котором инструмент android.test.InstrumentationTestRunner запускает созданный набор тестов. При тестировании Android-приложения в Android-устройстве ни Android-система, ни инструмент InstrumentationTestRunner сами по себе не запускают Android-приложение, это делают Android-тесты путем вызова соответствующих методов.

Проект Android-тестов имеет ту же структуру, что и проект Android-приложения. Для предотвращения конфликтов в Android-системе имя пакета Android-тестов состоит из имени пакета тестируемого Android-приложения плюс расширение. test. Кроме того, файл манифеста AndroidManifest. xml проекта Android-тестов содержит тэг <instrumentation>, устанавливающий в качестве инструмента запуска тестов инструмент InstrumentationTestRunner, а также определяющий имя пакета тестируемого Android-приложения. Так как код инструмента InstrumentationTestRunner содержится в отдельной библиотеке android.test.runner, тэг <application> файла манифеста AndroidManifest. xml проекта Android-тестов содержит тэг <uses-library>, указывающий необходимость загрузки библиотеки android.test.runner.

Для тестирования компонентов Android-приложения в каталоге src проекта Android-тестов в пакете Android-тестов требуется создание классов, расширяющих классы тестов программного интерфейса Android Testing API:

android.test.ActivityInstrumentationTestCase2 <T extends android.app.Activity> – обеспечивает тестирование отдельного Activity-компонента, с его запуском в экземпляре Android-приложения, использующим обычную инфраструктуру Android-системы.

android.test.ActivityUnitTestCase <T extends android.app.Activity> – обеспечивает тестирование отдельного Activity-компонента, изолированного от Android-системы.

android.test.SingleLaunchActivityTestCase <T extends android.app.Activity> – обеспечивает тестирование отдельного Activity-компонента с его загрузкой только один раз для тестирования режима загрузки, отличного от standard (атрибут android: launchMode тэга <activity>).

android.test.ProviderTestCase2 <T extends android.content.ContentProvider> – обеспечивает тестирование отдельного ContentProvider-компонента в изолированном окружении.

android.test.ServiceTestCase <T extends android.app.Service> – обеспечивает тестирование отдельного изолированного Service-компонента.

android. test. ApplicationTestCase <T extends android. app. Application> – обеспечивает тестирование Application-класса.

Для создания проекта Android-тестов в меню **File** среды Eclipse выберем команду **New | Other | Android | Android Test Project** и нажмем кнопку **Next**, введем имя проекта и нажмем кнопку **Next**, выберем Android-проект для тестирования и нажмем кнопку **Next,** выберем версию Android-платформы и нажмем кнопку **Finish**. В результате ADT-плагином будет сгенерирована основа проекта Android-тестов для выбранного Android-приложения.

Для создания класса тестов в окне **Project Explorer** среды Eclipse нажмем правой кнопкой мышки на узле пакета проекта Android-тестов и в контекстном меню выберем команду **New | Other | Java | Class**, нажмем кнопку **Next** и в окне мастера создания Java-класса в поле **Name:** введем имя класса тестов, а в поле **Superclass:** введем имя расширяемого класса библиотеки android. test платформы Android, например android.test.ActivityInstrumentationTestCase2 <NameActivity>, где NameActivity – имя класса Activity-компонента тестируемого Android-приложения. После нажатия кнопки **Finish** будет сгенерирована и открыта в Eclipse-редакторе основа класса тестов.

После заполнения необходимым кодом класса тестов, для запуска Android-тестов запустим Android-эмулятор, используя приложение AVD Manager, после этого в окне **Project**

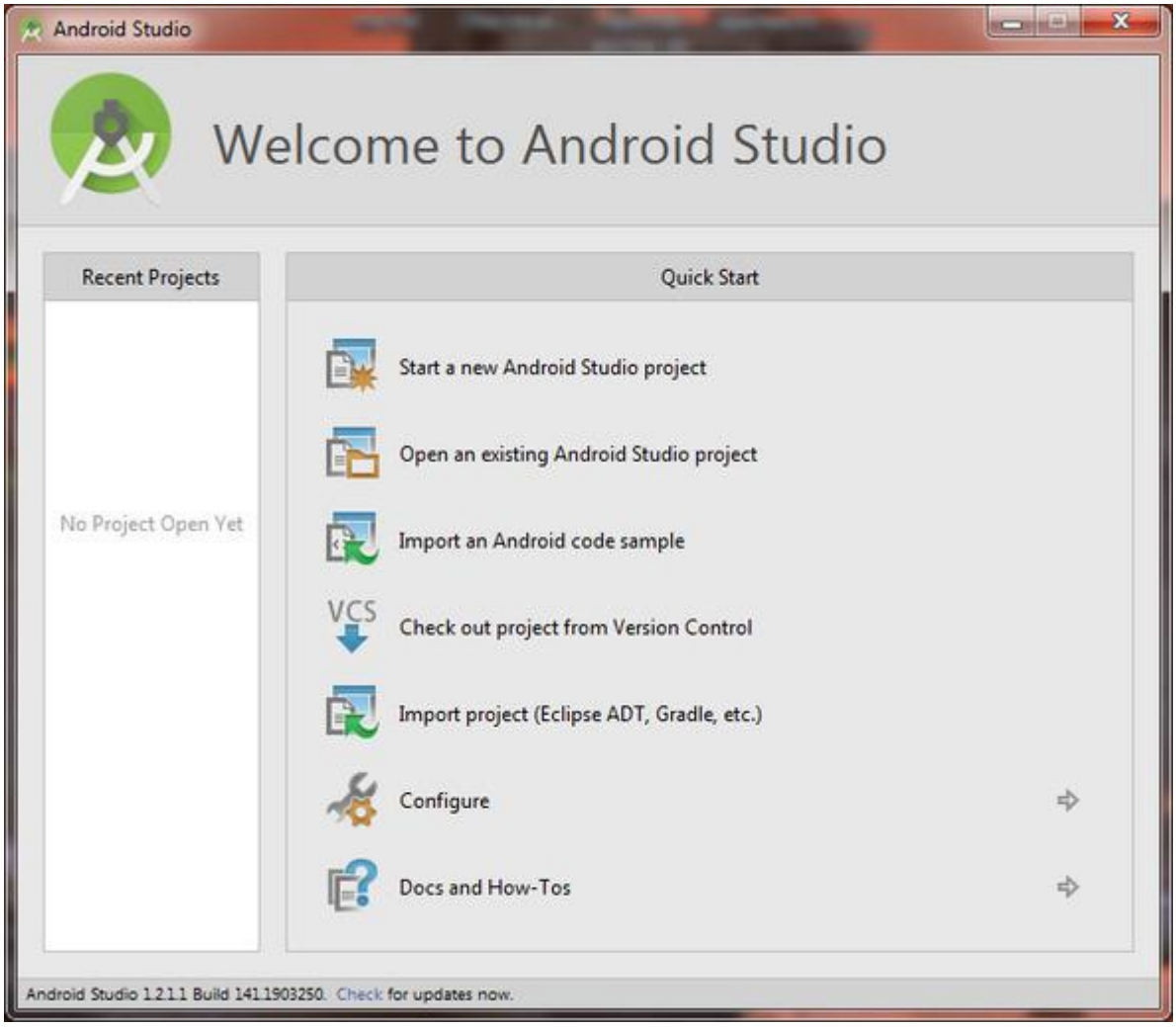
Explorer среды Eclipse нажмем правой кнопкой мышки на узле проекта Android-тестов и в контекстном меню выберем команду **Run As | Android JUnit Test**. В результате ADT-плагин загрузит пакет Android-тестов вместе с пакетом тестируемого Android-приложения в Android-устройство, в котором будут запущены созданные тесты. При этом результаты тестирования будут возвращены в среду Eclipse для отображения в автоматически открывающемся представлении **JUnit.**

Среда разработки Android Studio

Начало работы

К сожалению или к счастью, в настоящее время официальной средой разработки Android-приложений является среда Android Studio, основанная на интегрированной среде IntelliJ IDEA.

После загрузки Android Studio встречает приглашением создать новый проект.



Нужно отметить, что проект среды Android Studio является аналогом Workspace в Eclipse, а модуль Android Studio это аналог Eclipse-проекта.

Кроме того, Android Studio поставляется с системой сборки проектов Gradle, интегрированной в Android Studio с помощью плагина Android Gradle. Система сборки проектов Gradle позволяет для одного и того же проекта и модулей генерировать различные APK-файлы приложения, имеющие различные свойства, например, платные или бесплатные, предназначенные для работы на различных уствойствах, например, для работы на смартфонах или планшетах.

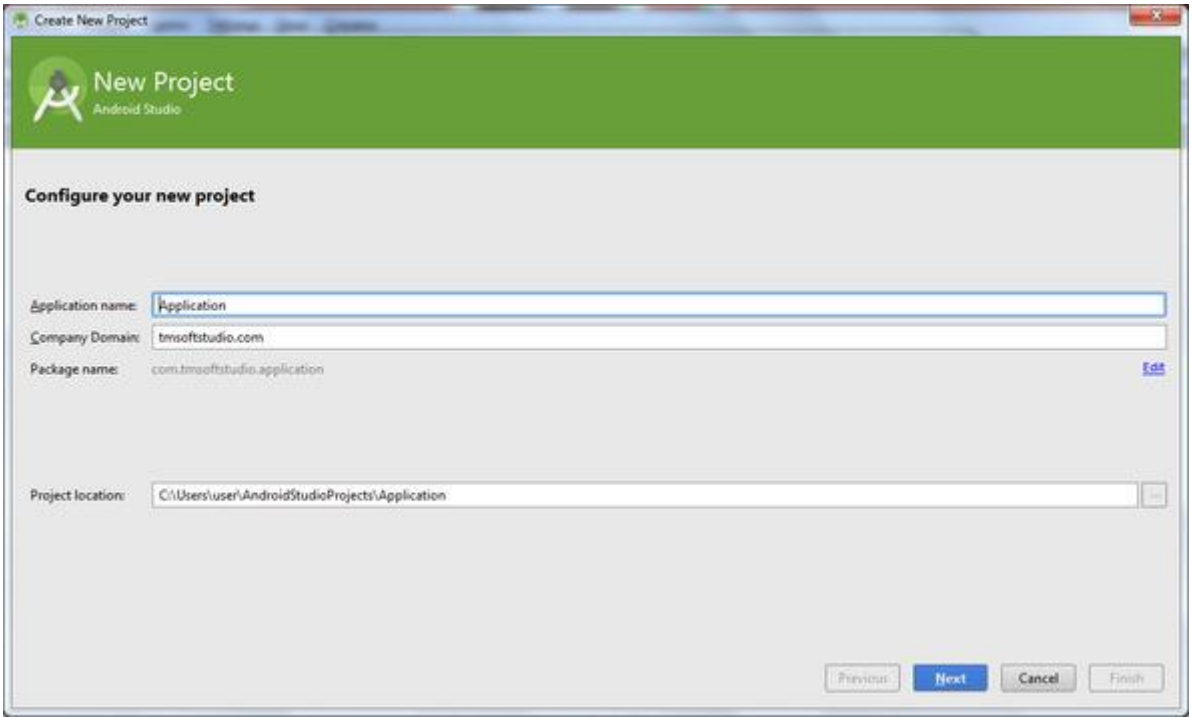
Также Android Studio поставляется с инструментом lint, автоматически анализирующим код при компиляции на наличие потенциальных багов и возможностей оптимизации кода.

Сжатие, оптимизация и обсфукация кода APK-файла приложения в Android Studio производится с помощью инструмента ProGuard.

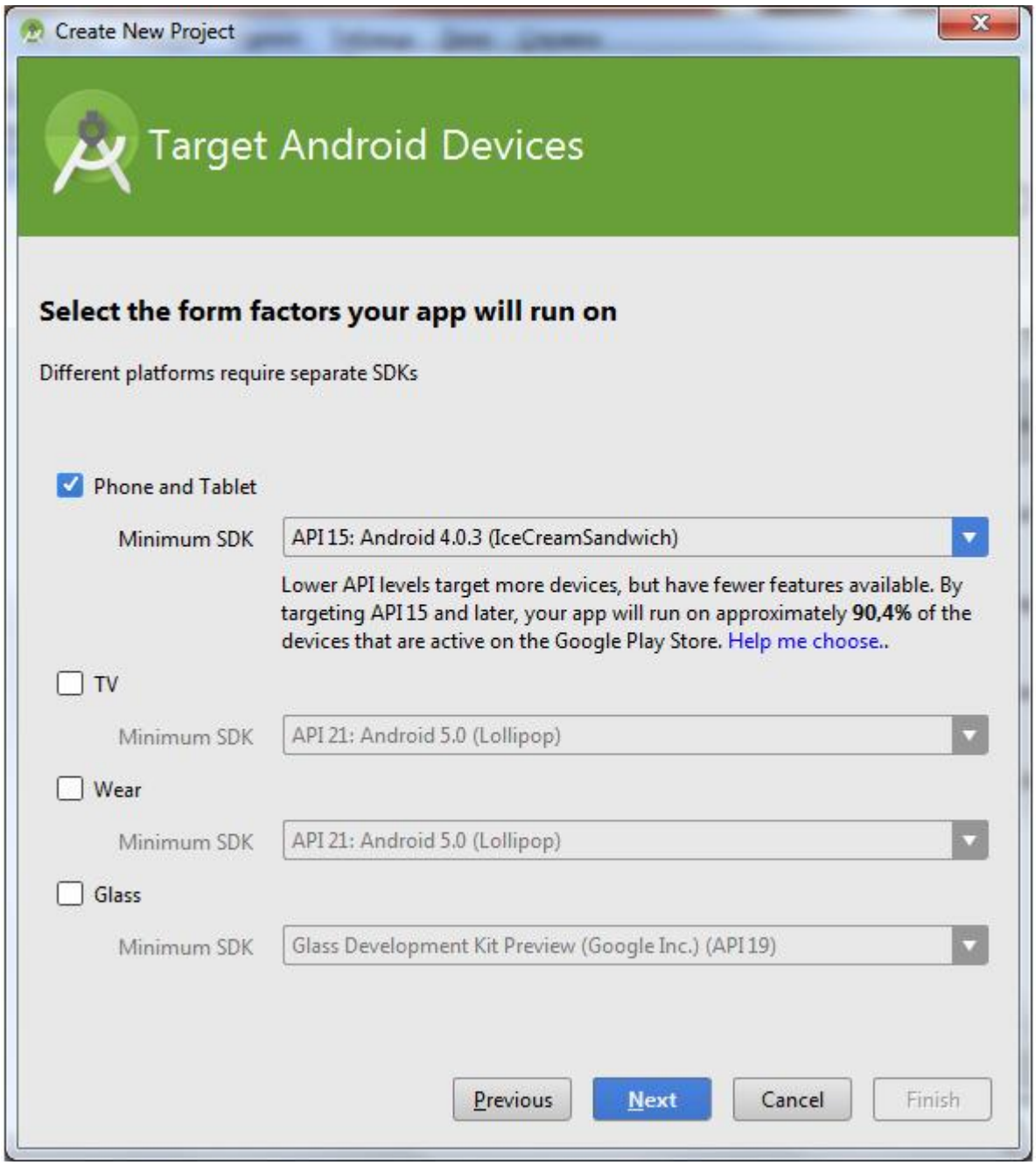
И lint и ProGuard запускаются в Android Studio системой сборки проектов Gradle.

Для облачных вычислений Android Studio позволяет создать и развернуть модуль App Engine.

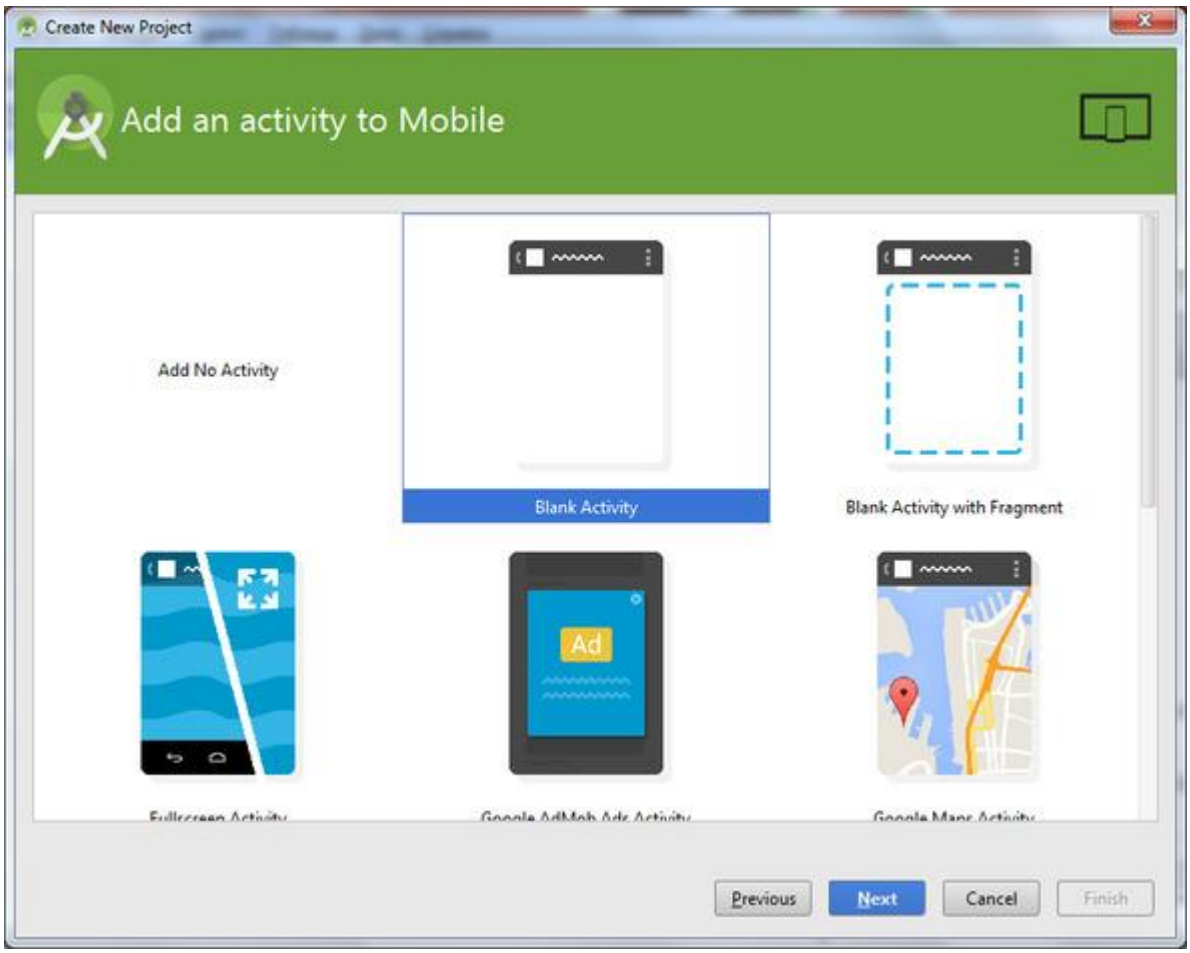
После выбора **Start a new Android Studio project** откроется диалоговое окно, в котором нужно определить имя папки проекта, имя пакета и расположение проекта.

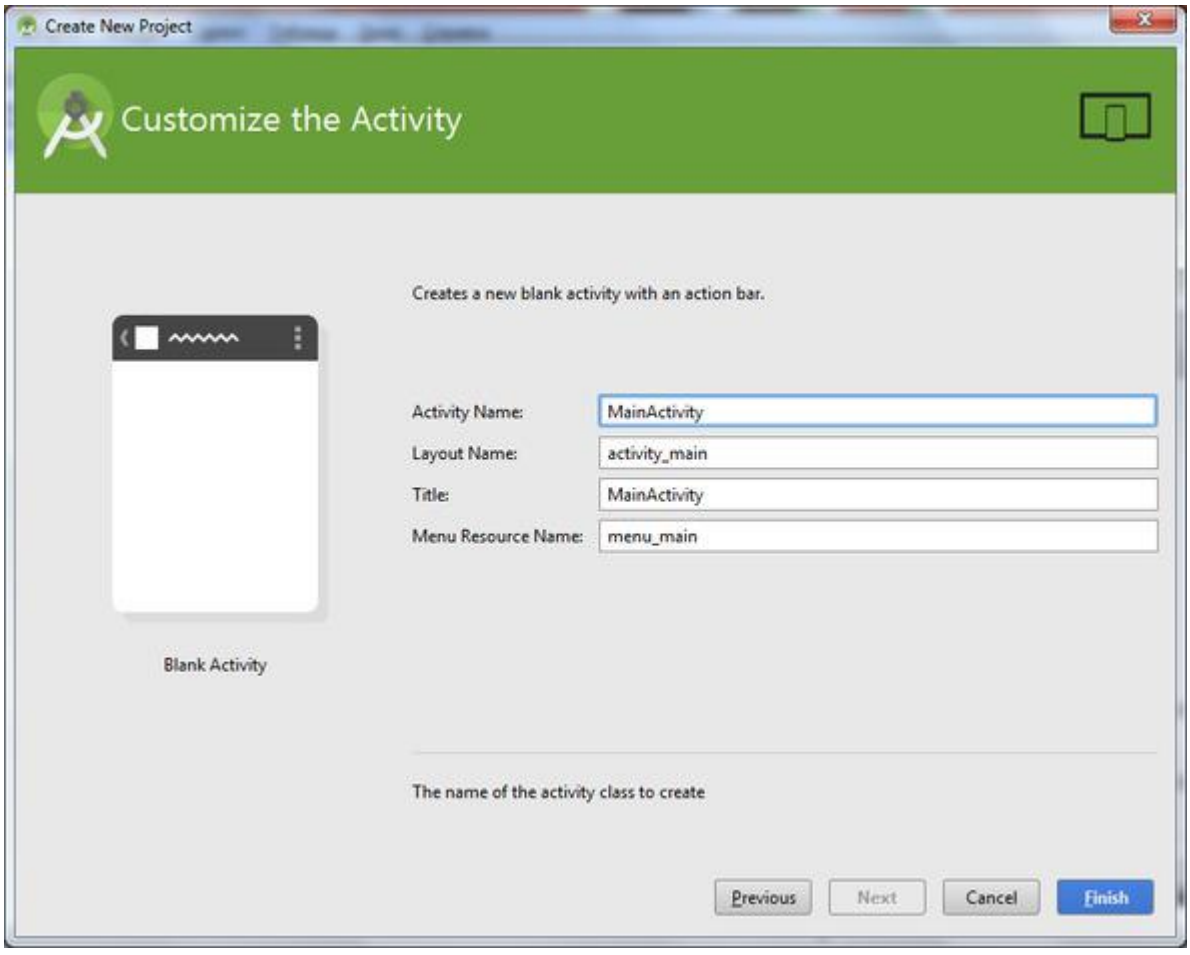


Далее оставим рекомендованный минимальный уровень Android API.



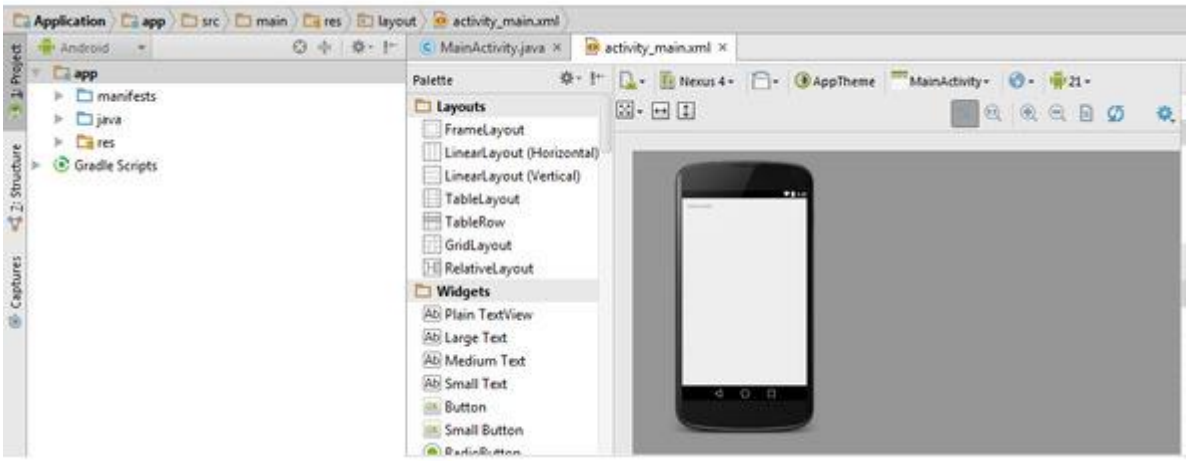
И выберем создание модуля с пустой активностью.


Create New Project
Add an activity to Mobile
Add No Activity
Blank Activity
Blank Activity with Fragment
Ad
Previous
Next
Cancel
Finish

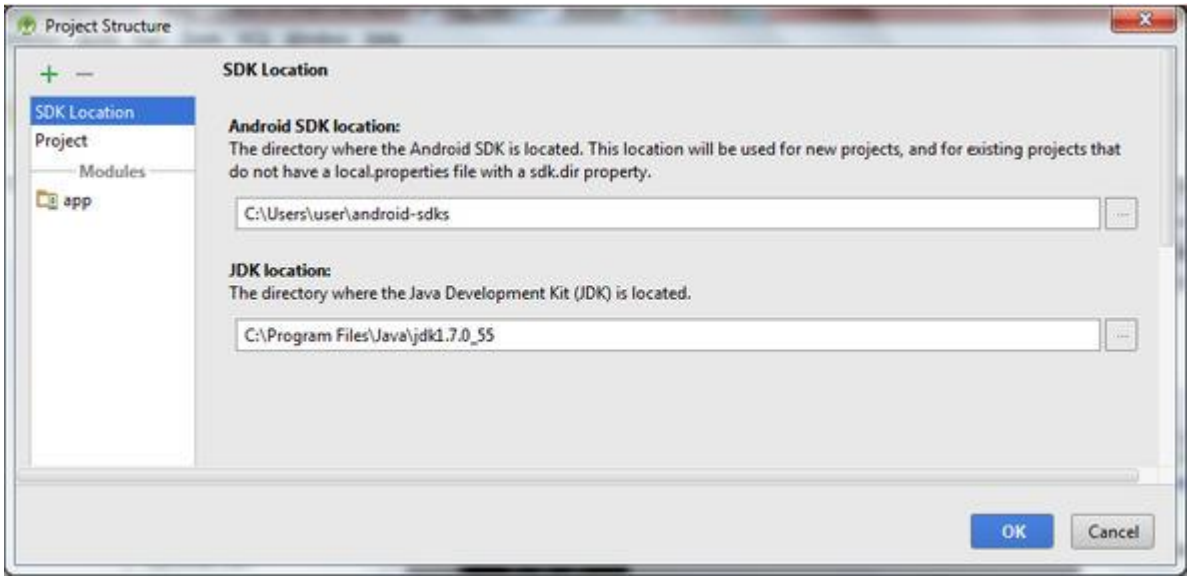

Create New Project
Customize the Activity
Creates a new blank activity with an action bar.
Activity Name:
MainActivity
Layout Name:
activity_main
Title:
MainActivity
Menu Resource Name:
menu_main
Blank Activity
The name of the activity class to create
Previous
Next
Cancel
Finish

Нажмем **Finish**.

В результате будет сгенерирована основа модуля Android-приложения, Android-структура которого отобразится в View-представлении **Project** среды разработки.

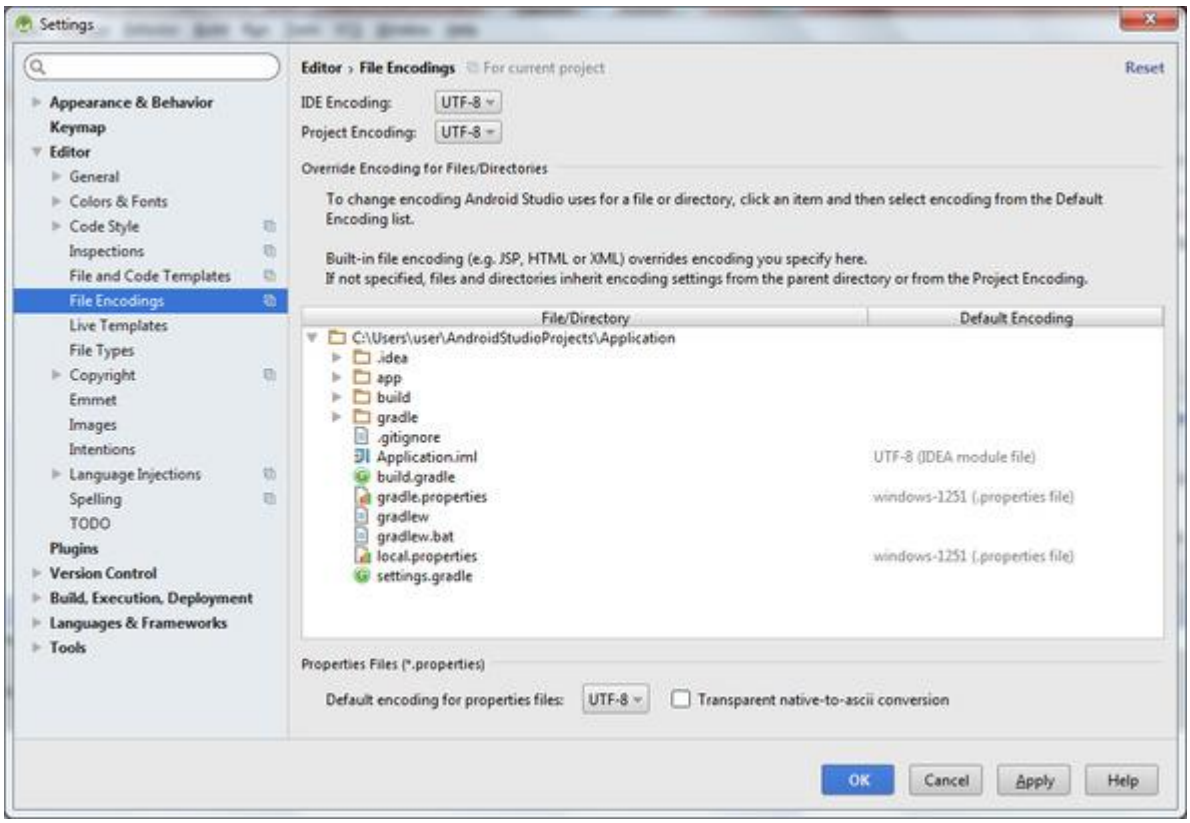


Чтобы изменить Java SDK или Android SDK для проекта, в меню **File** выберем **Project Structure** и в **SDK Location** изменим каталог SDK.

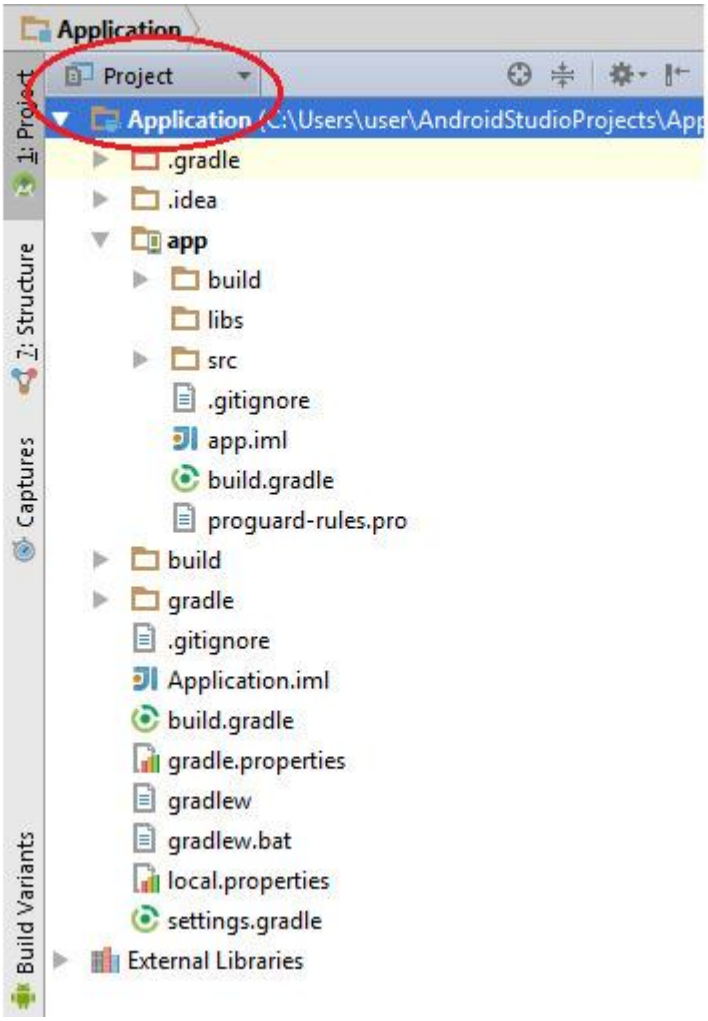


Тут же выбрав модуль, можно изменить для него уровень Android API.

Изменить кодировку проекта на UTF-8 можно, в меню **File** выбрав **Settings** и далее **File Encodings**.

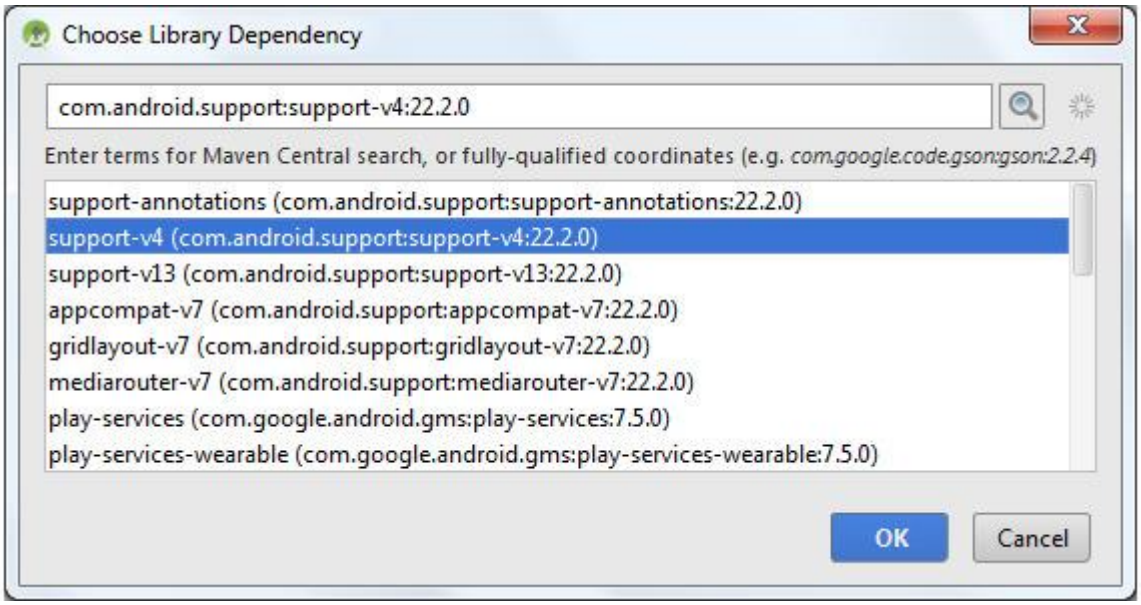


Android-структура приложения отличается от его файловой структуры.
Чтобы посмотреть файловую структуру приложения, нажмем на **Android** и выберем **Project**.



В отличие от Eclipse, здесь каталог src содержит пакет класса, расширяющего класс android.app.Activity, пакет тестов, папку ресурсов приложения и файл манифеста приложения.

Для добавления библиотек v4 support и v7 appcompat в модуль, в меню **File** выберем **Project Structure**и во вкладке **Dependencies** нажмем кнопку **Add**, выберем **Library dependency** и соответствующую библиотеку.

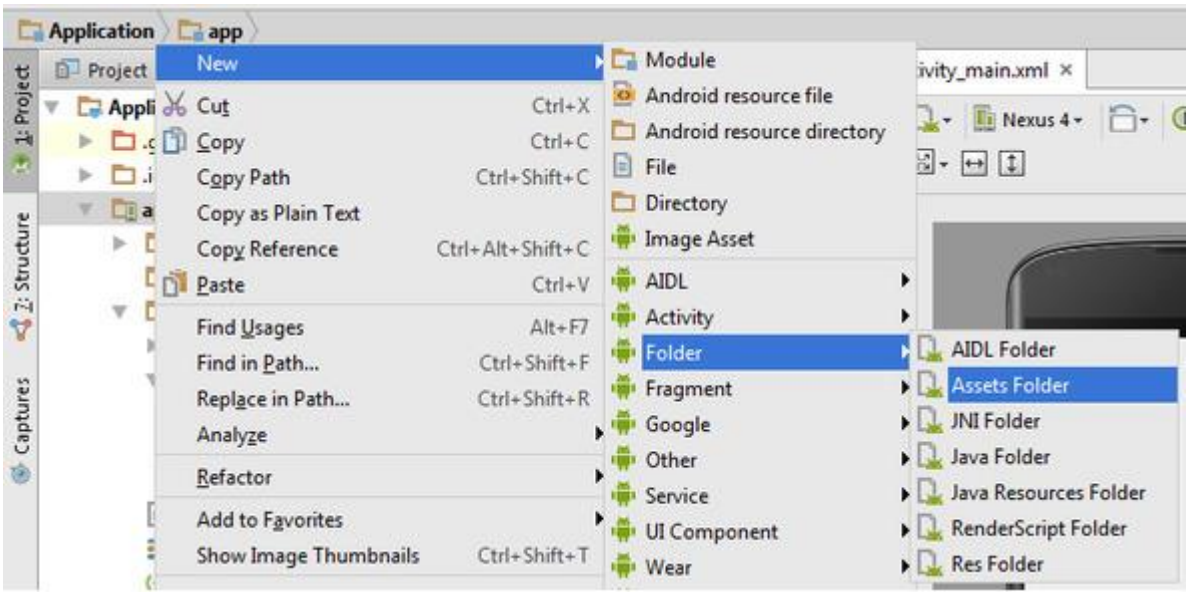


Для создания таких ресурсов, как animator (XML-файлы для создания объектов анимации), color (XML-файлы, определяющие цветовую гамму View-объектов), drawable (PNG, JPEG, GIF, 9-PNG и XML-файлы, формирующие графику), layout (XML-файлы для формирования структуры GUI-интерфейса Activity-объектов), menu (XML-файлы, описывающие меню приложения), raw (каталог предназначен для хранения таких данных приложения как файлов в формате MP3 или Ogg), values (XML-файлы для хранения строк, стилей, чисел, размеров и другое, используемых приложением, в виде пар имя-значение), xml (различные конфигурационные и ресурсные XML-файлы), нажмем правой кнопкой мышки на узле **res**модуля и выберем **New | Android resource** directory. Для создания ресурсного файла выберем **New | Android resource file**.

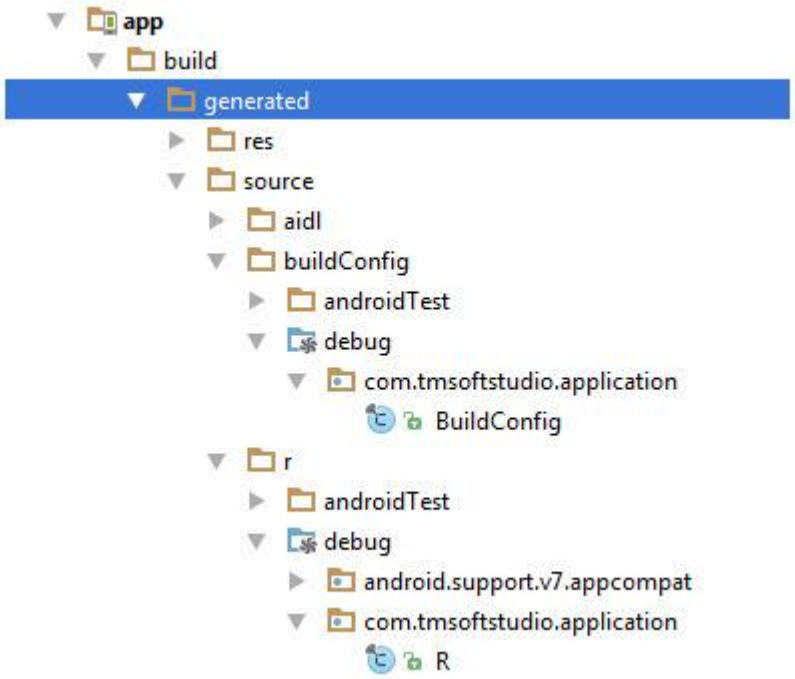
Дополнительно здесь предусмотрено создание папки anim для Tween-анимации, папки interpolator, папки mipmap для значков приложения, папки transition для Transition-анимации.

Значки приложения можно создать, нажав правой кнопкой мышки на узле **res** модуля и выбрав **New | Image Asset**.

Для создания каталога assets, предназначенного для хранения данных приложения, доступ к которым осуществляется с помощью класса android.content.res.AssetManager, в контекстном меню выберем **New | Folder | Assert Folder**.



R-класс и класс BuildConfig находятся в каталоге build/generated модуля.



Кнопка **AVD Manager** панели Android Studio или меню **Tools | Android | AVD Manager** позволяют запустить приложение AVD Manager для создания и запуска эмулятора Android-устройства.

Кнопка **Run «app'** панели Android Studio или меню **Run | Run «app'** позволяют запустить модуль в Android-устройстве.

Меню **Run | Edit Configurations** позволяет изменить настройки запуска модуля.

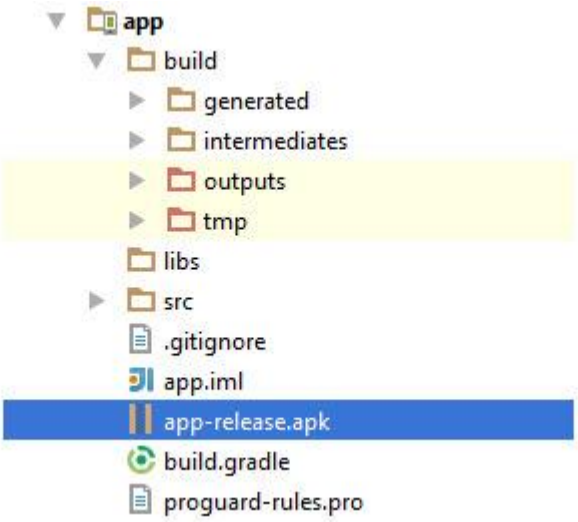
В результате запуска Android-приложения каталог **build** модуля пополнится компилированными и конвертированными в DEX-формат виртуальной машины Dalvik файлами модуля, а также APK-файлом Android-приложения.

Папка build/outputs/apk будет содержать два APK-файла app-debug-unaligned. apk и app-debug. apk. Сначала генерируется файл app-debug-unaligned. apk, а затем на его основе генерируется оптимизированный app-debug. apk файл, который и загружается для отладки.

Отдельно для сборки модуля щелкнем правой кнопкой мышки на модуле и выберем **Make Module «app'**.

Для подготовки к публикации Android-приложения и создания подписанного APK-файла приложения в меню **Build** выберем **Generate Signed APK**. Далее для создания хранилища закрытого ключа, которым будут подписываться Android-приложения, нажмем кнопку **Create new**, определим путь и имя файла хранилища, введем пароль и имя хранилища, параметры сертификата и нажмем кнопку **OK**. Нажмем кнопку **Next**, введем пароль и нажмем кнопку **OK** и далее кнопку **Finish.**

В результате в корневом каталоге модуля будет создан подписанный и готовый к публикации APK-файл приложения.



Для отладки Android-приложения в меню Run выберем **Debug «app'**. В результате откроется диалоговое окно выбора устройства, на котором будет запущено приложение, и после выбора, например, эмулятора и запуска приложения, в меню **View | Tool Windows** можно воспользоваться элементом **Debug**, открывающим или закрывающим окно отладчика.

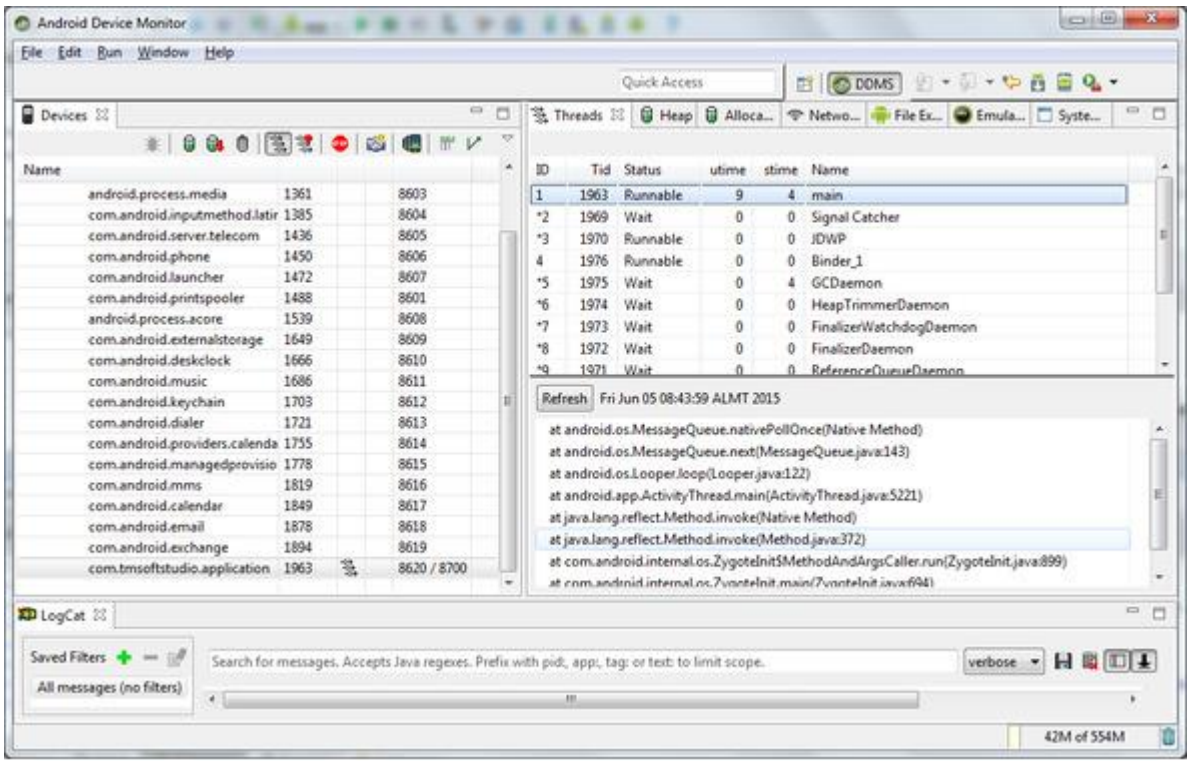
В меню **Run** с помощью выбора **Attach debugger to Android proccess** можно присоединить отладчик к конкретному процессу.

В меню **View | Tool Windows**, выбрав **Android**, можно открыть окно инструмента DDMS.

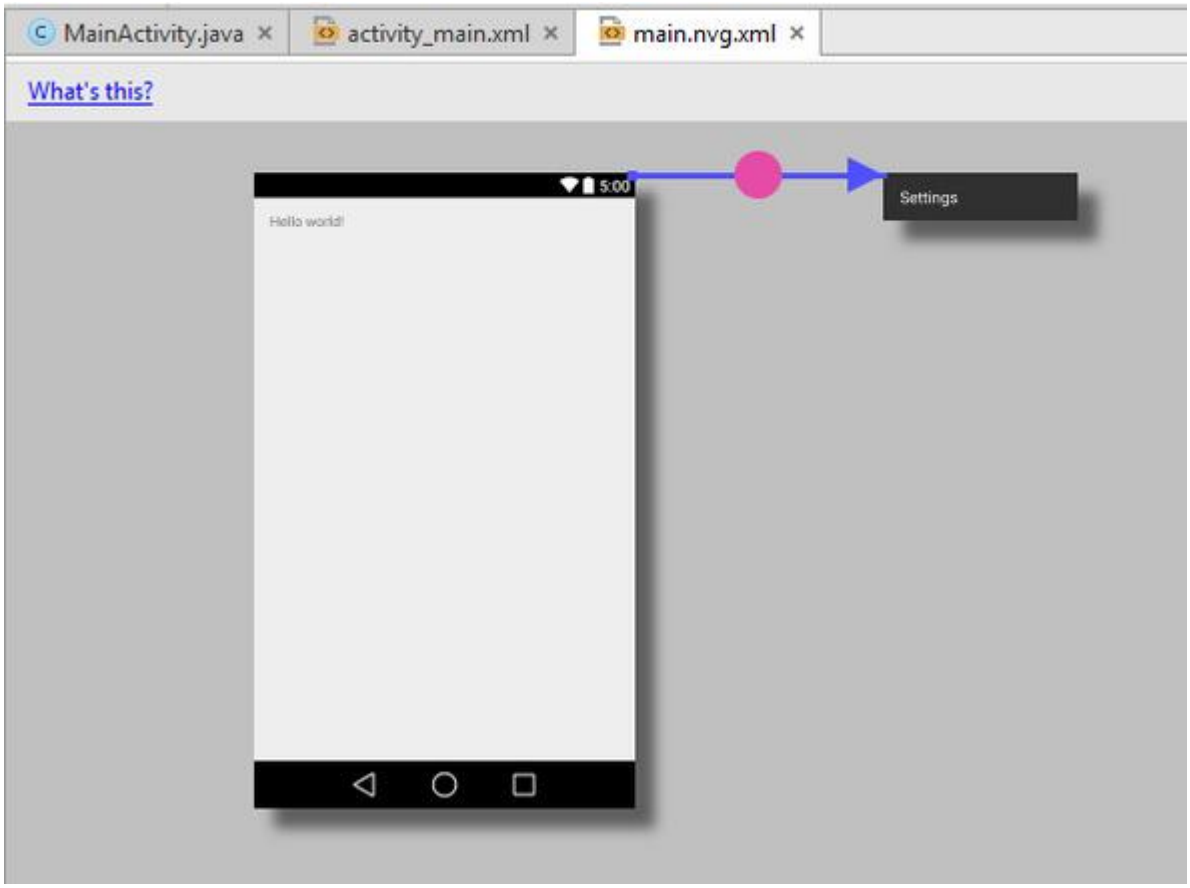
Окно **Structure** среды Android Studio показывает структуру кода исходного файла, открытого в редакторе.

Окно **Captures** среды Android Studio показывает информацию, сохраненную с помощью инструмента DDMS.

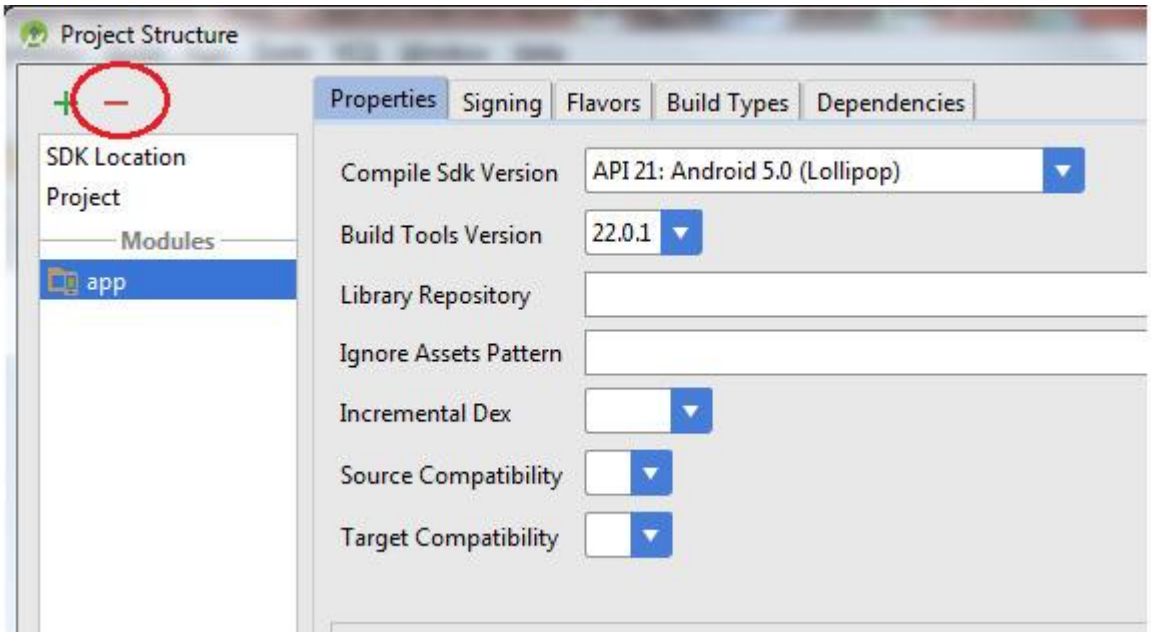
В меню **Tools | Android** с помощью выбора **Android Device Monitor** можно запустить ADM-приложение.

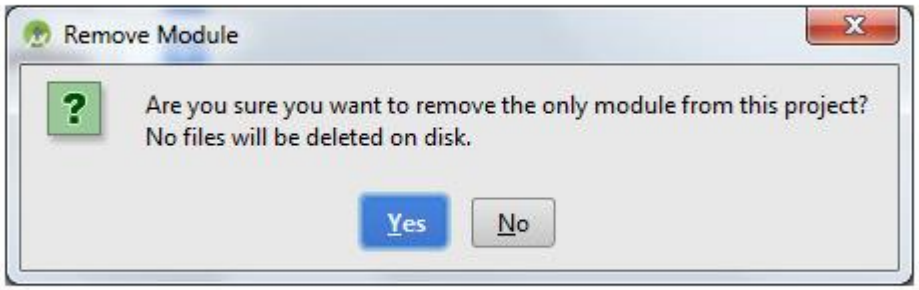


В меню **Tools | Android** с помощью выбора **Navigation Editor** можно открыть окно структуры и компоновки Android-приложения.



Удалить модуль из проекта можно, выбрав меню **File | Project Structure**, затем модуль и нажав кнопку минус.





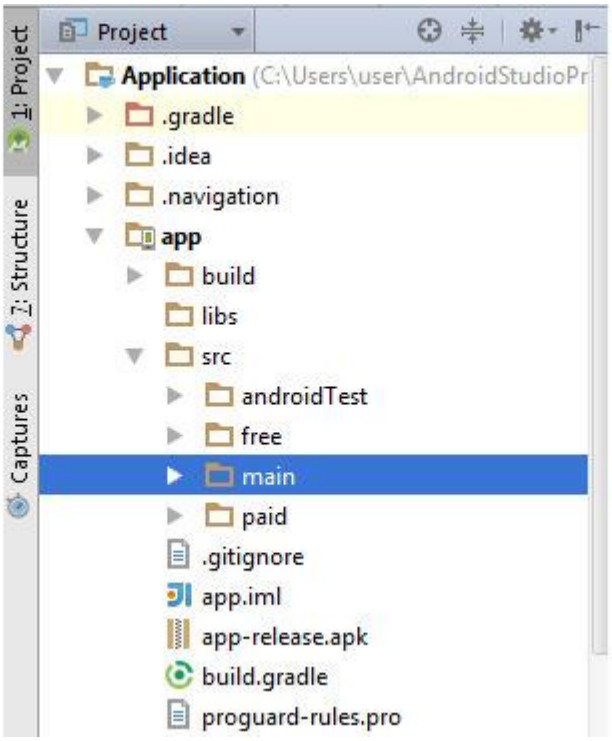
Создание различных версий Android-приложения с помощью Gradle

Система сборки Gradle обеспечивает в рамках одного модуля создание нескольких Android-приложений с разным контентом или создание одного Android-приложения для работы на разных устройствах пользователя или комбинацию этих двух случаев.

В первом случае в Google Play каждое Android-приложение публикуется отдельно, во втором случае разные версии публикуются в рамках одного Android-приложения.

Для создания нескольких Android-приложений в одном модуле в Android Studio откроем окно **Project** и выберем вариант **Project**.

В каталоге src скопируем и вставим папку main под именами free и paid.

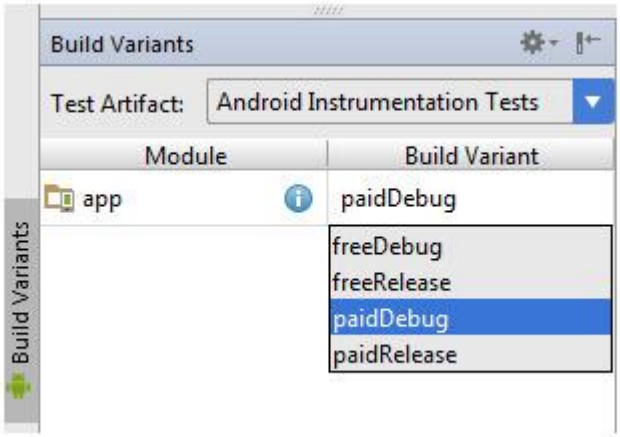


Чтобы не было ошибки сборки, в папке main удалим файл MainActivity. java.
Откроем в редакторе файл сборки build.gradle и добавим код:
productFlavors {
paid {
applicationId «com. application. full»

}

free {
applicationId "com.application.demo»

}
}
Нажмем кнопку **Sync Project with Gradle Files**.
Теперь можно вести отдельно разработку для папки free и для папки paid.
Для запуска конкретного Android-приложения в эмуляторе, откроем окно **Build Variants** и выберем вариант приложения, который хотим запустить.



Нажмем кнопку **Run «app'**.

Для создания одного Android-приложения для работы на разных устройствах пользователя, все версии этого приложения должны иметь один и тот же пакет и быть подписаны одним ключом. Разные версии должны иметь разное значение android: versionCode.

При этом, при публикации, для этого набора APK-файлов создается одно описание, один набор значков, скриншотов, назначается одна цена. Для публикации используется режим **Advanced mode** Google Play.

Если разные версии Android-приложения создаются для разных уровней API, версия с более высоким значением android: minSdkVersion должна иметь более высокое значение android: versionCode. То же самое относится и к увеличению размера экрана для разных версий приложения. При этом значение android: versionName будет одинаковым для всех версий.

Для создания разных версий приложения для разных уровней API, откроем файл сборки модуля build.gradle в редакторе кода и добавим код:

```
productFlavors {
flavor1 {
versionCode 1
minSdkVersion 8
}



flavor2 {
versionCode 2
minSdkVersion 15
}
}
```

Для создания разных версий приложения для устройств с разными экранами, откроем файл сборки модуля build.gradle в редакторе кода и добавим код:

```
productFlavors {
phone {
versionCode 1



}



tablet {
```

versionCode 2

}
}
В каталоге src модуля создадим две папки phone и tablet с файлами AndroidManifest. xml:
<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package=«com. application»>

<supports-screens
android: smallScreens=«true»
android: normalScreens=«true»
android: largeScreens=«false»
android: xlargeScreens=«false»/>

</manifest>
И
<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package=«com. application»>

<supports-screens
android: smallScreens=«false»
android: normalScreens=«false»
android: largeScreens=«true»
android: xlargeScreens=«true»/>

</manifest>

В папках phone и tablet также можно вести разработку приложения для разных устройств.

При сборке APK-файлов, в главный файл AndroidManifest. xml будут добавляться разные элементы <supports-screens>.

Отображение контента Android приложением

Галерея изображений

Самую простую галерею изображений в Android-приложении можно реализовать с помощью компонента android.widget.ImageView и слушателя setOnTouchListener () контейнера ImageView-компонента.

Галерею с более сложной анимацией можно реализовать с помощью компонента android.widget.ViewFlipper.

При реализации с ImageView-компонентом смена изображений осуществляется методом setImageDrawable () и в памяти хранится только одно изображение. В реализации с ViewFlipper-компонентом определяется набор дочерних View-компонентов, переключение между которыми обеспечивается ViewFlipper-компонентом. При этом при большом количестве дочерних View-компонентов может возникнуть ошибка переполнения памяти. Поэтому при

использовании ViewFlipper-компонента при переключении необходимо динамически очищать и добавлять дочерние компоненты для ViewFlipper-компонента.

Другой способ реализации галереи изображений – это применение компонента android.support.v4.view.ViewPager и фрагментов android.support.v4.app.Fragment. Однако при этом требуется минимальная версия 11 для Android API и использование дополнительной библиотеки Android Support Library.

ViewPager-компонент обеспечивает переключение между экранами, динамически создаваемыми с помощью адаптера android.support.v4.app.FragmentStatePagerAdapter, который определяет контент, создавая объекты android.support.v4.app.Fragment.

ImageView

Для создания галереи изображений в главном Activity-компоненте приложения определим показ начальной заставки и панели с кнопками переключения между разделами контента:

```
import java.io.IOException;
import java.io.InputStream;
import android.app.Activity;
import android.content.Intent;
import android.graphics. drawable. Drawable;
import android. os. Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View. OnClickListener;
import android.view.View. OnTouchListener;
import android. widget. Button;
import android.widget.ImageView;
import android.widget.ViewFlipper;



public class MainActivity extends Activity {



private float fromPosition;



@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
ImageView image;
image = (ImageView)findViewById(R.id.imageViewLaunch);
try
{
InputStream ims = getAssets().open("_FRONT.jpg»);
Drawable d = Drawable.createFromStream (ims, null);
image.setImageDrawable (d);
}
catch (IOException ex)
{
return;
}
final ViewFlipper viewFlipper = (ViewFlipper)findViewById(R.id.viewFlipperContents);
```

```
viewFlipper.setOnTouchListener (new OnTouchListener () {
public boolean onTouch (View v, MotionEvent event) {
switch (event.getAction ())
{
case MotionEvent.ACTION_DOWN:
fromPosition = event.getX ();
break;
case MotionEvent.ACTION_UP:
float toPosition = event.getX ();
if (fromPosition> toPosition)
viewFlipper.setDisplayedChild (1);
default:
break;
}
return true;
}
});
final Intent intent = new Intent (this, ContentActivity.class);
final Button btn1= (Button)findViewById(R.id.button1);
btn1.setOnClickListener (new OnClickListener () {
public void onClick (View v) {
String name=«title»;
String value=btn1.getText().toString ();
intent. putExtra (name, value);
startActivity (intent);
}
});



final Button btnn= (Button)findViewById(R.id.buttonn);
btnn.setOnClickListener (new OnClickListener () {
public void onClick (View v) {
String name=«title»;
String value=btnn.getText().toString ();
intent. putExtra (name, value);
startActivity (intent);
}
});



}
@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected (MenuItem item) {
switch (item.getItemId ()) {
case R.id.action_settings:
final Intent intent = new Intent (this, MainActivity.class);
startActivity (intent);
return true;
```

default:
return super. onOptionsItemSelected (item);
}}}

В классе MainActivity в методе onCreate () создается экран, содержимое которого определяется компоновкой activity_main. xml:

<ViewFlipper xmlns: android="http://schemas.android.com/apk/res/android"
android: id="@+id/viewFlipperContents»
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: paddingBottom="@dimen/activity_vertical_margin»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
tools:context=".MainActivity»>

<ImageView
android: id="@+id/imageViewLaunch»
android: layout_width=«fill_parent»
android: layout_height=«fill_parent»
android: contentDescription="@string/desc»/>

<include layout="@layout/contents»/>
</ViewFlipper>

Где ViewFlipper-компонент обеспечивает переключение с начальной заставки ImageView на панель с кнопками, содержимое которой определяется компоновкой contents. xml:

<?xml version=«1.0» encoding=«utf-8»? >

<ScrollView xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«match_parent»
android: layout_height=«match_parent»>

<LinearLayout android: layout_width=«fill_parent»
android: baselineAligned=«false»
android: layout_height=«wrap_content»
android: orientation=«horizontal»>

<LinearLayout android: layout_height=«wrap_content»
android: id="@+id/linearLayout1»
android: orientation=«vertical»
android: layout_width=«0dip»
android: layout_weight=». 8»>

```
<TextView
android: id="@+id/contentsTitle»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/title»
android: textStyle=«bold»
android: textAppearance=»? android: attr/textAppearanceMedium»/>
<Button
android: gravity=«left|center_vertical»
android: id="@+id/button1»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/introduction» />



<Button
android: gravity=«left|center_vertical»
android: id="@+id/buttonn»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/chx» />
</LinearLayout>



<LinearLayout android: layout_width=«0dip»
android: layout_height=«wrap_content»
android: orientation=«horizontal»
android: id="@+id/linearLayout_dummy»
android: layout_weight=». 20»>
</LinearLayout>
</LinearLayout>
</ScrollView>
```

Кнопки панели разделов контента обеспечивают запуск другого Activity-компонента приложения, передавая ему в качестве параметра свой текст, для идентификации содержимого раздела, который нужно отображать:

```
import java.io.IOException;
import java.io.InputStream;
import android.app.Activity;
import android.content.Intent;
import android.graphics. drawable. Drawable;
import android. os. Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.View. OnTouchListener;
import android.widget.ImageView;
import android.widget.LinearLayout;



public class ContentActivity extends Activity {
```

```
private float fromPosition;
private int index;
private LinearLayout linearLayout;
private ImageView image;



@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_content);
String name=«title»;
String value=this.getIntent().getStringExtra (name);
linearLayout= (LinearLayout)findViewById(R.id.layoutContent);
image=(ImageView)findViewById(R.id.imageViewContent);
if (value. equals («Introduction»)) {
this.setTitle («Introduction»);
setContent (1,8,value);
}



if (value. equals («Chaptern»)) {
this.setTitle («Chaptern»);
setContent (1,xxx, value);
}



}
@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected (MenuItem item) {
switch (item.getItemId ()) {
case R.id.action_settings:
final Intent intent = new Intent (this, MainActivity.class);
startActivity (intent);
return true;
default:
return super. onOptionsItemSelected (item);
}}



private void setImage (String path) {
InputStream ims;
try {
ims = getAssets ().open (path);
Drawable d = Drawable.createFromStream (ims, null);
```

```
image.setImageDrawable (d);
} catch (IOException e) {
e.printStackTrace ();
}}

private void setContent (int start, int end, String path) {
final String pathContent=path;
final int endContent=end;
final int startContent=start;
index=startContent;
setImage(pathContent+"/"+index+".jpg»);

linearLayout.setOnTouchListener (new OnTouchListener () {
public boolean onTouch (View v, MotionEvent event) {
switch (event.getAction ())
{
case MotionEvent.ACTION_DOWN:
fromPosition = event.getX ();
break;
case MotionEvent.ACTION_UP:
float toPosition = event.getX ();
if (fromPosition> toPosition) {
index++;
if (index <=endContent) {
setImage(pathContent+"/"+index+".jpg»);
} else {
index=endContent;
}}
else if (fromPosition <toPosition) {
index – ;
if (index> =startContent) {
setImage(pathContent+"/"+index+".jpg»);
} else {
index=startContent;
}}
default:
break;
}
return true;
}
});
}}
```

В классе ContentActivity в методе onCreate () создается экран, содержимое которого определяется компоновкой activity_content. xml:

```
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/layoutContent»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: orientation=«vertical»
android: paddingBottom="@dimen/activity_vertical_margin»
```

```
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
tools:context=".ContentActivity»>



<ImageView
android: id="@+id/imageViewContent»
android: layout_width=«fill_parent»
android: layout_height=«fill_parent»
android: contentDescription="@string/cont»/>



</LinearLayout>
```

ContentActivity-компонент, получая в качестве параметра текст кнопки панели компонента MainActivity, вызывает метод setContent (), в котором определяется переключение между изображениями соответствующей папки каталога assets проекта приложения.

ViewPager

Для создания галереи изображений с помощью ViewPager-компонента, в главном Activity-компоненте приложения также определим показ начальной заставки и панели с кнопками переключения между разделами контента:

```
import java.io.IOException;
import java.io.InputStream;
import java. util. Locale;
import android.app.ActionBar;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.graphics. drawable. Drawable;
import android. os. Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.View. OnClickListener;
import android. widget. Button;
import android.widget.ImageView;

public class MainActivity extends FragmentActivity implements
ActionBar. TabListener {
SectionsPagerAdapter mSectionsPagerAdapter;
ViewPager mViewPager;
@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
```

```
final ActionBar actionBar = getActionBar ();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
mSectionsPagerAdapter = new SectionsPagerAdapter (
getSupportFragmentManager ());
mViewPager = (ViewPager) findViewById(R.id.pager);
mViewPager.setAdapter (mSectionsPagerAdapter);
mViewPager.setOnPageChangeListener (new ViewPager.SimpleOnPageChangeListener () {
@Override
public void onPageSelected (int position) {
actionBar.setSelectedNavigationItem (position);
}
});
for (int i = 0; i <mSectionsPagerAdapter.getCount (); i++) {
actionBar.addTab(actionBar.newTab ()
.setText(mSectionsPagerAdapter.getPageTitle (i))
.setTabListener (this));
}}
@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected (MenuItem item) {
switch (item.getItemId ()) {
case R.id.action_settings:
final Intent intent = new Intent (this, MainActivity.class);
startActivity (intent);
return true;
default:
return super. onOptionsItemSelected (item);
}}
@Override
public void onTabSelected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
mViewPager.setCurrentItem(tab.getPosition ());
}
@Override
public void onTabUnselected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
}
@Override
public void onTabReselected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
}
public class SectionsPagerAdapter extends FragmentStatePagerAdapter {
public SectionsPagerAdapter (FragmentManager fm) {
super (fm);
}
@Override
public Fragment getItem (int position) {
Fragment fragment = new DummySectionFragment ();
Bundle args = new Bundle ();
args.putInt(DummySectionFragment.ARG_SECTION_NUMBER, position +1);
fragment.setArguments (args);
```

```
return fragment;
}
@Override
public int getCount () {
return 2;
}
@Override
public CharSequence getPageTitle (int position) {
Locale l = Locale.getDefault ();
switch (position) {
case 0:
return getString(R.string.title_main_section1).toUpperCase (l);
case 1:
return getString(R.string.title_main_section2).toUpperCase (l);
}
return null;
}}
public static class DummySectionFragment extends Fragment {
public static final String ARG_SECTION_NUMBER = «section_number»;
public DummySectionFragment () {
}
@Override
public View onCreateView (LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
int pos=getArguments().getInt (ARG_SECTION_NUMBER);
View rootView=null;
if (pos==1) {
rootView = inflater.inflate(R.layout.activity_main_launch, container, false);
ImageView image = (ImageView)rootView.findViewById(R.id.imageViewLaunch);
try
{
InputStream ims = rootView.getContext().getAssets().open("_FRONT.jpg»);
Drawable d = Drawable.createFromStream (ims, null);
image.setImageDrawable (d);
}
catch (IOException ex)
{
ex.printStackTrace ();
}}
if (pos==2) {
rootView = inflater.inflate(R.layout.contents, container, false);
final Intent intent = new Intent (getActivity (), ContentActivity.class);
final Button btn1= (Button)rootView.findViewById(R.id.button1);
btn1.setOnClickListener (new OnClickListener () {
public void onClick (View v) {
String nameTitle=«title»;
String valueTitle=btn1.getText().toString ();
intent. putExtra (nameTitle, valueTitle);
String namePages=«pages»;
int valuePages=8;
intent. putExtra (namePages, valuePages);
startActivity (intent);
}
});
```

```
final Button btnn= (Button)rootView.findViewById(R.id.buttonn);
btnn.setOnClickListener (new OnClickListener () {
public void onClick (View v) {
String nameTitle=«title»;
String valueTitle=btnn.getText().toString ();
intent. putExtra (nameTitle, valueTitle);
String namePages=«pages»;
int valuePages=xxx;
intent. putExtra (namePages, valuePages);
startActivity (intent);
}
});

}
return rootView;
}}}
```

В классе MainActivity в методе onCreate () создается экран, содержимое которого определяется компоновкой activity_main. xml:

```
http://schemas.android.com/apk/res/android"ewPager xmlns: android="<android.support.v4.view.Vi
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/pager»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
tools:context=".MainActivity» />
```

SectionsPagerAdapter-адаптер ViewPager-компонента создает объекты android.support.v4.app.Fragment, метод onCreateView () класса которых получает в качестве параметра номер экрана, на основе которого формирует отображаемый контент.

Если пользователь переключается на первый экран – загружается компоновка activity_main_launch. xml и отображается начальная заставка:

```
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
android: id="@+id/viewLaunch»
android: orientation=«vertical»
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: paddingBottom="@dimen/activity_vertical_margin»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
tools:context=".MainActivity»>

<ImageView
android: id="@+id/imageViewLaunch»
android: layout_width=«fill_parent»
android: layout_height=«fill_parent»
android: contentDescription="@string/desc» />
</LinearLayout>
```

Если же пользователь переключается на второй экран – загружается компоновка contents. xml и отображается панель с кнопками переключения между разделами контента:

```
<?xml version=«1.0» encoding=«utf-8»? >

<ScrollView xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«match_parent»
android: layout_height=«match_parent»>
```

```
<LinearLayout android: layout_width=«fill_parent»
android: baselineAligned=«false»
android: layout_height=«wrap_content»
android: orientation=«horizontal»>

<LinearLayout android: layout_height=«wrap_content»
android: id="@+id/linearLayout1»
android: orientation=«vertical»
android: layout_width=«0dip»
android: layout_weight=». 8»>

<TextView
android: id="@+id/contentsTitle»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/title»
android: textStyle=«bold»
android: textAppearance=»? android: attr/textAppearanceMedium»/>
<Button
android: gravity=«left|center_vertical»
android: id="@+id/button1»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/introduction» />

<Button
android: gravity=«left|center_vertical»
android: id="@+id/buttonn»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/chxxx» />
</LinearLayout>

<LinearLayout android: layout_width=«0dip»
android: layout_height=«wrap_content»
android: orientation=«horizontal»
android: id="@+id/linearLayout_dummy»
android: layout_weight=». 20»>
</LinearLayout>
</LinearLayout>
</ScrollView>
```

Кнопки панели разделов контента обеспечивают запуск другого Activity-компонента приложения, передавая ему в качестве параметра свой текст и количество страниц раздела, который нужно отображать:

```
import java.io.IOException;
import java.io.InputStream;
import java. util. Locale;
import android.app.ActionBar;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.graphics. drawable. Drawable;
import android. os. Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
```

```
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class ContentActivity extends FragmentActivity implements
ActionBar. TabListener {
SectionsPagerAdapter mSectionsPagerAdapter;
ViewPager mViewPager;
public static String title;
private int pages;
@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main_content);
title=this.getIntent().getStringExtra («title»);
pages=this.getIntent().getIntExtra («pages», 0);
final ActionBar actionBar = getActionBar ();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
mSectionsPagerAdapter = new SectionsPagerAdapter (
getSupportFragmentManager ());
mViewPager = (ViewPager) findViewById(R.id.pager);
mViewPager.setAdapter (mSectionsPagerAdapter);
mViewPager.setOnPageChangeListener (new ViewPager.SimpleOnPageChangeListener () {
@Override
public void onPageSelected (int position) {
actionBar.setSelectedNavigationItem (position);
}
});
for (int i = 0; i <mSectionsPagerAdapter.getCount (); i++) {
actionBar.addTab(actionBar.newTab ()
.setText(mSectionsPagerAdapter.getPageTitle (i))
.setTabListener (this));
}}
@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected (MenuItem item) {
switch (item.getItemId ()) {
case R.id.action_settings:
final Intent intent = new Intent (this, MainActivity.class);
startActivity (intent);
return true;
default:
return super. onOptionsItemSelected (item);
}}
@Override
public void onTabSelected (ActionBar. Tab tab,
```

```
FragmentTransaction fragmentTransaction) {
mViewPager.setCurrentItem(tab.getPosition ());
}
@Override
public void onTabUnselected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
}
@Override
public void onTabReselected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
}
public class SectionsPagerAdapter extends FragmentStatePagerAdapter {
public SectionsPagerAdapter (FragmentManager fm) {
super (fm);
}
@Override
public Fragment getItem (int position) {
Fragment fragment = new DummySectionFragment ();
Bundle args = new Bundle ();
args.putInt(DummySectionFragment.ARG_SECTION_NUMBER, position +1);
fragment.setArguments (args);
return fragment;
}
@Override
public int getCount () {
return pages;
}
@Override
public CharSequence getPageTitle (int position) {
Locale l = Locale.getDefault ();
String str=title+" page: "+position;
return str.toUpperCase (l);
}}
public static class DummySectionFragment extends Fragment {
public static final String ARG_SECTION_NUMBER = «section_number»;
public DummySectionFragment () {
}
@Override
public View onCreateView (LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
int pos=getArguments().getInt (ARG_SECTION_NUMBER);
View rootView = inflater.inflate(R.layout.activity_content, container, false);
ImageView image = (ImageView)rootView.findViewById(R.id.imageViewContent);
try
{
String path=«»;
if (ContentActivity. title. equals («Introduction»)) path=«intro»;

if (ContentActivity. title. equals («Chaptern»)) path=«chn»;
InputStream ims = rootView.getContext().getAssets().open(path+"/"+pos+".jpg»);
Drawable d = Drawable.createFromStream (ims, null);
image.setImageDrawable (d);
}
catch (IOException ex)
{
```

```
ex.printStackTrace ();
}
return rootView;
}}}
```

В классе ContentActivity в методе onCreate () создается экран, содержимое которого определяется компоновкой activity_main_content. xml:

```
http://schemas.android.com/apk/res/android"ewPager xmlns: android="<android.support.v4.view.Vi
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/pager»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
tools:context=".ContentActivity» />
```

SectionsPagerAdapter-адаптер ViewPager-компонента создает объекты android.support.v4.app.Fragment, метод onCreateView () класса которых получает в качестве параметра номер экрана, на основе которого формируется отображаемый контент. При этом содержимое экрана определяется компоновкой activity_content. xml:

```
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/layoutContent»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: orientation=«vertical»
android: paddingBottom="@dimen/activity_vertical_margin»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
tools:context=".ContentActivity»>

<ImageView
android: id="@+id/imageViewContent»
android: layout_width=«fill_parent»
android: layout_height=«fill_parent»
android: contentDescription="@string/cont»/>

</LinearLayout>
```

В методе onCreateView () класса DummySectionFragment динамически изменяется содержимое компонента ImageView на основе названия раздела контента и номера экрана, обеспечивая переключение между изображениями соответствующей папки каталога assets проекта приложения.

ImageView + Zoom + Scroll

Стандартный компонент ImageView платформы Android не позволяет прикосновением к экрану увеличить изображение и после этого прокручивать его.

Для добавления опции увеличения и прокручивания изображения можно воспользоваться сторонними библиотеками, такими как https://github.com/chrisbanes/PhotoView.

Для использования библиотеки PhotoView скачаем проект и с помощью инструмента Maven соберем JAR-файл библиотеки, который поместим в каталог libs проекта приложения.

Теперь в исходном коде приложения ImageView-компонент можно добавить в контейнер uk.co.senab.photoview.PhotoViewAttacher, который будет обеспечивать увеличение и прокрутку изображения:

```
private PhotoViewAttacher mAttacher;
private ImageView image;

image=(ImageView)findViewById(R.id.imageViewContent);
mAttacher = new PhotoViewAttacher (image);
```

```
image.setImageDrawable (d);
mAttacher. update ();
```

Галерея HTML контента

Простое поэкранное отображение HTML-контента можно организовать с помощью компонента android. webkit. WebView и слушателя setOnTouchListener () контейнера WebView-компонента. Более сложную анимацию добавит компонент android.widget.ViewFlipper, а более удобное переключение между экранами – компонент android.support.v4.view.ViewPager.

Для создания галереи HTML-контента с помощью ViewPager-компонента, в главном Activity-компоненте приложения определим показ начальной заставки и панели с кнопками переключения между разделами контента:

```
import java.io.IOException;
import java.io.InputStream;
import java. util. Locale;
import android.app.ActionBar;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.graphics. drawable. Drawable;
import android. os. Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.View. OnClickListener;
import android.view.ViewGroup;
import android. widget. Button;
import android.widget.ImageView;

public class MainActivity extends FragmentActivity implements ActionBar. TabListener {
SectionsPagerAdapter mSectionsPagerAdapter;
ViewPager mViewPager;
@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
final ActionBar actionBar = getActionBar ();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
mSectionsPagerAdapter = new SectionsPagerAdapter (getSupportFragmentManager ());
mViewPager = (ViewPager) findViewById(R.id.pager);
mViewPager.setAdapter (mSectionsPagerAdapter);
mViewPager.setOnPageChangeListener (new ViewPager.SimpleOnPageChangeListener () {
@Override
public void onPageSelected (int position) {
actionBar.setSelectedNavigationItem (position);
}
});
for (int i = 0; i <mSectionsPagerAdapter.getCount (); i++) {
actionBar.addTab (
actionBar.newTab ()
.setText(mSectionsPagerAdapter.getPageTitle (i))
.setTabListener (this));
```

```
}}
@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public void onTabSelected (ActionBar. Tab tab, FragmentTransaction fragmentTransaction) {
mViewPager.setCurrentItem(tab.getPosition ());
}
@Override
public void onTabUnselected (ActionBar. Tab tab, FragmentTransaction fragmentTransaction) {
}
@Override
public void onTabReselected (ActionBar. Tab tab, FragmentTransaction fragmentTransaction) {
}
public class SectionsPagerAdapter extends FragmentStatePagerAdapter {
public SectionsPagerAdapter (FragmentManager fm) {
super (fm);
}
@Override
public Fragment getItem (int position) {
Fragment fragment = new DummySectionFragment ();
Bundle args = new Bundle ();
args.putInt(DummySectionFragment.ARG_SECTION_NUMBER, position +1);
fragment.setArguments (args);
return fragment;
}
@Override
public int getCount () {
return 2;
}
@Override
public CharSequence getPageTitle (int position) {
Locale l = Locale.getDefault ();
switch (position) {
case 0:
return getString(R.string.title_main_section1).toUpperCase (l);
case 1:
return getString(R.string.title_main_section2).toUpperCase (l);
}
return null;
}}
public static class DummySectionFragment extends Fragment {
public static final String ARG_SECTION_NUMBER = «section_number»;
public DummySectionFragment () {
}
@Override
public View onCreateView (LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
int pos=getArguments().getInt (ARG_SECTION_NUMBER);
View rootView=null;
if (pos==1) {
rootView = inflater.inflate(R.layout.activity_main_launch, container, false);
ImageView image = (ImageView)rootView.findViewById(R.id.imageViewLaunch);
```

```
try
{
InputStream ims = rootView.getContext().getAssets().open("_FRONT.jpg»);
Drawable d = Drawable.createFromStream (ims, null);
image.setImageDrawable (d);
}
catch (IOException ex)
{
ex.printStackTrace ();
}}
if (pos==2) {
rootView = inflater.inflate(R.layout.contents, container, false);
final Intent intent = new Intent (getActivity (), ContentActivity.class);
final Button btn1= (Button)rootView.findViewById(R.id.button1);
btn1.setOnClickListener (new OnClickListener () {
public void onClick (View v) {
String name=«position»;
int value=0;
intent. putExtra (name, value);
startActivity (intent);
}
});

final Button btnn= (Button)rootView.findViewById(R.id.buttonn);
btnn.setOnClickListener (new OnClickListener () {
public void onClick (View v) {
String name=«position»;
int value=xxx;
intent. putExtra (name, value);
startActivity (intent);
}
});
}
return rootView;
}}}
```

В классе MainActivity в методе onCreate () создается экран, содержимое которого определяется компоновкой activity_main. xml:

```
http://schemas.android.com/apk/res/android"ewPager xmlns: android="<android.support.v4.view.Vi
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/pager»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
tools:context=".MainActivity» />
```

SectionsPagerAdapter-адаптер ViewPager-компонента создает объекты android.support.v4.app.Fragment, метод onCreateView () класса которых получает в качестве параметра номер экрана, на основе которого формирует отображаемый контент.

Если пользователь переключается на первый экран – загружается компоновка activity_main_launch. xml и отображается начальная заставка:

```
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
android: id="@+id/viewLaunch»
android: orientation=«vertical»
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: paddingBottom="@dimen/activity_vertical_margin»
```

```
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
tools:context=".MainActivity»>

<ImageView
android: id="@+id/imageViewLaunch»
android: layout_width=«fill_parent»
android: layout_height=«fill_parent»
android: contentDescription="@string/desc» />
</LinearLayout>
```

Если же пользователь переключается на второй экран – загружается компоновка contents. xml и отображается панель с кнопками переключения между разделами контента:

```
<?xml version=«1.0» encoding=«utf-8»? >

<ScrollView xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«match_parent»
android: layout_height=«match_parent»>

<LinearLayout android: layout_width=«fill_parent»
android: baselineAligned=«false»
android: layout_height=«wrap_content»
android: orientation=«horizontal»>

<LinearLayout android: layout_height=«wrap_content»
android: id="@+id/linearLayout1»
android: orientation=«vertical»
android: layout_width=«0dip»
android: layout_weight=». 8»>

<TextView
android: id="@+id/contentsTitle»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/title»
android: textStyle=«bold»
android: textAppearance=»? android: attr/textAppearanceMedium»/>
<Button
android: gravity=«left|center_vertical»
android: id="@+id/button1»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/introduction» />

<Button
android: gravity=«left|center_vertical»
android: id="@+id/buttonn»
android: layout_width=«fill_parent»
android: layout_height=«wrap_content»
android: text="@string/chxxx» />
</LinearLayout>

<LinearLayout android: layout_width=«0dip»
android: layout_height=«wrap_content»
android: orientation=«horizontal»
```

```
android: id="@+id/linearLayout_dummy»
android: layout_weight=». 20»>
</LinearLayout>
</LinearLayout>
</ScrollView>
```

Кнопки панели разделов контента обеспечивают запуск другого компонента ContentActivity приложения, передавая ему в качестве параметра номер раздела контента, который необходимо отображать:

```
import java. util. Locale;
import android.app.ActionBar;
import android.app.FragmentTransaction;
import android.content.Intent;
import android. os. Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentStatePagerAdapter;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android. webkit. WebView;

public class ContentActivity extends FragmentActivity implements
ActionBar. TabListener {
SectionsPagerAdapter mSectionsPagerAdapter;
ViewPager mViewPager;
private int position;
@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main_content);
position=this.getIntent().getIntExtra («position», 0);
final ActionBar actionBar = getActionBar ();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
mSectionsPagerAdapter = new SectionsPagerAdapter (
getSupportFragmentManager ());
mViewPager = (ViewPager) findViewById(R.id.pager);
mViewPager.setAdapter (mSectionsPagerAdapter);
mViewPager.setOnPageChangeListener (new ViewPager.SimpleOnPageChangeListener () {
@Override
public void onPageSelected (int position) {
actionBar.setSelectedNavigationItem (position);
}
});
for (int i = 0; i <mSectionsPagerAdapter.getCount (); i++) {
actionBar.addTab(actionBar.newTab ()
.setText(mSectionsPagerAdapter.getPageTitle (i))
.setTabListener (this));
}
mViewPager.setCurrentItem (position);
}
@Override
```

```
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected (MenuItem item) {
switch (item.getItemId ()) {
case R.id.action_settings:
final Intent intent = new Intent (this, MainActivity.class);
startActivity (intent);
return true;
default:
return super. onOptionsItemSelected (item);
}}
@Override
public void onTabSelected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
mViewPager.setCurrentItem(tab.getPosition ());
}
@Override
public void onTabUnselected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
}
@Override
public void onTabReselected (ActionBar. Tab tab,
FragmentTransaction fragmentTransaction) {
}
public class SectionsPagerAdapter extends FragmentStatePagerAdapter {
public SectionsPagerAdapter (FragmentManager fm) {
super (fm);
}
@Override
public Fragment getItem (int position) {
Fragment fragment = new DummySectionFragment ();
Bundle args = new Bundle ();
args.putInt(DummySectionFragment.ARG_SECTION_NUMBER, position +1);
fragment.setArguments (args);
return fragment;
}
@Override
public int getCount () {
return xxx;
}
@Override
public CharSequence getPageTitle (int position) {
Locale l = Locale.getDefault ();
switch (position) {
case 0:
return getString(R.string.introduction).toUpperCase (l);

case 17:
return getString(R.string.ch17).toUpperCase (l);
}
return null;
}}
```

```
public static class DummySectionFragment extends Fragment {
public static final String ARG_SECTION_NUMBER = «section_number»;
public DummySectionFragment () {
}
@Override
public View onCreateView (LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
int pos=getArguments().getInt (ARG_SECTION_NUMBER) -1;
View rootView = inflater.inflate(R.layout.activity_content, container, false);
WebView myWebView = (WebView) rootView.findViewById(R.id.webview);
myWebView.loadUrl("file:///android_asset/page"+pos+".html»);
return rootView;
}}}
```

В классе ContentActivity в методе onCreate () создается экран, содержимое которого определяется компоновкой activity_main_content. xml:

```
http://schemas.android.com/apk/res/android"ewPager xmlns: android="<android.support.v4.view.Vi
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/pager»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
tools:context=".ContentActivity» />
```

SectionsPagerAdapter-адаптер ViewPager-компонента создает объекты android.support.v4.app.Fragment, метод onCreateView () класса которых получает в качестве параметра номер экрана, на основе которого формируется отображаемый контент. При этом содержимое экрана определяется компоновкой activity_content. xml:

```
<?xml version=«1.0» encoding=«utf-8»? >
<WebView xmlns: android="http://schemas.android.com/apk/res/android"
android: id="@+id/webview»
android: layout_width=«fill_parent»
android: layout_height=«fill_parent» />
```

В методе onCreateView () класса DummySectionFragment динамически изменяется содержимое компонента WebView на основе номера экрана и соответственно номера раздела контента, обеспечивая переключение между HTML-страницами соответствующей папки каталога assets проекта приложения.

WebView + Pagination

При большом размере HTML-страницы, отображаемой WebView-компонентом появляется проблема постраничного отображения HTML-страницы.

Так как компонент WebView основан на движке WebKit, он поддерживает различные JavaScript-библиотеки, в частности библиотеку JQuery и ее плагины.

Для отображения HTML-страницы в виде электронной книги можно воспользоваться различными JQuery-плагинами, например плагином BookBlock (http://ru.tmsoftstudio.com/file/page/applications/ru/jquery-book-reader.html).

При совместном использовании JavaScript-библиотек и WebView-компонента нужно учитывать, что загрузка и выполнение JavaScript-кода требует дополнительных ресурсов и времени, поэтому Android-приложение может стать менее отзывчивым на действия пользователя.

Для того чтобы WebView-компонент выполнял JavaScript-код, в Android-приложение необходимо добавить следующий код:

```
WebView myWebView = (WebView) rootView.findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings ();
webSettings.setJavaScriptEnabled (true);
myWebView.loadUrl("file:///android_asset/page.html»);
```

Для постраничного отображения HTML-страницы скачаем JQuery-плагин BookBlock и применим к HTML-странице следующий шаблон:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=«UTF-8»>
<title> </title>
<link rel=«stylesheet» type=«text/css» href=«css/bookblock. css» />
<script src=«js/jquery. min. js»> </script>
<script src=«js/pagination. js»> </script>
<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/jquery.bookblock. min.
js»> </script>
</head>
<body>
<div class=«container»>
<div class=«main»>
<div class=«bb-custom-wrapper»>
<H1> </H1>
<div id=«bb-bookblock» class=«bb-bookblock»>
<! – Content – >
</div>
<nav>
<div>
<a id=«bb-nav-prev» href=«#» class=«bb-custom-icon bb-custom-icon-arrow-left»> Previous </a>
<a id=«bb-nav-next» href=«#» class=«bb-custom-icon bb-custom-icon-arrow-right»> Next </a>
</div>
<div>
<a id=«bb-nav-first» href=«#» class=«bb-custom-icon bb-custom-icon-first»> First page </a>
<a id=«bb-nav-last» href=«#» class=«bb-custom-icon bb-custom-icon-last»> Last page </a>
</div>
</nav>
</div>
</div>
</div>
</body>
</html>
```

Отображение электронных книг в формате ePub

Для создания Android EPub приложений существуют как платные, так и бесплатные библиотеки EPub SDK.

В качестве примера рассмотрим использование библиотеки AnFengde EPUB SDK (http://epub.anfengde.com/).

Для интеграции библиотеки с проектом Android-приложения скачаем и распакуем ZIP-архив библиотеки и импортируем проект EPUB_SDK-master\android\lib\EPUB_UI в Workspace-пространство, используя команду **File | Import | Existing Android Code Into Workspace** среды Eclipse.

В окне **Project Explorer** среды Eclipse нажмем правой кнопкой мышки на узле Android-приложения и в контекстном меню выберем команду **Properties**. В разделе **Android | Library** кнопкой **Add** добавим проект EPUB_UI и нажмем кнопку **OK**.

В файл AndroidManifest. xml Android-приложения добавим разрешения и Activity-компонент:

```
<uses-permission android:name="android.permission.INTERNET»/>
<uses-permission android:name="android.permission. WRITE_EXTERNAL_STORAGE»/>
```

<uses-permission android:name="android.permission. WRITE_SETTINGS»/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE»/>
<uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS»/>

<activity android:name="com.google.ads.AdActivity»
android:
configChanges=«keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreen Size»/>
</application>
Компоновку activity_main. xml компонента MainActivity определим как:
http://schemas.android.com/apk/res/android"okView xmlns: android="<com.anfengde.epub.ui.Bo
android: id="@+id/bookView»
android: layout_width=«match_parent»
android: layout_height=«fill_parent»>
</com.anfengde.epub.ui.BookView>
В метод onCreate () компонента MainActivity добавим следующий код:

```
import com.anfengde.epub.core.value.Constants;
import com.anfengde.epub.ui.BookView;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
BookView bookView = (BookView) findViewById(R.id.bookView);
bookView.setPath(Constants.CACHE_PAHT);
bookView.initBook ();
bookView. openShelf ();
}
```

Теперь при запуске Android-приложения будет загружаться оболочка Book Shelf, отображающая EPub-книги, расположенные в папке *assets\books* приложения.

Для установки значка Android-приложения необходимо изменить файлы ic_launcher.png проекта EPUB_UI.

Для повторной сборки и развертывания Android-приложения нужно очистить папку *bin* проекта Android-приложения и папку *sdcard/epub* устройства.

Оболочка Book Shelf формируется кодом файла *raw/jindex. js* библиотеки EPUB_UI. Для изменения размеров значка книги в оболочке Book Shelf нужно изменить CSS-стили *#container #afd_books ul* в файле *raw/sindex. css* библиотеки EPUB_UI.

Отображение PDF

Для отображения в Android-приложении PDF-контента существуют как платные, так и бесплатные библиотеки PDF SDK.

В качестве примера рассмотрим использование библиотеки Android-Pdf-Viewer-Library (https://github.com/jblough/Android-Pdf-Viewer-Library).

Для интеграции библиотеки с проектом Android-приложения скачаем и распакуем ZIP-архив библиотеки и скопируем файл PdfViewer. jar в папку libs проекта.

Создадим Activity-компонент, расширяющий класс net.sf.andpdf. pdfviewer. PdfViewerActivity:

```
public class PDFActivity extends net.sf.andpdf. pdfviewer. PdfViewerActivity {

public int getPreviousPageImageResource () {return R. drawable. left_arrow;}
public int getNextPageImageResource () {return R. drawable. right_arrow;}
public int getZoomInImageResource () {return R. drawable. zoom_in;}
public int getZoomOutImageResource () {return R. drawable. zoom_out;}
public int getPdfPasswordLayoutResource () {return R.layout. pdf_file_password;}
public int getPdfPageNumberResource () {return R.layout. dialog_pagenumber;}
```

```
public int getPdfPasswordEditField () {return R.id.etPassword;}
public int getPdfPasswordOkButton () {return R.id.btOK;}
public int getPdfPasswordExitButton () {return R.id.btExit;}
public int getPdfPageNumberEditField () {return R.id.pagenum_edit;}
}
```

Скопируем файлы left_arrow.png, right_arrow.png, zoom_in.png, zoom_out.png из папки res/drawable библиотеки в папку res/drawable проекта приложения.

Скопируем файлы dialog_pagenumber. xml, pdf_file_password. xml из папки res/layout библиотеки в папку res/layout проекта приложения.

В файле AndroidManifest. xml проекта приложения добавим разрешение:

```
<uses-permission android:name="android.permission. WRITE_EXTERNAL_STORAGE» />
```

Поместим PDF-файл в папку assets проекта приложения.

Изменим код метода onCreate () основного Activity-компонента приложения:

```
@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
File sdCard = Environment.getExternalStorageDirectory ();
File dir = new File (sdCard.getAbsolutePath () + "/pdf»);
dir.mkdirs ();
File f = new File (dir,«book. pdf»);
if (!f. exists ()) try {
InputStream is = getAssets ().open («book. pdf»);
int size = is. available ();
byte [] buffer = new byte [size];
is.read (buffer);
is.close ();
FileOutputStream fos = new FileOutputStream (f);
fos. write (buffer);
fos.close ();
} catch (Exception e) {throw new RuntimeException (e);}
String path = f.getPath ();
Intent intent = new Intent (this, PDFActivity.class);
intent. putExtra (PDFActivity. EXTRA_PDFFILENAME, path);
startActivity (intent);
}
```

В методе onCreate () основного Activity-компонента приложения PDF-файл копируется из папки assets в каталог pdf sdcard-карты, после чего вызывается Activity-компонент PDFActivity, обеспечивающий отображение скопированного PDF-файла.

Некоторые рецепты Android

Activity + IntentService + BroadcastReceiver

IntentService не может обрабатывать несколько запросов одновременно, он обрабатывает их последовательно, в порядке очереди.

В Android Studio выберем **New | Service | Service (IntentService)**.

В результате будет создан класс сервиса, код которого модифицируем:

```
package com. application;

import android.app.IntentService;
import android.content.Intent;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import org. json. JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net. URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class WebIntentService extends IntentService {
private static final String EXTRA_PARAM = "com.application.extra.PARAM»;
public static final String RESPONSE_STRING = «WebResponse»;

public static void startAction (Context context, String param) {
Intent intent = new Intent (context, WebIntentService.class);
intent. putExtra (EXTRA_PARAM, param);
context.startService (intent);
}

public WebIntentService () {
super («WebIntentService»);
}

@Override
protected void onHandleIntent (Intent intent) {
if (intent!= null) {
final String param = intent.getStringExtra (EXTRA_PARAM);
handleAction (param);

}
}

private void handleAction (String param) {
ConnectivityManager connMgr =
(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo ();
if (networkInfo!= null && networkInfo.isConnected () && networkInfo.isAvailable ()) {
InputStream is = null;
URL url = null;
try {
url = new URL (param);
HttpURLConnection conn = (HttpURLConnection) url. openConnection ();
conn.setReadTimeout (10000 /* milliseconds */);
conn.setConnectTimeout (15000 /* milliseconds */);
conn.setRequestMethod («GET»);
conn.setDoInput (true);
conn.connect ();
int response = conn.getResponseCode ();
is = conn.getInputStream ();
// Convert the InputStream into a string
BufferedReader br = null;
StringBuilder sb = new StringBuilder ();
String line;
br = new BufferedReader (new InputStreamReader (is));
while ((line = br.readLine ())!= null) {
sb. append (line);
}
String contentAsString=sb.toString ();
```

```
is.close ();
conn. disconnect ();

JSONObject reader = new JSONObject (contentAsString);
JSONObject main = reader.getJSONObject («main»);
String temp = main.getString («temp»);
Calendar c = Calendar.getInstance ();
SimpleDateFormat df = new SimpleDateFormat («yyyy-MM-dd HH: mm: ss»);
String formattedDate = df.format(c.getTime ());
String result=formattedDate + " temp: " +temp;
Intent broadcastIntent = new Intent ();
broadcastIntent.setAction(MainActivity.WebReceiver.PROCESS_RESPONSE);
broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
broadcastIntent. putExtra (RESPONSE_STRING, result);
sendBroadcast (broadcastIntent);

Context context = this;
SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY",Context.MODE_PRIVAT
E);
SharedPreferences. Editor editor = sharedPref. edit ();
editor. putString («temp», temp);
editor. putString («date», formattedDate);
editor.commit ();
} catch (Exception e) {
e.printStackTrace ();
}
} else {
Context context = this;
SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY»,
Context.MODE_PRIVATE);
String date=sharedPref.getString («date», «unknown date»);
String temp=sharedPref.getString («temp», «unknown temp»);
String result=date + " temp: " +temp;
Intent broadcastIntent = new Intent ();
broadcastIntent.setAction(MainActivity.WebReceiver.PROCESS_RESPONSE);
broadcastIntent.addCategory(Intent.CATEGORY_DEFAULT);
broadcastIntent. putExtra (RESPONSE_STRING, result);
sendBroadcast (broadcastIntent);
}
}
}
```

Здесь, при запуске сервиса, проверяется наличие соединения с Интернетом, затем получаются данные в JSON-формате по указанному URL адресу.

Далее необходимые данные извлекаются и отправляются BroadcastReceiver в виде строки.

Извлеченные данные записываются в настройки SharedPreferences.

Если же соединение с Интернетом отсутствует, тогда данные извлекаются из настроек SharedPreferences и отправляются BroadcastReceiver в виде строки.

Для отключения соединения с Интернетом в эмуляторе можно нажать F8.

В MainActivity создадим встроенный класс BroadcastReceiver:

```
package com. application;

import android.content.BroadcastReceiver;
import android.content.Context;
```

```
import android.content.Intent;
import android.content.IntentFilter;
import android. os. Bundle;
import android.support. v7.app. AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android. widget. TextView;

public class MainActivity extends AppCompatActivity {

private WebReceiver receiver;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);

IntentFilter filter = new IntentFilter(WebReceiver.PROCESS_RESPONSE);
filter.addCategory(Intent.CATEGORY_DEFAULT);
receiver = new WebReceiver ();
registerReceiver (receiver, filter);
http://api.openweathermap.org/data/2.5/weather?q=London,uk");is,
"WebIntentService.startAction(MainActivity.th
}

@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {
int id = item.getItemId ();
if (id == R.id.action_settings) {
return true;
}
return super. onOptionsItemSelected (item);
}

public class WebReceiver extends BroadcastReceiver {
public static final String PROCESS_RESPONSE =«com. WebReceiver»;
public WebReceiver () {}

@Override
public void onReceive (Context context, Intent intent) {
String responseString = intent.getStringExtra(WebIntentService.RESPONSE_STRING);
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (responseString);
}
}
}
```

Здесь в методе onCreate содается и регистрируется экземпляр внутреннего класса BroadcastReceiver.

Кроме того, запускается сервис, получающий данные по указанному URL адресу.

BroadcastReceiver получает данные в виде строки и отображает их в TextView.

AndroidManifest. xml будет иметь код:

```
<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package=«com. application»>

<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />

<application
android: allowBackup=«true»
android: icon="@mipmap/ic_launcher»
android: label="@string/app_name»
android: theme="@style/AppTheme»>
<activity
android:name=".MainActivity»
android: label="@string/app_name»>
<intent-filter>
<action android:name="android.intent.action.MAIN» />

<category android:name="android.intent.category.LAUNCHER» />
</intent-filter>
</activity>

<service
android: name=». WebIntentService»
android: exported=«false»>
</service>

</application>
</manifest>
```

Файл компоновки:

```
<RelativeLayout xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: tools="http://schemas.android.com/tools" android: layout_width=«match_parent»
android: layout_height=«match_parent» android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: paddingBottom="@dimen/activity_vertical_margin» tools:context=".MainActivity»>

<TextView
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: id="@+id/textView»/>
</RelativeLayout>
```

Для запуска сервиса в отдельном процессе, в тэге <service> файла AndroidManifest. xml атрибут android: process будет иметь значение, начинающееся с двоеточия:

```
android: process=":new_application_process»
```

Для ограничения доступа другим приложениям к сервису, в тэге <service> файла AndroidManifest. xml атрибут android: exported будет иметь значение false:

```
android: exported=«false»
```

Чтобы предотвратить остановку сервиса системой при нехватке памяти, можно вызвать метод startForeground () в методе обратного вызова onStartCommand ().

Для автоматического запуска сервиса при перезагрузке телефона, можно создать BroadcastReceiver для запуска сервиса:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED» />
<receiver android:name="com.example.StartServiceReceiver»>
<intent-filter>
<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED»/>
<action android:name="android.intent.action. QUICKBOOT_POWERON» />
</intent-filter>
</intent-filter>
public class StartServiceReceiver extends BroadcastReceiver {
@Override
public void onReceive (Context context, Intent intent)
{
Intent serviceIntent = new Intent (context, StartBootService.class);
context.startService (serviceIntent);
}
}
```

Activity + IntentService + createPendingResult

Создание объекта PendingIntent методом createPendingResult активности это другой способ получения данных из сервиса для их обработки в методе обратного вызова onActivityResult активности.

Код IntentService при этом будет иметь следующий вид:

```
package com. application;

import android.app.IntentService;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import org. json. JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net. URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class WebIntentService extends IntentService {

public WebIntentService () {
super («WebIntentService»);
}

@Override
protected void onHandleIntent (Intent intent) {
if (intent!= null) {
final String param = intent.getStringExtra (MainActivity. EXTRA_PARAM);
PendingIntent pi = intent.getParcelableExtra(MainActivity.PARAM_PINTENT);
handleAction (param, pi);
}
}
```

```
    /**
    * Handle action in the provided background thread with the provided
    * parameters.
    */
    private void handleAction (String param, PendingIntent pi) {
    ConnectivityManager connMgr =
(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo ();
    if (networkInfo!= null && networkInfo.isConnected () && networkInfo.isAvailable ()) {
    InputStream is = null;
    URL url = null;
    try {
    url = new URL (param);
    HttpURLConnection conn = (HttpURLConnection) url. openConnection ();
    conn.setReadTimeout (10000 /* milliseconds */);
    conn.setConnectTimeout (15000 /* milliseconds */);
    conn.setRequestMethod («GET»);
    conn.setDoInput (true);
    conn.connect ();
    int response = conn.getResponseCode ();
    is = conn.getInputStream ();
    // Convert the InputStream into a string
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder ();
    String line;
    br = new BufferedReader (new InputStreamReader (is));
    while ((line = br.readLine ())!= null) {
    sb. append (line);
    }
    String contentAsString=sb.toString ();
    is.close ();
    conn. disconnect ();

    JSONObject reader = new JSONObject (contentAsString);
    JSONObject main = reader.getJSONObject («main»);
    String temp = main.getString («temp»);
    Calendar c = Calendar.getInstance ();
    SimpleDateFormat df = new SimpleDateFormat («yyyy-MM-dd HH: mm: ss»);
    String formattedDate = df.format(c.getTime ());
    String result=formattedDate + " temp: " +temp;

    Intent intent = new Intent().putExtra(MainActivity.PARAM_RESULT, result);
    pi.send(WebIntentService.this,MainActivity.RESULT_CODE, intent);

    Context context = this;
    SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY",Context.MODE_PRIVAT
E);
    SharedPreferences. Editor editor = sharedPref. edit ();
    editor. putString («temp», temp);
    editor. putString («date», formattedDate);
    editor.commit ();
    } catch (Exception e) {
    e.printStackTrace ();
    }
```

```
} else {
Context context = this;
SharedPreferences sharedPref = context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY», Context.MODE_PRIVATE);
String date=sharedPref.getString («date», «unknown date»);
String temp=sharedPref.getString («temp», «unknown temp»);
String result=date + " temp: " +temp;
Intent intent = new Intent().putExtra(MainActivity.PARAM_RESULT, result);
try {
pi.send(WebIntentService.this,MainActivity.RESULT_CODE, intent);
} catch (PendingIntent.CanceledException e) {
e.printStackTrace ();
}
}
}
}
```

Здесь, при запуске сервиса, проверяется наличие соединения с Интернетом, затем получаются данные в JSON-формате по указанному URL адресу.

Далее необходимые данные извлекаются и передаются в объект PendingIntent в виде строки.

Объект PendingIntent отправляется обратно в создавшую его активность методом send.

Извлеченные данные записываются в настройки SharedPreferences.

Если же соединение с Интернетом отсутствует, тогда данные извлекаются из настроек SharedPreferences и передаются в объект PendingIntent в виде строки.

Для отключения соединения с Интернетом в эмуляторе можно нажать F8.

Активность, создавшая объект PendingIntent, будет иметь следующий код:

```
package com. application;

import android.app.PendingIntent;
import android.content.Intent;
import android. os. Bundle;
import android.support. v7.app. AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android. widget. TextView;

public class MainActivity extends AppCompatActivity {

public static final String EXTRA_PARAM = "com.application.extra.PARAM»;
public final static String PARAM_PINTENT = «PendingIntent»;
public final static String PARAM_RESULT = «Result»;
final int REQUEST_CODE = 0;
public final static int RESULT_CODE = 1;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
PendingIntent pi;
pi = createPendingResult (REQUEST_CODE, new Intent (), 0);
Intent intent = new Intent(MainActivity.this, WebIntentService.class);
intent. putExtra (EXTRA_PARAM, "http://api.openweathermap.org/data/2.5/weather?q=London,uk");
intent. putExtra (PARAM_PINTENT, pi);
startService (intent);
```

```
}

@Override
protected void onActivityResult (int requestCode, int resultCode, Intent data) {
// Check which request we're responding to
if (requestCode == REQUEST_CODE) {
// Make sure the request was successful
if (resultCode == RESULT_CODE) {
String responseString=data.getStringExtra(MainActivity.PARAM_RESULT);
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (responseString);
}
}
}

@Override
public boolean onCreateOptionsMenu (Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest. xml.
int id = item.getItemId ();
//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
return true;
}
return super. onOptionsItemSelected (item);
}
}
```

Здесь в методе onCreate создается объект PendingIntent, который передается в объект Intent, с помощью которого запускается сервис.

В методе onActivityResult из возвращаемого объекта PendingIntent извлекаются данные, отображаемые в TextView.

AndroidManifest. xml будет иметь код:

```
<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package=«com. application»>

<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />
<application
android: allowBackup=«true»
android: icon="@mipmap/ic_launcher»
android: label="@string/app_name»
android: theme="@style/AppTheme»>
<activity
android:name=".MainActivity»
android: label="@string/app_name»>
<intent-filter>
```

<action android:name="android.intent.action.MAIN» />
<category android:name="android.intent.category.LAUNCHER» />
</intent-filter>
</activity>

<service
android: name=». WebIntentService»
android: exported=«false»>
</service>
</application>

</manifest>
Файл компоновки:
<RelativeLayout xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: tools="http://schemas.android.com/tools" android: layout_width=«match_parent»
android: layout_height=«match_parent» android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: paddingBottom="@dimen/activity_vertical_margin» tools:context=".MainActivity»>

<TextView
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: id="@+id/textView»/>

</RelativeLayout>

Activity + IntentService + ResultReceiver

Использование класса ResultReceiver еще один способ получения данных из сервиса для их обработки в методе onReceiveResult класса ResultReceiver.

Использование класса ResultReceiver аналогично использованию объекта PendingIntent.

В методе onCreate активности создается объект класса ResultReceiver, который передается в объект Intent, запускающий сервис.

В сервисе объект ResultReceiver извлекается, в него записываются данные, и он отсылается обратно в активность методом send. После этого системой вызывается метод onReceiveResult объекта ResultReceiver, в котором записанные в сервисе данные извлекаются.

Код активности будет иметь следующий вид:

```
package com. application;

import android. os. Handler;
import android.content.Intent;
import android. os. Bundle;
import android.support. v7.app. AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android. widget. TextView;

public class MainActivity extends AppCompatActivity {

public static final String EXTRA_PARAM = "com.application.extra.PARAM»;
public final static String PARAM_RESULT = «Result»;
public final static int RESULT_CODE = 1;
```

```
public WebReceiver webReceiver;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
webReceiver = new WebReceiver (new Handler ());

webReceiver.setReceiver (new WebReceiver.Receiver () {
@Override
public void onReceiveResult (int resultCode, Bundle resultData) {
if (resultCode == RESULT_CODE) {
String responseString=resultData.getString(MainActivity.PARAM_RESULT);
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (responseString);
}
}
});
Intent intent = new Intent(MainActivity.this, WebIntentService.class);
intent. putExtra (EXTRA_PARAM,
"http://api.openweathermap.org/data/2.5/weather?q=London,uk");
intent. putExtra («receiver», webReceiver);
startService (intent);

}

public static class WebReceiver extends android.os.ResultReceiver {

public interface Receiver {
public void onReceiveResult (int resultCode, Bundle resultData);
}
private Receiver receiver;

public WebReceiver (android. os. Handler handler) {
super (handler);
}

public void setReceiver (Receiver receiver) {
this.receiver = receiver;
}

@Override
protected void onReceiveResult (int resultCode, Bundle resultData) {
if (receiver!= null) {
receiver. onReceiveResult (resultCode, resultData);
}
}
}

@Override
public boolean onCreateOptionsMenu (Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}
```

```
@Override
public boolean onOptionsItemSelected (MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest. xml.
int id = item.getItemId ();

//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
return true;
}
return super. onOptionsItemSelected (item);
}
}
```

Здесь ResultReceiver представлен внутренним классом WebReceiver активности, экземпляр которого создается в методе onCreate. Затем экземпляр ResultReceiver передается в объект Intent, запускающий сервис.

Код IntentService при этом будет иметь следующий вид:

```
package com. application;

import android.app.IntentService;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android. os. Bundle;
import android.os.ResultReceiver;
import org. json. JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net. URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class WebIntentService extends IntentService {

public WebIntentService () {
super («WebIntentService»);
}

@Override
protected void onHandleIntent (Intent intent) {
if (intent!= null) {
final String param = intent.getStringExtra (MainActivity. EXTRA_PARAM);
ResultReceiver receiver = intent.getParcelableExtra («receiver»);
handleAction (param, receiver);
}
}

/**
* Handle action in the provided background thread with the provided
```

```
    * parameters.
    */
    private void handleAction (String param, ResultReceiver receiver) {
    ConnectivityManager connMgr = (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo ();
    if (networkInfo!= null && networkInfo.isConnected () && networkInfo.isAvailable ()) {
    InputStream is = null;
    URL url = null;
    try {
    url = new URL (param);
    HttpURLConnection conn = (HttpURLConnection) url. openConnection ();
    conn.setReadTimeout (10000 /* milliseconds */);
    conn.setConnectTimeout (15000 /* milliseconds */);
    conn.setRequestMethod («GET»);
    conn.setDoInput (true);
    conn.connect ();
    int response = conn.getResponseCode ();
    is = conn.getInputStream ();
    // Convert the InputStream into a string
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder ();
    String line;
    br = new BufferedReader (new InputStreamReader (is));
    while ((line = br.readLine ())!= null) {
    sb. append (line);
    }
    String contentAsString=sb.toString ();
    is.close ();
    conn. disconnect ();

    JSONObject reader = new JSONObject (contentAsString);
    JSONObject main = reader.getJSONObject («main»);
    String temp = main.getString («temp»);
    Calendar c = Calendar.getInstance ();
    SimpleDateFormat df = new SimpleDateFormat («yyyy-MM-dd HH: mm: ss»);
    String formattedDate = df.format(c.getTime ());
    String result=formattedDate + " temp: " +temp;

    Bundle bundle = new Bundle ();
    bundle.putString(MainActivity.PARAM_RESULT, result);
    receiver.send(MainActivity.RESULT_CODE, bundle);

    Context context = this;
    SharedPreferences sharedPref = context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY",Context.MODE_PRIVATE);
    SharedPreferences. Editor editor = sharedPref. edit ();
    editor. putString («temp», temp);
    editor. putString («date», formattedDate);
    editor.commit ();
    } catch (Exception e) {
    e.printStackTrace ();
    }
    } else {
```

```
Context context = this;
SharedPreferences sharedPref = context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY», Context.MODE_PRIVATE);
String date=sharedPref.getString («date», «unknown date»);
String temp=sharedPref.getString («temp», «unknown temp»);
String result=date + " temp: " +temp;
Bundle bundle = new Bundle ();
bundle.putString(MainActivity.PARAM_RESULT, result);
receiver.send(MainActivity.RESULT_CODE, bundle);
}
}
}
```

Здесь, при запуске сервиса, проверяется наличие соединения с Интернетом, затем получаются данные в JSON-формате по указанному URL адресу.

Далее необходимые данные извлекаются и передаются в объект Bundle в виде строки.

Объект Bundle передается в объект ResultReceiver, который отправляется обратно в создавшую его активность методом send.

Извлеченные данные записываются в настройки SharedPreferences.

Если же соединение с Интернетом отсутствует, тогда данные извлекаются из настроек SharedPreferences и передаются в объект Bundle в виде строки.

Для отключения соединения с Интернетом в эмуляторе можно нажать F8.

Activity + Service + Binder

При создании Bound-сервиса, сервис запускается из активности методом bindService. При этом в сервисе вызывается метод onBind, возвращающий объект IBinder, который обеспечивает доступ к экземпляру сервиса и соответственно методам сервиса.

Методы сервиса при том выполняются в основном потоке.

Если же требуется выполнение задачи в фоновом потоке и возврат результата из него обратно в активность, можно передать сервису объект Handler для отправки и получения данных из фонового потока.

Код Bound-сервиса будет следующим:

```
package com. application;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Binder;
import android. os. Bundle;
import android. os. IBinder;
import android. os. Handler;
import android.os.Message;
import org. json. JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net. URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;
```

```
public class WebBoundService extends Service {

private final IBinder binder = new LocalBinder ();

public WebBoundService () {
}

@Override
public IBinder onBind (Intent intent) {
return binder;
}

public class LocalBinder extends Binder {
WebBoundService getService () {
return WebBoundService.this;
}
}

public void getResult (String str, Handler hlr) {
final Handler handler=hlr;
final String urlString=str;
final Context context = this;

new Thread (new Runnable () {
public void run () {
try {
ConnectivityManager connMgr = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo ();
if (networkInfo!= null && networkInfo.isConnected () && networkInfo.isAvailable ()) {
InputStream is = null;
URL url = null;
url = new URL (urlString);
HttpURLConnection conn = (HttpURLConnection) url. openConnection ();
conn.setReadTimeout (10000 /* milliseconds */);
conn.setConnectTimeout (15000 /* milliseconds */);
conn.setRequestMethod («GET»);
conn.setDoInput (true);
conn.connect ();
is = conn.getInputStream ();
// Convert the InputStream into a string
BufferedReader br = null;
StringBuilder sb = new StringBuilder ();
String line;
br = new BufferedReader (new InputStreamReader (is));
while ((line = br.readLine ())!= null) {
sb. append (line);
}
String contentAsString = sb.toString ();
is.close ();
conn. disconnect ();

JSONObject reader = new JSONObject (contentAsString);
JSONObject main = reader.getJSONObject («main»);
String temp = main.getString («temp»);
```

```
Calendar c = Calendar.getInstance ();
SimpleDateFormat df = new SimpleDateFormat («yyyy-MM-dd HH: mm: ss»);
String formattedDate = df.format(c.getTime ());
String msg = formattedDate + " temp: " + temp;

Message msgObj = handler. obtainMessage ();
Bundle b = new Bundle ();
b. putString («message», msg);
msgObj.setData (b);
handler.sendMessage (msgObj);

SharedPreferences sharedPref = context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY», Context.MODE_PRIVATE);
SharedPreferences. Editor editor = sharedPref. edit ();
editor. putString («temp», temp);
editor. putString («date», formattedDate);
editor.commit ();

} else {
SharedPreferences sharedPref = context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY», Context.MODE_PRIVATE);
String date = sharedPref.getString («date», «unknown date»);
String temp = sharedPref.getString («temp», «unknown temp»);
String msg=date + " temp: " +temp;

Message msgObj = handler. obtainMessage ();
Bundle b = new Bundle ();
b. putString («message», msg);
msgObj.setData (b);
handler.sendMessage (msgObj);
}
} catch (Exception e) {
e.printStackTrace ();
}
}
}).start ();
}
}
```

Здесь, в метод сервиса передается URL-адрес для получения данных и объект Handler для отправки полученных данных.

В фоновом потоке метода проверяется наличие соединения с Интернетом, затем получаются данные в JSON-формате по указанному URL адресу.

Далее необходимые данные извлекаются и передаются в объект Handler в виде строки.

Извлеченные данные записываются в настройки SharedPreferences.

Если же соединение с Интернетом отсутствует, тогда данные извлекаются из настроек SharedPreferences и передаются в объект Handler.

Код активности, запускающей сервис, будет следующим:

```
package com. application;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android. os. Bundle;
```

```
import android. os. Handler;
import android. os. IBinder;
import android.os.Message;
import android.support. v7.app. AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android. widget. TextView;

public class MainActivity extends AppCompatActivity {

private ServiceConnection sConn;
private boolean bound = false;
private WebBoundService service;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
sConn = new ServiceConnection () {
public void onServiceConnected (ComponentName name, IBinder binder) {
WebBoundService. LocalBinder localBinder = (WebBoundService. LocalBinder) binder;
service = localBinder.getService ();
Handler handler = new Handler () {
@Override
public void handleMessage (Message msg) {
super. handleMessage (msg);
Bundle b = msg.getData ();
String responseString=b.getString («message»);
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (responseString);
}
};

http://api.openweathermap.org/data/2.5/weather?q=London,uk",tResult (»service.ge handler);
bound = true;
}
public void onServiceDisconnected (ComponentName name) {
bound = false;
}
};
Intent intent = new Intent(MainActivity.this, WebBoundService.class);
bindService (intent, sConn, this.BIND_AUTO_CREATE);

}

@Override
protected void onDestroy () {
super. onDestroy ();
if (bound) {
unbindService (sConn);
bound = false;
}
}

@Override
```

```
public boolean onCreateOptionsMenu (Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest. xml.
int id = item.getItemId ();

//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
return true;
}

return super. onOptionsItemSelected (item);
}
}
```

Здесь в методе handleMessage объекта Handler, отправленные сервисом данные извлекаются и отображаются.

Код манифеста AndroidManifest. xml:

```
<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package="com.tmsoftstudio. application»>

<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />

<application
android: allowBackup=«true»
android: icon="@mipmap/ic_launcher»
android: label="@string/app_name»
android: theme="@style/AppTheme»>
<activity
android:name=".MainActivity»
android: label="@string/app_name»>
<intent-filter>
<action android:name="android.intent.action.MAIN» />
<category android:name="android.intent.category.LAUNCHER» />
</intent-filter>
</activity>

<service
android: name=». WebBoundService»
android: enabled=«true»
android: exported=«true»>
</service>
</application>

</manifest>
```

Activity + Service + Messenger

Класс Messenger обеспечивает еще один способ обмена данными между активностью и Bound-сервисом, только при этом находящимися в разных процессах.

К объекту Messenger прикрепляется обработчик Handler, отвечающий за обработку поступающих сообщений.

В классе активности для отправки сообщения сервису, в методе обратного вызова onServiceConnected создается объект Messenger на основе объекта IBinder, передаваемого из сервиса.

После того создается объект Message, в который записываются данные.

В объект Message записывается объект Messenger, к которому прикреплен обработчик Handler, отвечающий за обработку сообщений от сервиса.

Объект Message отправляется сервису с помощью объекта Messenger, созданного на основе объекта IBinder.

В сервисе вызывается метод handleMessage обработчика Handler, связанного с объектом Messenger, из которого получается объект IBinder, отправляемый активности.

В методе handleMessage из сообщения извлекается объект Messenger, с помощью которого данные отправляются обратно активности.

Код активности будет следующим:

```
package com. application;

import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android. os. Bundle;
import android. os. Handler;
import android. os. IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.support. v7.app. AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android. widget. TextView;

public class MainActivity extends AppCompatActivity {

private ServiceConnection sConn;
private boolean bound = false;
private Messenger mMessenger;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
sConn = new ServiceConnection () {
public void onServiceConnected (ComponentName name, IBinder binder) {
mMessenger = new Messenger (binder);
Message msg = Message. obtain (null, WebBoundService.MSG_TO_SERVICE);
msg.replyTo = new Messenger (new ActivityHandler ());
Bundle bundle = new Bundle ();
bundle. putString («url», "http://api.openweathermap.org/data/2.5/weather?q=London,uk");
msg.setData (bundle);
try {
mMessenger.send (msg);
```

```
} catch (RemoteException e) {
e.printStackTrace ();
}

bound = true;
}

public void onServiceDisconnected (ComponentName name) {
mMessenger = null;
bound = false;
}
};
Intent intent = new Intent(MainActivity.this, WebBoundService.class);
bindService (intent, sConn, this.BIND_AUTO_CREATE);
}

class ActivityHandler extends Handler {
@Override
public void handleMessage (Message msg) {
switch (msg. what) {
case WebBoundService.MSG_FROM_SERVICE: {
Bundle b = msg.getData ();
String responseString=b.getString («respData»);
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (responseString);
}
break;
default:
super. handleMessage (msg);
}
}
}

@Override
protected void onDestroy () {
super. onDestroy ();
if (bound) {
unbindService (sConn);
bound = false;
}
}

@Override
public boolean onCreateOptionsMenu (Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {
// Handle action bar item clicks here. The action bar will
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest. xml.
int id = item.getItemId ();
```

```
//noinspection SimplifiableIfStatement
if (id == R.id.action_settings) {
return true;
}
return super. onOptionsItemSelected (item);
}
}
```

Код Bound-сервиса:

```
package com. application;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android. os. Bundle;
import android. os. IBinder;
import android. os. Handler;
import android.os.Message;
import android.os.Messenger;
import org. json. JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net. URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;

public class WebBoundService extends Service {

final Context context = this;
static final int MSG_TO_SERVICE = 1;
static final int MSG_FROM_SERVICE = 2;

public WebBoundService () {
}

Messenger mMessenger = new Messenger (new ServiceHandler ());

// Incoming messages Handler
class ServiceHandler extends Handler {
@Override
public void handleMessage (Message msg) {
switch (msg. what) {
case MSG_TO_SERVICE: {
final Messenger messenger=msg.replyTo;
Bundle bundle = msg.getData ();
final String urlString = (String) bundle.get («url»);
new Thread (new Runnable () {
public void run () {
try {
ConnectivityManager connMgr = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
```

```
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo ();
if (networkInfo!= null && networkInfo.isConnected () && networkInfo.isAvailable ()) {
InputStream is = null;
URL url = null;
url = new URL (urlString);
HttpURLConnection conn = (HttpURLConnection) url. openConnection ();
conn.setReadTimeout (10000 /* milliseconds */);
conn.setConnectTimeout (15000 /* milliseconds */);
conn.setRequestMethod («GET»);
conn.setDoInput (true);
conn.connect ();
is = conn.getInputStream ();
// Convert the InputStream into a string
BufferedReader br = null;
StringBuilder sb = new StringBuilder ();
String line;
br = new BufferedReader (new InputStreamReader (is));
while ((line = br.readLine ())!= null) {
sb. append (line);
}
String contentAsString = sb.toString ();
is.close ();
conn. disconnect ();
JSONObject reader = new JSONObject (contentAsString);
JSONObject main = reader.getJSONObject («main»);
String temp = main.getString («temp»);
Calendar c = Calendar.getInstance ();
SimpleDateFormat df = new SimpleDateFormat («yyyy-MM-dd HH: mm: ss»);
String formattedDate = df.format(c.getTime ());
String data = formattedDate + " temp: " + temp;
Message resp = Message. obtain (null, MSG_FROM_SERVICE);
Bundle bResp = new Bundle ();
bResp. putString («respData», data);
resp.setData (bResp);
messenger.send (resp);
SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY»,
Context.MODE_PRIVATE);
SharedPreferences. Editor editor = sharedPref. edit ();
editor. putString («temp», temp);
editor. putString («date», formattedDate);
editor.commit ();
} else {
SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY»,
Context.MODE_PRIVATE);
String date = sharedPref.getString («date», «unknown date»);
String temp = sharedPref.getString («temp», «unknown temp»);
String data = date + " temp: " + temp;
Message resp = Message. obtain (null, MSG_FROM_SERVICE);
Bundle bResp = new Bundle ();
bResp. putString («respData», data);
resp.setData (bResp);
messenger.send (resp);
}
```

```
} catch (Exception e) {
e.printStackTrace ();
}
}
}).start ();
}
break;
default:
super. handleMessage (msg);
}
}
}

@Override
public IBinder onBind (Intent intent) {
return mMessenger.getBinder ();
}
}
```

Код манифеста:

```
<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package="com.tmsoftstudio. application»>

<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />

<application
android: allowBackup=«true»
android: icon="@mipmap/ic_launcher»
android: label="@string/app_name»
android: theme="@style/AppTheme»>
<activity
android:name=".MainActivity»
android: label="@string/app_name»>
<intent-filter>
<action android:name="android.intent.action.MAIN» />
<category android:name="android.intent.category.LAUNCHER» />
</intent-filter>
</activity>

<service
android: name=». WebBoundService»
android: enabled=«true»
android: exported=«true»
android: process=":webservice»
>
</service>
</application>

</manifest>
```

Здесь атрибут android: process=":webservice» указывает на запуск сервиса в отдельном процессе.

Activity + Service + AIDL

AIDL интерфейс используется, если взаимодействие с сервисом производится из разных приложений и ваш сервис должен обеспечивать многопоточность.

Для создания AIDL интерфейса в Android Studio нажмем правой кнопкой мышеи на модуле и выберем **New | AIDL | AIDL File**.

В результате, в каталоге src/main модуля будет создана папка aidl с файлом интерфейса, на основе которого при сборке модуля в каталоге build/generated будет сгенерирована Java-реализация AIDL интерфейса.

Сгенерированная Java-реализация AIDL интерфейса представляет собой класс-заглушку, который расширяет класс Binder и поэтому может быть возвращен методом onBind Bound-сервиса.

Возвращенный в активность экземпляр AIDL-класса может содержать методы, вызываемые активностью.

Код AIDL-интерфейса:

```
package com. application;
interface IAidlInterface {
String getResult (String url);
}
```

Код Bound-сервиса:

```
package com. application;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android. os. IBinder;;
import android.os.RemoteException;
import org. json. JSONObject;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net. URL;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.concurrent.Callable;
import java.util.concurrent. FutureTask;
import java.util.concurrent.RunnableFuture;

public class WebBoundService extends Service {

public WebBoundService () {
}

@Override
public IBinder onBind (Intent intent) {
return new IAidlInterface.Stub () {
public String getResult (String url) throws RemoteException {

final String urlString=url;
String resp=«»;

RunnableFuture f = new FutureTask (new Callable <String> () {
@Override
public String call () throws Exception {
```

```
String data=«»;
ConnectivityManager connMgr = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo ();
if (networkInfo!= null && networkInfo.isConnected () && networkInfo.isAvailable ()) {
InputStream is = null;
URL url = null;
url = new URL (urlString);
HttpURLConnection conn = (HttpURLConnection) url. openConnection ();
conn.setReadTimeout (10000 /* milliseconds */);
conn.setConnectTimeout (15000 /* milliseconds */);
conn.setRequestMethod («GET»);
conn.setDoInput (true);
conn.connect ();
is = conn.getInputStream ();
// Convert the InputStream into a string
BufferedReader br = null;
StringBuilder sb = new StringBuilder ();
String line;
br = new BufferedReader (new InputStreamReader (is));
while ((line = br.readLine ())!= null) {
sb. append (line);
}
String contentAsString = sb.toString ();
is.close ();
conn. disconnect ();
JSONObject reader = new JSONObject (contentAsString);
JSONObject main = reader.getJSONObject («main»);
String temp = main.getString («temp»);
Calendar c = Calendar.getInstance ();
SimpleDateFormat df = new SimpleDateFormat («yyyy-MM-dd HH: mm: ss»);
String formattedDate = df.format(c.getTime ());
data = formattedDate + " temp: " + temp;
return data;
} else {
return data;
}
}
});
new Thread(f).start ();
try {
resp= (String) f.get ();
} catch (Exception e) {
e.printStackTrace ();
}
return resp;
}
};
}
}
```

Здесь в реализации метода AIDL-интерфейса данные получаются из Интернета в отдельном потоке.

Для возврата данных из отдельного потока используется класс FutureTask.

Код активности:

```
package com. application;
```

```
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android. os. Bundle;
import android. os. IBinder;
import android.os.RemoteException;
import android.support. v7.app. AppCompatActivity;
import android. util. Log;
import android.view.Menu;
import android.view.MenuItem;
import android. widget. TextView;

public class MainActivity extends AppCompatActivity {

private final Context context=this;
private ServiceConnection sConn;
private boolean bound = false;
private IAidlInterface service;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
sConn = new ServiceConnection () {
public void onServiceConnected (ComponentName name, IBinder binder) {
service = IAidlInterface.Stub.asInterface ((IBinder) binder);
String data=«»;

try {
data = http://api.openweathermap.org/data/2.5/weather?q=London,uk");tResult (»service.ge
} catch (RemoteException e) {
e.printStackTrace ();
}

if (!data. equals (»»)) {
SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY»,
Context.MODE_PRIVATE);
SharedPreferences. Editor editor = sharedPref. edit ();
editor. putString («data», data);
editor.commit ();
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (data);
} else {
SharedPreferences sharedPref =
context.getSharedPreferences("com.application.PREFERENCE_FILE_KEY»,
Context.MODE_PRIVATE);
String savedData = sharedPref.getString («data», «unknown data»);
TextView textView = (TextView) findViewById(R.id.textView);
textView.setText (savedData);
}
```

```
bound = true;
}

public void onServiceDisconnected (ComponentName name) {
service = null;
bound = false;
}
};
Intent intent = new Intent(MainActivity.this, WebBoundService.class);
bindService(intent,sConn,this.BIND_AUTO_CREATE);
}

@Override
protected void onDestroy () {
super. onDestroy ();
if (bound) {
unbindService (sConn);
bound = false;
}
}

@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {

int id = item.getItemId ();
if (id == R.id.action_settings) {
return true;
}
return super. onOptionsItemSelected (item);
}
}
```

Здесь в методе onServiceConnected при соединении с Bound-сервисом получается экземпляр AIDL-класса, после этого вызывается его метод.

Отображение Toast сообщения из сервиса или потока

```
Handler handler = new Handler(Looper.getMainLooper ());
handler.post (new Runnable () {

@Override
public void run () {
Toast.makeText(context.getApplicationContext (),«No internet connection",Toast.LENGTH_SHORT).show ();
}
});
```

Изменение цвета шрифта для всей активности

В файл styles. xml в тэг <style> включим элемент:
<item name=«android: textColor»> @color/colorPrimary </item>

Получить выбранный RadioButton из RadioGroup

```
RadioGroup radioButtonGroup = (RadioGroup) view.findViewById(R.id.radiogroup);
int radioButtonID = radioButtonGroup.getCheckedRadioButtonId ();
RadioButton radioButton = (RadioButton) radioButtonGroup.findViewById (radioButtonID);
String str = radioButton.getText().toString ();
```

Сборка APK файла с большим количеством методов в коде

В Gradle файл добавить:

```
defaultConfig {

multiDexEnabled true
}
dependencies {

compile 'com.android.support: multidex:1.0.0»
}
android {

dexOptions {
incremental true
javaMaxHeapSize «4g»
}
}
```

В файл манифеста:

```
<application

android:name="android.support.multidex.MultiDexApplication»>
```

Получение Google аккаунта пользователя

```
Intent intent = AccountPicker.newChooseAccountIntent (null, null, new String [] {«com. google»},
true, null, null, null, null);
startActivityForResult (intent, REQUEST_CODE_ACCOUNT);

protected void onActivityResult (final int requestCode, final int resultCode, final Intent data) {
if (requestCode == REQUEST_CODE_ACCOUNT && resultCode == RESULT_OK)
{
String accountName = data.getStringExtra(AccountManager.KEY_ACCOUNT_NAME);
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («APP_PREFERENCES_ACCOUNT», accountName);
editor.commit ();
}
}
```

Присоединение слушателя к Button

```
Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener (new View. OnClickListener () {
```

```
public void onClick (View v) {
// Perform action on click
}
});
```

Передача изображения в Google App Engine Java Blobstore используя Google Cloud Endpoints

Выбор изображения в галереи:

```
Button imageButton=(Button)view.findViewById(R.id.change_profile_photo);
imageButton.setOnClickListener (new View. OnClickListener () {
public void onClick (View v) {
Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
photoPickerIntent.setType («image/*»);
getActivity().startActivityForResult (photoPickerIntent, REQUEST_CODE_PHOTO);
}
});
protected void onActivityResult (final int requestCode, final int resultCode, final Intent data) {
if (requestCode == REQUEST_CODE_PHOTO && resultCode == RESULT_OK) {
Uri selectedImage = data.getData ();
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («APP_PREFERENCES_PHOTO», selectedImage.toString ());
editor.commit ();
}
}
```

Кодирование изображения в Base64-строку:

```
Uri imageUri = Uri.parse(mSettings.getString («APP_PREFERENCES_PHOTO»,»»));
InputStream imageStream = null;
try {
imageStream = getContext().getContentResolver ().openInputStream (imageUri);
} catch (FileNotFoundException e) {
e.printStackTrace ();
}
Bitmap bitmap = BitmapFactory.decodeStream (imageStream);
ByteArrayOutputStream outputStream = new ByteArrayOutputStream ();
bitmap.compress(Bitmap.CompressFormat.JPEG, 10, outputStream);
bitmap.recycle ();
bitmap = null;
byte [] bitmapByte = outputStream.toByteArray ();
outputStream=null;
String stringEncodedImage = Base64.encodeToString (bitmapByte, Base64.DEFAULT);
bitmapByte=null;
```

Отправка изображения:

```
private static MyApi myApiService = null;
private MyBean profile=new MyBean ();

profile.setMImage (stringEncodedImage);
ProfileDialogAsyncTask task=new ProfileDialogAsyncTask ();
task. execute (profile);
private static class ProfileDialogAsyncTask extends AsyncTask <MyBean, Integer, MyBean> {
private MyApi myApiService = null;
private ProgressBar pb;

@Override
protected void onPreExecute () {
super. onPreExecute ();
pb = (ProgressBar)context.findViewById(R.id.progress);
```

```
pb.setVisibility(View.VISIBLE);
}

@Override
protected void onPostExecute (MyBean myBean) {
super. onPostExecute (myBean);
Toast.makeText(context.getApplicationContext (),«The profile has been saved",Toast.LENGTH_SHORT).show ();
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («APP_PREFERENCES_PHOTO_URL», myBean.getMImage ());
editor.commit ();
pb.setVisibility(View.INVISIBLE);
}

@Override
protected void onProgressUpdate (Integer… values) {
super. onProgressUpdate (values);
}

@Override
protected MyBean doInBackground (MyBean… params) {
MyBean profile=params [0];
MyBean response=null;
if (myApiService == null) {
MyApi. Builder builder = new MyApi.Builder(AndroidHttp.newCompatibleTransport (),
new AndroidJsonFactory (), null)
.setRootUrl (»https://meet-love-android.appspot.com/_ah/api/")
.setGoogleClientRequestInitializer (new GoogleClientRequestInitializer () {
@Override
public void initialize (AbstractGoogleClientRequest <?> abstractGoogleClientRequest) throws IOException {
abstractGoogleClientRequest.setDisableGZipContent (true);
}
});
myApiService = builder. build ();
}

try {
response=myApiService.setProfile (profile).execute ();
} catch (IOException e) {
}
return response;
}
}
```

Получение изображения:

```
import com.google.api.server.spi.config. Api;
import com.google.api.server.spi.config. ApiMethod;
import com.google.api.server.spi.config. ApiNamespace;
import com. google. appengine. api. blobstore. BlobInfo;
import com. google. appengine. api. blobstore. BlobInfoFactory;
import com. google. appengine. api. blobstore. BlobKey;
import com. google. appengine. api. blobstore. BlobstoreService;
import com. google. appengine. api. blobstore. BlobstoreServiceFactory;
import com. google. appengine. api. datastore. DatastoreService;
import com. google. appengine. api. datastore. DatastoreServiceFactory;
```

```
import com. google. appengine. api. datastore. Entity;
import com. google. appengine. api. datastore. EntityNotFoundException;
import com.google.appengine.api.datastore.Key;
import com.google.appengine.api.datastore.KeyFactory;
import com.google.appengine.api.images.ImagesService;
import com.google.appengine.api.images.ImagesServiceFactory;
import com.google.appengine.api.images.ServingUrlOptions;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.IOUtils;
import org.apache.http.HttpEntity;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.mime.MultipartEntityBuilder;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net. URL;
import java.util.Iterator;

@Api (
name = «myApi»,
version = «v1»,
namespace = @ApiNamespace (
ownerDomain = "backend.com»,
ownerName = "backend.com»,
packagePath=»»
)
)
public class MyEndpoint {
private DatastoreService datastore = DatastoreServiceFactory.getDatastoreService ();

@ApiMethod (
name = «setProfile»,
httpMethod = ApiMethod.HttpMethod.POST
)

public MyBean setProfile (MyBean profile) {
String accountName=profile.getmAccount ();
Key key = KeyFactory.createKey («User», accountName);
Entity user=null;
try {
user = datastore.get (key);
} catch (EntityNotFoundException e) {
e.printStackTrace ();
}
if (user==null) {
user = new Entity («User», accountName);
}
byte [] imageAsBytes = Base64.decodeBase64(profile.getmImage ());
BlobstoreService blobService = BlobstoreServiceFactory.getBlobstoreService ();

BlobInfoFactory bf = new BlobInfoFactory ();
Iterator itr = bf. queryBlobInfos ();
while (itr. hasNext ()) {
BlobInfo bi = (BlobInfo)itr.next ();
```

```
if (bi.getFilename().equals(accountName+".jpeg»)) {
blobService.delete(bi.getBlobKey ());
}
}

String uploadUrl = blobService.createUploadUrl (»/blob/upload»);
HttpEntity requestEntity = MultipartEntityBuilder.create ()
.addBinaryBody («file», imageAsBytes, ContentType.create («image/jpg»), accountName+".jpeg»)
.build ();

try {
URL url = new URL (uploadUrl);
HttpURLConnection connection = (HttpURLConnection) url. openConnection ();
connection.setDoOutput (true);
connection.setRequestMethod («POST»);
connection.addRequestProperty («Content-length», requestEntity.getContentLength () + «»);
connection.addRequestProperty(requestEntity.getContentType().getName (),
requestEntity.getContentType().getValue ());
requestEntity.writeTo(connection.getOutputStream ());
String responseBody = IOUtils.toString(connection.getInputStream ());

if(connection.getResponseCode () <200 || connection.getResponseCode ()> = 400) {
throw new IOException («HTTP Status " + connection.getResponseCode () + ": "
+ connection.getHeaderFields () + "\n» + responseBody);
}

BlobKey blobKey = new BlobKey (responseBody);
ImagesService imagesService = ImagesServiceFactory.getImagesService ();
ServingUrlOptions servingOptions = ServingUrlOptions. Builder. withBlobKey (blobKey);

String servingUrl = imagesService.getServingUrl (servingOptions);
user.setProperty («mBlobKey», responseBody);
user.setProperty («mServingUrl», servingUrl);
profile.setmImage (servingUrl);
} catch (Exception e) {
e.printStackTrace ();
}
datastore. put (user);
return profile;
}
}
import com. google. appengine. api. blobstore. BlobKey;
import com. google. appengine. api. blobstore. BlobstoreService;
import com. google. appengine. api. blobstore. BlobstoreServiceFactory;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class BlobUploadServlet extends HttpServlet {
```

```
@Override
public void doPost (HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
BlobstoreService blobService = BlobstoreServiceFactory.getBlobstoreService ();
List <BlobKey> blobs = blobService.getUploads(req).get («file»);
if (blobs == null || blobs.isEmpty ()) throw new IllegalArgumentException («No blobs given»);

BlobKey blobKey = blobs.get (0);
res.setStatus(HttpServletResponse.SC_OK);
res.setContentType («text/plain»);
PrintWriter out = res.getWriter ();
out.print(blobKey.getKeyString ());
out. flush ();
out.close ();
}
}
```

Fragment + GoogleMap

Файл app_bar_main. xml:

```
<?xml version=«1.0» encoding=«utf-8»? >
<android.support.design.widget.CoordinatorLayout
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/bar_main»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: fitsSystemWindows=«true»
tools:context="com.tmsoftstudio.meetlove.MainActivity»>

<android.support.design. widget. AppBarLayout
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: theme="@style/AppTheme. AppBarOverlay»>

<android.support.v7.widget.Toolbar
android: id="@+id/toolbar»
android: layout_width=«match_parent»
android: layout_height=»? attr/actionBarSize»
android: background=»? attr/colorPrimary»
app: popupTheme="@style/AppTheme. PopupOverlay»
/>

</android.support.design. widget. AppBarLayout>

<RelativeLayout
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: paddingBottom="@dimen/activity_vertical_margin»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
app: layout_behavior="@string/appbar_scrolling_view_behavior»
android: id="@+id/content»
```

```
>

</RelativeLayout>

<android.support.design. widget. FloatingActionButton
android: id="@+id/fab»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_gravity=«bottom|end»
android: layout_margin="@dimen/fab_margin»
android:          src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/@android:
drawable/ic_input_add»
app: layout_anchor="@id/content»
app: layout_anchorGravity=«top|start»
/>

</android.support.design.widget.CoordinatorLayout>
```
Файл activity_maps. xml:
```
RelativeLayout
xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: id="@+id/map_container»>

<fragment
xmlns: map="http://schemas.android.com/apk/res-auto"
android: id="@+id/map»
android:name="com.google.android.gms.maps.SupportMapFragment»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
/>
</RelativeLayout>
```
Код фрагмента:
```
import android.content.Context;
import android. os. Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps. GoogleMap;
import com.google.android.gms.maps. OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps. UiSettings;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class ViewMapFragment extends Fragment implements OnMapReadyCallback {

private static final String ARG_PARAM = «param»;

private String mParam;
private GoogleMap mMap;
```

```
private SupportMapFragment fragment;

private OnViewMapFragmentInteractionListener mListener;

public ViewMapFragment () {
// Required empty public constructor
}

public static ViewMapFragment newInstance (String param) {
ViewMapFragment fragment = new ViewMapFragment ();
Bundle args = new Bundle ();
args. putString (ARG_PARAM, param);
fragment.setArguments (args);
return fragment;
}

@Override
public void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
if (getArguments ()!= null) {
mParam = getArguments().getString (ARG_PARAM);
}
}

@Override
public View onCreateView (LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
// Inflate the layout for this fragment
View view= inflater.inflate(R.layout.activity_maps, container, false);
return view;
}

@Override
public void onActivityCreated (Bundle savedInstanceState) {
super. onActivityCreated (savedInstanceState);
FragmentManager fm = getChildFragmentManager ();
fragment = (SupportMapFragment) fm.findFragmentById(R.id.map);
if (fragment == null) {
fragment = SupportMapFragment.newInstance ();
fm.beginTransaction().replace(R.id.map_container, fragment).commit ();
}
fragment.getMapAsync (this);

}

@Override
public void onMapReady (GoogleMap googleMap) {
mMap = googleMap;
UiSettings settings = mMap.getUiSettings ();
settings.setZoomControlsEnabled (true);

LatLng sydney = new LatLng (-34, 151);
mMap.addMarker (new MarkerOptions ().position (sydney).title («Marker in Sydney»));
mMap.animateCamera(CameraUpdateFactory.newLatLngZoom (sydney, 14.0f));
}
```

```
public void onViewMapPressed () {
if (mListener!= null) {
mListener. onViewMapFragmentInteraction ();
}
}

@Override
public void onAttach (Context context) {
super. onAttach (context);
if (context instanceof OnViewMapFragmentInteractionListener) {
mListener = (OnViewMapFragmentInteractionListener) context;
} else {
throw new RuntimeException(context.toString ()
+ " must implement OnFragmentInteractionListener»);
}
}

@Override
public void onDetach () {
super. onDetach ();
mListener = null;
}

public interface OnViewMapFragmentInteractionListener {
void onViewMapFragmentInteraction ();
}
}
```

Вызов фрагмента:

```
FragmentManager fragmentManager = getSupportFragmentManager ();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction ();
ViewMapFragment fragmentMap = new ViewMapFragment ();
fragmentTransaction.replace(R.id.content, fragmentMap);
fragmentTransaction.addToBackStack (null);
fragmentTransaction.commit ();
fab. hide ();
```

Файл манифеста:

```
<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION» />

<meta-data
android:name="com.google.android.geo. API_KEY»
android: value="@string/google_maps_key» />
```

Файл google_maps_api. xml:

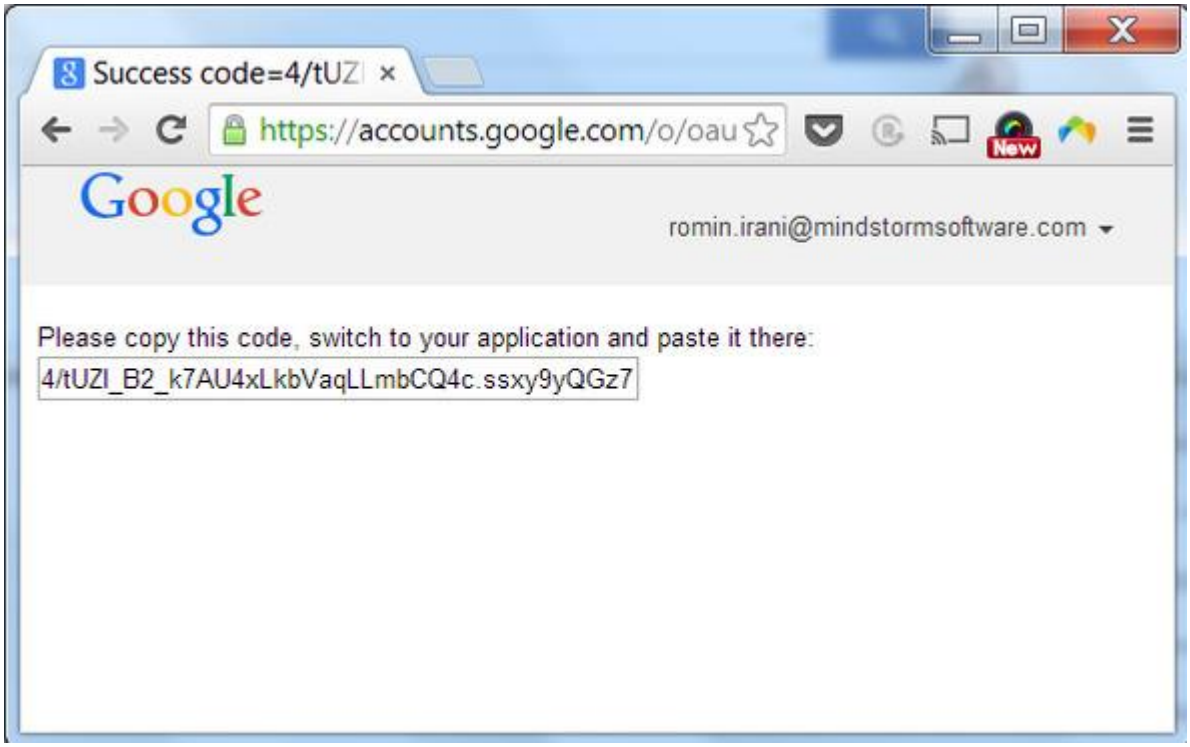
```
<resources>
<string name=«google_maps_key» templateMergeStrategy=«preserve» translatable=«false»>
Aiza…
</string>
</resources>
```

Использование нескольких Google аккаунтов для модулей App Engine

В файл buld.gradle backend модуля добавьте:

```
appengine {
```

downloadSdk = true
appcfg {
oauth2 = true
noCookies = true
email = «google email»
}
endpoints {
getClientLibsOnBuild = true
getDiscoveryDocsOnBuild = true
}
}
В окне Terminal Android Studio наберите:
gradlew task
Затем наберите:
gradlew appengineUpdate
После появления oauth2 запроса:

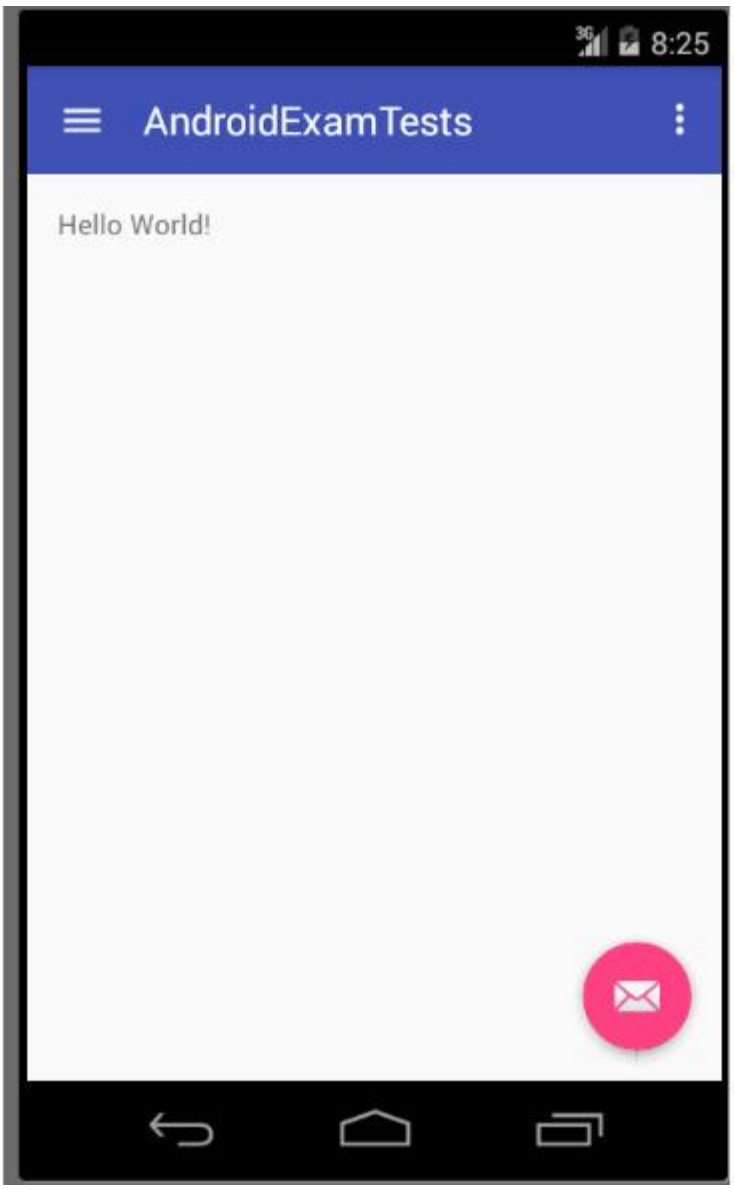


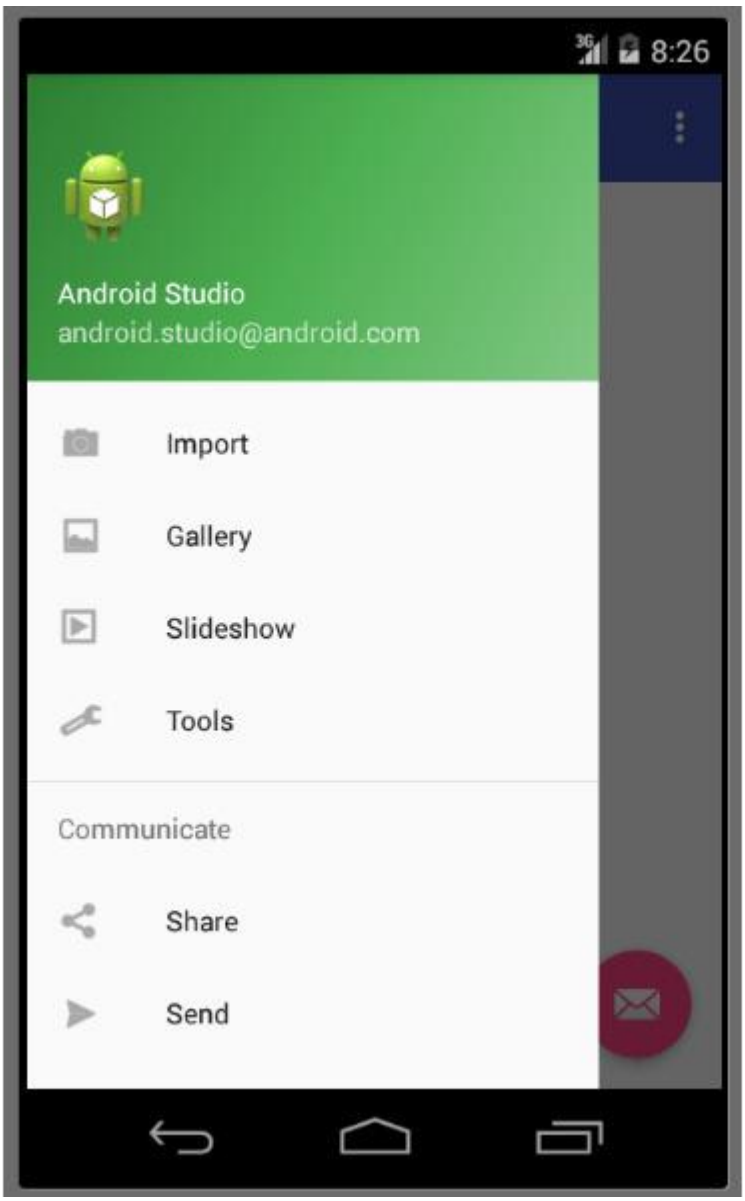
Скопируйте строку и вставьте ее в окне Terminal, нажмите Enter.
В результате backend модуль будет развернут по указанному аккаунту.

Android Design Support Library

Android Design Support Library обеспечивает фреймворк, позволяющий создавать приложения, включающие в себя навигационное представление navigation drawer view, плавающую метку для редактируемого текста floating labels for editing text, плавающую кнопку действия floating action button, snackbar, компоновки CoordinatorLayout и AppBarLayout, панель инструментов Toolbar вместо Action Bar, RecyclerView вместо ListView и др.

Наиболее полно отображает использование библиотеки Android Design Support Library шаблон проекта Navigation Drawer Activity среды разработки Android Studio.


8:25
AndroidExamTests
Hello World!



Класс главной активности проекта расширяет класс android.support. v7.app. AppCompatActivity и реализует интерфейс android.support.design.widget.NavigationView. OnNavigationItemSelectedListener.

Класс AppCompatActivity здесь дает возможность использования методов setSupportActionBar (toolbar) определяет Toolbar как ActionBar и onBackPressed () вызывается системой при нажатии кнопки назад.

Интерфейс NavigationView. OnNavigationItemSelectedListener дает метод onNavigationItemSelected (MenuItem item), вызываемый системой при выборе элемента navigation drawer view.

Компоновка главной активности начинается с корневого элемента android.support. v4.widget. DrawerLayout. Это является обязательным условием для того, чтобы добавить в приложение панель Navigation Drawer, отображающую основные элементы навигации приложения по левому краю экрана.

Атрибут android: fitsSystemWindows элемента DrawerLayout установлен в true, что дает достаточный отступ, чтобы не было наложения на строку состояния status bar.

Атрибут tools: openDrawer=«start» не влияет на работу приложения и обеспечивает отображение навигационного ящика при открытии компоновки во вкладке **Design** редактора Android Studio.

Первый дочерний элемент компоновки главной активности содержит компоновку основного содержимого активности, отображаемого, когда навигационный ящик скрыт. Атрибуты android: layout_width=«match_parent» и android: layout_height=«match_parent» обеспечивают заполнение всего окна основным контентом.

Второй элемент android.support.design.widget.NavigationView компоновки главной активности объявляет содержимое навигационного ящика. Атрибуты android: layout_width=«wrap_content» и android: layout_height=«match_parent» обеспечивают ширину равную содержимому и высоту равную высоте окна. Ширину навигационного ящика можно сделать фиксированной, например, 320dp. Атрибут android: layout_gravity=«start» обеспечивает отображение навигационного ящика на левой стороне окна. Атрибут app: headerLayout указывает компоновку заголовка навигационного ящика, атрибут app: menu указывает XML ресурс элементов навигационного ящика.

В компоновке заголовка навигационного ящика значение атрибута android: layout_height корневого элемента LinearLayout можем установить в android: layout_height=«wrap_content». Можно изменить форму и цвет фона заголовка в файле side_nav_bar. xml, например:

```
<shape xmlns: android="http://schemas.android.com/apk/res/android"
android: shape=«oval»>
<gradient
android: startColor=«#B87F7F»
android: centerColor=«#8C5A5A»
android: endColor=«#7E4949»
android: type=«linear»
android: angle=«135»/>
</shape>
```

В заголовке можно убрать значок и расположить текстовый элемент по центру, например:

```
<?xml version=«1.0» encoding=«utf-8»? >
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«match_parent» android: layout_height=«wrap_content»
android: background="@drawable/side_nav_bar»
android: paddingBottom=«5dp»
android: paddingTop=«5dp»
android: theme="@style/ThemeOverlay. AppCompat. Dark» android: orientation=«vertical»
android: gravity=«bottom»>



<TextView android: layout_width=«match_parent» android: layout_height=«wrap_content»
android: text=«Содержание»
android:textAppearance="@style/Base.TextAppearance.AppCompat.Medium»
android: textAlignment=«center» />



</LinearLayout>
```

С помощью атрибутов android: background="@color/nav_drawer_bg», app: itemTextColor="@color/item_drawer» и app: itemBackground="@drawable/navigation_drawer_background» элемента NavigationView можно установить фон навигационного ящика, цвет и фон элементов навигационного ящика. Фон элементов навигационного ящика здесь определяется файлом navigation_drawer_background. xml папки drawable:

```
<?xml version=«1.0» encoding=«utf-8»? >
<selector xmlns: android="http://schemas.android.com/apk/res/android">
<item android: state_checked=«true» android: drawable="@color/item_checked» />
</selector>
```

Цвета определяются файлом colors. xml папки values:

<?xml version=«1.0» encoding=«utf-8»? >
<resources>
<color name=«colorPrimary»> #8C5A5A </color>
<color name=«colorPrimaryDark»> #7E4949 </color>
<color name=«colorAccent»> #FF4081 </color>
<color name=«nav_drawer_bg»> #C6A9A9 </color>
<color name=«item_checked»> #FFFFFF </color>
<color name=«item_drawer»> #3B1B1B </color>
</resources>

Размер текста элементов навигационного ящика NavigationView можно установить с помощью атрибута android: theme="@style/NavigationViewStyle» и стиля:

<style name=«NavigationViewStyle»>
<item name=«android: textSize»> 22sp </item>
</style>

Для установки элементов навигационного ящика нужно изменить содержимое файла activity_main_drawer. xml папки menu, например:

<?xml version=«1.0» encoding=«utf-8»? >
<menu xmlns: android="http://schemas.android.com/apk/res/android">

<group android: checkableBehavior=«single»>
<item android: id="@+id/test_1»
android: title=«Тест 1» />
<item android: id="@+id/test_2»
android: title=«Тест 2» />
<item android: id="@+id/test_3»
android: title=«Тест 3» />
<item android: id="@+id/test_4»
android: title=«Тест 4» />
</group>

<item android: title=«Ответы тестов»>
<menu>
<group android: checkableBehavior=«single»>
<item
android: id="@+id/a_test_1»
android: title=«Тест 1» />
<item
android: id="@+id/a_test_2»
android: title=«Тест 2» />
<item
android: id="@+id/a_test_3»
android: title=«Тест 3» />
<item
android: id="@+id/a_test_4»
android: title=«Тест 4» />
</group>
</menu>
</item>
</menu>

В методе onCreate класса главной активности следующий код отвечает за появление в панели инструментов значка гамбургера, при нажатии на который выскакивает навигационный ящик:

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle (this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);

toggle.syncState ();

Вызов метода drawer.setDrawerListener (toggle) обеспечивает анимацию значка гамбургера при движении навигационного ящика.

Первый дочерний элемент компоновки главной активности указывает компоновку основного содержимого активности, отображаемого, когда навигационный ящик скрыт. Корневым элементом этой компоновки служит android.support.design.widget.CoordinatorLayout – ViewGroup, обеспечивающий такие эффекты, как перемещение кнопки действия Floating Action Button вверх и вниз, чтобы освободить место для Snackbar, отображение и скрытие панели инструментов Toolbar и кнопки действия FAB при прокрутке списка содержания вниз и вверх.

Первый дочерний элемент CoordinatorLayout это элемент android.support.design. widget. AppBarLayout, обеспечивающий, чтобы вложенный элемент android.support.v7.widget.Toolbar реагировал на прокрутку.

Здесь Toolbar выступает как ActionBar. Для этого в манифесте AndroidManifest. xml стиль активности объявлен как android:theme="@style/AppTheme.NoActionBar» и в методе onCreate активности вызван метод setSupportActionBar (toolbar).

Атрибуты android: layout_height=»? attr/actionBarSize"и app: popupTheme="@style/AppTheme. PopupOverlay» обеспечивают режим наложения при скрытии и показе панели инструметов. Для того чтобы Toolbar реагировал на прокрутку также нужно добавить атрибут app: layout_scrollFlags=«scroll|enterAlways».

Чтобы определить связь между AppBarLayout и видом, который будет прокручиваться, нужно добавить атрибут app: layout_behavior к любому виду, способному содержать прокрутку, такому как NestedScrollView. Поэтому в файле компоновки content_main. xml обернем элемент RelativeLayout в элемент NestedScrollView:

```
<?xml version=«1.0» encoding=«utf-8»? >
<android.support.v4.widget.NestedScrollView
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: paddingBottom="@dimen/activity_vertical_margin»
android: fillViewport=«true»
android: layout_gravity=«fill_vertical»
app: layout_behavior="@string/appbar_scrolling_view_behavior»
tools:context=".MainActivity»
tools: showIn="@layout/app_bar_main»
>



<RelativeLayout
android: layout_width=«match_parent»
android: layout_height=«match_parent»>



<TextView android: text=«Hello World!»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content» />
```

```
</RelativeLayout>


</android.support.v4.widget.NestedScrollView>
```

Теперь при прокрутке основного содержимого активности панель инструментов Toolbar будет отображаться и скрываться.

Для того чтобы кнопка действия FAB реагировала на прокрутку содержания вниз и вверх, добавим атрибут app:layout_behavior="com.example.ScrollFABBehavior» в элемент android.support.design. widget. FloatingActionButton и создадим класс ScrollFABBehavior:

```
import android.content.Context;
import android.support.design.widget.CoordinatorLayout;
import android.support.design. widget. FloatingActionButton;
import android.support.v4.view.ViewCompat;
import android.util.AttributeSet;
import android.view.View;



public class ScrollFABBehavior extends FloatingActionButton.Behavior {
public ScrollFABBehavior (Context context, AttributeSet attrs) {
super ();
}



@Override
public boolean onStartNestedScroll (final CoordinatorLayout coordinatorLayout, final FloatingActionButton child,
final View directTargetChild, final View target, final int nestedScrollAxes) {
return nestedScrollAxes == ViewCompat.SCROLL_AXIS_VERTICAL
|| super. onStartNestedScroll (coordinatorLayout, child, directTargetChild, target, nestedScrollAxes);
}



@Override
public void onNestedScroll (final CoordinatorLayout coordinatorLayout, final FloatingActionButton child,
final View target, final int dxConsumed, final int dyConsumed,
final int dxUnconsumed, final int dyUnconsumed) {
super. onNestedScroll (coordinatorLayout, child, target, dxConsumed, dyConsumed, dxUnconsumed, dyUnconsumed);
if (dyConsumed> 0 && child.getVisibility () == View.VISIBLE) {
child. hide ();
} else if (dyConsumed <0 && child.getVisibility ()!= View.VISIBLE) {
child.show ();
}
}
}
```

Изменить значок кнопки действия FAB можно с помощью атрибута android: src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/@android: drawable/ic_menu_send».

Кнопку действия FAB можно также расположить посередине экрана, например:

```
<?xml version=«1.0» encoding=«utf-8»? >
<android.support.design.widget.CoordinatorLayout
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«match_parent»
android: fitsSystemWindows=«true»
tools:context=".MainActivity»>



<android.support.design. widget. AppBarLayout
android: layout_height=«wrap_content»
android: layout_width=«match_parent»
android: theme="@style/AppTheme. AppBarOverlay»>



<android.support.v7.widget.Toolbar
android: id="@+id/toolbar»
android: layout_width=«match_parent»
android: layout_height=»? attr/actionBarSize»
android: background=»? attr/colorPrimary»
app: popupTheme="@style/AppTheme. PopupOverlay»
app: layout_scrollFlags=«scroll|enterAlways»
/>



</android.support.design. widget. AppBarLayout>



<android.support.v4.widget.NestedScrollView
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: paddingBottom="@dimen/activity_vertical_margin»
android: fillViewport=«true»
android: layout_gravity=«fill_vertical»
app: layout_behavior="@string/appbar_scrolling_view_behavior»
tools:context=".MainActivity»
tools: showIn="@layout/app_bar_main»
android: id="@+id/sc_view»
>
```

```
<RelativeLayout
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
>



<RelativeLayout
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: id="@+id/content»
>



<TextView android: text="@string/def_test_title»
android: id="@+id/num_test»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
style="@android:style/TextAppearance.Large»
android: textStyle=«bold»/>



</RelativeLayout>
<RelativeLayout
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: layout_below="@+id/content»
android: layout_alignParentLeft=«true»
android: layout_alignParentStart=«true»
android: paddingTop=«96dp»
>



<TableLayout
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_alignParentBottom=«true»>
<TableRow
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_centerVertical=«true»
android: layout_alignParentLeft=«true»
android: layout_alignParentStart=«true»>



</TableRow>
</TableLayout>
</RelativeLayout>
</RelativeLayout>
</android.support.v4.widget.NestedScrollView>
```

<android.support.design. widget. FloatingActionButton
android: id="@+id/fab»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_gravity=«bottom|end»
android: layout_margin="@dimen/fab_margin»
android: src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/@android: drawable/ic_input_add» app:layout_behavior="com.example.ScrollFABBehavior»
app: layout_anchor="@id/content»
app: layout_anchorGravity=«bottom|center»
/>

</android.support.design.widget.CoordinatorLayout>

Изменить фон сообщения Snackbar, отображаемого при нажатии кнопки FAB можно следующим образом:

Snackbar snack = Snackbar.make (view, result, Snackbar. LENGTH_LONG);
View snackbarView = snack.getView ();
snackbarView.setBackgroundColor(Color.parseColor («#7E4949»));
snack.setAction («Action», null).show ();

Пример приложения Baby Tracking

Данное приложение периодически собирает информацию о местоположении телефона и сохраняет ее. Временной интервал сбора информации выставляется в настройках приложения.

Собранная информация сохраняется и передается в облако. Временной период, за который информация сохраняется, также определяется в настройках приложения.

Интерфейс приложения позволяет запустить трекинг, отключить трекинг и очистить сохраненные данные. Доступ к интерфейсу приложения осуществляется по логину-паролю. Настойки приложения позволяют изменить логин-пароль.

Приложение позволяет отобразить сохраненный трекинг, загружая и отображая переданные в облако данные.

При запущенном трекинге перезагрузка телефона не останавливает трекинг.

Файл манифеста приложения:

<?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package="com.babytracking»>

<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED» />
<uses-permission android:name="android.permission. WAKE_LOCK» />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION» />

<application
android: allowBackup=«true»
android: icon="@mipmap/ic_launcher»
android: label="@string/app_name»
android: supportsRtl=«true»
android: theme="@style/AppTheme»>
<activity
android: name=». LoginActivity»
android: label="@string/title_activity_login»>
<intent-filter android: label="@string/app_name»>

```
<action android:name="android.intent.action.MAIN» />

<category android:name="android.intent.category.LAUNCHER» />
</intent-filter>
</activity>

<service
android: name=». LocationIntentService»
android: exported=«false»>
</service>

<receiver
android: name=». LocationReceiver»
android: enabled=«true»
android: exported=«true»
android: process=":remote»>
</receiver>

<activity
android:name=".MainActivity»
android: label="@string/app_name»
android:theme="@style/AppTheme.NoActionBar»>
</activity>

<service
android:name=".StartAlarmIntentService»
android: exported=«false»>
</service>

<receiver
android:name=".BootReceiver»
android: enabled=«true»
android: exported=«true»>
<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED» />
</intent-filter>
</receiver>

<meta-data
android:name="com.google.android.gms.version»
android: value="@integer/google_play_services_version» />
</application>

<activity
android:name="com.google.android.gms.ads.AdActivity»
android:
configChanges=«keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreen
Size»
android:theme="@android:style/Theme.Translucent» />

</manifest>
```

Приложение начинается с активности, обеспечивающей авторизированный доступ к интерфейсу приложения.

Здесь атрибут android: label="@string/title_activity_login» активности обеспечивает показ заголовка, отличного от заголовка приложения.

Код этой активности:

```
import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation. TargetApi;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android. os. Build;
import android. os. Bundle;
import android.support. v7.app. AppCompatActivity;
import android. text. TextUtils;
import android.view.KeyEvent;
import android.view.View;
import android.view.View. OnClickListener;
import android.view.inputmethod. EditorInfo;
import android. widget. Button;
import android. widget. EditText;
import android. widget. TextView;

public class LoginActivity extends AppCompatActivity {

private static final String DUMMY_CREDENTIALS_LOGIN = new String («admin»);
private static final String DUMMY_CREDENTIALS_PASSWORD = new String («1234»);

private UserLoginTask mAuthTask = null;

// UI references.
private EditText mLoginView;
private EditText mPasswordView;
private View mProgressView;
private View mLoginFormView;
private SharedPreferences mSettings;
private Activity activity;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_login);
activity=this;
// Set up the login form.
mLoginView = (EditText) findViewById(R.id.email);

mPasswordView = (EditText) findViewById(R.id.password);
mPasswordView.setOnEditorActionListener (new TextView. OnEditorActionListener () {
@Override
public boolean onEditorAction (TextView textView, int id, KeyEvent keyEvent) {
if (id == R.id.login || id == EditorInfo.IME_NULL) {
attemptLogin ();
return true;
}
return false;
}
});
```

```
mSettings = getSharedPreferences («APP_PREFERENCES», Context.MODE_PRIVATE);

if(!mSettings.contains («APP_PREFERENCES_LOGIN»)) {
mLoginView.setHint (DUMMY_CREDENTIALS_LOGIN);

} else {
String prefLogin=mSettings.getString («APP_PREFERENCES_LOGIN»,»»);
mLoginView.setHint (prefLogin);
}

if(!mSettings.contains («APP_PREFERENCES_PASSWORD»)) {

mPasswordView.setHint (DUMMY_CREDENTIALS_PASSWORD);

} else {
mPasswordView.setHint («Password»);
}

Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);
mEmailSignInButton.setOnClickListener (new OnClickListener () {
@Override
public void onClick (View view) {
attemptLogin ();
}
});

mLoginFormView = findViewById(R.id.login_form);
mProgressView = findViewById(R.id.login_progress);
}

/**
* Attempts to sign in or register the account specified by the login form.
* If there are form errors (invalid email, missing fields, etc.), the
* errors are presented and no actual login attempt is made.
*/
private void attemptLogin () {
if (mAuthTask!= null) {
return;
}

// Reset errors.
mLoginView.setError (null);
mPasswordView.setError (null);

// Store values at the time of the login attempt.
String login = mLoginView.getText().toString ();
String password = mPasswordView.getText().toString ();

boolean cancel = false;
View focusView = null;

if (TextUtils.isEmpty (password)) {
mPasswordView.setError(getString(R.string.error_field_required));
focusView = mPasswordView;
cancel = true;
}
```

```
if (TextUtils.isEmpty (login)) {
mLoginView.setError(getString(R.string.error_field_required));
focusView = mLoginView;
cancel = true;
}

if (cancel) {
focusView.requestFocus ();
} else {
showProgress (true);
mAuthTask = new UserLoginTask (login, password);
mAuthTask. execute ((Void) null);
}
}

/**
* Shows the progress UI and hides the login form.
*/
@TargetApi(Build.VERSION_CODES. HONEYCOMB_MR2)
private void showProgress (final boolean show) {
// On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow
// for very easy animations. If available, use these APIs to fade-in
// the progress spinner.
if (Build.VERSION.SDK_INT> = Build.VERSION_CODES. HONEYCOMB_MR2) {
int shortAnimTime = getResources().getInteger(android.R.integer.config_shortAnimTime);

mLoginFormView.setVisibility (show? View. GONE: View.VISIBLE);
mLoginFormView.animate().setDuration(shortAnimTime).alpha (
show? 0: 1).setListener (new AnimatorListenerAdapter () {
@Override
public void onAnimationEnd (Animator animation) {
mLoginFormView.setVisibility (show? View. GONE: View.VISIBLE);
}
});

mProgressView.setVisibility (show? View.VISIBLE: View. GONE);
mProgressView.animate().setDuration(shortAnimTime).alpha (
show? 1: 0).setListener (new AnimatorListenerAdapter () {
@Override
public void onAnimationEnd (Animator animation) {
mProgressView.setVisibility (show? View.VISIBLE: View. GONE);
}
});
} else {
// The ViewPropertyAnimator APIs are not available, so simply show
// and hide the relevant UI components.
mProgressView.setVisibility (show? View.VISIBLE: View. GONE);
mLoginFormView.setVisibility (show? View. GONE: View.VISIBLE);
}
}

/**
* Represents an asynchronous login/registration task used to authenticate
* the user.
*/
```

```
public class UserLoginTask extends AsyncTask <Void, Void, Boolean> {

private final String mLogin;
private final String mPassword;

UserLoginTask (String login, String password) {
mLogin = login;
mPassword = password;
}

@Override
protected Boolean doInBackground (Void… params) {

Boolean success=true;

if(!mSettings.contains («APP_PREFERENCES_LOGIN»)) {

if (!DUMMY_CREDENTIALS_LOGIN. equals (mLogin)) success=false;

} else {

String prefLogin=mSettings.getString («APP_PREFERENCES_LOGIN»,»»);
if (!prefLogin. equals (mLogin)) success=false;

}

if(!mSettings.contains («APP_PREFERENCES_PASSWORD»)) {

if (!DUMMY_CREDENTIALS_PASSWORD. equals (mPassword)) success=false;

} else {

String prefLogin=mSettings.getString («APP_PREFERENCES_PASSWORD»,»»);
if (!prefLogin. equals (mPassword)) success=false;

}

return success;
}

@Override
protected void onPostExecute (final Boolean success) {
mAuthTask = null;
showProgress (false);

if (success) {
Intent intent = new Intent (activity, MainActivity.class);
startActivity (intent);
} else {
mPasswordView.setError(getString(R.string.error_incorrect_password));
mPasswordView.requestFocus ();
}
}

@Override
protected void onCancelled () {
```

```
mAuthTask = null;
showProgress (false);
}
}

@Override
protected void onStop () {
this.finish ();
super. onStop ();
}
}
```
Файл компоновки этой активности:
```
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: tools="http://schemas.android.com/tools" android: layout_width=«match_parent»
android: layout_height=«match_parent» android: gravity=«center_horizontal»
android: orientation=«vertical» android: paddingBottom="@dimen/activity_vertical_margin»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
tools:context="com.tmsoftstudio.babytracking. LoginActivity»>

<! – Login progress – >
<ProgressBar android: id="@+id/login_progress» style=»? android: attr/progressBarStyleLarge»
android: layout_width=«wrap_content» android: layout_height=«wrap_content»
android: layout_marginBottom=«8dp» android: visibility=«gone» />

<ScrollView android: id="@+id/login_form» android: layout_width=«match_parent»
android: layout_height=«match_parent»>

<LinearLayout android: id="@+id/email_login_form» android: layout_width=«match_parent»
android: layout_height=«wrap_content» android: orientation=«vertical»>

<android.support.design. widget. TextInputLayout android: layout_width=«match_parent»
android: layout_height=«wrap_content»>

<EditText
android: id="@+id/email»
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: inputType=«textEmailAddress»
android: maxLines=«1»
android: singleLine=«true» />

</android.support.design. widget. TextInputLayout>

<android.support.design. widget. TextInputLayout android: layout_width=«match_parent»
android: layout_height=«wrap_content»>

<EditText android: id="@+id/password»
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: imeActionId="@+id/login»
android: imeActionLabel="@string/action_sign_in_short»
android: imeOptions=«actionUnspecified»
android: inputType=«textPassword»
```

```
android: maxLines=«1»
android: singleLine=«true» />

</android.support.design. widget. TextInputLayout>

<Button android: id="@+id/email_sign_in_button»
style=»? android: textAppearanceSmall»
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: layout_marginTop=«16dp»
android: text="@string/action_sign_in»
android: textStyle=«bold» />

</LinearLayout>
</ScrollView>
</LinearLayout>
```

Активность отображает два поля для ввода логина и пароля и кнопку, при нажатии на которую происходит попытка авторизации.

Первоначально поля показывают в виде подсказок логин и пароль, установленные по умолчанию. В дальнейшем, после изменения логина и пароля, только поле логина показывает в виде подсказки установленное значение.

Значения логина и пароля сохраняются в файле SharedPreferences. Соответственно, при попытке авторизации, введенные пользователем значения сравниваются со значениями файла SharedPreferences.

Вызов метода finish () в методе onStop () исключает возврат к уже открытой активности и заставляет авторизироваться заново.

После успешной авторизации вызывается главная активность приложения:

```
import android.app.AlarmManager;
import android.app.AlertDialog;
import android. app. Dialog;
import android.app.PendingIntent;
import android.content.Context;
import android.content. DialogInterface;
import android.content.Intent;
import android.content.IntentSender;
import android.content.SharedPreferences;
import android.graphics.Color;
import android. os. Bundle;
import android.support.design. widget. FloatingActionButton;
import android.support.design.widget.NavigationView;
import android.support.design.widget.Snackbar;
import android.support. v4.app. DialogFragment;
import android.support.v4.view.GravityCompat;
import android.support. v4.widget. DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support. v7.app. AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android. util. TypedValue;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android. widget. Button;
import android. widget. EditText;
```

```
import android. widget. TextView;
import android.widget.Toast;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdView;
import com.google.android.gms. appindexing. AppIndex;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common. GooglePlayServicesUtil;
import com.google.android.gms.common. api. GoogleApiClient;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms. drive. Drive;
import com.google.android.gms. drive. DriveFolder;
import com.google.android.gms. drive. DriveId;
import com.google.android.gms. drive. DriveResource;
import com.google.android.gms.drive.MetadataChangeSet;

public class MainActivity extends AppCompatActivity
implements NavigationView. OnNavigationItemSelectedListener,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient. OnConnectionFailedListener {

private SharedPreferences mSettings;
private Context context;
private static long TRACKING_INTERVAL=15L*1000*60;
private static long SAVING_INTERVAL=24L*1000*60*60;
private GoogleApiClient mGoogleDriveApiClient;
private static final int RESOLVE_CONNECTION_REQUEST_CODE=1;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar (toolbar);

context = this;
mSettings = getSharedPreferences («APP_PREFERENCES», Context.MODE_PRIVATE);

AdView mAdView = (AdView) findViewById(R.id.adViewBan);
AdRequest adRequest = new AdRequest. Builder ().build ();
mAdView. loadAd (adRequest);

if (!mSettings.contains («TRACKING_INTERVAL»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
}

if (!mSettings.contains («SAVING_INTERVAL»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
}

mGoogleDriveApiClient = new GoogleApiClient. Builder (this)
.addApi (Drive. API)
```

```
.addScope(Drive.SCOPE_FILE)
.addConnectionCallbacks (this)
.addOnConnectionFailedListener (this)
.addApi (AppIndex. API).build ();
mGoogleDriveApiClient.connect ();

FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener (new View. OnClickListener () {
@Override
public void onClick (View view) {

Boolean start = mSettings.getBoolean («APP_PREFERENCES_START_ALARM», false);

if (start == false) {
long interval = mSettings.getLong («TRACKING_INTERVAL», 15L * 1000 * 60);

StartAlarmIntentService.startActionAlarm (context, interval);

SharedPreferences. Editor editor = mSettings. edit ();
editor. putBoolean («APP_PREFERENCES_START_ALARM», true);
editor.commit ();

Snackbar snack = Snackbar.make (view, «Tracking started with " + interval / (1000 * 60) + "
minutes interval», Snackbar. LENGTH_LONG);
View snackbarView = snack.getView ();
snackbarView.setBackgroundColor(Color.parseColor («#e1da21»));
TextView tv = (TextView)
snackbarView.findViewByIdandroid.support.design.R.id.snackbar_text);
tv.setGravity (Gravity. CENTER);
snack.setAction («Action», null).show ();
} else {
long interval = mSettings.getLong («TRACKING_INTERVAL», 15L * 1000 * 60);

Snackbar snack = Snackbar.make (view, «Tracking is already working with " + interval / (1000 *
60) + " minutes interval», Snackbar. LENGTH_LONG);
View snackbarView = snack.getView ();
snackbarView.setBackgroundColor(Color.parseColor («#e1da21»));
TextView tv = (TextView)
snackbarView.findViewByIdandroid.support.design.R.id.snackbar_text);
tv.setGravity (Gravity. CENTER);
snack.setAction («Action», null).show ();
}

}
});
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle (
this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
drawer.setDrawerListener (toggle);
toggle.syncState ();

NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener (this);

}
```

```
@Override
public void onSaveInstanceState (Bundle savedInstanceState) {
// Save custom values into the bundle
savedInstanceState. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
savedInstanceState. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
// Always call the superclass so it can save the view hierarchy state
super. onSaveInstanceState (savedInstanceState);
}

public void onRestoreInstanceState (Bundle savedInstanceState) {
// Always call the superclass so it can restore the view hierarchy
super. onRestoreInstanceState (savedInstanceState);
// Restore state members from saved instance
TRACKING_INTERVAL = savedInstanceState.getLong («TRACKING_INTERVAL»);
SAVING_INTERVAL = savedInstanceState.getLong («SAVING_INTERVAL»);
}

@Override
public void onBackPressed () {
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
if (drawer.isDrawerOpen(GravityCompat.START)) {
drawer.closeDrawer(GravityCompat.START);
} else {
super. onBackPressed ();
}
}

@Override
public boolean onCreateOptionsMenu (Menu menu) {
getMenuInflater().inflate(R.menu.main, menu);
long interval=mSettings.getLong («TRACKING_INTERVAL», 15L * 1000 * 60);
int itemID = getResources().getIdentifier («action_" + (int) (interval / (1000 * 60)) + "_minute»,
«id», getPackageName ());
MenuItem item = menu.findItem (itemID);
item.setChecked (true);

long saving_interval=mSettings.getLong («SAVING_INTERVAL», 24L * 1000 * 60 * 60);
itemID = getResources().getIdentifier («action_data_" + (int) (saving_interval/ (1000*60*60*24))
+"_day», «id», getPackageName ());
item = menu.findItem (itemID);
item.setChecked (true);
return true;
}

@Override
public boolean onOptionsItemSelected (MenuItem item) {
int id = item.getItemId ();

if (id == R.id.action_setting_account) {
DialogFragment dialog = new LoginDialogFragment ();
dialog.show (getSupportFragmentManager (), «LoginDialogFragment»);

return true;
}
```

```
if (id == R.id.action_data_1_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_2_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*2;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_3_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*3;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_4_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*4;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_5_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*5;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_6_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*6;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_7_day) {
```

```
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*7;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_14_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*14;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_data_30_day) {
item.setChecked (true);
SAVING_INTERVAL=24L*1000*60*60*30;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («SAVING_INTERVAL», SAVING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_1_minute) {
item.setChecked (true);
TRACKING_INTERVAL=1L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_5_minute) {
item.setChecked (true);
TRACKING_INTERVAL=5L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_10_minute) {
item.setChecked (true);
TRACKING_INTERVAL=10L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_15_minute) {
item.setChecked (true);
```

```
TRACKING_INTERVAL=15L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_30_minute) {
item.setChecked (true);
TRACKING_INTERVAL=30L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_60_minute) {
item.setChecked (true);
TRACKING_INTERVAL=60L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_240_minute) {
item.setChecked (true);
TRACKING_INTERVAL=240L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_720_minute) {
item.setChecked (true);
TRACKING_INTERVAL=720L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

if (id == R.id.action_1440_minute) {
item.setChecked (true);
TRACKING_INTERVAL=1440L*1000*60;
SharedPreferences. Editor editor = mSettings. edit ();
editor. putLong («TRACKING_INTERVAL», TRACKING_INTERVAL);
editor.commit ();
return true;
}

return super. onOptionsItemSelected (item);
}
```

```
@SuppressWarnings («StatementWithEmptyBody»)
@Override
public boolean onNavigationItemSelected (MenuItem item) {
// Handle navigation view item clicks here.
int id = item.getItemId ();

if (id == R.id.nav_start_tracking) {

Boolean start=mSettings.getBoolean («APP_PREFERENCES_START_ALARM», false);

if (start==false) {
long interval = mSettings.getLong («TRACKING_INTERVAL», 15L * 1000 * 60);
StartAlarmIntentService.startActionAlarm (context, interval);

SharedPreferences. Editor editor = mSettings. edit ();
editor. putBoolean («APP_PREFERENCES_START_ALARM», true);
editor.commit ();

View view = (View) findViewById(R.id.content);
Snackbar snack = Snackbar.make (view, «Tracking started with " + interval / (1000 * 60) + " minutes interval», Snackbar. LENGTH_LONG);
View snackbarView = snack.getView ();
snackbarView.setBackgroundColor(Color.parseColor («#e1da21»));
TextView tv = (TextView) snackbarView.findViewById(android.support.design.R.id.snackbar_text);
tv.setGravity (Gravity. CENTER);
snack.setAction («Action», null).show ();
} else {
long interval = mSettings.getLong («TRACKING_INTERVAL», 15L * 1000 * 60);
View view = (View) findViewById(R.id.content);
Snackbar snack = Snackbar.make (view, «Tracking is already working with " + interval / (1000 * 60) + " minutes interval», Snackbar. LENGTH_LONG);
View snackbarView = snack.getView ();
snackbarView.setBackgroundColor(Color.parseColor («#e1da21»));
TextView tv = (TextView) snackbarView.findViewById(android.support.design.R.id.snackbar_text);
tv.setGravity (Gravity. CENTER);
snack.setAction («Action», null).show ();
}

} else if (id == R.id.nav_stop_tracking) {
Intent intent = new Intent (getApplicationContext (), LocationReceiver.class);
final PendingIntent pIntent = PendingIntent.getBroadcast (getApplicationContext (), LocationReceiver.REQUEST_CODE, intent, PendingIntent. FLAG_CANCEL_CURRENT);
AlarmManager alarm = (AlarmManager) getApplicationContext().getSystemService(Context.ALARM_SERVICE);
alarm.cancel (pIntent);

SharedPreferences. Editor editor = mSettings. edit ();
editor. putBoolean («APP_PREFERENCES_START_ALARM», false);
editor.commit ();

View view= (View) findViewById(R.id.content);
Snackbar snack = Snackbar.make (view, «Tracking has stopped», Snackbar. LENGTH_LONG);
View snackbarView = snack.getView ();
```

```
    snackbarView.setBackgroundColor(Color.parseColor («#e1da21»));
    TextView tv = (TextView) snackbarView.findViewById(android.support.design.R.id.snackbar_text);
    tv.setGravity (Gravity. CENTER);
    snack.setAction («Action», null).show ();
    } else if (id == R.id.nav_clear_data) {
    LocationDBHelper dbhelper=LocationDBHelper.getInstance (context);
    dbhelper.clearLocation ();

    View view= (View) findViewById(R.id.content);
    Snackbar snack = Snackbar.make (view, «Location data removed», Snackbar. LENGTH_LONG);
    View snackbarView = snack.getView ();
    snackbarView.setBackgroundColor(Color.parseColor («#e1da21»));
    TextView tv = (TextView) snackbarView.findViewById(android.support.design.R.id.snackbar_text);
    tv.setGravity (Gravity. CENTER);
    snack.setAction («Action», null).show ();
    }

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
    }

    @Override
    protected void onStop () {
    mGoogleDriveApiClient. disconnect ();
    this.finish ();
    super. onStop ();
    }

    @Override
    public void onConnected (Bundle bundle) {

    if (mSettings.contains («TRACKING_FILE»)) {
    String fileID = mSettings.getString («TRACKING_FILE», «»);
    DriveId sFileId = DriveId.decodeFromString (fileID);
    DriveResource file = Drive.DriveApi.getFile (mGoogleDriveApiClient, sFileId);

    file.getMetadata(mGoogleDriveApiClient).setResultCallback (new ResultCallback<DriveResource.MetadataResult> () {
    @Override
    public void onResult(DriveResource.MetadataResult result) {
    if (!result.getStatus().isSuccess ()) {
    createFile ();
    } else {
    if(result.getMetadata().isTrashed ()) {
    createFile ();
    }
    }
    }
    });

    } else {
    createFile ();
```

```
    }

    }

    private void createFile () {
    MetadataChangeSet changeSet = new MetadataChangeSet. Builder ()
    .setTitle («BabyTracking»).build ();
    Drive.DriveApi.getRootFolder(mGoogleDriveApiClient).createFolder (
    mGoogleDriveApiClient, changeSet).setResultCallback (new ResultCallback <DriveFolder. DriveFolderResult> () {
    @Override
    public void onResult (DriveFolder. DriveFolderResult result_) {
    final DriveFolder. DriveFolderResult result = result_;

    if (!result.getStatus().isSuccess ()) {
    return;
    }

    String folderID = result.getDriveFolder().getDriveId().toString ();

    SharedPreferences. Editor editor = mSettings. edit ();
    editor. putString («TRACKING_FOLDER», folderID);
    editor.commit ();

    Toast.makeText (getApplicationContext (), «Folder BabyTracking created», Toast.LENGTH_LONG).show ();

    DriveFolder folder = result.getDriveFolder ();
    MetadataChangeSet changeSet = new MetadataChangeSet. Builder ()
    .setTitle («Tracking»)
    .setMimeType («text/plain»).build ();
    folder.createFile (mGoogleDriveApiClient, changeSet, null)
    .setResultCallback (new ResultCallback <DriveFolder. DriveFileResult> () {
    @Override
    public void onResult (DriveFolder. DriveFileResult result_) {
    final DriveFolder. DriveFileResult result = result_;
    if (!result.getStatus().isSuccess ()) {
    return;
    }
    String fileID = result.getDriveFile().getDriveId().toString ();
    SharedPreferences. Editor editor = mSettings. edit ();
    editor. putString («TRACKING_FILE», fileID);
    editor.commit ();

    Toast.makeText (getApplicationContext (), «File Tracking created», Toast.LENGTH_LONG).show ();

    }
    });
    }
    });
    }

    @Override
    public void onConnectionSuspended (int i) {
```

```
}

@Override
public void onConnectionFailed (ConnectionResult connectionResult) {
if (connectionResult. hasResolution ()) {
try {
connectionResult.startResolutionForResult (this, RESOLVE_CONNECTION_REQUEST_CODE);
} catch (IntentSender.SendIntentException e) {
}
} else {
GooglePlayServicesUtil.getErrorDialog(connectionResult.getErrorCode (), this, 0).show ();
}
}

@Override
protected void onActivityResult (final int requestCode, final int resultCode, final Intent data) {
switch (requestCode) {
case RESOLVE_CONNECTION_REQUEST_CODE:
if (resultCode == RESULT_OK) {
mGoogleDriveApiClient.connect ();
}
break;
}
}

@Override
public void onStart () {
super. onStart ();
mGoogleDriveApiClient.connect ();
}

public static class LoginDialogFragment extends DialogFragment {

private View view;
private AlertDialog alert;
private SharedPreferences mSettings;

@Override
public Dialog onCreateDialog (Bundle savedInstanceState) {
AlertDialog. Builder builder = new AlertDialog. Builder (getActivity ());
LayoutInflater inflater = getActivity().getLayoutInflater ();
view = (View) inflater.inflate(R.layout. dialog_login, null);

builder.setView (view)
// Add action buttons
.setPositiveButton(R.string. button_submit, new DialogInterface. OnClickListener () {
@Override
public void onClick (DialogInterface dialog, int id) {
}
})
.setNegativeButton(R.string. button_cancel, new DialogInterface. OnClickListener () {
public void onClick (DialogInterface dialog, int id) {
LoginDialogFragment.this.getDialog().cancel ();
}
});
```

```
alert = builder.create ();
alert.setOnShowListener (new DialogInterface. OnShowListener () {
@Override
public void onShow (DialogInterface dialog) {
Button btnPositive = alert.getButton (Dialog. BUTTON_POSITIVE);
btnPositive.setTextSize(TypedValue.COMPLEX_UNIT_PX, 25.0f);
btnPositive.setTextColor(Color.parseColor («#e1da21»));
btnPositive.setGravity (Gravity. CENTER);

Button btnNegative = alert.getButton (Dialog. BUTTON_NEGATIVE);
btnNegative.setTextSize(TypedValue.COMPLEX_UNIT_PX, 25.0f);
btnNegative.setTextColor(Color.parseColor («#e1da21»));
btnNegative.setGravity (Gravity. CENTER);
}
});
return alert;
}

@Override
public void onStart () {
super. onStart ();

alert.getButton(AlertDialog.BUTTON_POSITIVE).setOnClickListener (new View.
OnClickListener () {
@Override
public void onClick (View v) {
Boolean wantToCloseDialog = false;
EditText login = (EditText) view.findViewById(R.id.dialog_username);
EditText password = (EditText) view.findViewById(R.id.dialog_password);

if ((login. length ()!= 0) && (password. length ()!= 0)) {
mSettings = getContext().getSharedPreferences («APP_PREFERENCES»,
Context.MODE_PRIVATE);
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («APP_PREFERENCES_LOGIN», login.getText().toString ());
editor. putString («APP_PREFERENCES_PASSWORD», login.getText().toString ());
editor.commit ();

wantToCloseDialog=true;
} else {
Toast.makeText (getContext (), «The login or password is empty»,
Toast.LENGTH_LONG).show ();
}

if (wantToCloseDialog)
alert. dismiss ();
}
});
}
}
}
Файл компоновки activity_main:
<?xml version=«1.0» encoding=«utf-8»? >
http://schemas.android.com/apk/res/android"pport. v4.widget. DrawerLayout xmlns:
android="<android.su
```

xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools" android: id="@+id/drawer_layout»
android: layout_width=«match_parent» android: layout_height=«match_parent»
android: fitsSystemWindows=«true» tools: openDrawer=«start»>

<include layout="@layout/app_bar_main»
android: layout_width=«match_parent»
android: layout_height=«match_parent» />

<android.support.design.widget.NavigationView
android: id="@+id/nav_view»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_gravity=«start»
android: fitsSystemWindows=«true»
app: headerLayout="@layout/nav_header_main»
app: menu="@menu/activity_main_drawer»
app: itemTextColor="@color/item_drawer»
android: theme="@style/NavigationViewStyle»
/>
</android.support. v4.widget. DrawerLayout>
Файл компоновки app_bar_main:
<?xml version=«1.0» encoding=«utf-8»? >
<android.support.design.widget.CoordinatorLayout
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools" android: layout_width=«match_parent»
android: layout_height=«match_parent» android: fitsSystemWindows=«true»
tools:context=".MainActivity»>

<android.support.design. widget. AppBarLayout android: layout_height=«wrap_content»
android: layout_width=«match_parent» android: theme="@style/AppTheme. AppBarOverlay»>

<android.support.v7.widget.Toolbar android: id="@+id/toolbar»
android: layout_width=«match_parent»
android: layout_height=»? attr/actionBarSize»
android: background=»? attr/colorPrimary»
app: popupTheme="@style/AppTheme. PopupOverlay»
app: layout_scrollFlags=«scroll|enterAlways»
/>

</android.support.design. widget. AppBarLayout>

<android.support.v4.widget.NestedScrollView
xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: app="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: paddingBottom="@dimen/activity_vertical_margin»
android: fillViewport=«true»
android: layout_gravity=«fill_vertical»

```
app: layout_behavior="@string/appbar_scrolling_view_behavior»
tools:context=".MainActivity»
tools: showIn="@layout/app_bar_main»
>
<RelativeLayout
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
>

<com.google.android.gms.ads.AdView
android: id="@+id/adViewBan»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
xmlns: ads="http://schemas.android.com/apk/res-auto"
ads: adSize=«BANNER»
ads: adUnitId="@string/banner_ad_unit_id»
android: layout_alignParentTop=«true»
android: layout_centerHorizontal=«true»>
</com.google.android.gms.ads.AdView>

<RelativeLayout
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_below="@+id/adViewBan»
android: id="@+id/content»
>

<TextView android: text=«Start Tracking!»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android:textAppearance="@style/Base.TextAppearance.AppCompat.Large»
android: gravity=«right»
android: textColor=«#261860»
android: textStyle=«bold»
android: paddingLeft="@dimen/activity_horizontal_margin»
/>

</RelativeLayout>
</RelativeLayout>

</android.support.v4.widget.NestedScrollView>

<android.support.design. widget. FloatingActionButton
android: id="@+id/fab»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_gravity=«bottom|end»
android: layout_margin="@dimen/fab_margin»
android:          src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/@android:
drawable/ic_input_add»
app:layout_behavior="com.tmsoftstudio.babytracking.ScrollFABBehavior»
app: layout_anchorGravity=«bottom|center»
app: layout_anchor="@id/content»
/>
```

```
</android.support.design.widget.CoordinatorLayout>
```

Файл компоновки заголовка навигационной панели:

```
<?xml version=«1.0» encoding=«utf-8»? >
<LinearLayout xmlns: android="http://schemas.android.com/apk/res/android"
android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: background="@drawable/side_nav_bar»
android: paddingBottom="@dimen/activity_vertical_margin»
android: paddingLeft="@dimen/activity_horizontal_margin»
android: paddingRight="@dimen/activity_horizontal_margin»
android: paddingTop="@dimen/activity_vertical_margin»
android: theme="@style/ThemeOverlay. AppCompat. Dark»
android: orientation=«vertical»
android: gravity=«bottom»>

<ImageView
android: layout_width=«wrap_content»
android: layout_height=«100dp»
android:
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/@drawable/logotracking»
android: id="@+id/imageView»
/>

<TextView android: layout_width=«match_parent»
android: layout_height=«wrap_content»
android: text=«Baby Tracking»
android:textAppearance="@style/Base.TextAppearance.AppCompat.Large»
android: gravity=«center»
android: textColor=«#261860»
android: textStyle=«bold»
/>
</LinearLayout>
```

В методе onCreate активности создается и включается объект GoogleApiClient, обеспечивающий доступ к облачному хранению Google Drive.

При первой загрузке приложения открывается диалоговое окно, предлагающее выбрать аккаунт для авторизации доступа к сервису Google Drive. После успешной авторизации – для этого также нужно зарегистрировать для проекта OAuth 2.0 client ID – создается папка и файл в Google Drive для хранения данных трекинга.

Кнопка FloatingActionButton запускает сервис StartAlarmIntentService, который в свою очередь запускает сервис оповещений с установленной периодичностью:

```
import android.app.AlarmManager;
import android.app.IntentService;
import android.app.PendingIntent;
import android.content.Intent;
import android.content.Context;

public class StartAlarmIntentService extends IntentService {

static final String ACTION_START_ALARM =
"com.tmsoftstudio.babytracking.action.START_ALARM»;
static final String EXTRA_PARAM_INTERVAL =
"com.tmsoftstudio.babytracking.extra.PARAM_INTERVAL»;
static final String ACTION_START_ALARM_BOOT =
"com.tmsoftstudio.babytracking.action.START_ALARM_BOOT»;
```

```
public static void startActionAlarm (Context context, long param) {
Intent intent = new Intent (context, StartAlarmIntentService.class);
intent.setAction (ACTION_START_ALARM);
intent. putExtra (EXTRA_PARAM_INTERVAL, param);
context.startService (intent);
}

public StartAlarmIntentService () {
super («StartAlarmIntentService»);
}

@Override
protected void onHandleIntent (Intent intent) {
if (intent!= null) {
final String action = intent.getAction ();
if (ACTION_START_ALARM. equals (action)) {
final long param = intent.getLongExtra (EXTRA_PARAM_INTERVAL, 15L*1000*60);
if (param!=0) {
startAlarm (param);
this.stopSelf ();
}
}
if (ACTION_START_ALARM_BOOT. equals (action)) {
final long param = intent.getLongExtra (EXTRA_PARAM_INTERVAL, 15L*1000*60);
if (param!=0) {
startAlarm (param);
BootReceiver.completeWakefulIntent (intent);
this.stopSelf ();
}
}

}
}

public void startAlarm (long interval) {
Intent intent = new Intent (getApplicationContext (), LocationReceiver.class);
final PendingIntent pIntent = PendingIntent.getBroadcast (getApplicationContext (), LocationReceiver.REQUEST_CODE, intent, PendingIntent. FLAG_CANCEL_CURRENT);
long firstMillis = System.currentTimeMillis ();
AlarmManager alarm = (AlarmManager) getApplicationContext().getSystemService(Context.ALARM_SERVICE);
// alarm.setInexactRepeating (AlarmManager. RTC_WAKEUP, firstMillis, interval, pIntent);
alarm.setRepeating (AlarmManager. RTC_WAKEUP, firstMillis, interval, pIntent);
}
}
```

Сервис оповещений с установленной периодичностью запускает приемник LocationReceiver, запускающий сервис LocationIntentService сбора и сохранения дислокации:

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class LocationReceiver extends BroadcastReceiver {

public static final int REQUEST_CODE = 1;
```

```
public LocationReceiver () {
}

@Override
public void onReceive (Context context, Intent intent) {
Intent i = new Intent (context, LocationIntentService.class);
context.startService (i);
}
}
```

При перезагрузке устройства и ранее запущенном трекинге, сервис StartAlarmIntentService запускается автоматически с помощью приемника BootReceiver:

```
import android.content.SharedPreferences;
import android.support.v4.content. WakefulBroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class BootReceiver extends WakefulBroadcastReceiver {

private SharedPreferences mSettings;

public BootReceiver () {
}

@Override
public void onReceive (Context context, Intent intent) {

mSettings = context.getSharedPreferences («APP_PREFERENCES», Context.MODE_PRIVATE);

if(mSettings.contains («APP_PREFERENCES_START_ALARM»)) {
Boolean start = mSettings.getBoolean («APP_PREFERENCES_START_ALARM», false);
if (start == true) {
Intent service = new Intent (context, StartAlarmIntentService.class);
service.setAction(StartAlarmIntentService.ACTION_START_ALARM_BOOT);
long interval=mSettings.getLong («TRACKING_INTERVAL», 15L*1000*60);
service. putExtra (StartAlarmIntentService. EXTRA_PARAM_INTERVAL, interval);
startWakefulService (context, service);
}
}
}
}
```

При запуске сервис LocationIntentService создает и включает два объекта GoogleApiClient – один для доступа к сервису локации LocationServices, а другой для доступа к сервису Google Drive.

При соединении с сервисом локации LocationServices инициируется обновление информации о местоположении, которая записывается в базу данных приложения с помощью объектов LocationDB и LocationDBHelper:

```
public class LocationDB {
public String date;
public double latitude;
public double longitude;
}
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android. database. sqlite. SQLiteDatabase;
```

```
import android. database. sqlite. SQLiteOpenHelper;

import java.util.ArrayList;
import java.util.List;

public class LocationDBHelper extends SQLiteOpenHelper {

private static LocationDBHelper ourInstance;

private static final String DATABASE_NAME = «locationDatabase»;
private static final int DATABASE_VERSION = 1;

private static final String TABLE_LOCATION = «location»;

private static final String KEY_LOCATION_ID = «id»;
private static final String KEY_LOCATION_DATE = «date»;
private static final String KEY_LOCATION_LATITUDE = «latitude»;
private static final String KEY_LOCATION_LONGITUDE = «longitude»;

public static LocationDBHelper getInstance (Context context) {

if (ourInstance == null) {
ourInstance = new LocationDBHelper(context.getApplicationContext ());
}

return ourInstance;
}

private LocationDBHelper (Context context) {
super (context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate (SQLiteDatabase db) {
String CREATE_LOCATION_TABLE = «CREATE TABLE " + TABLE_LOCATION +
«(» +
KEY_LOCATION_ID + " INTEGER PRIMARY KEY,» +
KEY_LOCATION_DATE + " TEXT,» +
KEY_LOCATION_LATITUDE + " DOUBLE,» +
KEY_LOCATION_LONGITUDE + " DOUBLE)»;

db. execSQL (CREATE_LOCATION_TABLE);

}

@Override
public void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion) {
if (oldVersion!= newVersion) {
db. execSQL («DROP TABLE IF EXISTS " + TABLE_LOCATION);
onCreate (db);
}
}

public void addLocation (LocationDB locationDB) {

SQLiteDatabase db = getWritableDatabase ();
```

```
db.beginTransaction ();
try {
ContentValues values = new ContentValues ();
values. put (KEY_LOCATION_DATE, locationDB. date);
values. put (KEY_LOCATION_LATITUDE, locationDB.latitude);
values. put (KEY_LOCATION_LONGITUDE, locationDB. longitude);

db.insertOrThrow (TABLE_LOCATION, null, values);
db.setTransactionSuccessful ();
} catch (Exception e) {

} finally {
db. endTransaction ();
}
}
public void clearLocation () {
SQLiteDatabase db = getWritableDatabase ();
db. execSQL («DROP TABLE IF EXISTS " + TABLE_LOCATION);
onCreate (db);
}

public List <LocationDB> getLocations () {
List <LocationDB> locations = new ArrayList <> ();

String LOCATION_SELECT_QUERY = «SELECT * FROM " + TABLE_LOCATION;

SQLiteDatabase db = getReadableDatabase ();
Cursor cursor = db. rawQuery (LOCATION_SELECT_QUERY, null);
try {
if (cursor.moveToFirst ()) {
do {
LocationDB locationDB = new LocationDB ();
locationDB. date = cursor.getString(cursor.getColumnIndex (KEY_LOCATION_DATE));
locationDB.latitude = cursor.getDouble(cursor.getColumnIndex (KEY_LOCATION_LATITUDE));
locationDB. longitude = cursor.getDouble(cursor.getColumnIndex
(KEY_LOCATION_LONGITUDE));
locations.add (locationDB);
} while(cursor.moveToNext ());
}
} catch (Exception e) {

} finally {
if (cursor!= null && !cursor.isClosed ()) {
cursor.close ();
}
}
return locations;
}
}
```

При превышении заранее установленного размера базы данных, она автоматически стирается и запись информации начинается заново.

После записи информации сервис LocationIntentService автоматически останавливается.

При соединении с сервисом Google Drive происходит извлечение информации из базы данных и запись ее в файл каталога Google Drive.

Размер базы данных и периодичность сбора информации о местоположении устанавливаются в Options меню класса MainActivity.

Панель навигации класса MainActivity позволяет запустить трекинг, остановить трекинг и очистить базу данных.

Вызов метода finish () в методе onStop () предотвращает возврат к уже открытому интерфейсу приложения и заставляет авторизироваться заново для доступа к настройкам и управлению приложением.

Для просмотра собранного и сохраненного трекинга создадим отдельное приложение.

Файл манифеста этого приложения:

```
?xml version=«1.0» encoding=«utf-8»? >
<manifest xmlns: android="http://schemas.android.com/apk/res/android"
package="com.babytrackingviewer»>

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION» />
<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE» />

<application
android: allowBackup=«true»
android: icon="@mipmap/ic_launcher»
android: label="@string/app_name»
android: supportsRtl=«true»
android: theme="@style/AppTheme»>

<! —
The API key for Google Maps-based APIs is defined as a string resource.
(See the file «res/values/google_maps_api. xml»).
Note that the API key is linked to the encryption key used to sign the APK.
You need a different API key for each encryption key, including the release key that is used to
sign the APK for publishing.
You can define the keys for the debug and release targets in src/debug/ and src/release/.
– >
<meta-data
android:name="com.google.android.geo. API_KEY»
android: value="@string/google_maps_key» />

<activity
android:name=".MapsActivity»
android: label="@string/title_activity_maps»>
<intent-filter>
<action android:name="android.intent.action.MAIN» />

<category android:name="android.intent.category.LAUNCHER» />
</intent-filter>
</activity>

<activity
android:name="com.google.android.gms.ads.AdActivity»
android:
configChanges=«keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreen
Size»
android:theme="@android:style/Theme.Translucent» />

</application>
```

</manifest>
Приложение начинается с активности, которая загружает Google карту:

```
import android.content.Context;
import android.content.Intent;
import android.content.IntentSender;
import android. os. Bundle;
import android.support.v4.app.FragmentActivity;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdView;
import com.google.android.gms. appindexing. AppIndex;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common. GooglePlayServicesUtil;
import com.google.android.gms.common. api. GoogleApiClient;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms. drive. Drive;
import com.google.android.gms. drive. DriveApi;
import com.google.android.gms. drive. DriveContents;
import com.google.android.gms. drive. DriveFile;
import com.google.android.gms. drive. DriveFolder;
import com.google.android.gms. drive. DriveId;
import com.google.android.gms. drive. OpenFileActivityBuilder;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps. GoogleMap;
import com.google.android.gms.maps. OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
GoogleApiClient.ConnectionCallbacks, GoogleApiClient. OnConnectionFailedListener {

private GoogleMap mMap;
private GoogleApiClient mGoogleDriveApiClient;
private static final int RESOLVE_CONNECTION_REQUEST_CODE=1;
private static final int REQUEST_CODE_SELECT = 2;
private Context context;
private List <LocationDB> locations;

@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_maps);
context=this;
locations=new ArrayList <LocationDB> ();
// Obtain the SupportMapFragment and get notified when the map is ready to be used.
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager ()
.findFragmentById(R.id.map);
mapFragment.getMapAsync (this);
```

```
AdView mAdView = (AdView) findViewById(R.id.adViewBan);
AdRequest adRequest = new AdRequest. Builder ().build ();
mAdView. loadAd (adRequest);

mGoogleDriveApiClient = new GoogleApiClient. Builder (this)
.addApi (Drive. API)
.addScope(Drive.SCOPE_FILE)
.addConnectionCallbacks (this)
.addOnConnectionFailedListener (this)
.addApi (AppIndex. API).build ();
mGoogleDriveApiClient.connect ();
}

/**
* Manipulates the map once available.
* This callback is triggered when the map is ready to be used.
* This is where we can add markers or lines, add listeners or move the camera. In this case,
* we just add a marker near Sydney, Australia.
* If Google Play services is not installed on the device, the user will be prompted to install
* it inside the SupportMapFragment. This method will only be triggered once the user has
* installed Google Play services and returned to the app.
*/
@Override
public void onMapReady (GoogleMap googleMap) {
mMap = googleMap;
}

@Override
public void onConnected (Bundle bundle) {

IntentSender intentSender = Drive. DriveApi
.newOpenFileActivityBuilder ()
.setMimeType (new String [] {DriveFolder. MIME_TYPE, «text/plain»})
.build (mGoogleDriveApiClient);
try {
startIntentSenderForResult (
intentSender, REQUEST_CODE_SELECT, null, 0, 0, 0);
} catch (IntentSender.SendIntentException e) {

}
}

@Override
public void onConnectionSuspended (int i) {

}

@Override
public void onConnectionFailed (ConnectionResult connectionResult) {
if (connectionResult. hasResolution ()) {
try {
connectionResult.startResolutionForResult (this, RESOLVE_CONNECTION_REQUEST_CODE);
} catch (IntentSender.SendIntentException e) {
}
} else {
```

```
GooglePlayServicesUtil.getErrorDialog(connectionResult.getErrorCode (), this, 0).show ();
}
}

@Override
protected void onActivityResult (final int requestCode, final int resultCode, final Intent data) {
switch (requestCode) {
case RESOLVE_CONNECTION_REQUEST_CODE:
if (resultCode == RESULT_OK) {
mGoogleDriveApiClient.connect ();
}
break;
case REQUEST_CODE_SELECT:
if (resultCode == RESULT_OK) {
DriveId driveId = (DriveId) data.getParcelableExtra (
OpenFileActivityBuilder. EXTRA_RESPONSE_DRIVE_ID);
DriveFile selectedFile = Drive.DriveApi.getFile (mGoogleDriveApiClient, driveId);
selectedFile. open (mGoogleDriveApiClient, DriveFile.MODE_READ_ONLY, null)
.setResultCallback (new ResultCallback <DriveApi. DriveContentsResult> () {
@Override
public void onResult (DriveApi. DriveContentsResult result) {
if (!result.getStatus().isSuccess ()) {
return;
}
DriveContents contents = result.getDriveContents ();
BufferedReader reader = new BufferedReader (new InputStreamReader(contents.getInputStream
()));
StringBuilder builder = new StringBuilder ();
String line;
try {
while ((line = reader.readLine ())!= null) {
String [] tokens = line. split (»,»);
LocationDB location=new LocationDB ();
location. date=tokens [0];
location.latitude=Double.parseDouble (tokens [1]);
location.longitude=Double.parseDouble (tokens [2]);
locations.add (location);
}
reader.close ();
} catch (IOException e) {
e.printStackTrace ();
}

mMap.clear ();

for (LocationDB location: locations) {
LatLng pos = new LatLng(location.latitude, location. longitude);
mMap.addMarker (new MarkerOptions ().position (pos).title (location. date));
mMap.moveCamera(CameraUpdateFactory.newLatLng (pos));
}
}
});
}
break;
}
```

```
}

@Override
protected void onStop () {
super. onStop ();
if (mGoogleDriveApiClient!= null) {
mGoogleDriveApiClient. disconnect ();
}
}
}
```

Файл компоновки активности:

```
<RelativeLayout xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: tools="http://schemas.android.com/tools"
android: layout_width=«wrap_content»
xmlns: ads="http://schemas.android.com/apk/res-auto"
android: layout_height=«wrap_content»
tools:context=".MapsActivity»>

<com.google.android.gms.ads.AdView
android: id="@+id/adViewBan»
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
ads: adSize=«BANNER»
ads: adUnitId="@string/banner_ad_unit_id»
android: layout_alignParentTop=«true»
android: layout_centerHorizontal=«true»>

</com.google.android.gms.ads.AdView>

<RelativeLayout
android: layout_width=«wrap_content»
android: layout_height=«wrap_content»
android: layout_below="@+id/adViewBan»
>

<fragment xmlns: android="http://schemas.android.com/apk/res/android"
xmlns: map="http://schemas.android.com/apk/res-auto"
xmlns: tools="http://schemas.android.com/tools"
android: id="@+id/map»
android:name="com.google.android.gms.maps.SupportMapFragment»
android: layout_width=«match_parent»
android: layout_height=«match_parent»
tools:context="com.tmsoftstudio.babytrackingviewer.MapsActivity» />
</RelativeLayout>

</RelativeLayout>
```

Активность приложения создает и включает объект GoogleApiClient, обеспечивающий доступ к сервису Google Drive.

После соединения с сервисом Google Drive открывается диалоговое окно, предлагающее выбрать файл каталога Google Drive, содержащий информацию о местоположении для отображения на Google карте.

После выбора файла, информация из него извлекается и делится на дату, долготу и широту. Далее информация о местоположении добавляется на Google карту в виде маркеров.

Для того чтобы готовое приложение отображало Google карты, в Google консоле нужно создать Android ключ с указанием пакета приложения и SHA-1 certificate fingerprint сертификата готового приложения.

Для получения SHA-1 certificate fingerprint можно воспользоваться командной строкой:

C:\Users\user> cd C:\Program Files\Java\jdk1.8.0_65\bin

C:\Program Files\Java\jdk1.8.0_65\bin> keytool -list -keystore C:\Users\user\.android\keystore. jks

Enter keystore password:

Keystore type: JKS

Keystore provider: SUN

Your keystore contains 1 entry

entry, 04.06.2015, PrivateKeyEntry,

Certificate fingerprint (SHA1): 1D:0E:08:20:D2:A7:F9:CB:9C:77:31:EA: CE:32:…

После получения Android ключа, уничтожьте файл google_maps_key и в файле манифеста в теге <meta-data> прямо укажите Android ключ.

Создание мобильных Web сайтов

С появлением мобильных устройств и соответственно мобильных Web браузеров возникла необходимость адаптации или даже создания отдельных мобильных версий существующих Web сайтов, созданных для отображения десктопными Web-браузерами.

Мобильные устройства требуют адаптации Web сайтов для отображения экранами различных размеров, а также использования возможностей сенсорного экрана и учета ограниченности Интернет соединения и ресурсов мобильного устройства.

В качестве первого шага создания мобильного Web сайта можно использовать готовые шаблоны Responsive HTML5 CSS3 Website Templates (https://w3layouts.com/free-responsive-html5-css3-website-templates/).

Использование viewport мета тега улучшает представление Web контента на мобильном устройстве, позволяя установить ширину и масштаб представления.

<meta name = «viewport» content=«width=device-width, initial-scale=1»/>

Или

<meta name = «viewport» content=«user-scalable=no, width=device-width»/>

Для фиксации заголовка и подвала Web страницы используется CSS элемент position: fixed.

```
header {
position: fixed;
top: 0;
left: 0;
width: 100%;
}
```

Для отложенной загрузки изображений можно использовать JavaScript.

```
<div>
<p data-hash=«hash»>
</p>
</div>
$ (window).load (function () {
$('p[data-hash]').prepend (function (index) {
var hash = $(this).attr (’data-hash’)
return '<img width=«100» height=«100» alt=«» src="http://sample.com/' + hash + «»>»
})
})
div {
padding-left: 10px;
min-height: 100px;
position: relative;
}
div p img {
```

display: block;
position: absolute;
top: 0;
left: 0;
}

Вместо изображений можно использовать HTML символы (http://www.w3schools.com/html/html_symbols.asp).

Для организации звонка из Web страницы можно использовать ссылку в виде:

Call Us 1—800—555—0199

Использовать разные CSS стили для разного типа экранов:

<link rel=«stylesheet» type=«text/css» href=«style. css» media=«screen, handheld» />

<link rel=«stylesheet» type=«text/css» href=«enhanced. css» media=«screen and (min-width: 30em)» />

<! – [if (lt IE 9) & (!IEMobile)]>

<link rel=«stylesheet» type=«text/css» href=«enhanced. css» />

<! [endif] – >

Или

```
/*Default styles*/
li {
float: left;
width: 50%;
}
/*Display 3 per row for medium displays (like mobile phones in landscape or smaller tablets) */
@media screen and (min-width: 28.75em) {
li {
width: 33.3333333%;
}
}
/*Display 6 to a row for large displays (like medium tablets and up) */
@media screen and min-width: 40.5em) {
li {
width: 16.6666667%;
}
}
```

Или

<link rel=«stylesheet» media=» (max-width: 640px)» href=«max-640px. css»>

<link rel=«stylesheet» media=» (min-width: 640px)» href=«min-640px. css»>

<link rel=«stylesheet» media=» (orientation: portrait)» href=«portrait. css»>

<link rel=«stylesheet» media=» (orientation: landscape)» href=«landscape. css»>

Для небольших изображений использовать Base64 кодировку.

```
.data {
background: url (data: image/png; base64,iVBORw…) no-repeat 100% 43%;
}
```

Использовать раскрывающееся при нажатии меню сайта, скрытое изначально или использовать меню Hamburger Menu.

Проверять наличие сенсорного экрана, например, с помощью библиотеки Modernizr, и добавлять обработку жестов, например, с помощью библиотеки Hammer.

Использовать раскрывающийся при нажатии на заголовок текст, скрытый изначально.

Обеспечить оффлайн доступ к ресурсам сайта, используя интерфейс Service Workers (https://github.com/slightlyoff/ServiceWorker/blob/master/explainer.md).

Использовать для перевода пикселей в проценты формулу:

target ÷ context = result

```
h1 {
width: 70.85%; /* 700px / 988px = 0.7085 */
}
```

Использовать гибкие изображения:

```
img {
max-width: 100%;
}
```

Проанализировать, адаптирован ли Web сайт для мобильных устройств, можно с помощью сервиса https://www.google.com/webmasters/tools/mobile-friendly. Это поможет избежать таких сообщений от поисковой машины Google:

These pages will not be seen as mobile-friendly by Google Search, and will therefore be displayed and ranked appropriately for smartphone users.

Разработка мобильного Web сайта с помощью Web Starter Kit

Установим на компьютере платформу Node. js (https://nodejs.org), обеспечивающую выполнение JavaScript кода на стороне сервера.

Для проверки инсталляции наберите в командной строке:

```
node —v
```

Далее скачаем набор разработчика Web Starter Kit (https://developers.google.com/web/tools/starter-kit/).

Создадим Workspace каталог и скопируем в него файлы набора Web Starter Kit.

Для установки компилятора c++ и необходимых библиотек установим Visual Studio Community (https://www.visualstudio.com/products/visual-studio-community-vs) и создадим в ней после установки C++ проект, при этом установим все необходимые инструменты и библиотеки.

В командной строке, используя инструмент npm среды выполнения Node. js, наберем:

```
cd C:\Users\user\WebStarterKit
npm install
```

В результате будут установлены необходимые Node. js модули.

Далее в командной строке наберем:

```
npm install gulp —g
```

В результате будет установлен сборщик JS проектов Gulp. js.

Для проверки инсталляции в командной строке можно набрать:

```
gulp -v
```

Далее запустим локальный сервер с помощью команды:

```
gulp serve
```

В результате в браузере будет открыта страница index.html папки app. Завершить работу сервера можно с помощью комбинации клавиш Ctr – C.

Другие команды Gulp. js:

```
gulp
```

– сборка и оптимизация проекта. В результате в папке dist появится собранный и оптимизированный сайт.

gulp serve: dist

– тестирование собранного и оптимизированного сайта.

gulp pagespeed

– анализ скорости загрузки сайта.

Открыть сайт можно также с помощью локального Web сервера, например, установив Python (https://www.python.org/) и используя командную строку:

python -m SimpleHTTPServer 80

которая запускает простой HTTP сервер, идущий в комплекте с дистрибутивом Python.

Далее можно открыть адрес http://localhost/ в Web браузере и перейти в каталог http://localhost/WebStarterKit/app/.

Используя инструменты Chrome DevTools можно вести разработку сайта.

Создание мобильного Web сайта на основе Vosao CMS

Помимо создания Backend модулей для Android приложений, платформа Google App Engine for Java обеспечивает полноценный бесплатный хостинг для развертывания динамических сайтов, созданных с использованием языка Java.

Vosao CMS представляет собой систему управления контентом, предназначенную для развертывания на платформе Google App Engine for Java.

Шаблон сайта по умолчанию, поставляемый с Vosao CMS, не адаптирован для мобильных устройств, поэтому требуется установка нового шаблона, реализующего принципы Responsive Web Design with HTML5 and CSS3.

Для развертывания Vosao CMS используем Eclipse IDE for Java EE Developers с установленным плагином Google Plugin for Eclipse.

По умолчанию среда Eclipse использует не набор JDK, а среду выполнения JRE, поэтому после установки плагина Google Plugin for Eclipse, перед созданием GAE-проекта, в меню Windows среды Eclipse выберем команду Preferences, в диалоговом окне которой откроем раздел Java | Installed JREs и кнопкой Add добавим предварительно инсталлированный набор JDK.

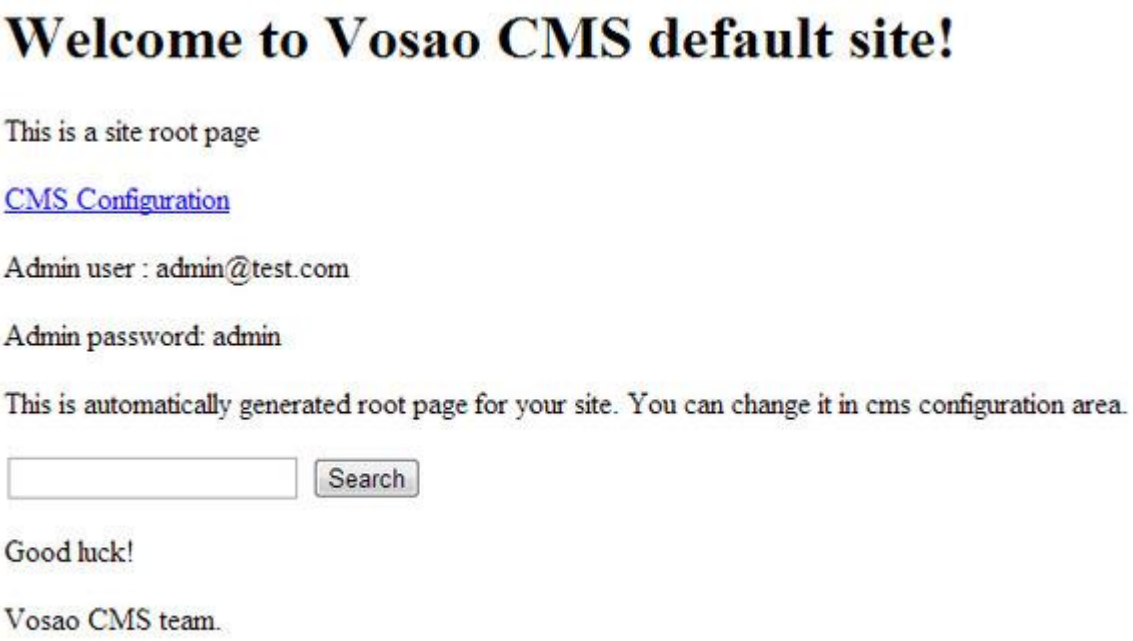
Скачаем WAR файл Vosao CMS (http://code.google.com/p/vosao/downloads/list) и распакуем его. Уберем из дистрибутива файл backends. xml.

Сделаем слияние содержимого каталога war проекта VosaoCMS с содержимым дистрибутива.

Для развертывания проекта на платформе GAE, его необходимо предварительно зарегистрировать. Для регистрации проекта откроем Web-браузер и войдем в консоль администрирования по адресу https://appengine.google.com/.

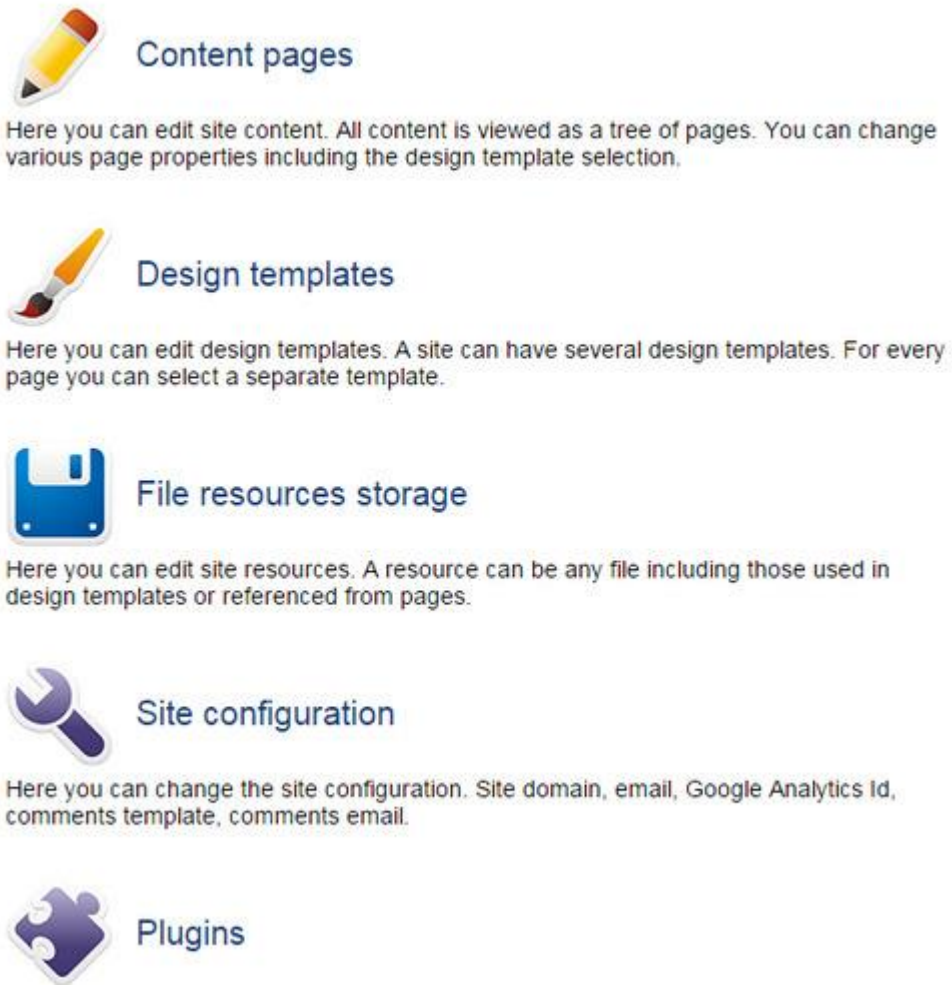
После регистрации проекта для его загрузки в App Engine в среде Eclipse откроем в редакторе файл appengine-web. xml папки war/WEB-INF проекта и в тэг <application> вставим зарегистрированный идентификатор приложения. Сохранив сделанные изменения, с помощью кнопки Sign in to Google пройдем авторизацию для доступа к Google-сервисам. В окне Project Explorer щелкнем правой кнопкой мышки на узле проекта и в контекстном меню выберем команду Google | Deploy to App Engine, в диалоговом окне мастера нажмем кнопку Deploy. В результате проект будет развернут на GAE-платформе.

В Web-браузере откроем консоль администрирования https://appengine.google.com/ GAE-платформы, в разделе Application которой, нажатием ссылки приложения откроем станицу администрирования приложения. В разделе Application Settings | Application Default Version URL: станицы администрирования приложением будет указан URL-адрес запроса к приложению, открывающий его страницу приветствия.



Откроем ссылку CMS Configuration.

Введем логин admin@test.com и пароль admin.



Откроем ссылку Конфигурация и меню Пользователи. Кнопкой Добавить добавим нового администратора сайта.

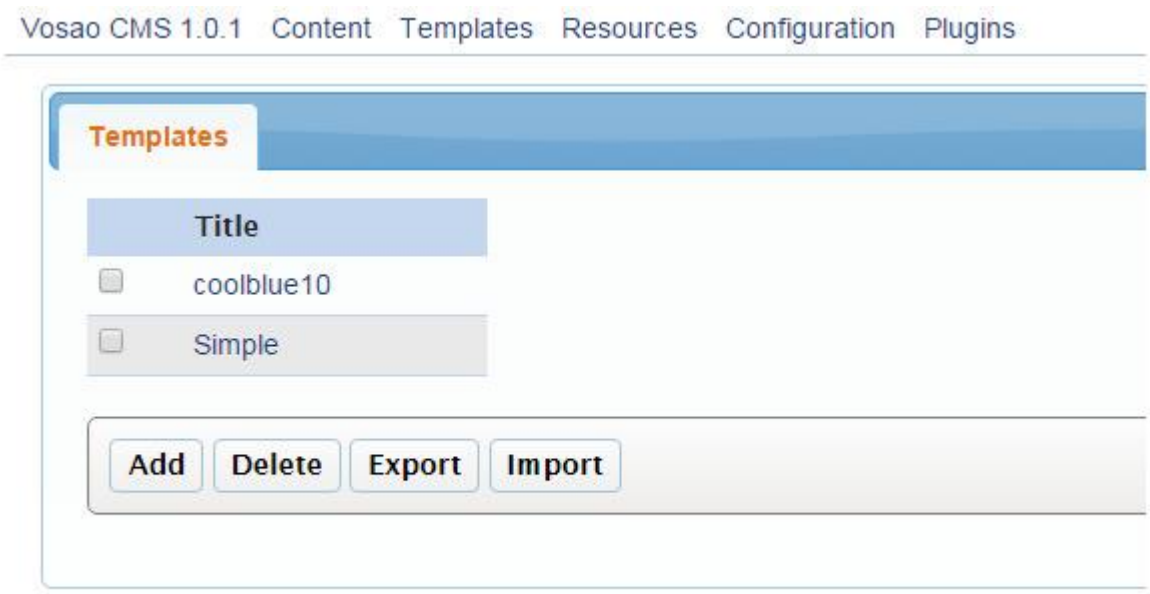
Зайдем в консоль управления сайтом .appspot.com/cms/# в качестве нового администратора сайта.

Откроем ссылку Конфигурация и меню Пользователи. Кнопкой Удалить удалим старого администратора сайта.

Теперь логин admin@test.com и пароль admin действовать не будут, а для входа в консоль управления сайтом .appspot.com/cms/# нужно будет вводить логин и пароль нового администратора сайта.

Меню Профиль консоли управления сайтом позволяет изменить пароль администратора, а меню Язык – локализацию консоли управления сайтом.

Откроем меню Шаблоны.



Шаблон сайта устанавливает общий внешний вид для всех страниц сайта, определяя содержимое шапки страницы, ее подвала и боковой панели.

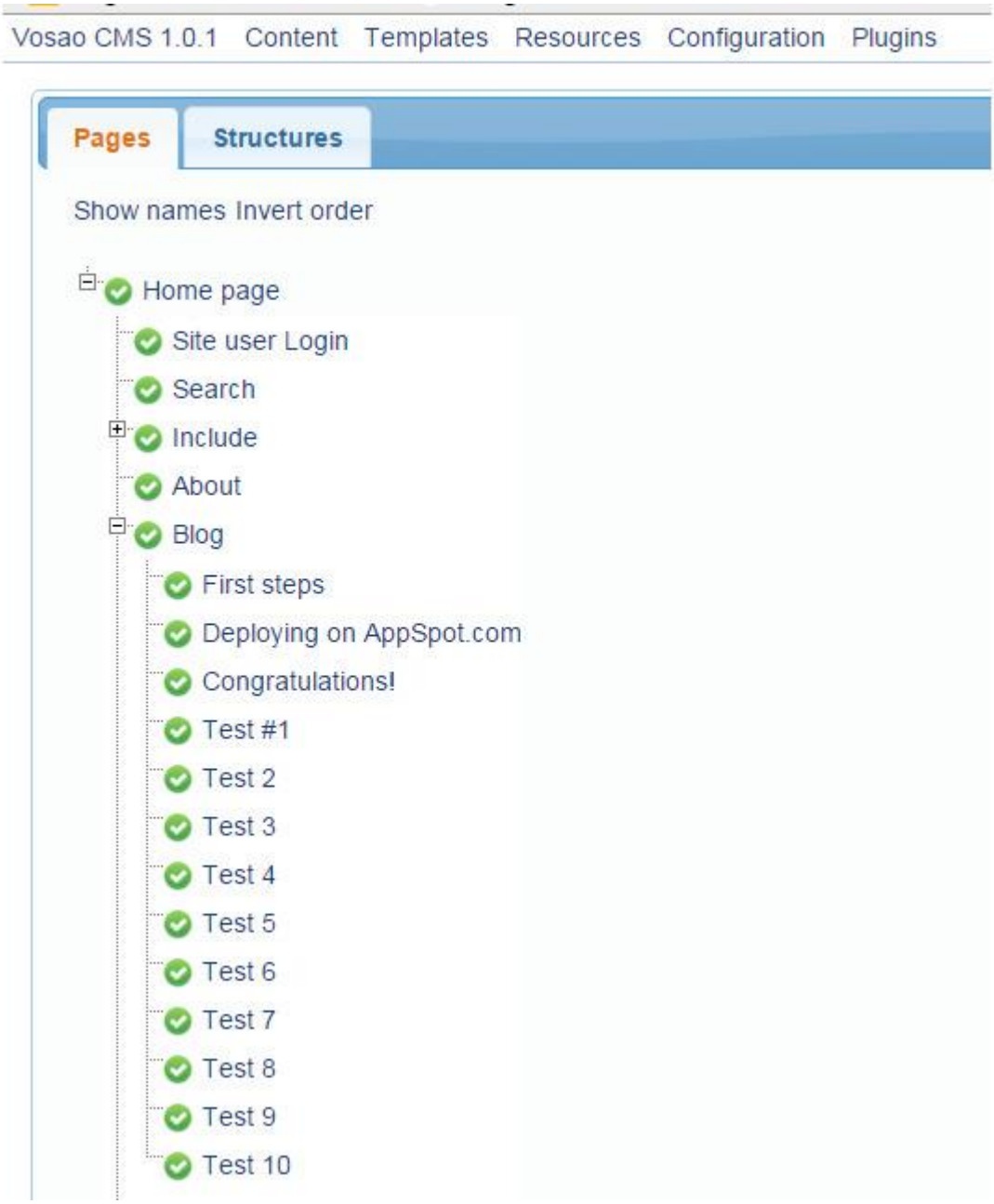
Шаблон сайта состоит из страницы с разметкой шапки, подвала и боковой панели, а также блока для включения содержимого остальных страниц сайта, файлов CSS-стилей и файлов изображений шаблона.

Кроме того, шаблон сайта может содержать команды для генерации динамических страниц и специальных эффектов.

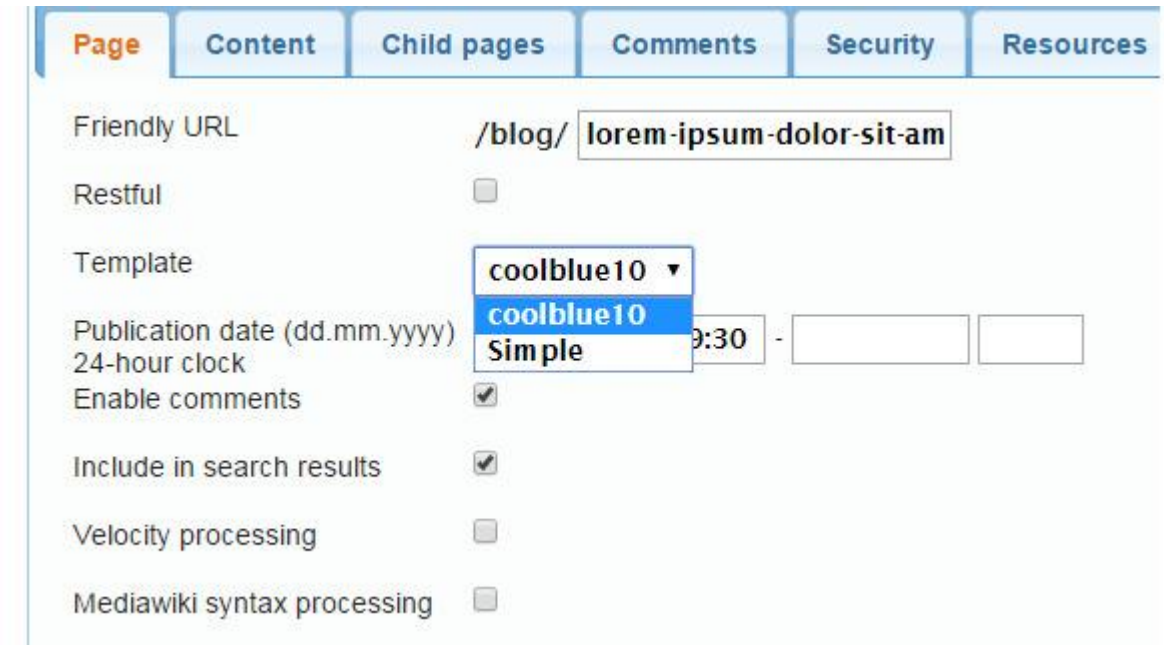
Каждая страница сайта может иметь свой шаблон или все страницы сайта могут иметь общий шаблон.

Шаблон назначается для страницы сайта следующим образом:

– Открываем меню Контент консоли.



– Выбираем страницу и открываем Редактировать свойства страницы.
– Открываем список шаблонов и выбираем шаблон.



– Нажимаем кнопку Сохранить.

Vosao-шаблоны обрабатываются движком Apache Velocity Engine.

Перечень некоторых Velocity-команд:

(PageEntity) $service.findPage (path) – возвращает последнюю опубликованную версию страницы по ее ЧПУ. Объект PageEntity, возвращаемый командой, представляет страницу и имеет такие свойства как title, friendlyURL, content и comments.

(List) $service.findPageChildren (path, [count]), (List) $service.findPageChildren (path, [publishDate]), (List) $service.findPageChildrenOrdered (path, [count]) – возвращают список последних опубликованных версий дочерних страниц.

(List) $service.getCommentsByPage (path) – возвращает список объектов CommentEntity страницы. Объект CommentEntity представляет комментарий и имеет свойства content, publishDate, pageID и disabled.

(Text) $service.findContent (path, [language]) – возвращает содержимое страницы.

(List) $service.findChildrenContent (path, [language]) – возвращает содержимое дочерних страниц.

(UserEntity) $service.findUser (email) – возвращает объект UserEntity по адресу электронной почты. Объект UserEntity представляет пользователя и имеет свойства name, email, password (зашифрованный) и role.

(Text) $page. title – возвращает заголовок страницы.

(Text) $page.content – возвращает содержимое страницы.

(Text) $page.friendlyURL – возвращает локальный адрес страницы.

(PageEntity) $page.parent – возвращает родительскую страницу.

(Text) $page. template – возвращает шаблон страницы.

(Date) $page. publishDate> – возвращает дату публикации страницы.

(Text) $page.comments – возвращает комментарии страницы.

(Text) $config.commentsEmail – возвращает адрес уведомления комментариев.

(Text) $config.commentsTemplate – возвращает шаблон комментариев.

(Text) $config.siteDomain – возвращает домен сайта.

(Text) $config.siteEmail – возвращает адрес электронной почты сайта.

(Text) $config.formTemplate – возвращает шаблон форм.

Для автоматического заполнения карты сайта скачаем плагин Instant Sitemap по адресу http://code.google.com/p/vosao/downloads/list – файл sitemap-0.9.war и установим его с помощью меню Расширения | Конфигурация расширений.

После установки, открыв ссылку плагина, можно определить глубину карты и исключить из нее URL-адреса.

Для создания файла sitemap. xml откроем меню Шаблоны и добавим шаблон с заголовком sitemap. xml, URL-именем sitemap. xml и содержимым в виде строки:

$plugin.sitemap.renderXML ()

Откроем меню Контент и добавим потомка к домашней странице с URL-адресом /sitemap. xml. Назначим в качестве шаблона страницы шаблон sitemap. xml и отметим флажок Пропустить пост обработку, после чего отметим флажок Подтвердить при сохранении и нажмем кнопку Сохранить.

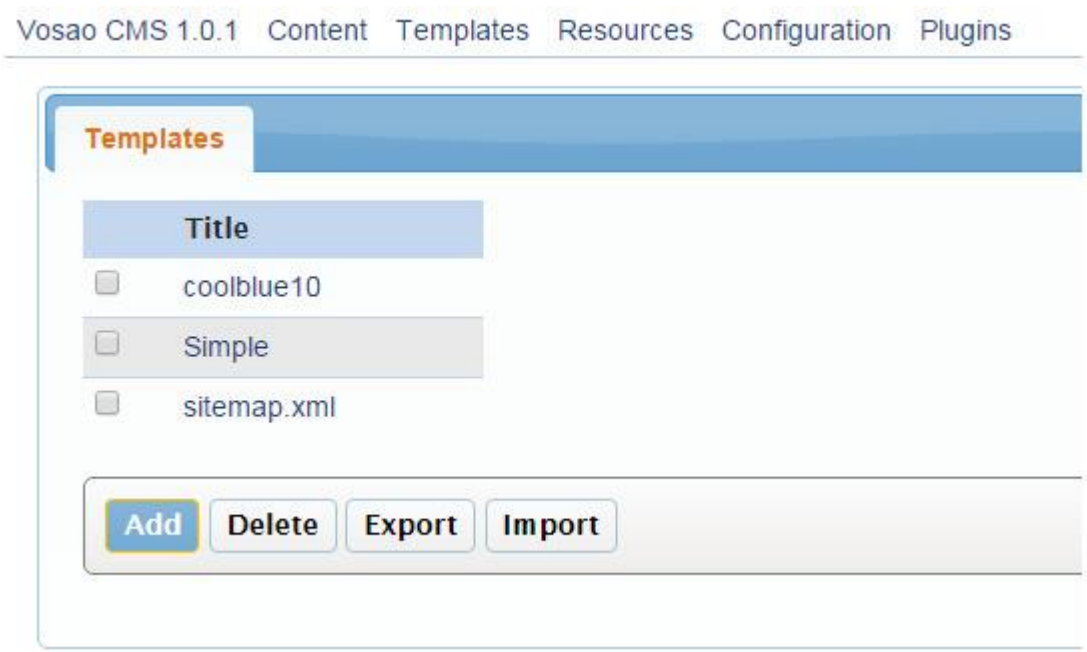
Для создания файла robots. txt откроем меню Шаблоны и добавим шаблон с заголовком robots. txt, URL-именем robots. txt и содержимым в виде строки:

Sitemap: http://…/sitemap.xml

Откроем меню Контент и добавим потомка к домашней странице с URL-адресом /robots. txt. Назначим в качестве шаблона страницы шаблон robots. txt и отметим флажок Пропустить пост обработку, после чего отметим флажок Подтвердить при сохранении и нажмем кнопку Сохранить.

Для изменения шаблона сайта скачаем бесплатный шаблон с сайта https://w3layouts.com и распакуем его архив.

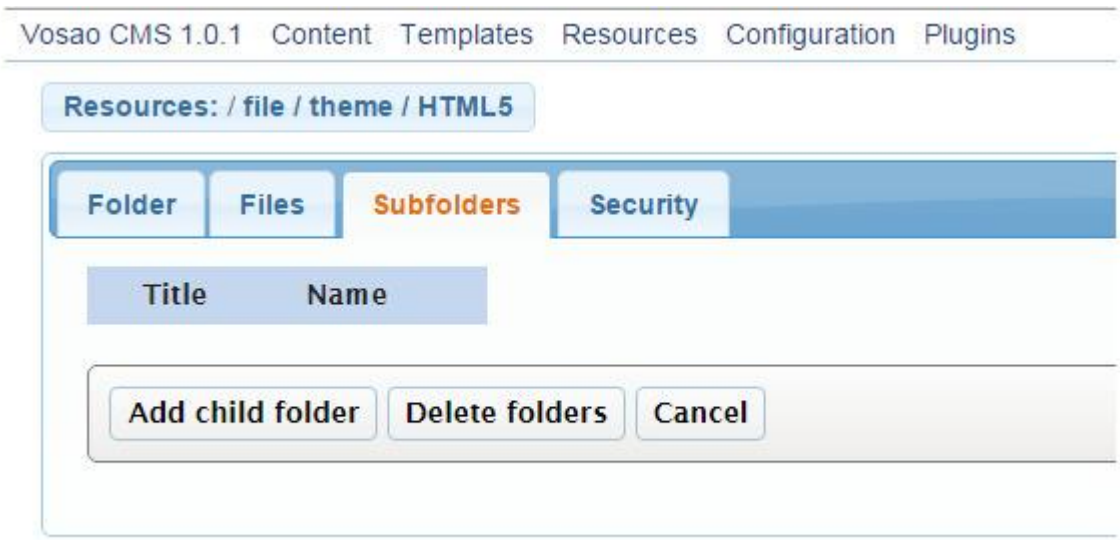
Войдем в Vosao-консоль и откроем меню Шаблоны, нажмем кнопку Добавить.



Введем заголовок и URL-имя шаблона и в рабочую область редактора шаблона скопируем содержимое файла index.html скачанного шаблона.

Нажмем кнопку Сохранить и продолжить.

Откроем вкладку Ресурсы, затем вкладку Подпапки.



С помощью кнопки Добавить подпапку создадим папки для CSS-стилей и изображений шаблона.

Открыв созданную папку откроем вкладку Файлы и с помощью кнопки Загрузить файл загрузим соответствующие файлы скачанного шаблона.

Далее требуется проверить все пути в HTML-файле и CSS-файле шаблона, добавляя /file/theme/HTML5/.

Возможно, для правильной работы скриптов, при назначении шаблона странице, нужно будет отметить флажок Пропустить пост обработку.

Уберем в шаблоне статический контент и добавим вместо него команду:

$page.content

Данная команда будет динамически вставлять контент страницы, к которой прикреплен шаблон.

Если требуется создание страницы, содержащей обзоры дочерних страниц каталога, тогда нужно сначала получить все страницы каталога:

#set ($pagesAll = ${service.findPageChildren (»/digital-camera»)})

Затем посчитать их количество:

#set ($count = $pagesAll.size ())

Вывести первые четыре обзора:

#set ($pagesActive = ${service.findPageChildren (»/digital-camera», 0, 4)})
#foreach ($p in $pagesActive)
<li class=«span3»>

${service.renderStructureContent($p.friendlyURL, «article-overview»)}

#end

Затем вставить остальные обзоры для прокрутки:

#set ($pages = ${service.findPageChildren (»/digital-camera», 5, $count)})
#foreach ($p in $pages)
<li class=«span3»>

${service.renderStructureContent($p.friendlyURL, «article-overview»)}

#end

В свойствах такой страницы нужно не забыть отметить флажок Velocity обработка.

Быстрая адаптация сайта для мобильных устройств с помощью Bootstrap

Возьмем в качестве примера простой шаблон с фиксированной шириной, с меню, боковой панелью, контентом и подвалом.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN» "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml: lang=«en» lang=«en»>
<head>
<title> </title>
<meta http-equiv=«content-type» content=«application/xhtml+xml; charset=UTF-8» />
<meta name=«author» content=«» />
<meta name=«description» content=«» />
<meta name=«robots» content=«index, follow» />
<link rel=«stylesheet» type=«text/css» media=«screen» href="/file/theme/css/screen. css» />
</head>

<body>

<! – header – >
<div id=«header-wrap»> <div id=«header»>
<a name=«top»> </a>
<h1 id=«logo-text»> <a href=«#» title=«»> </a> </h1>
<p id=«slogan»> </p>
<div id=«nav»>
<ul>
<li> <a href=«»> Home </a> </li>
<li> <a href=«#»> Page1 </a>
<ul style=«height: auto; overflow: auto»>
<li> <a href=«#»> Page1—1 </a> </li>
</ul>
</li>
<li> <a href=«#»> Page2 </a>
<ul style=«height: auto; overflow: auto»>
<li> <a href=«#»> Page2—1 </a> </li>
<li> <a href=«#»> Page2—1 </a> </li>
</ul>
</li>
</ul>
</div>

<div id=«google-search»>
<script>
(function () {
var cx = «»;
var gcse = document.createElement (’script’);
gcse. type = ’text/javascript’;
gcse.async = true;
gcse.src = (document.location.protocol == ’https:»? ’https:': ’http:») +
«//www.google.com/cse/cse.js?cx=' + cx;
var s = document.getElementsByTagName (’script’) [0];
s.parentNode.insertBefore (gcse, s);
}) ();
</script>
```

```
<gcse: search> </gcse: search>
</div>

</div>
</div>

<! – content-outer – >

<div id=«content-wrap» class=«clear»>
<div id=«content»>
<div style=«width:1150px; height:15px; background: url(/file/theme/images/top.png);»>
</div>
<div id=«main»>

</div>

<! – sidebar – >
<div id=«sidebar» style=«width:250px;»>
<div class=«sidemenu»>
<ul>
<li> <a href=«»> Home </a> </li>
<li> <a href=«#»> Page1 </a>
<ul style=«height: auto; overflow: auto»>
<li> <a href=«#»> Page1—1 </a> </li>
</ul>
</li>
<li> <a href=«#»> Page2 </a>
<ul style=«height: auto; overflow: auto»>
<li> <a href=«#»> Page2—1 </a> </li>
<li> <a href=«#»> Page2—1 </a> </li>
</ul>
</li>
</ul>
</div>
<! – /sidebar – > </div>

</div>
<! – /content-out – >
</div>

<! – footer-outer – >

<div id=«footer-outer» class=«clear»>
<div id=«footer-wrap»>
<div style=«margin-left:500px; float: left»>
<p> <strong> <a href=«#top»> Top </a> </strong> </p>
<p style=«margin-left:-100px;»>
&copy; 2015 <strong> </strong>
<strong style=«margin-left:20px;"> E-mail: </strong>
</p>
</div>
</div>

<! – /footer-outer – >
</div>
```

</body>
</html>
Заменим DOCTYPE на <!DOCTYPE html>.
Атрибут lang в теге html и атрибут charset тега meta уже имеются.
В начало тега <head> добавим:
<! – Latest compiled and minified CSS – >
<link rel=«stylesheet» href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
<! – jQuery library – >
<script src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
<! – Latest compiled JavaScript – >
<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
В тег <head> добавим:
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
Обернем заголовок в класс. container и, используя Bootstrap Grid System и Bootstrap Navigation Bar, разместим меню, логотип и форму поиска в шапке.
<div class=«container header»>

<nav class=«navbar navbar-inverse»>

<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=«#myNavbar»>

</button>

<div class=«collapse navbar-collapse» id=«myNavbar»>
<ul class=«nav navbar-nav»>
  Home
<li class=«dropdown»>
 Page1
<ul class=«dropdown-menu inverse-dropdown»>
  Page1—1

<li class=«dropdown»>
 Page2
<ul class=«dropdown-menu inverse-dropdown»>
  Page2—1
  Page2—2
  Page2—3
  Page2—4
  Page2—5

  Contacts

</nav>

```
</div>
<a name=«top»> </a>

<div class=«row»>
<div class=«col-sm-6 text-center»>
<h1> <a href="/"> </a> </h1>
<p> </p>
</div>
<div class=«col-sm-4 pull-right»>
<script>
(function () {
var cx = «»;
var gcse = document.createElement (’script’);
gcse. type = ’text/javascript’;
gcse.async = true;
gcse.src = (document.location.protocol == ’https:»? ’https:': ’http:») +
'//cse.google.com/cse. js? cx=' + cx;
var s = document.getElementsByTagName (’script’) [0];
s.parentNode.insertBefore (gcse, s);
}) ();
</script>
<gcse: searchbox-only> </gcse: searchbox-only>
</div>
<style>
.gsc-search-button {
display: none;
}

.gsib_a {
height:40px;
}

.gsc-input-box {

height: 40px;
}
</style>
<div style=«height:15px; background: black;" class=«col-sm-12»>
</div>
</div>
</div>
```
Уберем боковую панель и свой файл CSS, вместо него добавим CSS стили:
```
<style>
body {
font-size:18px;
font-famile: Georgia;
color: #000;
margin: 0;
padding: 0;
background: gray;

}

.header {
```

```
/* Permalink – use to edit and share this gradient: http://colorzilla.com/gradient-editor/#e1ffff+0,e1ffff+4,fdffff+7,fdffff+7,e1ffff+17,e6f8fd+28,c8eefb+47,bee4f8+68,b1d8f5+100 */
background: #e1ffff; /* Old browsers */
background: -moz-linear-gradient (top, #e1ffff 0%, #e1ffff 4%, #fdffff 7%, #fdffff 7%, #e1ffff 17%, #e6f8fd 28%, #c8eefb 47%, #bee4f8 68%, #b1d8f5 100%); /* FF3.6—15 */
background: -webkit-linear-gradient (top, #e1ffff 0%,#e1ffff 4%,#fdffff 7%,#fdffff 7%,#e1ffff 17%,#e6f8fd 28%,#c8eefb 47%,#bee4f8 68%,#b1d8f5 100%); /* Chrome10—25,Safari5.1—6 */
background: linear-gradient (to bottom, #e1ffff 0%,#e1ffff 4%,#fdffff 7%,#fdffff 7%,#e1ffff 17%,#e6f8fd 28%,#c8eefb 47%,#bee4f8 68%,#b1d8f5 100%); /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
filter: progid:DXImageTransform.Microsoft.gradient (startColorstr=«#e1ffff», endColorstr=«#b1d8f5», GradientType=0); /* IE6—9 */

}

.inverse-dropdown {
background-color: black;
}
.inverse-dropdown li a {
color: white;
}

.inverse-dropdown li a: hover {
color: black;
}

.content {
background: lightgrey;
}

.footer {
/* Permalink – use to edit and share this gradient: http://colorzilla.com/gradient-editor/#1e5799+0,2989d8+86,207cca+93,7db9e8+100 */
background: #1e5799; /* Old browsers */
background: -moz-radial-gradient (center, ellipse cover, #1e5799 0%, #2989d8 86%, #207cca 93%, #7db9e8 100%); /* FF3.6—15 */
background: -webkit-radial-gradient (center, ellipse cover, #1e5799 0%,#2989d8 86%,#207cca 93%,#7db9e8 100%); /* Chrome10—25,Safari5.1—6 */
background: radial-gradient (ellipse at center, #1e5799 0%,#2989d8 86%,#207cca 93%,#7db9e8 100%); /* W3C, IE10+, FF16+, Chrome26+, Opera12+, Safari7+ */
filter: progid:DXImageTransform.Microsoft.gradient (startColorstr=«#1e5799», endColorstr=«#7db9e8», GradientType=1); /* IE6—9 fallback on horizontal gradient */

}

.navigation {
margin: 10px 20px; padding-bottom: 10px;
width: 560px;
}
.navigation a: link,
.navigation a: visited {
float: left;
display: block;
margin: 10px 10px 0 0;
```

```
padding: 5px 7px;
text-transform: lowercase;
text-decoration: none;
font-weight: bold;
color: #fff;
background: #2C76A6;
border-width: 1px;
border-style: solid;
border-color: #86BBDF #245F86 #245F86 #86BBDF;
}
.navigation a: hover {
background: #FF3399;
border-width: 1px;
border-style: solid;
border-color: #FF75BA #EA0075 #EA0075 #FF75BA;
}

</style>
```

Таким образом, исключим из шаблона все фоновые изображения, которые могут создавать горизонтальную прокрутку, заменим фоновые изображения на CSS градиенты.

Обернем контент и подвал в класс. container и используем Bootstrap Grid System.

```
<! – content-outer – >
<div class=«container content»>
<div class=«row»>
<div class=«col-sm-11 col-md-offset-1»>
<span class=«pull-right»>
<img
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4//file/theme/add.jpg» alt=«»
width=200/>
</span>
<div style=«padding-top:50px;»>
<script async
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///pagead2.googlesyndication.
com/pagead/js/adsbygoogle. js»> </script>
<! – adds – >
</script>
</div>
</div>
</div>
<div class=«row»>
<div class=«col-sm-10»>
<! – content – >
<p>
 
</p>
</div>
</div>
</div>

<! – /content-out – >

<! – footer-outer – >
<div class=«container footer»>
```

<p style=«position: fixed; bottom: 12px; right: 5px; opacity: 1; cursor: pointer;"> Top

<div class=«col-*-* text-center»>

<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///yandex.st/share/share. js» charset=«utf-8»> </script>

<div class=«row» style=«color: darkblue;»>
<div class=«col-*-* text-center»>

© 2016
 admin@.com

<! – /footer-outer – >

Создание кросс-платформенного Backend приложения

Такая задача может возникнуть, если требуется создание для Backend модуля Android приложения полноценного приложения с отдельной функциональностью, которое должно быть как Web приложением, так и мобильным приложением.

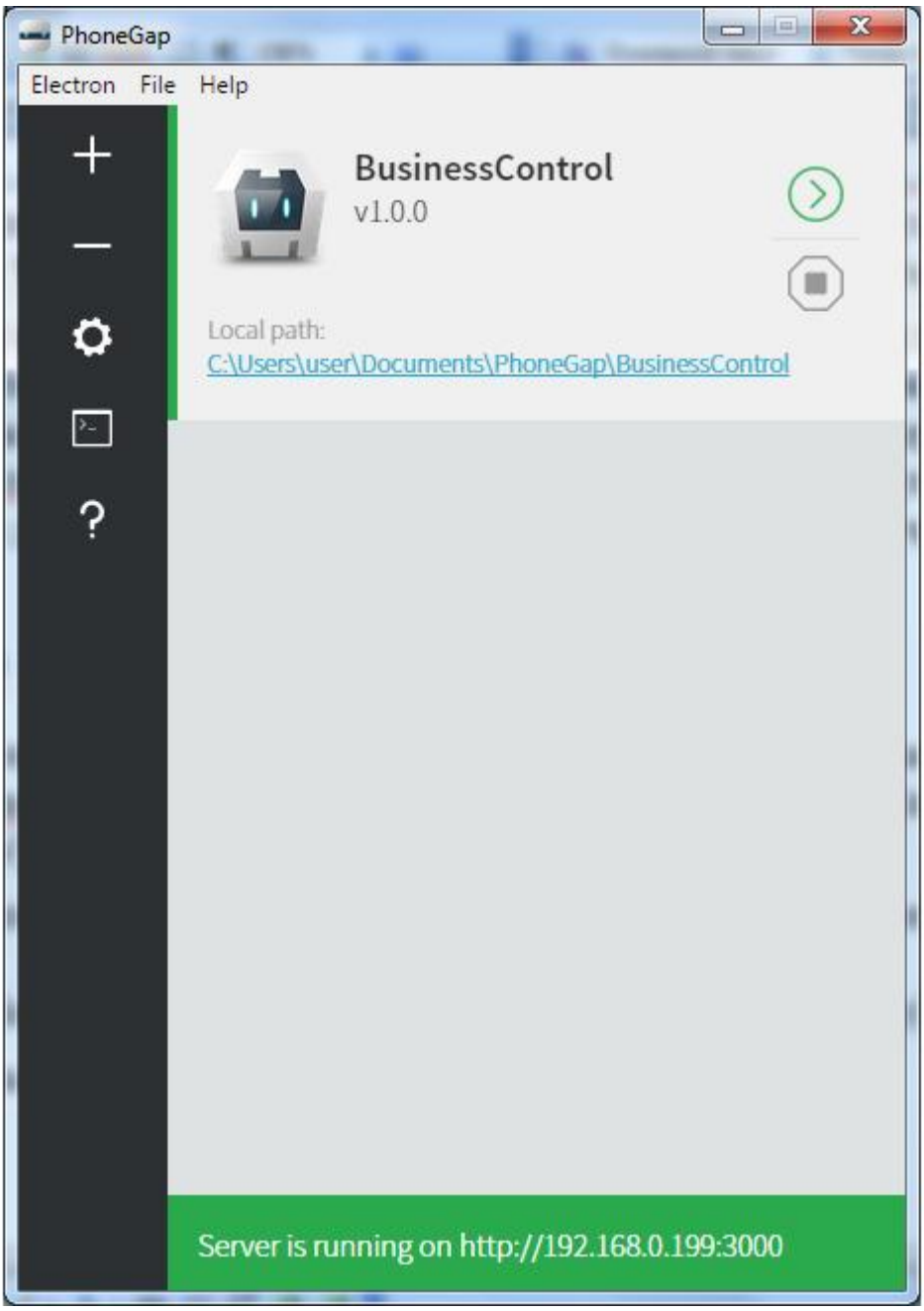
Примером такого приложения может быть приложение, предназначенное для управления деятельностью группы сотрудников компании, каждый из которых пользуется своим мобильным приложением. В этом случае лидер группы должен иметь кросс-платформенное приложение со своей функциональностью, предназначенное для администрирования мобильных приложений членов группы.

PhoneGap/Cordova

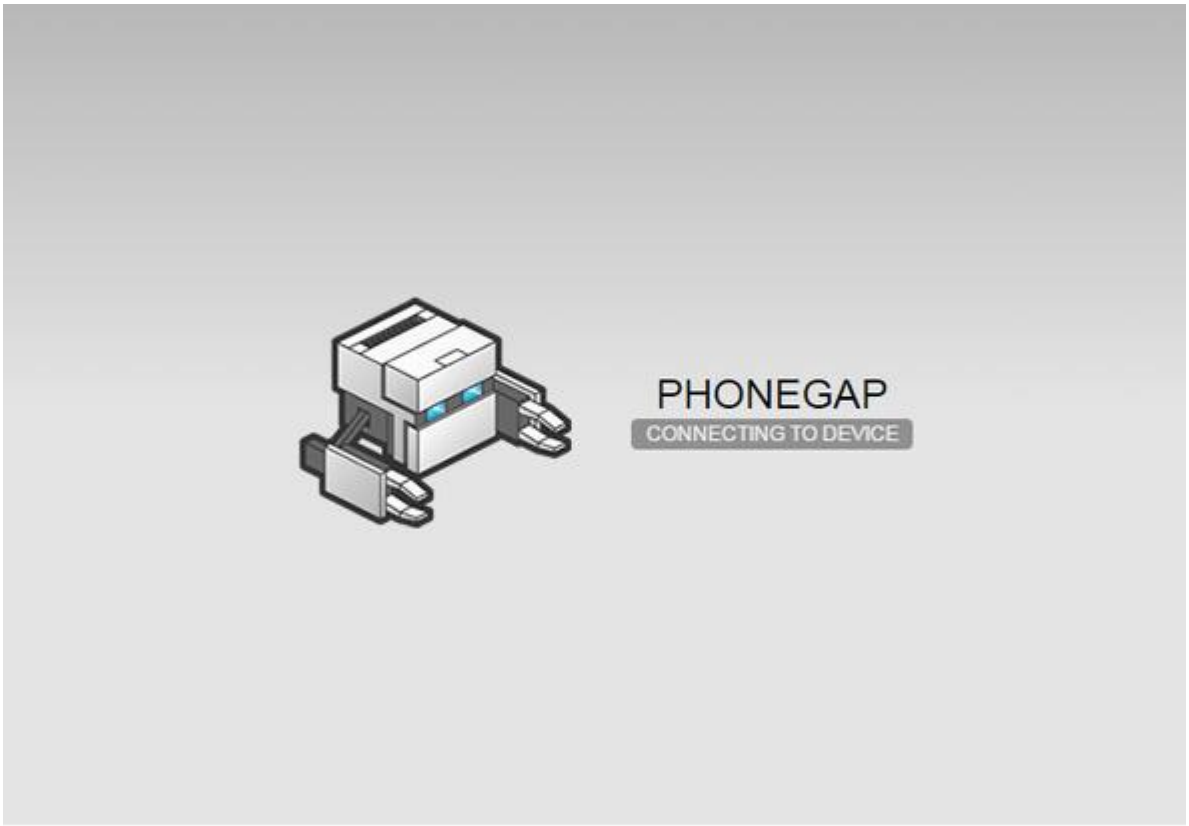
Скачаем и установим настольное PhoneGap приложение, обеспечивающее визуальное создание PhoneGap проекта (http://docs.phonegap.com/getting-started/1-install-phonegap/desktop/).

Скачаем и установим Android PhoneGap приложение, обеспечивающее тестирование PhoneGap проекта (https://play.google.com/store/apps/details?id=com.adobe.phonegap.app).

Запустим настольное PhoneGap приложение и создадим проект:

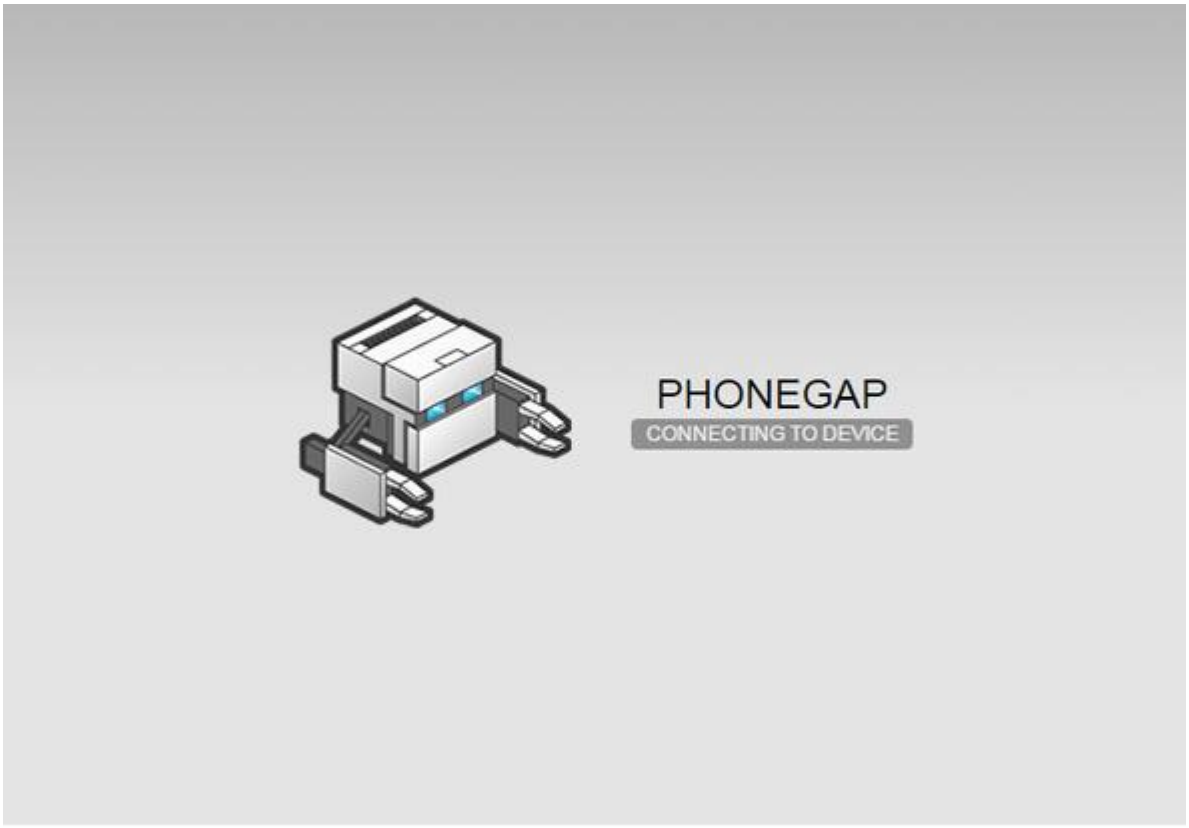


При этом запустится Web сервер, идущий с настольным PhoneGap приложением.
В строке браузера наберем адрес http://192.168.0.199:3000:



Запустим Android PhoneGap приложение, при этом мобильное устройство должно быть подключено к той же сети, что и компьютер:

Наберем адрес сервера и нажмем Connect:



Для запуска приложения на мобильном устройстве USB соединение с компьютером не требуется – соединение осуществляется по сети.

Сгенерированная основа приложения будет содержаться в папке www каталога проекта.

Для сборки проекта можно воспользоваться облачным сервисом Adobe PhoneGap Build (https://build.phonegap.com/).

Adobe PhoneGap является дистрибутивом Apache Cordova.

С Apache Cordova можно не использовать облачный сервис Adobe PhoneGap Build, а создавать и собирать проект приложения с помощью инструмента командной строки CLI.

Для создания проекта приложения с Apache Cordova необходимо установить инструмент командной строки cordova command-line interface (CLI).

Для установки Cordova требуется инструмент npm дистрибутива Node. js, потому нужно установить Node. js (https://nodejs.org/).

Для создания проекта инструмент cordova использует git, потому нужно установить git client (http://git-scm.com/).

Для проверки инсталляции в командной строке наберем npm и git.

Далее в командной строке наберем:

npm install -g cordova

Создадим каталог для проектов и создадим проект приложения:

C:\Users\user> cd C:\Users\user\Cordova

C:\Users\user\Cordova> cordova create test com. example. test Test

Creating a new cordova project.

Структура созданного проекта будет аналогична проекту, созданному с PhoneGap.

Папка www каталога проекта содержит уже готовое Web приложение. Для сборки Android приложения из проекта сначала нужно установить переменные среды для автоматического вызова инструментов Android SDK:

ANDROID_HOME=C:\ <installation location> \android-sdk-windows

Добавить в системную переменную Path:

;%ANDROID_HOME%\platform-tools;%ANDROID_HOME%\tools;

Набрать в командной строке:

cd C:\Users\user\Cordova\test
cordova platform add android
cordova build

Для тестирования Android приложения на устройстве наберите:

cordova run android

В результате отладочная версия APK будет установлена и запущена на мобильном устройстве.

Для сборки подписанного APK файла создайте файл release-signing.properties в каталоге \platforms\android:

storeFile=keystore. jks
storeType=jks
keyAlias=alias-name

Здесь хранилище ключей находится в том же каталоге.

И наберите в командной строке:

cordova build android —release

В результате в папке \platforms\android\build\outputs\apk будет создан файл android-release. apk.

Чтобы изменить значки и заставки приложения, создайте набор изображений в каталоге проекта и укажите в файле config. xml:

<platform name=«android»>
<allow-intent href=«market:*" />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/ldpi/ic_launcher.png» density=«ldpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/mdpi/ic_launcher.png» density=«mdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/hdpi/ic_launcher.png» density=«hdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/xhdpi/ic_launcher.png» density=«xhdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/xxhdpi/ic_launcher.png» density=«xxhdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/xxxhdpi/ic_launcher.png» density=«xxxhdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/land-hdpi/ic_launcher.png» density=«land-hdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/land-ldpi/ic_launcher.png» density=«land-ldpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/land-mdpi/ic_launcher.png» density=«land-mdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/land-xhdpi/ic_launcher.png» density=«land-xhdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/port-hdpi/ic_launcher.png» density=«port-hdpi» />

<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/port-ldpi/ic_launcher.png» density=«port-ldpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/port-mdpi/ic_launcher.png» density=«port-mdpi» />
<icon src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/res/port-xhdpi/ic_launcher.png» density=«port-xhdpi» />
</platform>

После сборки все эти изображения появятся в каталоге \platforms\android\res.

Однако в папках land и port будут еще изображения screen. Удалите их, скопируйте и переименуйте изображения icon в screen, так что будут два одинаковых изображения под именами icon и screen, которые будут служить как значок и заставка приложения.

После сборки проекта новый набор изображений будет включен в файл APK.

Приложения Cordova можно создавать не только в командной строке, но и с помощью среды разработки NetBeans IDE.

В среде NetBeans откроем меню Файл и выберем Создать проект. В открывшемся мастере в разделе HTML5/JavaScript выберем шаблон Приложение Cordova.

После создания проекта приложения, его можно собрать и протестировать на устройстве.

К сожалению, на данный момент среда NetBeans не обеспечивает автоматическое создание подписанного APK файла.

При разработке с Cordova нужно учитывать, что приложение Cordova это приложение SPA (Single Page Application), то есть приложение с одной страницей index.html, весь контент которой обеспечивается кодом JavaScript и CSS. Поэтому разработка Cordova приложения ведется на основе какого-либо JavaScript фреймворка.

Конвертация HTML5 Web сайта в приложение Cordova

Преобразовать HTML5 сайт в Android приложение очень просто.

Для примера скачаем шаблон HTML5 сайта (https://w3layouts.com/free-responsive-html5-css3-website-templates/) и скопируем все файлы шаблона в папку www каталога предварительно созданного Cordova проекта с помощью команды cordova create.

Далее добавим Android платформу:

cordova platform add android

И соберем Android приложение:

cordova build

При тестировании с помощью команды cordova run android на мобильном устройстве увидим тот же шаблон, что и в Web браузере.

Пример кросс-платформенного Backend приложения

Приложение представляет собой панель администратора, руководящего группами торговых представителей.

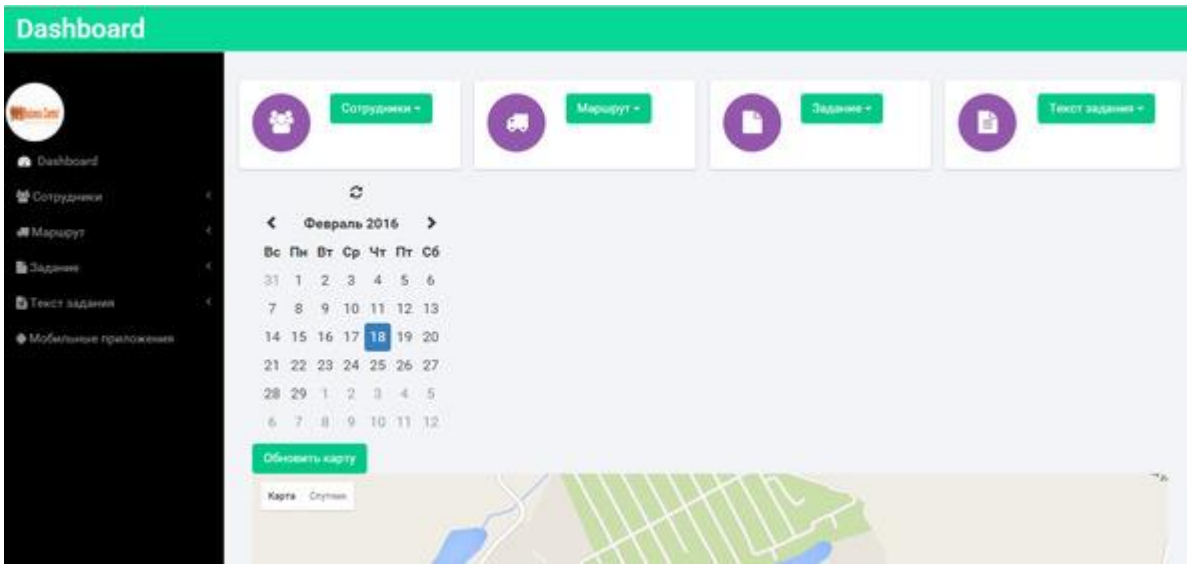
Приложение позволяет:

– Просмотреть список сотрудников,
– Сформировать из них группы,
– Выбрать маршрут движения сотрудников,
– Просмотреть ранее созданные маршруты,
– Составить текст задания для сотрудников,
– Просмотреть ранее созданные тексты заданий,

– Скомпоновать задание для сотрудников, включив в него группу сотрудников для выполнения задания, маршруты выполнения, описание задания,

– Посмотреть ранее созданные задания.

Выполненное задание содержит фактический трекинг движения сотрудников, отчеты сотрудников о выполнении задания.



Начальный экран приложения представляет собой общую панель, включающую в себя все опции.

```
<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>
```

```
</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href="index.html»> Dashboard </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
<a href="users-list.html»> Общий список </a>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
```

```
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>
```

```
</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>
<div class=«col-md-3 widget widget1»>
<div class=«r3_counter_box»>
<i class=«pull-left fa fa-users icon-rounded»> </i>



<div class=«dropdown»>
<button class=«btn btn-primary dropdown-toggle» type=«button» data-toggle=«dropdown»>
Сотрудники
<span class=«caret»> </span> </button>
<ul class=«dropdown-menu»>
<li> <a href="groups-list.html»> Группы </a> </li>
<li> <a href="users-list.html»> Общий список </a> </li>
</ul>
</div>
</div>
</div>
<div class=«col-md-3 widget widget1»>
<div class=«r3_counter_box»>
<i class=«pull-left fa fa-truck icon-rounded»> </i>
<div class=«dropdown»>
<button class=«btn btn-primary dropdown-toggle» type=«button» data-toggle=«dropdown»>
Маршрут
<span class=«caret»> </span> </button>
<ul class=«dropdown-menu»>
<li> <a href="route.html»> Выбрать маршрут </a> </li>
<li> <a href="new-routes.html»> Новые маршруты </a> </li>
<li> <a href="old-routes.html»> Архив маршрутов </a> </li>
</ul>
</div>
</div>
</div>
<div class=«col-md-3 widget widget1»>
<div class=«r3_counter_box»>
<i class=«pull-left fa fa-file icon-rounded»> </i>
<div class=«dropdown»>
<button class=«btn btn-primary dropdown-toggle» type=«button» data-toggle=«dropdown»>
Задание
<span class=«caret»> </span> </button>
<ul class=«dropdown-menu»>
<li> <a href="task.html»> Создать задание </a> </li>
<li> <a href="new-tasks.html»> Новые задания </a> </li>
<li> <a href="old-tasks.html»> Архив заданий </a> </li>
</ul>
</div>
</div>
```

```
</div>
<div class=«col-md-3 widget»>
<div class=«r3_counter_box»>
<i class=«pull-left fa fa-file-text icon-rounded»> </i>
<div class=«dropdown»>
<button class=«btn btn-primary dropdown-toggle» type=«button» data-toggle=«dropdown»> Текст задания
<span class=«caret»> </span> </button>
<ul class=«dropdown-menu»>
<li> <a href="desc.html»> Создать описание </a> </li>
<li> <a href="new-descs.html»> Новые описания </a> </li>
<li> <a href="old-descs.html»> Архив описаний </a> </li>
</ul>
</div>
</div>
</div>
<div class=«clearfix»> </div>
</div>



<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i> </div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div class=’row’>
<div class=’col-sm-4»>
```

```
<button type=’button’ class=’btn btn-primary btn-mapUpdate’> Обновить карту </button>
</div>
</div>
<div id=«map» style=«height:500px;"> </div>
<script>
var map;



function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 6
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}
}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>
```

```
</div>
<div class=«clearfix»> </div>
</div>



<script>
var url = 'http://buisness-control.appspot.com/tracks/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});



$ (".btn-mapUpdate» ).click (function () {
$.ajax ({
type: «GET»,
url: url,
```

```
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
});
</script>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```

Начальная страница, помимо разметки, содержит Javascript код инициализации Google карты и код получения трекинга движения сотрудников, выполняющих задания, с отображением трекинга на карте.

Страница «Общий список сотрудников» позволяет получить список сотрудников и отредактировать их персональные данные.

```
<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>

</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Общий список сотрудников </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
```

<ul class=«nav» id=«side-menu»>

 <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard

 <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Группы

 Общий список

<! – /.nav-second-level – >

 <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Выбрать маршрут

 Маршруты <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Создать задание

 Задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

```
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<! – Modal – >
<style type=«text/css»>
input {
outline: 1px solid lightgreen;
}
</style>
<div class=«modal fade» id=«profile» role=«dialog»>
<div class=«modal-dialog»>



<! – Modal content – >
<div class=«modal-content»>
```

```
<div class=«modal-header»>
<button type=«button» class=«close» data-dismiss=«modal»> &times; </button>
Профиль
</div>
<div class=«modal-body»>
<form role=«form»>



<div class=«form-group» style=«display: none;»>
<input type=«email» class=«form-control» id=«modal-account»>
</div>



<div class=«form-group»>
<label for=«modal-firstName»> Имя: </label>
<input type=«text» class=«form-control» id=«modal-firstName»>
</div>
<div class=«form-group»>
<label for=«modal-secondName»> Фамилия: </label>
<input type=«text» class=«form-control» id=«modal-secondName»>
</div>
<div class=«form-group»>
<label for=«modal-phone»> Телефон: </label>
<input type=«text» class=«form-control» id=«modal-phone»>
</div>
<div class=«form-group»>
<label for=«modal-position»> Должность: </label>
<input type=«text» class=«form-control» id=«modal-position»>
</div>
<div class=«form-group»>
<label for=«modal-group»> Группа: </label>
<input type=«text» class=«form-control» id=«modal-group»>
</div>



<button id=«modal-submit» class=«btn btn-primary»> Отправить </button>
</form>
</div>
<div class=«modal-footer»>
<button type=«button» class=«btn btn-default» data-dismiss=«modal»> Close </button>
</div>
</div>



</div>
</div>



<script type=«text/javascript»>
$ (document ).ready (function () {
```

```
var url = 'http://buisness-control.appspot.com/userslist/?callback=?';

$.ajax ({
type: «GET»,
url: url,
async: true,
jsonpCallback: ’getUsers’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.users, function (key, val) {
items. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val.account + "</a> <div id=«» + val.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class=’collapse’> <div class=’row’> <div class=’col-sm-6»> Имя </div> <div class='col-sm-6'>"+val.firstName+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Фамилия </div> <div class='col-sm-6'>"+val.secondName+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Телефон </div> <div class='col-sm-6'>"+val.phone+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Должность </div> <div class=’col-sm-6»>"+val. position+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Группа </div> <div class='col-sm-6'>"+val.group+"</div> </div> <button id='button"+val.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») +" type=’button’ class=’btn btn-primary btn-edit-profile’ data-firstName='"+val.firstName+«» data-secondName='"+val.secondName+«» data-phone='"+val.phone+«» data-position=«»+val. position+«» data-group='"+val.group+«» data-account='"+val.account+«»> Редактировать </button> </div>»);
});

$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo (".col_3»);

$ (".btn-edit-profile» ).click (function () {

var account = $(this).attr (’data-account’);
$("#modal-account").attr («value», account);

var firstName = $(this).attr (’data-firstName’);
$("#modal-firstName").attr («value», firstName);
```

```
var secondName = $(this).attr (’data-secondName’);
$("#modal-secondName").attr («value», secondName);



var phone = $(this).attr (’data-phone’);
$("#modal-phone").attr («value», phone);



var position = $(this).attr (’data-position’);
$("#modal-position").attr («value», position);



var group = $(this).attr (’data-group’);
$("#modal-group").attr («value», group);



$("#profile").modal ();
});



}
});



$ («#modal-submit» ).click (function () {
var account = $("#modal-account").val ();
var firstName = $("#modal-firstName").val ();
var secondName = $("#modal-secondName").val ();
var phone = $("#modal-phone").val ();
var position = $("#modal-position").val ();
var group = $("#modal-group").val ();



var url = 'http://buisness-control.appspot.com/putuser/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
«account»: account,
«firstName»: firstName,
«secondName»: secondName,
«phone»: phone,
«position»: position,
«group»: group
},
```

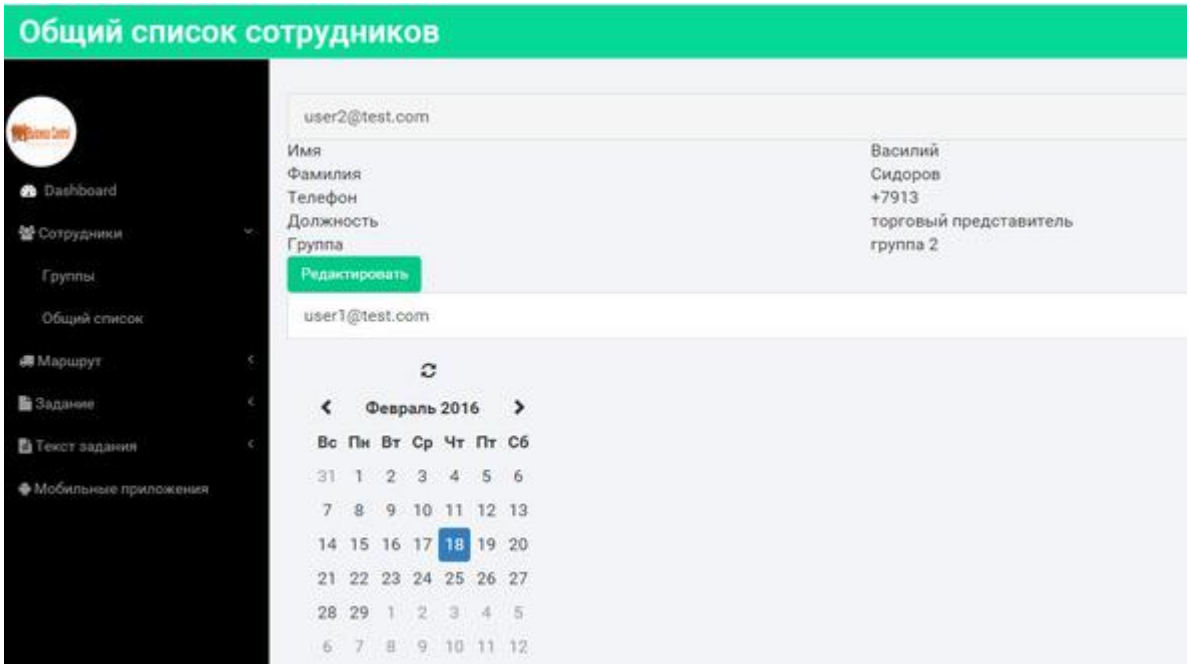
```
async: false,
jsonpCallback: ’putUser’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$("#profile").modal («hide»);
location.reload ();
}
});



});



});



</script>



<div class=«clearfix»> </div>
</div>



<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i>
</div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>



</div>
<div class=«clearfix»> </div>
```

```
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div class=’row’>
<div class=’col-sm-4»>
<button type=’button’ class=’btn btn-primary btn-mapUpdate’> Обновить карту </button>
</div>
</div>
<div id=«map» style=«height:500px;"> </div>



<script>
var map;



function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 6
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}
}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}
```

```
</script>

<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>

</div>
<div class=«clearfix»> </div>
</div>

<script>
var url = 'http://buisness-control.appspot.com/tracks/?callback=?';

$.ajax ({
type: «GET»,
url: url,
data: {

},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});

});
```

```
}
});



$ (".btn-mapUpdate» ).click (function () {
$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
});
</script>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
```

<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>



Страница «Группы сотрудников» позволяет получить все тот же список сотрудников, только с сортировкой по группам.

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >

```
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>

</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«»> Группы сотрудников </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
<a href="users-list.html»> Общий список </a>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
```

```
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>
```

```
</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<! – Modal – >
<style type=«text/css»>
input {
outline: 1px solid lightgreen;
}
</style>
<div class=«modal fade» id=«profile» role=«dialog»>
<div class=«modal-dialog»>



<! – Modal content – >
<div class=«modal-content»>
<div class=«modal-header»>
<button type=«button» class=«close» data-dismiss=«modal»> &times; </button>
Профиль
</div>
<div class=«modal-body»>
<form role=«form»>



<div class=«form-group» style=«display: none;»>
<input type=«email» class=«form-control» id=«modal-account»>
</div>



<div class=«form-group»>
<label for=«modal-firstName»> Имя: </label>
<input type=«text» class=«form-control» id=«modal-firstName»>
</div>
<div class=«form-group»>
<label for=«modal-secondName»> Фамилия: </label>
<input type=«text» class=«form-control» id=«modal-secondName»>
</div>
<div class=«form-group»>
```

<label for=«modal-phone»> Телефон: </label>
<input type=«text» class=«form-control» id=«modal-phone»>

<label for=«modal-position»> Должность: </label>
<input type=«text» class=«form-control» id=«modal-position»>

<label for=«modal-group»> Группа: </label>
<input type=«text» class=«form-control» id=«modal-group»>

<button id=«modal-submit» class=«btn btn-primary»> Отправить </button>
</form>

<button type=«button» class=«btn btn-default» data-dismiss=«modal»> Close </button>

<script type=«text/javascript»>
$ (document).ready (function () {

var url = 'http://buisness-control.appspot.com/groupslist/?callback=?';

$.ajax ({
type: «GET»,
url: url,
async: true,
jsonpCallback: ’getGroups’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.groups, function (key, val) {

var arrayName=Object.keys(json.groups[key]).toString ();

```
var subitems = [];

$.each (val [arrayName], function (subkey, subval) {
subitems. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + subval.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + subval.account + "</a> <div id=«» + subval.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class=’collapse’> <div class=’row’> <div class=’col-sm-6»> Имя </div> <div class='col-sm-6'>"+subval.firstName+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Фамилия </div> <div class='col-sm-6'>"+subval.secondName+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Телефон </div> <div class='col-sm-6'>"+subval.phone+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Должность </div> <div class=’col-sm-6»>"+subval. position+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Группа </div> <div class='col-sm-6'>"+subval.group+"</div> </div> <button id='button"+subval.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») +" type=’button’ class=’btn btn-primary btn-edit-profile’ data-firstName='"+subval.firstName+«» data-secondName='"+subval.secondName+«» data-phone='"+subval.phone+«» data-position=«»+subval. position+«» data-group='"+subval.group+«» data-account='"+subval.account+«»> Редактировать </button> </div>»);
});

items. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + arrayName.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + Object.keys(json.groups [key]) + "</a> <button type=’button’ class=’btn btn-danger btn-del-group’ data-name=«»+arrayName+«»> Удалить группу </button> <div id=«» + arrayName.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+subitems.join (»») +"</div>»);
});

$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo (".col_3»);

$ (".btn-del-group» ).click (function () {
var name = $(this).attr (’data-name’);

var url = 'http://buisness-control.appspot.com/deletegroup/?callback=?';

$.ajax ({
type: «GET»,
url: url,
data: {
name: name
},
async: false,
```

```
jsonpCallback: ’deleteGroup’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});



$ (".btn-edit-profile» ).click (function () {



var account = $(this).attr (’data-account’);
$("#modal-account").attr («value», account);



var firstName = $(this).attr (’data-firstName’);
$("#modal-firstName").attr («value», firstName);



var secondName = $(this).attr (’data-secondName’);
$("#modal-secondName").attr («value», secondName);



var phone = $(this).attr (’data-phone’);
$("#modal-phone").attr («value», phone);



var position = $(this).attr (’data-position’);
$("#modal-position").attr («value», position);



var group = $(this).attr (’data-group’);
$("#modal-group").attr («value», group);



$("#profile").modal ();
});



}
});
```

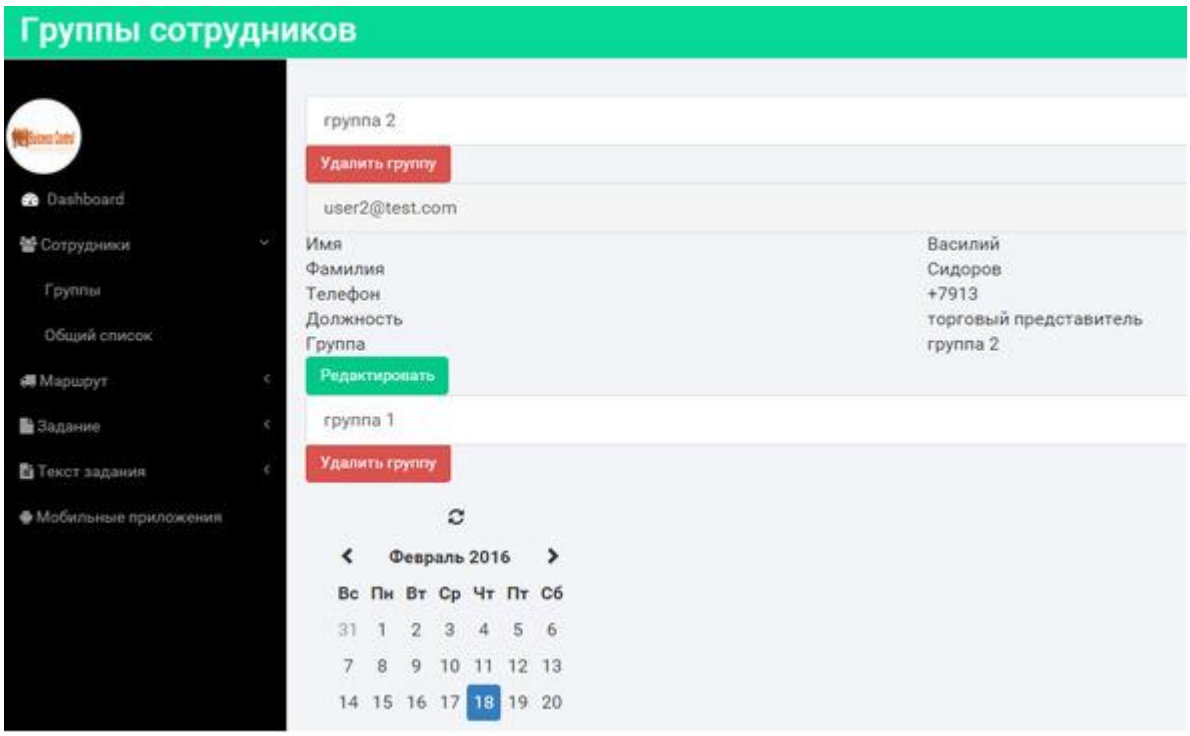
```
$ («#modal-submit» ).click (function () {
var account = $("#modal-account").val ();
var firstName = $("#modal-firstName").val ();
var secondName = $("#modal-secondName").val ();
var phone = $("#modal-phone").val ();
var position = $("#modal-position").val ();
var group = $("#modal-group").val ();



var url = 'http://buisness-control.appspot.com/putuser/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
«account»: account,
«firstName»: firstName,
«secondName»: secondName,
«phone»: phone,
«position»: position,
«group»: group
},
async: false,
jsonpCallback: ’putUser’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$("#profile").modal («hide»);
location.reload ();
}
});



});



});



</script>



<div class=«clearfix»> </div>
</div>
```

```
<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i>
</div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div class=’row’>
<div class=’col-sm-4»>
<button type=’button’ class=’btn btn-primary btn-mapUpdate’> Обновить карту </button>
</div>
</div>



<div id=«map» style=«height:500px;"> </div>



<script>
var map;



function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 6
});
var infoWindow = new google.maps.InfoWindow ({map: map});
```

```
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}
}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



</div>
<div class=«clearfix»> </div>
</div>



<script>
var url = 'http://buisness-control.appspot.com/tracks/?callback=?';



$.ajax ({
type: «GET»,
url: url,
```

```
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});



$ (".btn-mapUpdate» ).click (function () {
$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
```

```
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
});
</script>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



Страница «Выбор Маршрута» позволяет двойным нажатием правой кнопки мышки поставить на карте маркер, а затем сохранить все созданные маркеры в виде маршрута. Двойное нажатие левой кнопки мышки убирает ранее созданный маркер.

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>

```
</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Выбор Маршрута </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
<a href="users-list.html»> Общий список </a>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
```

```
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>
```

```
</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>

<button id=«btn-save-route» type=«button» class=«btn btn-primary»> Сохранить маршрут </button>

<div class=«clearfix»> </div>
</div>

<div class=«span_11»>
<div class=«col-md-12»>
<div id=«map» style=«height:500px;"> </div>

<script>
var markers = [];

function initMap () {
var map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});

if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
```

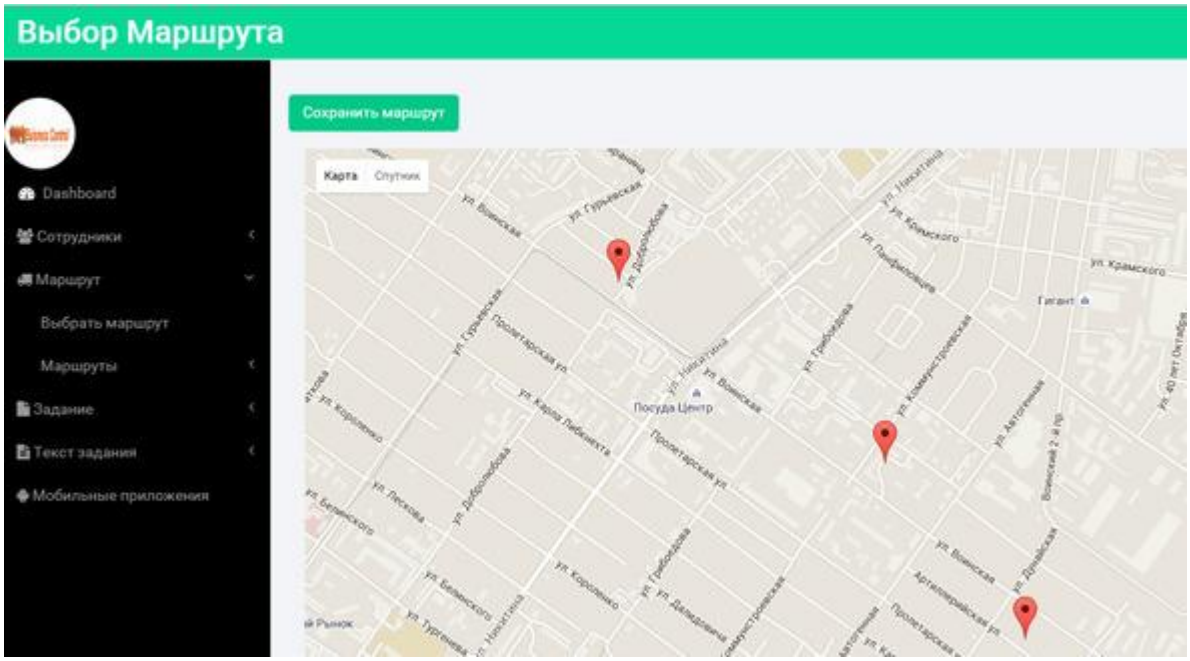
```
}


map.addListener (’dblclick’, function (e) {
var marker = new google.maps.Marker ({
position: e.latLng,
map: map
});


markers. push (marker);


marker.addListener (’rightclick’, function (e) {
marker.setMap (null);
$.each (markers, function (i) {
if (markers [i] === marker) {
markers. splice (i,1);
return false;
}
});
});


map.setZoom (14);
});


}


function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}


$ («#btn-save-route» ).click (function () {


if (markers. length!=0) {


var arrJ = new Array ();
```

```
$.each (markers, function (i) {
var latlngJ = markers[i].getPosition().toJSON ();
arrJ. push (latlngJ);
});



var url = 'http://buisness-control.appspot.com/putroute/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
json:JSON.stringify (arrJ),
state:’new’,
date: (new Date()).toLocaleString ()
},
async: false,
jsonpCallback: ’putRoute’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



}



});



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



</div>
<div class=«clearfix»> </div>
</div>
```

 Copyright © 2015 TM SoftStudio

<! – /#page-wrapper – >

<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>



Страница «Новые Маршруты» позволяет просмотреть ранее созданные маршруты, имеющие статус «new».

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->

```
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>

</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Новые Маршруты </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
<a href="users-list.html»> Общий список </a>
</li>
</ul>
<! – /.nav-second-level – >
</li>
```

```
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
```

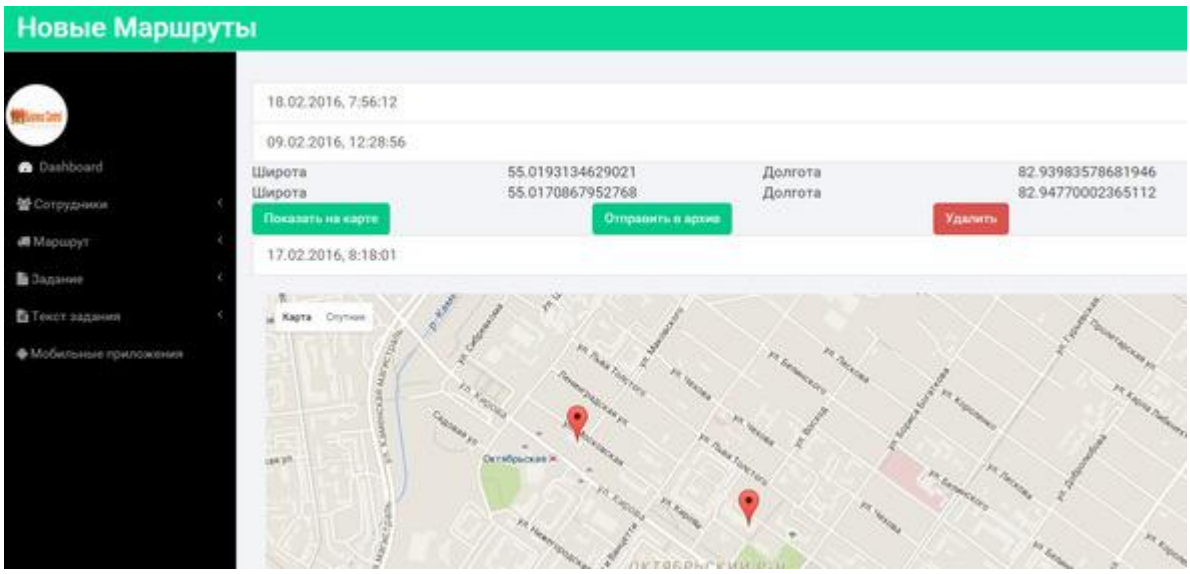
```
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div id=«map» style=«height:500px;"> </div>



<script>
var map;
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
```

```
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}



}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



<script>
var url = 'http://buisness-control.appspot.com/routes/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’
},
async: true,
jsonpCallback: ’getNewRoutes’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.routes, function (key, val) {
```

```
var subitems = [];

$.each (val.route, function (subkey, subval) {
subitems. push (» <div class=’row’> <div class=’col-sm-3»> Широта </div> <div class='col-sm-3'>"+subval.lat+"</div> <div class=’col-sm-3»> Долгота </div> <div class=’col-sm-3»>"+subval. lng+"</div> </div>»);
});

items. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val. date + "</a> <div id=«» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+subitems.join (»») +"<div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-map’ data-date=«»+val. date+«» data-route='"+JSON.stringify(val.route) +«»> Показать на карте </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-state’ data-date=«»+val. date+«»> Отправить в архив </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-danger btn-del’ data-date=«»+val. date+«»> Удалить </button> </div> </div> </div>»);

});

$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo (".col_3»);

$ (".btn-map» ).click (function () {
var route = jQuery.parseJSON($(this).attr (’data-route’));
var date = $(this).attr (’data-date’);

$.each (route, function (key, val) {

var latLng =new google.maps.LatLng(val.lat, val. lng);

var marker = new google.maps.Marker ({
position: latLng,
title: date,
map: map
});
```

```
map.setCenter (latLng);



});
});



$ (".btn-state» ).click (function () {
var date = $(this).attr (’data-date’);



var url = 'http://buisness-control.appspot.com/putroutestate/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’old’,
date: date
},
async: false,
jsonpCallback: ’putRouteState’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});



$ (".btn-del» ).click (function () {
var date = $(this).attr (’data-date’);



var url = 'http://buisness-control.appspot.com/deleteroute/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
```

```
date: date
},
async: false,
jsonpCallback: ’deleteRoute’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});



}
});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



Страница «Архив Маршрутов» позволяет просмотреть ранее созданные маршруты, имеющие статус «old».

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>

</head>
<body>

```
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Архив Маршрутов </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
<a href="users-list.html»> Общий список </a>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
```

```
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
```

```
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div id=«map» style=«height:500px;"> </div>



<script>
var map;
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}



}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
```

```
}


</script>


<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>


<script>
var url = 'http://buisness-control.appspot.com/routes/?callback=?';


$.ajax ({
type: «GET»,
url: url,
data: {
state:’old’
},
async: true,
jsonpCallback: ’getOldRoutes’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.routes, function (key, val) {


var subitems = [];


$.each (val.route, function (subkey, subval) {
subitems. push (» <div class=’row’> <div class=’col-sm-3»> Широта </div> <div class='col-sm-3'>"+subval.lat+"</div> <div class=’col-sm-3»> Долгота </div> <div class=’col-sm-3»>"+subval. lng+"</div> </div>»);
});


items. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val. date + "</a> <div id=«» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+subitems.join (»») +"<div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-map’ data-date=«»+val. date+«» data-route='"+JSON.stringify(val.route) +«»> Показать на карте </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-state’ data-date=«»+val. date+«»> Новый маршрут </button> </div> <div class=’col-sm-4»> <button
```

```
type=’button’ class=’btn btn-danger btn-del’ data-date=«»+val. date+«»> Удалить </button> </div> </div> </div>»);


});


$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo (".col_3»);


$ (".btn-map» ).click (function () {
var route = jQuery.parseJSON($(this).attr (’data-route’));
var date = $(this).attr (’data-date’);


$.each (route, function (key, val) {


var latLng =new google.maps.LatLng(val.lat, val. lng);


var marker = new google.maps.Marker ({
position: latLng,
title: date,
map: map
});


map.setCenter (latLng);


});
});


$ (".btn-state» ).click (function () {
var date = $(this).attr (’data-date’);


var url = 'http://buisness-control.appspot.com/putroutestate/?callback=?';
```

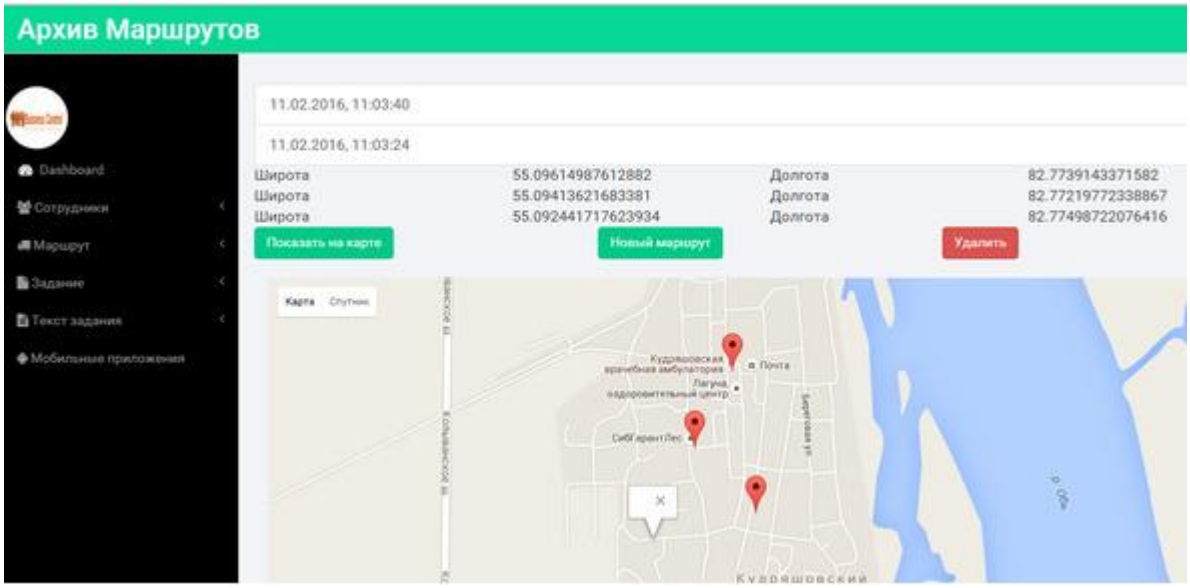
```
$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’,
date: date
},
async: false,
jsonpCallback: ’putRouteState’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});



$ (".btn-del» ).click (function () {
var date = $(this).attr (’data-date’);



var url = 'http://buisness-control.appspot.com/deleteroute/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
date: date
},
async: false,
jsonpCallback: ’deleteRoute’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});



}
});
```

</script>

 Copyright © 2015 TM SoftStudio

<! – /#page-wrapper – >

<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>



С помощью страницы «Описание Задания» можно создать и сохранить текстовое описание задания.

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />

```
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link  href='//fonts.googleapis.com/css?  family=Roboto:400,100,300,500,700,900»  rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>
<! – Editor – >
<script src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///cdn.tinymce.com/4/tinymce. min. js»> </script>
<script>tinymce.init ({selector: «#editor’}); </script>



</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Описание задания </a>
</div>
<! – /.navbar-header – >



<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
```

 <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Группы

 Общий список

<! – /.nav-second-level – >

 <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Выбрать маршрут

 Маршруты <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Создать задание

 Задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>


```
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<textarea id=«editor»> Текст… </textarea>
<button id=«btn-editor» type=«button» class=«btn btn-primary»> Сохранить </button>



<script>
$ («#btn-editor» ).click (function () {
var desc = tinyMCE.get('editor').getContent ({format: 'text'}).replace (/'|"/g,»&amp;quot;»);



var url = 'http://buisness-control.appspot.com/putdesc/?callback=?';



$.ajax ({
type: «GET»,
url: url,
```

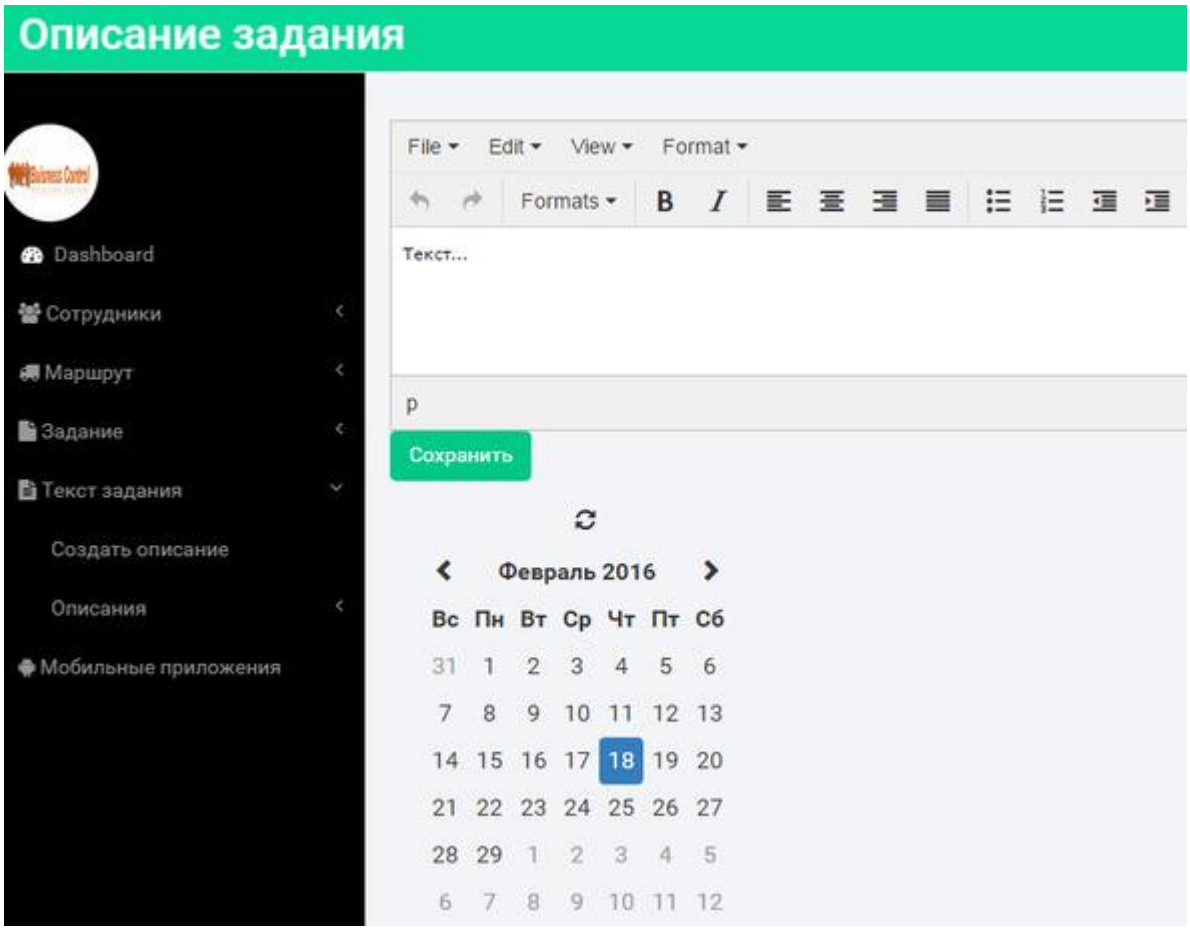
```
data: {
desc: desc,
state:’new’,
date: (new Date()).toLocaleString ()
},
async: false,
jsonpCallback: ’putDesc’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});
</script>



<div class=«clearfix»> </div>
</div>



<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i>
</div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>



</div>
<div class=«clearfix»> </div>
</div>
```

```
<div class=«span_11»>
<div class=«col-md-12»>
<div class=’row’>
<div class=’col-sm-4»>
<button type=’button’ class=’btn btn-primary btn-mapUpdate’> Обновить карту </button>
</div>
</div>



<div id=«map» style=«height:500px;"> </div>



<script>
var map;



function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 6
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}
}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}
```

```
</script>

<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>

</div>
<div class=«clearfix»> </div>
</div>

<script>
var url = 'http://buisness-control.appspot.com/tracks/?callback=?';

$.ajax ({
type: «GET»,
url: url,
data: {

},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});

});
```

```
}
});



$ (".btn-mapUpdate» ).click (function () {
$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
});
</script>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
```

<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>



Страница «Новые Описания» позволяет просмотреть ранее созданные текстовые описания заданий, имеющие статус «new».

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->

```
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>
<! – Editor – >
<script src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///cdn.tinymce.com/4/tinymce. min. js»> </script>
<script>tinymce.init ({selector: «#editor’}); </script>



</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Новые Описания </a>
</div>
<! – /.navbar-header – >



<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
```

 Общий список

<! – /.nav-second-level – >

 <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Выбрать маршрут

 Маршруты <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Создать задание

 Задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Создать описание

 Описания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>


```
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<textarea id=«editor»> Текст… </textarea>
<button id=«btn-editor» type=«button» class=«btn btn-primary»> Сохранить </button>



<script>
$ («#btn-editor» ).click (function () {
var desc = tinyMCE.get('editor').getContent ({format: 'text'}).replace (/'|"/g,»&amp;quot;»);



var url = 'http://buisness-control.appspot.com/putdesc/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
desc: desc,
state:’new’,
date: (new Date()).toLocaleString ()
},
async: false,
```

```
jsonpCallback: ’putDesc’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});
</script>



<div class=«clearfix»> </div>
</div>



<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i>
</div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div class=’row’>
<div class=’col-sm-4»>
<button type=’button’ class=’btn btn-primary btn-mapUpdate’> Обновить карту </button>
```

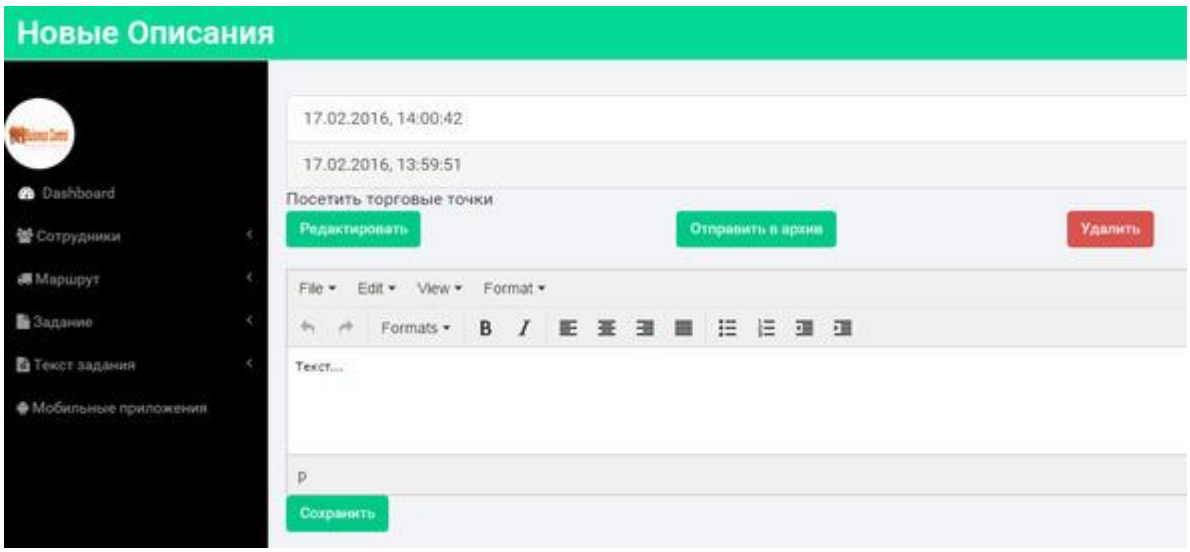
```
</div>
</div>
<div id=«map» style=«height:500px;"> </div>



<script>
var map;
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}



}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>
```

```
<script>
var url = 'http://buisness-control.appspot.com/descs/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’
},
async: true,
jsonpCallback: ’getNewDescs’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.descs, function (key, val) {



items. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val. date + "</a> <div id=«» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+val.desc+"<div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-ed’ data-date=«»+val. date+«» data-desc='"+val.desc+«»> Редактировать </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-state’ data-date=«»+val. date+«»> Отправить в архив </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-danger btn-del’ data-date=«»+val. date+«»> Удалить </button> </div> </div> </div>»);



});



$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).prependTo (".col_3»);



$ (".btn-ed» ).click (function () {
var desc = $(this).attr (’data-desc’);
tinyMCE.get('editor').setContent (desc);
});



$ (".btn-state» ).click (function () {
var date = $(this).attr (’data-date’);
```

```
var url = 'http://buisness-control.appspot.com/putdescstate/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’old’,
date: date
},
async: false,
jsonpCallback: ’putDescState’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});



$ (".btn-del» ).click (function () {
var date = $(this).attr (’data-date’);



var url = 'http://buisness-control.appspot.com/deletedesc/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
date: date
},
async: false,
jsonpCallback: ’deleteDesc’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});
```

```
}
});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<script>
var url = 'http://buisness-control.appspot.com/tracks/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
```

```
$ (".btn-mapUpdate» ).click (function () {
$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
});
</script>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



Страница «Архив Описаний» позволяет просмотреть ранее созданные текстовые описания заданий, имеющие статус «old».

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>
<! – Editor – >
<script src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///cdn.tinymce.com/4/tinymce. min. js»> </script>
<script>tinymce.init ({selector: «#editor’}); </script>

```
</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Архив Описаний </a>
</div>
<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>
<img src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/img/logo.png» />
<ul class=«nav» id=«side-menu»>
<li>
<a href="index.html»> <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard </a>
</li>
<li>
<a href=«#»> <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="groups-list.html»> Группы </a>
</li>
<li>
<a href="users-list.html»> Общий список </a>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
```

```
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>
```

```
</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<textarea id=«editor»> Текст… </textarea>
<button id=«btn-editor» type=«button» class=«btn btn-primary»> Сохранить </button>



<script>
$ («#btn-editor» ).click (function () {
var desc = tinyMCE.get('editor').getContent ({format: 'text'}).replace (/'|"/g,»&amp;quot;»);



var url = 'http://buisness-control.appspot.com/putdesc/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
desc: desc,
state:’new’,
date: (new Date()).toLocaleString ()
},
async: false,
jsonpCallback: ’putDesc’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});
</script>



<div class=«clearfix»> </div>
</div>
```

```
<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i>
</div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div class=’row’>
<div class=’col-sm-4»>
<button type=’button’ class=’btn btn-primary btn-mapUpdate’> Обновить карту </button>
</div>
</div>
<div id=«map» style=«height:500px;"> </div>



<script>
var map;
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
```

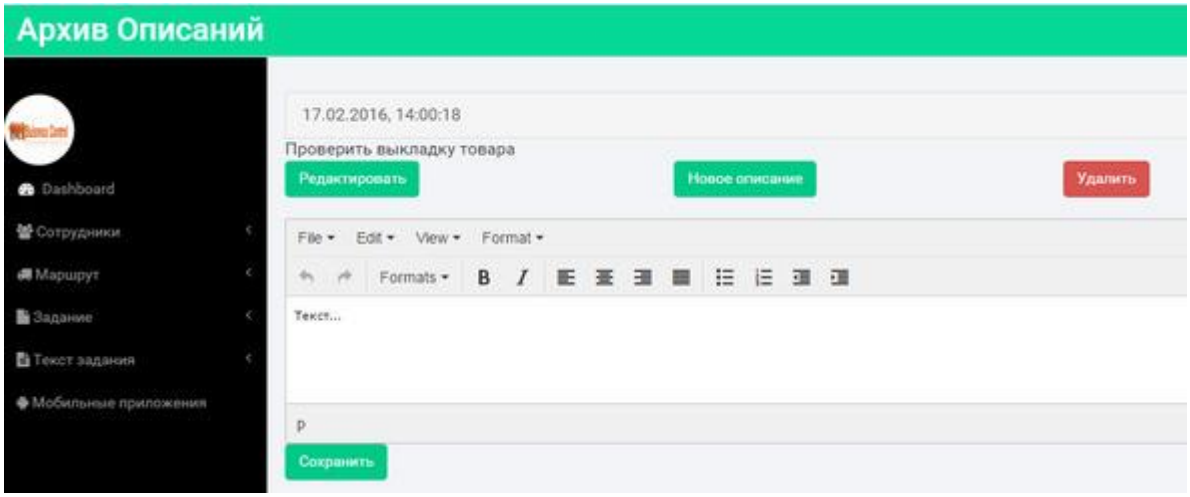
```
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}



}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



<script>
var url = 'http://buisness-control.appspot.com/descs/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’old’
},
async: true,
jsonpCallback: ’getNewDescs’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
```

```
var items = [];
$.each (json.descs, function (key, val) {

items. push (» <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val. date + "</a> <div id=«» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+val.desc+"<div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-ed’ data-date=«»+val. date+«» data-desc='"+val.desc+«»> Редактировать </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-state’ data-date=«»+val. date+«»> Новое описание </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-danger btn-del’ data-date=«»+val. date+«»> Удалить </button> </div> </div> </div>»);

});

$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).prependTo (".col_3»);

$ (".btn-ed» ).click (function () {
var desc = $(this).attr (’data-desc’);
tinyMCE.get('editor').setContent (desc);
});

$ (".btn-state» ).click (function () {
var date = $(this).attr (’data-date’);

var url = 'http://buisness-control.appspot.com/putdescstate/?callback=?';

$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’,
date: date
},
async: false,
jsonpCallback: ’putDescState’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
```

```
location.reload ();
}
});



});



$ (".btn-del» ).click (function () {
var date = $(this).attr (’data-date’);



var url = 'http://buisness-control.appspot.com/deletedesc/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
date: date
},
async: false,
jsonpCallback: ’deleteDesc’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});
}
});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<script>
var url = 'http://buisness-control.appspot.com/tracks/?callback=?';



$.ajax ({
```

```
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});



$ (".btn-mapUpdate» ).click (function () {
$.ajax ({
type: «GET»,
url: url,
data: {



},
async: true,
jsonpCallback: ’getTracks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$.each (json.tracks, function (key, val) {
var name = val.name;
var track = val.track;
```

```
$.each (track, function (key, val) {
var account=val.account;
var latLng =new google.maps.LatLng(val.lat, val. lng);
var marker = new google.maps.Marker ({
position: latLng,
title: name+" "+account,
map: map
});
map.setCenter (latLng);
map.setZoom (15);
});



});



}
});
});
</script>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



Страница «Новое Задание» позволяет скопмоновать задание из текстового описания и маршрута, прикрепив к нему сотрудников.

```
<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>


</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«»> Новое задание </a>
</div>
<! – /.navbar-header – >


<div class=«navbar-default sidebar» role=«navigation»>
```

<div class=«sidebar-nav navbar-collapse»>

<ul class=«nav» id=«side-menu»>

 <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard

 <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Группы

 Общий список

<! – /.nav-second-level – >

 <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Выбрать маршрут

 Маршруты <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Создать задание

 Задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив


```
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>



<div class = «col_3»>



<style type=«text/css»>
input {
outline: 1px solid lightgreen;
}
</style>
```

```
<! – Task – >
<form role=«form»>
<div class=«form-group»>
<label for=«task-name»> <h3> <span class=«label label-success»> Название: </span> </h3> </label>
<input type=«text» class=«form-control» id=«task-name»>
</div>
<div class=«form-group»>
<label for=«task-date»> <h3> <span class=«label label-success»> Дата: </span> </h3> </label>
<input type=«text» class=«form-control» id=«task-date»>
</div>
<div class=«form-group»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i> </div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {
$('#task-date').val(ev.date.toISOString().substring (0, 10));
});
</script>



</div>
<div class=«form-group»>
<label for=«task-user»> <h3> <span class=«label label-success»> Выбрать группу или сотрудника: </span> </h3> </label>



<div id=«task-user» class=«col_1»>



<div class=«clearfix»> </div>
</div>



</div>



<div class=«form-group»>
<label for=«task-desc»> <h3> <span class=«label label-success»> Выбрать описание задания: </span> </h3> </label>
```

<label for=«task-route»> <h3> <span class=«label label-success»> Выбрать маршрут: </h3> </label>

<button id=«task-submit» class=«btn btn-primary»> Сохранить </button>

</form>

<script type=«text/javascript»>
$ (document).ready (function () {

var url = 'http://buisness-control.appspot.com/groupslist/?callback=?';

$.ajax ({
type: «GET»,
url: url,
async: true,

```
jsonpCallback: ’getGroups’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.groups, function (key, val) {

var arrayName=Object.keys(json.groups[key]).toString ();

var subitems = [];

$.each (val [arrayName], function (subkey, subval) {
subitems. push (» <a href=«#task-user’ class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + subval.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»> <input type=’checkbox’ value=«» class=’task-checkbox task-checkbox-user’ data-user='"+subval.account+«»> " + subval.account + "</a> <div id=«» + subval.account.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class=’collapse’> <div class=’row’> <div class=’col-sm-6»> Имя </div> <div class='col-sm-6'>"+subval.firstName+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Фамилия </div> <div class='col-sm-6'>"+subval.secondName+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Телефон </div> <div class='col-sm-6'>"+subval.phone+"</div> </div> <div class=’row’> <div class=’col-sm-6»> Должность </div> <div class=’col-sm-6»>"+subval. position+"</div> </div> </div>»);
});

items. push (» <div> <a href=«#task-user’ class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + arrayName.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»> <input type=’checkbox’ value=«» class=’task-checkbox task-checkbox-group’> Группа: " + Object.keys(json.groups [key]) + "</a> <div id=«» + arrayName.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+subitems.join (»») +"</div> </div>»);
});

$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo («#task-user»);

$('.task-checkbox-group').change (function () {
if ($(this).is (»: checked’)) {
$(this).parent().closest('div').find(".task-checkbox-user").prop (’checked’, true);
} else {
$(this).parent().closest('div').find(".task-checkbox-user").prop (’checked’, false);
}
});
```

```
}
});


$ («#task-submit» ).click (function () {


var name=$('#task-name').val ();
var date=$('#task-date').val ();


var users = [];


$ (». task-checkbox-user’).each (function () {
if ($(this).is (»: checked’)) {
users.push($(this).attr (’data-user’));
}
});


var descs = [];


$ (». task-checkbox-desc’).each (function () {
if ($(this).is (»: checked’)) {
descs.push($(this).attr (’data-desc’));
}
});


var routes = [];


$ (». task-checkbox-route’).each (function () {
if ($(this).is (»: checked’)) {
var route = jQuery.parseJSON($(this).attr (’data-route’));
routes. push (route);
}
});


if (!confirm («Название "+name+"\nДата "+date+"\nСотрудники "+users+"\nОписание "+descs+"\nМаршрут "+JSON.stringify (routes) +«»)) {
```

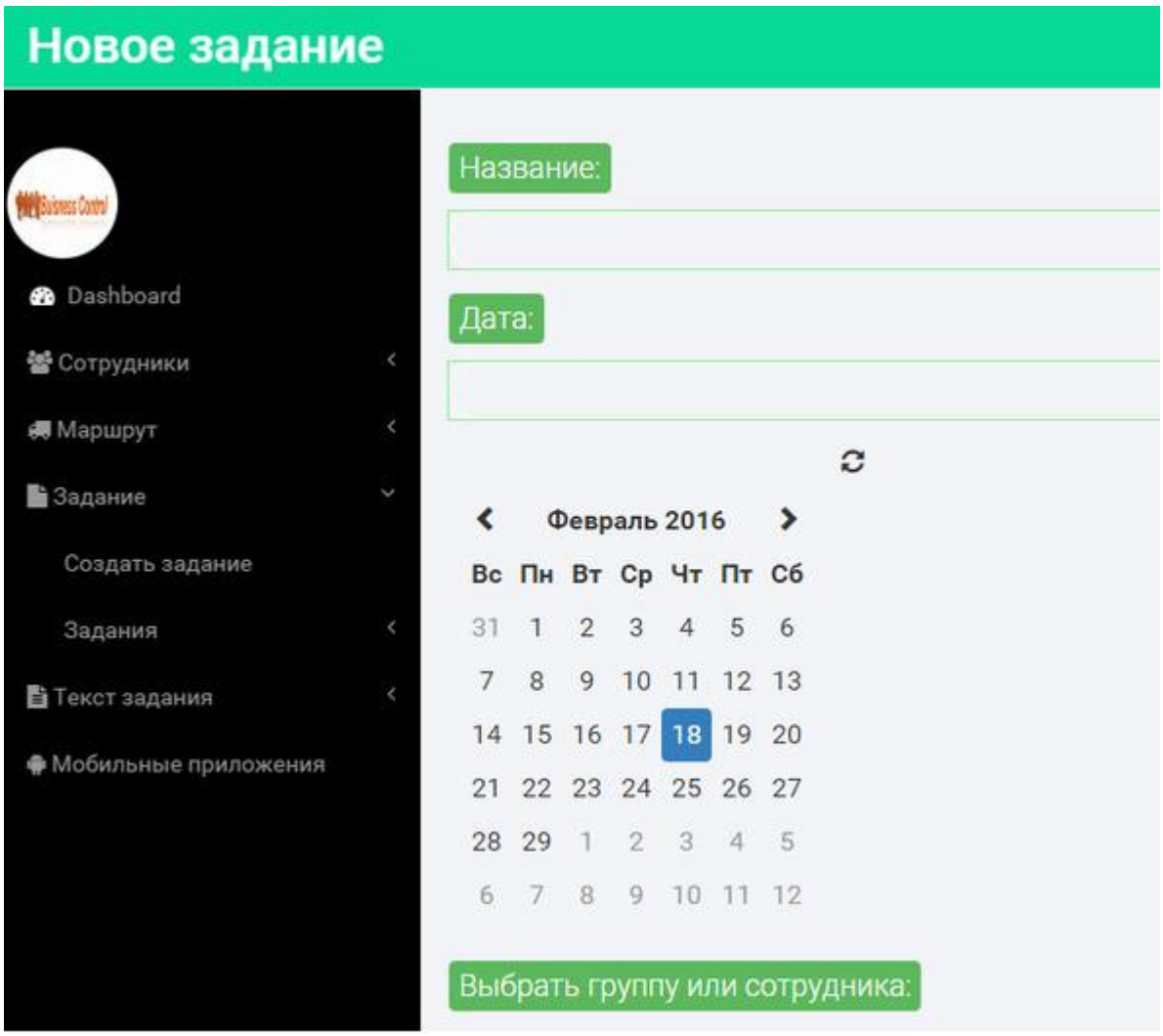
```
return false;
}


var url = 'http://buisness-control.appspot.com/puttask/?callback=?';


$.ajax ({
type: «GET»,
url: url,
data: {
«name»: name,
«date»: date,
«users»: JSON.stringify (users),
«descs»: JSON.stringify (descs),
«routes»: JSON.stringify (routes)
},
async: false,
jsonpCallback: ’putTask’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
$("#profile").modal («hide»);
location.reload ();
}
});


});


});


</script>


<script>
var url = 'http://buisness-control.appspot.com/descs/?callback=?';


$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’
},
async: true,
```

```
jsonpCallback: ’getNewDescs’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.descs, function (key, val) {

items. push (» <a href=«#task-desc’ class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»> <input type=’checkbox’ value=«» class=’task-checkbox task-checkbox-desc’ data-desc='"+val.desc+«»> " + val. date + "</a> <div id=«» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+val.desc+"</div>»);

});

$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo («#task-desc»);

}
});
</script>

<div class=«clearfix»> </div>
</div>

<div class=«col_1»>

<div class=«clearfix»> </div>
</div>

<div class=«span_11»>
<div class=«col-md-12»>
<div id=«map» style=«height:500px;"> </div>

<script>
var map;
```

```
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}
}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



<script>
var url = 'http://buisness-control.appspot.com/routes/?callback=?';



$.ajax ({
type: «GET»,
url: url,
```

```
data: {
state:’new’
},
async: true,
jsonpCallback: ’getNewRoutes’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json.routes, function (key, val) {



var subitems = [];



$.each (val.route, function (subkey, subval) {
subitems. push (» <div class=’row’> <div class=’col-sm-3»> Широта </div> <div class='col-sm-3'>"+subval.lat+"</div> <div class=’col-sm-3»> Долгота </div> <div class=’col-sm-3»>"+subval. lng+"</div> </div>»);
});



items. push (» <a href=«#task-route’ class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»> <input type=’checkbox’ value=«» class=’task-checkbox task-checkbox-route’ data-route='"+JSON.stringify(val.route) +«»> " + val. date + "</a> <div id=«» + val.date.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class='collapse'>"+subitems.join (»») +"<div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-map’ data-date=«»+val. date+«» data-route='"+JSON.stringify(val.route) +«»> Показать на карте </button> </div> </div> </div>»);



});



$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo («#task-route»);



$ (".btn-map» ).click (function () {
var route = jQuery.parseJSON($(this).attr (’data-route’));
var date = $(this).attr (’data-date’);



$.each (route, function (key, val) {
```

```
var latLng =new google.maps.LatLng(val.lat, val. lng);



var marker = new google.maps.Marker ({
position: latLng,
title: date,
map: map
});



map.setCenter (latLng);



});
});



}
});
</script>
</div>
<div class=«clearfix»> </div>
</div>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



Страница «Новые Задания» позволяет просмотреть задания со статусом «new», включая трекинги и отчеты о его выполнении, присланные сотрудниками.

<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >

<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>
<! – Editor – >
<script src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///cdn.tinymce.com/4/tinymce. min. js»> </script>
<script>tinymce.init ({selector: «#editor’}); </script>

</head>
<body>

<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>

<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
 Toggle navigation

</button>
 Новые Задания

<! – /.navbar-header – >

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>

<ul class=«nav» id=«side-menu»>

 <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard

 <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Группы

 Общий список

<! – /.nav-second-level – >


```
<li>
<a href=«#»> <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="route.html»> Выбрать маршрут </a>
</li>
<li>
<a href=«#»> Маршруты <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-routes.html»> Новые </a>
</li>
<li>
<a href="old-routes.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="task.html»> Создать задание </a>
</li>
<li>
<a href=«#»> Задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-tasks.html»> Новые </a>
</li>
<li>
<a href="old-tasks.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
```

```
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<div class=«clearfix»> </div>
</div>



<div class=«col_1»>
<div class=«col-md-6»>
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i>
</div>
<div class=«datepicker»> </div>



<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {



});
</script>
```

```
</div>
<div class=«clearfix»> </div>
</div>



<div class=«span_11»>
<div class=«col-md-12»>
<div id=«map» style=«height:500px;"> </div>



<script>
var map;
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});



if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}



}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>
```

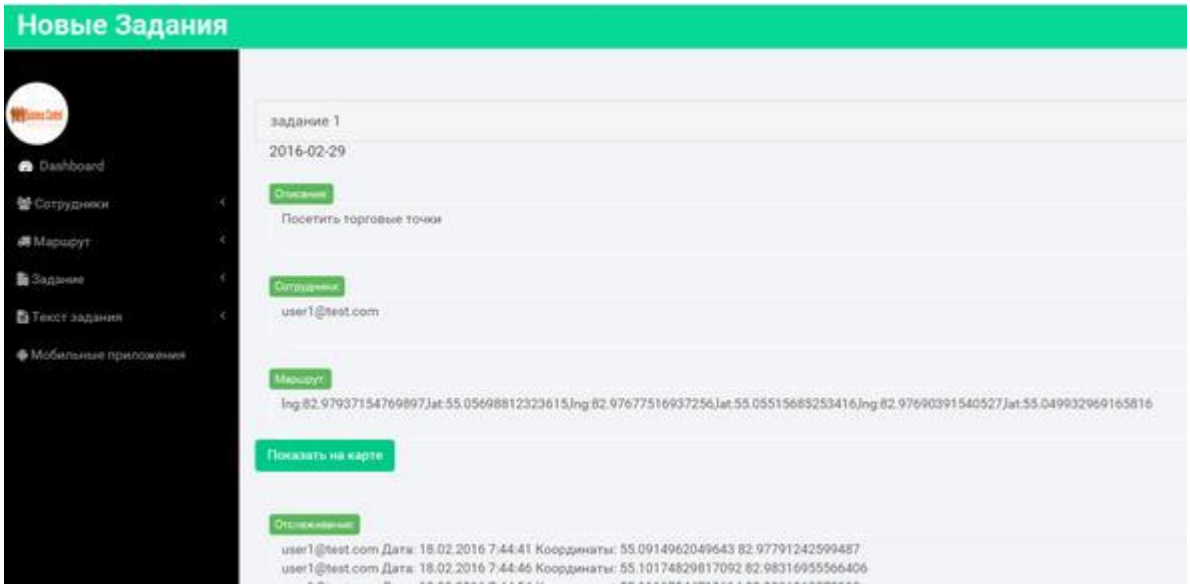
```
<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



<script>
var url = 'http://buisness-control.appspot.com/tasks/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’
},
async: true,
jsonpCallback: ’getNewTasks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json. tasks, function (key, val) {



var track=«»;
var trackRoute=«»;
var trackAccount=«»;



if (val.track===undefined) {
track=«»;
trackRoute=«»;
trackAccount=«»;
} else {
$.each (val.track, function (key, subval) {
track=track+"\n"+subval.account+" Дата: "+subval. date+" Координаты: "+subval.lat+"
"+subval. lng;
if(trackRoute=="")trackRoute=subval.lat+»,»+subval. lng;
trackRoute=trackRoute+","+subval.lat+»,»+subval. lng;
if(trackAccount=="")trackAccount=subval.account;
trackAccount=trackAccount+","+subval.account;
});
}



var subitems= [];
if (val.reports!==undefined) {
```

```
$.each (val.reports, function (key, subval) {
var reportAccount=subval.account;
var reportText=subval.report;
var reportPhotos=subval.photos;



var subsubitems= [];
if (reportPhotos!==undefined) {
$.each (reportPhotos, function (key, subsubval) {
var url=subsubval.servingUrl;
var key=subsubval. blobKey;
subsubitems. push (» <br/> <image class=’img-responsive’ src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/+url+" alt=«»> <br/>»);
});
}



subitems. push (» <div class=’row’> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Отчет: </p> <p>"+reportAccount+"</p> <p> <textarea class='form-control'>"+reportText+"</textarea></p><p>"+subsubitems.join (»») +"</p> </div> </div>»);



});
}



items. push (» <br/> <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.name.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val.name + "</a> <div id=«» + val.name.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class=’collapse’> <div class=’col-sm-12»>"+val. date+"</div> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Описание: </p> <textarea class='form-control'>"+JSON.stringify(val.descs).replace (/ [\"\ [\]] /g,»») +"</textarea> </div> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Сотрудники: </p> <textarea class='form-control'>"+JSON.stringify(val.users).replace (/ [\"\ [\]] /g,»») +"</textarea> </div> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Маршрут: </p> <textarea class='form-control'>"+JSON.stringify(val.routes).replace (/ [\"\ [\] \ {\}] /g,»») +"</textarea> </div> <div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-map’ data-title=«»+val. date+«» data-route='"+JSON.stringify(val.routes).replace (/ [\"\ [\] \ {\}] /g,»») +«»> Показать на карте </button> </div> </div> <br/> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Отслеживание: </p> <textarea class=’form-control’>"+track+"</textarea> </div> <br/> <div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-mapTrack’ data-title=«»+trackAccount+«» data-route=«»+trackRoute+«»> Показать на карте</button></div></div><br/>"+subitems.join (»») +"<br/> <div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-state’ data-name='"+val.name+«»> Отправить в архив </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-danger btn-del-task’ data-name='"+val.name+«»> Удалить задачу </button> </div> </div> </div>»);



});
```

```
$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo (".col_3»);



$ (".btn-map» ).click (function () {
var route = $(this).attr (’data-route’).split (»,»);
var title = $(this).attr (’data-title’);
for (var i = 0; i <route. length; i++) {



var latLng =new google.maps.LatLng(route[i+1].replace («lat:»,»»), route[i].replace («lng:»,»»));
var marker = new google.maps.Marker ({
position: latLng,
title: title,
map: map
});
map.setCenter (latLng);
i=i+1;
}



});



$ (".btn-mapTrack» ).click (function () {
var route = $(this).attr (’data-route’).split (»,»);



var title = $(this).attr (’data-title’).split (»,»);
var j=0;
for (var i = 0; i <route. length; i++) {



var latLng =new google.maps.LatLng (route [i], route [i+1]);
var marker = new google.maps.Marker ({
position: latLng,
title: title [j],
map: map
});
map.setCenter (latLng);
i=i+1;
j=j+1;
}
```

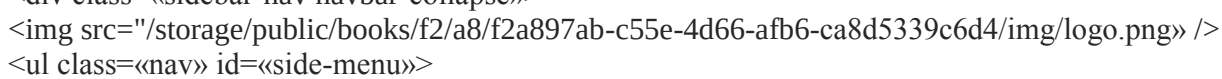
```
});


$ (".btn-state» ).click (function () {
var name = $(this).attr (’data-name’);


var url = 'http://buisness-control.appspot.com/puttaskstate/?callback=?';


$.ajax ({
type: «GET»,
url: url,
data: {
state:’old’,
name: name
},
async: false,
jsonpCallback: ’putTaskState’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});


});


$ (".btn-del-task» ).click (function () {
var name = $(this).attr (’data-name’);


var url = 'http://buisness-control.appspot.com/deletetask/?callback=?';


$.ajax ({
type: «GET»,
url: url,
data: {
name: name
},
async: false,
jsonpCallback: ’deleteTask’,
contentType: ’application/json’,
dataType: ’jsonp’,
```

```
success: function (json) {
location.reload ();
}
});



});
}
});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



Страница «Архив Заданий» позволяет просмотреть задания со статусом «old», включая трекинги и отчеты о его выполнении, присланные сотрудниками.

```
<!DOCTYPE HTML>
<html>
<head>
<title> Business Control </title>
<meta name=«viewport» content=«width=device-width, initial-scale=1»>
<meta http-equiv=«Content-Type» content=«text/html; charset=utf-8» />
<script type=«application/x-javascript»> addEventListener («load», function () {setTimeout (hideURLbar, 0);}, false); function hideURLbar () {window.scrollTo (0,1);} </script>
<! – Bootstrap Core CSS – >
<link href=«css/bootstrap. min. css» rel=’stylesheet’ type=’text/css’ />
<! – Custom CSS – >
<link href=«css/style. css» rel=’stylesheet’ type=’text/css’ />
<link href=«css/font-awesome. css» rel=«stylesheet»>
<! – jQuery – >
<script src=«js/jquery. min. js»> </script>
<! – — webfonts – ->
<link href='//fonts.googleapis.com/css? family=Roboto:400,100,300,500,700,900» rel=’stylesheet’ type=’text/css’>
<! – Nav CSS – >
<link href=«css/custom. css» rel=«stylesheet»>
<! – Metis Menu Plugin JavaScript – >
<script src=«js/metisMenu. min. js»> </script>
<script src=«js/custom. js»> </script>
<! – Calendar – >
<script type=«text/javascript» src=«js/bootstrap-datepicker. js»> </script>
<script type=«text/javascript» src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/js/bootstrap-datepicker.ru.js» charset=«UTF-8»> </script>
<link href=«css/datepicker. css» rel=«stylesheet»>
<! – Editor – >
<script src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4///cdn.tinymce.com/4/tinymce. min. js»> </script>
<script>tinymce.init ({selector: «#editor’}); </script>


</head>
<body>
<div id=«wrapper»>
<! – Navigation – >
<nav class=«top1 navbar navbar-default navbar-static-top» role=«navigation» style=«margin-bottom: 0»>
<div class=«navbar-header»>
<button type=«button» class=«navbar-toggle» data-toggle=«collapse» data-target=".navbar-collapse»>
<span class=«sr-only»> Toggle navigation </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
<span class=«icon-bar»> </span>
</button>
<a class=«navbar-brand» href=«#»> Архив Заданий </a>
</div>
<! – /.navbar-header – >
```

<div class=«navbar-default sidebar» role=«navigation»>
<div class=«sidebar-nav navbar-collapse»>

<ul class=«nav» id=«side-menu»>

 <i class=«fa fa-dashboard fa-fw nav_icon»> </i> Dashboard

 <i class=«fa fa-users»> </i> Сотрудники <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Группы

 Общий список

<! – /.nav-second-level – >

 <i class=«fa fa-truck»> </i> Маршрут <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Выбрать маршрут

 Маршруты <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

<! – /.nav-second-level – >

 <i class=«fa fa-file»> </i> Задание <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Создать задание

 Задания <span class=«fa arrow»>
<ul class=«nav nav-second-level»>

 Новые

 Архив

```
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-file-text»> </i> Текст задания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="desc.html»> Создать описание </a>
</li>
<li>
<a href=«#»> Описания <span class=«fa arrow»> </span> </a>
<ul class=«nav nav-second-level»>
<li>
<a href="new-descs.html»> Новые </a>
</li>
<li>
<a href="old-descs.html»> Архив </a>
</li>
</ul>
</li>
</ul>
<! – /.nav-second-level – >
</li>
<li>
<a href=«#»> <i class=«fa fa-android»> </i> Мобильные приложения </a>



</li>



</ul>
</div>
<! – /.sidebar-collapse – >
</div>
<! – /.navbar-static-side – >
</nav>
<div id=«page-wrapper»>
<div class=«graphs»>
<div class = «col_3»>



<div class=«clearfix»> </div>
</div>



<div class=«col_1»>
<div class=«col-md-6»>
```

```
<div class=«refresh» style=«cursor: pointer; margin-left:22%"> <i class=«fa fa-refresh»> </i> </div>
<div class=«datepicker»> </div>

<script type=«text/javascript»>
var dp = $ (». datepicker’).datepicker ({
language: «ru-RU»
});
$ (".refresh» ).click (function () {
dp. datepicker (’update’);
});
dp. on (’changeDate’, function (ev) {

});
</script>

</div>
<div class=«clearfix»> </div>
</div>

<div class=«span_11»>
<div class=«col-md-12»>
<div id=«map» style=«height:500px;"> </div>

<script>
var map;
function initMap () {
map = new google.maps.Map(document.getElementById (’map’), {
zoom: 15
});
var infoWindow = new google.maps.InfoWindow ({map: map});

if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition (function (position) {
var pos = {
lat: position.coords.latitude,
lng: position.coords. longitude
};
infoWindow.setPosition (pos);
map.setCenter (pos);
}, function () {
handleLocationError (true, infoWindow, map.getCenter ());
});
```

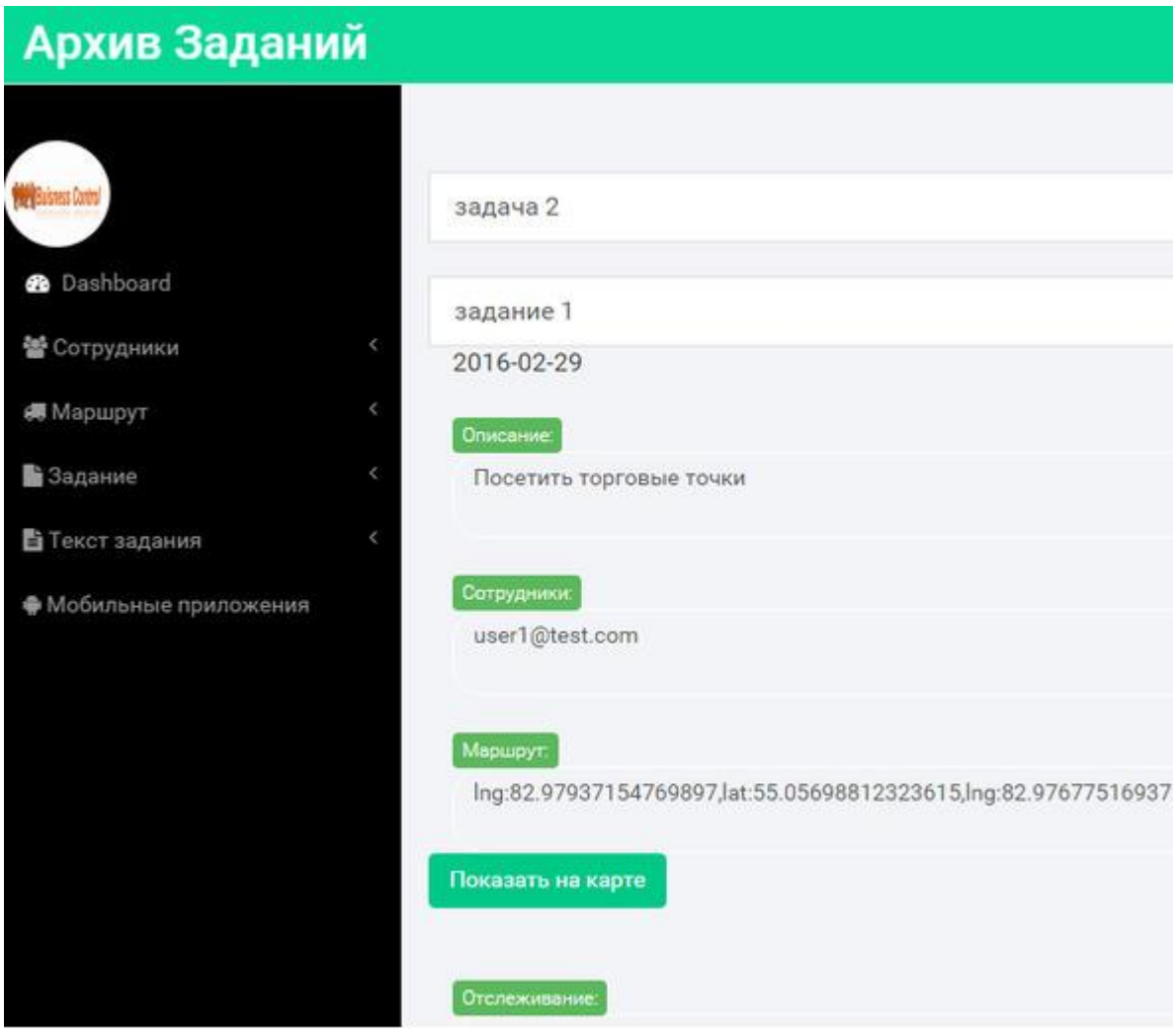
```
} else {
// Browser doesn’t support Geolocation
handleLocationError (false, infoWindow, map.getCenter ());
}



}



function handleLocationError (browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition (pos);
infoWindow.setContent (browserHasGeolocation?
«Error: The Geolocation service failed.» :
«Error: Your browser doesn\’t support geolocation.»);
}



</script>



<script
src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/https://maps.googleapis.com/maps/api/js?key=AIzaSyCb9Iir38HyF_E3_hmf3mHHum_tJbgvjXs&callback=initMap"
async defer> </script>



<script>
var url = 'http://buisness-control.appspot.com/tasks/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
state:’old’
},
async: true,
jsonpCallback: ’getNewTasks’,
contentType: «application/json»,
dataType: ’jsonp’,
success: function (json) {
var items = [];
$.each (json. tasks, function (key, val) {



var track=«»;
var trackRoute=«»;
var trackAccount=«»;
```

```
if (val.track===undefined) {
track=«»;
trackRoute=«»;
trackAccount=«»;
} else {
$.each (val.track, function (key, subval) {
track=track+"\n"+subval.account+" Дата: "+subval. date+" Координаты: "+subval.lat+" "+subval. lng;
if(trackRoute=="")trackRoute=subval.lat+»,»+subval. lng;
trackRoute=trackRoute+","+subval.lat+»,»+subval. lng;
if(trackAccount=="")trackAccount=subval.account;
trackAccount=trackAccount+","+subval.account;
});
}



var subitems= [];
if (val.reports!==undefined) {
$.each (val.reports, function (key, subval) {
var reportAccount=subval.account;
var reportText=subval.report;
var reportPhotos=subval.photos;



var subsubitems= [];
if (reportPhotos!==undefined) {
$.each (reportPhotos, function (key, subsubval) {
var url=subsubval.servingUrl;
var key=subsubval. blobKey;
subsubitems. push (» <br/> <image class=’img-responsive’ src="/storage/public/books/f2/a8/f2a897ab-c55e-4d66-afb6-ca8d5339c6d4/+url+" alt=«»> <br/>»);
});
}



subitems. push (» <div class=’row’> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Отчет: </p> <p>"+reportAccount+"</p> <p> <textarea class='form-control'>"+reportText+"</textarea></p><p>"+subsubitems.join (»») +"</p> </div> </div>»);



});
}



items. push (» <br/> <a href=«#» class=’list-group-item’ data-toggle=’collapse’ data-target=«#» + val.name.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «»>" + val.name + "</a> <div id=«» + val.name.replace (/ [\.,\:,\/,,\ (,\),=,?, -,@] /g,»») + «» class=’collapse’> <div
```

```
class=’col-sm-12»>"+val. date+"</div> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Описание: </p> <textarea class='form-control'>"+JSON.stringify(val.descs).replace (/ [\"\ [\]] /g,»») +"</textarea> </div> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Сотрудники: </p> <textarea class='form-control'>"+JSON.stringify(val.users).replace (/ [\"\ [\]] /g,»») +"</textarea> </div> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Маршрут: </p> <textarea class='form-control'>"+JSON.stringify(val.routes).replace (/ [\"\ [\] \ {\}] /g,»») +"</textarea> </div> <div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-map’ data-title=«»+val. date+«» data-route="'+JSON.stringify(val.routes).replace (/ [\"\ [\] \ {\}] /g,»») +«»> Показать на карте </button> </div> </div> <br/> <div class=’col-sm-12»> <br/> <p class=’label label-success’> Отслеживание: </p> <textarea class=’form-control’>"+track+"</textarea> </div> <br/> <div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-mapTrack’ data-title=«»+trackAccount+«» data-route=«»+trackRoute+«»> Показать на карте</button></div></div><br/>"+subitems.join (»») +"<br/> <div class=’row’> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-primary btn-state’ data-name="'+val.name+«»> Новое задание </button> </div> <div class=’col-sm-4»> <button type=’button’ class=’btn btn-danger btn-del’ data-name="'+val.name+«»> Удалить </button> </div> </div> </div>»);



});



$ (» <div/>», {
«class»: «list-group»,
html: items.join (»»)
}).appendTo (".col_3»);



$ (".btn-mapTrack» ).click (function () {
var route = $(this).attr (’data-route’).split (»,»);



var title = $(this).attr (’data-title’).split (»,»);
var j=0;
for (var i = 0; i <route. length; i++) {



var latLng =new google.maps.LatLng (route [i], route [i+1]);
var marker = new google.maps.Marker ({
position: latLng,
title: title [j],
map: map
});
map.setCenter (latLng);
i=i+1;
j=j+1;
}



});
```

```
$ (".btn-map» ).click (function () {
var route = $(this).attr (’data-route’).split (»,»);
var date = $(this).attr (’data-date’);
for (var i = 0; i <route. length; i++) {


var latLng =new google.maps.LatLng(route[i+1].replace («lat:»,»»), route[i].replace («lng:»,»»));
var marker = new google.maps.Marker ({
position: latLng,
title: date,
map: map
});
map.setCenter (latLng);
i=i+1;
}


});


$ (".btn-state» ).click (function () {
var name = $(this).attr (’data-name’);


var url = 'http://buisness-control.appspot.com/puttaskstate/?callback=?';


$.ajax ({
type: «GET»,
url: url,
data: {
state:’new’,
name: name
},
async: false,
jsonpCallback: ’putTaskState’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});


});
```

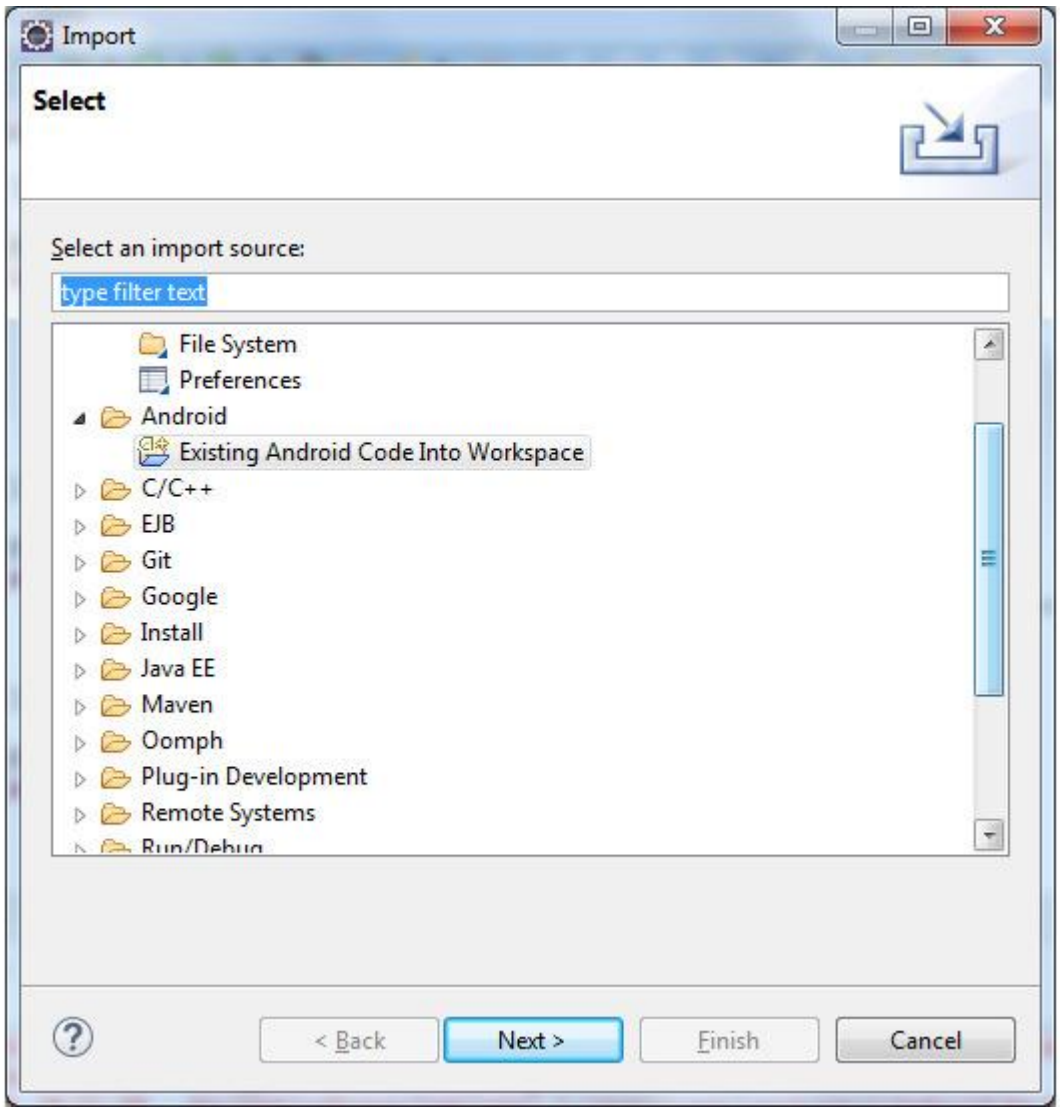
```
$ (".btn-del» ).click (function () {
var name = $(this).attr (’data-name’);



var url = 'http://buisness-control.appspot.com/deletetask/?callback=?';



$.ajax ({
type: «GET»,
url: url,
data: {
name: name
},
async: false,
jsonpCallback: ’deleteTask’,
contentType: ’application/json’,
dataType: ’jsonp’,
success: function (json) {
location.reload ();
}
});



});
}
});
</script>



</div>
<div class=«clearfix»> </div>
</div>



<div class=«copy»>
<p> Copyright &copy; 2015 TM SoftStudio </p>
</div>
</div>
</div>
<! – /#page-wrapper – >
</div>
<! – /#wrapper – >
<! – Bootstrap Core JavaScript – >
<script src=«js/bootstrap. min. js»> </script>
</body>
</html>
```



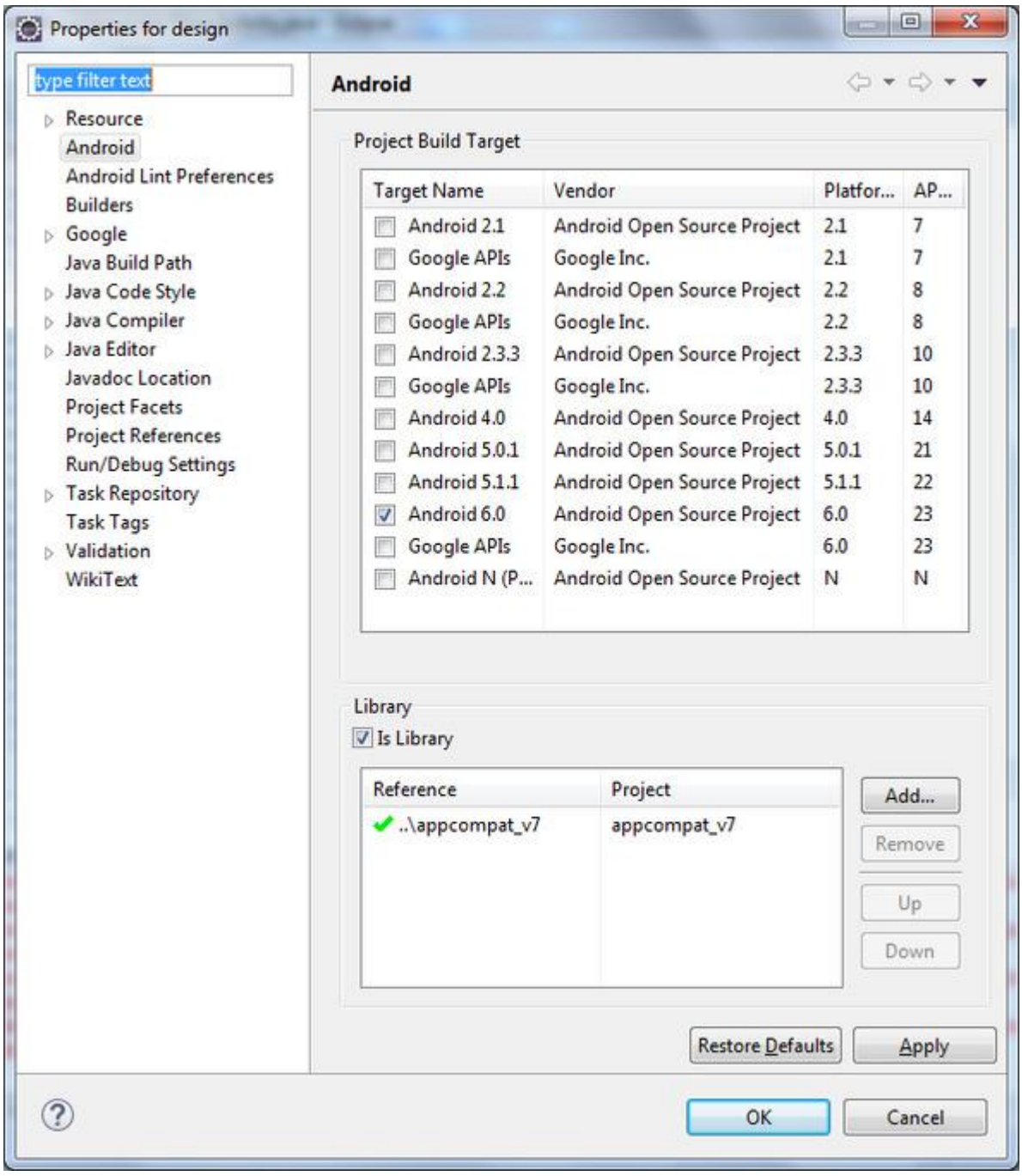
Использование Material Design в Eclipse

При создании Android проектов, работающих с Material Design, в среде Eclipse, в проект необходимо добавить библиотеки andoid design support и recyclerview.

Для добавления библиотеки andoid design support в среде Eclipse импортируем с сохранением папку android-sdks\extras\android\support\design.



В свойствах библиотеки в разделе Android в Project Build Target отметим последнюю версию API и добавим зависимость от библиотеки appcompat_v7.



Для добавления библиотеки recyclerview в среде Eclipse импортируем с сохранением папку android-sdks\extras\android\support\v7\recyclerview.

В свойствах библиотеки в разделе Android в Project Build Target отметим последнюю версию API и добавим зависимость от библиотеки appcompat_v7.

Создадим проект Android приложения на основе шаблона Blank Activity.

В свойствах проекта в разделе Android в Project Build Target добавим зависимость от библиотек appcompat_v7, andoid design support и recyclerview.

В Android Studio создадим проект на основе шаблона Navigation Drawer Activity.

Скопируем содержимое папок app\src\main Android Studio проекта в Eclipse проект.

Push уведомления

Android + Google Cloud Messaging (GCM)

В консоли Google Cloud Platform создадим проект.

Включим Google Cloud Messaging API для этого проекта.

Создадим server API key для Google Cloud Messaging.

В среде Eclipse создадим проект Android приложения, в который добавим библиотеку Google Play Services https://developers.google.com/android/guides/setup#add_google_play_services_to_your_project.

В файл манифеста добавим разрешения.

```
<uses-permission android:name="android.permission.INTERNET» />
<uses-permission android:name="android.permission. WAKE_LOCK» />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE» />
<permission android:name="com.tmsoftstudio.breakingnews. C2D_MESSAGE»
android: protectionLevel=«signature» />
<uses-permission android:name="com.tmsoftstudio.breakingnews. C2D_MESSAGE» />
```

Для регистрации Android приложения в сервисе GCM в класс активности добавим код:

```
private String regid;
private String PROJECT_NUMBER = «1222…»;
private SharedPreferences mSettings;
private Context context;
final int REQUEST_CODE = 0;
public final static int RESULT_CODE = 1;
public final static String PARAM_PINTENT = «PendingIntent»;
public final static String PARAM_RESULT = «Result»;



@Override
protected void onCreate (Bundle savedInstanceState) {
super. onCreate (savedInstanceState);
setContentView(R.layout.activity_main);
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar (toolbar);



context = this;
mSettings = getSharedPreferences («APP_PREFERENCES», Context.MODE_PRIVATE);



if (!mSettings.contains («SENDER_ID»)) {
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («SENDER_ID», PROJECT_NUMBER);
editor.commit ();
}



if (!mSettings.contains («REG_ID»)) {
PendingIntent pi;
pi = createPendingResult (REQUEST_CODE, new Intent (), 0);
Intent intent = new Intent (context, InstanceIDIntentService.class);
intent. putExtra (PARAM_PINTENT, pi);
startService (intent);
}
```

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener (new View. OnClickListener () {
@Override
public void onClick (View view) {
Snackbar.make (view, «Replace with your own action», Snackbar. LENGTH_LONG)
.setAction («Action», null).show ();
}
});



DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle (
this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
drawer.addDrawerListener (toggle);
toggle.syncState ();



NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener (this);
}



@Override
protected void onActivityResult (int requestCode, int resultCode, Intent data) {
if (requestCode == REQUEST_CODE) {
if (resultCode == RESULT_CODE) {
regid=data.getStringExtra(MainActivity.PARAM_RESULT);
View view = (View) findViewById(R.id.content);
Snackbar.make (view, regid, Snackbar.LENGTH_LONG).setAction («Action», null).show ();
SharedPreferences. Editor editor = mSettings. edit ();
editor. putString («REG_ID», regid);
editor.commit ();



}
}
}
```

Здесь PROJECT_NUMBER это номер созданного проекта в Google Cloud Platform.

Объект PendingIntent создается методом createPendingResult активности для получения уникального регистрационного токена из сервиса регистрации для его сохранения в методе обратного вызова onActivityResult активности.

Создадим сервис регистрации.

```
<service android:name=".InstanceIDIntentService» android: exported=«false»/>
import java.io.BufferedWriter;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net. URL;
```

```
import com.google.android.gms. gcm. GoogleCloudMessaging;
import com.google.android.gms.iid.InstanceID;



import android.app.IntentService;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net. Uri;
import android. os. Handler;
import android. os. Looper;
import android.widget.Toast;



public class InstanceIDIntentService extends IntentService {



// abbreviated tag name
private static final String TAG = «InstanceIDIntentService»;
private SharedPreferences mSettings;



public InstanceIDIntentService () {
super (TAG);
}



@Override
protected void onHandleIntent (Intent intent) {
InstanceID instanceID = InstanceID.getInstance (this);
mSettings = getSharedPreferences («APP_PREFERENCES», Context.MODE_PRIVATE);
String senderId = mSettings.getString («SENDER_ID», «»);
try {
String token = instanceID.getToken (senderId, GoogleCloudMessaging.INSTANCE_ID_SCOPE);
int code=sendRegistrationToServer (token);
if (code==200) {
PendingIntent pi = intent.getParcelableExtra(MainActivity.PARAM_PINTENT);
Intent intentResult = new Intent().putExtra(MainActivity.PARAM_RESULT, token);
pi.send(InstanceIDIntentService.this,MainActivity.RESULT_CODE, intentResult);
} else {
Handler handler = new Handler(Looper.getMainLooper ());
handler.post (new Runnable () {
@Override
public void run () {Toast.makeText(InstanceIDIntentService.this.getApplicationContext
(),«Registration failed!",Toast.LENGTH_SHORT).show ();
}
});
}
```

```
} catch (Exception e) {
e.printStackTrace ();
}
}


private int sendRegistrationToServer (String token) {
URL url;
HttpURLConnection conn=null;
int code=0;
try {
url = new URL (»http://backend.appspot.com/backend");
conn = (HttpURLConnection) url. openConnection ();
conn.setReadTimeout (10000);
conn.setConnectTimeout (15000);
conn.setRequestMethod («POST»);
conn.setDoInput (true);
conn.setDoOutput (true);


Uri. Builder builder = new Uri. Builder ()
.appendQueryParameter («token», token);
String query = builder.build().getEncodedQuery ();


OutputStream os = conn.getOutputStream ();
BufferedWriter writer = new BufferedWriter (
new OutputStreamWriter (os, «UTF-8»));
writer. write (query);
writer. flush ();
writer.close ();
os.close ();
code=conn.getResponseCode ();
} catch (Exception e) {
e.printStackTrace ();
} finally {
if (conn!= null) {
conn. disconnect ();
}
}
return code;
}
}
```

Здесь с помощью номера проекта создается запрос к сервису GCM и в ответ получается уникальный регистрационный токен, который затем отсылается на сервер и передается с помощью объекта PendingIntent обратно в активность.

Для создания серверной части в среде Eclipse создадим проект Web Application Project, в файле appengine-web. xml которого укажем идентификатор проекта Google Cloud Platform.

В сервлет GAE приложения добавим код сохранения токена.

```
import java.io.IOException;
```

```
import javax.servlet.http.*;



import com. google. appengine. api. datastore. DatastoreService;
import com. google. appengine. api. datastore. DatastoreServiceFactory;
import com. google. appengine. api. datastore. Entity;



@SuppressWarnings («serial»)
public class BackendServlet extends HttpServlet {
public void doPost (HttpServletRequest req, HttpServletResponse resp) throws IOException {
String token=req.getParameter («token»);
DatastoreService datastore = DatastoreServiceFactory.getDatastoreService ();
Entity tokenEntity = new Entity («Token»);
tokenEntity.setProperty («token», token);
datastore. put (tokenEntity);
}
}
```

Так как токен периодически обновляется сервисом GCM, нужно создать слушатель, который будет автоматически запускаться при обновлении токена сервисом GCM и обновлять его для Android приложения и для серверной части.

```
<service
android:name=".RefreshInstanceIDListenerService»
android: exported=«false»>
<intent-filter>
<action android:name="com.google.android.gms.iid.InstanceID»/>
</intent-filter>
</service>
import com.google.android.gms.iid.InstanceIDListenerService;
import android.content.Intent;



public class RefreshInstanceIDListenerService extends InstanceIDListenerService {



@Override
public void onTokenRefresh () {
// Fetch updated Instance ID token and notify of changes
Intent intent = new Intent (this, InstanceIDIntentService.class);
startService (intent);
}
}
```

Для отправки сообщения от сервера Android приложению, в папку lib и в Build Path проекта GAE приложения добавим библиотеки json-simple-1.1.jar (http://grepcode.com/snapshot/repo1.maven.org/maven2/com.googlecode.json-simple/json-simple/1.1) и gcm-server. jar (https://github.com/slorber/gcm-server-repository/tree/master/deployer).

В код сервлета добавим:

```
import java.io.IOException;
import javax.servlet.http.*;
```

```
import com.google.android.gcm.server.Message;
import com.google.android.gcm.server.Sender;
import com.google.android.gcm.server.Result;
import com. google. appengine. api. datastore. DatastoreService;
import com. google. appengine. api. datastore. DatastoreServiceFactory;
import com. google. appengine. api. datastore. Entity;



@SuppressWarnings («serial»)
public class BackendServlet extends HttpServlet {
private String SERVER_KEY=«AIza…»;



public void doPost (HttpServletRequest req, HttpServletResponse resp) throws IOException {
String token=req.getParameter («token»);
DatastoreService datastore = DatastoreServiceFactory.getDatastoreService ();
Entity tokenEntity = new Entity («Token»);
tokenEntity.setProperty («token», token);
datastore. put (tokenEntity);



Sender sender = new Sender (SERVER_KEY);
Message message = new Message. Builder ()
.addData («sender», «Project Name»)
.addData («message», «this is the message»)
.build ();
Result result = sender.send (message, token, 1);
}
}
```

Здесь SERVER_KEY это созданный server API key для Google Cloud Messaging.

Для получения сообщения Android приложением, в файл манифеста добавим объявление сервиса, который будет обрабатывать GCM сообщение.

```
<receiver
android:name="com.google.android.gms. gcm. GcmReceiver»
android: exported=«true»
android:permission="com.google.android.c2dm.permission.SEND»>
<intent-filter>
<action android:name="com.google.android.c2dm.intent.RECEIVE» />
<category android:name="com.tmsoftstudio.breakingnews» />
</intent-filter>
<intent-filter>
<action android:name="com.google.android.c2dm.intent.REGISTRATION»/>
<category android:name="com.tmsoftstudio.breakingnews»/>
</intent-filter>
</receiver>
```

А также создадим сервис обработки входящего сообщения.

```
<service
android: name=». GcmMessageHandler»
android: exported=«false»>
<intent-filter>
<action android:name="com.google.android.c2dm.intent.RECEIVE» />
```

```
</intent-filter>
</service>
import com.google.android.gms. gcm. GcmListenerService;
import android.app.NotificationManager;
import android.content.Context;
import android. os. Bundle;
import android.support.v4.app.NotificationCompat;



public class GcmMessageHandler extends GcmListenerService {
public static final int MESSAGE_NOTIFICATION_ID = 123123;



@Override
public void onMessageReceived (String from, Bundle data) {
String title = data.getString («sender»);
String message = data.getString («message»);



Context context = getBaseContext ();
NotificationCompat. Builder mBuilder = new NotificationCompat. Builder (context)
.setSmallIcon(R.drawable.ic_launcher).setContentTitle (title)
.setContentText (message);
NotificationManager mNotificationManager = (NotificationManager) context
.getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify (MESSAGE_NOTIFICATION_ID, mBuilder. build ());
}
}
```

Для рассылки topic сообщений в проекте GAE приложения заменим библиотеку gcm-server. jar (https://github.com/slorber/gcm-server-repository/tree/master/deployer) на библиотеку https://github.com/google/gcm.

Это позволит заменить строку sender.send (message, token, 1); на sender.send (message, "/topics/news», 1);

Для подписки Android приложения на topic сообщения в сервисе InstanceIDIntentService в методе onHandleIntent добавим код:

```
String token = instanceID.getToken (senderId, GoogleCloudMessaging.INSTANCE_ID_SCOPE);
GcmPubSub pubSub = GcmPubSub.getInstance (this);
pubSub.subscribe (token, "/topics/news», null);
```

Изменим сервис GcmMessageHandler.

```
import com.google.android.gms. gcm. GcmListenerService;
import android.app.NotificationManager;
import android.content.Context;
import android. os. Bundle;
import android.support.v4.app.NotificationCompat;



public class GcmMessageHandler extends GcmListenerService {
public static final int MESSAGE_NOTIFICATION_ID = 123123;
```

```
@Override
public void onMessageReceived (String from, Bundle data) {
String title = data.getString («sender»);
String message = data.getString («message»);



if (from.startsWith (»/topics/news»)) {
Context context = getBaseContext ();
NotificationCompat. Builder mBuilder = new NotificationCompat. Builder (context)
.setSmallIcon(R.drawable.ic_launcher).setContentTitle (title+«Topic: News»)
.setContentText (message);
NotificationManager mNotificationManager = (NotificationManager) context
.getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify (MESSAGE_NOTIFICATION_ID, mBuilder. build ());
} else {
Context context = getBaseContext ();
NotificationCompat. Builder mBuilder = new NotificationCompat. Builder (context)
.setSmallIcon(R.drawable.ic_launcher).setContentTitle (title)
.setContentText (message);
NotificationManager mNotificationManager = (NotificationManager) context
.getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify (MESSAGE_NOTIFICATION_ID, mBuilder. build ());
}
}
}
```

Cordova + Google Cloud Messaging (GCM)

Для Cordova приложений регистрацию и получение Push уведомлений обеспечивает Cordova плагин phonegap-plugin-push (https://github.com/phonegap/phonegap-plugin-push).

В среде Intel SDK в разделе Plugin Management вкладки Projects добавим плагин phonegap-plugin-push. В разделе Bower Managed Libraries добавим библиотеку jquery.

В файл index.html добавим библиотеку jquery.

```
<script src=«bower_components/jquery/dist/jquery. min. js»> </script>
```

В файл app. js добавим код регистрации и обработки сообщений.

```
function onAppReady () {
if (navigator. splashscreen && navigator. splashscreen. hide) {navigator. splashscreen. hide ();
}
var push = PushNotification.init ({
android: {
senderID: «1222…»
},
ios: {
alert: «true»,
badge: «true»,
sound: «true»
},
windows: {}
});

push. on (’registration’, function (data) {
var token=data.registrationId;
jQuery. ajax ({
method: «POST»,
url: "http://backend.appspot.com/backend",
data: {token: token}
```

```
});
});

push. on (’notification’, function (data) {
alert (data. title+" Message: " +data.message);
});

push. on (’error’, function (e) {
alert(e.message);
});

}
document.addEventListener("app.Ready», onAppReady, false);
```
Здесь senderID это номер проекта Google Cloud Platform.
Код сервлета, посылающего сообщение:
```
import java.io.IOException;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.android.gcm.server.Message;
import com.google.android.gcm.server.Sender;
import com. google. appengine. api. datastore. DatastoreService;
import com. google. appengine. api. datastore. DatastoreServiceFactory;
import com. google. appengine. api. datastore. Entity;

@SuppressWarnings («serial»)
public class Backend_Breaking_NewsServlet extends HttpServlet {
private String SERVER_KEY=«AIzaS…»;

public void doPost (HttpServletRequest req, HttpServletResponse resp) throws IOException {
String token=req.getParameter («token»);
DatastoreService datastore = DatastoreServiceFactory.getDatastoreService ();
Entity tokenEntity = new Entity («Token»);
tokenEntity.setProperty («token», token);
datastore. put (tokenEntity);

Sender sender = new Sender (SERVER_KEY);
Message message = new Message. Builder ()
.addData («title», «Breaking News»)
.addData («message», «this is the message»)
.build ();
sender.send (message, token, 1);
}
}
```
Здесь SERVER_KEY это созданный server API key для Google Cloud Messaging.