МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой _____
Кулгильдеев З.К. профессор
(Ф.И.О., ученая степень, звание)
_____ «_____»_____20__ г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: „Исследование кроссплатформенных приложений, на примере мобильного приложения для Android OS."

Специальность „Вычислительная техника и

Выполнил (а) Прончешко Д.В ВТу 11-1
(Фамилия и инициалы) группа

Научный руководитель Сербин В.В. ст. преп., К.Т.Н
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Ержанова З.Р. ст. преп.
(Фамилия и инициалы, ученая степень, звание)
Ержанова «20» мая 20 14 г.
(подпись)

по безопасности жизнедеятельности:
Белгибаева А.С. ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «28» мая 20 14 г.
(подпись)

по применению вычислительной техники:
_____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

Нормоконтролер: Тусупов Д.М. ассистент
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

Рецензент: Сыздыков Е.К. канд. техн. наук.
(Фамилия и инициалы, ученая степень, звание)
_____ «30» 05 20 14 г.
(подпись)

Алматы 2014 г.

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Прокопенко Дмитрий Викторович
(фамилия, имя, отчество)

Тема проекта Исследование кроссплатформенных приложений на примере мобильного приложения для Android OS

утверждена приказом ректора № 115 от «24» сентября 2014 г.
Срок сдачи законченной работы «21» июня 2014 г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Рынок мобильных приложений растет с каждым годом. Появляются все новые мобильные платформы. Приложения персонализации являются одними из самых популярных в топах магазинов приложений. Следует разработать приложение представляющее стильный внешний вид и широкие возможности применения интерфейса.

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1. Анализ рынка мобильных приложений.
2. Анализ предметной области
3. Разработка интерфейса
4. Проектирование приложения
5. Отладка приложения
6. Расчет технико-экономического обоснования
7. Описание безопасности жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей)

Рисунки архитектуры системы
Рисунки интерфейса инструментов разработки
Рисунки, описывающие требования к приложению
Рисунки интерфейса приложения
Таблицы для описания технико-экономического обоснования
Рисунки, описывающие технико-экономическое обоснование
Рисунки, описывающие безопасность жизнедеятельности
Таблицы с данными о безопасности жизнедеятельности

Рекомендуемая основная литература

http://startandroid.ru
http://developer.android.com
http://www.youtube.com/user/freshgamer10/videos

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
Экономика	Евсеева З.Ф.	10.04 - 20.05.14	Евсеева
БЖД	Великий А.С.	10.04 - 23.05.14	
Нормоконтроль			

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Сбор необходимого материала	01.04.14	
2	Исследование предметной области	03.04.14	
3	Постановка задачи	04.04.14	
4	Выбор платформы и среды разработки	04.04.14	
5	Разработка интерфейса	05.04.14	
6	Разработка форм приложения	15.04.14	
7	Тестирование и отладка	25.04.14	
8	Оформление отчета	05.05.14	
9	Написание раздела "технико-экономическое обоснование"	10.05.14	
10	Написание раздела "безопасность жизнедеятельности"	26.05.14	

Дата выдачи задания «03» марта 2014 г.

Заведующий кафедрой _____ Курамбаев З.К.
(подпись) (Фамилия и инициалы)

Руководитель _____ Сербин В.В.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Прончено Д.В
(подпись) (Фамилия и инициалы)

Аннотация

В данном дипломном проекта рассмотрена разработка прикладного приложения для операционной системы Android, являющего так называемым «Домашним экраном», которое представляет альтернативную системному часть интерфейса. Данное приложение может быть опубликовано в магазине мобильных приложений Google Play и полностью соответствует всем предъявляемым для публикации требованиям.

Кроме того, сделан анализ условий труда для разработки рассматриваемого приложения.

Также составлено экономическое обоснование проекта, подтверждающее его экономическую целесообразность.

Аңдатпа

Бұл дипломдық жобада «Үй экраны» деп аталатын, интерфейстің баламалы жүйесі түріндегі бөлігі ретінде Android операциялық жүйесі үшін қолданбалы қосымшаның әзірлемесі қарастырылды. Бұл қосымша Google Play мобильдік қосымшалар дүкенінде жариялануы мүмкін.

Сонымен қатар, қарастырылатын қосымшаны әзірлеу үшін еңбек жағдайларына талдау жасалды.

Сондай-ақ, жобаның экономикалық тиімділігін бекітетін экономикалық негіздемесі құрастырылды.

Annotation

In this diplom project examined the development of application software for the operating system Android, is the so-called "home screen", which is an alternative part of the system user interface. This application may be published in the mobile application store Google Play and meets all the requirements for publication requirements.

In addition, the analysis made working conditions for the development of the application under consideration.

Also composed the economic justification of the project, confirming its economic viability.

Содержание

Введение

Актуальность темы исследования. В современном мире, в так называемой «Пост-ПК эре», рынок мобильных операционных систем (далее ОС) достигает невероятных размеров и растет с каждым годом. Смартфоны все глубже проникают в нашу жизнь, охватывая все большую аудиторию пользователей и объем их продаж увеличивается с невероятной скоростью. Не последнюю роль в этом играет появления все большего количества недорогих аппаратов, предлагающих большие возможности, хоть и немного уступающих среднему и высокому ценовому сегментам. Однако, их качество улучшается с каждым «поколением», а цена же, наоборот, падает.

Основными игроками на рынке, на текущий момент, являются компании Apple и Google, со своими мобильными операционными системами iOS и Android, соответственно. По состоянию на ноябрь 2013 года их доли составляют 81% для Android и 12,9% для iOS, по данным компании IDC. Всего за исследуемый квартал было отгружено 261,1 млн. штук смартфонов. Замыкает тройку лидеров Windows Phone от компании Microsoft, которой принадлежит 3,6% рынка.

Все продажи приложений для данных ОС осуществляются через специальные магазины, содержащиеся держателями платформ: Google Play (Android, Google), App Store (iOS, Apple) и Market Place (Windows Phone, Microsoft). В связи с крайне низким количеством устройств на рынке, а также малыми продажами, магазин приложений и мобильная платформа от Microsoft далее в данной работе рассматриваться не будут.

Несмотря на доминирующую позицию Android на рынке устройств, App Store для iOS остается более доходной площадкой для разработчиков. Так, по данным исследовательской компании App Annie, за первый квартал 2014 года, суммарных доход разработчиков, разместивших свои продукты в магазине Apple, превышает на 85% таковой в Google Play. Количество же загрузок наоборот, на стороне Google Play и на 45% больше, чем у App Store. Данная ситуация складывается по причине того, что iOS распространяется исключительно самой Apple, которая выпускает по 1 устройству одного класса в год («топовый» смартфон, «бюджетный» смартфон, 7-дюймовый и 10-дюймовый планшеты), относящиеся к высокому ценовому сегменту. Устройства на Android же производятся сторонними производителями, в совершенно разных ценовых категориях. В связи с большим количеством недорогих смартфонов, операционная система от Google больше распространена в развивающихся регионах, где он безоговорочно лидирует по продажам. В высоком ценовом сегменте устройств лидерство Android не столь очевидное, в некоторых развитых странах количество «гаджетов», работающих на iOS от Apple и вовсе превосходит количество таковых на операционной системе от Google.

Несмотря на все вышеизложенное, для разрабатываемого в данном проекте приложения платформа от Google, а именно Android, является приоритетной, так как данное прикладное приложение (далее ПП) относится к категории приложений для изменения интерфейса системы и как следствие может быть без труда опубликовано в Google Play. Для запуска аналогичного ПО на устройствах с iOS необходим jailbreak, в связи с чем оно не может быть размещено в App Store.

Разрабатываемое ПП заменяет стандартный, предустановленный домашний экран и является основной частью интерфейса системы для запуска установленных приложений, позволяет изменять и настраивать интерфейс ОС под нужды пользователя, придерживаясь при этом рекомендаций разработчикам от Google (далее «гайдлайны») и сохраняя основной опыт использования оригинального, для Android Kit Kat, домашнего экрана – «Google Старт».

Данный тип ПП входит в категорию «Персонализация» в Google Play, наравне с виджетами, обоями для рабочего стола, экранами блокировки, модификациями для системы уведомлений, сборниками значков для приложений, шрифтами, а также готовыми темами для систем имеющих в своем составе приложение «темы» (например, Cyanogenmod, Android Open Kang Project, Paranoid Android, OmniRom и т.д.).

Данный тип приложений занимает лидирующие позиции в магазине приложений и делит топ с играми, мультимедиа приложения и утилитами. Самыми успешными аналогами разрабатываемого ПП являются GO launcher EX, среди бесплатных, с количеством установок от 100 000 000 до 500 000 000 и Nova Launcher Prime с количеством установок от 500 000 до 1 000 000, при цене в 4$ США. Самым дорогим же домашним экраном является Next Launcher 3D от GO Launcher Dev Team, с количеством установок равным от 100 000 до 500 000. Также среди представителей данной категории программ стоит отметить такие продукты как: Action Launcher, Apex Launcher, Mi Launcher, а также измененные версии домашнего экрана из Android Open Source Project (далее AOSP) для неофициальных прошивок Cyanogenmod и OmniRom.

Целью данного проекта является изучение разработки ПП для мобильных ОС на примере приложения для Android.

1 Принципы разработки

1.1 Операционная система Android

Android – ОС для смартфонов, планшетов и нетбуков. Компания Google приобрела разработчика программного обеспечения Android inc. в 2005 году. Операционная система Android основана на модифицированном ядре Linux и собственной реализации Java от Google.

Приложения для ОС Android включают в себя java-приложения и библиотеки, которые запускаются виртуальной машиной Dalvik с JIT компилятором. Сами приложения устанавливаются в виде файла формата .APK, который по своей сути является архивом с не скомпилированными компонентами приложения, задача же виртуальной машины Dalvik – компиляция кода приложения, которая происходит непосредственно во время запуска ПП. Библиотеки включают в себя систему управления, графику OpenGL ES 2.0, движок WebKit, графический движок SGL, SSL и библиотеки Bionic.

С выпуском последней версии Android Kit Kat в тестовом режиме, доступном только в меню для разработки появилась возможность выбора альтернативной Dalvik среды запуска приложений ART (Android Runtime), которая имеет ряд преимуществ перед Dalvik. Основное достоинство ART заключается в компиляции приложения непосредственно во время его установки, что значительно снижает нагрузку на центральный процессор во время его выполнения. Однако у данного метода есть и свои отрицательные стороны, одной из которых является увеличение размера занимаемой памяти устройства приложением, а также неправильное компилирование некоторых ПП, в результате чего требуется их адаптация разработчиками. Основным же недостатком новой среды является значительное увеличение времени установки приложения в систему, а также необходимость повторной их компиляции после каждого обновления системы. Так, активно используемое устройство, при после обновления ОС, может загружаться свыше часа. Помимо этого, в новой версии ОС была добавленная поддержка OpenGL ES 3.0, что позволяет создавать невероятной красоты объемные объекты, сравнимые по качеству изображения с современными игровыми консолями.

Разработчикам Google предлагает для свободного скачивания инструментарий для разработки (Android SDK), который предназначен для x86-машин под операционными системами Windows (XP или выше), Mac OS X (10.4.8 или выше) и Linux. Для разработки требуется также наличие установленного Java Development Kit (JDK) версии 5 – 7, версия 8 на данный момент не поддерживается. Помимо этого, существуют плагины для Eclipse – «Android Development Tools» (ADT), предназначенный для Eclipse версий 3.3 – 3.5, плагин для IntelliJ IDEA. А также предлагается официальная среда

разработки Android Studio, которая является альтернативой для вышеперечисленных, основанная на IntelliJ IDEA.

На рисунке 1.1 представлена архитектура ОС Android.

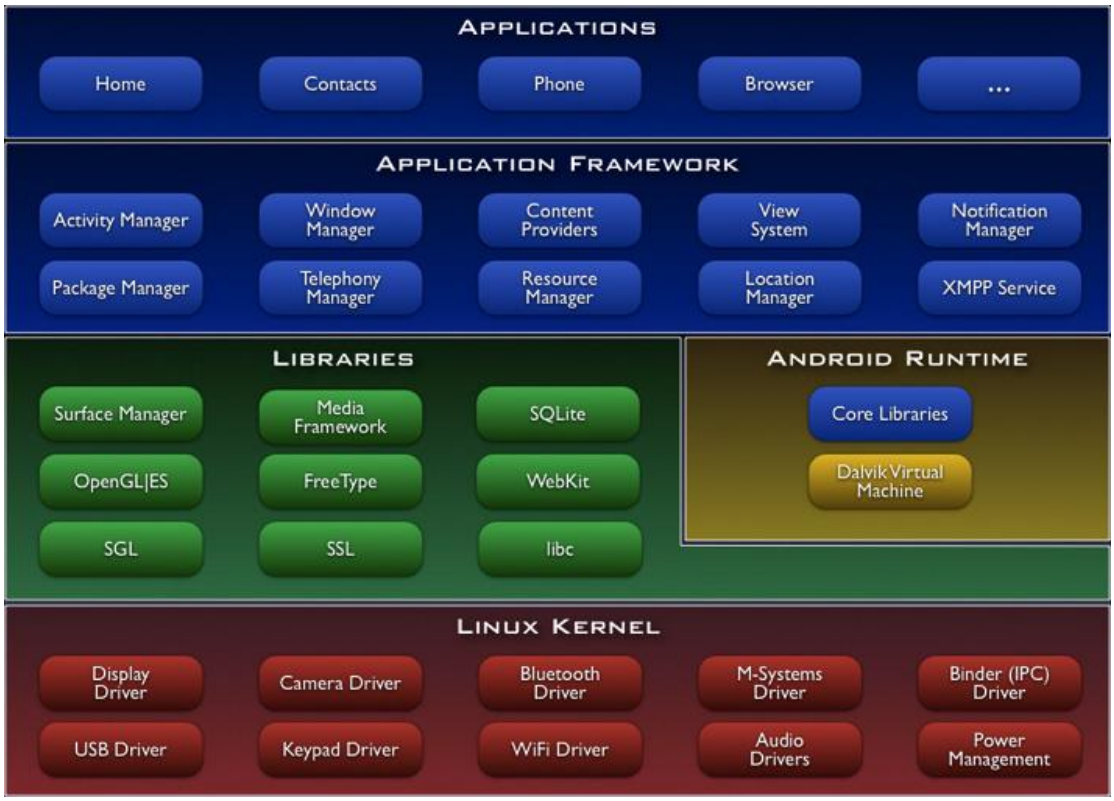

Рисунок 1.1 – Архитектура ОС Android

На основе рисунка 1.1 можно выделить следующие уровни ОС Android:

– Уровень приложений (Applications) – в состав Android входит комплект базовых приложений: клиенты электронной почты и SMS, календарь, различные карты, браузер, программа для управления контактами и много другое. Все приложения, запускаемые на платформе Android написаны на языке Java.

– Уровень каркаса приложений (Application Framework) – Android позволяет использовать всю мощь API, используемого в приложениях ядра. Архитектура построена таким образом, что любое приложение может использовать уже реализованные возможности другого приложения при условии, что последнее откроет доступ на использование своей функциональности. Таким образом, архитектура реализует принцип многократного использования компонентов ОС и приложений. Основой всех приложений является набор систем и служб:

- Система представлений (View System) – это богатый набор представлений с расширяемой функциональностью, который служит для построения внешнего вида приложений, включающий такие компоненты, как списки, таблицы, поля ввода, кнопки и т.п.

- Контент-провайдеры (Content Providers) – это службы, которые позволяют приложениям получать доступ к данным других приложений, а также предоставлять доступ к своим данным.
- Менеджер ресурсов (Resource Manager) предназначен для доступа к строковым, графическим и другим типам ресурсов.
- Менеджер извещений (Notification Manager) позволяет любому приложению отображать пользовательские уведомления в строке статуса.
- Менеджер действий (Activity Manager) управляет жизненным циклом приложений и предоставляет систему навигации по истории работы с действиями.

– Уровень библиотек (Libraries) – платформа Android включает набор C/C++ библиотек, используемых различными компонентами ОС. Для разработчиков доступ к функциям этих библиотек реализован через использование Application Framework. Ниже представлены некоторые из них:

- System C library – BSD-реализация стандартной системной библиотеки C (libc) для встраиваемых устройств, основанных на Linux.
- Media Libraries – библиотеки, основанные на PacketVideo‘s OpenCORE, предназначенные для поддержки проигрывания и записи популярных аудио- и видео- форматов (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG и т.п.).
- Surface Manager – менеджер поверхностей управляет доступом к подсистеме отображения 2D- и 3D- графических слоев.
- LibWebCore – современный движок web-браузера, который предоставляет всю мощь встроенного Android-браузера.
- SGL – движок для работы с 2D-графикой.
- 3D libraries – движок для работы с 3D-графикой, основанный на OpenGL ES 1.0 API.
- FreeType – библиотека, предназначенная для работы со шрифтами.
- SQLite – мощный легковесный движок для работы с реляционными БД.

– Уровень среды исполнения (Android Runtime) – в состав Android входит набор библиотек ядра, которые предоставляют большую часть функциональности библиотек ядра языка Java. Платформа использует оптимизированную, регистр-ориентированную виртуальную машину Dalvik, в отличии от нее стандартная виртуальная машина Java – стек-ориентированная. Каждое приложение запускается в своем собственном процессе, со своим собственным экземпляром виртуальной машины. Dalvik использует формат Dalvik Executable (*.dex), оптимизированный для минимального использования памяти приложением. Это обеспечивается такими базовыми функциями ядра Linux, как организация поточной обработки и низкоуровневое управление

памятью. Байт-код Java, на котором написаны ваши приложения, компилируются в dex-формат при помощи утилиты dx, входящей в состав SDK.

– Уровень ядра Linux (Linux Kernel) – Android основан на ОС Linux версии 2.6, тем самым платформе доступны системные службы ядра, такие как управление памятью и процессами, обеспечение безопасности, работа с сетью и драйверами. Также ядро служит слоем абстракции между аппаратным и программным обеспечением.

Для разработчиков доступные следующие библиотеки:

– Bionic – библиотека стандартных функций, несовместимая с libc.

– SSL – шифрование.

– Media Framework (PacketVideo OpenCORE, MPEG4, H.264, MP3, AAC, AMR, JPG, PNG).

– Surface Manager.

– LibWebCore (на базе WebKit).

– SGL – 2D-графика.

– OpenGL ES – 3D-библиотека.

– FreeType – шрифты.

– SQLite – легковесная СУБД.

По сравнению с обычными приложениями Linux, приложения Android подчиняются дополнительным правилам:

– Content Providers – обмен данными между приложениями.

– Resource Manager – доступ к таким ресурсам, как файлы XML, PNG, JPEG.

– Notification Manager – доступ к строке состояния.

– Activity Manager – управление активными приложениями.

1.2 Правила и рекомендации для разработки приложений

Как и большинство других платформодержателей, Google выдвигает определенные требования к разработчикам приложений, которые должны быть выполнены для обеспечения работоспособности продукта, а также публикации в Google Play. Помимо этого, имеются некоторые рекомендации по дизайну и организации интерфейса приложений, для получения лаконичного, удобного и не выбивающегося из общего стиле системы пользовательского интерфейса (User Interface, UI). Со всеми ими можно ознакомится на официальных профильных сайтах компании Google.

1.2.1 Разработка интерфейса

Все требования и рекомендации к внешнему виду приложений изложена в разделе «Проектирование», который состоит из 6 разделов:

– Начать (Get Started).

– Стиль (Style).

– Шаблоны (Patterns).
– Строительные блоки (Building Blocks).
– Загрузки (Downloads).
– Видео (Videos).

В свою очередь раздел «Начать» поделен на 3 части. Первая часть носит название «Creative Vision», и не несет в себе конструктивной информации, здесь разработчика подталкивают к тому, чтобы интерфейс его ПП был максимально красив, удобен и интуитивно понятен. Чтобы приложение было оригинально, помогало решать поставленную задачу быстро и легко.

Вторая часть называется «Design Principles» – принципы дизайна или принципы проектирования. Здесь также содержаться довольное общие рекомендации для разработчика, с образцами на примерами последней версии Android 4.4 Kit Kat.

Третья часть именуется «UI Overview» (Обзор интерфейса) и описывает основные системные элементы интерфейса ОС, такие как панель навигации, шторка уведомлений и т.д.

Раздел «Стиль» уже содержит непосредственную информацию по оформлению интерфейса, какие цвета, шрифты, иконки рекомендуется использовать.

Раздел «Шаблоны» содержит информацию по оформлению различных элементов интерфейса, а также рекомендации о навигации в приложении и взаимодействии с ним.

Раздел «Строительные Блоки» несет в себе информацию о формах использующихся в приложении.

Раздел «Загрузки» позволяет загрузить базовые для ОС элементы оформления, а также предоставляет использующуюся в Android палитру.

Раздел «Видео» содержит ссылки на видео конференций для разработчиков и видеоуроки.

Оформление

На Android работают сотни миллионов телефонов, планшетов и других устройств с самыми разнообразными размерами экрана и форм-факторами. Используя преимущества гибкой графической системы Android, вы можете создавать приложения, которые изящно смотреться на различных устройствах, от маленьких телефонов до больших планшетов. На рисунке 1.2 представленные основные форм-факторы устройств.

Темы оформления – механизм Android для приведения приложений к единому стилю. Стиль определяет визуальные свойства элементов, которые составляют пользовательский интерфейс, такие как цвет, высота и размера шрифта. В целях закрепления единого стиля между всеми приложениями на платформе Android? Google предоставляет две системных темы, которые вы можете выбрать при создании приложений:

– Holo Light.

– Holo Dark.

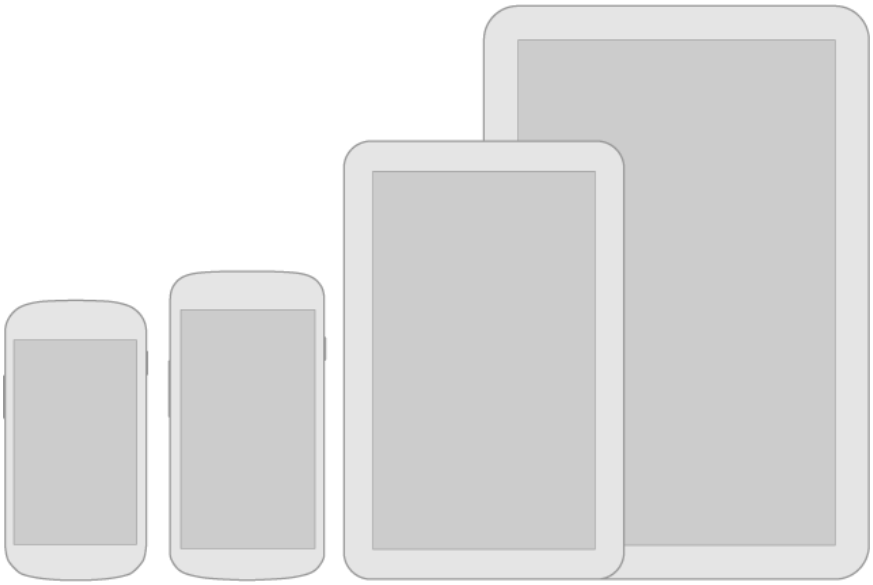

Рисунок 1.2 – Форм-факторы устройств

Применение этих помогут вам создавать приложения, которые соответствуют общему визуальному стилю Android.

Выберите тему, которая наиболее соответствует потребностям и эстетики дизайна для вашего приложения. Если вы желаете создать более оригинальный облик для вашего приложения, то используйте одну из системных тем в качестве отправной точки для вашей идеи.

Для отображения активности, отключённой кнопки или нажатия на нее используйте различные цвета, а также эффекты осветления и затемнения.

На рисунке 1.3 представлены стандартные шаблоны кнопок.

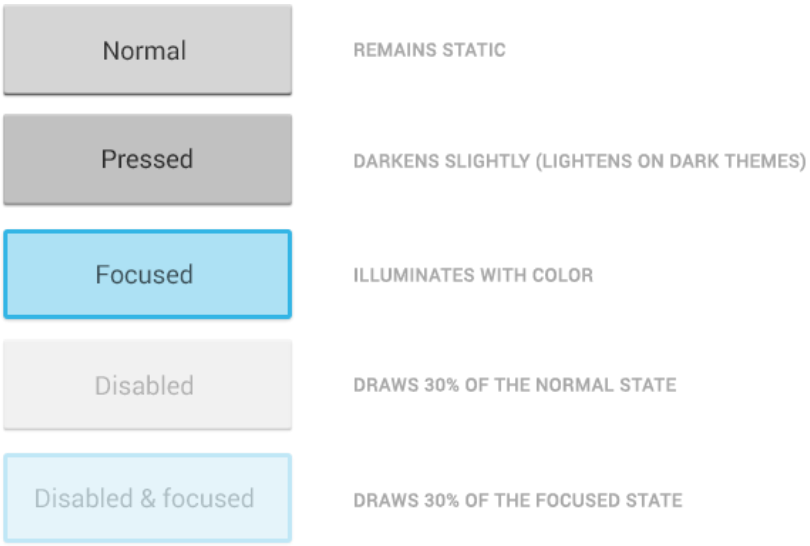

Рисунок 1.3 – Шаблоны кнопок

Используйте жесты любой сложности для навигации в приложении или же совершения каких-либо действий. На рисунке 1.4 изображен пример жеста.

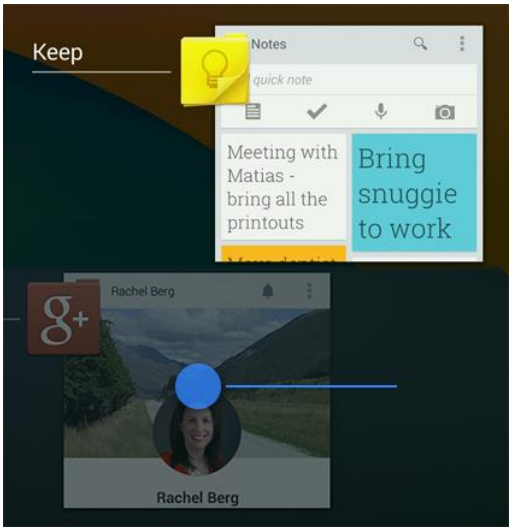

Рисунок 1.4 – Шаблон жестов

При использовании прокручиваемых элементов интерфейса, используйте эффект границы экрана. На рисунке 1.5 продемонстрирован шаблон границы экрана.

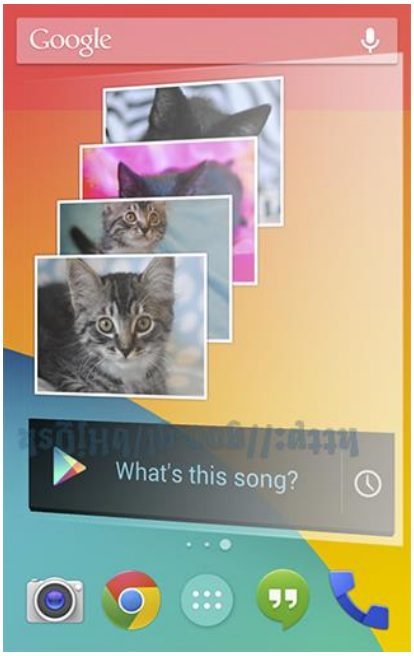

Рисунок 1.5 – Шаблон границы экрана

Устройства различаются не только по физическими размерами экранов, но и плотностью пикселей на нем (DPI). Для упрощения работы дизайнеров существуют несколько заготовленных образцов для телефонов и планшетов,

благодаря чему вам не нужно беспокоиться, как будет выглядеть ваше приложения на каждом отдельном устройстве.

Существуют DPI различных размеров, для простоты они поименованы LDPI, MDPI, HDPI, XHDPI, XXHDPI и XXXHDPI, и изначально в прошивке каждого устройства прописан соответствующий ему уровень плотности.

Необходимо всегда учитывать данные макеты экрана, так как в противном случае ваше приложение может исказиться и полностью испортиться интерфейс ПП, а также впечатление о нем. Однако это не составит никакого труда, потому что все шаблоны уже заложения в инструментарий для разработчиков, который автоматически подгонит интерфейс под требуемую плотность, если же вас что-то будет не устраивать, то вы непосредственно в процессе разработки, в графическом видео сможете все исправить и организовать так, как вам требуется.

На рисунке 1.6 представлены возможные варианты плотности экрана.

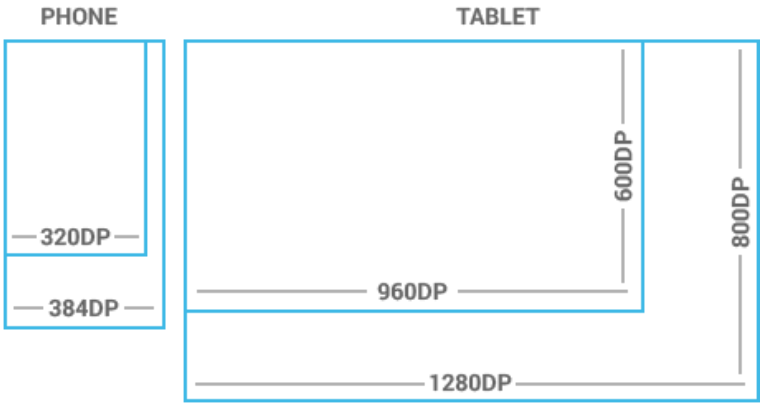

Рисунок 1.6 – Макеты плотности экрана

Все элементы интерфейса выполняются в размере кратном 48dp, то есть от 48x48 dp и выше. Данный размер соответствует примерно 9мм. на экране и является оптимальным по удобству. На рисунках 1.7 и 1.8 отображены рекомендуемые размеры для элементов интерфейса.

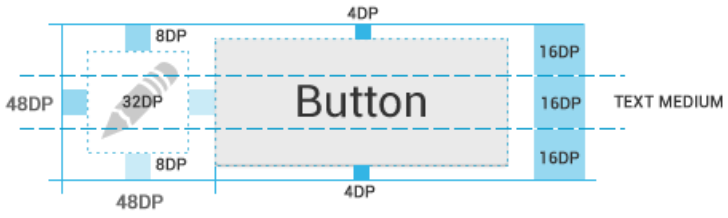

Рисунок 1.7 – Пример значка и кнопки

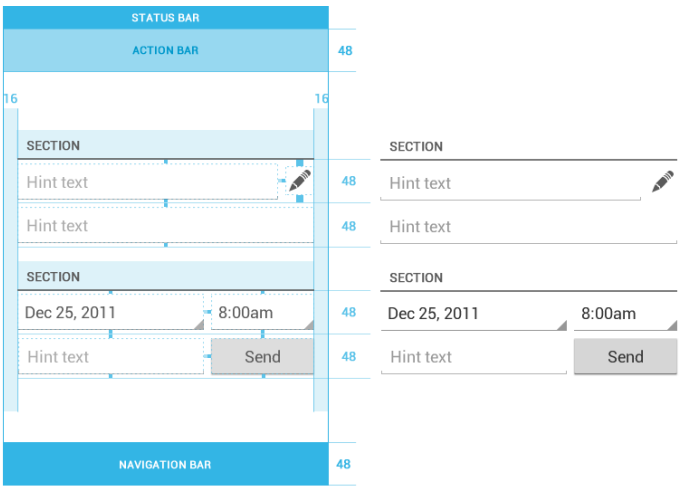

Рисунок 1.8 – Пример элементов размером 48dp в интерфейсе

В качестве системного шрифта в Android версии 4+ используется шрифт Roboto различных видов (Roboto Thin, Roboto Light, Roboto Regular, Roboto Medium, Roboto Bold, Roboto Black, Roboto Condensed Light, Roboto Condensed и Roboto Condensed Bold). Начиная с Android 4.4 используется модификация Roboto Condensed и ее виды.

Android UI использует следующие цветовые стили по умолчанию: textColorPrimary и textColorSecondary. Для легких тем используется textColorPrimaryInverse и textColorSecondaryInverse. Рамки, используемые вокруг текста также поддерживают светлую и темную темы, а также имеют графическую обратную связь при нажатии. Размер же варьируется от 12sp до 22sp. На рисунке 1.9 изображены примеры светлой и темной тем текста.

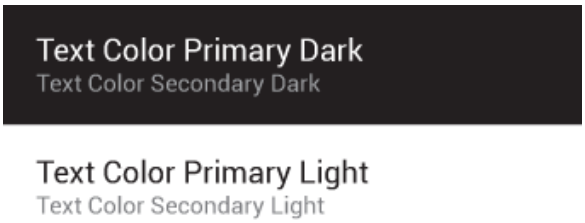

Рисунок 1.9 – Примеры светлого и темного текстов

При работе с цветом предлагается использование так называемых «Holo-цветов», которая изображена на рисунке 1.10.

Значок (иконка) приложения – небольшой графический ярлык расположенный в меню приложений или же вынесенный на один рабочих столов домашнего экрана, служащий для запуска приложения, а также его графического отображения в системе. При проектировании значка необходимо учитывать, что ваше приложение может быть установлено на различных устройствах, с различной плотностью экрана, а также необходимо его отображение в магазине Google Play.

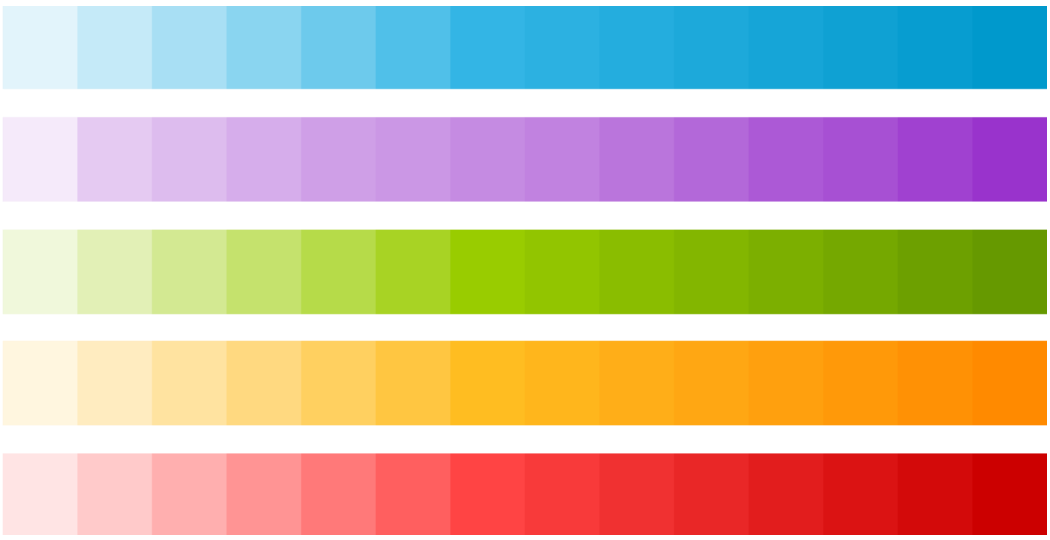

Рисунок 1.10 – Палитра цветов

Поэтому используя макеты плотности экрана создайте несколько иконок разных размеров, для разных экранов. За отправную точку следует брать экраны с MDPI плотностью, с дальнейшим увеличением в пропорция равных 1,5x для HDPI, 2x для XHDPI, 3x для XXHDPI и 4x для XXXHDPI. Исходный размер должен быть равен 48dp, в результате на экранах с плотность MDPI размер значка будет равен 48x48 точек, для экранов HDPI – 72x72 точек, для XHDPI – 96x96 и т.д. Для публикации в Google Play также требуется изображение размером 512x512 точек. На рисунке 1.11 приведены размеры значка приложения.

Рисунок 1.11 – Пример размеров значка

Для значков различных действий непосредственно в приложении рекомендуется использование стандартных Holo-иконок, предоставляемых Google в векторном формате для графического редактора Adobe Illustrator. Физический размер таких элементов должен составлять 32x32 точек, а видимая часть 24x24.

Цвета также рекомендуется использовать стандартные. Более светлый (#333333) для темного фона, с прозрачностью 40% и более темный (#FFFFFF) для светлого фона, с прозрачностью 20%.

На рисунке 1.12 показаны примеры иконок действий и их размер.

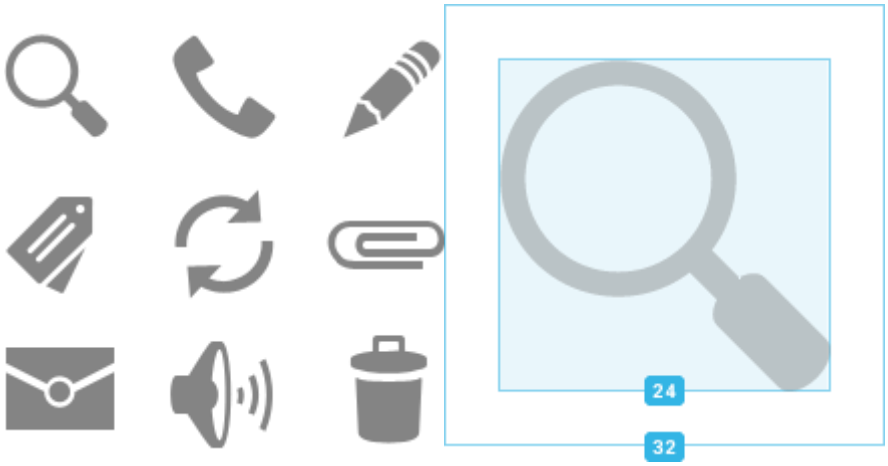

Рисунок 1.12 – Примеры иконок для действий

При использовании контекстных иконок, следует использовать физический размер равный 16x16 с видимой частью 12x12, как показано на рисунке 1.13.

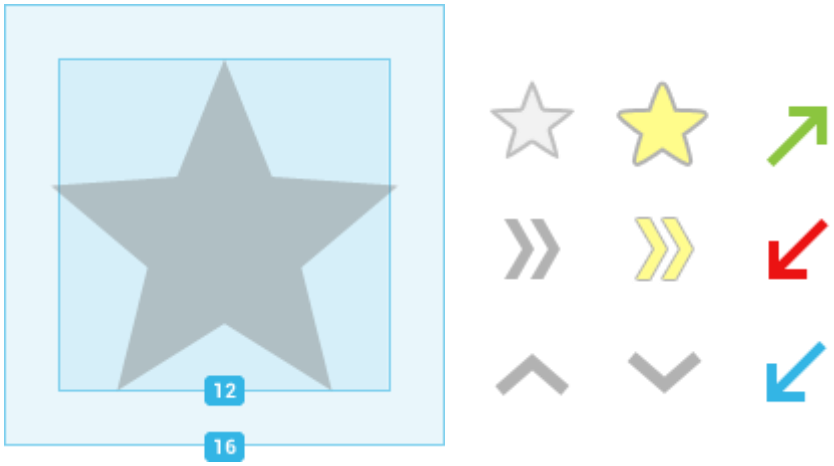

Рисунок 1.13 – Пример контекстной иконки

Если ваше приложение использует систему уведомлений, то для отображения в шторке уведомления требуемой информации, также нужен значок размером 24x24 белого цвета.

При создании элементов интерфейса рекомендуется использование векторных элементов там, где это возможно, с максимальным избеганием растровых изображений. Это поможет избежать проблем с оптимизацией для различных размеров экранов и различной плотности.

При названии файлов лучше использовать префиксы, тогда иконки будут группироваться внутри директории и их поиск существенно облегчиться.

В коде программы также лучше разбить все элементы интерфейса на отдельные группы, что снизит затраты времени в случае необходимости что-либо исправить.

Например:

```
res /...
    drawable – ldpi /...
        finished_asset . png
    drawable – mdpi /...
        finished_asset . png
    drawable – hdpi /...
        finished_asset . png
    drawable – xhdpi /...
        finished_asset . png
```

При использовании текста в интерфейсе, описании его элементов и прочего, старайтесь максимально сократить его размер и упросить его восприятие. Избегайте использования ненужных слов, несущих не требующуюся информацию, сокращайте размер текста используя цифры вместо букв. Однако ненужно переусердствовать, если данный текст не отражает всю суть информации, которую требуется донести, то лучше сделать его более объемным, но пользователь полностью поймет, что от него хотят.

1.2.2 Принципы навигации

Главным элементом управления с современных смартфонах и планшетах является сенсорный экран. У ОС Android есть набор жестов, на которые особым образом реагирует система. Некоторые из них зарезервированы системой, такие как нажатие кнопки «домой» на навигационной панели, другие же зависят от приложения, запущенного в данный момент. Однако для во избежание путаницы пользователя рекомендуется использование стандартных действий и жестов, работающих в системе.

Так, например, одиночное, кратковременное нажатие активирует элемент, на который направленно данное действие. Длительное удержание активирует выбор объекта или нескольких. Длительное нажатие и движение пальца по экрану – перемещение объекта. «Свайп» (сдвиг) в сторону вызывает переключение на соседнюю вкладку. Двойное нажатие на активирует увеличение масштаба в 2 раза (например, в галереи, при просмотре изображений), а «Свайп» вверх или вниз вызывает пролистывание списка. Разведение или сведение двух пальцев вызывает увеличение, или уменьшение изображения соответственно.

Возможные жесты показаны на рисунке 1.14.

Рисунок 1.14 – Пример жестов

Построение правильно навигации – одна из главных задач и проектировании интерфейса ПП, так как она должна полностью совпадать с навигацией в ОС и не вводить пользователя в положение, когда она не знает, что вызовет нажатие той или иной кнопки.

Должное внимание при этом следует уделить клавише «назад». Она должна выполнять следующие функции:

– Если пользователь находится на главном экране приложения, то нажатие клавиши «назад» вызовет закрытие приложения.

– Если пользователь находится в какой-либо вкладке ПП, на 1 уровень, то нажатие клавиши возвращает его на главный экран приложения.

– Если пользователь последовательно перешел на несколько уровней, то нажатие клавиши возвращает его на предыдущий уровень.

На рисунках 1.15 – 1.20 продемонстрированы основные правила навигации для приложений Android.

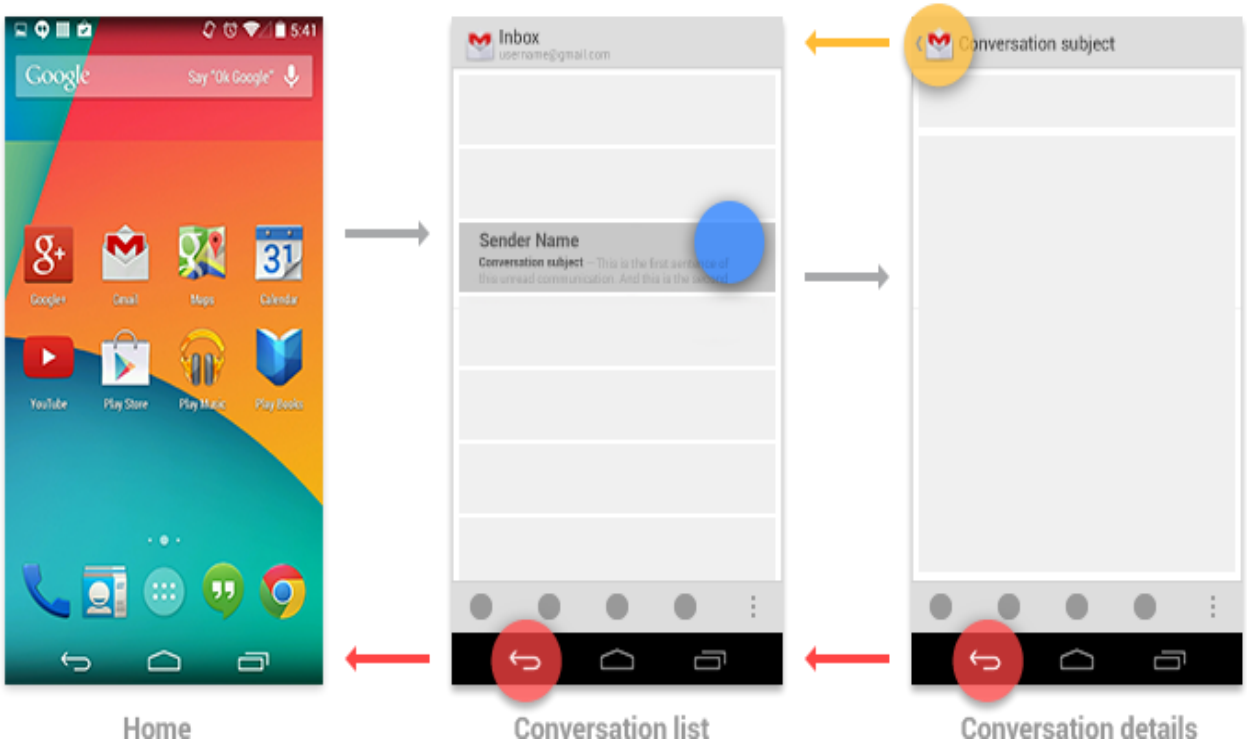

Рисунок 1.15 – Нажатие клавиши «назад» при нахождении на 1 уровне

В случае моментального необходимости возврата на главный экран приложения, требуется нажать на его иконку, сверху, слева.

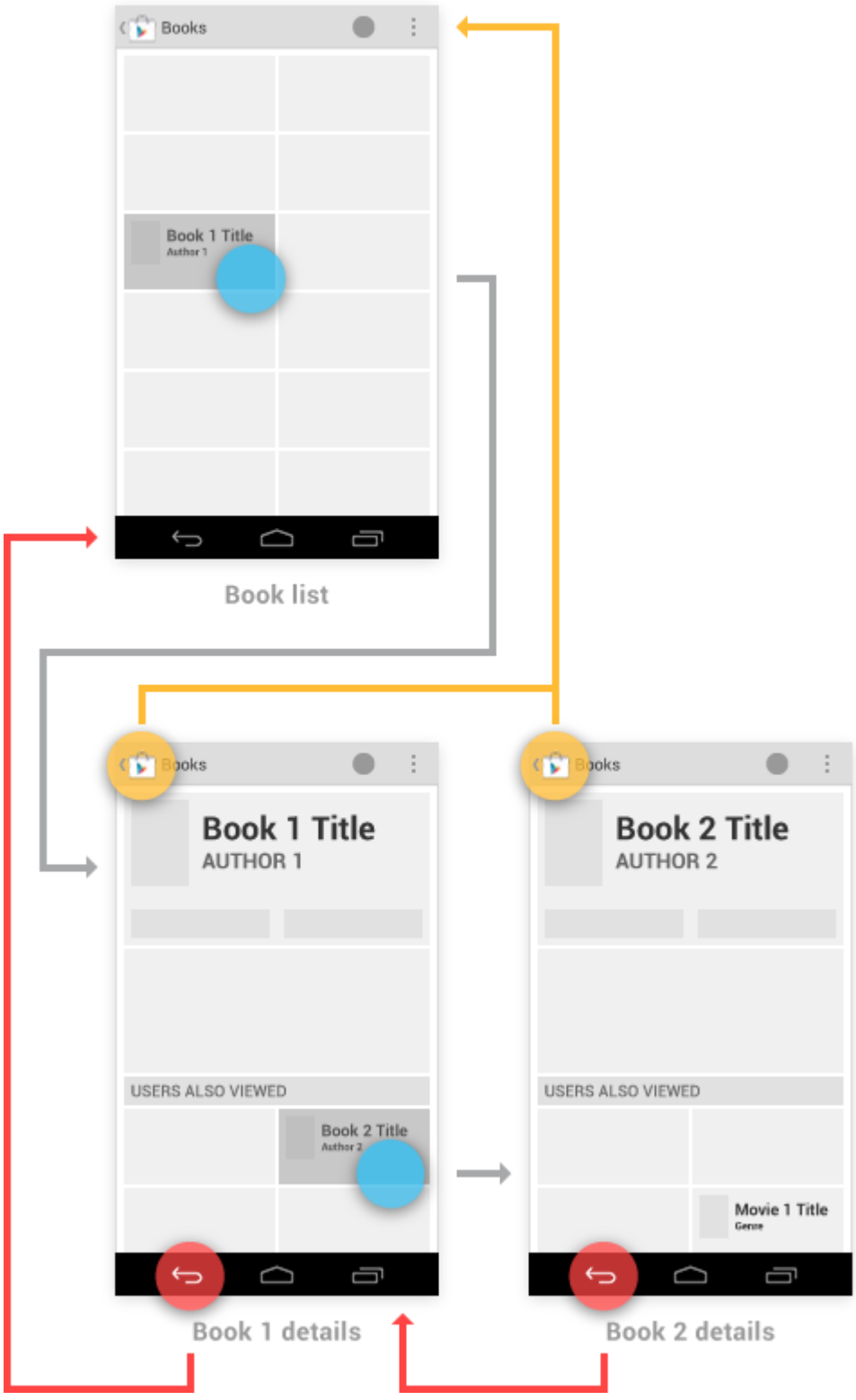

Рисунок 1.16 – Возврат к главному экрану

В случае, если у приложения несколько главных экранов (вкладок), то обходимо организовать динамическую клавишу, находящуюся сверху, слева.

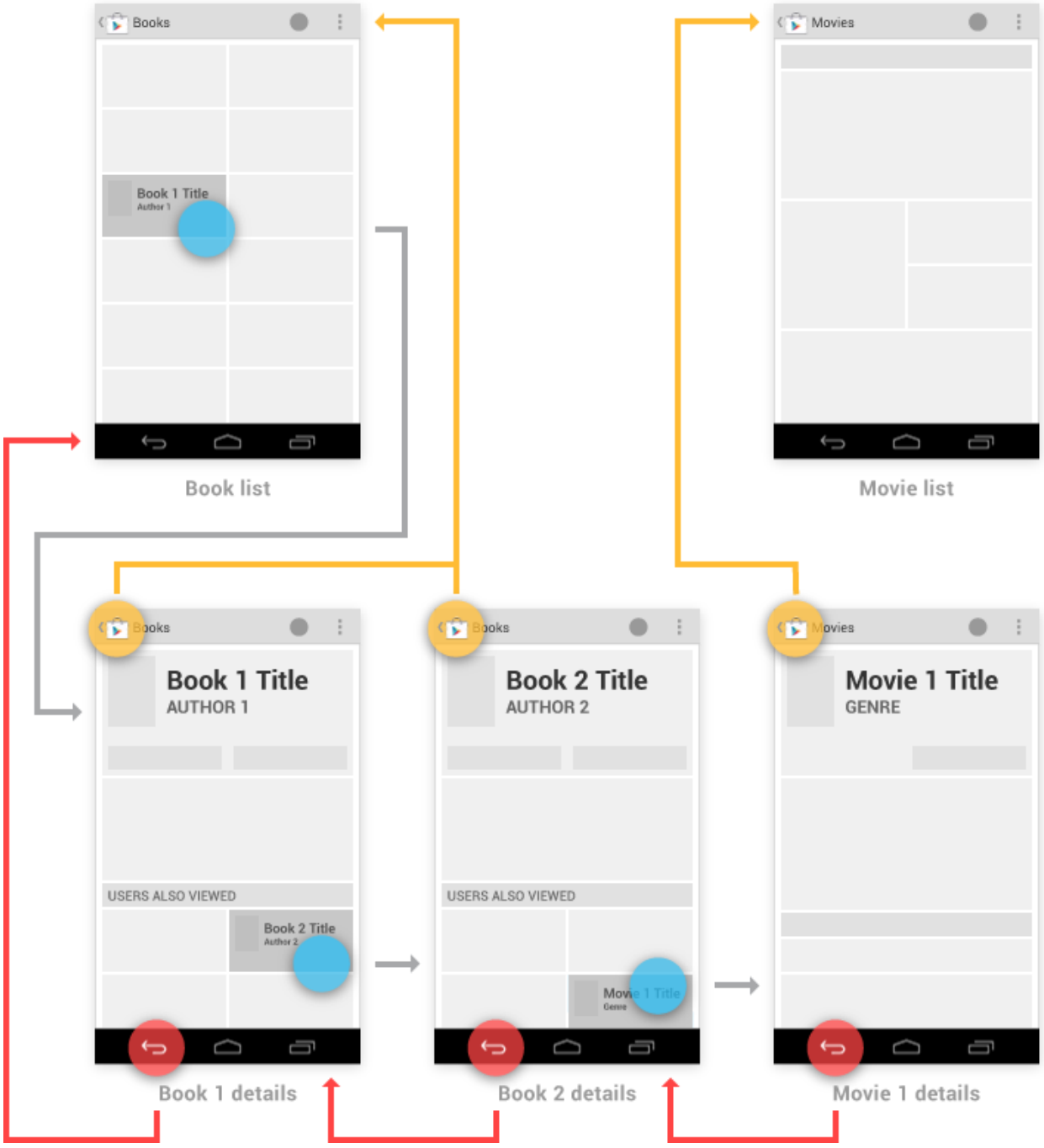

Рисунок 1.17 – Возврат к разным главным экранам

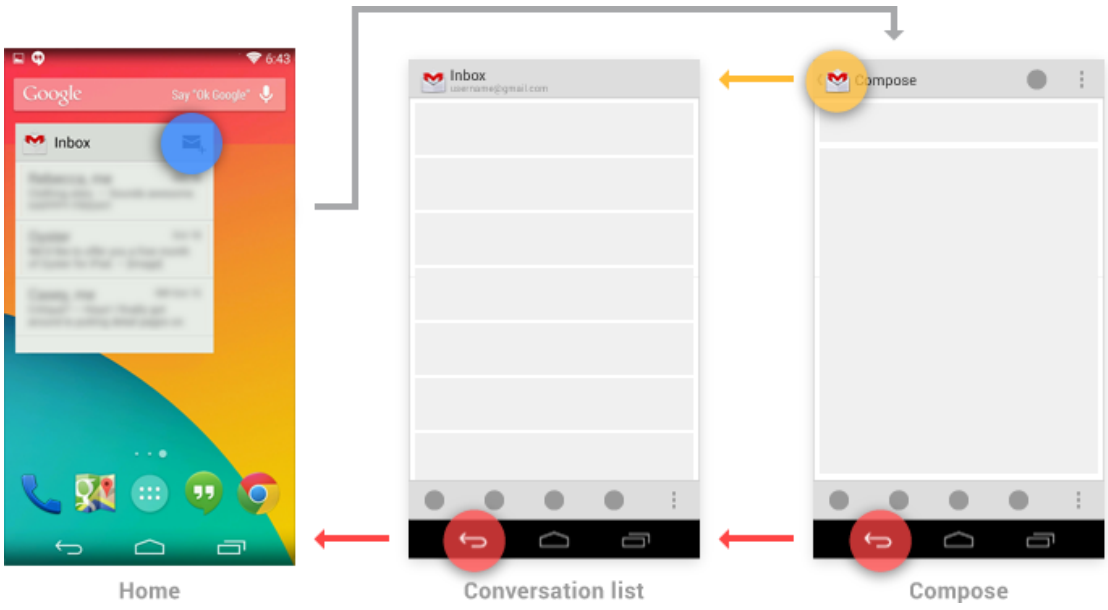

Рисунок 1.18 – Возврат после активизации виджета

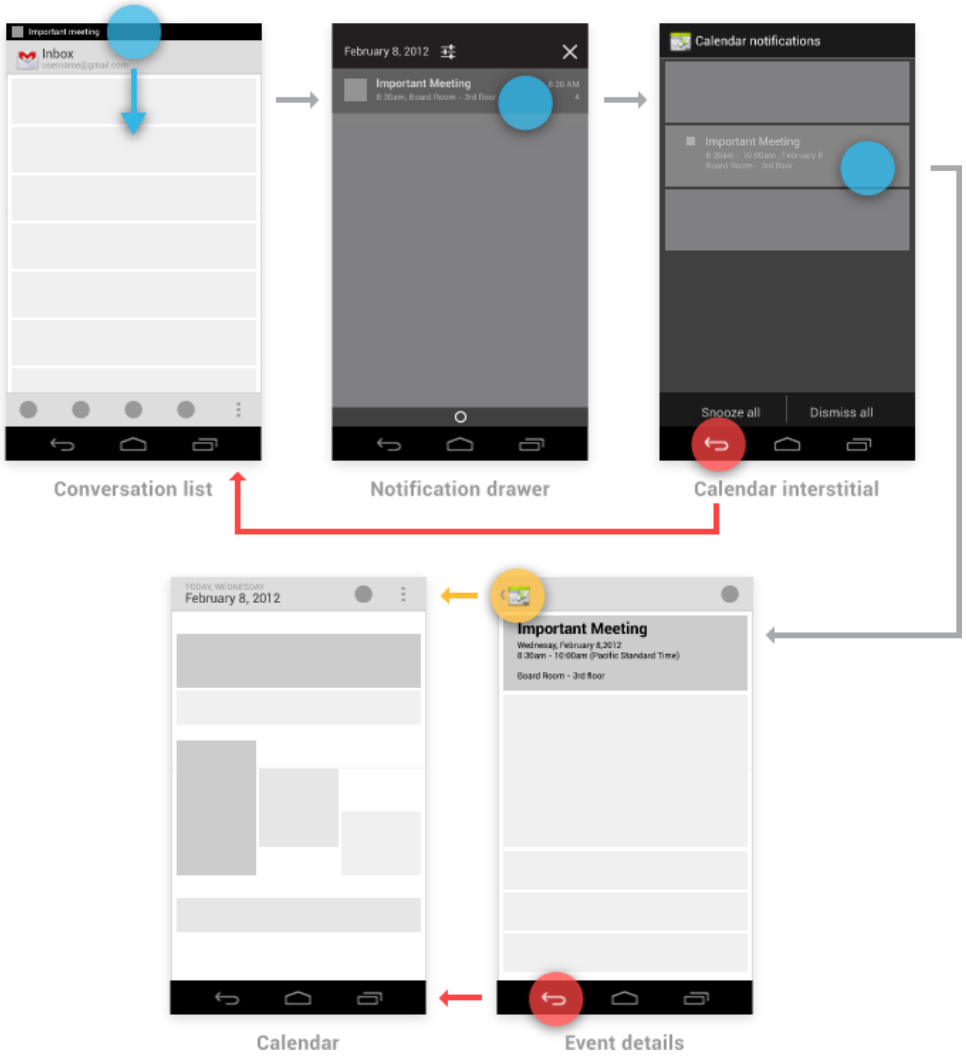

Рисунок 1.19 – Работа клавиши «назад» со шторкой уведомлений

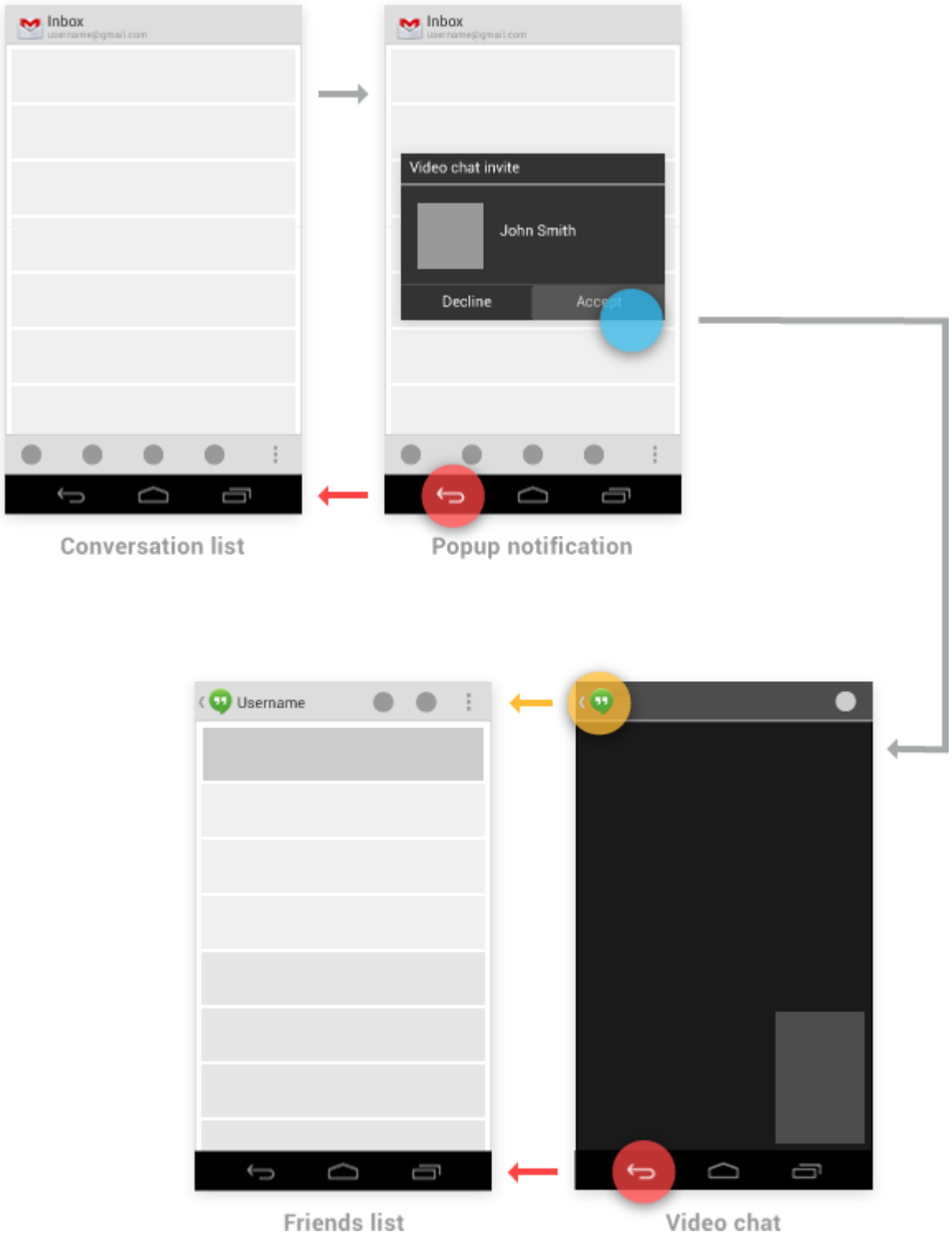

Рисунок 1.20 – Работа клавиши «назад» с всплывающими уведомлениями

При переходе из приложения «А» по ссылке в приложение «Б», после завершения действий в приложении «Б», оно автоматически свернется, и пользователь вернется к приложению «А». В случае отмены действия, нажатие клавиши «назад» работает также, как и обычно, за исключением того момента, когда вы достигните главной вкладки приложение «Б», дальнейшее нажатие клавиши «назад» вызовет закрытие приложения «Б» и возврат к «А».

По причине слишком большого объема рекомендаций и правил приведенных компанией Google для разработчиков, дальнейшие их части, касающиеся дизайна приложений и организации пользовательского интерфейса, невозможно расписать в данном описании проекта. Ссылка на полную базу «гайдлайнов» от компании Google для приложений на базе ОС Android: http://developer.android.com/intl/ru/design/index.html.

1.2.3 Основные принципы работы приложений

Приложения для Android пишутся на языке программирования Java. Android SDK позволяет компилировать весь код, вместе с данными и всеми ресурсами в файлы формата .APK – Android package, который является архивный файл расширением .APK. Эти файлы содержат в себе весь исходный приложения и используются устройствами на базе ОС Android для установки данного APK.

После установки на устройство, каждое Android-приложение «живет» в своем собственном, изолированном «security sandbox»:

– ОС Android – является многопользовательской системой Linux, в которой каждое приложение является отдельным пользователем.

– По умолчанию система присваивает каждому приложению уникальный идентификатор пользователя Linux (ID используется только системой и неизвестен приложению). Система устанавливает только те права доступа, которые советуют идентификатору пользователя, присвоенный этому приложению.

– Каждый процесс использует свою виртуальную машину (VM), поэтому код приложение в работает изоляционно от других приложений.

– По умолчанию каждое приложение выполняется как отдельный процесс в Linux. Android запускает данный процесс, только тогда, когда необходимо выполнение какого-либо компонента приложения, когда же необходимость в нем пропадает, то система автоматически завершает процесс. Также завершение процесса происходит в случаи необходимости высвободить память под другие приложения.

Таким образом, Android система реализует принцип привилегий. То есть, каждое приложение, по умолчанию, имеет доступ только к тем компонентам, которые требуются для его работоспособности и не более того. Это создает безопасную среду, в которой приложение не может получить доступ к части системы, для которой она не предоставила разрешений.

Тем не менее, есть пути для приложений, для обмена данными с другими приложениями и для его доступа к системным сервисам:

– Это можно если оба взаимодействующих приложения имеют одинаковый идентификатор пользователя, в этом случае они в могут получить доступ к файлам друг друга. Чтобы сэкономить системные ресурсы, приложения с одинаковым идентификатором пользователя также могут быть

запущены в одном и том же процессе Linux на одной VM (Однако оба приложения должны иметь идентичную подпись сертификата).

– Приложение может запросить разрешение на доступ к данным устройства, такие как контакты пользователя, SMS сообщений, файлы на карте памяти (SD карты), камера, Bluetooth и многое другое. Все разрешения приложение должны быть предоставлены пользователем во время установки.

Выше перечислены основные принципы организации работы приложений в ОС Android. Далее будут рассмотрены следующие вопросы:

– Основные компоненты, необходимые для работы разрабатываемого приложения.

– Manifest file, в котором вы объявляются компоненты и необходимые функции устройства для вашего приложения.

– Ресурсы, которые возможно отделить от основного кода приложения, что позволит корректно оптимизировать ваше приложения для работы на устройствах различной конфигурации.

1.2.4 Компоненты приложений

Существует несколько стандартных компонентов приложений, блоков из которых они состоят. Каждый вид компонента выполняет свою функцию и их можно разделить на 4 вида:

– Activity.
– Intent Receiver.
– Service.
– Content Provider.

Не у каждого приложения должны быть все четыре блока, но Ваше приложение будет написано с их некоторой комбинацией.

Как только Вы решили, в каких компонентах Вы нуждаетесь для своего приложения, Вы должны перечислить их в файле по имени AndroidManifest.xml. Это – файл XML, где Вы объявляете компоненты своего приложения и каковы их возможности и требования.

Activity

Activity – самый распространенный из четырех стандартных блоков Андроида. Activity обычно – единственный экран в Вашем приложении. Каждый Activity осуществлен как единственный класс, который расширяет базовый класс Activity. Ваш класс отобразит пользовательский интерфейс, составленный из Views, и ответит на события. Большинство приложений состоит из множественных экранов. Например, у приложения обмена сообщениями мог бы быть один экран, который показывает список контактов, второй экран, чтобы написать сообщение выбранному контакту, и другие экраны, чтобы делать обзор старых сообщений или изменить настройку. Каждый из этих экранов был бы осуществлен как Activity. Перемещение в

другой экран достигнуто стартом нового Activity. В некоторых случаях Activity может возвратить значение предыдущего Activity – например Activity, которая позволяет пользователю выбирать фотографию, возвратил бы выбранную фотографию вызывающей программе. Когда новый экран открывается, предыдущий экран приостановлен и помещен на стек хронологии. Пользователь может переместиться назад через ранее открытые экраны в хронологии. Экраны могут также хотеть быть удаленными от стека хронологии, когда было бы неуместно для них остаться. Андроид сохраняет стеки хронологии для каждого приложения, начатого от начала экрана.

Intent и фильтры Intent

Андроид использует специальный класс под названием Intent, чтобы двигаться от экрана к экрану. Intent описывает то, что приложение собирается сделать. Две самых важных части структуры Intent – действие и данные к действию. Типичные значения для действия – MAIN (главный экран приложения), VIEW, PICK, EDIT, и т.д. Данные выражены как Uniform Resource Indicator (URI). Например, чтобы рассмотреть вебсайт в браузере, Вы создали бы Intent с действием VIEW и набором данных – адресом сайта:

```
new Intent(android.content.Intent.VIEW_ACTION,
ContentURI.create("http://anddev.org"));
```

Есть связанный класс, названный IntentFilter. В то время как Intent – запрос сделать кое-что, IntentFilter – описание того, что Intent Activity (или intent receiver, см. ниже), способен к обработке. Activity, который в состоянии отобразить информацию для человека, издала бы IntentFilter, который сказал, что знает, как обработать VIEW действия. Activity издает свой IntentFilters в файле AndroidManifest.xml.

Навигация от экрана к экрану достигнута достигается с помощью Intent. Чтобы переместиться вперед, Activity вызывает startActivity (myIntent). Система тогда смотрит на IntentFilter для всех установленных приложений и выбирает Activity, Intent которого фильтрует myIntent. Новому Activity сообщают о Intent, которое заставляет его начаться. Процесс решения Intent происходит, когда startActivity вызывают. Процесс предлагает две ключевых льготы:

– Действия могут многократно использовать функциональные возможности от других компонентов, просто делая запрос в форме Intent.

– Действия могут быть заменены в любое время новым Activity с эквивалентным IntentFilter.

Intent Receiver

Вы можете использовать IntentReceiver, когда Вы хотите, чтобы код в своем приложении выполнился в реакции на внешнее событие, например, когда телефон звонит, или когда сеть передачи данных доступна, или когда это – полночь. Intent Receiver не отображают UI, хотя они могут отобразить

Уведомления, чтобы привести пользователя в готовность, если кое-что интересное случилось. Поглощенные получатели также регистрированы в AndroidManifest.xml, но Вы можете также регистрировать их в коде, используя Context.registerReceiver(). Ваше приложение не должно работать для его Intent Receiver, которые вызываются; система запустит Ваше приложение, в случае необходимости, когда Intent Receiver будет вызван. Приложения могут также послать свои собственные Intent Receiver другим с Context.broadcastIntent().

Service

Service – код, который долговечен и выполняется без UI. Хороший пример этого – универсальный проигрыватель, запускающий песни из плейлиста. В приложении универсального проигрывателя, вероятно, были бы одно или более Activity, которые позволяют пользователю выбирать песни и запускать их. Однако, воспроизведение самой музыки не должно быть обработано Activity, потому что пользователь будет ожидать, что музыка продолжит играть даже после сворачивания проигрывателя. В этом случае, деятельность универсального проигрывателя могла запустить Service, используя Context.startService(), чтобы работать на заднем плане и сохранить воспроизведение музыки. Тогда система сохранит воспроизведение музыки, пока оно не закроется само. (Вы можете узнать больше о приоритете, данном службам в системе, читая Цикл Жизни Приложения Андроида). Отметьте, что Вы можете соединиться с Service (и запустить его, если он уже не работает) с методом Context.bindService(). Когда есть подключение с Service, Вы можете общаться с этим через интерфейс, выставленный Service. Для Service музыки это могло бы позволить Вам приостанавливать, перематывать, и т.д.

Content Provider

Приложения могут хранить свои данные в файлах, базе данных SQLite, персональных настройках или любом другом механизме, который имеет смысл. Content Provider, однако, полезен, если Вы хотите, чтобы данные Вашего приложения были разделены с другими приложениями. Content Provider – класс, который осуществляет стандартный набор методов, чтобы позволить другим приложениям сохранять и восстанавливать тип данных, которые обработаны другим(that) Content Provider.

1.2.5 Пользовательские интерфейсы в Android

Пользовательские интерфейсы (UI) в Android могут быть созданы двумя путями, через XML-код или в java-коде. Создание структуры графического интерфейса пользователя в XML очень предпочтительно, потому что по принципу Образцового управления средства просмотра, UI должен всегда отделяться от логики программы. К тому же, приспосабливание программы от одной разрешающей способности экрана до другой намного более просто.

Определение UI в XML очень похоже к созданию общего документа HTML, где вы имеете такой простой файл:

```
<html>
<head>
<title>Page Title</title>
</head>
<body>
The content of the body element.
</body>
</html>
```

Все равно как в Android XML-Layouts. Все хорошо структурировано и может быть выражено древовидными структурами:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World"/>
</LinearLayout>
```

Иерархия Элементов Экрана

Основной функциональный модуль приложения Android – Activity – объект класса android.app.activity. Activity может сделать много вещей, но отдельно у него нет присутствия на экране. Чтобы дать Вашему Activity присутствие на экрана и проектировать его UI, Вы работаете с Views и Viewgroups – основными единицами выражения пользовательского интерфейса на платформе Android.

Views

View – объект, расширяющий базовый класс android.view.view . Это – структура данных, свойства которой сохраняют Layouts и информационное наполнение для определенной прямоугольной области экрана. Объект View обрабатывает измерение, его схему размещения, рисунок, изменения центра, прокрутку, и клавиши/знаки для области экрана, которую он представляет. Класс View служит базовым классом для всех графических фрагментов – ряд полностью осуществленных подклассов, которые рисуют интерактивные элементы экрана. Графические фрагменты обрабатывают свое собственное измерение и рисунок, таким образом Вы можете использовать их, чтобы создать Ваш UI более быстро. Список доступных графических фрагментов включает TextView, EditText, Button, RadioButton, Checkbox, ScrollView и т.д.

Viewgroups

Viewgroup – объект класса android.view.viewgroup. Viewgroup – специальный тип объекта View, функция которого – содержать набором View и Viewgroup и управлять ими. Viewgroups позволяют Вам добавлять структуру к Вашему UI и создавать сложные элементы экрана, к которым можно обратиться как к единственному объекту. Класс Viewgroup служит базовым классом для Layouts – ряда полностью осуществленных подклассов, обеспечивающего общие типы Layouts экрана. Layouts дают Вам способ встроить структуру для ряда View.

UI с древовидной структурой

На платформе Android Вы определяете UI Activity использование дерева View и Viewgroup узлов, как показано в диаграмме ниже. Дерево может быть столь же простым или сложным, как Вы его сделаете, и Вы можете построить его, используя наборы предопределенных графических фрагментов и Layouts Android, или заказных типов View, которые Вы создаете самостоятельно. На рисунке 1.21 изображена древовидная система пользовательского интерфейса.

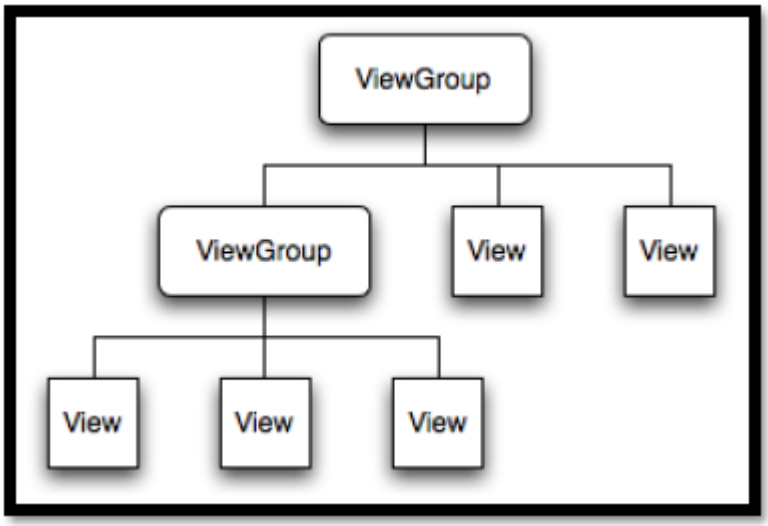

Рисунок 1.21 – UI ОС Android – древовидная структура

Чтобы прикрепить дерево к экрану и просчитать его, Ваш Activity вызывает свой метод setContentView() и передает информацию на корневой объект узла. Как только у система Android получает информацию на корневой объект узла, она начинает работать непосредственно с узлом, чтобы измерить, и просчитать дерево. Когда Ваш Activity становится активным и получает приоритет, система регистрирует Ваш Activity и просит корневой узел измерить и просчитать дерево. Тогда корневой узел просит, чтобы его дочерние вершины просчитывали себя – в свою очередь, каждый Viewgroup узел в дереве ответственен за просчет его прямых дочерних узлов. Как упомянуто ранее, у каждой группы View есть ответственность измерения ее доступного

пространства, расположения ее дочерних узлов, и вызов draw() на каждом дочернем узле, чтобы позволить все им просчитывать себя. Дочерние узлы могут просить размер и местоположение в родителе, но у родительского объекта есть конечное решение, где и насколько большой каждый ребенок может быть.

Сравнение Android Элементов UI с Swing Элементами UI

Поскольку некоторые разработчики, которые читают это, возможно, нашли, что UIs схож с Swing, сейчас будет немного общих черт между Андроидом и Swing:

- Activity в Android – почти (J) Frame в Swing.
- View в Android – (J) Component в Swing.
- TextViews в Android – (J) TextField в Swing.
- EditTexts в Android – (J) TextField в Swing.
- Button в Android – (J) Button в Swing.

Установка слушателей к View в Android является почти тем же самым, чем и в Swing:

```
// Android
myView.setOnClickListener(new OnClickListener(){ ...
// Swing
myButton.addActionListener(new ActionListener() {...
```

1.2.6 AndroidManifest.xml

AndroidManifest.xml – необходимый файл для каждого приложения Android. Он расположен в папке приложения и описывает глобальные значения для Вашего пакета, включая прикладные компоненты (действия, службы, и т.д.), который пакет выставляет ‘внешнему миру’, какие данные каждое из Activity приложения может обработать, и как они могут быть начаты.

Важная вещь к упоминанию об этом файле, свой так называемый IntentFilters. Эти фильтры описывают где и когда Activity может быть начат. Когда Activity (или операционная система) хочет выполнить действие, такое, как открыть Web-страницу или открыть экран выбора, это создает объект Intent. Этот Intent может хранить информацию, описывающую, что Вы хотите сделать, какие данные необходимы, чтобы достигнуть этого и другую информацию. Android сравнивает информацию в объекте Intent с IntentFilters, выставленным каждым приложением, и находит Activity, соответствующие, чтобы обработать данные или действия, определенные вызывающей программой. Если есть больше чем одно приложение, способное к обработке этого Intent, пользователя спрашивают, каким приложением он предпочел бы обрабатывать это.

Помимо объявления Activity, Content Provider, Service и Intent Receivers Вашего приложения, Вы можете также определить разрешения в AndroidManifest.xml.

Основное

Очень простой AndroidManifest.xml выглядит следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="org.anddev.android.hello_android">
<application android:icon="@drawable/icon">
<activity android:name=".Hello_Android"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

Почти каждый AndroidManifest.xml (так же как многие другой Android файлы XML) будет включать объявление пространства имен в его первый элемент. Это делает множество стандартных атрибутов Android доступным в файле, который будет использоваться, чтобы снабдить большинством данных для элементов в том файле.

Почти каждый AndroidManifest.xml включает единственный <application> тэг, который непосредственно содержит несколько тэгов, описывающих Applications, IntentReceivers, и т.д., которые доступны в этом приложении.

Если Вы хотите сделать Activity запускаемым непосредственно через пользователя, то Вы должны будете прописать ему поддержку действия MAIN и категории LAUNCHER.

Дальше следует, детальный разбор структуры файла AndroidManifest с описыванием всех доступных <tags>, с примером для каждого. (Примеры не везде, но я точно ничего не потерял. И фраза “для каждого” в англ. версии (“for each”) тоже была. Не знаю, где они потеряли эти примеры. – прим. переводчика). Кто чей потомок, можно определить по отступу:

<manifest>

Это корневой узел каждого AndroisManifest.xml. Он содержит пакета атрибутов, который указывает на какой-то конкретный пакет в Activity. Другой путь к Activities будет базироваться относительно его значения:

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="org.anddev.android.smstretcher">
```

<uses-permission>

Описывает разрешения безопасности, которы нужно предоставить Вашему пакету. Количество не ограничено:

```
    <uses-permission    android:name    =    "android.permission.
RECEIVE_SMS"/>
```

<*permission*>

Объявляет разрешение безопасности, которое может использоваться, чтобы указать, какие приложения могут обратиться к компонентам или особенностям в этом пакете. Количество не ограничено.

<*instrumentation*>

Объявляет код контрольно-измерительного компонента, который доступен, чтобы проверить функциональные возможности этого или другого пакета. См. Оснащение аппаратурой для большего количества подробностей. Количество не ограничено.

<*application*>

Корневой элемент, содержащий объявления компонентов на уровне приложения, содержится в пакете. Этот элемент может также включать глобальные и/или заданные по умолчанию атрибуты для приложения, такие как метка, значок, тема, требование разрешения, и т.д. Количество – от нуля до единицы:

```
<application android:icon="@drawable/icon">
```

<*activity*>

Activity – первичная вещь для приложения, чтобы взаимодействовать с пользователем. Экраном, который пользователь видит, запуская приложение, является Activity, и большинство других экранов, которые они используют, будет осуществлено как отдельные действия, объявленные с дополнительными тэгами Activity:

```
    <activity                                  android:name=".Welcome"
android:label="@string/app_name">
```

1.2.7 Объявление компонентов

Основная задача манифесте является информирование систему о компонентах приложения. Например, файл манифеста может объявить деятельность следующим образом:

```
    <? XML version = "1.0" encoding = "utf-8" ?>
    <manifest ... >
        <application  android:icon = "@drawable/app_icon.png" ...
>
            <activity  android:name =
"com.example.project.ExampleActivity"
```

```
                        android:label = "@string/example_label" ...
>
            </activity>
            ...
        </application>
    </manifest>
```

В <application> элемента, Android: значок атрибут указывает на ресурсы для значка, который идентифицирует приложение.

В <Activity> элемента, Android: Название атрибута указывает полное имя класса активность подкласса и Android: этикетка атрибуты определяет строку для использования в качестве видимого пользователю метки для деятельности.

Вы должны объявить все компоненты приложение следующим образом:

– <Activity> элементы для деятельности.
– <service> элементы для услуг.
– <receiver> элементы для радиовещательных приемников.
– <provider> элементы для контент-провайдеров.

Деятельность, услуги и контент-провайдеров, что вы включаете в источнике, но не заявляют в манифесте не видны в системе и, следовательно, никогда не может работать. Тем не менее, радиовещательных приемников может быть либо объявлены в манифесте или создаются динамически в коде (как BroadcastReceiver объектов) и зарегистрирован в системе, вызвав registerReceiver ().

Более подробную информацию о том, как структурировать файл манифеста для вашего приложения, см. The AndroidManifest.xml файла документации.

<intent-filter>

Объявляет, какие Intent компонент поддерживает. В дополнение к различным видам значений, которые могут быть определены под этим элементом, атрибуты могут быть даны здесь, чтобы поставлять уникальную метку, значок, и другую информацию для описываемого действия.

<action>

Тип действия, который компонент поддерживает. Пример:

```
<action android:name="android.intent.action.MAIN" />
```

<category>

Тип категории, который компонент поддерживает. Пример:

```
<action android:name="android.intent.category.LAUNCHER" />
```

<data>

Тип MIME, URI схема или URI путь поддержки компонента.

Вы можете также связаться 1+ части метаданных с вашим Activity.

<meta-data>

Добавляет новую часть метаданных к Activity, которую клиенты могут восстановить через ComponentInfo.metaData.

<receiver>

IntentReceiver позволяет приложению сообщать о замене данных или о действиях, которые происходят, даже если приложение не выполняется в настоящее время. Как и с тэгом activity, в этот тег Вы можете произвольно включить больше одного <intent-filter> или <meta-data>, так же, как с <activity>:

```
<receiver android:name=".SMSReceiver">
```

<service>

Service – компонент, который работает на заднем плане произвольное количество времени. В этот тег, как с тэгом activity, Вы можете произвольно включить один или более <intent-filter> или <meta-data>.

<provider>

ContentProvider – компонент, который управляет постоянными (persistent) данными и открывает к ним доступ другим приложениям. Вы можете также произвольно присоединить один или более <meta-data>, как и в activity. <intent-filter> тут нету.

Конечно, все <теги> должны быть или так </закрыты>, или так, <напрямую/>.

1.2.8 Ресурсы

Ресурсы – внешние файлы (не код), которые используются Вашим кодом, закомпилированы в Ваше приложение и встраиваются в него во время работы. Андроид поддерживает многие различные виды файлов ресурсов, включая XML, PNG и JPEG. Файлы XML имеют сильно различающиеся форматы в зависимости от того, что они описывают. Ресурсы описаны в исходном коде, и файлы XML откомпилированы в двоичный код для быстрой и эффективной загрузки. Строки сжаты в более форму, более экономящую память.

Список ресурсов

Типы ресурсов и их местоположение:

- Layout-файлы – “/res/layout/”.
- Изображения – “/res/drawable/”.
- Анимация – “/res/anim/”.
- Стили, строки и массивы – “/res/values/”.

Названия не могут отличаться:

– ‘arrays.xml’ для определения массивов.
– ‘colors.xml’ для определения цветов.
– #RGB, #ARGB, #RRGGBB, #AARRGGBB.
– ‘dimens.xml’ для определения размеров (dimensions).
– ‘strings.xml’ для определения строк.
– ‘styles.xml’ для определения стилей объектов.
– Необработанные файлы вроде mp3 или видео – “/res/raw/”.

Использование ресурсов в коде

Для использования ресурса в коде нужно знать только полный ID ресурса и в какой тип объекта Ваш ресурс был откомпилирован. Вот синтаксис обращения к ресурсу:

```
R.resource_type.resource_nameт
```

или

```
android.R.resource_type.resource_name
```

resource_type – подкласс R, который содержит определенный тип ресурса. resource_name – атрибут ресурсов, определенный в файлах XML, или имя файла (без расширения) для ресурса, определенных другими типами файла. Каждый тип ресурса будет добавлен в подкласс R, в зависимости от его типа.

Ресурсы, откомпилированные Вашим приложением, могут быть использованы без названия пакета (просто как R.resource_type.resource_name). Android содержит многие стандартные ресурсы, такие как стили экрана и фоны кнопки. Обращаться к ним в коде Вы можете через android.R.resource_type.resource_name, для примера:

```
android.R.drawable.button_background
```

Ссылка на Ресурсы

Значение в атрибуте или ресурсе может также быть ссылкой на другой ресурс. Это часто используется в layout файлах, чтобы хранить строки (таким образом можно локализовать приложение) и изображения (находящиеся в другом файле), хотя ссылка может быть на любой тип ресурса, включая цвета и числа.

Например, если у нас есть ресурсы с цветами, мы можем написать layout файл, который установит цвет текста на значение, содержащееся в одном из ресурсов:

```
<EditText
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:textColor="@color/opaque_red"
```

```
    android:text="Hello, World!" />
```

Обратите внимание на префикс "@", показывающий, что это – ссылка на ресурс, текст после него – название ресурса в форме @[пакет:]тип/имя. В примере мы не определяем пакет, потому что ссылаемся на ресурс в нашем собственном пакете. Чтобы сослаться на системный ресурс, Вы должны были бы написать:

```
    <EditText
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
android:textColor="@android:color/opaque_red"
    android:text="Hello, World!" />
```

В следующем примере, мы используем ссылку на ресурс, храня строки в layout файле так, чтобы они могли быть локализованы:

```
    <EditText
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
android:textColor="@android:color/opaque_red"
    android:text="@string/hello_world" />
```

Альтернативные ресурсы и локализация

Альтернативные ресурсы и локализация – серьезная проблема, достаточно хорошо решенная в Android. Обычно чаще всего Вы должны были бы проектировать UI, хорошо подходящий для каждого из возможных разрешений экрана одновременно, что почти невозможно.

Вы можете добавлять в свое приложение различные UI, языки или поддержку устройств с различной конфигурацией компонентов.

Заметьте, что даже если Вы добавите много разных языков, UI и всех других ресурсов, SDK сам определит набор ресурсов, который будет использоваться. К примеру, Android сам догадается, где какой язык Вам нужен и выберет его. Или UI. Чтобы включать дополнительные ресурсы, создайте параллельные папки с ресурсами и к каждому названию через черточку добавьте параметр (спецификатор), куда эта папка относится (язык, ориентация экрана, точки на дюйм, разрешение и т.д.). Вот, например, у этого проекта есть английская и немецкая локализация:

```
    MyApp/
    res/
    values-en/
    strings.xml
    values-de/
    strings.xml
```

Android поддерживает несколько типов спецификаторов, с различными значениями для каждого. Добавьте их концу названия папки ресурса, отделив от названия черточкой. Вы можете добавить много спецификаторов, отделяя их друг от друга черточками. Например, папка, содержащая drawable ресурсы только для определенной конфигурации:

```
MyApp/
     res/
          drawable-en-rUS-port-92dpi-finger-keyshidden-12key-
dpad-480x320/
```

Более того, вы можете определить только несколько определенных опций конфигурации, для которых определен ресурс:

```
MyApp/
res/
drawable-en-rUS-finger/
drawable-port/
drawable-port-160dpi/
drawable-qwerty/
```

Android выберет, какой из различных основных файлов ресурса подходит лучше всего во время выполнения, в зависимости от текущей конфигурации устройства.

R.java

R.java проекта – автоматически сгенерированный файл, индексирующий все ресурсы Вашего проекта. Вы используете этот класс в своем исходном тексте как своего рода способ обратиться к ресурсам, которые Вы включили в свой проект. Это особенно важно, учитывая особенности интегрированных сред разработки, потому что позволяет Вам быстро и в интерактивном режиме определять местонахождение определенной информации, которую Вы ищете. Дополнительно во время компиляции Вы получаете уверенность, что ресурс, который Вы хотите использовать, действительно существует.

2 Инструменты разработки

2.1 Графическая составляющая

Для создания графических элементов проекта используется программное обеспечение, разработанное компанией Adobe для работы с векторной графикой – Illustrator CC. Для его приобретения необходимо пройти по адресу http://www.adobe.com/ru/products/illustrator.html и оформить годовую подписку на приложение, затем загрузить и установить программу. Помимо этого, существует возможность приобретения DVD-диска в специализированных магазинах. Начало процесса установки изображено на рисунке 2.1.

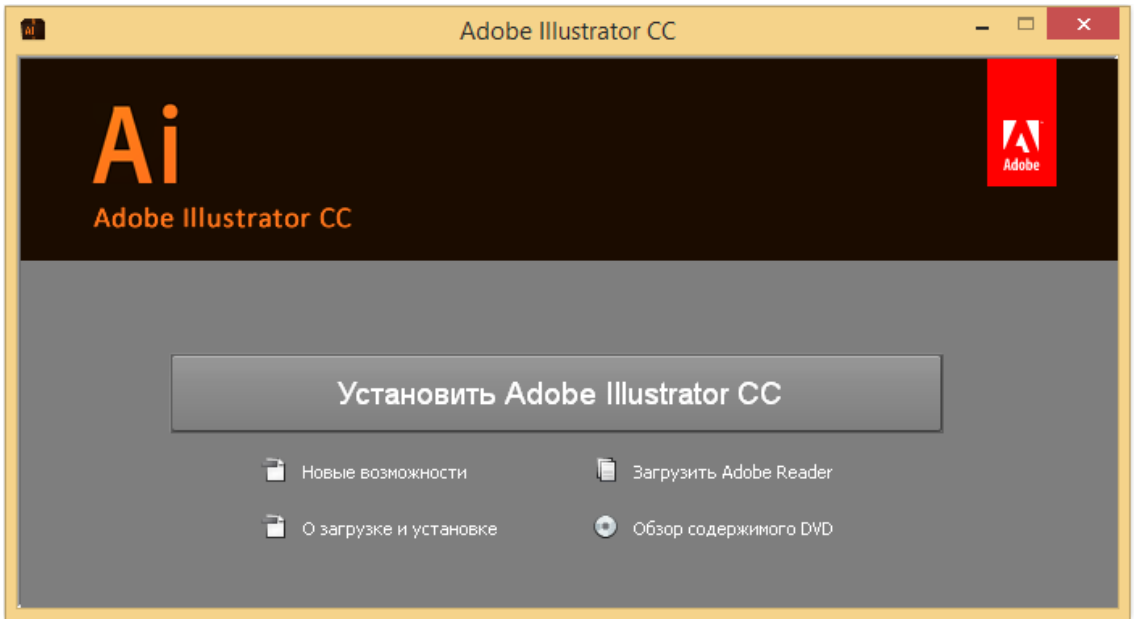

Рисунок 2.1 – Начало установки Adobe Illustrator CC

Существует 2 версии Adobe Illustrator CC, для 32 и 64 разрядных ОС соответственно. При использовании 64х системы, не требуется установка обоих версий, достаточно лишь версии для вашей ОС. Устанавливая одну версию программы вы сохраните пространство на диске. После выбора необходимой версии нажимаем кнопку «установить». На рисунке 2.2 изображен выбор устанавливаемой версии, за ним следует рисунок 2.3 с интерфейсом уже установленного приложения.

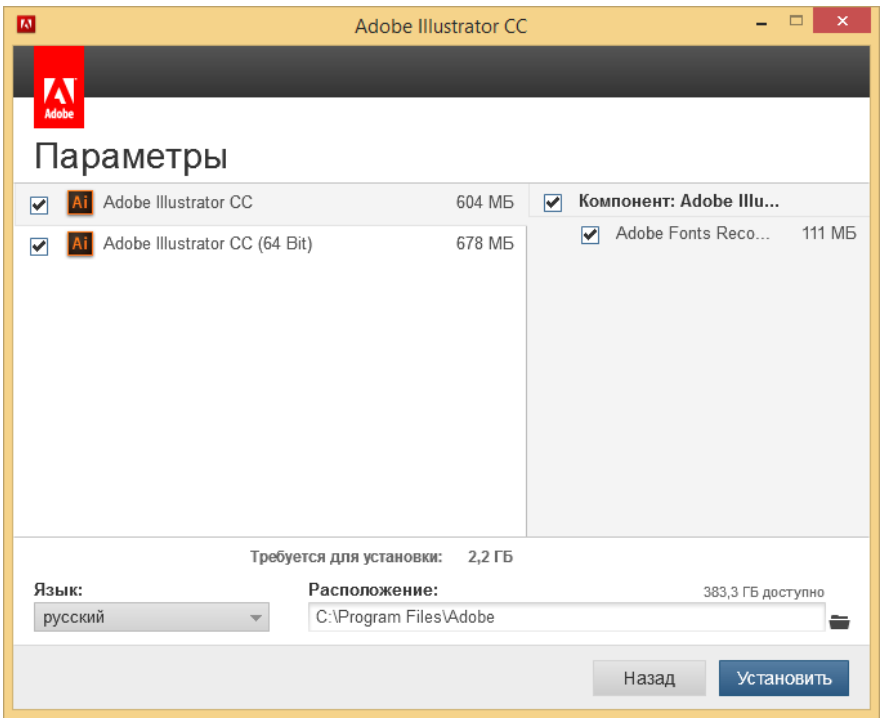

Рисунок 2.2 – Выбор устанавливаемых версий приложения

После завершения установки вы можете запустить программу.

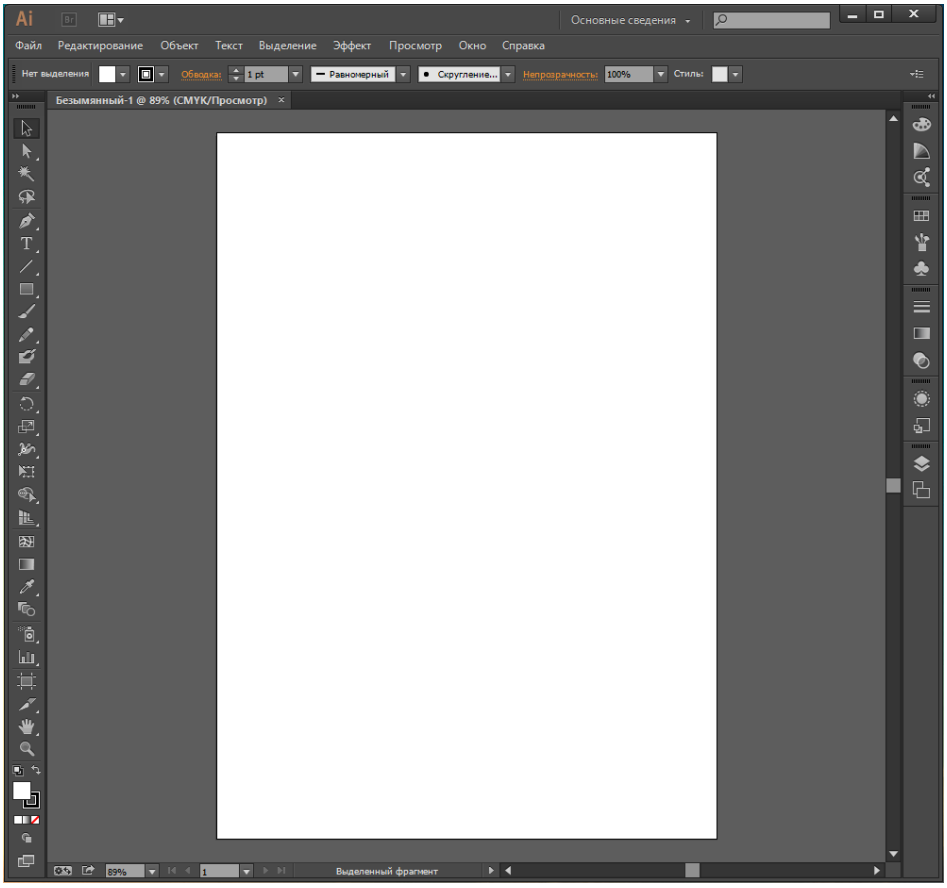

Рисунок 2.3 – Окно запущенного приложения Illustrator CC

Adobe Illustrator CC – это новейшая версия редактора векторных изображений, которая предоставляет новые возможности для дизайнеров, а также улучшает уже существующие инструменты.

Динамические углы

Сделайте свои работы изящными благодаря более точному и интуитивно понятному пользовательскому интерфейсу. Экспериментируйте со округлением углов фигур и контуров с помощью маркеров или путем ввода значений параметров в новом диалоговом окне «Углы» или прямо на панели управления. Один или несколько углов можно одновременно скруглять, переворачивать или делать их скошенными.

Полностью модернизированный инструмент «Карандаш»

Теперь с помощью этого инструмента можно более точно вычерчивать кривые с возможностью добавления прямых линий, а также продления и замыкания контуров. Используйте предварительно заданные настройки для создания плавных контуров с меньшим числом точек или более точной отрисовки мазков натуральной кистью. Эта новая технология расширяет возможности инструментов «Кисть», «Кисть-клякса» и «Сглаживание».

Изменение формы сегмента контура

Свободно перетаскивайте сегменты контура для создания нужной формы. Новая технология изменения формы контура, которая доступна для инструментов «Опорная точка» и «Прямое выделение» в меню инструмента «Перо», обеспечивает более точный и интуитивно понятный способ редактирования сегментов контура.

Интеграция с Typekit

Выберите нужный шрифт Adobe Typekit, открыв сайт Typekit прямо из меню «Шрифт» или «Гарнитура». Выберите любой из более чем 700 шрифтов Typekit, а затем синхронизируйте его со своим компьютером. В меню «Шрифт» можно быстро применить нужный фильтр к шрифтам Typekit.

Поддержка ОС Windows 8

Используйте для рисования перо с функцией определения степени нажатия, которое идет в комплекте с новейшими планшетами Windows 8, а также улучшенную поддержку функции прямого сенсорного ввода. Оцените по достоинству поддержку дисплеев HiDPI на компьютерах под управлением Windows 7 и 8.

Настраиваемая панель инструментов

Персонализируйте свое рабочее пространство и инструменты для определенных задач. Создавайте специализированные наборы инструментов, перетаскивая на пользовательскую панель только нужные инструменты,

например, инструменты для рисования, редактирования или выделения. После чего можно скрыть всю панель «Инструменты», получив больше пространства для творчества.

Импорт и экспорт настроек

Синхронизируйте свои настройки приложения Illustrator на нескольких компьютерах. Просто экспортируйте их в папку, из которой другие пользователи смогут затем импортировать их. Функция синхронизации настроек позволяет вам не только стандартизировать настройки на своих устройствах, но и делиться ими с коллегами.

Новые функции, помогающие экономить время

Воспользуйтесь такими новыми функциями экономии времени, как возможность рисовать монтажные области и изменять их размер из центра, доступ к элементам управления заливкой и обводкой прямо на панели «Образцы», а также ползунки для настройки непрозрачности.

Улучшенные возможности рисования с учетом перспективы

Экспериментируйте с изменением атрибутов сетки перспективы, таких как исправление перспективы и линия горизонта, и просматривайте динамические изменения графических объектов с учетом этих правок.

Обновления функции экспорта в формат SVG

Экспортируйте свои работы в масштабируемый и оптимизированный формат файлов SVG, которые можно адаптировать к экранам различных размеров и разрешений, а затем выполняйте сквозное редактирование файлов SVG с сохранением точного выравнивания пикселов.

Инструмент «Изменение текста»

Создавайте проекты и добавляйте в них текст с помощью эффективного инструмента «Изменение текста». Теперь с символами можно работать как с отдельными объектами. Экспериментируйте со шрифтами, перемещайте, масштабируйте и поворачивайте текст. Теперь можно создавать работы не только с помощью мыши или стилуса, но и просто касаясь сенсорного экрана мобильного устройства.

Изображения в кистях

Рисуйте кистью, в которую была помещена фотография. Объектная, узорчатая и дискретная кисти могут содержать растровые изображения, что позволяет создавать сложный дизайн за несколько минут, рисуя обводки, имитирующие мазки натуральной кистью. Какой бы кистью Illustrator вы ни воспользовались, форму и внешний вид обводки можно изменить по вашему желанию.

Поиск шрифта

Быстро находите идеально подходящий шрифт. В палитре «Символ» введите стиль шрифта, например, «полужирный» или «курсив», название семейства шрифтов или часть названия шрифта. Отобразятся только те результаты поиска, которые отвечают параметрам.

Использование нескольких файлов

Импортируйте в Illustrator несколько файлов одновременно и управляйте процессом с помощью новых функций. Теперь можно определить местоположение файлов (изображений, графики и текста), применить к ним масштабирование, а также воспользоваться новым видом миниатюр, чтобы уточнить расположение файла в проекте.

Извлечение каскадных таблиц стилей (CSS)

Написание кода для таких веб-элементов, как значки и узоры, может быть утомительным. Однако теперь создавать веб-сайты стало еще проще благодаря программе Illustrator, которая сама создает код CSS даже для логотипов, включающих в себя градиенты. Копируйте и вставляйте код прямо в ваш веб-редактор.

Синхронизация цвета

Фиксируйте найденные и понравившиеся вам цветовые темы с помощью приложения Adobe Kuler для iPhone. Публикуйте свои цветовые темы и оцените тысячи тем других пользователей на веб-сайте Kuler. Синхронизируйте любимые цветовые темы и получайте к ним мгновенный доступ из Illustrator.

Преобразование текста из точки в текст в области и наоборот

Мгновенно переключайтесь между текстом в области и текстом из точки. Преобразование текстового объекта теперь можно выполнить всего за секунду, что в значительной степени упрощает процесс создания дизайна в текстовых макетах. Работать с импортированным текстом стало еще проще благодаря функции изменения формата.

Автоуглы для узорчатых кистей

Создавайте углы всего за несколько шагов. Создавайте узорчатые кисти всего за несколько секунд благодаря функции автоматического создания углов, которые идеально подойдут к остальному оформлению обводки. Больше не нужно заниматься утомительным созданием специальных острых углов.

Применение инструмента «Свободное трансформирование»

Совершенствуйте свои навыки в создании дизайна с помощью инструмента «Свободное трансформирование». Перемещайте, масштабируйте и поворачивайте объекты прямо на сенсорном экране мобильного устройства.

Либо воспользуйтесь мышью или другим манипулятором, чтобы интуитивно и быстро трансформировать объекты прямо на монтажной области.

Интеграция с Behance

Сохраняйте свою работу прямо из программы Illustrator CC в сервис Behance, чтобы продемонстрировать свои готовые проекты или опубликовать те, над которыми работаете. По мере доработки проекта загружайте новые версии и моментально получайте отзыв о своей работе от других дизайнеров со всего мира.

Синхронизация настроек

Создавайте проекты на любом компьютере: Mac или PC. Синхронизируйте настройки вашего рабочего пространства в облачном сервисе Creative Cloud, включая установки, стили, кисти и библиотеки Illustrator, и используйте их, где бы вы ни находились.

Упаковка файлов

Функция упаковки файлов позволяет автоматически собрать все необходимые шрифты, связанную графику и отчет об упаковке в одной папке. Можно воспользоваться упаковкой файлов для более удобной сдачи проектов заказчику или их систематизации на компьютере.

Извлечение изображений

С легкостью извлекайте изображения, которые были помещены и встроены в файл llustrator. Извлекайте файлы за несколько секунд и приступайте к их редактированию. Можно также извлечь файлы, встроенные в иллюстрацию, которая была получена от другого пользователя. Ссылки на файлы изображений создадутся автоматически.

Использование нескольких монтажных областей

Упорядочивайте и просматривайте до 100 монтажных областей разных размеров, расположенных каскадом или в виде сетки. С легкостью добавляйте, удаляйте и переименовывайте области, а также меняйте порядок их расположения. Сохраняйте, экспортируйте и печатайте монтажные области по отдельности или вместе.

Обводки переменной ширины

Рисуйте обводки переменной ширины, легко выполняя корректировку на любом этапе работы. Создавайте и сохраняйте профили ширины и применяйте их к обводкам — либо используйте стили переменной ширины.

изображения

Преобразуйте растровые изображения в редактируемые векторы при помощи эффективного механизма трассировки, обеспечивающего

исключительный уровень контроля работы с цветами и фигурами. Простые, интуитивно понятные функции обеспечивают высокую точность линий, четкость подгонки и получение надежных результатов.

Система Adobe Mercury Performance

Обрабатывайте большие, сложные файлы с высокой точностью, скоростью и надежностью. Благодаря встроенной поддержке 64-разрядных вычислений Mac OS и Windows система Adobe Mercury Performance позволяет приложению получать доступ ко всей оперативной памяти, чтобы с легкостью открывать, сохранять и экспортировать объемные файлы, а также осуществлять предварительный просмотр сложных дизайнов.

2.2 Инструменты разработки кода приложения

Для написания кода ПП в данном проекте будет использованная среда для разработки Android Studio, а языком будет служить Java. Для работы в Android Studio необходимо наличие установленного Java SE Development Kit (JDK) на используемой системе. Рекомендуемая версия JDK не ниже 7. Установить его можно следуя по ссылке:

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.

Java SE Development Kit является бесплатным инструментарием для разработки и распространяется бесплатно. Таким образом, после перехода по приведенной выше ссылке, вам необходимо выбрать версию, соответствующую вашей ОС, Windows, Linux или же Mac OS X. После загрузки исполняемого файла его необходимо запустить и следуя инструкциям, установить данный инструментарий.

После этого следует загрузить установочный файл Android Studio с официального сайта Android для разработчиков. Установочные файлы доступны по адресу:

http://developer.android.com/intl/ru/sdk/installing/studio.html.

Здесь также следует выбрать нужную версию, подходящую для вашей ОС и после окончания загрузки исполняемого файла, установить программу следуя инструкциям. Здесь стоит обратить внимание на то, что загрузка Android NDK для работы в Android Studio не требуется, так как все необходимые утилиты уже включены в комплект поставки приложения.

После запуска приложения, вас будет ожидать экран приветствия изображенный на рисунке 2.4. Здесь вы сможете:

– Создать новый проект.
– Импортировать уже существующий проект (например, если вы до этого вели разработку в Eclipse).
– Открыть уже существующий проект.
– Открыть любой недавний проект, а также выполнить поиск среди них.

– Проверить актуальность версии ПО.
– Изменить настройки Android Studio.
– Получить доступ к справе.

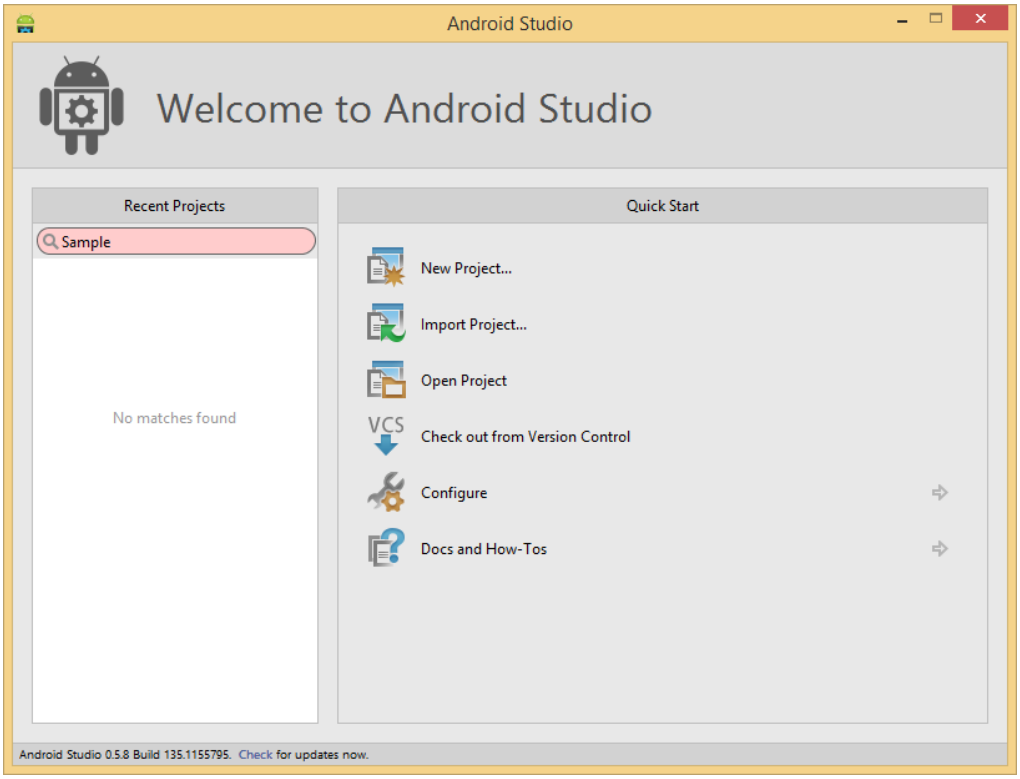

Рисунок 2.4 – Экран приветствия Android Studio

Для создания нового проекта необходимо нажать «New Project…» и за этим последует окно создания проекта, показанный на рисунке 2.5.

На этом экране вы можете задать имя приложения, название модуля, название файлы для приложения, а также место расположения каталога с данными проекта. Также имеется возможность выбора минимальной версии ОС, поддерживаемой приложением и основной, на которую оно ориентированно в первую очередь. Помимо этого, существует вариант выбора уровня языка. Сразу же доступен выбор базовой темы приложения, светлой и темной. Если требуется, то можно поставить маркер в бокс выбора «Create custom launcher icon», для создания иконки приложения непосредственно в Android Studio. И заключением является возможность выбора «Support Mode».

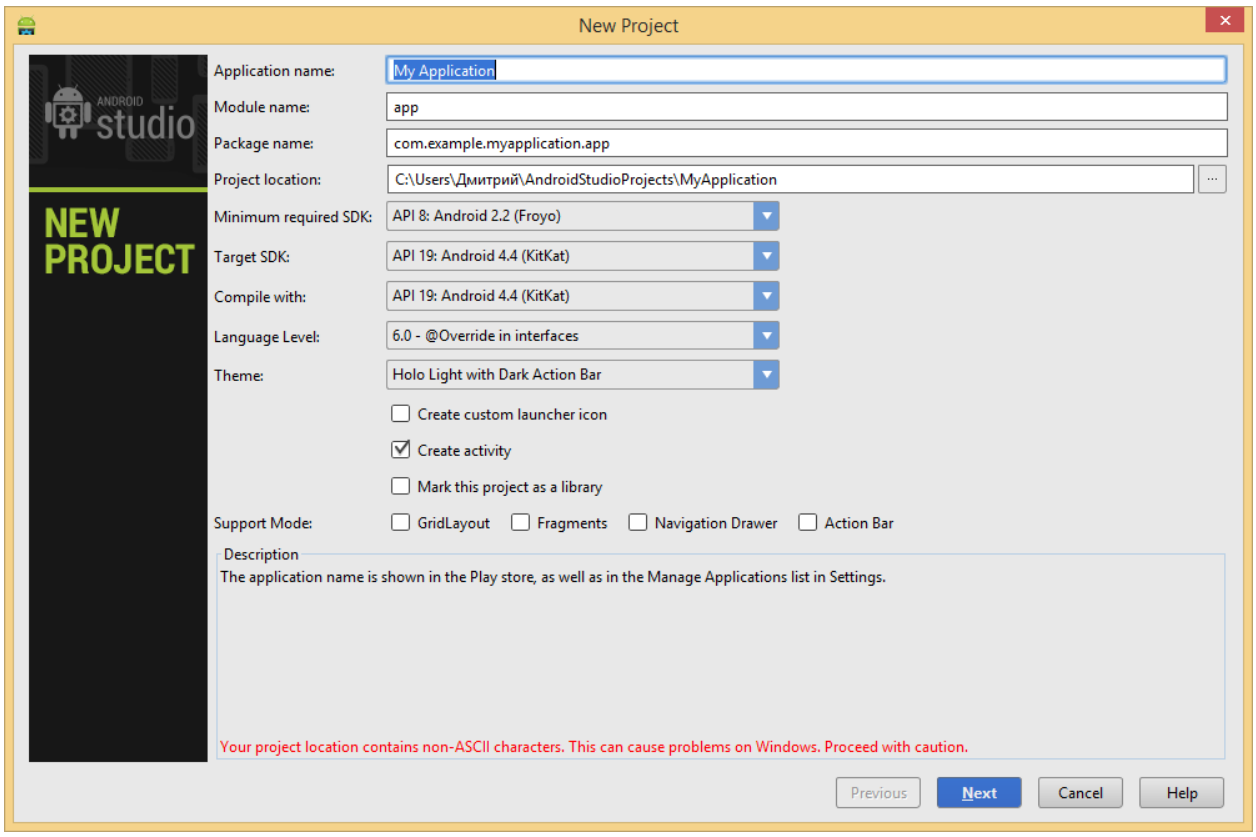

Рисунок 2.5 – Окно создания проекта Android Studio

Если в окне создания проекта было выбрано «Create custom launcher icon», то после нажатия кнопки «next» вы увидите окно создания и редактирования значка приложения, которое также изображено на рисунке 2.6. Этот редактор удовлетворит потребности практически всех разработчиков, так как позволяет создать быстро и качественно значки приложения для разных плотностей экрана и сразу же увидеть результат.

В этом окне вы можете выбрать, что именно использовать в качестве значка: готовое изображение, которые вы можете загрузить со своего компьютера; клипарт или текст. Здесь же присутствует возможность изменения размера содержимого значка, назначить, при необходимости, фон, выбрать его форму и цвет.

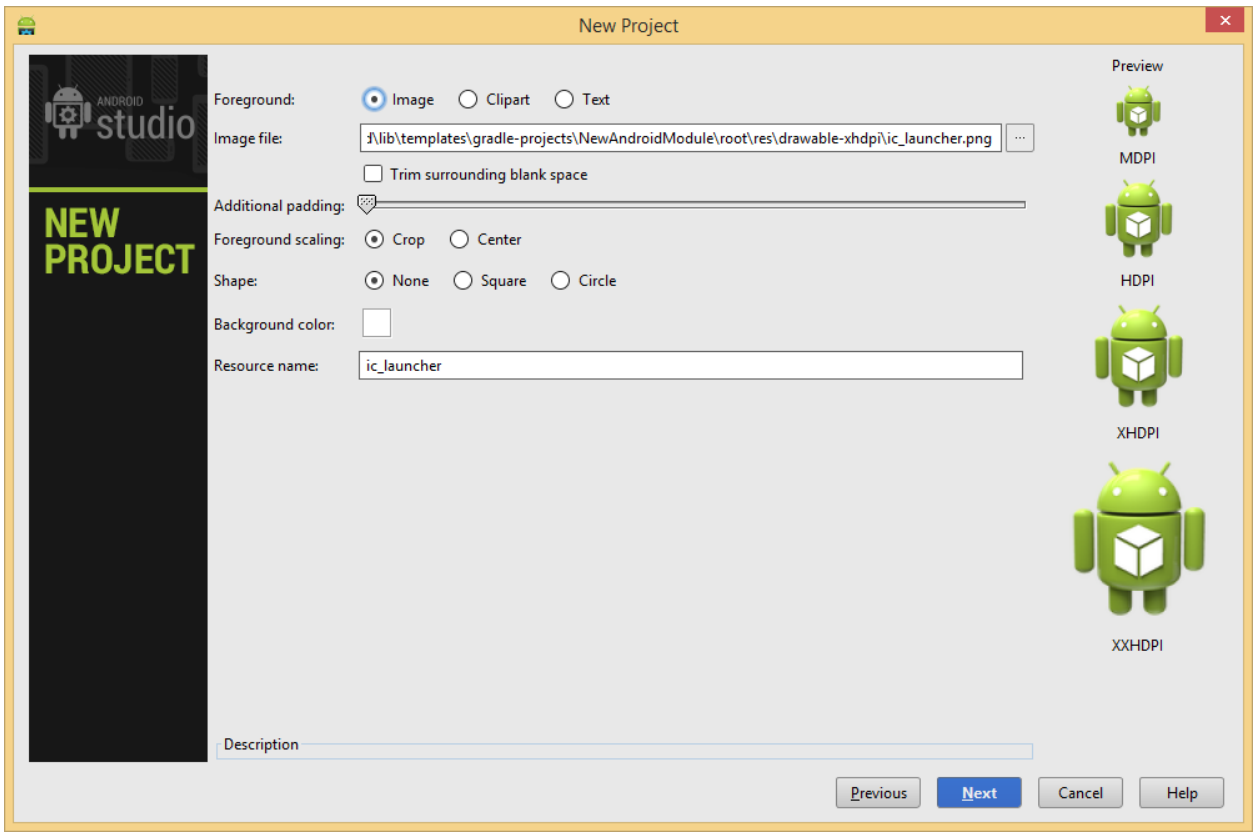

Рисунок 2.6 – Окно создания значка приложения

После завершения этапа создания значка, вы переходите на экран выбора базовой activity, изображенный на рисунке 2.7.

Имеется выбор из девяти вариантов:

– Blank Activity.
– Blank Activity with Fragment.
– Empty Activity.
– Fullscreen Activity.
– Google Maps Activity.
– Google Play Services Activity.
– Master/Detail Activity.
– Navigation Drawer Activity.
– Tabbed Activity.

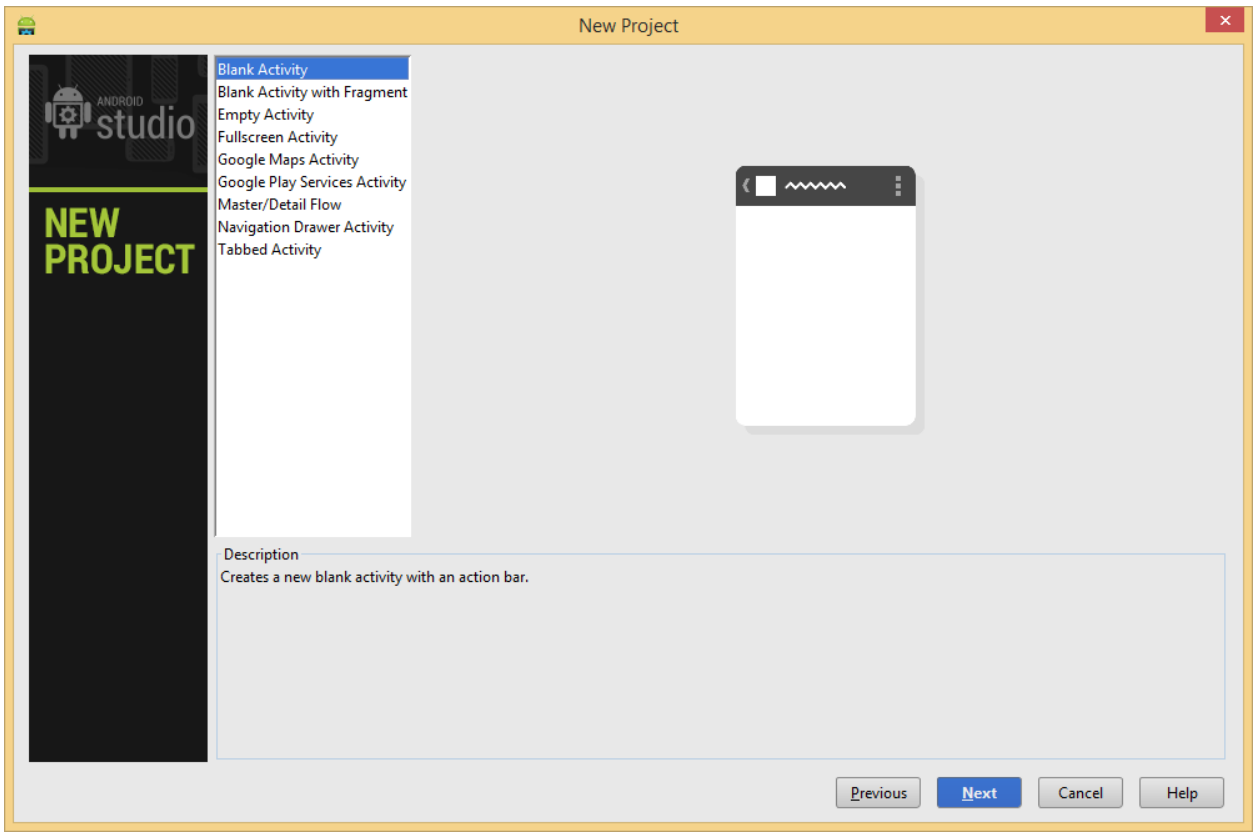

Рисунок 2.7 – Окно выбора базовой Activity

Далее указываем имя нашей стартовой Activity и имя её layout'а, а также имя первого фрагмента, поскольку мы выбрали поддержку фрагментов при создании проекта. Также можно выбрать тип навигации по Activity.

После чего нас переносят в окно самой IDE. Первое на что вы должны были обратить внимание – это структура проекта. И да, она не такая как в Эклипсе.

По-прежнему есть привычные папки src и res, но res теперь лежит внутри папки src, наравне с новой папкой java, в которую перекочевали наши пакеты и классы. Такой структурой проекта Android Studio обязана новой системе сборки проекта – Gradle. Она помогает управлять нам зависимостями в нашем проекте и подключать внешние библиотеки, но подробнее о ней не в этой статье.

Стоит упомянуть, что при импорте проектов с обычной структурой – всё так же отлично работает.

На рисунке 2.8 изображено основное рабочее окно среды разработки.

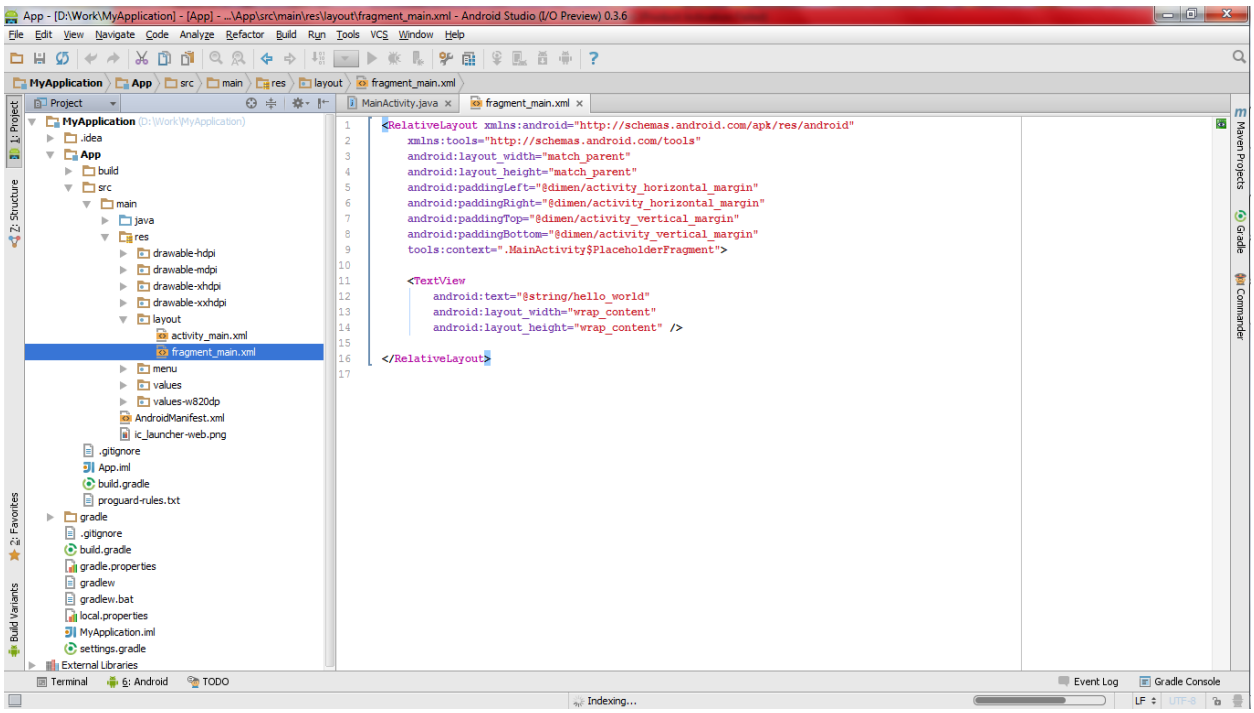

Рисунок 2.8 – Основное окно IDE

Xml редактор, изображённый на рисунках 2.9 и 2.10.

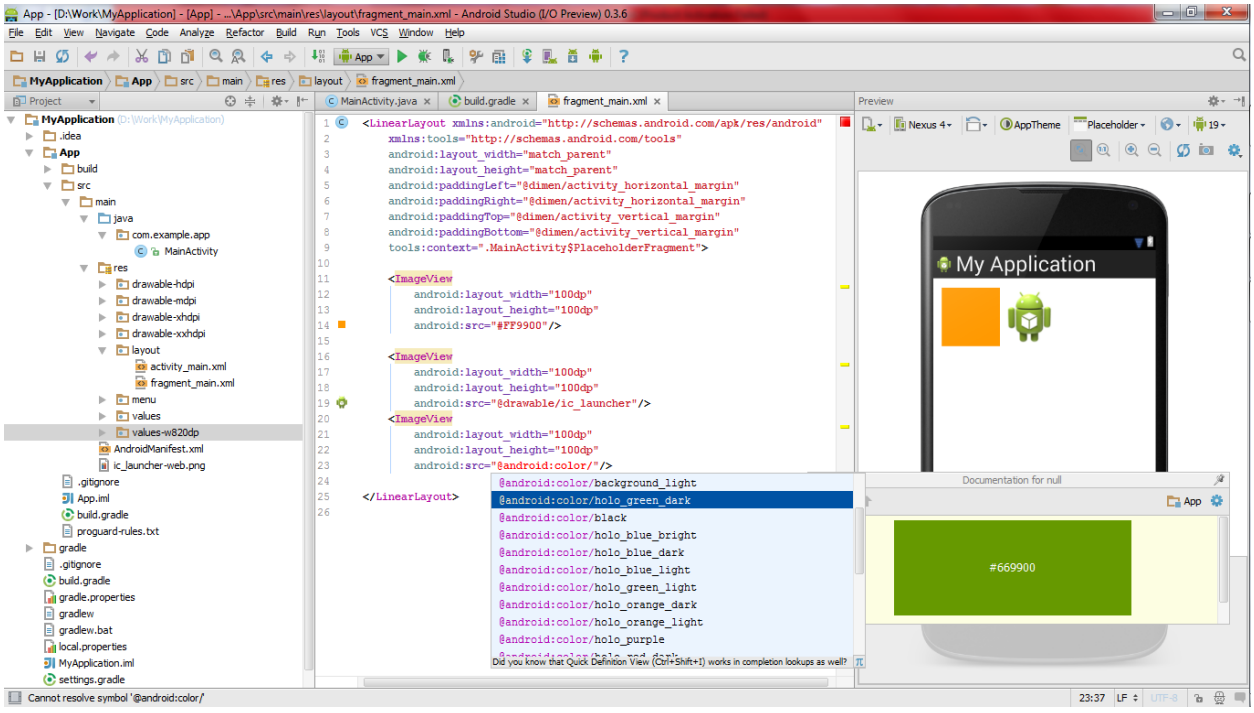

Рисунок 2.9 – Пример окна xml ректора №1

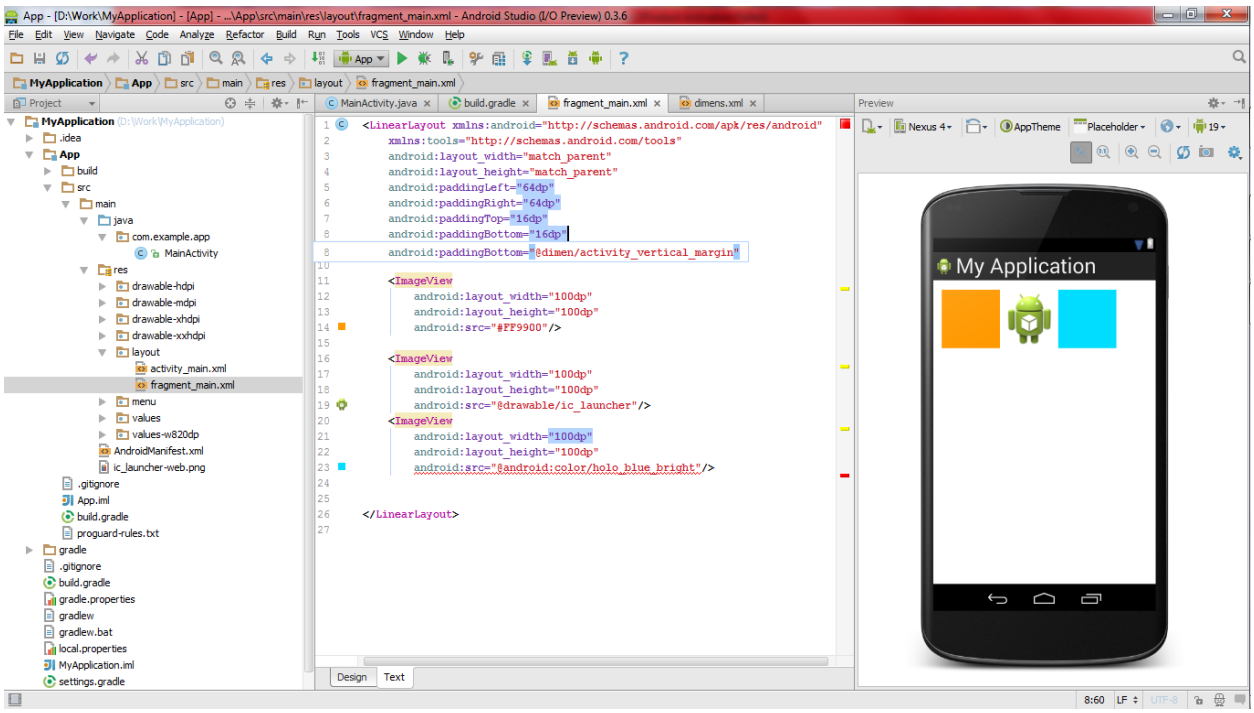

Рисунок 2.10 – Пример окна xml ректора №2

Рассмотрев рисунки 2.9 и 2.10 можно отметить следующее:

– При редактировании xml в текстовом режиме теперь также есть превью.

– Указанные цвета и рисунки, использованные в layout'е отображаются на границе в виде небольших превьюшек, которые легко помогают понять какой конкретно ресурс вы используете.

– При выборе ресурса, его содержимое отображается во всплывающих JavaDoc'ах, как например @android:color/holo_green_dark.

– Ресурсы из dimens автоматически показываются значениями, а при наведении вы всегда можете узнать какой именно ресурс вы используете.

Улучшенная интеграция с Android компонентами.

Попробуем добавить новый класс. Становимся внутрь пакета, куда хотим разместить класс, и нажимаем чудесное сочетание alt-insert. Хочу отметить что hotkey's в Android studio иногда достаточно сложно запомнить, в сравнении с Eclipse, но несут в себе гораздо более сложный и гибкий функционал.

Студия предлагает нам на выбор несколько объектов для создания. Кратко о каждом:

– Java Class – на самом деле Java Component. Позволяет создать один из основных компонентов Java: Class, Interface, Enum, Annotation и даже Singleton.

– Module – создание, собственно, модуля. Модуль – это обычно вспомогательный проект в Android Studio. Модулями в проекте будут являться все внешние библиотечные проекты (например, ActionBarSherlock или Facebook SDK).

– File – обычный файл любого фактически с любым разрешением (txt, json, xml и др.).

– Package – пакет нашего приложения.

Создание компонентов Android

Рисунок 2.11 демонстрирует окно для создания нового компонента приложения.

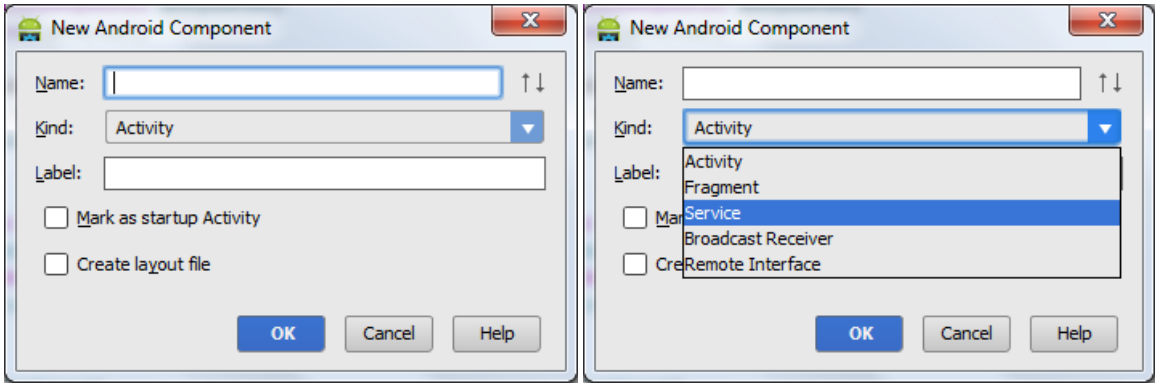

Рисунок 2.11 – Окно создания нового компонента

Activity – создает Activity по одному из готовых шаблонов, сразу регистрируя её в Манифесте.

Android Component – универсальная штука, позволяющая создать любой из ключевых компонентов нашего Android, сразу создать layout для него при необходимости, а также обозначить Activity как стартовую. По умолчанию регистрирует компонент в Манифесте, при необходимости.

Package-info.java – файл описания информации про пакет.

HTML File – собственно создает html файл.

Также в составе Android Studio имеется продвинутый текстовый редактор, которому присуще следующие преимущества:

– Соединяющие линии между началом и концом if, while,switch конструкций или метода.

– Более интеллектуальный анализ кода.

– Встроенное подключение Android Sources.

– Возможность наследоваться от класса, либо создавать тесты на него в 2 клика.

3. Этапы разработки приложения

3.1 Проектирование

ProdLaunch – приложение для платформы Android, принадлежащее к классу приложений «Персонализация». Представляет из себя одну и основных частей пользовательского интерфейса, предоставляя доступ для других, установленных в системе приложений. В связи с этим его интерфейс должен быть максимально прост и удобен, а также обладать достаточными эстетическими качествами. Основной целью приложения является обеспечение быстрого доступа к другим приложениям в необходимое для пользователя время. Внешний вид приложения должен сочетаться с остальными элементами оформления пользовательского интерфейса операционной системы, передовая задуманный разработчиками платформы опыт использования. Дизайн ПП должен строго следовать установленным нормам и требованиям платформодержателя (в данном случае Google).

Используя все вышеописанные нормы, разрабатывается концепция интерфейса программы ProdLaunch.

Основным ориентиром при проектировании интерфейса приложения был выбран Google Experience (Google Старт), так как в него заложены основные идеи организации пользовательского интерфейса новейшей версии ОС Android 4.4 Kit Kat. На рисунке 3.1 представлен внешний вид приложения от Google.

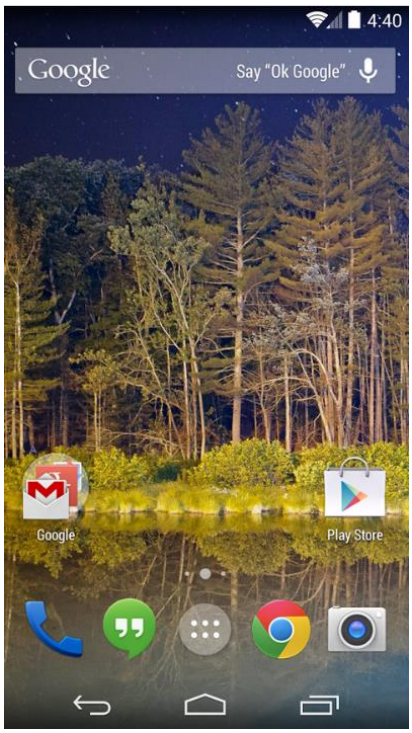

Рисунок 3.1 – Google Старт

Основной интерфейс программы представлен в виде 2 основные части: рабочего стола и меню приложений.

Меню приложений представляет из себя список всех установленных пользовательских программ. Все программы располагаются в виде «сетки» и отсортированы в алфавитном порядке. Размер сетки зависит от плотности экрана устройства. Так, например, для устройств типа смартфон, размер сетки составляет 4x5, а для 7-дюймовых планшетов – 5x6. Если пространство на экране заканчивается, то создается дополнительная страница в меню приложений. Для упрощения навигации между страницами внизу экрана расположен индикатор, представленный в виде точек, каждая их которых символизирует одну страницу меню. Индикатор отображает на какой странице сейчас находится пользователь, для этого используется более крупный размер точки. При построении списка программ, используется очередность:

1 Программы с названием, начинающимся с символов кириллицы.

2 Программы с названием, начинающимся с символов латиницы.

3 Программы с названием, начинающимся с иероглифов.

На рисунке 3.2 представлен вид меню приложений.

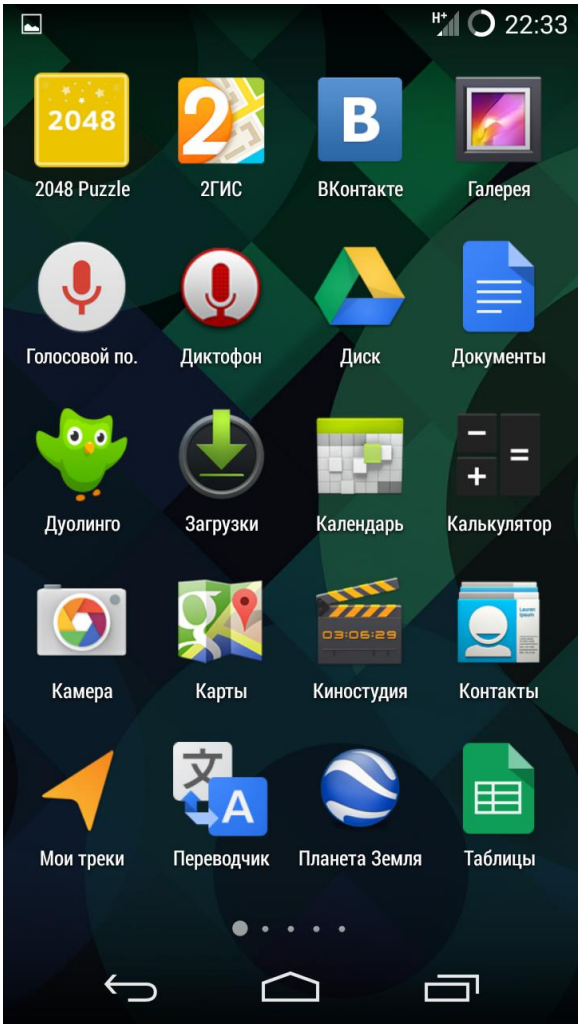

Рисунок 3.2 – Меню приложений

Рабочий стол же представлен пространством под иконки приложений (рабочая область), неподвижной панелью (док-панель) и строкой быстрого поиска.

Основную часть рабочего стола занимает рабочая область для расположения значков программ. Данная область используется для вынесения наиболее часто используемых приложений и является своего рода аналогом «избранного». Одна такая область вмещает в себя до 16 ярлыков, расположенных в виде сетки размером 4x4 для телефонов и 5x5 для планшетов. Изначально существуют лишь области, на которых расположены иконки. Как и меню приложений, рабочие области могут иметь несколько страниц, их создание также происходит в случае нехватки пространства для размещения знака программы или же в случае желания пользователя отделить одни приложения от других. В любое время вы можете беспрепятственно создать новую страницу рабочей области. Существует возможность выбрать главную рабочую область, при наличии нескольких, по умолчанию ей является крайняя слева. Для экономии места и удобной организации приложений существует возможность создания папок, размер которых ограничен по аналогии с размером рабочей области, то есть 4x4 для телефонов и 5x5 для планшетов.

На рисунке 3.3 представлен внешний вид рабочей области, а на рисунке 3.4 продемонстрирован внешний вид созданной папки.

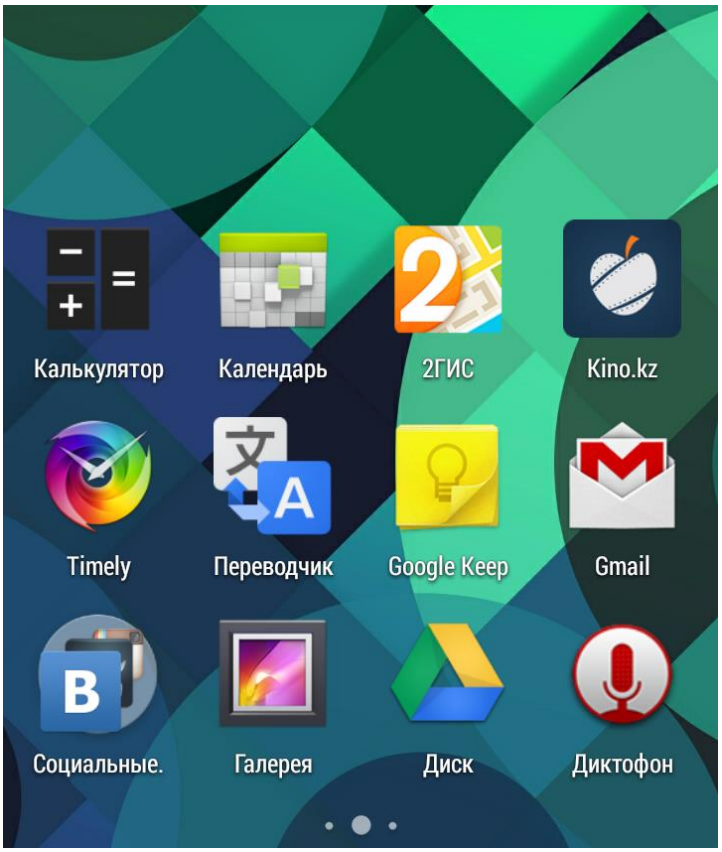

Рисунок 3.3 – Рабочая область

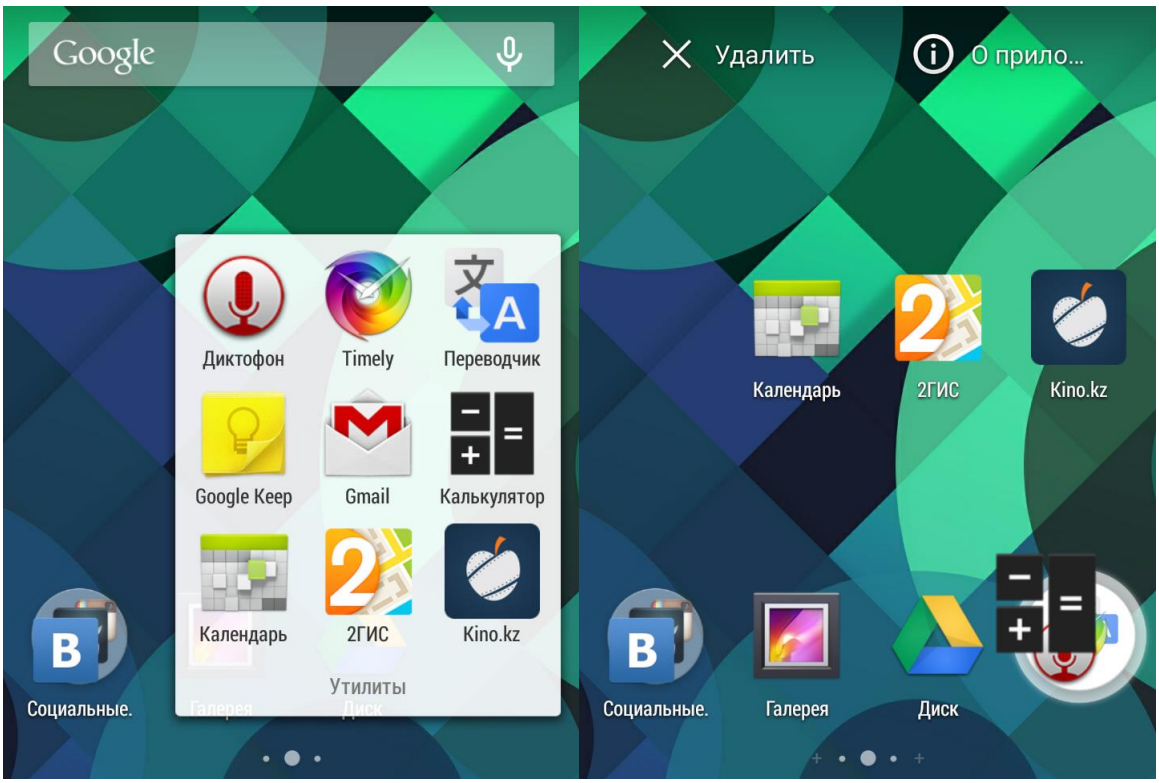

Рисунок 3.4 – Организация папки

Одной из отличительных особенностей операционной системы Android является поддержка виджетов. Они могут располагаться на рабочей области экрана или же, начиная с версии Android 4.2 Jelly Bean, на экране блокировки устройства. Сам по себе виджет – это небольшой программный модуль, имеющий возможность работать самостоятельно, за загружая в память все приложение. Большинство ПП для Android имеют в своем составе виджеты, позволяющие быстро выполнять какие-либо действия непосредственно с рабочей области, не запуская основное приложение. Также могут служить для вывода определённой информации, требующейся постоянно в актуальном состоянии, расположенной непосредственно на рабочем столе или экране блокировки.

На рисунке 3.5 представлен пример виджета.

Рисунок 3.5 – Изображение виджета

Док-панель – это небольшая панель, расположенная внизу рабочего стола. Состоит из 5 иконок, одна из которых – кнопка для входа в меню приложений, расположенная по центру. Слева и справа от значка меню приложения находится место для еще двух значков. Эти 4 значка пользователь может менять на свое усмотрение, так же размещая на этом месте папки. Значок меню приложения невозможно удалить или же переместить на другое место. Док-панель, в отличии от рабочей области, не может иметь несколько страниц и не сменяется на другую при смене рабочей области, она неподвижна.

Рисунок 3.6 содержит изображение док-панели.

Рисунок 3.6 – Док-панель

Строка быстрого поиска по аналогии с док-панелью всегда статична и не изменяет своего местоположения. Представляет собой закрепленный, независящий от рабочей области виджет и не может быть удалена по аналогии с ними. Состоит из двух активных частей, полосы, эмитируешь строку для ввода текста и значка микрофона. Служит в качестве ярлыка для запуска различного вида поиска, строки для текстового и значка для голосового. Отключение строки поиска возможно лишь в настройках домашнего экрана.

На рисунке 3.7 изображена строка поиска. Рисунок 3.8 же демонстрирует общий вид рабочего стола.

Рисунок 3.7 – Строка быстрого поиска

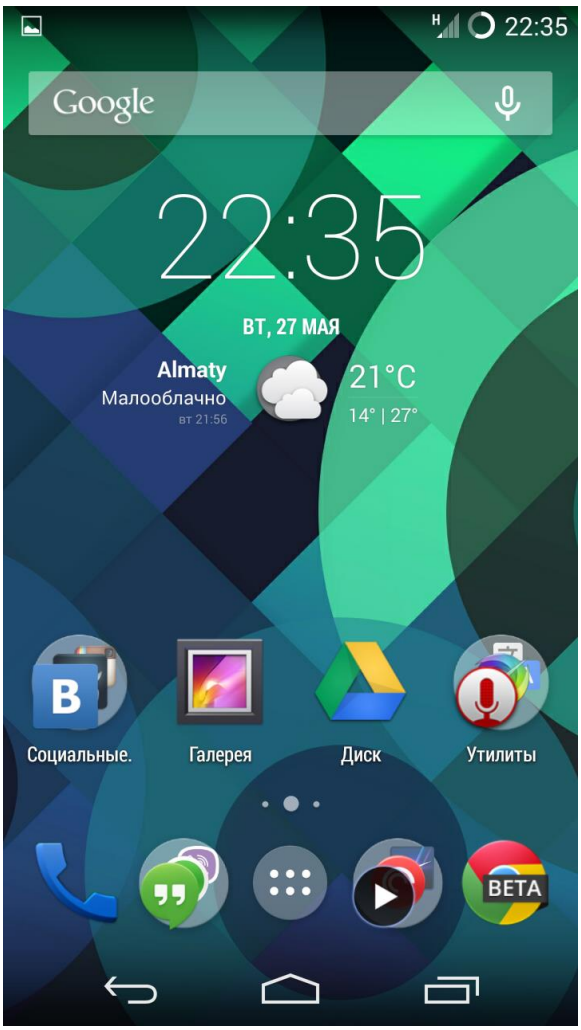

Рисунок 3.8 – Рабочий стол

Для изменения внешнего вида домашнего экрана, а также добавления виджетов и перехода в настройки устройства, создано окно настройки. Для более глубокого изменения домашнего экрана имеется меню настроек, которое доступно из настроек телефона.

3.2 Разработка графических элементов

При оформлении интерфейса приложения используются материалы, доступные в Android Open Source Project. Эти материалы представляют собой набор иконок основных элементов управления для приложения. Данные иконки используются для создания единого стиля разрабатываемого приложения и операционной системы. Для нестандартных элементов управления, таких как настройка внешнего вида домашнего экрана, разрабатываются новые, индивидуальные значки и изображения, схожие по стилю с Holo темой, используемой в Android.

На рисунке 3.9 изображен список доступных графических материалов для загрузки с http://developer.android.com/.

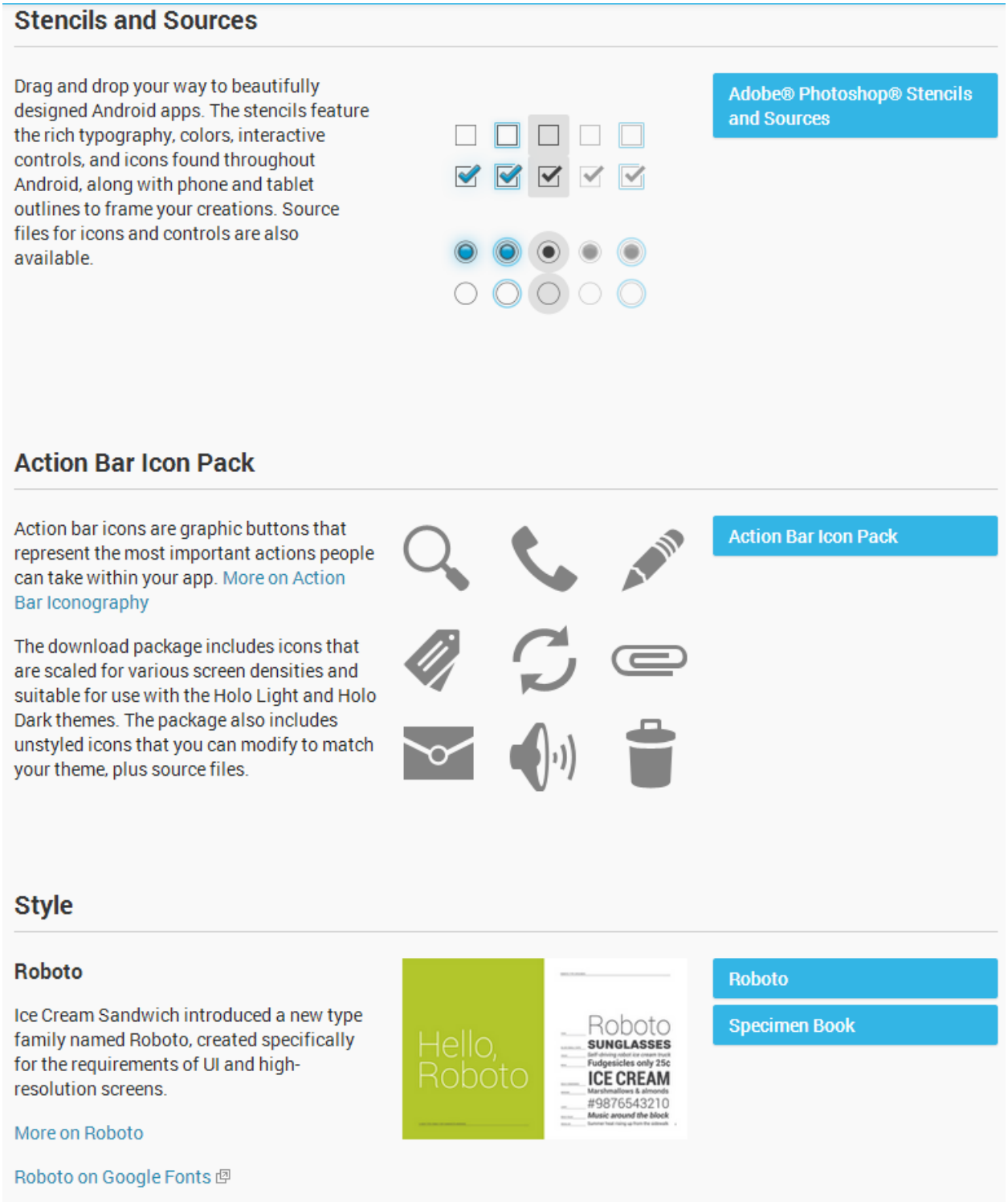

Рисунок 3.9 – Доступные для загрузки графические материалы

Для разработки оригинальных иконок, необходимых для уникальных возможностей приложения, используется Adobe Illustrator CC. Основное его преимущество перед продуктом той же компании Adobe Photoshop CC в том, что создаваемые изображения получаются векторными, а не растровыми, то есть изменения их масштаба не как не сказывается на качестве получаемого

материала, что в свою очередь очень удобно при разработке под различные плотности экрана.

Все разрабатываемые уникальные графические элементы используются в оформлении, так называемого, окна настройки.

Рисунок 3.10 отдельно демонстрирует разработанные иконки, а на рисунке 3.11 изображен результат разработки интерфейса для окна настройки.

Рисунок 3.10 – Разработанные иконки

Рисунок 3.11 – Окно настройки

3.3 Навигация и управление

Для навигации в разрабатываемом приложении и управлением им используются механические или сенсорные кнопки навигации, в устройствах оборудованными ими, или наэкранные, в устройствах не имеющих таковых. Помимо этого, используются жесты и кнопки управления.

Система навигации и управление полностью соответствует требованиям Google к приложениям для операционной системы Android.

Домашний экран является основным интерфейсным приложением, поэтому при включении устройства пользователь первым делом видит его (при условии, что устройство уже активировано и настроено). Для возврата к домашнему экрану из любого приложения необходимо нажать на кнопку «домой» (home). Если пользователь находится на главной рабочей области, то нажатие кнопки «домой» не вызовет никаких действий, если же пользователь находится на «второстепенной рабочей области» (не главной), то нажатие данной кнопки вернет его на нее (на главную).

На рисунках 3.12 и 3.13 изображены основные действия на возврата на главный экран.

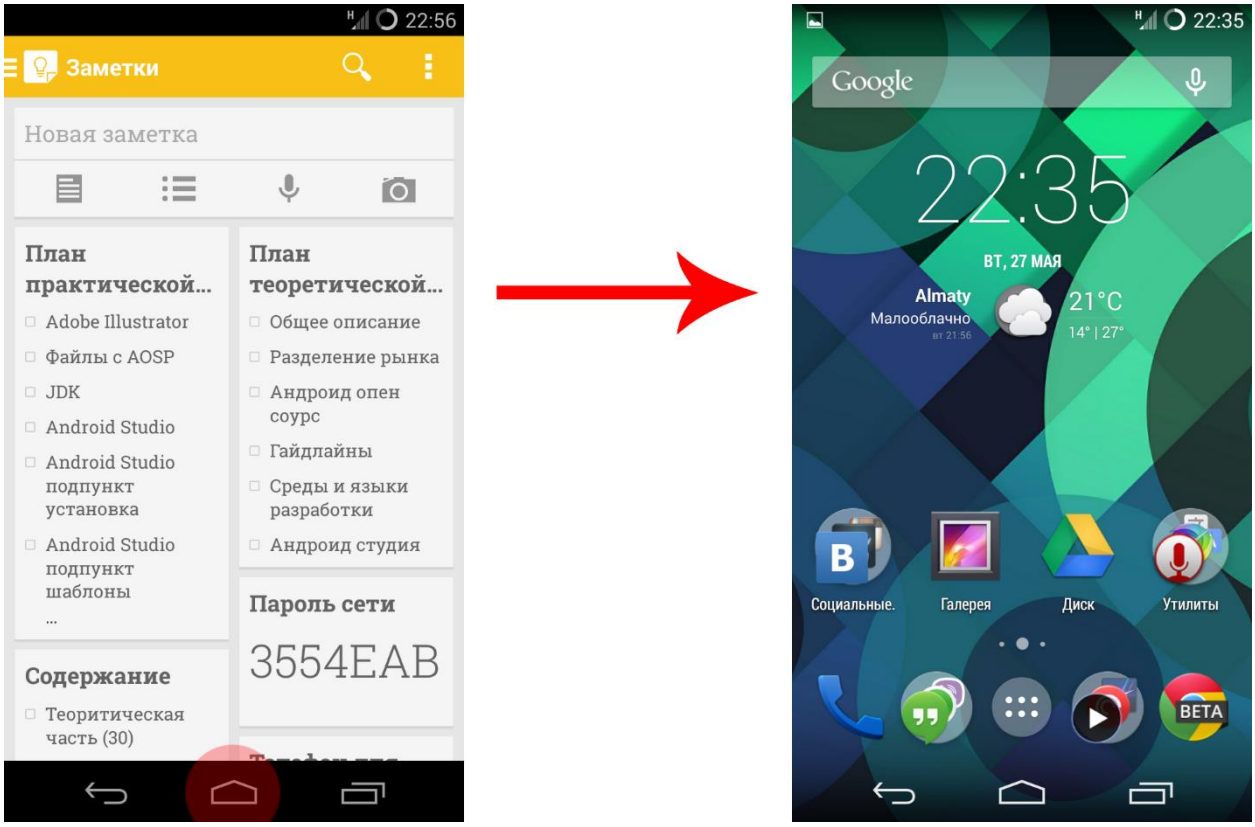

Рисунок 3.12 – Возврат к главному экрану из приложения

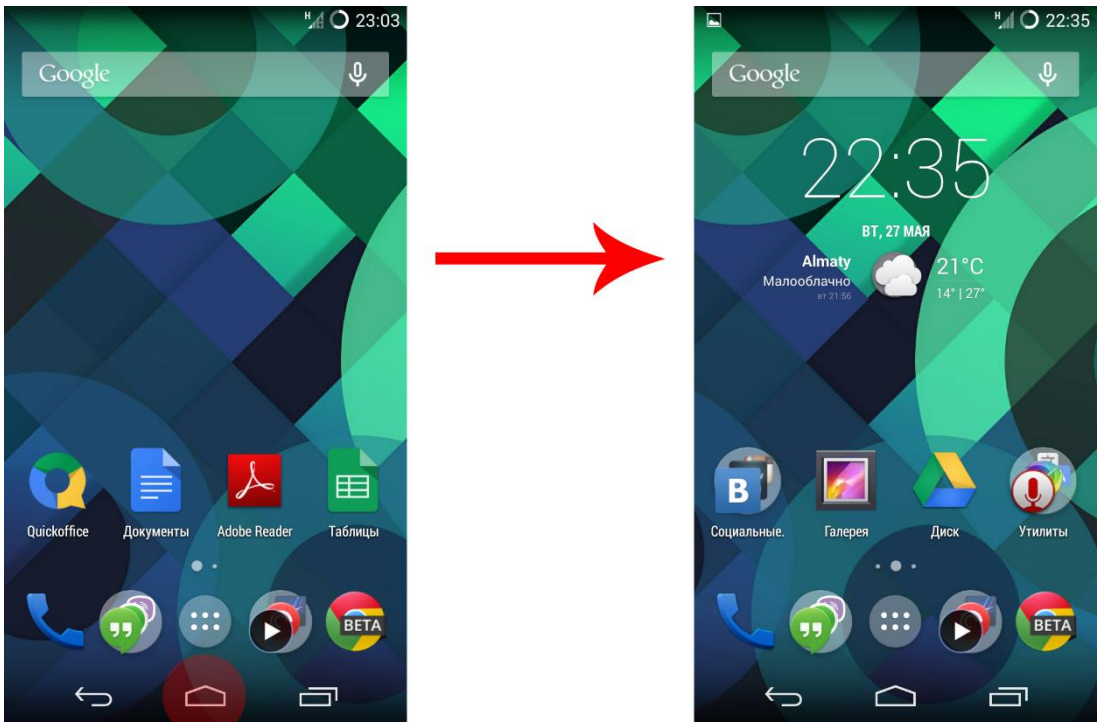

Рисунок 3.13 – Возврат к главному экрану со второстепенного

Переключение между рабочими областями осуществляется проведением пальца в сторону (swipe), влево или право.

На рисунке 3.14 изображен переход от одного экрана к другому «свайпом».

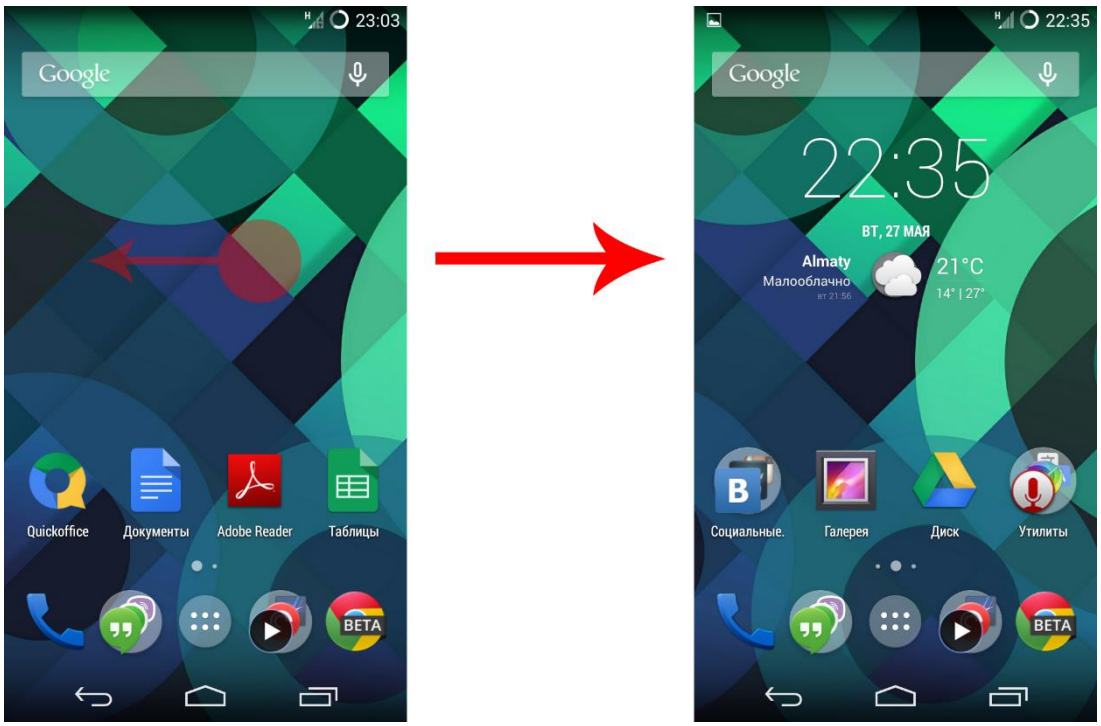

Рисунок 3.14 – Переключение экранов

Для доступа к меню приложений необходимо нажать на соответствующий значок, расположенный на док-панели. При это пользователь попадет на первую страницу меню приложений. Для возврата к рабочему столу необходимо нажать клавишу «домой» или клавишу «назад» (back), в результате чего, он вернется к той рабочей области, находясь на которой был активирован значок меню приложений.

Для доступа к окну настройки необходимо нажать и удержать палец на пустом месте рабочего стола. В этом окне пользователь может:

– Выбрать обои.
– Перейти к меню выбора виджетов.
– Открыть настройки системы.
– Задать экран по умолчанию (главную рабочую область).
– Выбрать эффект перехода между рабочими областями (эффект прокрутки).
– Установить набор значков.

Для возврата из данного меню нужно также нажать «назад» или «домой».

На рисунке 3.15 изображён процесс перехода от рабочего стола к окну настройки.

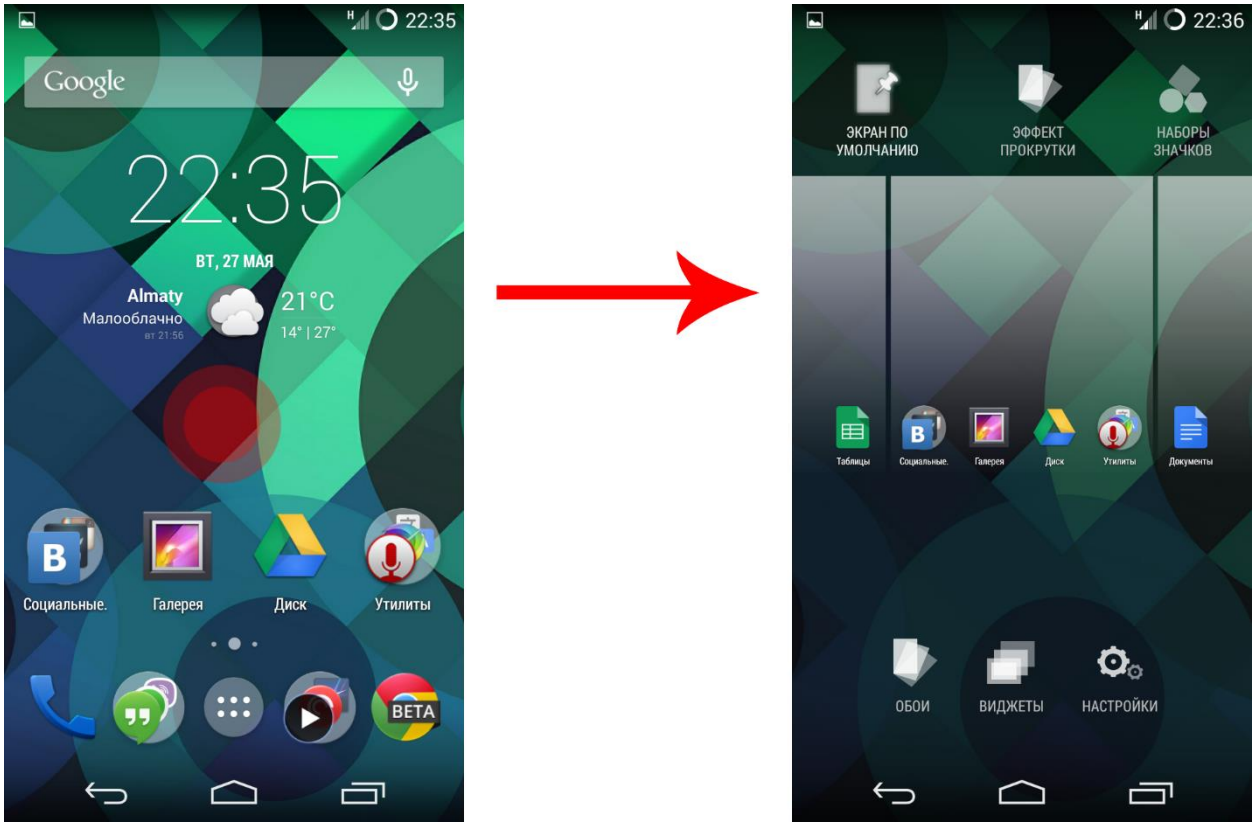

Рисунок 3.15 – Переход к окну настройки

3.4 Уникальные функции

Одними из отличительных особенностей данного приложения являются, упомянутые ранее возможность выбора главной рабочей области, установить эффект прокрутки и применить набор значков, доступные в окне настройки.

В отличии от стандартного домашнего экрана, здесь присутствует возможность выбрать, какой из экранов будет главным, то есть на какой из них вы будете возвращаться каждый раз, закрыв какое-либо приложение.

Еще одной уникальной функцией является возможность изменить анимацию перехода от одной рабочей области к другой. Выбрать можно из 13 вариантов, либо вообще ее отключить. Кроме этого здесь же можно активировать отображение границ рабочих областей и активировать затемнение краев при переходе. На рисунке 3.16 продемонстрированы примеры эффектов переключения между экранами.

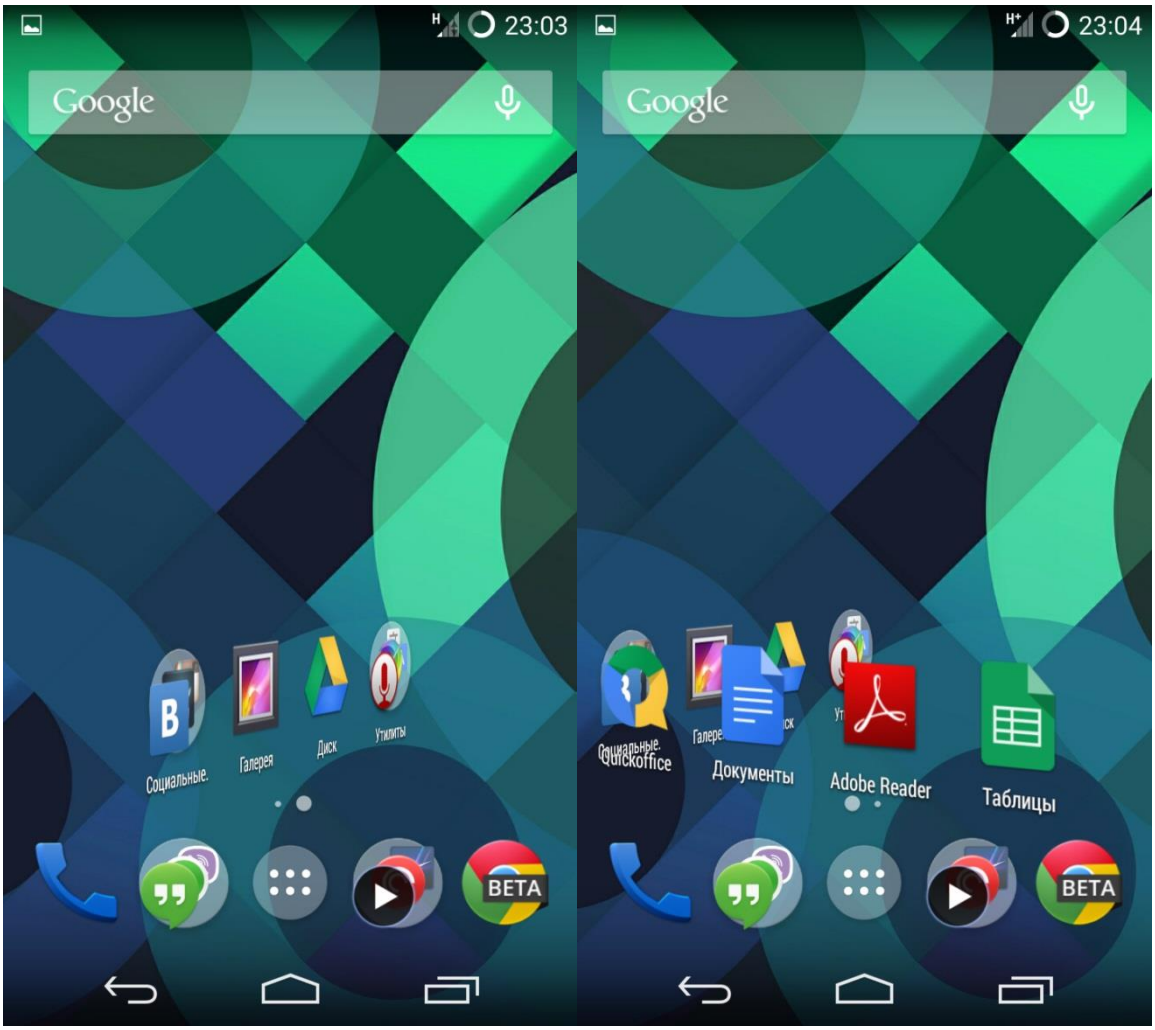

Рисунок 3.16 – Анимация переключения экранов

И наиболее эффектной функцией является возможность заменить стандартные значки приложений, на те, что вам нравятся, подходят по стилю к

обоям и общему оформлению системы. Работа данной функции представлена на рисунке 3.17.

Рисунок 3.17 – Сравнение стандартного и пользовательского набор значков

Остальные же уникальные функции не так зрелищны, но чрезвычайно полезны. Требуют от приложения перерисовки рабочих столов и меню, из-за чего вынесены в меню настроек домашнего экрана. Доступ к ним можно получить выполнив последовательность действий:

1 Настройки системы.

2 Домашний экран.

3 Выбрать ProdLaunch.

4 Нажать «настройки».

Для реализации данного меню был использован следующий код:

```
public class SettingsActivity extends PreferenceActivity
        implements
SharedPreferences.OnSharedPreferenceChangeListener {
        private static final String TAG =
"Launcher3.SettingsActivity";
```

```
        private SharedPreferences mSettings;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            mSettings                                                    =
getSharedPreferences(SettingsProvider.SETTINGS_KEY,
                    Context.MODE_PRIVATE);

            getActionBar().setDisplayHomeAsUpEnabled(true);
        }

        @Override
        public    SharedPreferences    getSharedPreferences(String
name, int mode) {
            return
super.getSharedPreferences(SettingsProvider.SETTINGS_KEY,
                    Context.MODE_PRIVATE);
        }

        @Override
        protected void onResume() {
            super.onResume();

mSettings.registerOnSharedPreferenceChangeListener(this);
        }

        @Override
        protected void onPause() {
            super.onPause();

mSettings.unregisterOnSharedPreferenceChangeListener(this);
        }

        @Override
        public boolean isValidFragment(String fragmentName) {
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
                case android.R.id.home:
                    onBackPressed();
                    finish();
                    return true;
                default:
                    break;
            }
            return super.onOptionsItemSelected(item);
```

```
        }

        @Override
        public void onBuildHeaders(List<Header> target) {
            loadHeadersFromResource(R.xml.preferences_headers,
target);
            updateHeaders(target);
        }

        private void updateHeaders(List<Header> headers) {
            int i = 0;
            while (i < headers.size()) {
                Header header = headers.get(i);

                if                  (header.id                  ==
R.id.preferences_application_version) {
                    header.title                                =
getString(R.string.cm_application_name)     +     "     "     +
getString(R.string.application_version);
                }

                if (headers.get(i) == header) {
                    i++;
                }
            }
        }

        @Override
        public void setListAdapter(ListAdapter adapter) {
            if (adapter == null) {
                super.setListAdapter(null);
            } else {
                List<Header>              headers               =
getHeadersFromAdapter(adapter);
                super.setListAdapter(new      HeaderAdapter(this,
headers));
            }
        }

        @Override
        public  void  onSharedPreferenceChanged(SharedPreferences
sharedPreferences, String key) {
            SharedPreferences.Editor editor = mSettings.edit();
            editor.putBoolean(SettingsProvider.SETTINGS_CHANGED,
true);
            editor.commit();
        }

        private   List<Header>   getHeadersFromAdapter(ListAdapter
adapter) {
            List<Header> headers = new ArrayList<Header>();
            int count = adapter.getCount();
```

```
            for (int i = 0; i < count; i++) {
                headers.add((Header)adapter.getItem(i));
            }
            return headers;
        }

        public static class HomescreenFragment extends
PreferenceFragment {
            @Override
            public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);


addPreferencesFromResource(R.xml.preferences_homescreen);
            }
        }

        public static class GeneralFragment extends
PreferenceFragment {
            @Override
            public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);


addPreferencesFromResource(R.xml.preferences_general);
            }
        }

        public static class DrawerFragment extends
PreferenceFragment {
            @Override
            public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);


addPreferencesFromResource(R.xml.preferences_drawer);
            }
        }

        private static class HeaderAdapter extends
ArrayAdapter<Header> {
            private static final int HEADER_TYPE_NORMAL = 0;
            private static final int HEADER_TYPE_CATEGORY = 1;

            private static final int HEADER_TYPE_COUNT =
HEADER_TYPE_CATEGORY + 1;

            private static class HeaderViewHolder {
                ImageView icon;
                TextView title;
                TextView summary;
            }
```

```
        private LayoutInflater mInflater;

        static int getHeaderType(Header header) {
            if (header.id == R.id.preferences_application_section) {
                return HEADER_TYPE_CATEGORY;
            } else {
                return HEADER_TYPE_NORMAL;
            }
        }

        @Override
        public int getItemViewType(int position) {
            Header header = getItem(position);
            return getHeaderType(header);
        }

        @Override
        public boolean areAllItemsEnabled() {
            return false; // because of categories
        }

        @Override
        public boolean isEnabled(int position) {
            return getItemViewType(position) != HEADER_TYPE_CATEGORY;
        }

        @Override
        public int getViewTypeCount() {
            return HEADER_TYPE_COUNT;
        }

        @Override
        public boolean hasStableIds() {
            return true;
        }

        public HeaderAdapter(Context context, List<Header> objects) {
            super(context, 0, objects);

            mInflater = (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            HeaderViewHolder holder;
```

```
                Header header = getItem(position);
                int headerType = getHeaderType(header);
                View view = null;

                if (convertView == null) {
                    holder = new HeaderViewHolder();
                    switch (headerType) {
                        case HEADER_TYPE_CATEGORY:
                            view   =   new   TextView(getContext(),
null,

android.R.attr.listSeparatorTextViewStyle);
                            holder.title = (TextView) view;
                            break;

                        case HEADER_TYPE_NORMAL:
                            view = mInflater.inflate(

R.layout.preference_header_item, parent,
                                    false);
                            holder.icon       =       (ImageView)
view.findViewById(android.R.id.icon);
                            holder.title = (TextView)

view.findViewById(android.R.id.title);
                            holder.summary = (TextView)

view.findViewById(android.R.id.summary);
                            break;
                    }
                    view.setTag(holder);
                } else {
                    view = convertView;
                    holder = (HeaderViewHolder) view.getTag();
                }

                switch (headerType) {
                    case HEADER_TYPE_CATEGORY:

holder.title.setText(header.getTitle(getContext().getResources()))
;
                        break;

                    case HEADER_TYPE_NORMAL:

holder.icon.setImageResource(header.iconRes);

holder.title.setText(header.getTitle(getContext().getResources()))
;
                        CharSequence           summary           =
header.getSummary(getContext().getResources());
                        if (!TextUtils.isEmpty(summary)) {
```

```
holder.summary.setVisibility(View.VISIBLE);
                                holder.summary.setText(summary);
                            } else {

holder.summary.setVisibility(View.GONE);
                            }
                            break;
                    }

                    return view;
                }
            }
        }
```

Раздел настройки поделен на 3 меню:

– Домашний экран.
– Меню приложений.
– Общие.

В меню «Домашний экран» пользователь может отключить постоянную строку поиска на рабочем столе и крыть названия значков приложений, находящихся на рабочем столе.

В меню «Меню приложений» вынесены настройки для одноименной части ПП. Здесь можно выбрать список скрытых приложений, тех, что будут оставаться установленными в системе, но не будут отображаться в меню приложений. Так, например, если вы пользуетесь сторонним музыкальным проигрывателем, а предустановленный плеер остается без дела, вы можете скрыть его из списка меню приложений, чтобы он занимал пространство в меню. Помимо этого, данная возможность имеет опции в виде возможности убрать ярлыки скрытых приложений с рабочего стола, а также виджеты, относящиеся к ним. Второй возможностью является отключение подписей значков в меню, по аналогии с рабочим столом.

Для реализации функции скрытия используем следующий код:

```
    public class HiddenAppsActivity extends ListActivity {

        private boolean mSaved;

        private static final int MENU_RESET = 0;

        private PackageManager mPackageManager;

        private AppsAdapter mAppsAdapter;

        protected void onCreate(Bundle savedInstanceState) {

requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
            super.onCreate(savedInstanceState);
```

```
            setTitle(R.string.hidden_apps_title);
            setContentView(R.layout.hidden_apps_list);

            getActionBar().setDisplayHomeAsUpEnabled(true);
            setProgressBarIndeterminateVisibility(true);
            setProgressBarIndeterminate(true);

            mPackageManager = getPackageManager();
            mAppsAdapter       =       new       AppsAdapter(this,
R.layout.hidden_apps_list_item);
            mAppsAdapter.setNotifyOnChange(true);

            setListAdapter(mAppsAdapter);

            AsyncTask<Void, Void, List<AppEntry>> refreshAppsTask
= new AsyncTask<Void, Void, List<AppEntry>>() {

                @Override
                protected void onPostExecute(List<AppEntry> apps)
{
                    mAppsAdapter.clear();
                    mAppsAdapter.addAll(apps);
                    restoreCheckedItems();
                    setProgressBarIndeterminateVisibility(false);
                    setProgressBarIndeterminate(false);
                }

                @Override
                protected  List<AppEntry>  doInBackground(Void...
params) {
                    return refreshApps();
                }
            };
            refreshAppsTask.execute(null, null, null);
        }

        @Override
        public void onPause() {
            super.onPause();
            save();
        }

        private void save() {
            if (mSaved) {
                return;
            }
            String string = "";

            SparseBooleanArray              checked              =
getListView().getCheckedItemPositions();
```

```
            AppsAdapter      listAdapter      =      (AppsAdapter)
getListAdapter();
            for (int i = 0; i < checked.size(); i++) {
                if (checked.valueAt(i)) {
                    AppEntry                app                 =
listAdapter.getItem(checked.keyAt(i));
                    if (!string.isEmpty())
                        string += "|";
                    string                                      +=
app.componentName.flattenToString();
                }
            }

            SharedPreferences.Editor           editor           =
SettingsProvider.get(this).edit();

editor.putString(SettingsProvider.SETTINGS_UI_DRAWER_HIDDEN_APPS,
string);
            editor.putBoolean(SettingsProvider.SETTINGS_CHANGED,
true);
            editor.apply();

            mSaved = true;
        }

        private void restoreCheckedItems() {
            List<ComponentName>         apps         =         new
ArrayList<ComponentName>();
            String[]                  flattened                 =
SettingsProvider.getStringCustomDefault(this,

SettingsProvider.SETTINGS_UI_DRAWER_HIDDEN_APPS, "").split("\\|");
            for (String flat : flattened) {

apps.add(ComponentName.unflattenFromString(flat));
            }

            AppsAdapter      listAdapter      =      (AppsAdapter)
getListAdapter();

            for (int i = 0; i < listAdapter.getCount(); i++) {
                AppEntry info = listAdapter.getItem(i);
                if (apps.contains(info.componentName)) {
                    getListView().setItemChecked(i, true);
                }
            }

            mSaved = true;
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
```

```
            menu.add(0,                MENU_RESET,                0,
R.string.menu_hidden_apps_delete)

.setShowAsActionFlags(MenuItem.SHOW_AS_ACTION_IF_ROOM             |
MenuItem.SHOW_AS_ACTION_WITH_TEXT);

            return true;
        }

        private void reset() {
            for (int i = 0; i < getListView().getCount(); i++) {
                getListView().setItemChecked(i, false);
            }

            mSaved = false;
        }

        private List<AppEntry> refreshApps() {
            Intent  mainIntent  =  new  Intent(Intent.ACTION_MAIN,
null);
            mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
            List<ResolveInfo>                apps                =
mPackageManager.queryIntentActivities(mainIntent, 0);
            Collections.sort(apps,                                 new
ResolveInfo.DisplayNameComparator(mPackageManager));
            List<AppEntry>        appEntries        =        new
ArrayList<AppEntry>(apps.size());
            for (ResolveInfo info : apps) {
                appEntries.add(new AppEntry(info));
            }
            return appEntries;
        }

        @Override
        public boolean onMenuItemSelected(int featureId, MenuItem
item) {
            if (item.getItemId() == android.R.id.home) {
                finish();
                return true;
            } else if (item.getItemId() == MENU_RESET) {
                reset();
                return true;
            }
            return super.onMenuItemSelected(featureId, item);
        }

        @Override
        protected  void  onListItemClick(ListView  l,  View  v,  int
position, long id) {
            mSaved = false;
        }
```

```
        private final class AppEntry {

            public final ComponentName componentName;
            public final String title;

            public AppEntry(ResolveInfo info) {
                componentName                =                new
ComponentName(info.activityInfo.packageName,
info.activityInfo.name);
                title                                          =
info.loadLabel(mPackageManager).toString();
            }
        }

        private static class AppViewHolder {
            public final TextView title;
            public final ImageView icon;

            public AppViewHolder(View parentView) {
                icon                =                (ImageView)
parentView.findViewById(R.id.icon);
                title                 =                (TextView)
parentView.findViewById(R.id.title);
            }
        }

        public class AppsAdapter extends ArrayAdapter<AppEntry> {

            private final LayoutInflater mInflator;

            private ConcurrentHashMap<String, Drawable> mIcons;
            private Drawable mDefaultImg;
            private List<AppEntry> mApps;

            public      AppsAdapter(Context      context,      int
textViewResourceId) {
                super(context, textViewResourceId);

                mApps = new ArrayList<AppEntry>();

                mInflator = LayoutInflater.from(context);

                mDefaultImg                                        =
context.getResources().getDrawable(android.R.mipmap.sym_def_app_ic
on);
                mIcons     =     new     ConcurrentHashMap<String,
Drawable>();
            }

            @Override
            public   View   getView(final   int   position,   View
convertView, ViewGroup parent) {
```

```
                AppViewHolder viewHolder;

                if (convertView == null) {
                    convertView                                    =
mInflator.inflate(R.layout.hidden_apps_list_item, parent, false);
                    viewHolder = new AppViewHolder(convertView);
                    convertView.setTag(viewHolder);
                } else {
                    viewHolder          =          (AppViewHolder)
convertView.getTag();
                }

                AppEntry app = getItem(position);

                viewHolder.title.setText(app.title);

                Drawable                    icon                    =
mIcons.get(app.componentName.getPackageName());
                viewHolder.icon.setImageDrawable(icon  !=  null  ?
icon : mDefaultImg);

                return convertView;
            }

            @Override
            public boolean hasStableIds() {
                return true;
            }

            @Override
            public void notifyDataSetChanged() {
                super.notifyDataSetChanged();
                List<AppEntry>        newApps        =        new
ArrayList<AppEntry>(getCount());
                List<AppEntry>        oldApps        =        new
ArrayList<AppEntry>(getCount());
                for (int i = 0; i < getCount(); i++) {
                    AppEntry app = getItem(i);
                    if (mApps.contains(app)) {
                        oldApps.add(app);
                    } else {
                        newApps.add(app);
                    }
                }

                if (newApps.size() > 0) {
                    new
LoadIconsTask().execute(newApps.toArray(new AppEntry[] {}));
                    newApps.addAll(oldApps);
                    mApps = newApps;
                } else {
                    mApps = oldApps;
```

```
                }
            }

            private class LoadIconsTask extends
AsyncTask<AppEntry, Void, Void> {
                @Override
                protected Void doInBackground(AppEntry... apps) {
                    for (AppEntry app : apps) {
                        try {
                            if
(mIcons.containsKey(app.componentName.getPackageName())) {
                                continue;
                            }
                            Drawable icon =
mPackageManager.getApplicationIcon(app.componentName
                                    .getPackageName());

mIcons.put(app.componentName.getPackageName(), icon);
                            publishProgress();
                        } catch
(PackageManager.NameNotFoundException e) {

                        }
                    }

                    return null;
                }

                @Override
                protected void onProgressUpdate(Void... progress)
{
                    notifyDataSetChanged();
                }
            }
        }
    }
```

И последнее меню – «Общие», здесь расположены настройки, изменяющие внешний вид как рабочего стола, так и меню приложений. Если снять отметку с пункта «Крупные значки», то все значки, отображаемые на домашнем экране уменьшатся до размера, который был стандартом для старых версий ОС Android. Также здесь можно выбрать тип шрифта, который будет использоваться в разрабатываемом приложении.

Функция смены шрифта реализуется следующим образом:

```
public class FontStylePreference extends ListPreference {
    private String mValue;
    public int mClickedDialogEntryIndex;
    private boolean mValueSet;
```

```
        public FontStylePreference(Context context, AttributeSet
attrs) {
            super(context, attrs);
        }

        @Override
        protected void onPrepareDialogBuilder(AlertDialog.Builder
builder) {
            if (getEntries() == null || getEntryValues() == null)
{
                throw new IllegalStateException(
                        "ListPreference requires an entries array
and an entryValues array.");
            }

            mClickedDialogEntryIndex = getValueIndex();
            builder.setSingleChoiceItems(new
FontStyleAdapter(getContext()), mClickedDialogEntryIndex,
                    new DialogInterface.OnClickListener() {
                        public    void    onClick(DialogInterface
dialog, int which) {
                            mClickedDialogEntryIndex = which;


FontStylePreference.this.onClick(dialog,
DialogInterface.BUTTON_POSITIVE);
                            dialog.dismiss();
                        }
                    });

            builder.setPositiveButton(null, null);
        }

        @Override
        protected void onDialogClosed(boolean positiveResult) {
            super.onDialogClosed(positiveResult);

            if (positiveResult && mClickedDialogEntryIndex >= 0
&& getEntryValues() != null) {
                String                    value                =
getEntryValues()[mClickedDialogEntryIndex].toString();
                if (callChangeListener(value)) {
                    setValue(value);
                }
            }
        }

        @Override
        protected  void  onSetInitialValue(boolean  restoreValue,
Object defaultValue) {
            setValue(restoreValue  ?  getPersistedString(mValue)  :
(String) defaultValue);
```

```
        }

        @Override
        public void setValue(String value) {
            // Always persist/notify the first time.
            final boolean changed = !TextUtils.equals(mValue,
value);
            if (changed || !mValueSet) {
                mValue = value;
                mValueSet = true;
                persistString(value);
                if (changed) {
                    notifyChanged();
                }
            }
        }
        @Override
        public String getValue() {
            return mValue;
        }

        private int getValueIndex() {
            return findIndexOfValue(mValue);
        }
        private class FontStyleAdapter extends
ArrayAdapter<CharSequence> {
            private LayoutInflater mInflater;
            private List<CharSequence> mEntries;
            private List<CharSequence> mValues;

            public FontStyleAdapter(Context context) {
                super(context, -1, getEntryValues());

                mInflater = LayoutInflater.from(context);
                mEntries = Arrays.asList(getEntries());
                mValues = Arrays.asList(getEntryValues());
            }

            @Override
            public View getView(int position, View convertView,
ViewGroup parent) {
                CheckedTextView textView;

                if (convertView == null) {
                    textView = (CheckedTextView)
mInflater.inflate(R.layout.list_item_checkable, parent, false);
                } else {
                    textView = (CheckedTextView) convertView;
                }
                if (textView != null) {
                    textView.setText(mEntries.get(position));
                    textView.setTag(mValues.get(position));
```

```
                    textView.setTypeface(Typeface.create((String)
mValues.get(position), Typeface.NORMAL));
                }
                return textView;
            }
        }
    }
```

4 Технико-экономическое обоснование

4.1 Цель проекта

Целью данного дипломного проекта является создание мобильного приложения ProdLaunch для операционной системы Android, являющегося настраиваемым пользовательским интерфейсом для запуска приложений на смартфоне или же планшете, соответствующего всем «гайдлайнам» Android OS. Разработка приложения производится в среде Android Studio на языке Java. Графические элементы приложения создаются в Adobe Illustrator CC.

4.2 Трудовые ресурсы, используемые в работе

В работе над данным проектом задействованы:

– Руководитель проекта – постановка задачи и разработка схемы ПП.
– Дизайнер – создание интерфейса приложения.
– Программист – разработка алгоритмов и программирование.

Количество сотрудников, задействованных в проекте, и их месячная заработная плата представлено в таблице 4.1.

Т а б л и ц а 4.1 – Данные о сотрудниках

Должность	Количество	Заработная плата в месяц
Руководитель проекта	1	100 000
Дизайнер	1	100 000
Программист	1	120 000
Итого	3	360 000

4.3 Оборудование, используемое в работе

Характеристики оборудования, используемого в работе, а также его стоимость приведены в таблице 4.2.

Т а б л и ц а 4.2 – Перечень оборудования

Название оборудования	Характеристики	Количество	Стоимость за единицу с учетом НДС, тенге
Ноутбук HP Pavilion 15-e082er	Intel Core i5 3230M, 6 Gb DDR3, 1000 Gb HDD, Radeon HD 7670M	1	112 810

Продолжение таблицы 4.2

Название оборудования	Характеристики	Количество	Стоимость за единицу с учетом НДС, тенге
Ноутбук HP Pavilion 15-n060er	Intel Core i5-4200U, 8 Gb DDR3, 1000 Gb HDD, GeForce GT 740M	1	126 110
Смартфон LG Nexus 5 (LG-D821)	16 Gb ROM, 5.0" Full HD IPS 1920x1080 , 2 Gb RAM, Qualcomm Snapdragon 800	1	96 270
Планшет Asus Google Nexus 7 Второе поколение	16 Gb ROM, 7.0" Full HD IPS 1920x1080 , 2 Gb RAM, Qualcomm Snapdragon S4 Pro	1	47 620
Итого		4	335857

4.4 Программное обеспечение, используемое в работе

При разработке приложения используется данное следующее программного обеспечения:

– Windows 8.1 – операционная система.
– Adobe Illustrator CC – векторный графический редактор.
– Android Studio – среда для разработки.
– Java Development Kit – комплект разработчика приложений.

ПО и соответствующая ему стоимость приведены в таблице 4.3.

Т а б л и ц а 4.3 – Программное обеспечение, использованное в работе

Наименование	Количество копий	Стоимость с учетом НДС, тенге
Windows 8.1	2	Предустановлено, цена ПО входит в стоимость ноутбука
Adobe Illustrator CC	1	37 593 (Годовая подписка)
Android Studio	1	Бесплатно
Java Development Kit	1	Бесплатно
Итого		37 593

4.5 Сроки реализации проекта

Процесс проектирования и разработки приложения включает в себя 5 этапов:

– Постановка задачи и сбор данных.
– Разработка дизайна интерфейса приложения.
– Разработка форм приложения.
– Тестирование на работоспособность.
– Оформление и сдача отчета.

В таблице 4.4 приведен график реализации проекта.

Т а б л и ц а 4.4 – Этапы и сроки реализации проекта

Перечень работ		Неделя от начала работ							
		1	2	3	4	5	6	7	8
1 этап	Постановка задачи								
	Выбор среды разработки								
	Изучение литературы								
	Изучение «гайдлайнов»								
2 этап	Разработка интерфейса								
	Создание необходимых графических элементов								
3 этап	Разработка форм приложения								
4 этап	Тестирование приложения								
	Отладка приложения								
5 этап	Оформление и сдача отчета								

4.6 Расчет затрат и стоимости работ по реализации проекта

Разработка мобильного приложения требует большого количества интеллектуальных затрат сотрудников, выполняющих работу, а также необходимых технических средств для ее реализации. Все это требует

финансовых вложений, на основе которых высчитывается конечная стоимость проекта.

Затраты на разработку данного приложения вычисляются по формуле 4.1

$$С = ФОТ + С_н + А + Э + Н \quad (4.1)$$

где ФОТ – фонд оплаты труда;
$С_н$ – социальный налог;
А – амортизационные отчисления;
Э – затраты на электроэнергию;
$С_{пр}$ – прочие расходы;
Н – накладные расходы.

4.6.1 Расчет фонда оплаты труда

ФОТ складывается из основной и дополнительной заработной платы сотрудников и рассчитывается по формуле 4.2

$$ФОТ = З_{осн} + З_{доп} \quad (4.2)$$

где $З_{осн}$ – основная заработная плата;
$З_{доп}$ – дополнительная заработная плата.

Для расчета затрат на основную заработную плату используются данные о среднедневном заработке и фактическом времени работы каждого сотрудника.

Среднедневной заработок рассчитывается по формуле 4.3

$$D = \frac{ЗПм}{Др} \quad (4.3)$$

где ЗПм – ежемесячный размер заработной платы;
Др – количество рабочих дней в месяце (21 день).

1) Руководитель проекта:

$$D = \frac{100000}{21} = 4762 \text{ тенге/день};$$

2) Дизайнер:

$$D = \frac{100000}{21} = 4762 \text{ тенге/день};$$

3) Программист:

$$D = \frac{120000}{21} = 5714 \text{ тенге/день.}$$

Заработная плата за один час работы сотрудника рассчитывается по формуле 4.4

$$\text{Н} = \frac{D}{\text{Чр}} \quad (4.4)$$

где D – средний дневной заработок работника;
Чр – количество часов рабочего дня (8 часов).

Руководитель проекта:

$$H = \frac{4762}{8} = 595 \text{ тенге/час;}$$

Дизайнер:

$$H = \frac{4762}{8} = 595 \text{ тенге/час;}$$

Программист:

$$H = \frac{5714}{8} = 714 \text{ тенге/час.}$$

Длительность цикла в днях по каждому виду работ определяется по формуле 4.5

$$t_n = \frac{T}{q_n * z * K} \quad (4.5)$$

где T – трудоемкость этапа, норма-час;
q_n – количество исполнителей по этапу;
z –продолжительность рабочего дня, z = 8 часов;
К – коэффициент выполнения норм времени, К = 1,1.
Полученную величину t_nокругляем в большую сторону до целых дней.

$t_1 = \frac{16}{1*8*1,1} \approx 2$ дня; Руководитель проекта, постановка задачи;

$t_2 = \frac{16}{1*8*1,1} \approx 2$ дня; Руководитель проекта, изучение литературы;

$t_3 = \frac{16}{1*8*1,1} \approx 2$ дня; Программист, изучение литературы;

$t_4 = \frac{8}{1*8*1,1} \approx 1$ день; Программист, выбор среды разработки;

$t_5 = \frac{24}{1*8*1,1} \approx 3$ дня; Дизайнер, изучение «гайдлайнов»;

$t_6 = \frac{48}{1*8*1,1} \approx 6$ дней; Дизайнер, разработка интерфейса;

$t_7 = \frac{56}{1*8*1,1} \approx 7$ дней; Дизайнер, создание необходимых графических элементов;

$t_8 = \frac{96}{1*8*1,1} \approx 12$ дней; Программист, разработка форм приложения;

$t_9 = \frac{16}{1*8*1,1} \approx 2$ дня; Программист, тестирование приложения;

$t_{10} = \frac{16}{1*8*1,1} \approx 2$ дня; Программист, отладка приложения;

$t_{11} = \frac{8}{1*8*1,1} \approx 1$ день; Руководитель проекта, оформление и сдача отчета.

В таблице 4.5 приведены сводные результаты расчета затрат на основную заработную плату сотрудников.

Таблица 4.5 – Сводные результаты расчета затрат на основную заработную плату

Наименование этапов работ	Исполнитель	Трудоемкость		Длительность цикла, дни	Зар. плата за час работы, тенге	Сумма зар. платы, тенге
		Нормы-часы	% от общей трудоемкости			
Постановка задачи	Руководитель проекта	16	5	2	595	9520
Изучение литературы	Руководитель проекта	16	5	2	595	9520

Продолжение таблицы 4.5

Наименование этапов работ	Исполнитель	Трудоемкость		Наименование этапов работ	Зар. плата за час работы, тенге	Сумма зар. платы, тенге
		Нормы-часы	% от общей трудоемкости			
Изучение литературы	Программист	16	5	2	714	11424
Выбор среды разработки	Программист	8	2,5	1	714	5712
Изучение «гайдлайнов»	Дизайнер	24	7,5	3	595	14280
Разработка интерфейса	Дизайнер	48	15	6	595	28560
Создание необходимых графических элементов	Дизайнер	56	17,5	7	595	33320
Разработка форм приложения	Программист	96	30	12	714	68544
Тестирование приложения	Программист	16	5	2	714	11424
Отладка приложения	Программист	16	5	2	714	11424
Оформление и сдача отчета	Руководитель проекта	8	2,5	1	595	4760
Итого		320	100	40		205488

Дополнительная заработная плата составляет 10% от основной заработной платы и рассчитывается по формуле 4.6

$$З_{доп} = З_{осн} * 0,1 \quad (4.6)$$

$$З_{доп} = 205488 * 0,1 = 20548,8 \text{ тенге}$$

В результате расчетов, согласно формуле 3.2, суммарный фонд оплаты труда составит:

$$ФОТ = 205488 + 20548{,}8 = 226036{,}8 \text{ тенге}$$

4.6.2 Расчет затрат по социальному налогу

Социальный налог составляет 11% от дохода сотрудника и рассчитывается по формуле 4.7

$$С_н = (ФОТ - ПО) * 11\% \quad (4.7)$$

где ПО – пенсионные отчисления, который составляют 10% от ФОТ и не облагаются социальным налогом, рассчитываются по формуле 4.8

$$ПО = ФОТ * 10\% \quad (4.8)$$

$$ПО = 226036{,}8 * 0{,}1 = 22603{,}68 \text{ тенге.}$$

Таким образом социальный налог составит:

$$С_н = (226036{,}8 - 22603{,}68) * 0{,}11 = 22377{,}64 \text{ тенге.}$$

4.6.3 Расчет амортизационных отчислений

Амортизационные отчисления рассчитываются по формуле 4.9

$$А_i = \frac{Н_А * С_{пер} * N}{100 * 12 * n} \quad (4.9)$$

где $Н_А$ – норма амортизации;

$С_{пер}$ - первоначальная стоимость оборудования;

N – количество дней на выполнение работ;

n – количество рабочих дней в месяце.

Следовательно, амортизационные отчисления по используемому оборудованию и ПО, в соответствии с формулой 3.9 составят:

1 На оборудование:

$$А_1 = \frac{40 * 335857 * 40}{100 * 12 * 21} = 21324 \text{ тенге}$$

2 На программное обеспечение:

$$A_2 = \frac{40*37593*40}{100*12*21} = 2387 \text{ тенге}$$

Суммарные затраты на амортизацию рассчитываются по формуле 4.10

$$A = A_1 + A_2 \quad (4.10)$$

A = 21324 + 2387 = 23711 тенге

4.6.4 Расчет затрат на электроэнергию

Так как в процессе реализации проекта используется техническое оборудование, необходимо рассчитать затраты на электроэнергию, потребляемую данных оборудованием.

Для расчета затрат на электроэнергию используется формула 4.11

$$Э = З_{эл.эн.об.} + З_{доп.} \quad (4.11)$$

где $З_{эл.эн.об.}$ – затраты на электроэнергию для оборудования;

$З_{доп.}$ – затраты электроэнергии на дополнительные нужды;

Расходы электроэнергии на оборудование рассчитываются по формуле 4.12.

$$З_{эл.эн.об.} = W * T * S * K_{исп} \quad (4.12)$$

где W – потребляемая оборудованием мощность, кВт;

T – время работы, часы;

S – тариф (1кВт/час = 16,9 тенге);

$K_{исп}$ – коэффициент использования ($K_{исп}$ = 0,9).

Потребляемая мощность HP Pavilion 15-e082er составляет 90 Вт.

Потребляемая мощность HP Pavilion 15-n060er составляет 65 Вт.

Потребляемой мощностью адаптеров для зарядки смартфона и планшета пренебрегаем, т.к. они используются исключительно для тестирования и отладки приложения, при этом постоянно подключены к ноутбуку HP Pavilion 15-e082er.

Время высчитывается на основе количества рабочих дней и рабочих часов в день.

Таким образом общая сумма затрат на электроэнергию для оборудования:

$$З_{эл.эн.об.} = (0{,}09 + 0{,}065) * (40 * 8) * 16{,}9 * 0{,}9 = 754{,}4 \text{ тенге.}$$

Затраты на дополнительные нужды берутся в размере 5% от затрат на оборудование и рассчитываются по формуле 4.13

$$З_{доп.} = З_{эл.эн.об.} * 5\% \quad (4.13)$$

И составляют:

$$З_{доп.} = 754{,}4 * 0{,}05 = 37{,}7 \text{ тенге.}$$

Суммарные затраты на электроэнергию составляют:

Э = 754,4 + 37,7 = 792,1 тенге.

4.6.5 Расчет накладных расходов

Накладные расходы рассчитываются в размере 50% от всех затрат по формуле 4.14

$$Н = (ФОТ + О_с + А + Э) * 50\% \quad (4.14)$$

$$Н = (226036{,}8 + 22377{,}64 + 23711 + 792{,}1) * 0{,}5 = 136458{,}77 \text{ тенге.}$$

4.6.6 Суммарные затраты на реализацию проекта

Таким образом себестоимость разработки данного приложения, согласно формуле 3.1 составляет:

$$С = 226036{,}8 + 22377{,}64 + 23711 + 792{,}1 + 136458{,}77 = 409376{,}31 \text{ тенге.}$$

Сводные результаты расчета стоимости разработки приложения и их структура представлены на рисунке 4.1 и в таблице 4.6

Рисунок 4.1 – Затраты на разработку приложения

Т а б л и ц а 4.6 – Затраты на разработку приложения

Наименование затрат	Сумма, тенге
ФОТ	226036,8
Социальный налог	22377,64
Амортизационные отчисления	23711
Затраты на электроэнергию	792,1
Накладные расходы	136458,77
Итого	409376,31

4.6.7 Цена реализации проекта

Цена проекта складывается из себестоимости и желаемого чистого дохода, по формуле 4.15

$$Ц_п = С + П \quad (4.15)$$

где С – стоимость приложения;

П – чистый доход.

Для определения начальной цены используется желаемый уровень рентабельности. Для данной отрасли он составляет 25% и рассчитывается по формуле 4.16

$$Ц_п = С * \left(1 + \frac{Р}{100}\right) \quad (4.16)$$

где Р – рентабельность.

$$Ц_п = 409376{,}31 * \left(1 + \frac{25}{100}\right) = 511720{,}39$$

Цена реализации проекта складывает из цены проект и НДС, рассчитывается по формуле 4.17

$$Ц_р = Ц_п + НДС \quad (4.17)$$

где НДС – налог на добавленную стоимость.

Согласно Налоговому Кодексу Республики Казахстан НДС составляет 12%, то есть в данном случае равен:

НДС = 61406,45 тенге

В итоге получаем цену реализации проекта равной:

$$Ц_р = 511720{,}39 + 61406{,}45 = 573126{,}84 \text{ тенге.}$$

5 Безопасность жизнедеятельности

5.1 Анализ условий труда при разработке мобильного приложения

Разработка приложения осуществляется с использованием компьютерной техники и электронного оборудования. В рассматриваемом помещении работает 3 сотрудника, двое из которых имею свое рабочее место.

При разработке прикладного приложения большую роль играет правильная организация условий труда в рабочем помещении. Так как данная работа связана с длительным нахождением за компьютером, необходимо учесть нормы естественного и искусственного освещения, для обеспечения благоприятных условий работы.

Характеристики рабочего помещения

Рассматриваемое рабочее помещение, расположено в здании, находящемся вдали от железнодорожных путей или нагруженных автомагистралей, аэропорта и так далее, поэтому внешних источников шума, влияющих на процесс работы – нет.

План помещения представлен на рисунке 5.1.

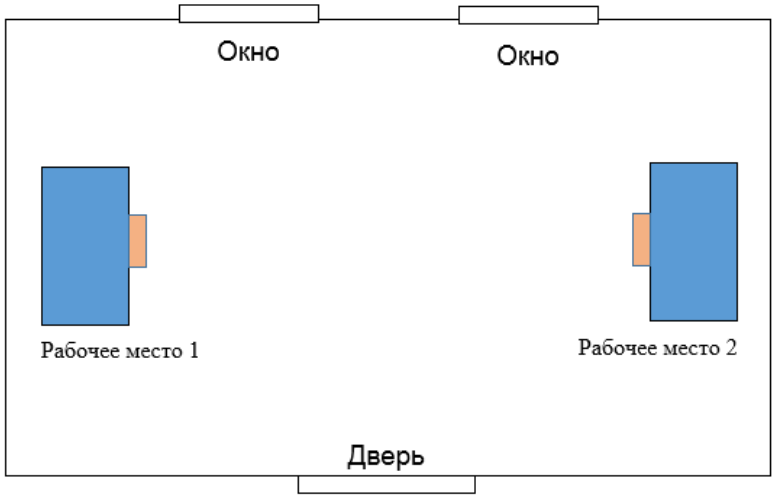

Рисунок 5.1 – План помещения

Помещение имеет следующие параметры:

- Находится на первом этаже одноэтажного здания.
- Размеры помещения (комнаты): длина 6 м, ширина 4 м, высота 3м.
- Вид светопропускающего материала – стекло листовое, двойное.
- Вид переплета – стальные, двойные, открывающиеся.
- Солнцезащитные устройства – убирающиеся регулируемые жалюзи.
- Два окна размером 1,5*1,2.

– Внутренняя отделка стен – светлая.

– Помещение по зрительным условиям работы относится к категории легких работ (легкая физическая, категория Iа, работа производится сидя и не требует физического напряжения).

– Искусственное освещение – 2 светильника с двумя люминесцентными лампами.

Характеристики используемого в работе оборудования:

– Ноутбук HP Pavilion 15-e082er, Intel Core i5 3230M, 6 Gb DDR3, 1000 Gb HDD, Radeon HD 7670M.

– Ноутбук HP Pavilion 15-n060er, Intel Core i5-4200U, 8 Gb DDR3, 1000 Gb HDD, GeForce GT 740M.

– Электропитание: переменное напряжение 220 – 250 В, частотой 50 Гц и мощностью 90 Вт и 65 Вт соответственно.

– 2 светильника, 4 люминесцентные лампы.

– Электропитание: переменное напряжение 220 – 250 В, частотой 50 Гц, мощность светильника 2х28 Вт.

Электротехническое оборудование является потенциальным источником возникновения пожарной опасности. Оборудование малошумящее – вредность в качестве повышенного шума отсутствует.

5.2 Расчет естественного освещения

Естественное освещение не обеспечивает в течение всего рабочего времени необходимого освещения, так как может измениться погода, либо работы могут быть в позднее время, когда уже темнеет и естественного освещения может быть не достаточно, поэтому в рабочем помещении предусмотрена система искусственного общего освещения, состоящая из светильников с люминесцентными лампами. Нормативы на источники света приведены в таблице 5.1.

Т а б л и ц а 5.1 – Рекомендуемые источники света при системе общего освещения.

Характеристика зрительной работы по требованию к цветоразличию	Освещенность, лк	Диапазон цветов температуры источника света T_C, ^0K	Применяемый тип источника света
Различие цветных объектов при невысоких требованиях к цветоразличию	300, 400	3500 – 5500	ЛД, ЛДЦ, ЛБ,
	150, 200	3000 – 4500	ЛБ, (ЛХБ), НЛВД+МТЛ, ДРЛ

Освещённость, необходимая для нормального выполнения работ в данном помещении: 400 лк. Для этого в помещении используются 4 люминесцентные лампы белого цвета. Этот выбор обусловлен тем, что люминесцентные лампы более экономичны, чем обычные лампы накаливания. Известно, что различный спектральный состав по-разному влияет на настроение человека. Лампы “теплого” цвета способствуют расслаблению, их лучше использовать в спальнях, местах для отдыха. Лампы “холодного” света способствуют зрительному и психоэмоциональному комфорту, их лучше использовать в гостиных, местах приема гостей и кухнях. Лампы “дневного” света способствуют концентрации внимания и увеличения работоспособности, их лучше использовать в кабинетах, местах, предназначенных для умственной работы. Так как лампы будут использоваться в рабочем помещении, были выбраны люминесцентные лампы белого цвета ЛД-40.

Здание относится к I степени огнестойкости (Здания с несущими и ограждающими конструкциями из естественных или искусственных материалов, бетона или железобетона с применением листовых негорючих материалов). Рабочее помещение по вопросам пожарной безопасности относится к классу “Д”. В соответствии с типовыми правилами пожарной безопасности административные здания и отдельные помещения, и технологические установки обеспечиваются первичными средствами пожаротушения согласно нормативам.

Рассчитаем площадь боковых световых проемов помещения, необходимую для создания нормируемой освещенности на рабочем месте.

Помещение имеет размеры: длина $L = 6\,м$, ширина $B = 4\,м$, высота $H = 3\,м$. Высота рабочей поверхности над уровнем пола $h_{рп}$, $h_{рп} = 0{,}725\,м$, окно начинаются с высоты $h_{но}$, $h_{но} = 0{,}8м$, высота окна $h_о$, $h_о = 1{,}5м$. Рабочее помещение находится в IV часовом поясе – в г. Алматы (пояс светового климата – IV 50^0 северной широты и южнее).

Рабочие места расположены в $l_{рт}$, $l_{рт} = 0{,}5\,м$ от наружной стены помещения, где проектируем оконные проемы. Минимальная освещенность будет в точке, отстоящей на расстояние 4 м от оконного проема.

Общую площадь окон S_0, $м^2$, определим по формулам 5.1 и 5.2

$$100 \cdot \frac{S_0}{S_п} = \frac{e_н \cdot \eta_0}{\tau_0 \cdot r_1} \cdot k_{зд} \cdot k_з \qquad (5.1)$$

$$S_0 = \frac{S_п \cdot e_н \cdot \eta_0}{100 \cdot \tau_0 \cdot r_1} \cdot k_{зд} \cdot k_з \qquad (5.2)$$

где: $S_п$ – площадь помещения $м^2$;

$S_п = 24\ м^2$;

$e_н$ – нормированное значение КЕО, выбираемое по таблице 5.1.

Для высокой точности зрительных работ принимаем $e_H = 1{,}2$

m_N – коэффициент светового климата, который находится по таблице 5.2.

Учитывая заданный световой пояс, приняв ориентацию световых проемов на Север, определим по формуле 5.3, при m_N=0,9.

$$e_н^{IV} = e_н \cdot m \cdot c \qquad (5.3)$$

где $m = 0{,}7$;

$c = 0{,}75$ (в наружных стенах зданий);

$e_н = 1{,}2$ для работ высокой точности III разряда зрительной работы;

$e_н^{IV} = 1{,}2 \cdot 0{,}7 \cdot 0{,}75 = 0{,}63$;

Учитывая тип помещения, найдем коэффициент запаса по таблице 5.3 $k_з = 1{,}2$ (учебные помещения, лаборатории, конструкторские бюро);

τ_0 – общий коэффициент светопропускания равный $\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4$;

$\tau_1 = 0{,}5$ (пустотелые стеклянные блоки);

$\tau_2 = 0{,}6$ (деревянные двойные раздельные переплеты);

$\tau_3 = 0{,}8$ (железобетонные фермы и арки);

$\tau_4 = 1$ (убирающиеся регулируемые жалюзи и шторы);

$\tau_0 = 0{,}5 \cdot 0{,}6 \cdot 0{,}8 \cdot 1 = 0{,}24$;

η_0 – световая характеристика окон.

Определяем η_0 по формуле 5.4

$$l = B - 1$$

$$l = 4 - 1 = 3 \text{ м}$$

$$\frac{L}{l} = \frac{L}{B-1} = \frac{6}{3} = 2$$

$$h_{расч} = h_{но} + h_о - h_{рп} \qquad (5.4)$$

$$h_{расч} = 0{,}8 + 1{,}5 - 0{,}72 = 1{,}57$$

$$\frac{B}{h_{расч}} = \frac{4}{2{,}3} = 1{,}7$$

По таблице 4.2 определим $\eta_0 = 10$.

r_1 – коэффициент, учитывающий повышение КЕО при боковом освещении благодаря свету, отраженному от поверхностей помещения и подстилающего слоя.

Средний коэффициент отражения в помещении $\rho_{СР} = 0,5$, принимаем одностороннее боковое освещение.

$$\frac{l_{рт}}{B} = \frac{0,5}{4} = 0,125$$

Тогда $r_1 = 1,05$.

$k_{зд}$ – коэффициент, учитывающий затенение окон противостоящими зданиями.

Поскольку затеняющих зданий поблизости нет, то $k_{зд} = 1$.

Вычислим общую площадь окон:

$$S_0 = \frac{24 \cdot 1,35 \cdot 10 \cdot 1 \cdot 0.63}{100 \cdot 0,24 \cdot 1,05} = 8,16 \text{ м}^2$$

Площадь световых проемов равна $S_{сп} = 8,16 \text{ м}^2$.

Таким образом площадь световых проемов (1,5*1,2*2=3,6<8,16) не обеспечивает необходимых условий труда на рабочих местах. С целью создания наиболее благоприятных условий труда в помещении с параметрами 6x4x3 для обеспечения нормированного значения КЕО, e_N=0,84 при III характеристике зрительных работ, совместно с естественным освещением используется искусственное освещение.

5.3 Расчет искусственного освещения

5.3.1 Точеный метод

Разряд зрительной работы – V.

Нормируемая освещенность по таблице 4.2 – 400 лк.

Для искусственного освещения применяем люминесцентные лампы ЛД 40. Параметры люминесцентных ламп приведены в таблице 5.2.

Высота подвеса светильников над освещаемой поверхностью Н=2,145 м, коэффициент запаса равен $k_Z = 1,2$.

Схема освещенности представлена на рисунке 5.2.

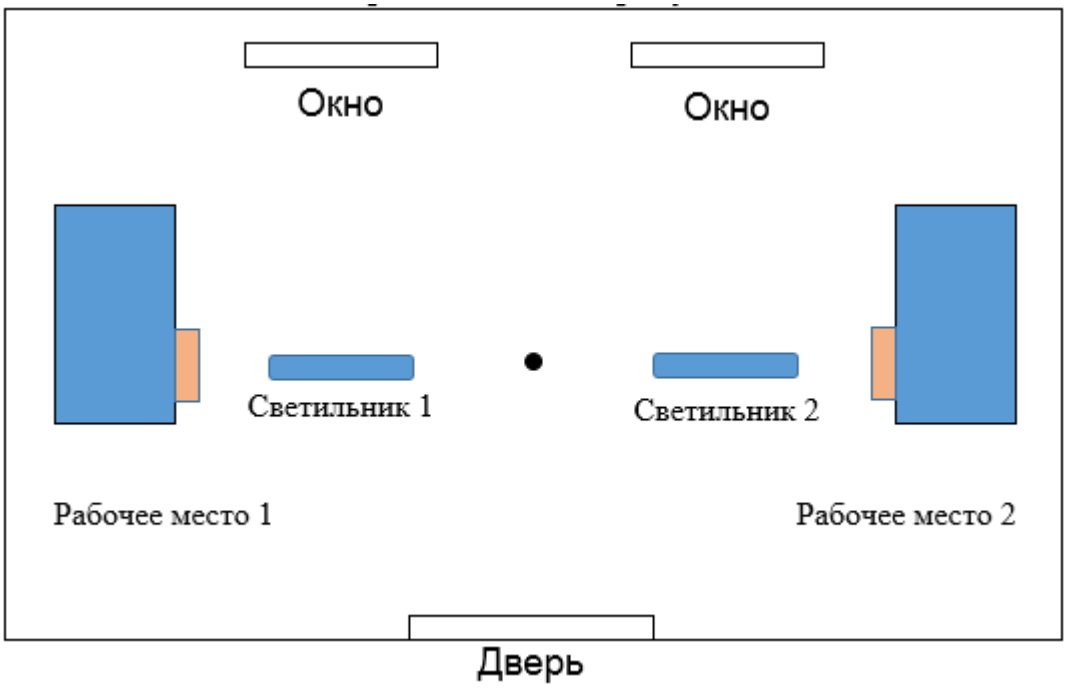

Рисунок 5.2 – Схема расчета освещенности

Намечаем контрольную точку О, находим расстояние от этой точки до каждого светильника и затем по графику пространственных кривых (изолюкс) находим освещенность от каждого светильника.

Т а б л и ц а 5.2 – Освещенность от каждого светильника

№ светильника	1	2
d (м)	1,2	1,2
E (лк)	25	25

Суммарная освещенность от всех светильников в контрольной точке О по формуле 5.5

$$\sum E_o = E_1 + E_2 \qquad (5.5)$$

$$\sum E_o = 25 + 25 = 50$$

Световой поток вычисляется по формуле:

$$\phi = \frac{1000 \cdot E \cdot k_z}{\mu \cdot \sum E_o} \qquad (5.6)$$

где: μ – коэффициент, учитывающий действие “удаленных” светильников (принимается 1,1-1,2);

E – нормируемая освещенность помещения.

$$\phi = \frac{1000 \cdot 400 \cdot 1{,}2}{1{,}2 \cdot 67} = 5970{,}15 \text{ лм}$$

5.3.2 Метод коэффициента использования

Разряд зрительной работы – V. Нормируемая освещённость – 400 лк.

В качестве светильника возьмем ЛСП64-2. Длина светильника 1540 мм, ширина 276 мм.

Т а б л и ц а 5.3 – Технические характеристики газоразрядных ламп ЛБ

Номинальная мощность, Вт	Номинальный световой поток ламп типа ЛБ, лм	Размеры ламп, мм	
65	3570	Диаметр	Длина по штырькам
		40	1514,2

Коэффициенты отражения от потолка стен и пола соответственно равны:

$\rho_{пот} = 70\%;$

$\rho_{ст} = 50\%;$

$\rho_{пол} = 30\%;$

Вычислим высоту подвеса светильника над рабочей поверхностью по формуле 5.7

$$H = h - h_P - h_c \qquad (5.7)$$

где: h_C – расстояние от светильника до перекрытия, h_C=0,05 м;
h_p – высота рабочей поверхности над полом, h_p=0,7 м;
h – высота помещения, h=3 м;
$H = 3 - 0{,}7 - 0{,}05 = 2{,}275 м.$

Лучшее расстояние от окна до светильника определяется по формуле 5.8.

$$L = \lambda \cdot H \qquad (5.8)$$

где: $\lambda = 1{,}2 \div 1{,}4;$

$$L = 1{,}25 \cdot 2{,}275 = 2{,}84 \text{ м}$$

Расстояние от стены до ближайшего светильника, когда работа у стены не проводится, определяем по формуле 5.9

$$l_1 = (0{,}4 \div 0{,}5) \cdot L \qquad (5.9)$$

$$l_1 = 0{,}4 \cdot 2{,}73 = 1{,}138\,м$$

Определяем индекс помещения по формуле 5.10

$$i = \frac{l \cdot s}{H \cdot (l + s)} \qquad (5.10)$$

$$i = \frac{4 \cdot 2}{2{,}275 \cdot (4 + 2)} = 1{,}319$$

Коэффициент использования в данном случае равен $\eta = 65\%$, коэффициент запаса равен $k_Z = 1{,}2$

Определим количество люминесцентных ламп по формуле 5.11

$$N = \frac{E \cdot k_Z \cdot S_{OC} \cdot Z}{n \cdot \Phi_Л \cdot \eta} \qquad (5.11)$$

где: S_{OC} – площадь помещения;
k_Z – коэффициент запаса;
E – заданная минимальная освещённость, E=400 лк.;
Z – коэффициент неравномерности освещения, Z=1,1;
n – количество ламп в светильнике;
$\Phi_л$ – световой поток выбранной лампы, Фл=3570 лм;
η – коэффициент использования, η=65%.

$$N = \frac{400 \cdot 1{,}2 \cdot 32 \cdot 1{,}1}{4 \cdot 3570 \cdot 0{,}65} \approx 2$$

Всего для создания нормируемой освещенности 400 лк необходимо 4 люминесцентных ламп серии ЛД, мощность каждой лампы должна быть не меньше 65 Вт, что соответствует действительности, а значит имеющегося в наличии освещения достаточно для соответствия санитарным нормам.

Заключение

Во время выполнения данного дипломного проекта были изучены основные принципы разработки мобильных приложений, а также требований, предъявляемых к ним. Изучены принципы построения и работы ПП для операционной системы Android. Помимо этого, было также установлено, что исследуемая тема является в наивысшей степени актуальной в реалиях современного рынка мобильных устройств.

Также было разработано приложение для соответствующей платформы, которое, при желании, может быть перенесено на другие мобильные операционные системы, использующие архитектуру процессора ARM. Данное приложение отвечает всем стандартам и требованиям платформодержателя. Предоставляет удобный способ работы с устройством и позволяет гибко изменять его интерфейс под нужны пользователя. Имеет в своем наличии уникальные функции, которые не доступны в стандартных, системных приложениях, а также в аналогичных продуктах сторонних разработчиков.

Была рассчитана общая стоимость продукта, а основе суммирования различных видов затрат, которые возможно при производстве. Данный расчет соответствует любому виду реализации продукта, как на разработка для заказчика, так и самостоятельная разработка, для последующей публикации в интернет магазине приложений. В процессе исследования аналогичных продуктов на рынке, установлено, что стоимость разработки полностью соответствует качеству разрабатываемого ПП и позволяет получить большой объем прибыли.

В разделе «Безопасность жизнедеятельности» были рассчитаны необходимые условия для организации соответствующего всем установленным нормам, рабочего места для сотрудников, занятых в реализации проекта. Рассчитано необходимое естественное и искусственное освещение, требующееся для комфортной работы без нанесения вреда здоровью сотрудников.

Список используемой литературы

2 Сайт https://www.youtube.com/user/freshgamer10/videos
3 Сайт http://startandroid.ru/ru/
4 Сайт http://developer.android.com/intl/ru/index.html
5 Бадагуев Б.Т. Документация по охране труда в организации. М.,Альфа-пресс, 2010.
6 Дюсебаев М.К., Бегимбетова А.С. Методические указания к выпускной работе (для студентов всех форм обучения специальностей 050719 – Радиотехника электроника и телекоммуникации, 050704 – Вычислительная техника и программное обеспечение) Алматы: АИЭС, 2008.
7 СНиП РК 2.04-05-2002 «Естественное и искусственное освещение. Общие требования».
8 Коробко, В.И. Охрана труда: Учебное пособие для студентов вузов / В.И. Коробко. – М.: ЮНИТИ-ДАНА, 2013. – 239 с.
9 Абдимуратов Ж.С, Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. Бакалавриат. – Алматы, АИЭС, 2009.