МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой _____
_____
(Ф.И.О., ученая степень, звание)
_____ «_____»_____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: «Реализация мобильного приложения «Справочник по С++» на платформе Android»

Специальность 5В070400 – Вычислительная техника и программное обеспечение

Выполнил (а) Шамбулова А.Ж.
(Фамилия и инициалы) группа

Научный руководитель к.ф.-м.-н, доцент Шайхин Б.М.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекишева А.У., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «03» 05 2016 г.
(подпись)

по безопасности жизнедеятельности:
Прокофьева И.Г., д.х.н., профессор
(Фамилия и инициалы, ученая степень, звание)
_____ «28» 04 2016 г.
(подпись)

по применению вычислительной техники:
Шайхин Б.М. к.ф.-м.-н., доц.
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20___ г.
(подпись)

Нормоконтролер: Шайхин Б.М. к.ф.-м.н., доц.
(Фамилия и инициалы, ученая степень, звание)
_____ «28» 05 2016 г.
(подпись)

Рецензент: _____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20___ г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Аэрокосмических и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Шамбулова Айя Науановна
(фамилия, имя, отчество)

Тема проекта Реализация мобильного приложения "Справочник по С++" на платформе Android

утверждена приказом ректора № 21 от «10» марта 2016 г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Изучение технологий разработки для мобильной платформы Android. Выбор технологий разработки и проектирование приложения "Справочник по С++" для обучения языку программирования С++.

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

Перечень графического материала (с точным указанием обязательных чертежей)

Рисунки архитектуры ОС Android
Рисунки интерфейса инструментов разработки
Рисунки, описывающие требования к приложению
Рисунки интерфейса приложения
Таблицы экономического обоснования
Рисунки, описывающие безопасность жизнедеятельности

Рекомендуемая основная литература

Рето М., Android 2. Программирование приложений для планшетных компьютеров и смартфонов – М. Эксмо, 2011.
Сайт: http://startdroid.ru/ru
Сайт: http://developer.android.com/design/material/
Аланбас У. А. Экономика предприятия – А.: „Бастау", 2012.

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Приходько И.Г.	01.04 – 28.04.16	
экономич. часть	Бекмуреева А.О.	15.03. – 03.05.16	
Раздел 1	Шайхин Б.М.	29.02 – 29.02.16	
Раздел 2	Шайхин Б.М.	01.03 – 30.03.16	
Раздел 3	Шайхин Б.М.	01.04 – 30.04.16	
Нормоконтроль	Шайхин Б.М.	28.05.16	

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Принципы разработки приложения	29.01-29.02	
2.	Инструменты разработки	01.03-30.03	
3.	Практическая часть	03.04-30.04.	
4.	Технико-экономическое об-ние	10.03-06.05	
5.	Безопасность жизнедеятельности	14.03-28.04.	

Дата выдачи задания «____»_____20___ г.

Заведующий кафедрой _____ _____
(подпись) (Фамилия и инициалы)

Руководитель _____ Шайхин Б.М.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Шамбулова А.Н.
(подпись) (Фамилия и инициалы)

Аннотация

В дипломной работе рассмотрены инструменты и технологии для разработки приложения на мобильную платформу Android. Разработанное приложение является справочником для студентов, изучающих язык программирования C++. Приложение соответствует всем предъявленным требованиям для публикации в магазине мобильных приложений Google Play.

Произведены расчеты экономической части с обоснованиями и даны рекомендации по улучшению условий труда в соответствующем разделе дипломной работы.

Аңдатпа

Дипломдық жұмыста Android мобильдік платформасына арналған қосымшаларды құрастыру технологиялары мен құралдары қарастырылған. Жасалған бағдарлама C ++ дамыған қолданбалы бағдарламалау тілін оқитын студенттер үшін нұсқау болып табылады. Мобильді қосымша Google Play дүкенінде жариялау үшін барлық талаптарына сай келеді.

Дипломдық жұмыстың бөлек бөлімдерінде экономикалық есептеулер жургізілген және басқа бөлімде еңбек жағдайын жақсарту қарастырылған.

Annotation

In the diploma project the technology and tools for application development on the mobile platform Android. The developed application is a guide for students studying programming language C ++. This application can be published in the shop of mobile appendixes of Google Play and fully conforms to all requirements produced for a publication.

The relevant section of the thesis of the economic calculations is made with the reasons and recommendations to improve working conditions.

СОДЕРЖАНИЕ

Введение

В настоящее время с развитием технологий актуальность темы исследования лежит в области мобильных устройств, которое достигает невероятных масштабов, его рынок насыщается и увеличивается с каждым кварталом следующего года. Жесткая конкуренция среди производителей фирм и постоянный рост характеристик заставило рынок снижать ценовую категорию на устройства. Это способствовало тому, что стали появляется недорогие аппараты, предлагающие хорошие возможности, которые не уступают среднему и высокому ценовому сегментам. Благодаря этому с невероятной скоростью увеличивается аудитория пользователей смартфонов, они стали неотъемлемой частью повседневной, деловой жизней, стали глубже проникать и в другие ее секторы.

По последним исследованиям на второй квартал 2015 года, сделанные американской компанией IDC (International Data Corporation), были установлены три лучшие мобильные платформы: Android, iOS и Windows Phone. Основными конкурентами в выпуске операционных систем (ОС) для смартфонов являются такие технические гиганты как Google и Apple, с своими ОС как Android и iOS, соответственно. Их доли рынка между ними распределились таким образом, что показатель как 13,9% отражает долю рынка айфонов на рынке смартфонов, так как iOS устанавливается только на яблочных устройствах (смартфонах от компании Apple), в то время как Android используются различными производителями, его показатели намного выше своих конкурентов - 82,8% рынка. Третье место занимает мобильная операционная система, разработанная американской компанией Microsoft - Windows Phone. Она установлена на 2,6% долю смартфонов, проданных во втором квартале 2015 года.

Каждая ОС содержит в себе специальные онлайн-магазины, в которых можно загрузить и скачать любое мобильное приложение, как на платной, так и на бесплатной основе: Play Market для Android, (Google), App Store для iOS (Apple) и Market Place для Windows Phone (Microsoft). У разработчиков программного обеспечения (ПО) появилась уникальная возможность брать плату за свои приложения с помощью магазинов приложений, что послужило сделать создание приложений под эти платформы ещё более привлекательной. Это означает, что потребность в разработчиках мобильных приложений с каждым днем будет только расти.

За несколько лет Android стал одним из успешных проектов для мобильных телефонов в компании Google Inc. Платформа все больше захватывает рынок мобильных телефонов и планшетов, постепенно вытесняя с него признанных всеми лидеров. Главная идея IT-гиганта состоит в том, что она предлагает открытый доступ исходного кода для своей ОС, предоставляет набор инструментов для разработки и подробный документированный комплект SDK, что делает привлекательным для многих успешных компаний

про производству мобильных компьютеров. Система Android внедряется теперь не только на смартфоны, еще она адаптирована для планшетов и умных часов. Для разработки и откладки программ выбирают интегрированные среды разработки Eclipse и Android Studio, которые набирают больше последователей системы Android, а именно это программисты, поскольку она является бесплатной и легко интегрируется с любой популярной ОС.

В то время как за системой Visual Studio стоит компания Microsoft, для которой первой очередью была прибыль. В принципе языки между собой похожи, оба являются кроссплатформенными, пожалуй, со своей стороны предпочтение было отдано на языку Java. Решающую роль сыграло наличие смартфона на платформе Android. Так как основная масса людей приобретают телефоны на платформе Android, нежели на Windows Phone, хотя, при этом iOS является одним из основных конкурентов Android, но по актуальным данным, даже iPhone в сравнении с Android уже не столь популярен, как телефон на ОС Android. Производители стали чаще внедрять систему в различного рода оборудования: телефоны, планшеты, телевизоры, часы, интерактивные доски и т. д. Факт остается фактом, что в будущем главным лидером будет компания Google.

Целью моей дипломной роботы является изучения технологий реализации мобильного приложения «Справочник по С++» для мобильных устройств на платформе Android. Приложение является альтернативой карманного справочника по языку программирования С++, которое позволит пользователям удобно и свободно пользоваться материалом, представленным по выбору на двух языках: русский и английский, а также применять свои знания на практике, используя встроенные примеры код программ.

1 Принципы разработки приложения

1.1 Операционная система Android

Android – это одна из некоторых операционных систем нового поколения, которые созданы для работы с аппаратным обеспечением современных мобильных и планшетных устройств. Многие ведущие платформы как Windows Mobile, iOS, FireFoxOS, конечно же предлагают достаточно более простые, мощные и удобные в использовании среды для разработки мобильных приложений. Однако, Android отличается тем, что это запатентованные ОС, в которых в определенном случае приоритет будет отдан встроенному программному обеспечению (ПО), а не приложениям сторонних разработчиков-программистов. Вместе с тем, эти операционные системы стараются ограничивать возможности взаимодействия приложений с данными смартфона, а также контролируют процесс и ограничивают распространения сторонних программ, созданных для данных систем.

Безграничные возможности для мобильных приложений, вот что дает система Android, предлагая открытую среду разработки, построенную на открытом исходном ядре Linux. С помощью специальных серий API-библиотек, приложения имеют открытый доступ к аппаратным средствам устройства. В дополнение здесь включена контролируемая и полная поддержка взаимодействия всех программ.

Система Android присуждает всем приложениям одинаковый статус.

Во всех программ стоит одинаковое время исполнения, как и сторонних приложениях, написанные на API, так и встроенное ПО. Любой пользователь может удалять или заменять корневые приложения на альтернативные сторонние программы, к примеру, калькулятор или камеру.

Для более простого понимания системы, можно разбить и скомбинировать на три компоненты:

1. свободная операционная система с открытым исходным кодам;
2. среда разработки для создания мобильных приложений с открытыми исходными кодами;
3. технические аппараты, на которых встроена ОС Android вместе с разработанными для неё приложениями.

ОС включает несколько взаимозависимых элементов, которые необходимы:

– референс-дизайн аппаратного обеспечения с перечнем требований к мобильным устройствам, чтобы гарантировать совместимость с ПО;

– ядро операционной системы Linux, которое предоставляет низко-уровневый интерфейс для управления аппаратным обеспечением, памятью и процессами, оптимизированными для работы на мобильных устройствах;

– библиотеки с открытыми исходными кодами, предназначенными разработки приложений SQLite, WebKit, OpenGL и медиа-менеджер;

– среду исполнения для приложений, включающую виртуальную

машину Dalvik и библиотеки ядра, которые отвечают за функционал Android;

Среда исполнения отличается небольшим размером, что позволяет

эффективно использовать ее на мобильных устройствах;

– набор программных компонентов, обеспечивающих доступ к системным службам на уровне приложений; среди них менеджер окон и менеджер местоположения, контент-провайдеры, возможности работы с телефонией и сенсорным дисплеем;

– набор компонентов пользовательского интерфейса для размещения и запуска приложений;

– предустановленные приложения, поставляемые в общем программном наборе;

– комплект программ для разработки приложений, включающий инструменты, плагины и справочную документацию.

Особо стоит подчеркнуть, что открытая архитектура Android позволяет исправлять любые ошибки в пользовательском интерфейсе или дизайне встроенных приложений путем написания расширений или замещений ошибок. Android предоставляет возможность создавать собственные интерфейсы для мобильных телефонов, а также приложения с функционалом и дизайном, максимально отвечающими вашим потребностям. [1]

Разработчикам Google предлагает для свободного скачивания инструментарий для разработки (Android SDK), который предназначен для x86- машин под операционными системами Windows (начиная 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, калькулятор, 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.

– 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.

– 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.

– 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.

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

a) Content Providers – обмен данными между приложениями.
b) Resource Manager – доступ к таким ресурсам, как файлы XML, PNG, JPEG.
c) Notification Manager – доступ к строке состояния.
d) Activity Manager – управление активными приложениями.

1.2 Фреймворк разработчика

Основным языком программирования приложений на Android является Java. Однако, он выполняется на виртуальной машине Java (Dalvik Virtual Machine), а не на классическом Java VM.

Внутри собственного экземпляра машины Dalvik в отдельном процессе функционирует каждое приложение для Android. Всю ответственность за управление процессами и памятью берет на себя операционная система, которая либо останавливает, либо может убивать процессы, чтобы освободить свои ресурсы.

Dalvik и Android располагаются на одной вершине ядра Linux, которое занимается низкоуровневым взаимодействием с аппаратным обеспечением устройства, включая работу за управление памятью и драйверов. Виртуальная машина, при его исполнении, не компилирует приложение, это дает обеспечение удобства при использовании многозадачности приложения. В отличие от Dalvik, ART загружает в оперативную память абсолютно все инструкции, для этого необходимо большой объем оперативной памяти, меньше используется процессор, но при этом программы открываются быстрее. Виртуальная машина ART, пока еще является экспериментальным объектом, однако Google предоставляет возможность разработчику переключиться с Dalvik на ART через скрытые инструменты. Выбор виртуальной машины ART и Dalvik представлен на рисунке 1.2.

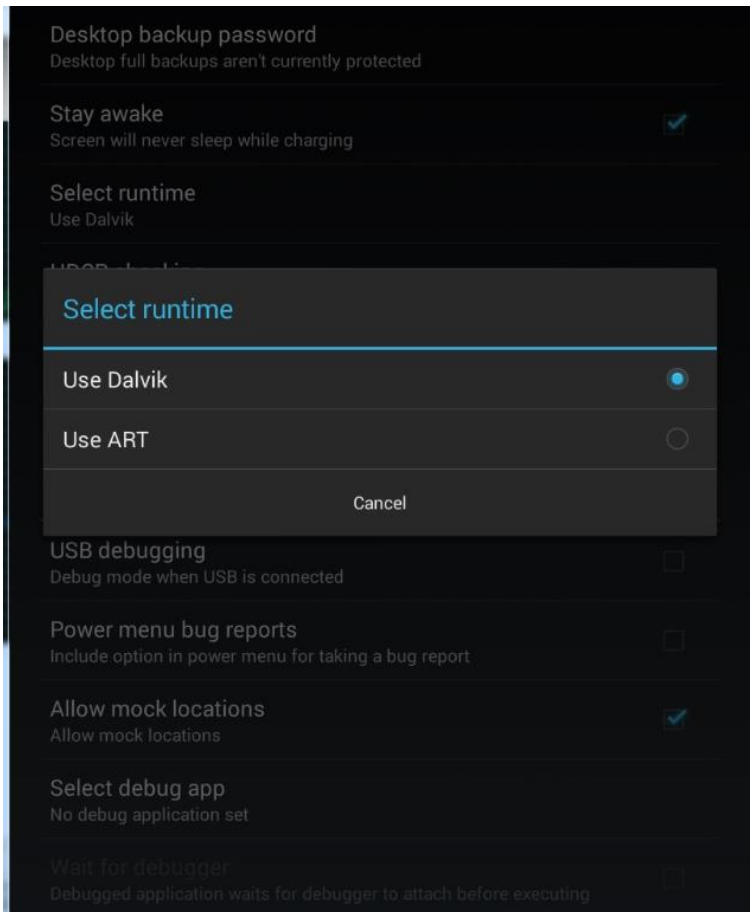


Рисунок 1.2 – Выбор виртульной машины

В добавок, набор встроенного API позволяет получить полный доступ ко всем службам, функционалу начинки и аппаратной части устройства.

1.3 Программный стек Android

Android - это программный стек для мобильных устройств, который включает в себя операционную систему, программное обеспечение промежуточного слоя (middleware), а также основные пользовательские приложения (почта, календарь, карты, будильник, контакты и другие). Программный стек Android состоит из множества элементов, где их подробное описание упрощенно приводится ниже на рисунке 1.3.

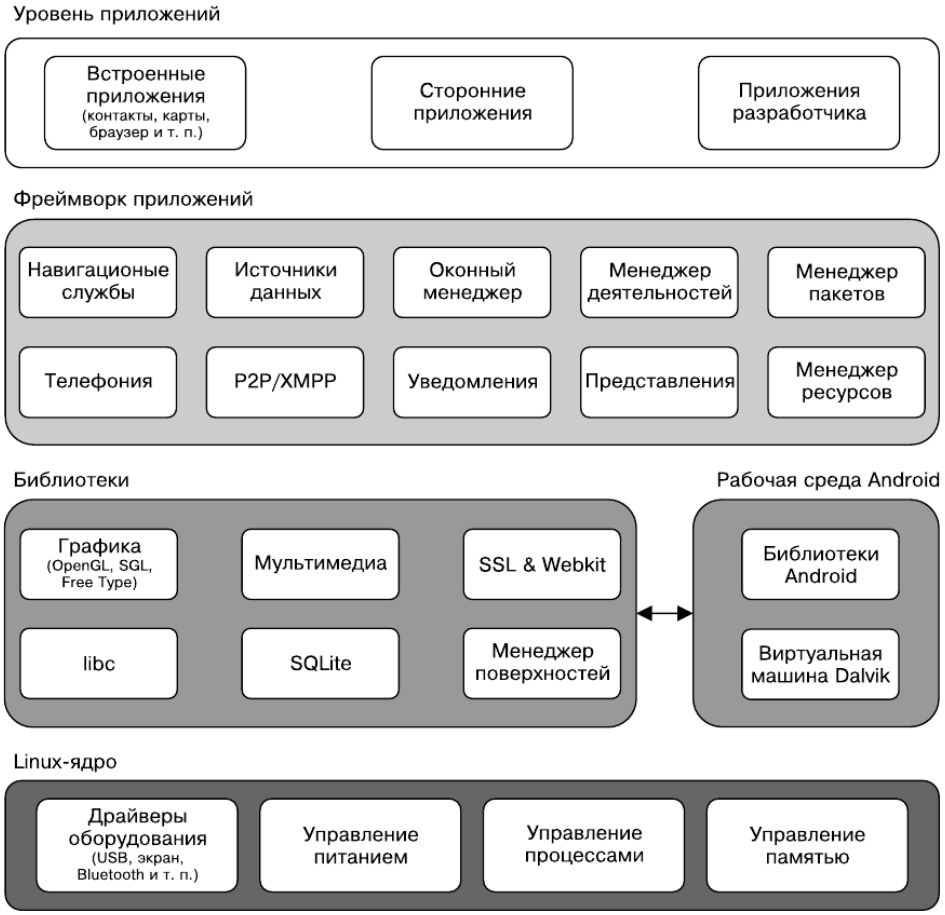


Рисунок 1.3 – Программный стек Android

Их можно представить, как комбинацию ядра Linux и набора библиотек C/C++, которые доступны в Фреймворке приложения. Последний обеспечивает управление и функционирование рабочей среды и приложений.

Ядро Linux. Работу системных служб (драйверы устройств, управление процессами и памятью, питанием, безопасность, сетевые службы) обеспечивает ядро Linux версии 2.6. Оно также отвечает за уровень абстракции между аппаратной начинкой и остальной частью программного стека.

Библиотеки. Android включает разнообразные системные библиотеки C/C++ (например, SSL и libc), которые работают поверх ядра. Среди них можно выделить:

a) библиотеку для работы с мультимедиа, которая обеспечивает проигрывание аудио- и видеофайлов;

b) менеджер интерфейса, отвечающий за управление отображением;

c) графические библиотеки, такие как SQL и OpenGL, для работы 2D- и 3D-графикой;

d) библиотеку SQLite, обеспечивающую работу встроенных баз данных;

e) SSL и WebKit для работы встроенного веб-браузера и обеспечения интернет-безопасности.

Рабочая среда Android. Особенным телефон на платформе Android делает не столько мобильная версия ОС Linux, сколько рабочая среда Android. Она включает в себя библиотеки ядра и виртуальную машину Dalvik и обеспечивает функционирование программ, а вместе с библиотеками формирует основу фреймворка приложений.

Библиотеки ядра. Хотя приложения для Android разрабатываются на языке Java, Dalvik – это не виртуальная Java-машина. Библиотеки ядра Android обеспечивают основную функциональность библиотек ядра Java, а также присущий Android уникальный функционал.

Виртуальная машина Dalvik. Dalvik – это виртуальная машина на основе регистров, которая оптимизирована таким образом, чтобы на устройстве можно было запускать несколько приложений одновременно. В ее основе ядро Linux, которое обеспечивает работу потоков и низкоуровневое управление памятью.

Фреймворк приложений. Фреймворк включает набор классов, которые используются для разработки приложений. Он также предоставляет обобщенные абстрактные классы для доступа к оборудованию и обеспечивает управление пользовательским интерфейсом и ресурсами приложения.

Уровень приложений. Все программы, как встроенные, так и сторонние, разрабатываются на уровне приложений с использованием одних и тех же библиотек API. Уровень приложений функционирует внутри рабочей среды Android, используя классы и службы, открытые для доступа на этом уровне.

1.4 Виртуальная машина Dalvik

Главный ключевой компонент Android является виртуальная машина (ВМ) Dalvik. Вместо классической виртуальной Java-машины, такой как Java ME (Java Mobile Edition), Android использует собственную ВМ, разработанную для обеспечения эффективной работы нескольких приложений на одном устройстве.

Если для приложения важны присущие скорость и эффективность работы, Android предоставляет доступ к нативной среде разработки (NDK). Она позволяет разрабатывать библиотеки C++ с использованием библиотек libc и libm, а также обеспечивает нативный доступ к OpenGL.

В основе ВМ Dalvik лежит ядро Linux, которое обеспечивает отличную работу низкоуровневых функций, таких как потоки, безопасность, управление процессами и памятью. Вы можете также писать приложения C/C++, которые будут работать непосредственно на базовом уровне ОС Linux. Хотя такая возможность и существует, необходимости в этом нет никакой.

Доступ к устройствам и системным службам Android, которая считается промежуточным слоем, осуществляется через виртуальную машину Dalvik. Благодаря использованию ВМ для выполнения кода программы управляющей системы, разработчики получают в свое распоряжение уровень

абстракции, позволяющий им не беспокоиться об особенностях конструкции того или иного устройства.

ВМ Dalvik запускает исполняемые файлы, формат которых оптимизирован под минимальное использование памяти. Создается исполняемый файл с расширением (.dex) путем трансформирования компилированных классов, написанных на языке программирования Java, используя для всего этого инструменты, входящие в состав среды разработки.

1.5 Компоненты приложения в Android

Приложения в Android состоят из слабосвязанных компонентов, собирающиеся воедино с помощью программного манифеста. Манифест – это файл, который описывает все компоненты и способы их взаимодействия в приложении, а также мета-данные, включая требования к платформе и аппаратной конфигурации.

Приложение состоит из элементов или перечисленные ниже компонентов, од которых подробнее будет описано ниже.

Активности. Уровень представления. С точки зрения разработки под настольные платформы Активность – эквивалент Формы (Form). Каждый экран приложения наследником класса Activity. Активности используют представления для формирования графического пользовательского интерфейса, отображающего информацию и взаимодействующего с пользователем.

Виджеты. Визуальные программные компоненты, которые можно добавлять на домашний экран. Этот особый вид Широковещательных приемников позволяет создавать интерактивные, динамические компоненты, которые пользователи могут встраивать в свои домашние экраны.

Источники данных. Хранилища информации. Источники данных задействуются при обмене информацией между разными программами. Данные компоненты нужны для управления базами данных в пределах одного приложения и предоставления к ним доступа извне. Это обозначает, что вы можете настраивать собственные объекты ContentProvider, открывая к ним доступ из других приложений, а также использовать чужие источники, чтобы работать с данными, которые открыли для вас внешние программы. Устройства под управлением ОС Android содержат несколько стандартных Источников, которые предоставляют доступ к полезным базам данных, включая хранилища мультимедийных файлов и контактной информации.

Намерения. Система передачи сообщений между приложениями. Используя Намерения, вы можете транслировать сообщения на системном уровне или для конкретных Активностей или Сервисов. Тем самым диктуется необходимость выполнения заданных действий. После этого Android сам определит компоненты, которые должны обработать поступивший запрос.

Сервисы. Невидимые двигатели вашего приложения. Используются для регулярных операций, которые должны продолжаться даже тогда, когда

Активности вашего приложения не на переднем плане. Сервисные компоненты работают в фоновом режиме, запуская уведомления, обновляя Источники данных и видимые Активности.

Уведомления. Система пользовательских уведомлений. Механизм уведомлений лучше всего подходит для Сервисов и Широковещательных приемников, когда необходимо привлечь внимание пользователя. Позволяет сигнализировать о чем-либо, не обращая на себя внимание или не прерывая работу текущей Активности. Например, принимая текстовое сообщение или входящий звонок, устройство оповещает вас, мигая светодиодами, воспроизводя звуки, отображая значки или показывая сообщения.

Широковещательные приемники. Компоненты, принимающие транслируемые Намерения. Широковещательные приемники автоматически запустят программу, чтобы она могла ответить на принятое Намерение. Благодаря этому данный механизм идеально подходит для создания приложений, использующих событийную модель. Если вы создадите и зарегистрируете объект BroadcastReceiver, ваше приложение сможет отслеживать трансляцию Намерений, которые соответствуют заданным критериям.

1.6 Что такое AndroidManifest.xml

AndroidManifest.xml – это 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 – это 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.

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.

<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:

<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:

1.7 Ресурсы

Ресурсы – это внешние файлы, которые используются вашим кодом, за компилированы в проект приложения и встраиваются в него во время работы

программы. Платформа Android поддерживает разные типы файлов ресурсов, в том числе такие как: XML, PNG и JPEG. Файлы вида XML очень сильно отличаются, так их форматы в зависимости от того, что они описывают. Файлы XML описаны в исходном коде, и ресурсы откомпилированы в двоичный код для эффективной и быстрой загрузки. Строки максимально сжаты в такую форму, которая экономит 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 того, чтобы использовать ресурс в коде, необходимо знать только полный ID ресурса и в какой тип объекта ваш ресурс был откомпилирован и отлажен.

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

Ресурсы, откомпилированные приложением, можно оставить без названия пакета (просто как R.resource_type.resource_name). Android содержит в себе многие стандартные ресурсы, такие как стили экрана и фоны кнопки. Обращаться к ним в коде, конечно пользователь может через android.R.resource_type.resource_name, для примера:

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

Например, если у нас есть ресурсы с цветами или числами, разработчик сможет написать layout файл, который установит цвет текста на значение, содержащееся в одном из указанных ресурсов:

Использование ресурсов в коде. Для использования ресурса в коде необходимо знать только полный ID ресурса и в какой тип объекта необходимый ресурс был откомпилирован.

resource_type – это подкласс R, который содержит определенный тип ресурса. resource_name – атрибут ресурсов, определенный в XML файлах, 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 приложением, могут быть использованы также без названия пакета (просто как R.resource_type.resource_name). Android содержит многие стандартные ресурсы, такие как фоны кнопки и стили экрана. Обращаться к ним в коде разработчик через android.R.resource_type.resource_name, для примера:

Ссылка на Ресурсы. Значение в атрибуте или ресурсе может также быть ссылкой на другой ресурс. Это довольно часто используется в layout файлах, для того 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:

Нужно обратить внимание на такой префикс "@", он показывает, что это – ссылка на ресурс, текст после него – название ресурса в форме @[пакет:]тип/имя. В этом примере разработчик определяет пакет, потому что ссылаемся на ресурс в нашем собственном пакете. Чтобы сослаться на системный ресурс, нужно написать следующий код:

Альтернативные ресурсы и локализация – это главная и серьезная проблема, которая достаточно решена в ОС Android. Обычно чаще всего разработчики должны были бы проектировать UI, хорошо подходящий для каждого разрешения экрана одновременно, что почти невозможно.

Можно заметить, что даже, если при добавлении много разных языков, UI и всех других ресурсов, SDK сама определит тот набор необходимых ресурсов, который будет в последующем использоваться. Так примеру, Android догадается сам, где и какой язык можно добавить в приложение различные UI, поддержку или языки устройств с всевозможной конфигурацией компонентов.

Вы как разработчик, чтобы включить 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.).

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

Более того, теперь можно определить только несколько определенных опций конфигурации, для которых определен ресурс:

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

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

1.8 Пользовательские интерфейсы в Android

Пользовательские интерфейсы (UI) могут быть созданы двумя путями в Android, через java-код или XML-код. К тому же, приспосабливание программы от одной разрешающей способности экрана до другой намного более просто. Создание структуры графического интерфейса пользователя в XML очень предпочтительно, потому что по принципу Образцового управления средства просмотра, UI должен всегда отделяться от логики программы. Определение UI в XML очень схоже к созданию общего документа HTML, где вы имеете такой простой файл.

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

Иерархия Элементов Экрана. Основной android.app.activity функциональный модуль приложения Android – Activity – объект класса android. 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. Activity может сделать много вещей, но отдельно у него нет присутствия на экране.

Views. View – объект расширяет базовый класс android.view.view. Класс View служит базовым классом для всех графических фрагментов – ряд полностью осуществленных подклассов, которые рисуют интерактивные элементы экрана. 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. Графические фрагменты обрабатывают свое собственное измерение и рисунок, таким образом можно 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. Класс Viewgroup служит базовым классом для Layouts – ряда полностью осуществленных подклассов, обеспечивающего общие типы Layouts экрана. 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. 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. На рисунке 1.4 изображена древовидная система пользовательского интерфейса. 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.

Чтобы прикрепить дерево к экрану и просчитать его, этот 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. 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. Когда Ваш Activity становится активным и получает приоритет, система регистрирует Ваш Activity и просит корневой узел измерить и просчитать дерево. Дочерние узлы могут просить размер и местоположение в родителе, но у родительского объекта есть конечное решение, где и насколько большой каждый ребенок может быть. 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.

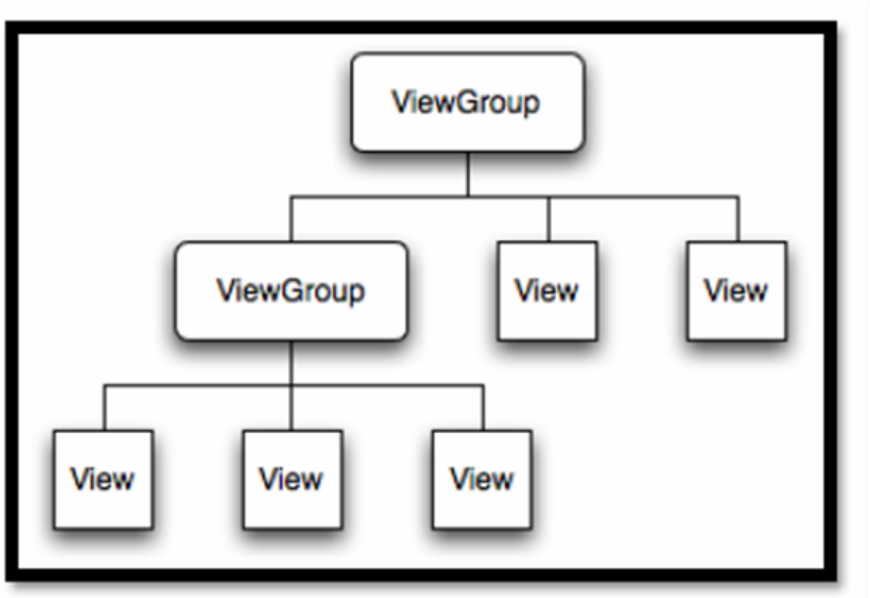


Рисунок 1.4 – UI ОС Android – древовидная структура

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.

В Android установка слушателей к View, также является почти тем же самым, как чем и в Swing. Такое сравнение приведено на рисунке 1.5.

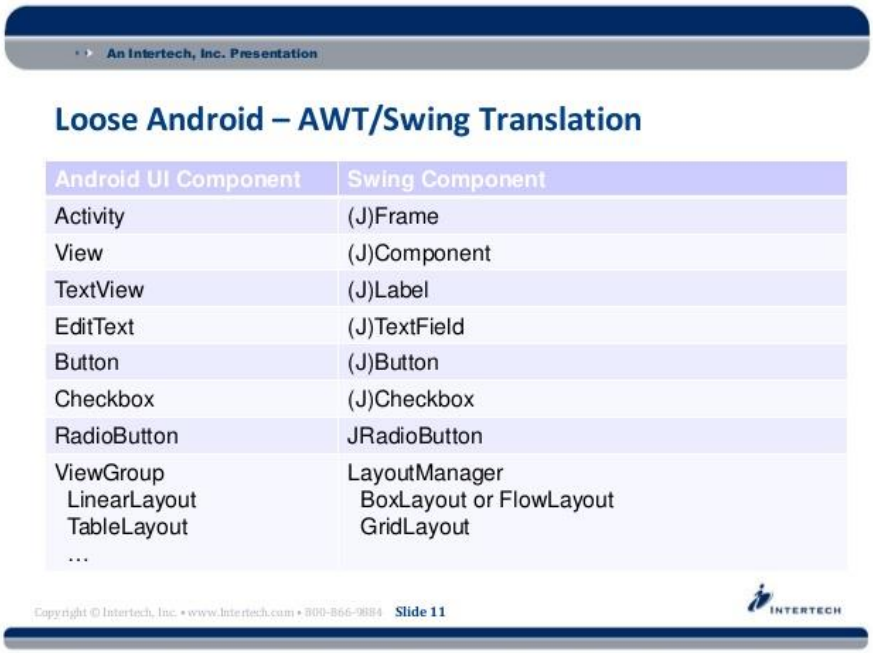


Рисунок 1.5 – Сравнение элементов Android с Swing

2 Инструменты разработки

2.1 Правила и рекомендации для разработки приложений

Каждая компания выдвигает ряд требований разработчикам приложений к созданию приложения для обеспечения работоспособности продукта, а также публикации в Google Play. Помимо всего, имеются некоторые рекомендации по концепции дизайна и организации интерфейса приложений, для того чтобы получить лаконичный, удобный и не выбивающегося из общего стиле системы пользовательского интерфейса (User Interface, UI). Эти принципы проектирования были разработаны группой Android по взаимодействию приложений с пользователями с целью соблюдения интересов пользователей. Для создателей приложений под Android они лежат в основе более подробных рекомендаций по проектированию для конкретных типов устройств. Со всеми ими можно ознакомится на профильных официальных сайтах компании Google.

2.1.1 Разработка дизайна приложений

В июне 2014 года на конференции Google I/O был представлен дизайн программного обеспечения (ПО) и приложений ОС Android, который называется Material Design (по рус. Материальный дизайн). По задумке дизайнеров компании, в приложении не должны содержаться острые углы, карточки должны переключаться между собой плавно и практически незаметно. Его идея заключается в приложениях, которые открываются и сворачиваются как карточки, используя эффекты теней. Материальный дизайн используется полноценно в таких операционных системах Android Lollipop, Android Marshmallow, Android N и в некоторых приложениях более старых версий.

Material Design держится на четырех основных принципах:

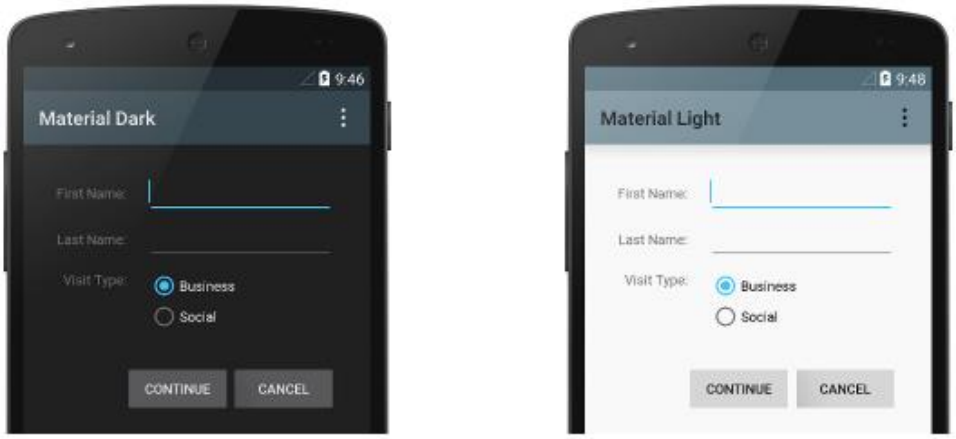
– Тактильные поверхности. В Material Design интерфейс складывается из осязаемых слоёв так называемой «цифровой бумаги». Эти слои расположены на разной высоте и отбрасывают тени друг на друга, что помогает пользователям лучше понимать анатомию интерфейса и принцип взаимодействия с ним.

– Полиграфический дизайн. Если считать слои кусками «цифровой бумаги», то в том, что касается «цифровых чернил» (всего того, что изображается на «цифровой бумаге»), используется подход из традиционного графического дизайна: например, журнального и плакатного.

– Осмысленная анимация. В реальном мире предметы не возникают из ниоткуда и не исчезают в никуда — такое бывает только в кино.

– Адаптивный дизайн. Речь идет о том, как применяется предыдущие три концепции на разных устройствах с разными разрешениями и размерами экранов. [2]

Тема Material Design предоставляет из себя уже новый стиль для ваших приложений. Системные виджеты, для которых можно настраивать цветовую палитру, и анимации, выполняемые по умолчанию в качестве реакции на касание и при переходах между действиями.

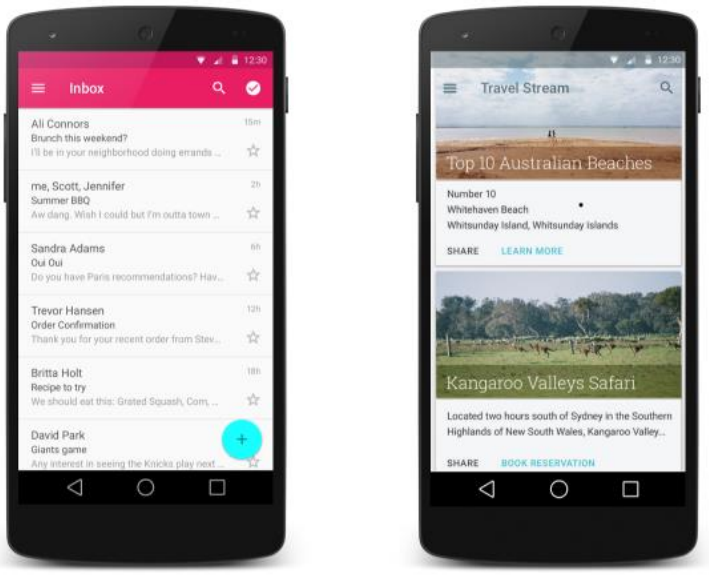


Тема Material Design в темных тонах

Тема Material Design в светлых тонах

Рисунок 2.1 – Темы Material Design, представленные в двух цветах

Android предоставляет два новых виджета для отображения подсказок и списков с использованием стилей и анимаций Material Design:



Виджет `RecyclerView` представляет собой более гибкую версию `ListView`, которая поддерживает различные типы макетов и способствует повышению производительности.

Виджет `CardView` позволяет отображать важные элементы информации внутри подсказок, имеющие согласованный внешний вид и поведение.

Рисунок 2.2 – Material Design в приложениях

2.1.2 Анимация

Новые API-интерфейсы анимации позволяют создавать нестандартную анимацию для реакции на касание в элементах пользовательского интерфейса, изменения состояния представления и переходов между действиями.

Эти API-интерфейсы позволяют:

– реагировать на касание в представлениях, используя анимацию для реакции на касание;

– скрывать и отображать представление с помощью анимации для кругового появления;

– переключаться между действиями с помощью настраиваемой анимации для переходов между действиями;

– создавать более естественное движение с помощью анимации для перемещения по кривой;

– анимировать изменение одного или нескольких свойств представления с помощью анимации для изменения состояния представления;

– отображать анимацию в графических элементах списков состояний в промежутке между изменением состояний представления.

Анимация для реакции на касание встроена в некоторые стандартные представления, например кнопки. Новые API-интерфейсы позволяют разработчику настраивать эти анимации и добавлять их в свои нестандартные представления.

2.1.3 Графические объекты

Следующие возможности по работе с графическими объектами облегчают реализацию приложений с элементами Material Design:

- векторные объекты можно масштабировать без ущерба для четкости, и они отлично подходят в качестве одноцветных значков приложения;
- тонированы графических объектов позволяет определять растровые изображения как альфа-маску и тонировать их нужным цветом во время выполнения;
- извлечение цвета позволяет автоматически извлекать главные цвета из растровых изображений.

2.1.4 Настройка строки состояния

В теме Material Design можно с легкостью настроить строку состояния, указав нужный цвет в соответствии с фирменным стилем и задав достаточную контрастность для отображения белых значков состояния. Чтобы установить настраиваемый цвет для строки состояния, воспользуйтесь атрибутом android:statusBarColor при наследовании темы Material Design. По умолчанию параметр android:statusBarColor наследует значение android:colorPrimaryDark.

Кроме того, можно самостоятельно разместить элемент за строкой состояния. Например, если требуется наложить прозрачную строку состояния поверх фотографии, применив еле уловимый темный градиент, чтобы были видны белые значки состояния. Для этого задайте для атрибута android:statusBarColor значение @android:color/transparent и настройте флаги окна требуемым образом. Также можно воспользоваться методом Window.setStatusBarColor() для применения анимации или эффекта постепенного исчезания.

В примечании можно указать, что строка состояния почти всегда должна иметь четкую границу, отделяющую ее от основной панели инструментов, за исключением случаев, когда за этими панелями от края и до края экрана отображается большое количество изображений или мультимедийный контент, а также в случае, когда вы используете градиент, чтобы обеспечить видимость значков.

При настройке панели навигации и строки состояния сделайте их прозрачными либо измените только строку состояния. Во всех остальных случаях панель навигации должна оставаться черной.

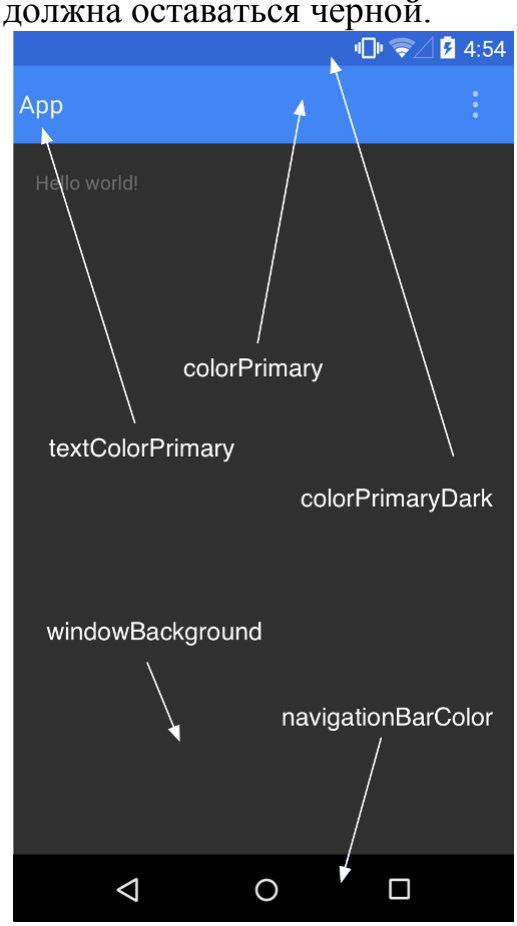


Рисунок 2.3 – Элементы в Material Design

При настройке панели навигации и строки состояния сделайте их прозрачными либо измените только строку состояния. Во всех остальных случаях панель навигации должна оставаться черной.

Элементы в определениях макета XML могут задавать атрибут android:theme, который ссылается на ресурс темы. Этот атрибут изменяет тему для элемента и любых дочерних элементов, что можно использовать для изменения цветовых палитр темы в определенной области интерфейса. [3,4]

2.2 Инструменты разработки графической оболочки

Illustrator CC – это программное обеспечение для создания графических элементов векторной графики, разработанное компанией Adobe для работы с проектами. Для того, чтобы приобрести его, необходимо пройти по адресу http://www.adobe.com/ru/products/illustrator.html, оформить годовую подписку на программный продукт, после этого загрузить и установить программу. Помимо прочего, можно приобрести лицензионный DVD-диск в специализированных магазинах. Начало процесса установки изображено на рисунке 2.4.

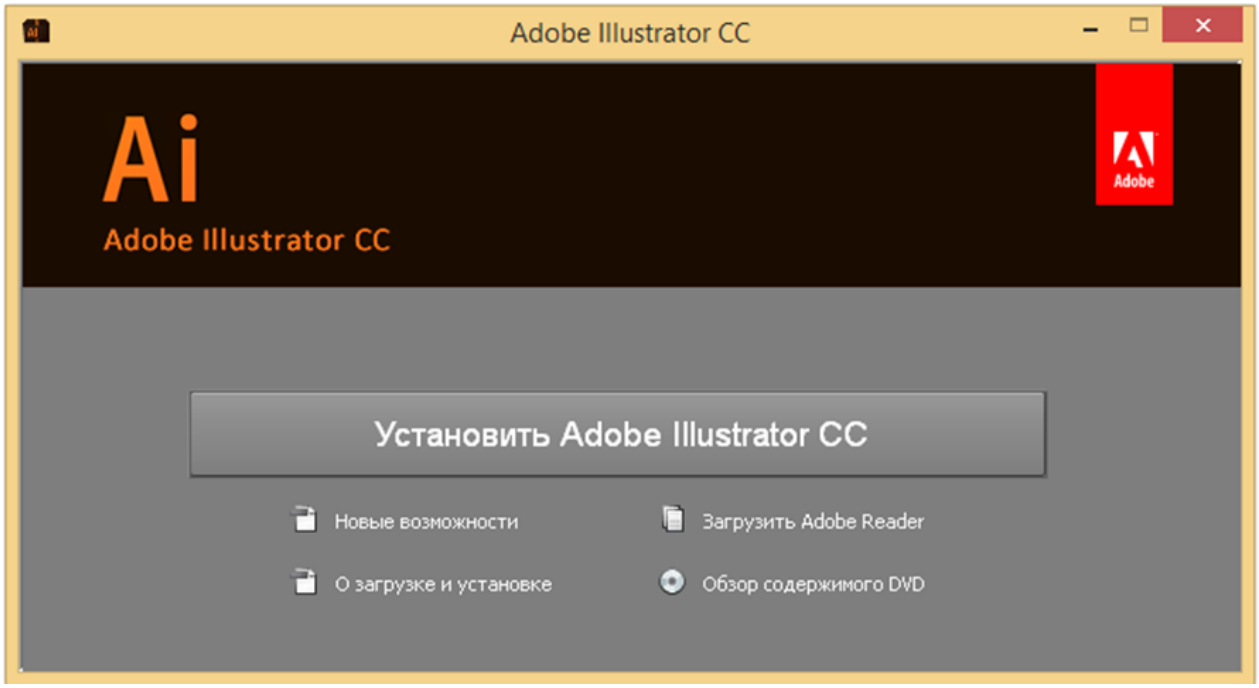


Рисунок 2.4 – Окно установки Adobe Illustrator CC

Существует, соответственно, две версии программы Adobe Illustrator CC, для 32 и 64-разрядных ОС. При использовании 64-разрядной системы, установка обеих версий не требуется, достаточно лишь той версии, которая есть для вашей ОС. Устанавливая одну версию из версий программы, экономится пространство на жестком диске. После выбора необходимой версии, нужно нажать кнопку «установить Adobe Illustrator CC». На рисунке 2.5 изображен выбор устанавливаемой версии, а рисунок 2.6 отображает с интерфейсом уже установленного приложения.

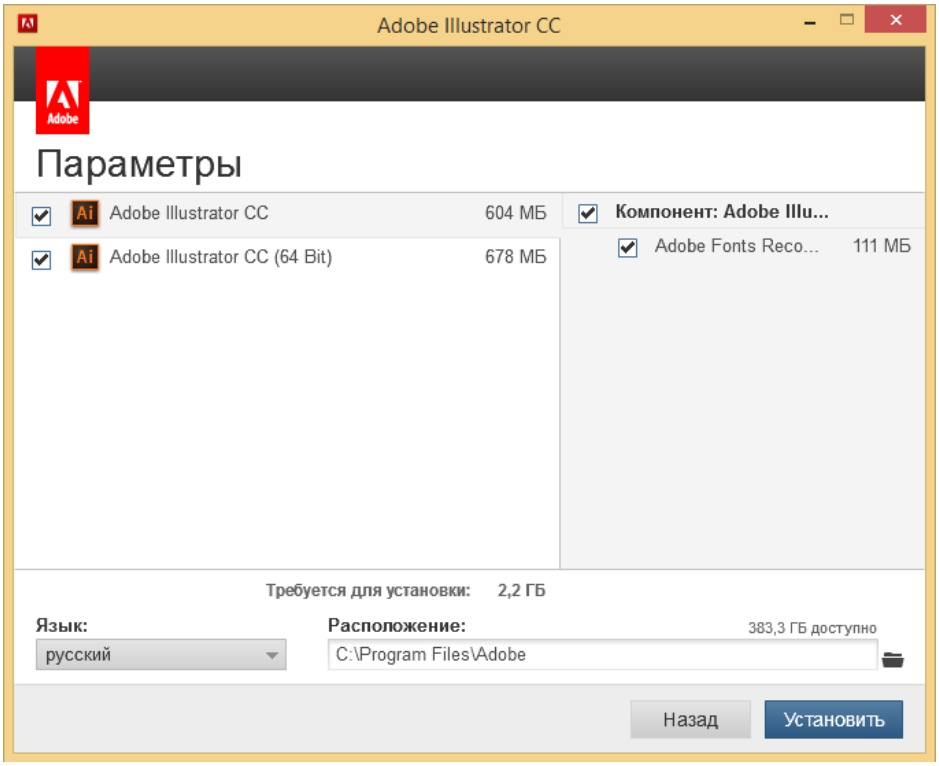


Рисунок 2.5 – Выбор версий приложения для установки

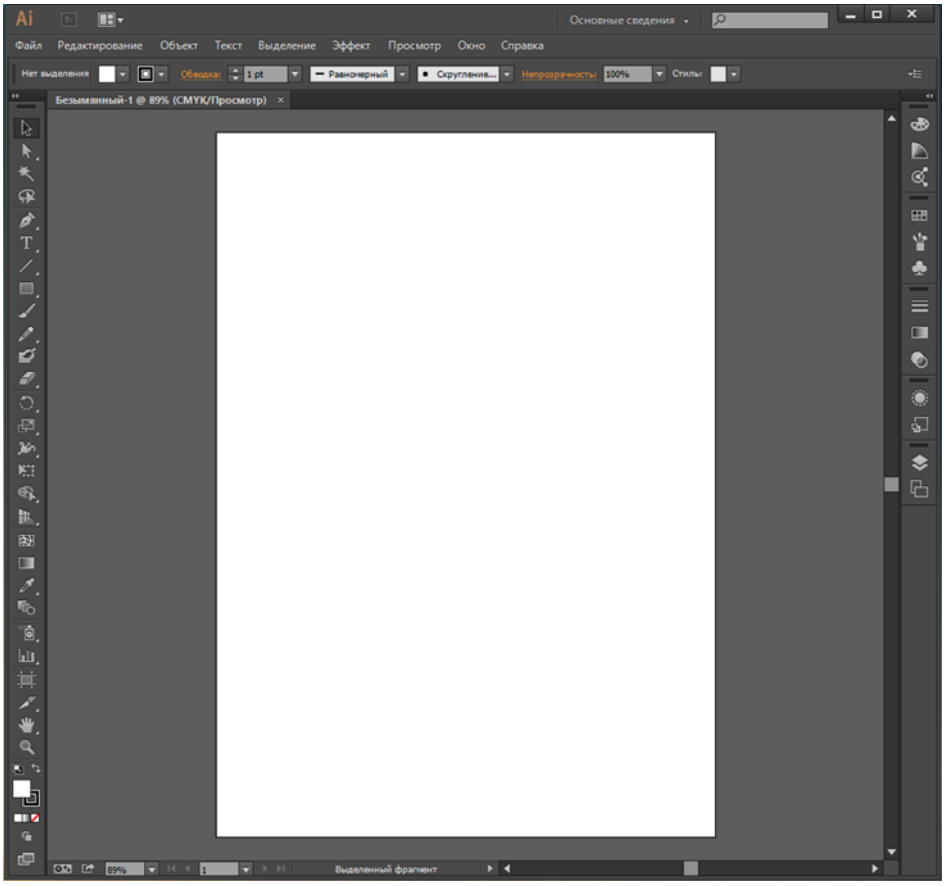


Рисунок 2.6 – Окно приложения Adobe Illustrator CC

Последняя версия редактора Adobe Illustrator CC, предназначенная для векторных изображений, которая предоставляет новые возможности как и для дизайнеров, так и для верстальщиков, а также в ней улучшены существующие инструменты.

Динамические углы

Сделайте ваши работы более изящными, благодаря точному и интуитивно понятному пользовательскому интерфейсу программы. Один или несколько углов можно одновременно переворачивать, скруглять или делать их скошенными, можно экспериментировать со округлением углов фигур и контуров с помощью маркеров, либо вводить значения параметров в новом диалоговом окне «Углы», либо прямо на панели управления.

Полностью модернизированный инструмент «Карандаш»

Эта новая технология расширяет все возможности инструментов «Кисть», «Кисть-клякса» и «Сглаживание». С помощью этого инструмента можно более точно вычерчивать кривые, добавляя прямые линии, а также продлевая и замыкая контуры. Предварительно использовать заданные настройки для создания плавных контуров уже с меньшим числом точек или более точной обрисовки мазков натуральной кистью.

Изменение формы сегмента контура

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

Интеграция с Typekit

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

Поддержка ОС Windows 8

Оцените по достоинству поддержку дисплеев HiDPI на компьютерах под управлением ОС Windows 7, 8 и 10. Для рисования перо с функцией определения степени нажатия, которое идет в комплекте с планшетами на базе ОС Windows 8 и 8.1 , а также улучшенную поддержку функции прямого сенсорного ввода на экране.

Настраиваемая панель инструментов

С помощью панель инструментов создавайте специализированные наборы инструментов, перетаскивая на панель только нужные инструменты, например, инструменты для выделения, рисования, или редактирования. Персонализируйте ваше рабочее пространство под определенные нужды и инструменты для определенных задач. Все это можно скрыть за панель «Инструменты», получив больше пространства для своего творчества.

Импорт и экспорт настроек

Функция синхронизации настроек позволит вам стандартизировать настройки на своих устройствах, а также и делиться ими с другими пользователями. Возможность синхронизировать свои настройки приложения Illustrator на нескольких компьютерах. Для этого просто экспортируйте их в ту папку, из которой другие пользователи смогут импортировать их.

Улучшенные возможности рисования с учетом перспективы

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

Обновления функции экспорта в формат SVG

Экспортируйте уже созданные работы в оптимизированный и масштабируемый формат файлов SVG, которые адаптируются к экранам различных размеров, разрешений и соотношений, а затем выполняйте с сохранением точного выравнивания пикселов сквозное редактирование файлов SVG.

Инструмент «Изменение текста»

Теперь можно создавать работы, не только с помощью стилуса или мыши, но и просто касаясь сенсорного экрана мобильного аппарта. Создавать проекты и добавлять в них текст с помощью инструмента «Изменение текста». Работайте теперь с символами, как с отдельными объектами. Экспериментируйте со шрифтами, поворачивайте текст, перемещайте и масштабируйте.

Изображения в кистях

Какой бы кистью Illustrator вы ни воспользовались, благодаря этому внешний вид обводки и форму можно изменить по вашему желанию. Рисуйте кистью, в которую уже помещена фотография. Дискретная, объектная, узорчатая кисти содержат растровые изображения, что позволит создавать сложный дизайн, рисуя обводки, имитирующие мазки натуральной кистью за несколько минут.

Поиск шрифта

В палитре «Символ» вводите уже тот понравившееся стиль шрифта, например, «полужирный», название семейства шрифтов или часть названия шрифта. Быстро находите идеально подходящий шрифт. Отобразятся именно те результаты поиска, которые отвечают вашим параметрам.

Использование нескольких файлов

Теперь можно определить местоположение любых файлов, применить к ним масштабирование, а также воспользоваться новым видом миниатюр, уточняя расположение файла уже в проекте. Импортировать в Illustrator несколько файлов одновременно и управлять процессом с помощью новых функций.

Извлечение каскадных таблиц стилей (CSS)

Копируйте и вставляйте код уже прямо в веб-редактор. Написание кода для таких элементов, как узоры или значки, может быть очень утомительным

процессом. Однако, сейчас создавать вебсайты стало еще проще, программа сама создает код CSS даже для логотипов, которые включают в себя градиенты.

Синхронизация цвета

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

Преобразование текста из точки в текст в области и наоборот

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

Автоуглы для узорчатых кистей

Больше не нужно заниматься утомительным созданием специальных острых углов в проекте. Теперь это сделать можно за несколько шагов. Создавайте узорчатые кисти всего за несколько секунд, благодаря функции автоматического создания углов, идеально подойдящие к остальному оформлению обводки.

Применение инструмента «Свободное трансформирование»

Масштабируйте, перемещайте и поворачивайте объекты прямо на сенсорном экране смартфона. Совершенствуйте ваши навыки в создании дизайна с помощью эффективного инструмента «Свободное трансформирование». Либо воспользуйтесь мышью или другим манипулятором, чтобы быстро и интуитивно трансформировать объекты прямо на монтажной области.

Интеграция с Behance

Сохраняйте вашу работу прямо из программы Illustrator CC в удобный сервис для дизайнеров Behance, чтобы продемонстрировать и опубликовать свои готовые проекты. По мере доработки проекта загружайте новые версии и моментально получайте отзыв о работах от других дизайнеров со всего мира.

Синхронизация настроек

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

Упаковка файлов

Функция упаковки файлов позволяет пользователю автоматически собрать все необходимые шрифты, связать графику и отправить отчет об упаковке в одной папке. Можно воспользоваться упаковкой файлов для того, чтобы сдачи проектов заказчику были вовремя или их систематизировать на компьютере.

Извлечение изображений

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

Использование нескольких монтажных областей

Сохраняйте монтажные области, а также экспортируйте и печатайте их по отдельности или вместе. Упорядочивайте и просматривайте до 200 монтажных областей любых размеров, расположенных в виде сетки или каскадом. С легкостью добавляйте, переименовывайте и удаляйте области, а также меняйте их порядок расположения.

Обводки переменной ширины

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

Изображения

Простые, понятные функции обеспечивают лучшую четкость подгонки, точность линий, и получение надежных результатов работы. При помощи эффективного механизма трассировки, обеспечивающего исключительный уровень контроля работы с цветами и фигурами, преобразуйте растровые изображения в редактируемые векторы.

Система Adobe Mercury Performance

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

2.3 Инструменты разработки кода приложения

Используемая среда для разработки для написания кода программы будет Android Studio, она уникальна тем, что все встроенные библиотеки уже подключены, программирование приложения будет производится на языке Java. Для того, чтобы приступить к работе на Android Studio, необходимо наличие установленного Java SE Development Kit (JDK) на компьютере. Рекомендуемая версия JDKдолжна быть не ниже 8 версии. Установить его можно следуя по ссылке.

Java SE Development Kit – это бесплатный инструментарий для разработки и распространяется он бесплатно. Для установки вам необходимо выбрать версию, соответствующую вашей ОС: Windows, Linux или Mac OS X. После загрузки файла, следуя инструкциям его необходимо запустить и, установить данную среду.

После этого следует загрузить уже с официального сайта Android для разработчиков специальный установочный файл Android Studio.

Также необходимо выбрать нужную версию, подходящую для вашей ОС, после окончания загрузки файла, прибегая к установке программы следующие инструкции. Загрузка Android NDK для работы в Android Studio не требуется, так как все утилиты, необходимые в работе, уже включены в комплект поставки приложения. После запуска приложения, выплывает экран приветствия, изображенного на рисунке 2.7.

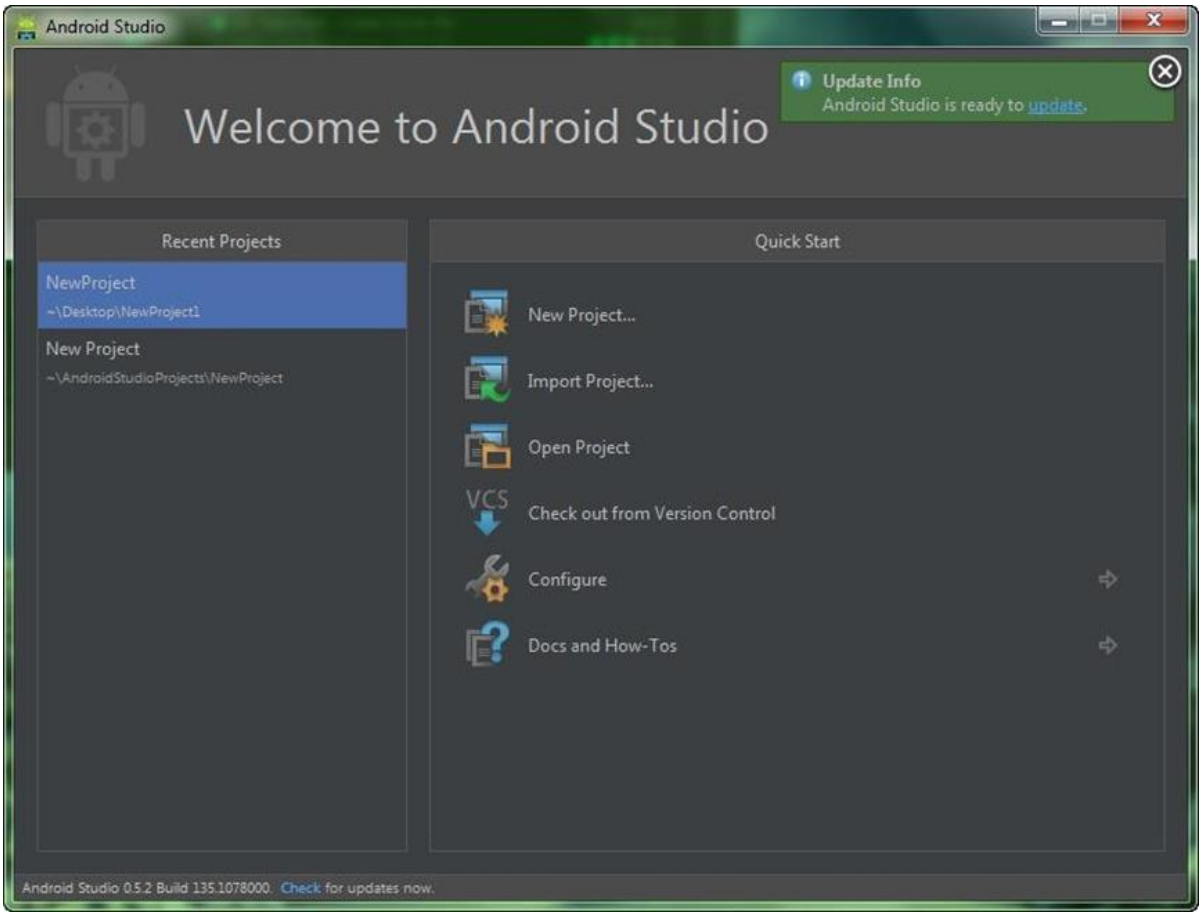


Рисунок 2.7 – Экран приветствия Android Studio

Для создания нового проекта необходимо нажать «New Project…» и за этим последует окно создания проекта, показанное на рисунке 2.8.

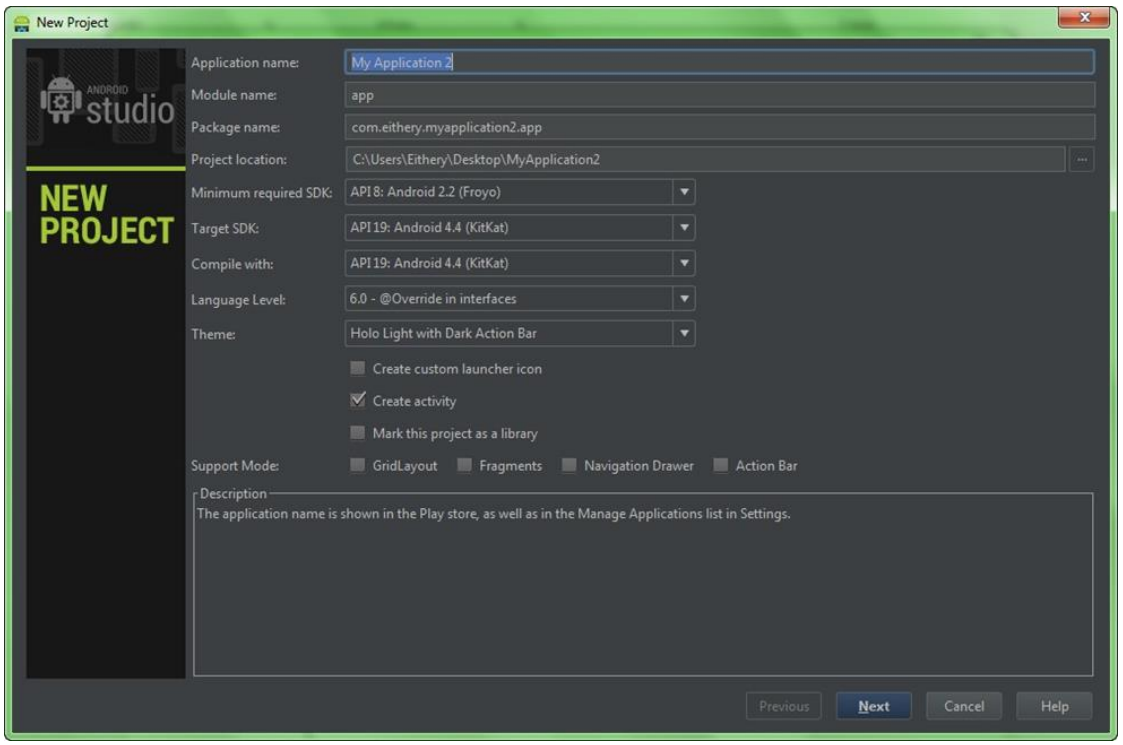


Рисунок 2.8 – Окно создания проекта Android Studio

В этом окне дается для того, чтобы задать имя приложения, название модуля, название файла для приложения, включая место расположения каталога с данными проекта. Также имеется уникальная возможность выбора минимальной версии ОС, которая поддерживается приложением и основной, на которую оно в первую очередь ориентированно. Помимо всего, есть возможность варианта выбора уровня языка. Доступен выбор темы приложения: светлое и темное оформление. Если требуется, то можно поставить маркер в бокс выбора «Create custom launcher icon», для создания непосредственно в Android Studio иконки приложения. В заключение есть возможность выбора «Support Mode».

Если в окне создания проекта «Create custom launcher icon» было выбрано, то после нажатия кнопки «next», вы увидите окно создания и редактирования значка, которое также изображено на рисунке 2.9. Этот редактор удовлетворит потребности практически всех разработчиков, поскольку позволяет создать быстро и качественно значки и иконки приложения для разных плотностей экрана и, где сразу же можно увидеть результат.

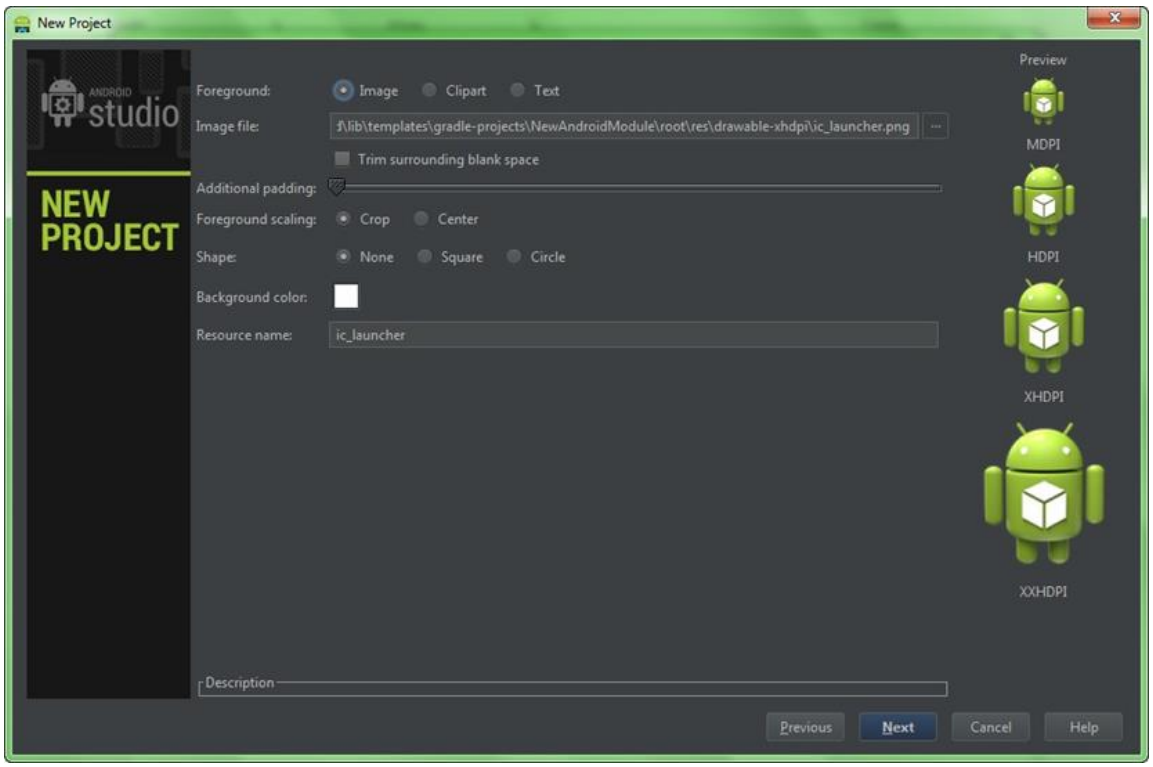


Рисунок 2.9 – Окно создания значка для приложения

После завершения этапа создания значка, переходите на экран выбора базовой activity, это наглядно изображено на рисунке 2.10.

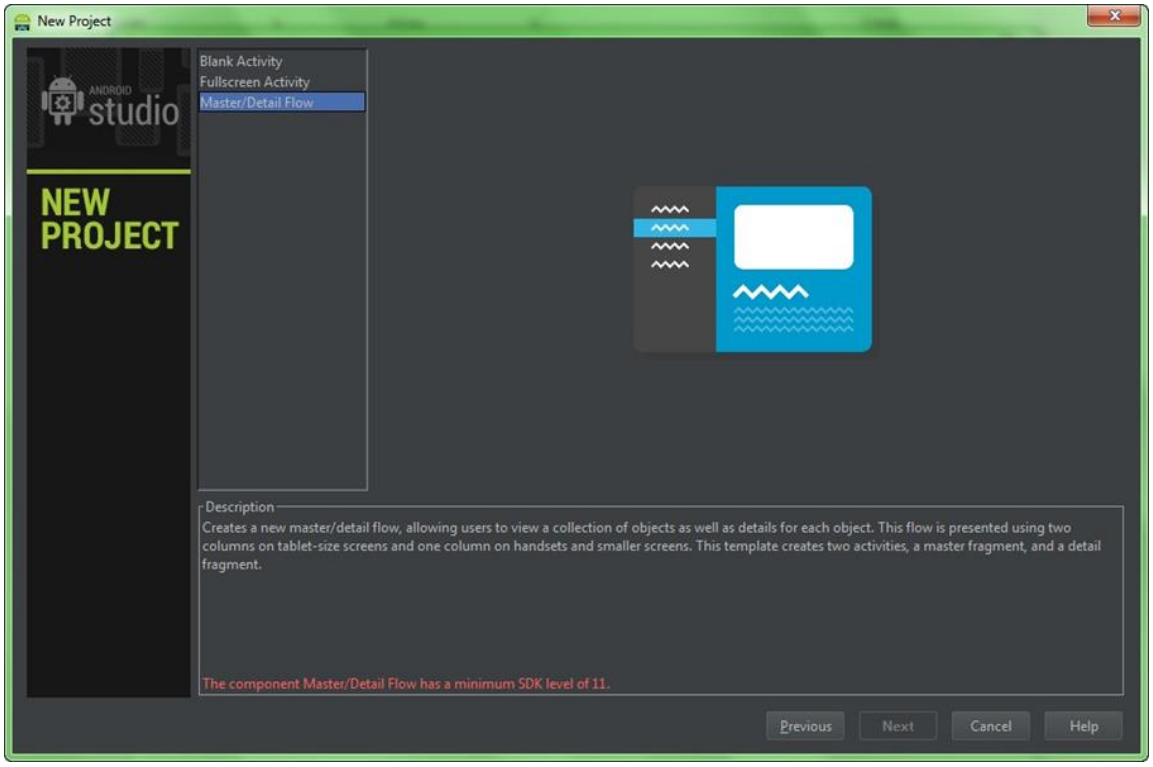


Рисунок 2.10 – Окно выбора базовой Activity

Есть три варианта по выбору:

– Blank Activity.
– Fullscreen Activity.
– Master/Detail Activity.

Далее нужно указать имя нашей стартовой Activity и имя layout, а также имя первого фрагмента, поскольку мы выбрали поддержку фрагментов при создании проекта. Также можно выбрать тип навигации по Activity, представленным на рисунке 2.11.

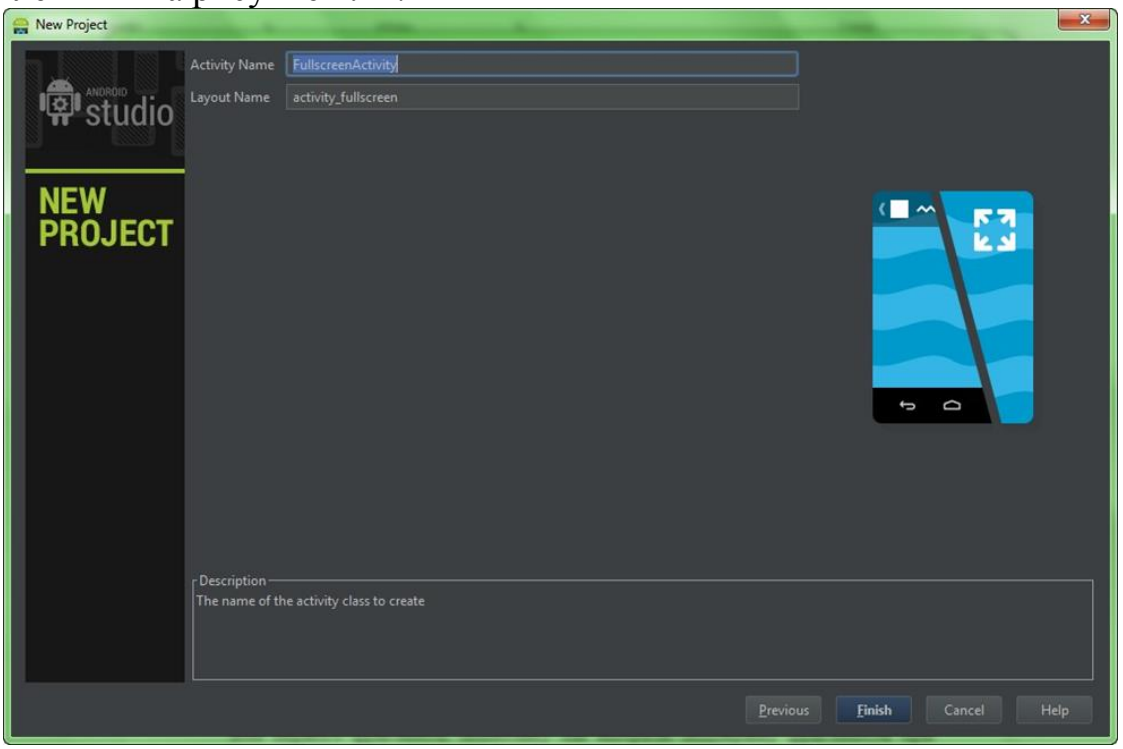


Рисунок 2.11 – Ввод имени нашей стартовой Activity и имени её layout

После этого следующим окном будет сама среда разработки(IDE).

Новая структура проекта Android Studio обязана новой системе сборки проекта – Gradle. Слева находится меню, где изображены папки src и res, но res теперь лежит внутри папки src, наравне с новой папкой java, в которую перенесли уже наши пакеты и классы. Она помогает разработчику управлять зависимостями в нашем проекте, подключать внешние библиотеки.

Хочется отметить, что всё так же отлично работает при импорте проектов с обычной структурой

Основное рабочее окно среды разработки хорошо изображено на рисунке 2.12.

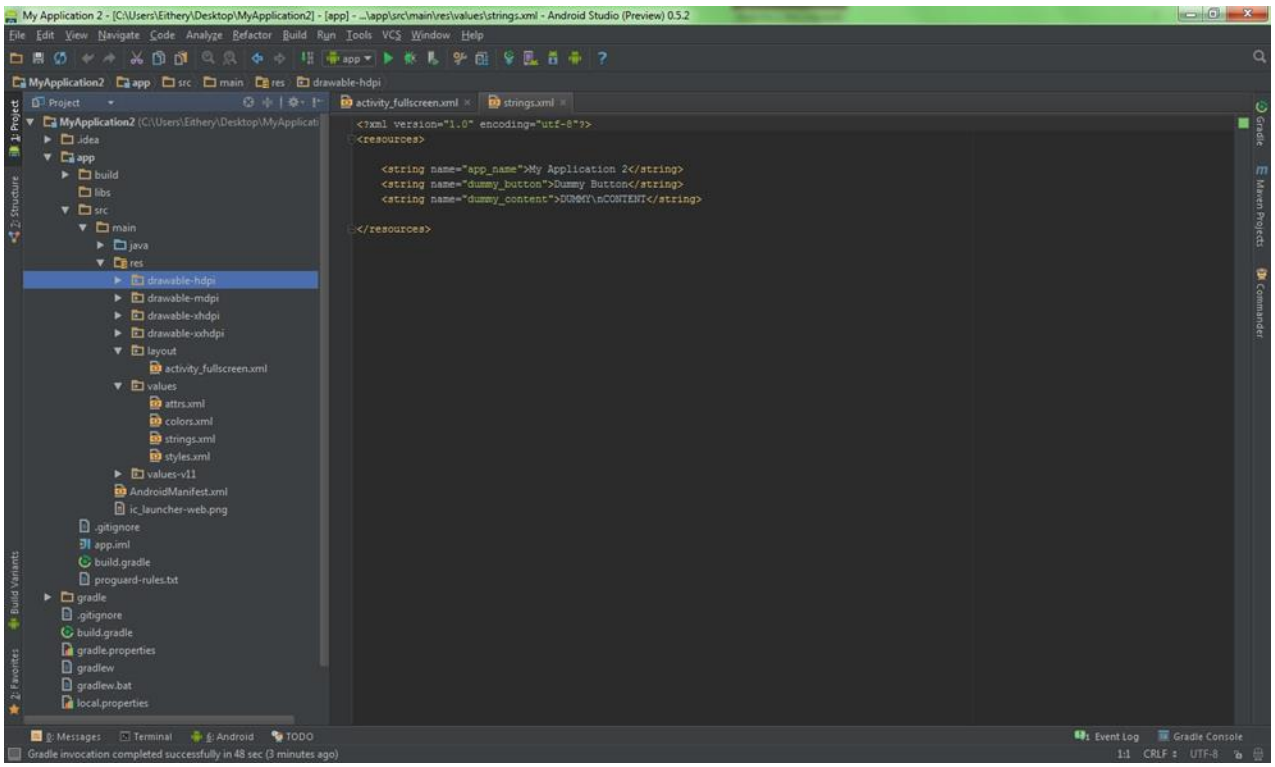


Рисунок 2.12 – Основное окно IDE

XML редактор наглядно изображен на рисунке 2.13

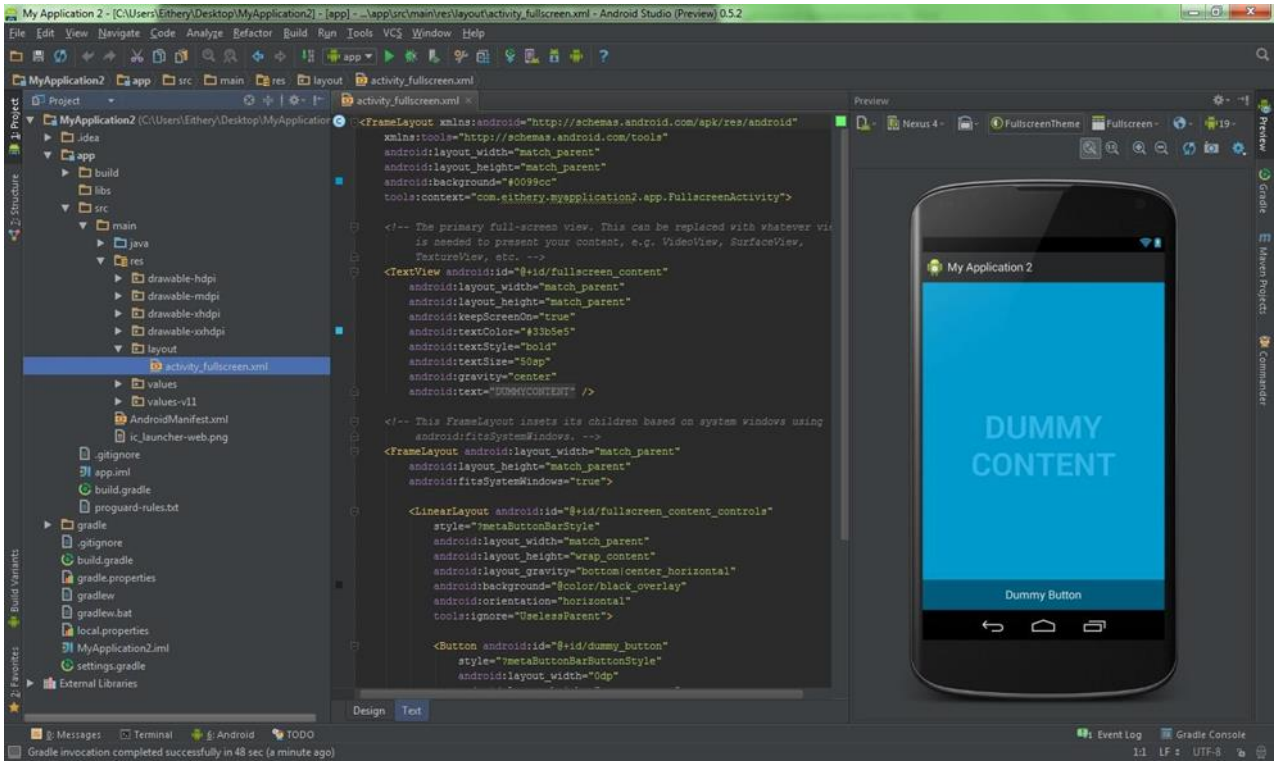


Рисунок 2.13 – Пример окна XML ректора

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.

Добавим 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. [5]

Студия предлагает на выбор разработчику несколько таких объектов для создания:

– 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 Android компонентов

Рисунок 2.14 демонстрирует окна для создания нового компонента приложения.

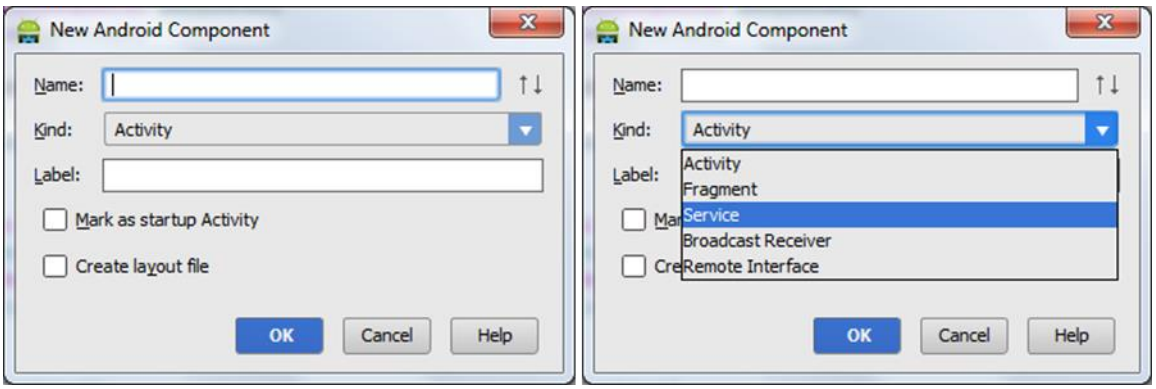


Рисунок 2.14 – Окно создания нового компонента

Package-info.java – файл описания информации про пакет. HTML File – собственно создает html файл.

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

По умолчанию регистрирует компонент в Манифесте, при необходимости.

Android Component – это универсальная штука, позволяющая создать любой из ключевых компонентов нашего Android, чтобы сразу создать layout для него при необходимости, а также установить Activity как стартовую.

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

– Встроенное подключение Android Sources.

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

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

– Интеллектуальный анализ кода.

3 Практическая часть

3.1 Анализ магазина приложений Google Play

Магазин приложений подразумевает под собой не только размещение приложений, но организацию их поиска, разделение на категории, выставление оценок и получение отзывов. Наличие богатого ассортимента в магазине приложений является главным критерием успешного продвижения операционной системы. Google Play — это магазин приложений от компании Google, который позволяет владельцам Android-устройств загружать и устанавливать любые приложения на свой смартфон на любой вкус для самых разнообразных задач. Google Play появился на свет в следствии ребрендинга Android Market в марте 2012 года. К сожалению, в следствие этого, новые возможности доступны только жителям США, но Google обещает в скором времени наладить это положение дел. Разработчики приложений, загружаемых в Google Play, получают до 70% прибыли от их продаж. Google Play был создан, как аналог App Store для iOS и Windows Phone Store для мобильных устройств для базе ОС Windows Phone.

Магазин приложений Google Play не является крайне новым сервисом IT-гиганта Google. Потребности современных пользователей всегда растут, на сегодня каждому нужны удобные инструменты для работы и развлечений. И новая операционная система Android - это ответ на вызов времени. Абсолютно логично, что и магазин приложений предстал в новом варианте, подходящим под требования современного пользователя.

Сегодня Google Play является одним из самых быстрорастущих онлайн-магазинов. Ежедневно в сервис разгружается огромное количество новинок всевозможных приложений. В среднем, за сутки загружается более 1000 новых игр и программ. К началу этого года общее количество приложений в магазине для Android достигло 400 млн штук . Идет постоянное расширение географии магазина: в Google Play распространять приложения бесплатно могут граждане более 70 государств. В блоге для разработчиков Google было объявлено о том, что в онлайн-магазине все приложения, оптимизированные под планшеты, получат обособленный, отдельный раздел.

По большому счету в Google Play можно найти всё что нужно для вашего досуга и бизнеса: игры, полезные утилиты, приложения для работы, программы для навигации, карты, виджеты, обои и т.д, то есть одним словом, каждый пользователь сможет выбрать нужное приложение для своего Android смартфона. Доступ и выбор программ организован очень удобно, все приложения разбиты по категориям: всего их 34, включая Игры. Здесь представлены списки как и платных, так и бесплатных программ, новинок, а также ТОП игр.

Внешний вид магазина через интернет-браузер выглядит следующим образом, представленным на рисунке 3.1.

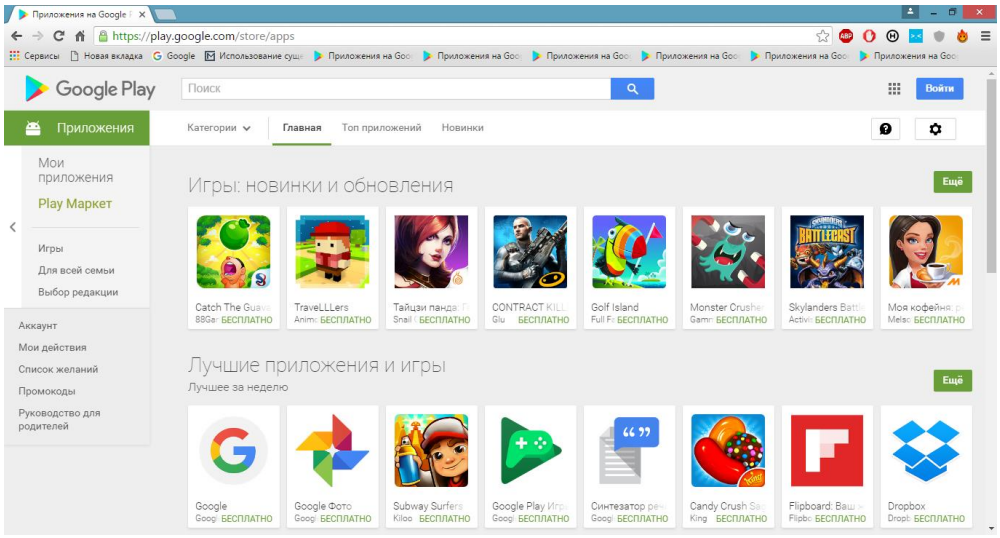


Рисунок 3.1 - Внешний вид Google Play

Визуально Google Play представляет собой несколько экранов, где главный экран содержит ссылки на программы и игры. Также здесь может быть представлен каталог программ от производителя устройства. Вид магазина с мобильного устройства изображен на рисунке 3.2.

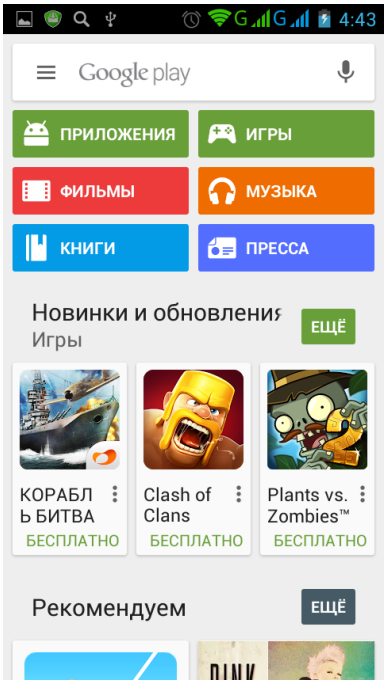


Рисунок 3.2 - Внешний вид Google Play на мобильном устройстве

Для того чтобы начать совершать покупки в Google Play, необходимо зарегистрироваться с помощью почтового ящика, привязать к своему аккаунту, электронную карту оплаты, в последующем все расчеты можно будет производить только в случае, если у пользователя есть учетная запись в этой системе. Также в числе других преимуществ Google Play – это наличие ссылки на все остальные приложения разработчика в выбранном окне

приложения. Таким образом, пользователь сможет оценить и другие продукты данного производителя, если человеку понравилось одно, то вероятно, понравится и другое.

3.2 Обзор и анализ аналогов приложений

Тематикой приложения будет справочник для языка программирования С++.В данный момент рынок очень остро нуждается в высоко квалифицированных специалистов в области программирования С++, это обусловлено тем, что не хватает специальной адаптированной литературы для обучения. Программирование на С++ являются одной из самых актуальных тем. Особенно это важно для новичков студентов. Ведь для них программирование всегда сопряжено с постоянным вниманием и напряжением. Но и опытные программисты тоже должны хорошо ориентироваться в постоянно изменяющихся условиях развития языка. Еще пару десятилетий тому назад никто и представить не мог, что книга может помещаться в любом кармане. Сегодня это стало реальностью, благодаря развитию мобильных технологий. Теперь в решении данного вопроса может помочь приложение «Справочник по С++».

На рынке мобильных приложений существует множество приложений на данную тематику, для этого нам необходимо проанализировать аналогичные приложения в Google Play. В магазине были найдены два аналогичных приложения, которые показаны на рисунке 3.3.

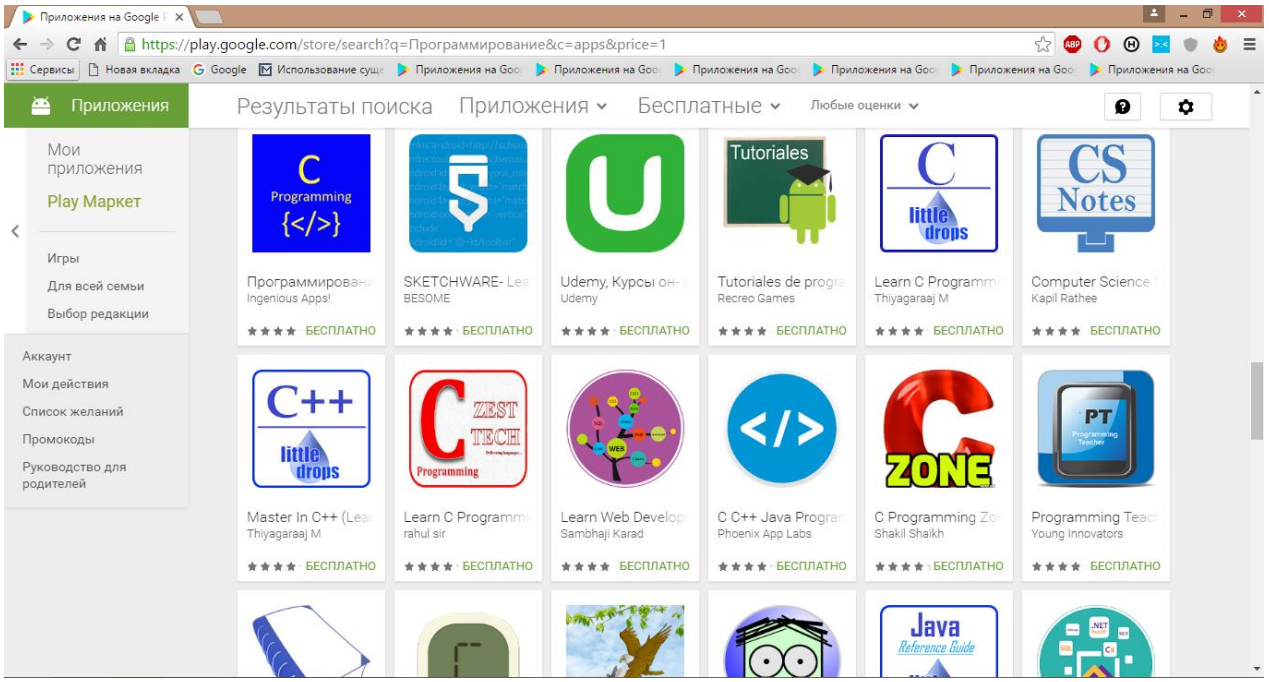


Рисунок 3.3 - Похожие приложения Android

Основным минусом данных приложений является отсутствие поддержки нескольких языков обучения.

На рисунке 3.4 изображена страничка установки приложения «C++ programming».

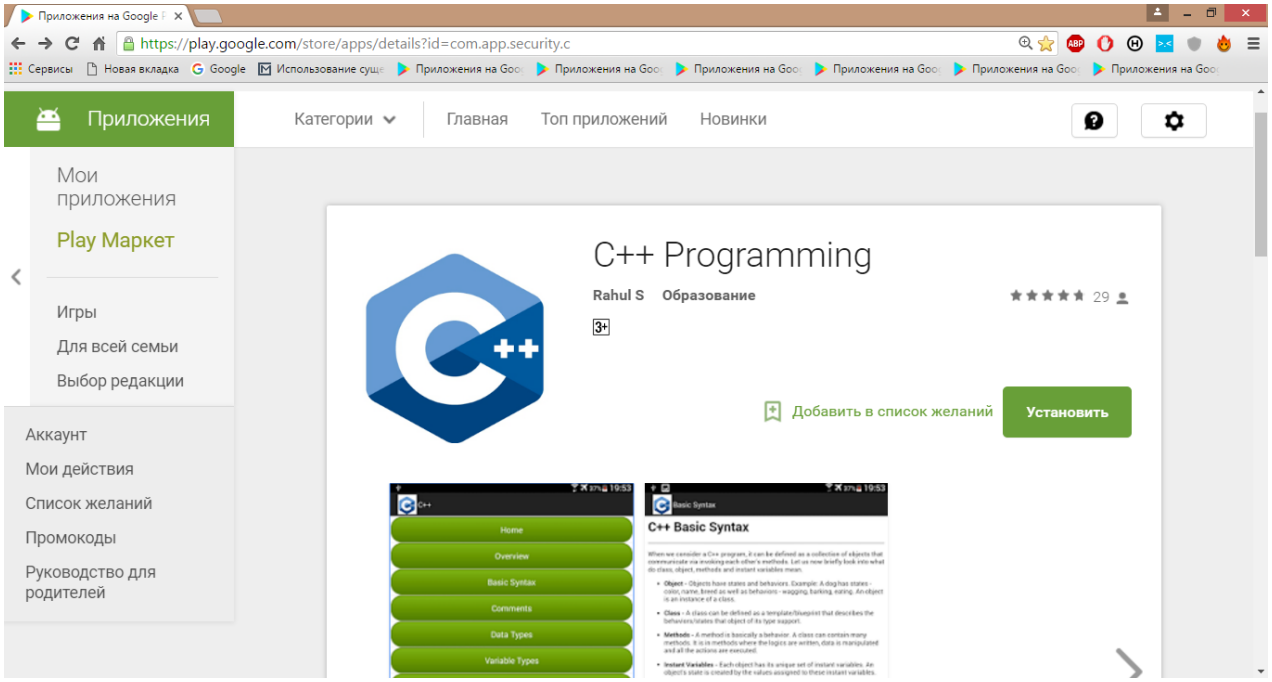


Рисунок 3.4 - Страница установки приложения «C++ programming»

3.3 Проектирование приложения

«Справочник по C++» – это справочное приложение, предоставляющее базовые знания и основы курса изучения языка программирования C++. Приложение имеет примеры различных задач и программ с использованием функции подцветки кода и контрольные вопросы для самопроверки и закрепления приобретенных знаний. Разработанное приложение имеет удобный и яркий интерфейс с применением единого стиля Material Design. Material Design — это единая логика построения приложения и его внешнего оформления. Внешний вид этого приложения должен также сочетаться с другими элементами оформления пользовательского интерфейса, передовая задуманный разработчиками операционной системы опыт использования. Дизайн «Справочник по C++» должен следовать установленным требованиям и нормам платформ держателя от компании Google.

Поэтому используя все вышеописанные стандарты, реализуется концепция интерфейса программы «Справочник по C++».

Интерфейс приложения состоит следующих компонентов:

– Главное меню;
– Меню Обучения;
– Меню Программ;
– Меню Контрольных вопросов;
– Менюбар (SlidingMenu);

Рисунок 3.5 – Главный экран приложения с выбором языка

При запуске приложения отображается главное меню программы, где на экране располагается логотип приложения, ниже представлено приветствие пользователя на русском и английском языках и расположены две кнопки для того, чтобы студент мог выбрать язык программы, это наглядно отражена на рисунке 3.5.

Следующая форма открывается меню Обучения, все управление в приложение между формами: меню Обучение, меню Программы и меню Вопросы производится с помощью таскбара (taskbar). В приложении очень интересна реализация переключения между окнами, все действия производятся в одно касание, пожалуй многим пользователям это придется по вкусу, рисунок 3.6 наглядно показывает все выше перечисленные манипуляции.

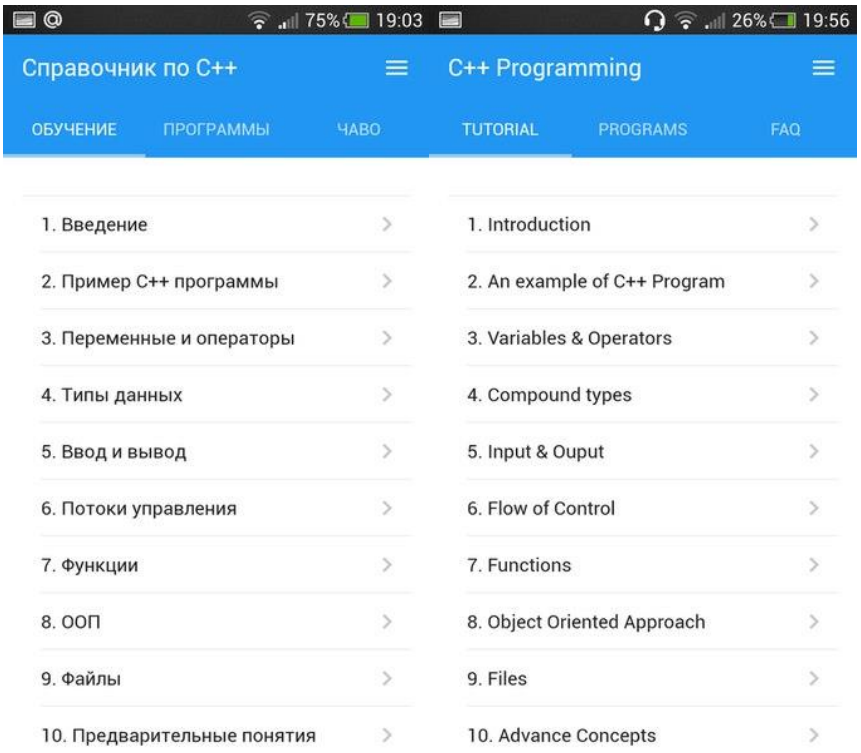


Рисунок 3.6 – Меню Обучение в двух языках

Материал в приложении специально разделен по главам, где каждая глава состоит из нескольких разделов, посвященных ключевым понятиям, все это реализовано с помощью элемента управления вертикальное меню в стиле "Аккордеон", изображённое на рисунке 3.7.

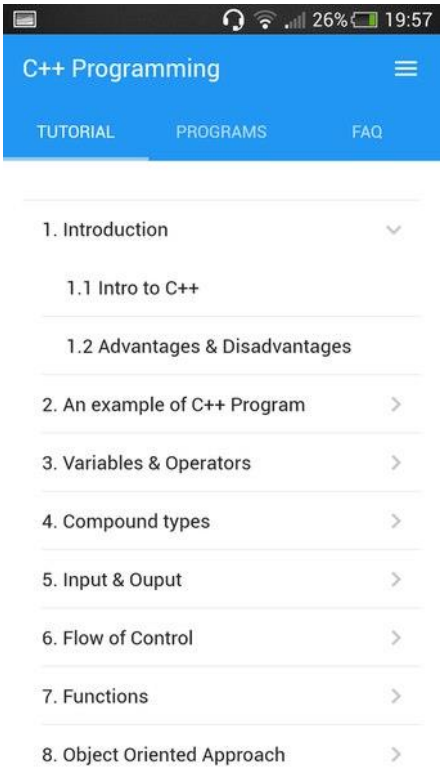


Рисунок 3.7 – Вертикальное меню «Аккордеон»

Чтобы помочь пользователю сконцентрироваться на важных аспектах в процессе изучения курса, имеется возможность поделиться материалом любого раздела. Для начала нужно нажать на специальную кнопку «Функционал», в правом нижнем углу, которая раскроется на две кнопки: «Настройки» и «Поделиться». При помощи последнего кнопки можно поделиться в любые доступные социальные сети. Представленная функция одна из незаменимых и удобных в современное время приложениях, так как обеспечивает своевременное освоение нужного материала. Все управление в программе представлено на рисунке 3.8.

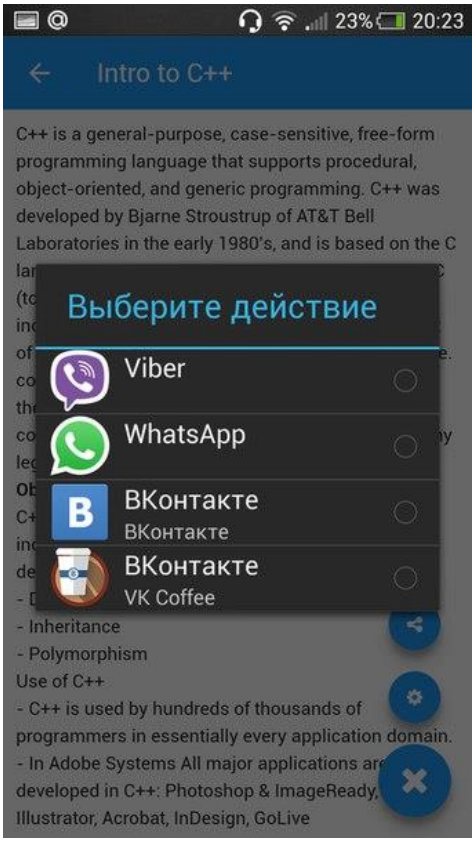


Рисунок 3.8 – Фукция «Поделиться»

При переходе в меню Программы собран список примеров популярных программ для обучения. Эти программы могут продемонстрировать все различные особенности работы на языке C++. Это позволит студенту сразу приспособляться и применять свои знания языка и упражняться на конкретном примере. Меню «Программы» представлено на рисунке 3.9.

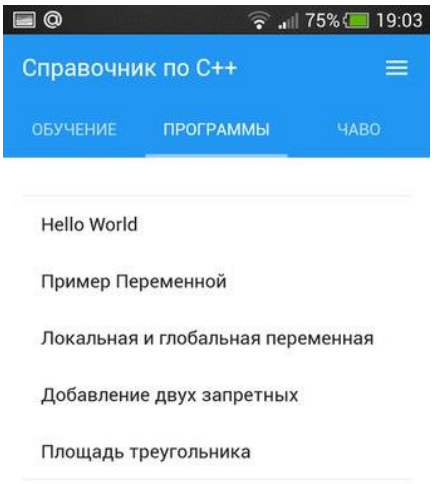


Рисунок 3.9 – Меню «Программы»

Далее можно выбрать одно упражнение из списка программ. При переходе в следующее активити, выдается решение программы на языке C++ с подцветкой синтаксиса кода. Такая технология, изображенная на рисунке 3.10, удобна и незаменима в современных программах для обучения, так как значительно сокращает время на рассмотрение и понимание кода.

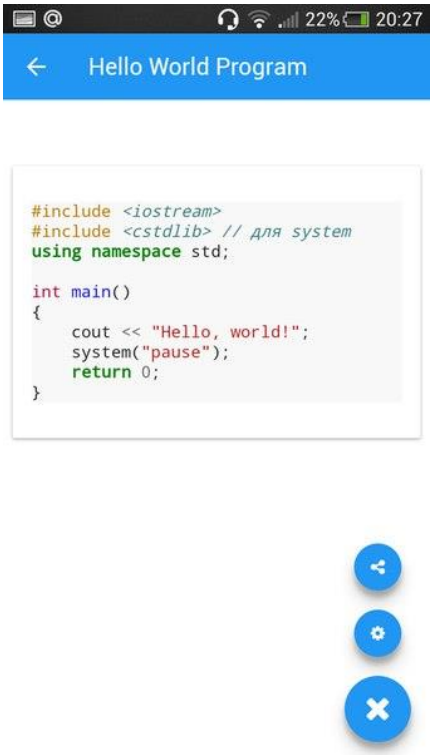


Рисунок 3.10 – Код программы «Hello World»

В главном меню также есть компонента MenuBar, который изображен на рисунке 3.11. При нажатии на иконку MenuBar, с правой стороны появляется выдвижное меню, в котором есть элементы управления Settings, About и Exit.

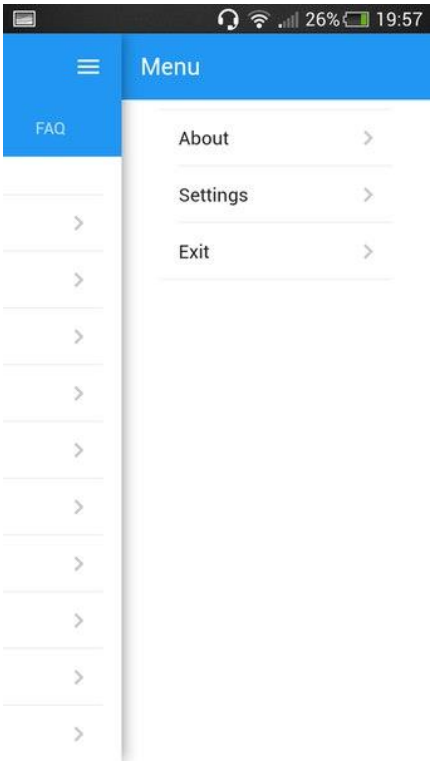


Рисунок 3.11 – Выдвижное меню Менюбар

В разделе About расположна информация о разработчике приложения. Окно About представлен на рисунке 3.12.

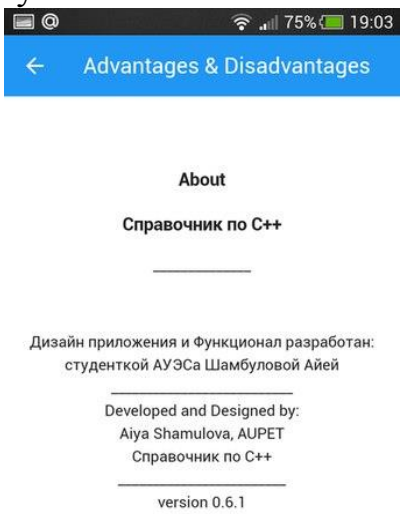


Рисунок 3.5 – Окно About

В разделе управления Settings можно изменить тему интерфейса приложения. На выбор пользователю дается три темы различных цветов: фиолетовый, голубой и розовый. Под пунктом «По умолчанию» подразумевается голубой цвет, потому что он продолжает единый стиль иконки приложения. Возможность редактирования темы оформления предоставлена на рисунке 3.13.

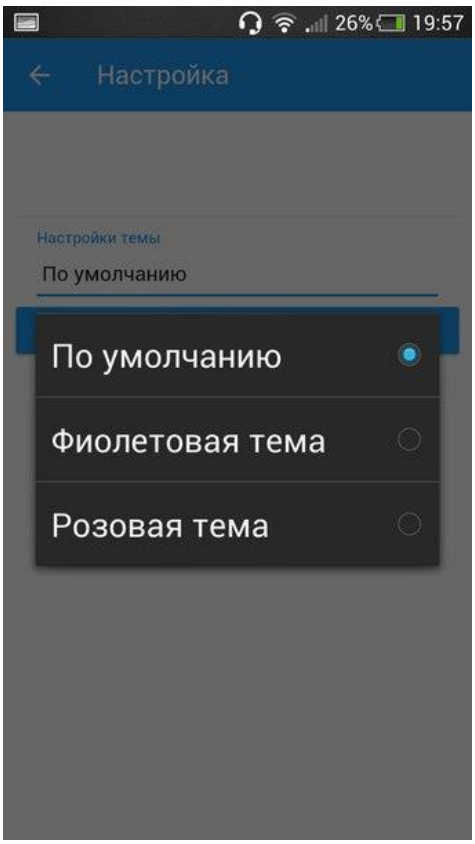


Рисунок 3.13 – Настройки Темы

3.4 Разработка приложения

Проект дипломной работы состоит из нескольких папок и других файлов. Основные из них следующие:

– *gen* – содержаться файлы сгенерированные самой Java;

– *src* – представлен исходный код на Java. Здесь находится основной файл для работы. Здесь же будут находиться новые классы;

– *res* – хранятся файлы ресурсов. Он в свою очередь содержит несколько подкаталогов:

1 res/drawable-dpi – в этих папках содержатся ресурсы, предназначенные для разных расширений экрана. Если зайти в каждую папку, то можно найти там файл *ic_launcher.png*, который является значком вашего приложения. В папке drawable-ldpi ничего нет, так как это папка для старых телефонов, которые уже не стоит поддерживать.

2 res/layout – в данной папке содержатся xml-файлы, описывающие

внешний вид форм и различных элементов форм. После создания проекта там уже имеются файлы activity_main.xml и fragment_main.xml.

3 res/menu – здесь находятся ресурсы для меню.

4 res/values – тут у нас располагаются какие-либо строковые ресурсы, ресурсы цветов, тем, стилей и измерений, которые мы можем использовать в нашем проекте.

При разработке приложения использовалась программа Illustrator CC для обрисовки отдельных компонентов приложения:

- иконка приложения;
- маркеры файлов;
- иконки различных кнопок: меню, функционала, поделиться, справки и другие.

Для реализации форм в меню выбора файла был создан файл activity_main.xml в каталоге res/layout.

```
        <?xml version="1.0" encoding="utf-8"?>
<widget xmlns:android="http://cordova.apache.org/ns/1.0"
id="io.cordova.hellocordova" version="0.0.1">
    <preference name="loglevel" value="DEBUG" />
    <feature name="Whitelist">
        <param name="android-package"
value="org.apache.cordova.whitelist.WhitelistPlugin" />
        <param name="onload" value="true" />
    </http://www.w3.org/ns/widgets:feature>
    <preference name="webView" value="org.crosswalk.engine.XWalkWebViewEngine"
/>
    <preference default="17+" name="xwalkVersion" />
    <preference default="--disable-pull-to-refresh-effect"
name="xwalkCommandLine" />
    <preference default="embedded" name="xwalkMode" />
    <preference default="true" name="xwalkMultipleApk" />
    <feature name="SocialSharing">
        <param name="android-package"
value="nl.xservices.plugins.SocialSharing" />
    </http://www.w3.org/ns/widgets:feature>
    <allow-intent href="market:*" />
    <name
        C++ Programming />
    <description
        A sample Apache Cordova application that responds to the deviceready
event. />
    <author email="dev@cordova.apache.org" href="http://cordova.io"
        Apache Cordova Team />
    <content src="index.html" />
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <preference name="xwalkVersion" value="17+" />
    <preference name="xwalkCommandLine" value="--disable-pull-to-refresh-
effect" />
```

```
    <preference name="xwalkMode" value="embedded" />
    <preference name="xwalkMultipleApk" value="true" />
</http://www.w3.org/ns/widgets:widget>
```

Форма отображения элемента управления выдвижного меню реализовывалось в файле select_action_menu.xml

```
        <?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:icon="?unknown_attr_ref: 101037e"
android:id="@+id/select_action_menu_select_all" android:title="0x104000d"
android:alphabeticShortcut="a" android:showAsAction="always|withText" />
    <item android:icon="?unknown_attr_ref: 1010311"
android:id="@+id/select_action_menu_cut" android:title="0x1040003"
android:alphabeticShortcut="x" android:showAsAction="always|withText" />
    <item android:icon="?unknown_attr_ref: 1010312"
android:id="@+id/select_action_menu_copy" android:title="0x1040001"
android:alphabeticShortcut="c" android:showAsAction="always|withText" />
    <item android:icon="?unknown_attr_ref: 1010313"
android:id="@+id/select_action_menu_paste" android:title="0x104000b"
android:alphabeticShortcut="v" android:showAsAction="always|withText" />
    <item android:id="@+id/select_action_menu_share"
android:title="@string/actionbar_share" android:showAsAction="always|withText"
style="@style/SelectActionMenuShare" />
    <item android:id="@+id/select_action_menu_web_search"
android:title="@string/actionbar_web_search"
android:showAsAction="always|withText"
style="@style/SelectActionMenuWebSearch" />
</menu>
```

При создании интерфейса приложения используют графический редактор дизайна. Пример XML файла в графическом редакторе изображён на рисунке 3.14.

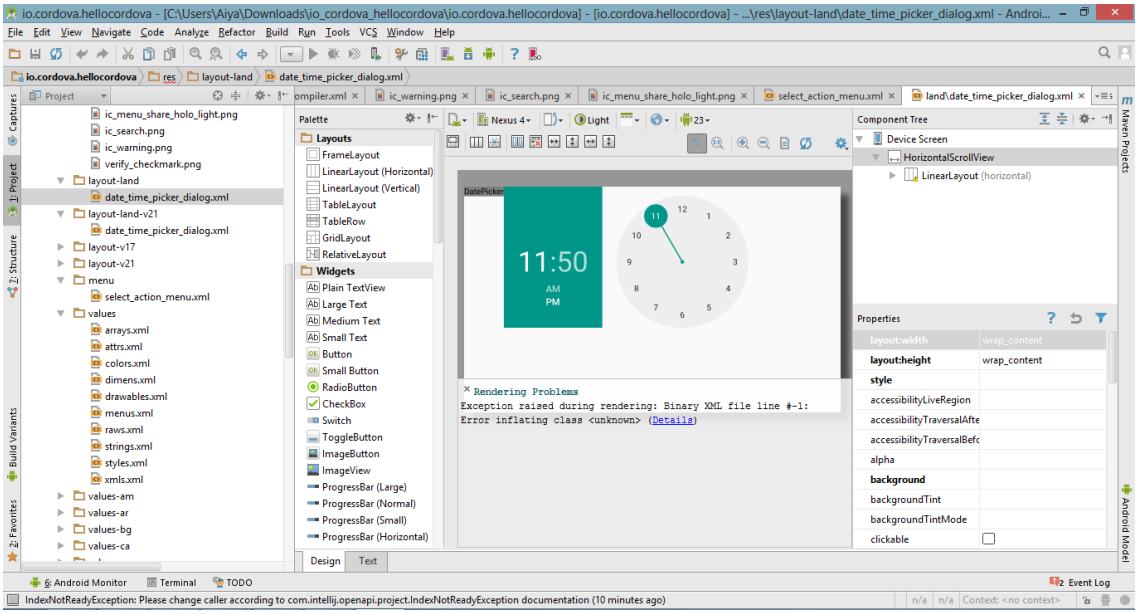


Рисунок 3.6 – Графическое изображение XML файла

3.5 Создание и реализация классов

В процессе разработки были созданы и реализованы множество классов:

- MainAtivity – главное окно;
- SocialSharing – класс для функции «Поделиться»;
- BuildConfig – класс для функции создание конфигураций языка;
- NavigationParams– класс для навигации;
- ContentMain– класс для контент меню;
- ContentView– класс для просмотра контент меню;
- ViewConfigurationHelper– класс для функции «Настойки»;
- TwoFieldDatePickerDialog – класс для функции «Поделиться»;
- ColorPickerDialog– класс для настройки темы;
- DropdownAdapter– класс для функции «Выход»;

Несколько примеров классов из программы реализации:

```
        package io.cordova.hellocordova;

import android.os.Bundle;
import org.apache.cordova.CordovaActivity;

public class MainActivity extends CordovaActivity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        loadUrl(this.launchUrl);
    }
}
```

```
        package io.cordova.hellocordova;

public final class BuildConfig {
    public static final String APPLICATION_ID = "io.cordova.hellocordova";
    public static final String BUILD_TYPE = "debug";
    public static final boolean DEBUG = Boolean.parseBoolean("true");
    public static final String FLAVOR = "armv7";
    public static final int VERSION_CODE = 12;
    public static final String VERSION_NAME = "0.0.1";
}
```

```
        package nl.xservices.plugins;
import …
        public class SocialSharing extends CordovaPlugin {
    private static final String ACTION_AVAILABLE_EVENT = "available";
    private static final String ACTION_CAN_SHARE_VIA = "canShareVia";
    private static final String ACTION_CAN_SHARE_VIA_EMAIL =
"canShareViaEmail";
    private static final String ACTION_SHARE_EVENT = "share";
    private static final String ACTION_SHARE_VIA = "shareVia";
    private static final String ACTION_SHARE_VIA_EMAIL_EVENT =
"shareViaEmail";
    private static final String ACTION_SHARE_VIA_FACEBOOK_EVENT =
"shareViaFacebook";
```

```
    private boolean isEmailAvailable() {
        if
(this.cordova.getActivity().getPackageManager().queryIntentActivities(new
Intent("android.intent.action.SENDTO", Uri.fromParts("mailto",
"someone@domain.com", null)), 0).size() > 0) {
            return true;
        }
        return false;
    }

        });
        return true;
    }



public class NavigationParams {
    public final boolean hasUserGesture;
    public final boolean hasUserGestureCarryover;
    public final boolean isExternalProtocol;
    public final boolean isMainFrame;
    public final boolean isPost;
    public final boolean isRedirect;
    public final int pageTransitionType;
    public final String referrer;
    public final String url;

    public NavigationParams(String url, String referrer, boolean isPost,
boolean hasUserGesture, int pageTransitionType, boolean isRedirect, boolean
isExternalProtocol, boolean isMainFrame, boolean hasUserGestureCarryover) {
        this.url = url;
        if (TextUtils.isEmpty(referrer)) {
            referrer = null;
        }
        this.referrer = referrer;
        this.isPost = isPost;
        this.hasUserGesture = hasUserGesture;
        this.pageTransitionType = pageTransitionType;
        this.isRedirect = isRedirect;
        this.isExternalProtocol = isExternalProtocol;
        this.isMainFrame = isMainFrame;
        this.hasUserGestureCarryover = hasUserGestureCarryover;
    }

    @CalledByNative
    public static NavigationParams create(String url, String referrer, boolean
isPost, boolean hasUserGesture, int pageTransitionType, boolean isRedirect,
boolean isExternalProtocol, boolean isMainFrame, boolean
hasUserGestureCarryover) {
        return new NavigationParams(url, referrer, isPost, hasUserGesture,
pageTransitionType, isRedirect, isExternalProtocol, isMainFrame,
hasUserGestureCarryover);
    }
}


       package org.chromium.content.app;

        import android.content.Context;
        import org.chromium.base.annotations.JNINamespace;

@JNINamespace("content")
```

```
public class ContentMain {
    private static native void nativeInitApplicationContext(Context context);

    private static native int nativeStart();

    public static void initApplicationContext(Context context) {
        nativeInitApplicationContext(context);
    }

    public static int start() {
        return nativeStart();
    }
}

public class DropdownAdapter extends ArrayAdapter<DropdownItem> {
    private boolean mAreAllItemsEnabled = checkAreAllItemsEnabled();
    private Context mContext;
    private Set<Integer> mSeparators;

    public DropdownAdapter(Context context, List<DropdownItem> items,
Set<Integer> separators) {
        super(context, R.layout.dropdown_item, items);
        this.mSeparators = separators;
        this.mContext = context;
    }

    public DropdownAdapter(Context context, DropdownItem[] items, Set<Integer>
separators) {
        super(context, R.layout.dropdown_item, items);
        this.mSeparators = separators;
        this.mContext = context;
    }

    private boolean checkAreAllItemsEnabled() {
        for (int i = 0; i < getCount(); i++) {
            DropdownItem item = (DropdownItem) getItem(i);
            if (item.isEnabled() && !item.isGroupHeader()) {
                return false;
            }
        }
        return true;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        View layout = convertView;
        if (convertView == null) {
            layout = ((LayoutInflater)
this.mContext.getSystemService("layout_inflater")).inflate(R.layout.dropdown_i
tem, null);
            ApiCompatibilityUtils.setBackgroundForView(layout, new
DropdownDividerDrawable());
        }
        DropdownDividerDrawable divider = (DropdownDividerDrawable)
layout.getBackground();
        int height =
this.mContext.getResources().getDimensionPixelSize(R.dimen.dropdown_item_heigh
t);
        if (position == 0) {
            divider.setColor(0);
        } else {
            int dividerHeight =
this.mContext.getResources().getDimensionPixelSize(R.dimen.dropdown_item_divid
er_height);
```

```
            height += dividerHeight;
            divider.setHeight(dividerHeight);
            if (this.mSeparators == null ||
!this.mSeparators.contains(Integer.valueOf(position))) {

divider.setColor(this.mContext.getResources().getColor(R.color.dropdown_divide
r_color));
            } else {

divider.setColor(this.mContext.getResources().getColor(R.color.dropdown_dark_d
ivider_color));
            }
        }
        layout.findViewById(R.id.dropdown_label_wrapper).setLayoutParams(new
LayoutParams(-2, height, 1.0f));
        DropdownItem item = (DropdownItem) getItem(position);
        TextView labelView = (TextView)
layout.findViewById(R.id.dropdown_label);
        labelView.setText(item.getLabel());
        labelView.setEnabled(item.isEnabled());
        if (item.isGroupHeader()) {
            labelView.setTypeface(null, 1);
        } else {
            labelView.setTypeface(null, 0);
        }
        TextView sublabelView = (TextView)
layout.findViewById(R.id.dropdown_sublabel);
        CharSequence sublabel = item.getSublabel();
        if (TextUtils.isEmpty(sublabel)) {
            sublabelView.setVisibility(8);
        } else {
            sublabelView.setText(sublabel);
            sublabelView.setVisibility(0);
        }
        ImageView iconView = (ImageView)
layout.findViewById(R.id.dropdown_icon);
        if (item.getIconId() == 0) {
            iconView.setVisibility(8);
        } else {
            iconView.setImageResource(item.getIconId());
            iconView.setVisibility(0);
        }
        return layout;
    }

    public boolean areAllItemsEnabled() {
        return this.mAreAllItemsEnabled;
    }

    public boolean isEnabled(int position) {
        if (position < 0 || position >= getCount()) {
            return false;
        }
        DropdownItem item = (DropdownItem) getItem(position);
        if (!item.isEnabled() || item.isGroupHeader()) {
            return false;
        }
        return true;
    }
}
       public class ColorPickerDialog extends AlertDialog implements
OnColorChangedListener {
    private final ColorPickerAdvanced mAdvancedColorPicker;
```

```
    private int mCurrentColor = this.mInitialColor;
    private final View mCurrentColorView;
    private final int mInitialColor;
    private final OnColorChangedListener mListener;
    private final Button mMoreButton;
    private final ColorPickerSimple mSimpleColorPicker;

    public ColorPickerDialog(Context context, OnColorChangedListener listener,
int color, ColorSuggestion[] suggestions) {
        super(context, 0);
        this.mListener = listener;
        this.mInitialColor = color;
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService("layout_inflater");
        View title = inflater.inflate(R.layout.color_picker_dialog_title,
null);
        setCustomTitle(title);
        this.mCurrentColorView = title.findViewById(R.id.selected_color_view);
        ((TextView)
title.findViewById(R.id.title)).setText(R.string.color_picker_dialog_title);
        setButton(-1, context.getString(R.string.color_picker_button_set), new
OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {

ColorPickerDialog.this.tryNotifyColorSet(ColorPickerDialog.this.mCurrentColor)
;
            }
        });
        setButton(-2, context.getString(R.string.color_picker_button_cancel),
new OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {

ColorPickerDialog.this.tryNotifyColorSet(ColorPickerDialog.this.mInitialColor)
;
            }
        });
        setOnCancelListener(new OnCancelListener() {
            public void onCancel(DialogInterface arg0) {

ColorPickerDialog.this.tryNotifyColorSet(ColorPickerDialog.this.mInitialColor)
;
            }
        });
        View content = inflater.inflate(R.layout.color_picker_dialog_content,
null);
        setView(content);
        this.mMoreButton = (Button)
content.findViewById(R.id.more_colors_button);
        this.mMoreButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                ColorPickerDialog.this.showAdvancedView();
            }
        });
        this.mAdvancedColorPicker = (ColorPickerAdvanced)
content.findViewById(R.id.color_picker_advanced);
        this.mAdvancedColorPicker.setVisibility(8);
        this.mSimpleColorPicker = (ColorPickerSimple)
content.findViewById(R.id.color_picker_simple);
        this.mSimpleColorPicker.init(suggestions, this);
        updateCurrentColor(this.mInitialColor);
    }

    public void onColorChanged(int color) {
```

```
            updateCurrentColor(color);
        }

        private void showAdvancedView() {
            findViewById(R.id.more_colors_button_border).setVisibility(8);
            findViewById(R.id.color_picker_simple).setVisibility(8);
            this.mAdvancedColorPicker.setVisibility(0);
            this.mAdvancedColorPicker.setListener(this);
            this.mAdvancedColorPicker.setColor(this.mCurrentColor);
        }

        private void tryNotifyColorSet(int color) {
            if (this.mListener != null) {
                this.mListener.onColorChanged(color);
            }
        }

        private void updateCurrentColor(int color) {
            this.mCurrentColor = color;
            if (this.mCurrentColorView != null) {
                this.mCurrentColorView.setBackgroundColor(color);
            }
        }
    }

    public abstract class TwoFieldDatePickerDialog extends AlertDialog implements
    OnClickListener, OnMonthOrWeekChangedListener {
        private static final String POSITION_IN_YEAR = "position_in_year";
        private static final String YEAR = "year";
        protected final OnValueSetListener mCallBack;
        protected final TwoFieldDatePicker mPicker;

        public interface OnValueSetListener {
            void onValueSet(int i, int i2);
        }
        public TwoFieldDatePickerDialog(Context context, OnValueSetListener
    callBack, int year, int positionInYear, double minValue, double maxValue) {
            this(context, 0, callBack, year, positionInYear, minValue, maxValue);
        }

        public TwoFieldDatePickerDialog(Context context, int theme,
    OnValueSetListener callBack, int year, int positionInYear, double minValue,
    double maxValue) {
            super(context, theme);
            this.mCallBack = callBack;
            setButton(-1, context.getText(R.string.date_picker_dialog_set), this);
            setButton(-2, context.getText(17039360), (OnClickListener) null);
            setIcon(0);
            this.mPicker = createPicker(context, minValue, maxValue);
            setView(this.mPicker);
            this.mPicker.init(year, positionInYear, this);
        }
        protected TwoFieldDatePicker createPicker(Context context, double
    minValue, double maxValue) {
            return null;
        }
        public void onClick(DialogInterface dialog, int which) {
            tryNotifyDateSet();
        }
        protected void tryNotifyDateSet() {
            if (this.mCallBack != null) {
                this.mPicker.clearFocus();
                this.mCallBack.onValueSet(this.mPicker.getYear(),
```

```
this.mPicker.getPositionInYear());
        }
    }

    public void onMonthOrWeekChanged(TwoFieldDatePicker view, int year, int
positionInYear) {
        this.mPicker.init(year, positionInYear, null);
    }

    public void updateDate(int year, int weekOfYear) {
        this.mPicker.updateDate(year, weekOfYear);
    }
}


public class ContentView extends FrameLayout implements
InternalAccessDelegate, SmartClipProvider {
    private static final String TAG = "cr.ContentView";
    protected final ContentViewCore mContentViewCore;

    public static ContentView newInstance(Context context, ContentViewCore
cvc) {
        if (VERSION.SDK_INT < 16) {
            return new ContentView(context, cvc);
        }
        return new JellyBeanContentView(context, cvc);
    }
    protected ContentView(Context context, ContentViewCore cvc) {
        super(context, null, 16842885);
        if (getScrollBarStyle() == 0) {
            setHorizontalScrollBarEnabled(false);
            setVerticalScrollBarEnabled(false);
        }
        setFocusable(true);
        setFocusableInTouchMode(true);
        this.mContentViewCore = cvc;
    }
    @VisibleForTesting
    public void onProvideVirtualStructure(ViewStructure structure) {
        this.mContentViewCore.onProvideVirtualStructure(structure);
    }

    public boolean drawChild(Canvas canvas, View child, long drawingTime) {
        return super.drawChild(canvas, child, drawingTime);
    }
    public void onScrollChanged(int l, int t, int oldl, int oldt) {
        super.onScrollChanged(l, t, oldl, oldt);
    }
    public InputConnection onCreateInputConnection(EditorInfo outAttrs) {
        return this.mContentViewCore.onCreateInputConnection(outAttrs);
    }
    public boolean onCheckIsTextEditor() {
        return this.mContentViewCore.onCheckIsTextEditor();
    }

    protected void onFocusChanged(boolean gainFocus, int direction, Rect
previouslyFocusedRect) {
        try {
            TraceEvent.begin("ContentView.onFocusChanged");
            super.onFocusChanged(gainFocus, direction, previouslyFocusedRect);
            this.mContentViewCore.onFocusChanged(gainFocus);
        } finally {
```

```
            TraceEvent.end("ContentView.onFocusChanged");
        }
    }
        public void setSmartClipResultHandler(final Handler resultHandler) {
        if (resultHandler == null) {
            this.mContentViewCore.setSmartClipDataListener(null);
        } else {
            this.mContentViewCore.setSmartClipDataListener(new
SmartClipDataListener() {
                public void onSmartClipDataExtracted(String text, String html,
Rect clipRect) {

            });
        }
    }
       @JNINamespace("content")
class DateTimeChooserAndroid {
    private final InputDialogContainer mInputDialogContainer;
    private final long mNativeDateTimeChooserAndroid;

    private native void nativeCancelDialog(long j);

    private native void nativeReplaceDateTime(long j, double d);

    private DateTimeChooserAndroid(Context context, long
nativeDateTimeChooserAndroid) {
        this.mNativeDateTimeChooserAndroid = nativeDateTimeChooserAndroid;
        this.mInputDialogContainer = new InputDialogContainer(context, new
InputActionDelegate() {
            public void replaceDateTime(double value) {

DateTimeChooserAndroid.this.nativeReplaceDateTime(DateTimeChooserAndroid.this.
mNativeDateTimeChooserAndroid, value);
            }
            public void cancelDateTimeDialog() {

DateTimeChooserAndroid.this.nativeCancelDialog(DateTimeChooserAndroid.this.mNa
tiveDateTimeChooserAndroid);
            }
        });
    }
    private void showDialog(int dialogType, double dialogValue, double min,
double max, double step, DateTimeSuggestion[] suggestions) {
        this.mInputDialogContainer.showDialog(dialogType, dialogValue, min,
max, step, suggestions);
    }
```

4 Технико-экономическое обоснование

4.1 Цель проекта

Целью данного дипломного проекта является создание мобильного приложения «Справочник C++» для операционной системы Android, являющегося информационным приложением для ознакомления с языком программирования C++, соответствующего всем критерием для Android OS. Разработка приложения производится в среде Android Studio на языке Java. Графические элементы приложения создаются в Adobe Illustrator CC и используется ресурс MockingBot для создания макета будущего приложения.

В этом разделе дипломной работы рассматривается экономическая часть, которая отражает трудовые, временные и финансовые затраты на данный проект.

4.2 Трудовые ресурсы, используемые в работе

В работе над данным проектом приняли участие:

- дизайнер – создание интерфейса приложения;
- программист – программирование и разработка алгоритмов.

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

Таблица 4.1 – Данные о сотрудниках

Должность	Количество	Заработная плата в месяц
Дизайнер	1	60 000
Программист	1	80 000
Итого	2	140 000

4.3 Оборудование, использованное в работе

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

Таблица 4.2 – Перечень оборудования

Название оборудования	Характеристики	Количество	Стоимость за единицу без учета НДС, тенге
Ноутбук Acer TravelMate 7750G-32314G50Mnss	Intel Core i3 3230M, 6 Gb DDR3, 1000 Gb HDD, Radeon HD 7670M	1	92 810

Продолжение таблицы 4.2

Ноутбук Acer v5-552g-10578g50akk	Intel Core i5-4200U, 8 Gb DDR3, 1000 Gb HDD, GeForce GT 740M	1	126 110
Смартфон HTC One mini M4	4 Gb ROM, 512 Mb RAM, MediaTek MT6577T	1	53 810
Итого	3	272730	Итого

4.4 Программное обеспечение, используемое в работе

При разработке приложения используется следующее программное обеспечение:

– Windows 8– операционная система;
– Adobe Illustrator CC – векторный графический редактор;
– MockingBot – инструмент, содержащий все необходимые функции для создания макета;
– Android Studio – среда для разработки;
– Java Development Kit – комплект разработчика приложений.

Программное обеспечение, использованное в работе и соответствующая ему стоимость приведены в таблице 4.3.

Таблица 4.3 – Перечень программного обеспечения

Наименование	Количество копий	Стоимость без учета НДС, тенге
Windows 8	2	Предустановлено, цена ПО входит в стоимость ноутбука
Adobe Illustrator CC	1	37 593 (Годовая подписка)
MockingBot	1	Бесплатно (период 30 дней)
Android Studio	1	Бесплатно
Java Development Kit	1	Бесплатно
Итого		37593

4.5 Сроки реализации проекта

Такой процесс проектирования 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 приведена таблица реализации проекта.

Таблица 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 современных трендов								
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:

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

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:

$$ФОТ = Зосн + Здоп \quad (4.2)$$

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.

Средний дневной заработок каждого работника рассчитывается по формуле:

$$D = \frac{ЗПм}{Др} \quad (4.3)$$

где $З_{Пм}$ – ежемесячный размер заработной платы;
$Д_р$ – количество рабочих дней в месяце (21 день).

Дизайнера:

$$D = \frac{60000}{21} = 2857 \text{ тенге/день}$$

Программиста:

$$D = \frac{80000}{21} = 3809 \text{ тенге/день}$$

Заработная плата за один час работы сотрудника рассчитывается по формуле:

$$Н = D/Чр \quad (4.4)$$

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

Дизайнера:

$$Н = 2857/8 = 357 \text{ тенге/час}$$

Программиста:

$$Н = 3809/8 = 476 \text{ тенге/час}$$

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

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

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_n округляем в большую сторону до целых дней.

$$t_1 = \frac{16}{1 * 8 * 1{,}1} \approx 2 \text{ дня} - \text{Программист, постановка задачи;}$$

$$t_2 = \frac{16}{1 * 8 * 1{,}1} \approx 2 \text{ дня} - \text{Дизайнер, изучение литературы;}$$

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

$$t_4 = \frac{16}{1 * 8 * 1{,}1} \approx 1 \text{ дня} - \text{Программист, выбор среды разработки;}$$

$$t_5 = \frac{16}{1 * 8 * 1{,}1} \approx 3 \text{ дня} - \text{Дизайнер, изучение «туториалов»;}$$

$$t_6 = \frac{16}{1 * 8 * 1{,}1} \approx 6 \text{ дня} - \text{Дизайнер, разработка интерфейса;}$$

$$t_7 = \frac{16}{1 * 8 * 1{,}1} \approx 7 \text{ дня} - \text{Дизайнер, создание графических элементов;}$$

$$t_8 = \frac{16}{1 * 8 * 1{,}1} \approx 12 \text{ дня} - \text{Программист, разработка форм приложения;}$$

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

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

$$t_{11} = \frac{16}{1 * 8 * 1{,}1} \approx 1 \text{ дня} - \text{Программист, оформление и сдача отчета.}$$

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

Т а б л и ц а 4.5 – Сводные результаты расчета затрат на основную заработную плату.

Наименование этапов работ	Исполнитель	Трудоемкость		Длительность цикла, дни	З-плата за час работы, тенге	Сумма зар. платы, тенге
		Нормы-часы	% от общей трудоемкости			
Постановка задачи	Программист	16	5	2	476	7618
Изучение литературы	Дизайнер	16	5	2	357	5714
Изучение литературы	Программист	16	5	2	476	7618
Выбор среды разработки	Программист	8	2,5	1	476	5712
Изучение современных трендов	Дизайнер	24	7,5	3	357	8571
Разработка интерфейса	Дизайнер	48	15	6	357	17142
Создание необходимых графических элементов	Дизайнер	56	17,5	7	357	20000
Разработка форм приложения	Программист	96	30	12	476	45708
Тестирование приложения	Программист	16	5	2	476	7618
Отладка приложения	Программист	16	5	2	476	7618
Оформление и сдача отчета	Программист	8	2,5	1	595	3809
Итого		320	100	40		137128

Таким образом, согласно произведенным расчётам основная заработная плата составляет 137128 тенге.

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

$$Здоп = Зосн * 0,1 \quad (4.6)$$

$$Здоп = 137128 * 0,1 = 13712,8 \text{ тенге}$$

В результате расчетов, согласно формуле 4.2, суммарный фонд оплаты труда составит

$$ФОТ = 137128 + 13712 = 150840 \text{ тенге}$$

4.6.2 Расчет затрат по социальному налогу

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

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

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

$$ПО = ФОТ * 10\% \quad (4.8)$$

$$ПО = 150840 * 0{,}1 = 15084 \text{ тенге}$$

Таким образом социальный налог составит:

$$Сн = (150840 - 15084) * 0{,}11 = 14933 \text{ тенге}$$

4.6.3 Расчет амортизационных отчислений

Амортизационные отчисления рассчитываются по формуле:

$$Ai = \frac{H_A * Спер * N}{100 * 12 * n} \quad (4.9)$$

где H_A– 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 4.9 составят

На оборудование:

$$A1 = \frac{40 * 272730 * 40}{100 * 12 * 21} = 17316 \text{ тенге}$$

На программное обеспечение:

$$A2 = \frac{40 * 37593 * 40}{100 * 12 * 21} = 2387 \text{ тенге}$$

Суммарные затраты на амортизацию рассчитываются по формуле:

$$A = A1 + A2 \quad (4.10)$$

$$А = 17316 + 2387 = 19703 \text{ тенге}$$

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.

$$Э = Зэл.эн.об + Здоп \quad (4.11)$$

где = Зэл.эн.об – затраты на электроэнергию для оборудования;

Здоп – затраты электроэнергии на дополнительные нужды.

Расходы электроэнергии на оборудование рассчитываются по формуле 4.12.

$$Зэл.эн.об = W * T * S * КИСП \quad (4.12)$$

где W – потребляемая оборудованием мощность, кВт;

T – время работы, часы;

S – тариф (1кВт/час = 16,9 тенге);

$К_{ИСП}$ – коэффициент использования ($К_{ИСП}$ = 0,9).

Потребляемая мощность Acer TravelMate 7750G-32314G50Mnss составляет 90 Вт.

Потребляемая мощность Acer v5-552g-10578g50akk 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 Acer v5-552g-10578g50akk.

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

Таким образом общая сумма затрат на электроэнергию для оборудования:

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

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

$$Здоп = Зэл.эн.об * 5\% \quad (4.13)$$

И составляют

$$Здоп = 754,4 * 0,05 = 37,7 \text{ тенге}$$

Суммарные затраты на электроэнергию составляют

$$Э = 754{,}4 + 37{,}7 = 792{,}1 \text{ тенге}$$

4.6.5 Расчет накладных расходов

Накладные расходы рассчитываются в размере 50% от всех затрат.

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

$$Н = (150840{,}8 + 14933{,}23 + 19703 + 792{,}1) * 0{,}5 = 93134{,}56 \text{ тенге}$$

4.6.6 Суммарные затраты на реализацию проекта

Таким образом себестоимость разработки данного приложения, согласно формуле 4.1 составляет

$$С = 150840{,}8 + 14933{,}23 + 19703 + 792{,}1 + 93134{,}5 = 279403{,}69 \text{ тенге}$$

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

Таблица 4.6 – Затраты на разработку приложения

Наименование затрат	Сумма, тенге
ФОТ	150840,8
Социальный налог	14933,23
Амортизационные отчисления	19703
Затраты на электроэнергию	792,1
Накладные расходы	93134,56
Итого	279403,69

Рисунок 4.1 – Затраты на разработку приложения

4.6.7 Цена реализации проекта

Цена проекта складывается из себестоимости и желаемого чистого дохода.

$$Цп = С + П \quad (4.15)$$

Где С – стоимость приложения;

П – чистый доход.

Для определения начальной цены используется желаемый уровень рентабельности. Для данной отрасли он составляет 25%.

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

где Р – рентабельность.

$$Цп = 279403{,}69 * (1 + 25/100) = 349254{,}61 \text{ тенге}$$

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

$$Цр = Цп + НДС \quad (4.17)$$

где НДС – налог на добавленную стоимость.

Согласно Налоговому Кодексу Республики Казахстан НДС составляет 12%, то есть в данном случае равен $НДС = 41910{,}55$ тенге.

В итоге получаем цену реализации проекта равной

$$Цр = 349254{,}61 + 41910{,}55 = 391165{,}16 \text{ тенге}$$

Вывод

Итоговая стоимость разработки мобильного приложения «Справочник С++» для платформы Android OS составила 391165,16 тенге, в которую вложены все возможные затраты при разработки программного продукта.

На создание таких проектов большое внимание уделяется к специалистам, которые будут задействованы в проекте, так как наибольшую долю расходов в стоимости проекта составляют затраты на оплату труда сотрудников порядка в размере 150840,8 тенге, что составляет 54% от всех затрат, участвующих в разработки проекта.

Темой разработки мобильного приложения стало альтернативное создание карманного справочника по языку программирования С++ для студентов с малым опытом программирования или вовсе без него. Приложение поможет пользователям сконцентрироваться на важных аспектах в процессе обучения, свободно пользоваться материалом в любое удобное

время, практически применять свои знания по каждой главе, используя, встроенные примеры коды программ.

Разработка мобильного приложения являет одним из дорогих проектов, требующих больших интеллектуальных и финансовых затрат. Однако, все средства, вложенные в него легко окупаются при условии отличного качества продукта, ориентированного на определенный контингент пользователей и хорошей PR-кампании, позволяя получить прибыль по истечению определенного времени.

5 Безопасность жизнедеятельности

5.1 Анализ рабочего помещения

Производственная практика была проведена в одном из филиалов акционерного общества «республиканский научно-методический центр развития технического и профессионального образования и присвоения квалификации» на втором этаже четырехэтажного здания. Филиал управления состоит из трех отделов, где работают в общей сложности около пятидесяти сотрудников. Площадь, рассматриваемого рабочего помещения составляет около 30 квадратных метров. Мебельная фурнитура, техника и оборудование расположены таким образом, что позволяли свободно двигаться и перемещаться по кабинету и совершать свои непосредственные поставленные обязанности. Перемещение между этажами можно было производить с помощью лифта и двух лестничных площадок, одна из которых проветриваемая. Второй этаж был оборудован двумя санитарными узлами для обоих полов и одной столовой для обеденного перерыва.

Источником света в помещении являются: 3 люминесцентных ламп равномерно размещенные по всему помещению офиса, а также 2 окна размерами 1м х 1м с небольшим затемнение стекла. На окнах установлены специальные шторы - жалюзи для регулировки поступающего солнечного света. Помещение кабинета оснащено датчиками для предупреждения пожаров и дыма, также в офисе расположены приборы для тушения легковоспламеняющихся электрических приборов: пожарный кран и порошковый огнетушитель. В здании есть всегда проветриваемая лестница для случаев с пожарами, которая ограждена от основного здания.

Все электропровода и коммутирующие приборы находятся в подпольном пространстве, а видимые их части закрыты специальными пластиковыми оградителями для исключения какого-либо физического повреждения.

Климатические условия, искусственно создаваемые в рабочем помещении для защиты от неблагоприятных внешних воздействий, определяется действующими на организм человека сочетаниями влажности и температурой окружающих поверхностей, а также скорости движения воздуха.

В рабочей зоне микроклимат индивидуально регулируется для каждой зоны помещения с помощью установленных кондиционеров с программой климат контроля. Зимой офис обогревается с помощью тех же кондиционеров, при помощи программы нагрева воздуха. Благодаря окнам есть возможность естественного проветривания помещения, что благоприятно сказывается на самочувствии и настроении сотрудников.

В целом рабочее место имеет эргономичную мебель и техническое оборудование, что способствует положительной работе в офисе. Стены офиса и мебель должны способствовать созданию благоприятных условий для

зрительного восприятия имеют светлые тона, подобранные под общую стилистику филиала. Каждый сотрудник имеет собственное рабочее место и необходимое оборудование для выполнения своих прямых обязанностей. Здание также имеет собственную столовую и парковку.

Рабочее пространство рассчитано на 7 человек. На полную рабочую смену выходят 5 сотрудников, а остальные 5 человек – на частичную. Режим работы с 9:00 до 18:00.

План помещения представлен на рисунке 5.1.

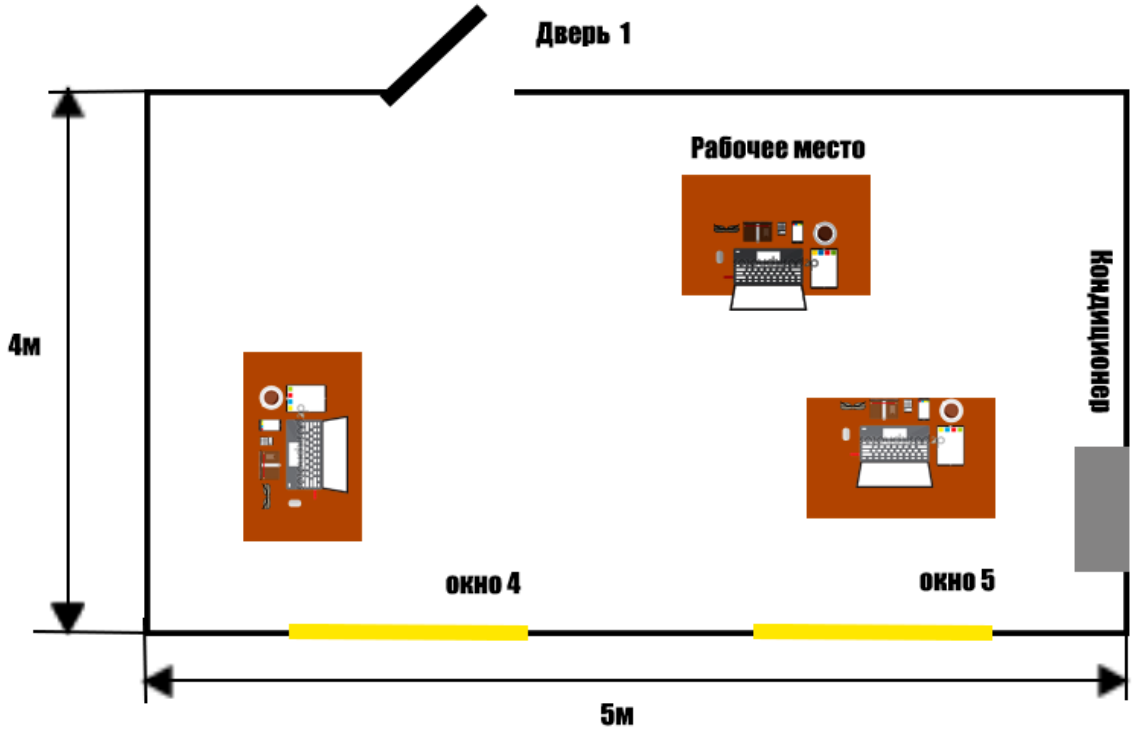


Рисунок 5.1 – План рабочего помещения

В заключение можно сказать, что условия труда и рабочего пространства соблюдены, но одним из главных факторов, которые влияют на работоспособность сотрудника является освещенность, поэтому необходимо заменить искусственное освещение, т.к. лампы потеряли свои первоначальные свойства. Также стоит установить дополнительные приборы для кондиционирования воздуха, так как не хватает мощности для создания комфортного климата всего помещения.

5.1.1 Параметры микроклимата

Принцип нормирования микроклимата – это создание оптимальных условий для теплообмена тела человека с окружающей средой. Параметры микроклимата могут меняться в широких пределах, где необходимостью для условий жизнедеятельности человека является поддержание постоянства температуры тела благодаря терморегуляции, то есть способности организма регулировать отдачу тепла в окружающую среду. Принцип нормирования

микроклимата – создание оптимальных условий для теплообмена человеческого тела с окружающей средой.

Вычислительная техника является источником существенных тепловыделений, что может привести к повышению температуры и снижению относительной влажности в помещении. В помещениях, где установлены компьютеры, должны соблюдаться определенные параметры микроклимата. СН-245-71 в санитарных нормах установлены величины параметров микроклимата, создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (см. табл. 5.1).

Объем помещений, в которых размещены работники вычислительных центров, не должен быть меньше 19,5 $м^3$/человека, с учетом максимального числа одновременно работающих в смену. Нормы подачи свежего воздуха в помещения, где расположены компьютеры, приведены в табл. 5.2.

Таблица 5.1 - Параметры микроклимата для помещений, где установлены компьютеры

Период года	Параметр микроклимата	Величина
Холодный	Температура воздуха в помещении Относительная влажность Скорость движения воздуха	22…24 °С 40…60 % до 0,1 м/с
Теплый	Температура воздуха в помещении Относительная влажность Скорость движения воздуха	23…25 °С 40…60 % 0,1…0,2 м/с

Таблица 5.2 - Нормы подачи свежего воздуха в помещения, где расположены компьютеры

Характеристика помещения	Объемный расход подаваемого в помещение свежего воздуха, $м^3$ /на одного человека в час
Объем до 20 $м^3$ на человека 20…40 $м^3$ на человека Более 40 $м^3$ на человека	Не менее 30 Не менее 20 Естественная вентиляция

Для обеспечения комфортных условий используются как организационные методы (рациональная организация проведения работ в зависимости от времени года и суток, чередование труда и отдыха), так и технические средства (вентиляция, кондиционирование воздуха, отопительная система).

5.1.2 Окраска и коэффициенты отражения

Обдуманный дизайн здания представляется эффективным орудием увеличения производительности работы. В любом компании рядом интенсивном участии самих работников формировались подобные обстоятельства работы, рядом каковых процедура давал б трудящемуся эстетическое удовлетворенность, а итоги работы делались более действенными. Установлено, что же скверные обстоятельства работы понижают трудоспособность, приводят к болезням и травматизму, а эффективность работы существенно уменьшается в связи малой освещенности, неразумной окраски внутреннего убранства и оснащения. Создание оптимальных условий труда тесно связано с современным уровнем культуры производства, т. е. со степенью совершенства технологии, техники, организации производства и труда.

Окраска помещений и мебели должна способствовать созданию благоприятных условий для зрительного восприятия, хорошего настроения.

Источники света, такие как светильники и окна, которые дают отражение от поверхности экрана, значительно ухудшают точность знаков и влекут за собой помехи физиологического характера, которые могут выразиться в значительном напряжении, особенно при продолжительной работе. Отражение, включая отражения от вторичных источников света, должно быть сведено к минимуму. Для защиты от избыточной яркости окон могут быть применены шторы и экраны.

В зависимости от ориентации окон рекомендуется следующая окраска стен и пола в таблице 5.3

Таблица 5.3 - Нормы рекомендуемой окраски помещения

Ориентация окон	Окрас помещения
окна ориентированы на юг	стены зеленовато-голубого или светло-голубого цвета; пол - зеленый;
окна ориентированы на север	стены светло-оранжевого или оранжево-желтого цвета; пол - красновато-оранжевый
окна ориентированы на восток	– стены желто-зеленого цвета; – пол зеленый или красновато-оранжевый;
окна ориентированы на запад	стены желто-зеленого или голубовато-зеленого цвета; пол зеленый или красновато-оранжевый.

В помещениях, где находится компьютер, необходимо обеспечить следующие величины коэффициента отражения: для стен: 40-50%, для пола: около 30%, для потолка: 60-70%. Для других поверхностей рабочей мебели: 30-40%.

5.1.3 Рекомендации по снижению шума в офисном помещении

С физиологической точки зрения, шум всегда рассматривается как посторонний звук, мешающий рабочему процессу человека и очень негативно сказывающиеся на его самочувствие, настроение, что тем самым снижает его производительность труда.

При прохождении практики серверная комната находилась в углу второго этажа и стены были оборудованы дополнительной изоляцией, также комната заперта железной дверью. Здание находится в тихом районе города, что благоприятно сказывается на общую атмосферу. Однако, основным источником шума является, непосредственно сам рабочий процесс сотрудников

Общие требования безопасности, эквивалентный уровень звука не должен превышать 50 дБА. В качестве мер по снижению шума можно предложить следующее:

- сделать рациональную планировку помещения;
- покрыть облицовку стен и потолков звукопоглощающим покрытием;
- оградить рабочие зоны офиса дополнительными стенами.

5.2 Расчет системы автоматического пожаротушения

Автоматическое пожаротушение предполагает перед собою присутствие денег предотвращения пожара, приборов выявления пожара, сигнализации и денег пожаротушения. Комплекс упомянутых денег, определенных безусловно ведь специалистами–пожарниками, дает возможность полагать этот совокупность никак не по-другому равно как концепцией механического спецпожаротушение. При этом роль человека объединяется к проведению постоянных исследовательских трудов и смене денег пожаротушения согласно истечению сроков годности.

Разрабатываемая система автоматического пожаротушения представляется аппаратной частью. В этом зале находится электрическое оснащение, бег и положение оператора. Угроза возгорания истекает только лишь с попадания стороннего усилия в платы, либо ведь проход токов высочайших номиналов согласно проводникам, что же спровоцирует загорание обособленности. Принимая во внимание этот период, что же оснащение крайне дорогое и кризисное к непосредственному попаданию вода и соли, в таком случае механическая концепция пожаротушения обязана употреблять надлежащие огнегасящие элемента и обладать большой степень прочности.

Использование организации механического пожаротушения (САП) базируется в применении конструкции газового пожаротушения, что же даст

возможность уменьшить вещественные утраты. САП газового пожаротушения применяет последующие элемента:

– двуокись углерода;
– хладон 114В(2) / тетрофтордибромэтан;
– хладон 13В(1) / бромтрифторметан;
– комбинированный углекислотно–хладоновый состав;
– азот;
– аргон.

САП должна отвечать следующим требованиям:

– дистанционное и местное включение;
– выполнять функции пожарной сигнализации;
– соответствовать требованиям помещения, где установлена САП.

Тип САП и огнетушащие вещества выбираются с учётом пожарной опасности и физико–химических свойств, а также в зависимости от принадлежности помещения по СниП.

В этом случае следует подобрать надлежащую САП, подходящей с целью использования в обстановках присутствия правил, пребывающих перед усилием и в замкнутом помещении. Наиболее часто используемые газовые САП в нашем случае, это:

– установки объёмного пожаротушения;
– установки локального пожаротушения по площади;
– установки локального пожаротушения по объёму.

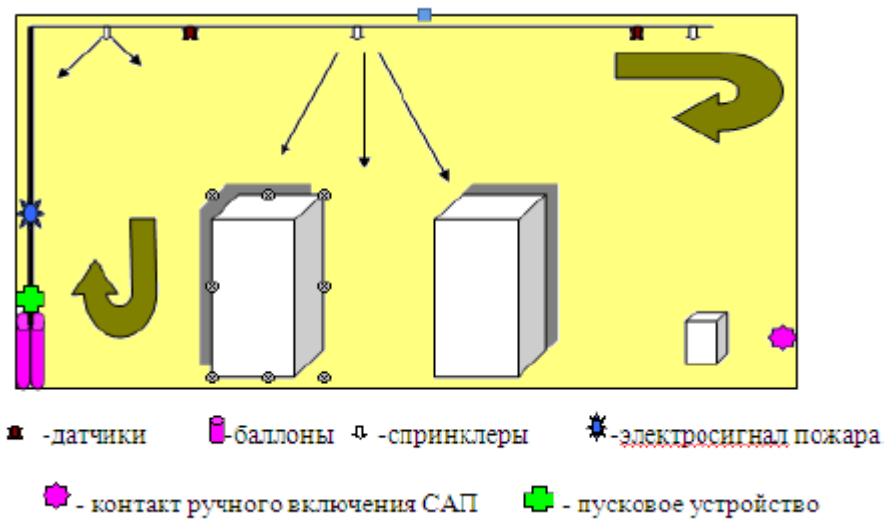


Рисунок 5.1 – Схема системы автоматического пожаротушения

Расчетная масса комбинированного углекислотно–хладонового состава md кг, для объемного пожаротушения определяется по формуле

$$md = k6 \cdot gn \cdot V \qquad (5.1)$$

где k_6 – коэффициент компенсации не учитываемых потерь состава, k_6=1,13;

g_n – нормативная массовая концентрация состава – 0,27 кг/м,

V – объём защищаемого помещения (м3).

$$md = 1{,}13 \cdot 0{,}27 \cdot 122{,}5 = 37 \text{ кг.}$$

Расчетное число баллонов ξ_2 определяется из расчета вместимости в 40– литровый баллон 25 кг углекислотно–хладонового состава.

Количество баллонов –2 (25кг).

Внутренний диаметр магистрального трубопровода d_i, мм, определяется по формуле:

$$di = d1 \cdot \varepsilon 2 \qquad (5.2)$$

где d_i – диаметр сифонной трубки баллона, мм;

ξ_2– число одновременно разрежаемых баллонов.

$$di = 10 \cdot \sqrt{2} = 14{,}2 \text{ мм.}$$

Эквивалентная длина магистрального трубопровода l_2, м, определяется по формуле

$$l2 = k7 \cdot l \qquad (5.3)$$

где k_7 – коэффициент увеличения длины трубопровода для компенсации

l – длина трубопровода по проекту, l=60 м.

$$l_2 = 1{,}2 \cdot 60 = 72 \text{ м}$$

Площадь сечения выходного отверстия оросителя определим по формуле, м3:

$$A3 = S/\varepsilon 1 \qquad (5.4)$$

где S – площадь сечения магистрального трубопровода, мм2;

ξ_1 – число оросителей;

$$A = 122{,}5/6 = 20{,}4 \text{ мм}^2.$$

Расход углекислотно–хладонового состава Q, кг/с, зависит от эквивалентной длины и диаметра трубопровода, Q=0,8.

Расчетное время подачи углекислотно–хладонового состава t, мин, опредется по формуле

$$t = m * \frac{d}{60} * Q \quad (5.5)$$

$$t = 37/60 \cdot 0{,}8 = 0{,}77 \text{ мин.}$$

Масса основного запаса углекислотно–хладонового состав m, кг, определяется по формуле

$$m = 1{,}1 \cdot md \cdot (1 + k8 / k6) \quad (5.6)$$

где k_8 – коэффициент, учитывающий остаток углекислотно – хладонового состава в баллонах и трубопроводах, k_8=0,2.

$$m = 1{,}1 \cdot 37 \cdot \left(1 + \frac{0{,}2}{1{,}13}\right) = 48 \text{ кг}$$

Отталкиваясь с расчетов в размещение с оборудованием, с целью соблюдения необходимых характеристик и конструкции организации механического пожаротушения следует определить 2 таких баллона с углекислотно–хладоновым формулой.

5.3 Расчёт искусственного освещения

Для помещений, в которых предусматриваются общее равномерное освещение горизонтальных поверхностей, освещение рассчитывают методом коэффициента использования светового потока. Длина комнаты *A*=4 м, ширина комнаты *B*=5 м, высота *H*=4 м. Высота рабочей поверхности над уровнем пола $h_р$=0,8 – 1 м. Расстояние от светильника до перекрытия $h_с$=0 – 1,5 м.

Данный метод заключается в определении значения коэффициента η , равного отношению светового потока падающего на расчетную поверхность, к полному потоку осветительного прибора.

Значение коэффициента η находится из таблиц, связывающих геометрические параметры помещений (индекс помещений *i*) с их оптическими характеристиками (коэффициентами отражения потолка $\rho_{пот}$, стен $\rho_{ст}$ и пола $\rho_п$.

Индекс помещения *i* определяется по формуле 5.9.

$$i = \frac{A*B}{h*(A+B)} \quad (5.9)$$

где A – длина помещения,

B – ширина помещения,

$h = H – h_с - h_р$ – расчетная высота.

Определяем значение h:

$$h = 4 - 0 - 0{,}8 = 3{,}2\,(\text{м})$$

При найденном значении расчетной высоты определяем индекс помещения:

$$i = \frac{4 * 5}{3{,}2 * (4 + 5)} = 0{,}69$$

Значения коэффициентов отражения примем следующими:

$\rho_{пот} = 70\ (\ \%\ )$ $\rho_{ст} = 50\ (\ \%\ )$ $\rho_{п} = 30\ (\ \%\ )$

Для найденного индекса помещения и выбранных значений коэффициентов отражения по таблице «Значения коэффициента использования светового потока» определяем коэффициент η , который равен:

$$\eta = 28\ (\ \%\ ).$$

Для освещения помещений буду использовать люминесцентные лампы ЛБ мощностью 40 Вт и номинальным световым потоком 3120 лм. В качестве светильников буду использовать светильники типа ЛОУ –2х40–1001. В каждый светильник устанавливается по две лампы.

С учетом вышесказанного можно определить количество светильников по формуле 5.10.

$$N = \frac{E * k_З * S * z}{n * Ф * \eta} \qquad (5.10)$$

Где Е – освещенность, для IV (б) разряда работ нормируемая освещенность – 200 лк,$k_з$ – коэффициент запаса, S – площадь помещения, z=1,1÷1,2 – коэффициент неравномерности освещения, n – число ламп в светильнике, $Ф_л$ – световой поток одной лампы, η – коэффициент использования.

Для помещений общественных зданий (рабочих помещений) при искусственном освещении газоразрядными лампами, исходя их СНиП РК 2.04–05–2002, коэффициент запаса $К_з = 1{,}2$.

Площадь рассматриваемого помещения равна: $S = A \cdot B = 4 \cdot 5 = 20$ [м^2].

Число светильников равно:

$$N = \frac{200 * 1{,}2 * 20 * 1{,}1}{2 * 3120 * 0{,}28} = 3(\text{шт})$$

Рассчитаю расстояние между светильниками по формуле:

$$L\text{A,B} = \lambda \cdot h\text{p}$$

Приму $\lambda = 1{,}2$, тогда:

$$L\text{a} = 1{,}2 \cdot 3{,}2 = 3{,}84\ \text{м}$$

Далее найду расстояние до стены по следующей формуле:

$$l_{\text{A,B}} = \frac{0{,}3}{0{,}5} * L_{A,B} \qquad (5.11)$$

$$l_{\text{a}} = 0{,}5 \cdot 3{,}84 = 1{,}92\text{м}$$

Расстояние L_A в моём случае отсутствует, так как имеется всего 3 светильника. Расстояние $l_в$ приму равным 1,4 м, так как длина светильника составляет 1,2 м, ширина помещения равна 5 м. Схема размещения светильников приведена на рисунке 5.3.

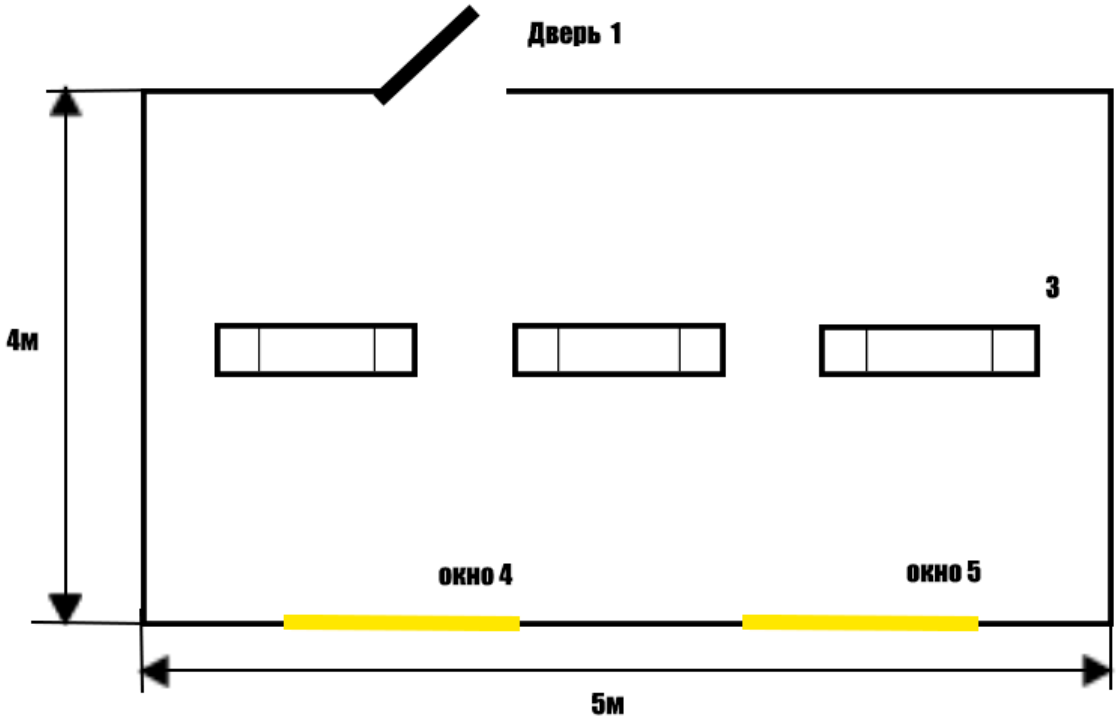


Рисунок 5.3 – Схема размещения светильников
1– дверь; 2 – светильники; 3 – стена; 4 – окно;

Здания предприятий, которые расположенные в индустриальных зонах, допускается сооружать одноэтажными со сплошной застройкой, а здания предприятий, расположенных в пределах городской селитебной территории многоэтажными. Объем производственных помещений и площадь на одного работающего должны составлять, соответственно, не менее 15 $м^3$ и не менее 4,5 $м^2$.

Высота потолков выбирается в зависимости от характера технологического процесса. Для удаления избыточной теплоты, влаги и газов она должна быть не менее 3,0 м.

Заключение

В ходе выполнения данного дипломного проекта были усвоены основные принципы разработки приложений, а также предъявляемых к ним требований и стандартов. Изучение таких принципов, как проектирование и построение работы приложения для операционной системы Android стало наибольшей степени актуальной темой в современных реалиях, диктующего мобильного рынка.

На основе проведенного анализа магазина приложений Google Play было разработано мобильное приложение для изучения курса языка программирования C++ для соответствующей платформы. Приложение проектировалась на Android Studio, так как это наиболее удовлетворяющая по функциональным возможностям среда проектирования, которая имеет высокое качество безопасности и надежность в использовании большим числом пользователей. В ходе разработки внедрены уникальные функции, которые не были доступны в стандартных приложениях, а также в аналогичных проектируемых продуктах сторонних разработчиков. Данное приложение отвечает всем стандартам и требованиям платформ держателя. В ней предоставляется более удобный способ представление материала, что позволяет легко использовать интерфейс под нужны пользователя. Самым главным в реализации мобильного приложения стало получение опыта и участие в полном жизненном цикле в разработке ПО.

В экономической части дипломной работы была рассчитана общая стоимость продукта, в основе суммирования различных видов затрат, которые возможны при реализации. Данный расчет соответствует любому виду производства 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 приложения и позволяет получить хороший объем прибыли.

В разделе безопасность жизнедеятельности были рассчитаны необходимые условия для организации, соответствующего всем установленным нормам рабочего места для сотрудников, занятых в реализации такого проекта. Рассчитано необходимое искусственное освещение в кабинете сотрудника, требующееся для комфортной работы без нанесения ему вреда здоровью и произведен расчет системы автоматического пожаротушения.

Список литературы и источников

1. Рето М., Android 2. Программирование приложений для планшетных компьютеров и смартфонов. – М. Эксмо, 2011.
2. Сайт https://habrahabr.ru/company/redmadrobot/blog/252773/
3. Сайт https://developer.android.com/design/material/index.html?hl=ru
4. Сайт https://developer.android.com/training/material/index.html
5. Сайт http://startandroid.ru/ru/
6. 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.
7. Абдимуратов Ж. С., Маманбаева С. Е. Расчет производственного помещения. – Алматы: АУЭС, 2009.
8. Коробко, В.И. Охрана труда: Учебное пособие для студентов вузов/В.И. Коробко. – М.: ЮНИТИ-ДАНА, 2013. – 239 с.
9. СНиП РК 2.04-05-2002 «Естественное и искусственное освещение. Общие требования. – Астана, 2002.
10. Баклашов Н.И., Китаева Н.Ж., Терехов Б.Д. Охрана труда на предприятиях связи и охрана окружающей

Приложение А

Листинг программы

```
package io.cordova.hellocordova;

import android.os.Bundle;
import org.apache.cordova.CordovaActivity;

public class MainActivity extends CordovaActivity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        loadUrl(this.launchUrl);
    }
}

package io.cordova.hellocordova;

public final class BuildConfig {
    public static final String APPLICATION_ID = "io.cordova.hellocordova";
    public static final String BUILD_TYPE = "debug";
    public static final boolean DEBUG = Boolean.parseBoolean("true");
    public static final String FLAVOR = "armv7";
    public static final int VERSION_CODE = 12;
    public static final String VERSION_NAME = "0.0.1";
}


package nl.xservices.plugins;
import …
public class SocialSharing extends CordovaPlugin {
    private static final String ACTION_AVAILABLE_EVENT = "available";
    private static final String ACTION_CAN_SHARE_VIA = "canShareVia";
    private static final String ACTION_CAN_SHARE_VIA_EMAIL =
"canShareViaEmail";
    private static final String ACTION_SHARE_EVENT = "share";
    private static final String ACTION_SHARE_VIA = "shareVia";
    private static final String ACTION_SHARE_VIA_EMAIL_EVENT =
"shareViaEmail";
    private static final String ACTION_SHARE_VIA_FACEBOOK_EVENT =
"shareViaFacebook";

    private abstract class SocialSharingRunnable implements Runnable {
        public CallbackContext callbackContext;

        SocialSharingRunnable(CallbackContext cb) {
            this.callbackContext = cb;
        }
    }

    public boolean execute(String action, JSONArray args, CallbackContext
callbackContext) throws JSONException {
        this._callbackContext = callbackContext;
        this.pasteMessage = null;
        if (ACTION_AVAILABLE_EVENT.equals(action)) {
            callbackContext.sendPluginResult(new PluginResult(Status.OK));
            return true;
        } else if (ACTION_SHARE_EVENT.equals(action)) {
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
```

```
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), null, false);
        } else if (ACTION_SHARE_VIA_TWITTER_EVENT.equals(action)) {
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), "twitter", false);
        } else if (ACTION_SHARE_VIA_FACEBOOK_EVENT.equals(action)) {
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), "com.facebook.katana", false);
        } else if
(ACTION_SHARE_VIA_FACEBOOK_WITH_PASTEMESSAGEHINT.equals(action)) {
            this.pasteMessage = args.getString(4);
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), "com.facebook.katana", false);
        } else if (ACTION_SHARE_VIA_WHATSAPP_EVENT.equals(action)) {
            if (notEmpty(args.getString(4))) {
                return shareViaWhatsAppDirectly(callbackContext,
args.getString(0), args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), args.getString(4));
            }
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), "whatsapp", false);
        } else if (ACTION_SHARE_VIA_INSTAGRAM_EVENT.equals(action)) {
            if (notEmpty(args.getString(0))) {
                copyHintToClipboard(args.getString(0), "Instagram paste
message");
            }
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), "instagram", false);
        } else if (ACTION_CAN_SHARE_VIA.equals(action)) {
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), args.getString(4), true);
        } else if (ACTION_CAN_SHARE_VIA_EMAIL.equals(action)) {
            if (isEmailAvailable()) {
                callbackContext.sendPluginResult(new PluginResult(Status.OK));
                return true;
            }
            callbackContext.sendPluginResult(new PluginResult(Status.ERROR,
"not available"));
            return false;
        } else if (ACTION_SHARE_VIA.equals(action)) {
            return doSendIntent(callbackContext, args.getString(0),
args.getString(ACTIVITY_CODE_SEND),
args.getJSONArray(ACTIVITY_CODE_SENDVIAEMAIL),
args.getString(ACTIVITY_CODE_SENDVIAWHATSAPP), args.getString(4), false);
        } else if (ACTION_SHARE_VIA_SMS_EVENT.equals(action)) {
            return invokeSMSIntent(callbackContext, args.getJSONObject(0),
}
            callbackContext.error("socialSharing." + action + " is not a
supported function. Did you mean '" + ACTION_SHARE_EVENT + "'?");
            return false;
```

```
        }
    }

    private boolean isEmailAvailable() {
        if
(this.cordova.getActivity().getPackageManager().queryIntentActivities(new
Intent("android.intent.action.SENDTO", Uri.fromParts("mailto",
"someone@domain.com", null)), 0).size() > 0) {
            return true;
        }
        return false;
    }

        });
        return true;
    }



public class NavigationParams {
    public final boolean hasUserGesture;
    public final boolean hasUserGestureCarryover;
    public final boolean isExternalProtocol;
    public final boolean isMainFrame;
    public final boolean isPost;
    public final boolean isRedirect;
    public final int pageTransitionType;
    public final String referrer;
    public final String url;

    public NavigationParams(String url, String referrer, boolean isPost,
boolean hasUserGesture, int pageTransitionType, boolean isRedirect, boolean
isExternalProtocol, boolean isMainFrame, boolean hasUserGestureCarryover) {
        this.url = url;
        if (TextUtils.isEmpty(referrer)) {
            referrer = null;
        }
        this.referrer = referrer;
        this.isPost = isPost;
        this.hasUserGesture = hasUserGesture;
        this.pageTransitionType = pageTransitionType;
        this.isRedirect = isRedirect;
        this.isExternalProtocol = isExternalProtocol;
        this.isMainFrame = isMainFrame;
        this.hasUserGestureCarryover = hasUserGestureCarryover;
    }

    @CalledByNative
    public static NavigationParams create(String url, String referrer, boolean
isPost, boolean hasUserGesture, int pageTransitionType, boolean isRedirect,
boolean isExternalProtocol, boolean isMainFrame, boolean
hasUserGestureCarryover) {
        return new NavigationParams(url, referrer, isPost, hasUserGesture,
pageTransitionType, isRedirect, isExternalProtocol, isMainFrame,
hasUserGestureCarryover);
    }
}



package org.chromium.content.app;
```

```
        import android.content.Context;
        import org.chromium.base.annotations.JNINamespace;

@JNINamespace("content")
public class ContentMain {
    private static native void nativeInitApplicationContext(Context context);

    private static native int nativeStart();

    public static void initApplicationContext(Context context) {
        nativeInitApplicationContext(context);
    }

    public static int start() {
        return nativeStart();
    }
}
```

```
public class DropdownAdapter extends ArrayAdapter<DropdownItem> {
    private boolean mAreAllItemsEnabled = checkAreAllItemsEnabled();
    private Context mContext;
    private Set<Integer> mSeparators;

    public DropdownAdapter(Context context, List<DropdownItem> items,
Set<Integer> separators) {
        super(context, R.layout.dropdown_item, items);
        this.mSeparators = separators;
        this.mContext = context;
    }

    public DropdownAdapter(Context context, DropdownItem[] items, Set<Integer>
separators) {
        super(context, R.layout.dropdown_item, items);
        this.mSeparators = separators;
        this.mContext = context;
    }

    private boolean checkAreAllItemsEnabled() {
        for (int i = 0; i < getCount(); i++) {
            DropdownItem item = (DropdownItem) getItem(i);
            if (item.isEnabled() && !item.isGroupHeader()) {
                return false;
            }
        }
        return true;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        View layout = convertView;
        if (convertView == null) {
            layout = ((LayoutInflater)
this.mContext.getSystemService("layout_inflater")).inflate(R.layout.dropdown_i
tem, null);
            ApiCompatibilityUtils.setBackgroundForView(layout, new
DropdownDividerDrawable());
        }
        DropdownDividerDrawable divider = (DropdownDividerDrawable)
layout.getBackground();
```

```
        int height =
this.mContext.getResources().getDimensionPixelSize(R.dimen.dropdown_item_heigh
t);
        if (position == 0) {
            divider.setColor(0);
        } else {
            int dividerHeight =
this.mContext.getResources().getDimensionPixelSize(R.dimen.dropdown_item_divid
er_height);
            height += dividerHeight;
            divider.setHeight(dividerHeight);
            if (this.mSeparators == null ||
!this.mSeparators.contains(Integer.valueOf(position))) {

divider.setColor(this.mContext.getResources().getColor(R.color.dropdown_divide
r_color));
            } else {

divider.setColor(this.mContext.getResources().getColor(R.color.dropdown_dark_d
ivider_color));
            }
        }
        layout.findViewById(R.id.dropdown_label_wrapper).setLayoutParams(new
LayoutParams(-2, height, 1.0f));
        DropdownItem item = (DropdownItem) getItem(position);
        TextView labelView = (TextView)
layout.findViewById(R.id.dropdown_label);
        labelView.setText(item.getLabel());
        labelView.setEnabled(item.isEnabled());
        if (item.isGroupHeader()) {
            labelView.setTypeface(null, 1);
        } else {
            labelView.setTypeface(null, 0);
        }
        TextView sublabelView = (TextView)
layout.findViewById(R.id.dropdown_sublabel);
        CharSequence sublabel = item.getSublabel();
        if (TextUtils.isEmpty(sublabel)) {
            sublabelView.setVisibility(8);
        } else {
            sublabelView.setText(sublabel);
            sublabelView.setVisibility(0);
        }
        ImageView iconView = (ImageView)
layout.findViewById(R.id.dropdown_icon);
        if (item.getIconId() == 0) {
            iconView.setVisibility(8);
        } else {
            iconView.setImageResource(item.getIconId());
            iconView.setVisibility(0);
        }
        return layout;
    }

    public boolean areAllItemsEnabled() {
        return this.mAreAllItemsEnabled;
    }

    public boolean isEnabled(int position) {
        if (position < 0 || position >= getCount()) {
            return false;
        }
        DropdownItem item = (DropdownItem) getItem(position);
```

```
        if (!item.isEnabled() || item.isGroupHeader()) {
            return false;
        }
        return true;
    }
}
```

```
public class ColorPickerDialog extends AlertDialog implements
OnColorChangedListener {
    private final ColorPickerAdvanced mAdvancedColorPicker;
    private int mCurrentColor = this.mInitialColor;
    private final View mCurrentColorView;
    private final int mInitialColor;
    private final OnColorChangedListener mListener;
    private final Button mMoreButton;
    private final ColorPickerSimple mSimpleColorPicker;

    public ColorPickerDialog(Context context, OnColorChangedListener listener,
int color, ColorSuggestion[] suggestions) {
        super(context, 0);
        this.mListener = listener;
        this.mInitialColor = color;
        LayoutInflater inflater = (LayoutInflater)
context.getSystemService("layout_inflater");
        View title = inflater.inflate(R.layout.color_picker_dialog_title,
null);
        setCustomTitle(title);
        this.mCurrentColorView = title.findViewById(R.id.selected_color_view);
        ((TextView)
title.findViewById(R.id.title)).setText(R.string.color_picker_dialog_title);
        setButton(-1, context.getString(R.string.color_picker_button_set), new
OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {

ColorPickerDialog.this.tryNotifyColorSet(ColorPickerDialog.this.mCurrentColor)
;
            }
        });
        setButton(-2, context.getString(R.string.color_picker_button_cancel),
new OnClickListener() {
            public void onClick(DialogInterface dialogInterface, int i) {

ColorPickerDialog.this.tryNotifyColorSet(ColorPickerDialog.this.mInitialColor)
;
            }
        });
        setOnCancelListener(new OnCancelListener() {
            public void onCancel(DialogInterface arg0) {

ColorPickerDialog.this.tryNotifyColorSet(ColorPickerDialog.this.mInitialColor)
;
            }
        });
        View content = inflater.inflate(R.layout.color_picker_dialog_content,
null);
        setView(content);
        this.mMoreButton = (Button)
content.findViewById(R.id.more_colors_button);
```

```
        this.mMoreButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                ColorPickerDialog.this.showAdvancedView();
            }
        });
        this.mAdvancedColorPicker = (ColorPickerAdvanced)
content.findViewById(R.id.color_picker_advanced);
        this.mAdvancedColorPicker.setVisibility(8);
        this.mSimpleColorPicker = (ColorPickerSimple)
content.findViewById(R.id.color_picker_simple);
        this.mSimpleColorPicker.init(suggestions, this);
        updateCurrentColor(this.mInitialColor);
    }

    public void onColorChanged(int color) {
        updateCurrentColor(color);
    }

    private void showAdvancedView() {
        findViewById(R.id.more_colors_button_border).setVisibility(8);
        findViewById(R.id.color_picker_simple).setVisibility(8);
        this.mAdvancedColorPicker.setVisibility(0);
        this.mAdvancedColorPicker.setListener(this);
        this.mAdvancedColorPicker.setColor(this.mCurrentColor);
    }

    private void tryNotifyColorSet(int color) {
        if (this.mListener != null) {
            this.mListener.onColorChanged(color);
        }
    }

    private void updateCurrentColor(int color) {
        this.mCurrentColor = color;
        if (this.mCurrentColorView != null) {
            this.mCurrentColorView.setBackgroundColor(color);
        }
    }
}

public abstract class TwoFieldDatePickerDialog extends AlertDialog implements
OnClickListener, OnMonthOrWeekChangedListener {
    private static final String POSITION_IN_YEAR = "position_in_year";
    private static final String YEAR = "year";
    protected final OnValueSetListener mCallBack;
    protected final TwoFieldDatePicker mPicker;

    public interface OnValueSetListener {
        void onValueSet(int i, int i2);
    }

    public TwoFieldDatePickerDialog(Context context, OnValueSetListener
callBack, int year, int positionInYear, double minValue, double maxValue) {
        this(context, 0, callBack, year, positionInYear, minValue, maxValue);
    }

    public TwoFieldDatePickerDialog(Context context, int theme,
OnValueSetListener callBack, int year, int positionInYear, double minValue,
double maxValue) {
        super(context, theme);
        this.mCallBack = callBack;
        setButton(-1, context.getText(R.string.date_picker_dialog_set), this);
        setButton(-2, context.getText(17039360), (OnClickListener) null);
```

```
        setIcon(0);
        this.mPicker = createPicker(context, minValue, maxValue);
        setView(this.mPicker);
        this.mPicker.init(year, positionInYear, this);
    }

    protected TwoFieldDatePicker createPicker(Context context, double
minValue, double maxValue) {
        return null;
    }

    public void onClick(DialogInterface dialog, int which) {
        tryNotifyDateSet();
    }

    protected void tryNotifyDateSet() {
        if (this.mCallBack != null) {
            this.mPicker.clearFocus();
            this.mCallBack.onValueSet(this.mPicker.getYear(),
this.mPicker.getPositionInYear());
        }
    }

    public void onMonthOrWeekChanged(TwoFieldDatePicker view, int year, int
positionInYear) {
        this.mPicker.init(year, positionInYear, null);
    }

    public void updateDate(int year, int weekOfYear) {
        this.mPicker.updateDate(year, weekOfYear);
    }
}
public class ContentView extends FrameLayout implements
InternalAccessDelegate, SmartClipProvider {
    private static final String TAG = "cr.ContentView";
    protected final ContentViewCore mContentViewCore;

    public static ContentView newInstance(Context context, ContentViewCore
cvc) {
        if (VERSION.SDK_INT < 16) {
            return new ContentView(context, cvc);
        }
        return new JellyBeanContentView(context, cvc);
    }
    protected ContentView(Context context, ContentViewCore cvc) {
        super(context, null, 16842885);
        if (getScrollBarStyle() == 0) {
            setHorizontalScrollBarEnabled(false);
            setVerticalScrollBarEnabled(false);
        }
        setFocusable(true);
        setFocusableInTouchMode(true);
        this.mContentViewCore = cvc;
    }

    @VisibleForTesting
    public void onProvideVirtualStructure(ViewStructure structure) {
        this.mContentViewCore.onProvideVirtualStructure(structure);
    }

    public boolean drawChild(Canvas canvas, View child, long drawingTime) {
        return super.drawChild(canvas, child, drawingTime);
    }
```

```
    public void onScrollChanged(int l, int t, int oldl, int oldt) {
        super.onScrollChanged(l, t, oldl, oldt);
    }

    protected void onSizeChanged(int w, int h, int ow, int oh) {
        try {
            TraceEvent.begin("ContentView.onSizeChanged");
            super.onSizeChanged(w, h, ow, oh);
            this.mContentViewCore.onSizeChanged(w, h, ow, oh);
        } finally {
            TraceEvent.end("ContentView.onSizeChanged");
        }
    }

    public InputConnection onCreateInputConnection(EditorInfo outAttrs) {
        return this.mContentViewCore.onCreateInputConnection(outAttrs);
    }

    public boolean onCheckIsTextEditor() {
        return this.mContentViewCore.onCheckIsTextEditor();
    }

    protected void onFocusChanged(boolean gainFocus, int direction, Rect
previouslyFocusedRect) {
        try {
            TraceEvent.begin("ContentView.onFocusChanged");
            super.onFocusChanged(gainFocus, direction, previouslyFocusedRect);
            this.mContentViewCore.onFocusChanged(gainFocus);
        } finally {
            TraceEvent.end("ContentView.onFocusChanged");
        }
    }
    public void onWindowFocusChanged(boolean hasWindowFocus) {
        super.onWindowFocusChanged(hasWindowFocus);
        this.mContentViewCore.onWindowFocusChanged(hasWindowFocus);
    }

    public boolean onKeyUp(int keyCode, KeyEvent event) {
        return this.mContentViewCore.onKeyUp(keyCode, event);
    }

    public boolean dispatchKeyEventPreIme(KeyEvent event) {
        return this.mContentViewCore.dispatchKeyEventPreIme(event);
    }

    public boolean dispatchKeyEvent(KeyEvent event) {
        if (isFocused()) {
            return this.mContentViewCore.dispatchKeyEvent(event);
        }
        return super.dispatchKeyEvent(event);
    }

    public boolean onTouchEvent(MotionEvent event) {
        return this.mContentViewCore.onTouchEvent(event);
    }

    public boolean onHoverEvent(MotionEvent event) {
        boolean consumed = this.mContentViewCore.onHoverEvent(event);
        if (!this.mContentViewCore.isTouchExplorationEnabled()) {
            super.onHoverEvent(event);
        }
        return consumed;
```

```
    }

    public boolean onGenericMotionEvent(MotionEvent event) {
        return this.mContentViewCore.onGenericMotionEvent(event);
    }

    public boolean performLongClick() {
        return false;
    }

    protected void onConfigurationChanged(Configuration newConfig) {
        this.mContentViewCore.onConfigurationChanged(newConfig);
    }

    public void scrollBy(int x, int y) {
        this.mContentViewCore.scrollBy((float) x, (float) y, false);
    }

    public void scrollTo(int x, int y) {
        this.mContentViewCore.scrollTo((float) x, (float) y);
    }

    protected int computeHorizontalScrollExtent() {
        return this.mContentViewCore.computeHorizontalScrollExtent();
    }

    protected int computeHorizontalScrollOffset() {
        return this.mContentViewCore.computeHorizontalScrollOffset();
    }

    protected int computeHorizontalScrollRange() {
        return this.mContentViewCore.computeHorizontalScrollRange();
    }

    protected int computeVerticalScrollExtent() {
        return this.mContentViewCore.computeVerticalScrollExtent();
    }

    protected int computeVerticalScrollOffset() {
        return this.mContentViewCore.computeVerticalScrollOffset();
    }

    protected int computeVerticalScrollRange() {
        return this.mContentViewCore.computeVerticalScrollRange();
    }

    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        ContentViewClient client =
this.mContentViewCore.getContentViewClient();
        int desiredWidthMeasureSpec = client.getDesiredWidthMeasureSpec();
        if (MeasureSpec.getMode(desiredWidthMeasureSpec) != 0) {
            widthMeasureSpec = desiredWidthMeasureSpec;
        }
        int desiredHeightMeasureSpec = client.getDesiredHeightMeasureSpec();
        if (MeasureSpec.getMode(desiredHeightMeasureSpec) != 0) {
            heightMeasureSpec = desiredHeightMeasureSpec;
        }
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);
    }

    public boolean awakenScrollBars(int startDelay, boolean invalidate) {
        return this.mContentViewCore.awakenScrollBars(startDelay, invalidate);
    }
```

```
    public boolean awakenScrollBars() {
        return super.awakenScrollBars();
    }

    public void onInitializeAccessibilityNodeInfo(AccessibilityNodeInfo info)
{
        super.onInitializeAccessibilityNodeInfo(info);
        this.mContentViewCore.onInitializeAccessibilityNodeInfo(info);
    }

    public void onInitializeAccessibilityEvent(AccessibilityEvent event) {
        super.onInitializeAccessibilityEvent(event);
        this.mContentViewCore.onInitializeAccessibilityEvent(event);
    }

    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        this.mContentViewCore.onAttachedToWindow();
    }

    protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        this.mContentViewCore.onDetachedFromWindow();
    }
    protected void onVisibilityChanged(View changedView, int visibility) {
        super.onVisibilityChanged(changedView, visibility);
        this.mContentViewCore.onVisibilityChanged(changedView, visibility);
    }

    public void extractSmartClipData(int x, int y, int width, int height) {
        this.mContentViewCore.extractSmartClipData(x, y, width, height);
    }

    public void setSmartClipResultHandler(final Handler resultHandler) {
        if (resultHandler == null) {
            this.mContentViewCore.setSmartClipDataListener(null);
        } else {
            this.mContentViewCore.setSmartClipDataListener(new
SmartClipDataListener() {
                public void onSmartClipDataExtracted(String text, String html,
Rect clipRect) {
                    Bundle bundle = new Bundle();
                    bundle.putString("url",
ContentView.this.mContentViewCore.getWebContents().getVisibleUrl());
                    bundle.putString("title",
ContentView.this.mContentViewCore.getWebContents().getTitle());
                    bundle.putParcelable("rect", clipRect);
                    bundle.putString("text", text);
                    bundle.putString("html", html);
                    try {
                        Message msg = Message.obtain(resultHandler, 0);
                        msg.setData(bundle);
                        msg.sendToTarget();
                    } catch (Exception e) {
                        Log.e(ContentView.TAG, "Error calling handler for
smart clip data: ", e);
                    }
                }
            });
        }
    }
```

```
    public boolean super_onKeyUp(int keyCode, KeyEvent event) {
        return super.onKeyUp(keyCode, event);
    }

    public boolean super_dispatchKeyEventPreIme(KeyEvent event) {
        return super.dispatchKeyEventPreIme(event);
    }

    public boolean super_dispatchKeyEvent(KeyEvent event) {
        return super.dispatchKeyEvent(event);
    }

    public boolean super_onGenericMotionEvent(MotionEvent event) {
        return super.onGenericMotionEvent(event);
    }

    public void super_onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
    }

    public boolean super_awakenScrollBars(int startDelay, boolean invalidate)
{
        return super.awakenScrollBars(startDelay, invalidate);
    }
}
@JNINamespace("content")
class DateTimeChooserAndroid {
    private final InputDialogContainer mInputDialogContainer;
    private final long mNativeDateTimeChooserAndroid;

    private native void nativeCancelDialog(long j);

    private native void nativeReplaceDateTime(long j, double d);

    private DateTimeChooserAndroid(Context context, long
nativeDateTimeChooserAndroid) {
        this.mNativeDateTimeChooserAndroid = nativeDateTimeChooserAndroid;
        this.mInputDialogContainer = new InputDialogContainer(context, new
InputActionDelegate() {
            public void replaceDateTime(double value) {

DateTimeChooserAndroid.this.nativeReplaceDateTime(DateTimeChooserAndroid.this.
mNativeDateTimeChooserAndroid, value);
            }
            public void cancelDateTimeDialog() {

DateTimeChooserAndroid.this.nativeCancelDialog(DateTimeChooserAndroid.this.mNa
tiveDateTimeChooserAndroid);
            }
        });
    }
    private void showDialog(int dialogType, double dialogValue, double min,
double max, double step, DateTimeSuggestion[] suggestions) {
        this.mInputDialogContainer.showDialog(dialogType, dialogValue, min,
max, step, suggestions);
    }
```