**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН**

**Некоммерческое акционерное общество**
**АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ**

**кафедра** Компьютерных технологий

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

# ДИПЛОМНЫЙ ПРОЕКТ

**На тему:** „Разработка приложения под Android"

**Специальность** Вычислительная техника и программное обеспечение

**Выполнил (а)** Сапов П.О. ВТу-11-1
(Фамилия и инициалы) группа

**Научный руководитель** Сербин В.В., ст. преп., к.т.н
(Фамилия и инициалы, ученая степень, звание)

**Консультанты:**
**по экономической части:**
Еркемева З.Д. ст. преп.
(Фамилия и инициалы, ученая степень, звание)
Еркемева «20» мая 2014 г.
(подпись)

**по безопасности жизнедеятельности:**
Бишибаева А.С., ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «2» июня 2014 г.
(подпись)

**по применению вычислительной техники:**
Сербин В.В., ст. преп., к.т.н.
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

_____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

**Нормоконтролер:** _____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

**Рецензент:** Сванбаев Е.А. к.ф-м.н. ст. преп.
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

**Алматы 2014 г.**

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН**

**Некоммерческое акционерное общество**

**АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ**

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

# ЗАДАНИЕ

на выполнение дипломного проекта

Студент Сапов Павел Олегович
(фамилия, имя, отчество)

Тема проекта „Разработка приложения для Android"

утверждена приказом ректора № ___ от «___» сентября 20___ г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:

1. Анализ рынка мобильных приложений
2. Анализ предметной области
3. Разработка интерфейса
4. Проектирование приложения
5. Отладка приложения
6. Расчет технико-экономического обоснования
7. Описание безопасности жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей)

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

Рекомендуемая основная литература

Сайт http://startandroid.ru
Сайт http://developer.android.com
Рето М. Android 2. Программирование приложений для планшетных компьютеров и смартфонов. – М. Эксмо, 2011.
СНиП РК 2.04.-05.2001. Естественное и искусственное освещение. Айдархановы
Айдарханова М. Основы экономической теории – М.: Фоливант, 2000.

Консультанты по проекту с указанием относящихся к ним разделов

| Раздел | Консультант | Сроки | Подпись |
|---|---|---|---|
| Экономика | Ерешева З.Д. | 20.05 – 30.05.14 | Ерешева |
| БЖД | Вишнева А.С. | 20.04 – 02.06.14 | |
| рецензия | Сванбаев Е.А. | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Г Р А Ф И К

подготовки дипломного проекта

| № п/п | Наименование разделов, перечень разрабатываемых вопросов | Сроки представления руководителю | Примечание |
|---|---|---|---|
| 1 | Сбор необходимого материала | 02.04.14 | |
| 2 | Исследование предметной области | 04.04.14 | |
| 3. | Постановка задачи | 04.04.14 | |
| 4 | Выбор платформы | 04.04.14 | |
| 5 | Выбор среды разработки | 04.04.14 | |
| 6 | Проектирование интерфейса | 15.04.14 | |
| 7 | Разработка приложения | 05.05.14 | |
| 8 | Тестирование и отладка | 07.05.14 | |
| 9 | Написание раздела „Технико- | 20.05.14 | |
| | экономического обоснования" | | |
| 10 | Написание раздела „Безопасность | 25.05.14 | |
| | жизнедеятельности" | | |
| 11 | Оформление отчета | 28.05.14 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Дата выдачи задания «____»_____20___ г.

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

Руководитель _____ _____
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ _____
(подпись) (Фамилия и инициалы)

**Аннотация**

В данном дипломном проекта рассмотрена разработка прикладного приложения для операционной системы Android, являющегося мультимедийным приложением для создания и обработки музыки. Данное приложение может быть опубликовано в магазине мобильных приложений Google Play и полностью соответствует всем предъявляемым для публикации требованиям.

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

Также составлено экономическое обоснование проекта, подтверждающее его экономическую целесообразность.

**Аңдатпа**

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

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

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

**Annotation**

In this diplom project examined the development of application software for the operating system Android, which is a multimedia application for the creation and processing of music. This application may be published in the mobile application store Google Play and meets all the requirements for publication requirements.

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

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

**Содержание**

## Введение

Актуальность темы исследования. На данный момент перед нами, как разработчиками приложений для мобильных платформ, открываются широкие возможности. Несколько лет назад казалось невероятным, что обычный разработчик сможет продавать свои приложения миллионам пользователей по всему миру с помощью магазинов приложений, а пользователи будут эти приложения покупать. По разным прогнозам объем мирового рынка мобильных приложений в 2011 году составил от 9 до 12 млрд долларов США, и в ближайшие 5 лет данный показатель вырастет в 4 раза. Одной из причин является то, что конкуренция между основными производителями операционных систем для смартфонов, такими как Microsoft с платформой Windows Phone, Apple с IOS и Google с Android, постоянно усиливается. Это позволяет предположить, что в скором времени данные платформы продолжат развиваться быстрыми темпами, а это означает, что потребность в разработчиках мобильных приложений станет только расти.

В чём же заключается уникальность платформы Android? Основная идея Google состоит в том, что компания предлагает открытый доступ исходные коды своей операционной системы, предоставляет набор удобных инструментов для разработки и хорошо документированный комплект SDK, что должно со временем привести к появлению большого количества программного обеспечения для этой платформы. За несколько лет Android стал самым успешным проектом для мобильных телефонов. Android захватывает рынок мобильных телефонов, постепенно вытесняя с него общепризнанных лидеров. Система Android устанавливается теперь не только на смартфоны, данная платформа была адаптирована для планшетов и нетбуков.

Большим шагом в развитии Google Android стало открытие в октябре 2008 года онлайн-магазина приложений – Android Market, в котором можно приобрести программы и другой софт для устройств на базе новой платформы. Кроме того, для разработчиков программного обеспечения появилась возможность брать плату за свои приложения в Android Market, что делает разработку приложений под эту платформу ещё более привлекательной.

Раньше, когда система android не была совершенной случались проблемы crash системы, плохая оптимизация и так далее. Программировать было достаточно тяжело, но с выходом android V.2 можно сказать что мир изменился и почти каждый выход новой ОС на Android доказывал, что Google стремится сделать лучше в целом структуру организации работы ОС.

Так же на данный момент мы видим таких монстров системы как Eclipse и Android Studio, которые набирают обороты и набирают больше сторонников системы android в данном случае это программисты всё же система является бесплатной и за ней стоит Google. В то время как за системой VS стоит монстр Microsoft за которым огромные деньги. Но в принципе языки между собой

очень схожи оба являются кроссплатформенными, но я со своей стороны решил отдать большее предпочтение Java чем C#, так как имеется сотовый аппарат на android для тестирования данных. Да и вообще на данный момент Windows Phone последняя версия 8.1 против android 4.4 Kit-Kat большее предпочтение на данный момент люди отдают андроидовидным телефонам нежели Windows Phone, так же IOS является одним из конкурентов Android, но по последним данным даже iPhone уже не столь популярен как телефоны на android. Данная система сейчас везде в коммуникаторах, сотовых, планшетах, телевизорах и можно бесконечно перечислять данные виды оборудования, но факт на лицо и будущее по моему мнению стоит за Google.

Целью данного дипломного проекта было создание мобильного приложения «MusicPro» для операционной системы Android, являющегося мультимедийным приложением для создания и редактирования музыки, соответствующего всем «гайдлайнам» Android OS.

# 1. Принципы разработки

## 1.1 Операционная система Android

Android – одна из операционных систем нового поколения, созданных для работы с аппаратным обеспечением современных мобильных устройств. На сегодняшний день Windows Mobile, Apple iPhone и Palm Pre предлагают достаточно мощные и более простые в использовании среды разработки мобильных приложений. Однако в отличие от Android это запатентованные операционные системы, в которых в определенных случаях приоритет отдается встроенному ПО, а не приложениям сторонних программистов. Кроме того, эти операционные системы ограничивают возможности взаимодействия приложений с данными телефона, а также ограничивают или контролируют процесс распространения сторонних приложений, созданных для данных платформ.

Android дает новые возможности для мобильных приложений, предлагая открытую среду разработки, построенную на открытом ядре Linux. У всех приложений есть доступ к аппаратным средствам устройства, для чего используются специальные серии API-библиотек. Кроме того, здесь включена полная и контролируемая поддержка взаимодействия приложений.

На платформе Android все программы имеют одинаковый статус. Сторонние приложения написаны на том же API, что и встроенное ПО, при этом во всех программах одинаковое время исполнения. Пользователи могут удалять или заменять встроенные ПО на альтернативные сторонние разработки, будь то номеронабиратель или Рабочий стол.

Упрощенно Android можно представить как комбинацию трех компонентов:

1 свободной операционной системы с открытыми исходными кодами;

2 среды разработки с открытыми исходными кодами для создания мобильных приложений;

3 устройств, по большей части мобильных телефонов, на которых установлена операционная система Android вместе с разработанными для нее приложениями.

Android включает несколько необходимых и взаимозависимых элементов:

– 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.

### 1.1.1 Встроенные приложения Android

Телефоны с системой Android снабжены набором предустановленных программ, разработанных в рамках проекта Android Open Source Project (AOSP) (Проект открытых исходных кодов для Android).

Перечислим основные из них:

– e-mail-клиент;

– приложение для работы с SMS;

– полный набор инструментов для управления личными данными, включая календарь и адресную книгу;

– браузер на базе WebKit;

– музыкальный плеер и фотогалерея;

– калькулятор;

– «Рабочий стол»;

– будильник.

Во многих случаях Android включает также следующее лицензионное ПО от Google:

– приложение Android Market для загрузки сторонних программ, разработанных для платформы Android;

– полноценное приложение Google Maps, включающее функции Street-View («Просмотр улиц»), Driving Directions («Показ проезда»), маршрутизируемую навигацию, спутниковую карту и информацию о пробках;

– программу для работы с почтой Gmail;

– программу для обмена мгновенными сообщениями Google Talk;

– видеоплеер для работы с сервисом YouTube.

Данные, к которым имеют доступ многие из этих приложений, например адресная книга, открыты и для программ сторонних разработчиков. Кроме этого приложения могут обрабатывать такие события, как входящий звонок или получение SMS.

Внешний вид программ, которые установлены на новых телефонах под управлением Android, может сильно варьироваться в зависимости от производителя аппаратного обеспечения и/или оператора, дистрибьютора.

Открытый характер платформы Android означает, что операторы или производители комплектного оборудования (OEM) могут менять пользовательский интерфейс и набор программ на любом устройстве под управлением Android. Некоторые производители разработали свои собственные интерфейсы на базе Android, например Sense от HTC, MotoBlur от Motorola и пользовательский интерфейс от Sony Ericsson.

Важно отметить, что для всех совместимых устройств платформа и среда разработки остаются неизменными независимо от производителя или оператора. Пользовательский интерфейс может меняться, однако программы будут работать абсолютно одинаково на всех совместимых с Android устройствах.

### 1.1.2 Основные характеристики среды разработки для платформы Android

Главным сокровищем Android как среды разработки стал ее API.

Android как нейтральная к приложениям платформа предоставляет возможность создавать программы, которые станут такой же неотъемлемой частью телефона, как и компоненты, поставляемые в комплекте.

Следующий список иллюстрирует основные характеристики Android:

– отсутствие расходов на использование лицензии, распространение и разработку, а также каких-либо механизмов сертификации готовых программных продуктов;

– доступ к Wi-Fi-устройству;

– в сетях GSM, EDGE и 3G, предназначенных для телефонии и передачи данных, можно звонить или принимать звонки и SMS, отправлять и получать данные;

– комплексный API для работы с навигационными службами, например GPS;

– полный контроль над мультимедийными устройствами, включая проигрывание или запись информации с камеры и микрофона;

– API для работы с сенсорными устройствами, например акселерометром и компасом;

– библиотеки для работы с Bluetooth с возможностью передачи данных по протоколу p2p;

– передача IPC-сообщений;
– хранилища для общих данных;
– фоновые приложения и процессы;
– виджеты для «Рабочего стола», Живые каталоги (Live Folders) и Живые обои (Live Wallpaper);
– возможность интеграции результатов поиска приложения в системный поиск;
– встроенный браузер на базе WebKit с открытыми исходными кодами и поддержкой HTML5;
– полная поддержка приложений, которые используют функционал работы с картами в своем пользовательском интерфейсе;
– оптимизированная под мобильные устройства графическая система с аппаратным ускорением, включающая библиотеку для работы с векторной 2D-графикой и поддержку трехмерной графики с использованием OpenGL ES 2.0;
– мультимедийные библиотеки для проигрывания и записи аудио, видеофайлов или изображений;
– локализация с помощью инструментов для работы с динамическими ресурсами;
– набор программных компонентов для повторного использования компонентов и замещения встроенных приложений.

### 1.1.3 Фреймворк разработчика

Язык программирования приложений для платформы Android – Java. Однако они исполняются не на классической Java VM, а на специальной виртуальной машине Dalvik.

Каждое приложение для Android функционирует в отдельном процессе внутри собственного экземпляра машины Dalvik. Вся ответственность за память и управление процессами возлагается на Android, который останавливает или убивает процессы, если нужно освободить ресурсы.

Dalvik и Android находятся на вершине ядра Linux, которое занимается низкоуровневым взаимодействием с аппаратным обеспечением, включая работу драйверов и управление памятью. При этом набор встроенного API позволяет получить доступ ко всем службам, функционалу и аппаратной начинке.

### 1.1.4 Программный стек Android

Программный стек Android состоит из элементов, показанных на рисунке 1.1. Их подробное описание приводится ниже. Упрощенно 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.

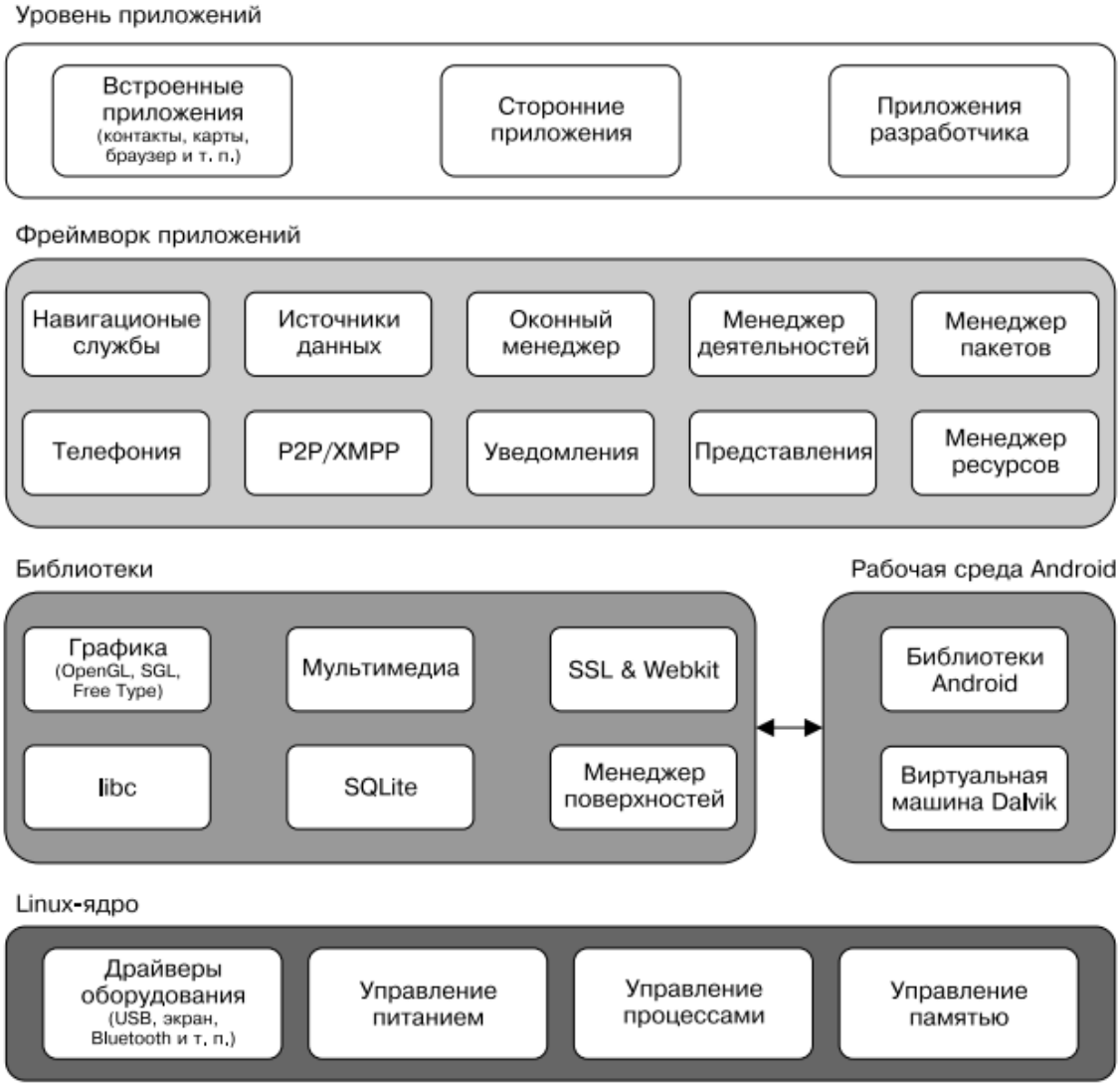


Рисунок 1.1 – Программный стек Android

*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:

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

2 менеджер интерфейса, отвечающий за управление отображением;

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

4 библиотеку SQLite, обеспечивающую работу встроенных баз данных;

5 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.

### 1.1.5 Виртуальная машина Dalvik

Один из ключевых компонентов Android – 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.

В основе ВМ Dalvik ядро Linux, которое обеспечивает работу таких низкоуровневых функций, как безопасность, потоки, 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.

Если для приложения важны присущие C/C++ 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.

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

который позволяет им не беспокоиться об особенностях конструкции того или иного устройства.

ВМ Dalvik запускает исполняемые файлы, формат которых оптимизирован под минимальное использование памяти. Вы создаете исполняемый файл с расширением .dex путем трансформирования компилированных классов, написанных на языке Java, используя для этого инструменты, входящие в состав среды разработки. В следующей главе вы узнаете о создании исполняемых файлов формата Dalvik.

## 1.2 Из чего состоят приложения в Android

Приложения в Android состоят из слабосвязанных компонентов, которые собираются воедино с помощью программного манифеста. Манифест – файл, описывающий все компоненты приложения и способы их взаимодействия, а также метаданные, в том числе требования к платформе и аппаратной конфигурации.

Компоненты, перечисленные ниже – кирпичики, из которых состоят приложения.

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

*Сервисы.* Невидимые двигатели вашего приложения. Сервисные компоненты работают в фоновом режиме, запуская уведомления, обновляя Источники данных и видимые Активности. Используются для регулярных операций, которые должны продолжаться даже тогда, когда Активности вашего приложения не на переднем плане.

*Источники данных.* Хранилища информации. Данные компоненты нужны для управления базами данных в пределах одного приложения и предоставления к ним доступа извне. Источники данных задействуются при обмене информацией между разными программами. Это значит, 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 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.

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

*Виджеты.* Визуальные программные компоненты, которые можно добавлять на домашний экран. Этот особый вид Широковещательных приемников позволяет создавать динамические, интерактивные компоненты, которые пользователи могут встраивать в свои домашние экраны. В главе 10 вы узнаете, как создавать собственные виджеты.

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

### 1.2.1 Что такое AndroidManifest.xml

Любое приложение, создаваемое в Android, содержит файл манифеста, AndroidManifest.xml, который хранится в корневом каталоге проекта. Манифест позволяет описывать структуру и метаданные вашего приложения, его компоненты и требования.

Манифест включает в себя узлы (теги) для каждого компонента (Активностей, Сервисов, Источников данных и Широковещательных приемников), из которых состоит ваше приложение, и с помощью Фильтров намерений (Intent Filters) и полномочий определяет, каким образом они взаимодействуют друг с другом и со сторонними программами.

В манифесте предусмотрены атрибуты для указания метаданных (значков и визуальных стилей). Надо отметить, что дополнительные узлы верхнего уровня можно использовать для описания настроек безопасности, модульных тестов (юнит-тестов), аппаратных и системных требований.

Манифест содержит корневой тег <manifest> с атрибутом package, который ссылается на пакет проекта. Как правило, этот тег также включает в себя атрибут xmlns:android, поддерживаемый системными узлами внутри файла.

Атрибут versionCode предназначен для задания текущей версии приложения в виде целого числа. Это внутреннее значение используется для сравнения версий программы. Примените атрибут versionName для указания публичной версии, которая выводится для пользователей.

Типичный тег <manifest> показан во фрагменте кода.

```
        <manifest
xmlns:android=http://schemas.android.com/apk/res/android
                  package="com.my_domain.my_app"
                  android:versionCode="1"
                  android:versionName="0.9 Beta">
                  [ ... вложенные узлы манифеста ... ]
        </manifest>
```

Тег <manifest> включает в себя узлы, описывающие программные компоненты, настройки безопасности, классы для тестирования и требования, из которых состоит ваше приложение. Укажем теги, доступные внутри узла <manifest>, а также фрагменты кода в формате XML, демонстрирующие, как этими тегами пользоваться.

*Uses-sdk*

Позволяет задать минимальную, максимальную и целевую версии SDK, которые должны быть доступны на устройстве, чтобы ваше приложение смогло правильно функционировать. Основываясь на версии SDK, которая поддерживается установленной платформой, и используя сочетание атрибутов minSDKVersion, maxSDKVersion и targetSDKVersion, вы можете ограничить круг устройств, способных запускать приложение.

Атрибут minSDKVersion указывает на минимальную версию SDK, содержащую API, которая используется в вашей программе. Если не зададите минимальную версию, применится значение по умолчанию, а ваше приложение не сможет корректно работать, если попытается получить доступ к API, которые недоступны на текущем устройстве.

Атрибут maxSDKVersion позволяет определить самую позднюю версию, которую вы готовы поддерживать. Ваше приложение будет невидимым в Android Market для устройств, управляемых системой с более свежей версией. Устанавливать значение для этого атрибута рекомендуется только в том случае, если вы абсолютно уверены, что приложение не работает на платформе с версией, выше заданной.

targetSDKVersion позволяет указать платформу, для которой вы разрабатывали и тестировали приложение. Устанавливая значение для этого атрибута, вы сообщаете системе, что для поддержки этой конкретной версии не требуется никаких изменений, связанных с прямой или обратной совместимостью.

```
<uses-sdk android:minSdkVersion="4"
          android:targetSdkVersion="5">
</uses-sdk>
```

*Uses-configuration*

Используйте теги uses-configuration, чтобы указать все механизмы ввода данных, поддерживаемые вашим приложением. Вы можете задать любую комбинацию, содержащую следующие устройства:

1 reqFiveWayNav – укажите для этого атрибута значение true, если вам необходимо устройство ввода, поддерживающее навигацию вверх, вниз, влево, вправо, а также нажатие выделенного элемента; в эту категорию входят как трекболы, так и манипуляторы D-pad;

2 reqHardKeyboard – если вашему приложению нужна аппаратная клавиатура, укажите значение true;

3 reqKeyboardType – позволяет задать тип клавиатуры – nokeys, qwerty, twelvekey или undefined;

4 reqNavigation – если требуется устройство для навигации, укажите одно из следующих значений – nonav, dpad, trackball, wheel или undefined;

5 reqTouchScreen – если вашему приложению понадобится сенсорный экран, выберите одно из следующих значений – notouch, stylus, finger или undefined.

Вы можете задать несколько поддерживаемых конфигураций, например устройство с емкостным сенсорным экраном, трекболом и аппаратной клавиатурой (либо qwerty, либо twelvekey), как показано ниже.

```
<uses-configuration android:reqTouchScreen=["finger"]
                    android:reqNavigation=["trackball"]
                    android:reqHardKeyboard=["true"]
                    android:reqKeyboardType=["qwerty"/>
<uses-configuration android:reqTouchScreen=["finger"]
                    android:reqNavigation=["trackball"]
                    android:reqHardKeyboard=["true"]
                 android:reqKeyboardType=["twelvekey"]/>
```

*Uses-feature*

Одно из преимуществ Android – широкий диапазон аппаратных платформ, на которых он может работать. Используйте простые теги uses-feature, чтобы задать все необходимые приложению аппаратные возможности. Это предотвратит установку вашей программы на устройства, которые не соответсвуют аппаратным требованиям. Можете запросить поддержку любого необя- зательного для совместимых устройств оборудования. На сегодняшний день аппаратные возможности предлагают следующие варианты:

– android.hardware.camera (если для работы приложения нужна аппаратная камера);

– android.hardware.camera.autofocus (если требуется камера с автоматической фокусировкой).

Вы также можете использовать тег uses-feature, чтобы задать минимальную версию OpenGL, которая требуется для работы вашего приложения. С помощью атрибута glEsVersion укажите версию OpenGL ES в

виде целого числа. Первые 16 бит соответствуют мажорной версии, а последние – минорной.

```
<uses-feature android:glEsVersion=" 0x00010001"
               android:name="android.hardware.camera" />
```

*Supports-screens*

После первой волны устройств с экранами HVGA в 2009 году список аппаратов под управлением Android пополнился моделями с поддержкой WVGA и QVGA. Поскольку будущие устройства, вероятно, станут оснащаться большими дисплеями, с помощью тега supports-screen вы можете указать экранные размеры, которые поддерживаются (и не поддерживаются) вашим приложением.

Точные цифры будут варьироваться в зависимости от аппаратного обеспечения, но в целом соответствие размеров и разрешений экранов определяется следующим образом:

1 smallScreens – экраны с разрешением меньшим, чем обычное HVGA, как правило, речь идет о QVGA;

2 normalScreens – используется для описания экранов стандартных мобильных телефонов, как минимум HVGA, включая HVGA и WQVGA;

3 largeScreens – экраны больших размеров, значительно больше, чем у мобильного телефона;

4 anyDensity – установите значение true, если ваше приложение способно масштабироваться для отображения на экране с любым разрешением.

В версии SDK 1.6 (API level 4) значения по умолчанию для каждого атрибута – true. Используйте этот тег для указания размеров экранов, которые вы не поддерживаете.

```
<supports-screens android:smallScreens=["false"]
                  android:normalScreens=["true"]
                  android:largeScreens=["true"]
                  android:anyDensity=["false"] />
```

*Application*

В манифесте может присутствовать только один экземпляр данного тега. В нем используются атрибуты, содержащие метаданные для вашего приложения (включая его название, значок и визуальный стиль). Во время разработки вы должны устанавливать атрибуту debuggable значение true, чтобы активизировать режим отладки, хотя для конечных версий, скорее всего, его нужно отключить.

Тег <application> также играет роль контейнера, который включает в себя узлы для Активностей, Сервисов, Источников данных и Широковещательных приемников, описывающих компоненты приложения. Кроме того, вы можете задать собственную реализацию класса Application. Далее в этой главе вы

узнаете, как наследовать данный класс и использовать его для управления состоянием приложения.

```
<application android:icon="@drawable/icon"
              android:theme="@style/my_theme"
              android:name="MyApplication"
              android:debuggable="true">
               [ ... вложенные теги ... ]
</application>
```

*Activity*

Тег <activity> требуется для каждой Активности, которую отображает приложение. Используйте атрибут android:name для указания имени класса Активности.

С помощью этих тегов добавьте главнуюАктивность, которая будет запускаться первой, а также остальные экраны и диалоговые окна, которые могут показываться. Попытка запустить Активности без соответствующего описания в манифесте приведет к выбросу исключения. Каждый тег <activity> поддерживает вложенные узлы <intent-filter>, указывающие, какие именно Намерения могут запустить Активность.

```
<activity                        android:name=".MyActivity"
android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category
android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

*Service*

Как и в предыдущем случае, каждый класс Сервиса должен иметь тег service. Теги service поддерживают вложенные узлы <intent-filter>, с помощью которых происходит латентное связывание.

```
<service                             android:enabled="true"
android:name=".MyService"></service>
```

*Provider*

С помощью этого тега указываются все Источники данных в приложении. Источники данных, описанные в главе 7, используются для управления доступом к базам данных и для обмена информацией в рамках одной или нескольких программ.

```
<provider android:permission="com.paad.MY_PERMISSION"
          android:name=".MyContentProvider"
          android:enabled="true"
```

```
android:authorities="com.paad.myapp.MyContentProvider">
    </provider>
```

*Receiver*

Добавляя в манифест тег receiver, можно зарегистрировать Широковещательный приемник, не запуская при этом приложение. Широковещательные приемники отслеживают события на глобальном уровне: пройдя регистрацию, они начнут срабатывать при трансляции системой или приложением соответствующего Намерения. Регистрируя их в манифесте, можете сделать этот процесс полностью анонимным. При трансляции соответствующего Намерения ваше приложение стартует автоматически, запуская зарегистрированный Приемник.

```
    <receiver android:enabled="true"
              android:label="My Intent Receiver"
              android:name=".MyIntentReceiver">
    </receiver>
```

*Uses-permission*

Теги uses-permission как часть системы безопасности описывают полномочия, которые, по вашему мнению, нужны приложению для полноценной работы. Добавленные полномочия предоставляются пользователю до установки. Для использования многих стандартных сервисов в Android требуются полномочия (в частности, для действий, связанных с платными услугами и безопасностью, таких как телефонные звонки, прием SMS или использование геолокационных сервисов).

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

*Permission*

Сторонние приложения также могут указывать полномочия, прежде чем предоставлять доступ к общим программным компонентам. Чтобы ограничить доступ к компоненту приложения, вы должны описать соответствующие полномочия в манифесте. Для этого необходимо использовать тег permission.

Компоненты текущего приложения могут требовать полномочия с помощью атрибутов android:permission. Другие программы должны содержать в своем манифесте теги uses-permission, чтобы использовать эти защищенные компоненты.

Внутри тегаpermission вы можете указать уровень доступа, который обеспечивается данным полномочием (normal, dangerous, signature, signatureOrSystem), метку и внешний ресурс, содержащий описание и объяснение рисков, которыми сопровождается выдача этого полномочия.

```
<permission android:name="com.paad.DETONATE_DEVICE"
                 android:protectionLevel="dangerous"
                 android:label="Self Destruct"

android:description="@string/detonate_description">
        </permission>
```

*Instrumentation*

Классы, производные от Instrumentation, предоставляют фреймворк для тестирования программных компонентов во время их выполнения. Они содержат методы-перехватчики, с помощью которых отслеживаются работа программы и ее взаимодействия с системными ресурсами.

```
        <instrumentation android:label="My Test"
                         android:name=".MyTestClass"

android:targetPackage="com.paad.aPackage">
        </instrumentation>
```

Подробное описание манифеста и всех этих тегов можно найти по адресу *http://developer.android.com/guide/topics/manifest/manifest-intro.html*

Мастер создания проектов в составе ADT (New Project Wizard) автоматически добавляет файл с манифестом для каждого нового проекта.

### 1.2.2 Создание простых значений

Поддерживаются простые значения – строки, цвета, размеры и массивы (строковые и целочисленные), эти данные хранятся в формате XML внутри каталога res/values.

Используя теги, указываются типы хранимых значений.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">To Do List</string>
  <color name="app_background">#FF0000FF</color>
  <dimen name="default_border">5px</dimen>
  <array name="string_array">
    <item>Item 1</item>
    <item>Item 2</item>
    <item>Item 3</item>
  </array>
  <array name="integer_array">
    <item>3</item>
    <item>2</item>
    <item>1</item>
  </array>
```

```
</resources>
```

В этом примере содержатся все доступные типы простых значений. Каждый тип ресурсов принято хранить в отдельном файле, например файл res/values/strings.xml включает только строковые константы.

### 1.2.3 Ресурсы

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

*Список ресурсов*

Типы ресурсов и их местоположение:

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

Названия не могут отличаться:

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

*Использование ресурсов в коде*

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

```
R.resource_type.resource_namет
```

или

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

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

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

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

*Ссылка на Ресурсы*

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

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

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

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

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

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

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

*Альтернативные ресурсы и локализация*

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

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

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

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

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

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

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

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

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

*R.java*

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

### 1.2.4 Пользовательские интерфейсы в Android

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

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

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

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

```
</LinearLayout>
```

*Иерархия Элементов Экрана*

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

*Views*

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

*Viewgroups*

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

*UI с древовидной структурой*

На платформе Android Вы определяете UI Activity использование дерева View и Viewgroup узлов, как показано в диаграмме ниже. Дерево может быть столь же простым или сложным, как Вы его сделаете, и Вы можете построить его, используя наборы предопределенных графических фрагментов и Layouts Android, или заказных типов View, которые Вы создаете самостоятельно. На рисунке 1.2 изображена древовидная система пользовательского интерфейса.

Чтобы прикрепить дерево к экрану и просчитать его, Ваш Activity вызывает свой метод setContentView() и передает информацию на корневой объект узла. Как только у система Android получает информацию на корневой объект узла, она начинает работать непосредственно с узлом, чтобы измерить, и

просчитать дерево. Когда Ваш Activity становится активным и получает приоритет, система регистрирует Ваш Activity и просит корневой узел измерить и просчитать дерево. Тогда корневой узел просит, чтобы его дочерние вершины просчитывали себя – в свою очередь, каждый Viewgroup узел в дереве ответственен за просчет его прямых дочерних узлов. Как упомянуто ранее, у каждой группы View есть ответственность измерения ее доступного пространства, расположения ее дочерних узлов, и вызов draw() на каждом дочернем узле, чтобы позволить все им просчитывать себя. Дочерние узлы могут просить размер и местоположение в родителе, но у родительского объекта есть конечное решение, где и насколько большой каждый ребенок может быть.

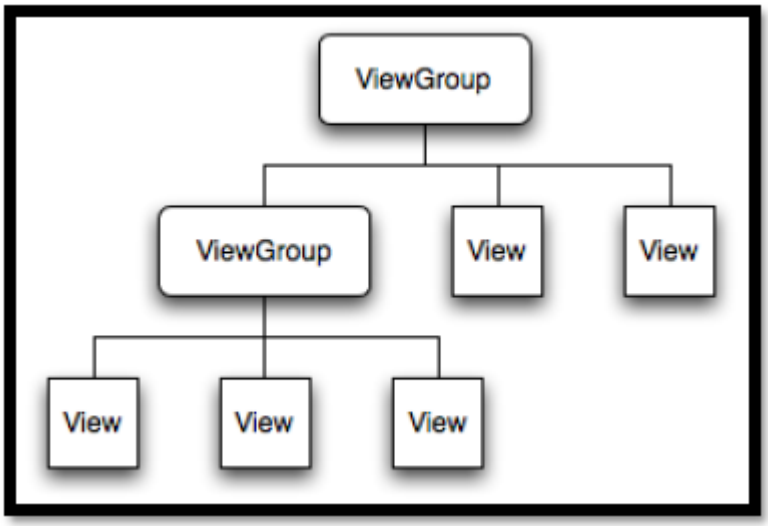


Рисунок 1.2 – UI ОС Android – древовидная структура

*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.

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

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

# 2 Инструменты разработки

## 2.1 Инструменты разработки графической оболочки

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

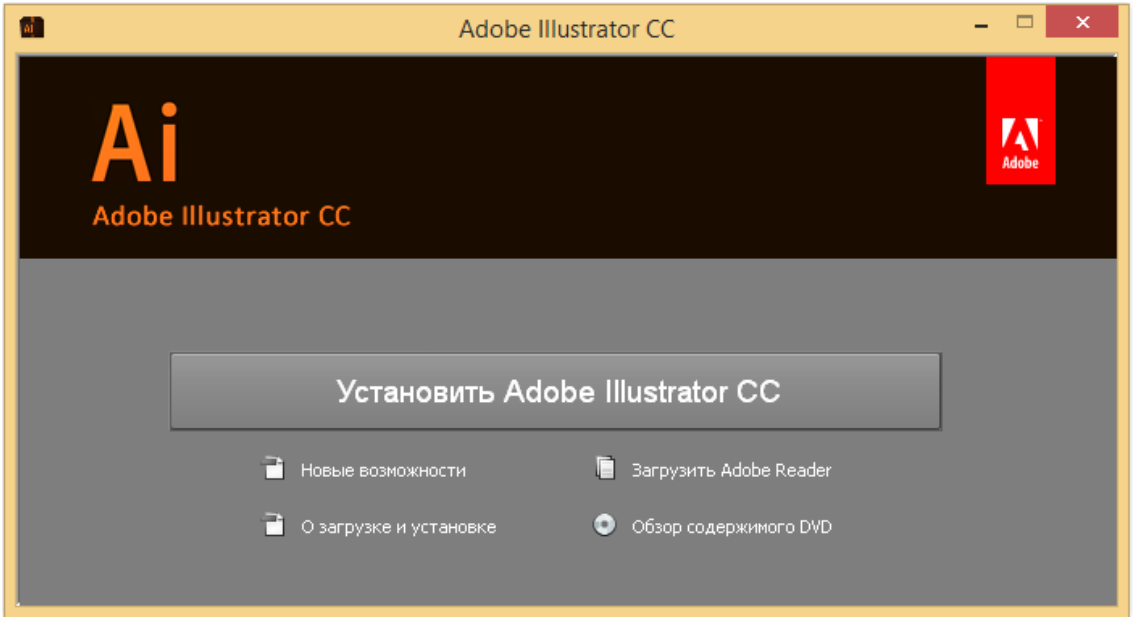


Рисунок 2.1 – Начало установки Adobe Illustrator CC

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

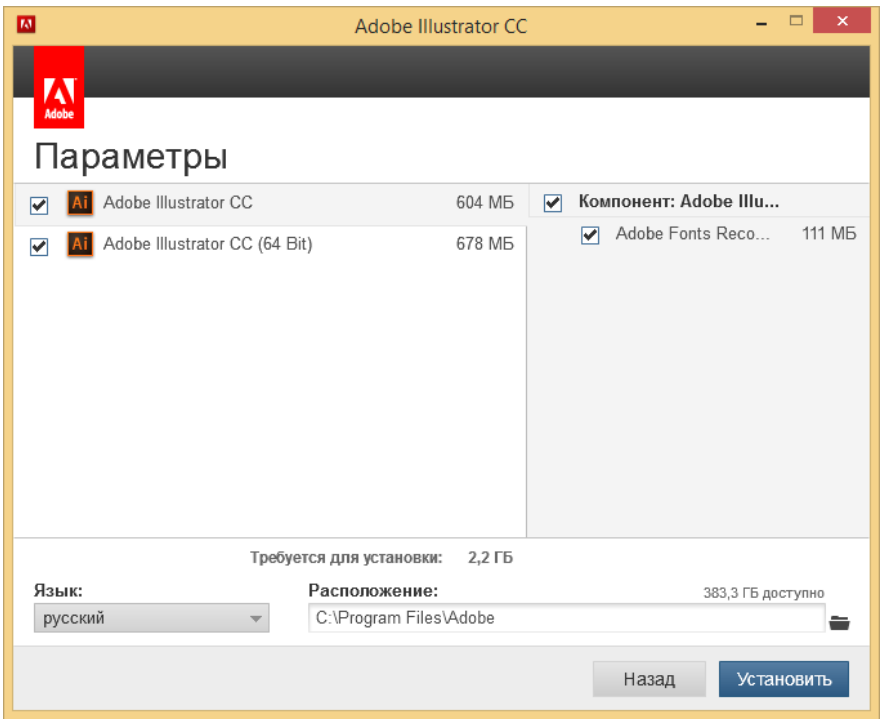


Рисунок 2.2 – Выбор устанавливаемых версий приложения

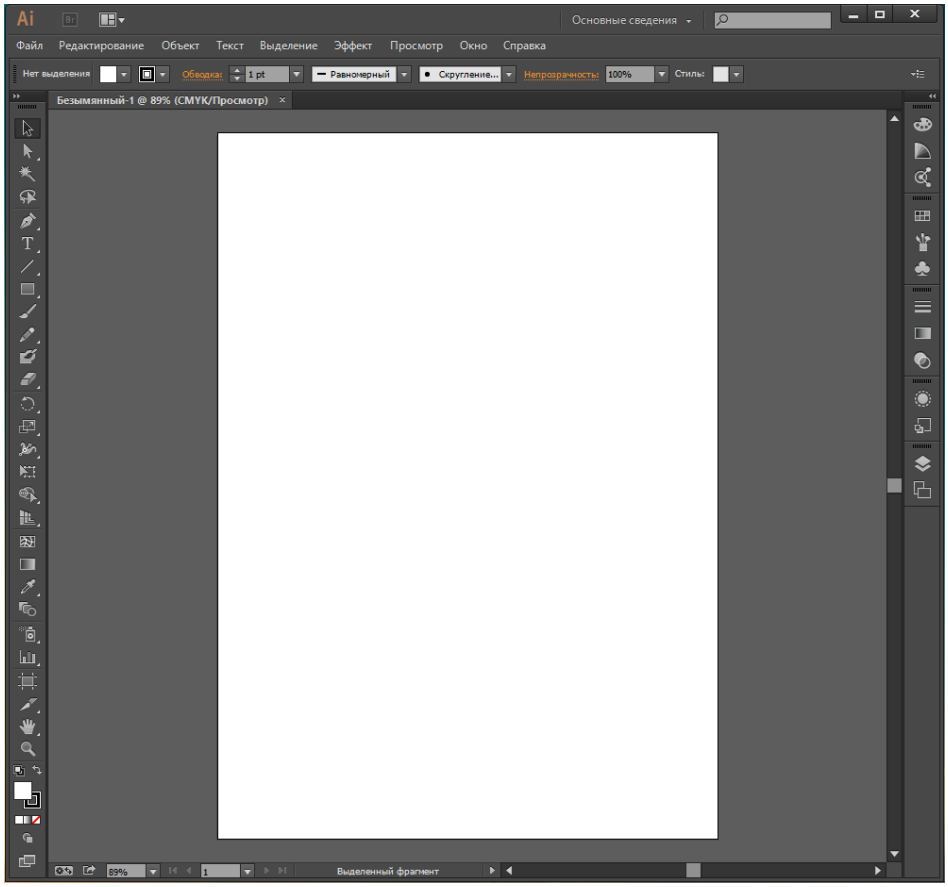


Рисунок 2.3 – Окно запущенного приложения Illustrator CC

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

*Динамические углы*

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

*Полностью модернизированный инструмент «Карандаш»*

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

*Изменение формы сегмента контура*

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

*Интеграция с Typekit*

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

*Поддержка ОС Windows 8*

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

*Настраиваемая панель инструментов*

Персонализируйте свое рабочее пространство и инструменты для определенных задач. Создавайте специализированные наборы инструментов, перетаскивая на пользовательскую панель только нужные инструменты, например, инструменты для рисования, редактирования или выделения. После чего можно скрыть всю панель «Инструменты», получив больше пространства для творчества.

*Импорт и экспорт настроек*

Синхронизируйте свои настройки приложения Illustrator на нескольких компьютерах. Просто экспортируйте их в папку, из которой другие

пользователи смогут затем импортировать их. Функция синхронизации настроек позволяет вам не только стандартизировать настройки на своих устройствах, но и делиться ими с коллегами.

*Новые функции, помогающие экономить время*

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

*Улучшенные возможности рисования с учетом перспективы*

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

*Обновления функции экспорта в формат SVG*

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

*Инструмент «Изменение текста»*

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

*Изображения в кистях*

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

*Поиск шрифта*

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

*Использование нескольких файлов*

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

*Извлечение каскадных таблиц стилей (CSS)*

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

*Синхронизация цвета*

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

*Преобразование текста из точки в текст в области и наоборот*

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

*Автоуглы для узорчатых кистей*

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

*Применение инструмента «Свободное трансформирование»*

Совершенствуйте свои навыки в создании дизайна с помощью инструмента «Свободное трансформирование». Перемещайте, масштабируйте и поворачивайте объекты прямо на сенсорном экране мобильного устройства. Либо воспользуйтесь мышью или другим манипулятором, чтобы интуитивно и быстро трансформировать объекты прямо на монтажной области.

*Интеграция с Behance*

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

*Синхронизация настроек*

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

*Упаковка файлов*

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

*Извлечение изображений*

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

*Использование нескольких монтажных областей*

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

*Обводки переменной ширины*

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

*Изображения*

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

*Система Adobe Mercury Performance*

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

## 2.2 Инструменты разработки кода приложения

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

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

Java SE Development Kit является бесплатным инструментарием для разработки и распространяется бесплатно. Таким образом, после перехода по приведенной выше ссылке, вам необходимо выбрать версию, соответствующую

вашей ОС, Windows, Linux или же Mac OS X. После загрузки исполняемого файла его необходимо запустить и следуя инструкциям, установить данный инструментарий.

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

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

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

После запуска приложения, вас будет ожидать экран приветствия изображенный на рисунке 2.4.

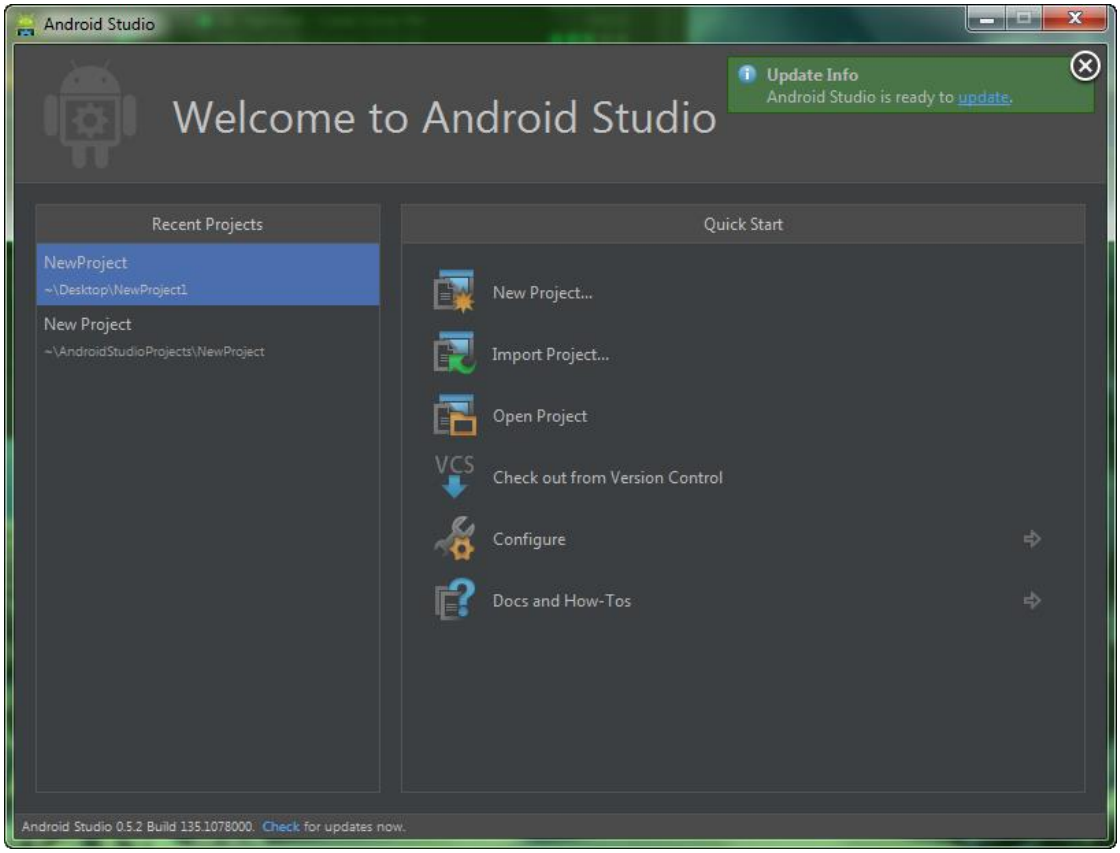


Рисунок 2.4 – Экран приветствия Android Studio

Здесь вы сможете:

– Создать новый проект.
– Импортировать уже существующий проект.
– Открыть уже существующий проект.

– Открыть любой недавний проект, а также выполнить поиск среди них.
– Проверить актуальность версии ПО.
– Изменить настройки Android Studio.
– Получить доступ к справе.

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

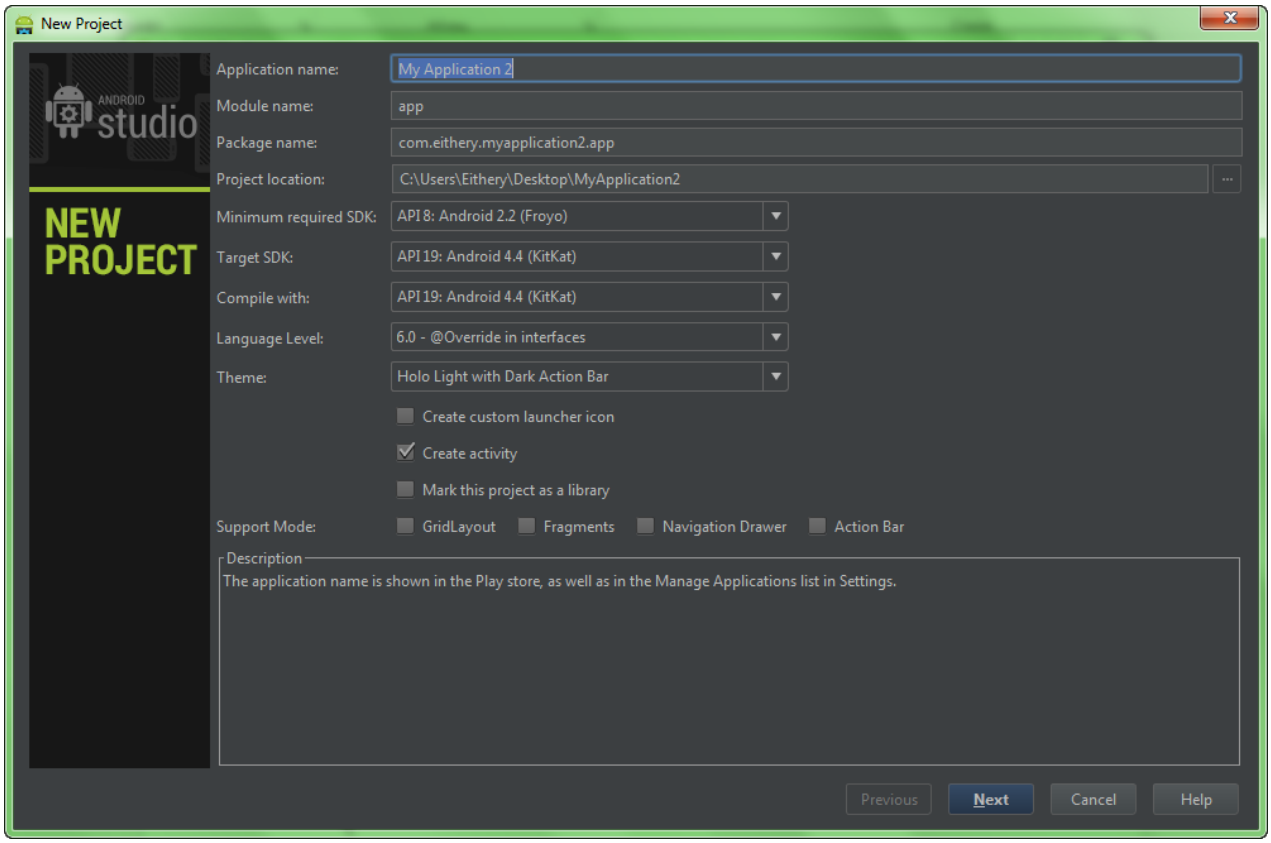


Рисунок 2.5 – Окно создания проекта Android Studio

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

Если в окне создания проекта было выбрано «Create custom launcher icon», то после нажатия кнопки «next» вы увидите окно создания и редактирования значка приложения, которое также изображено на рисунке 2.6. Этот редактор удовлетворит потребности практически всех разработчиков, так

как позволяет создать быстро и качественно значки приложения для разных плотностей экрана и сразу же увидеть результат.

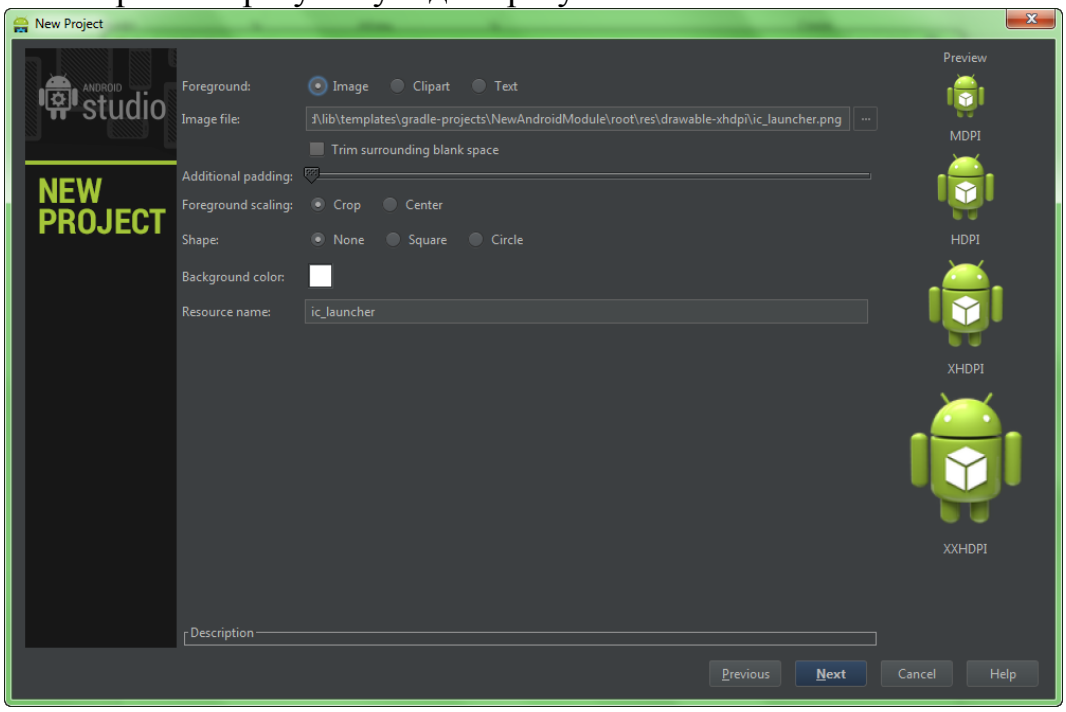


Рисунок 2.6 – Окно создания значка приложения

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

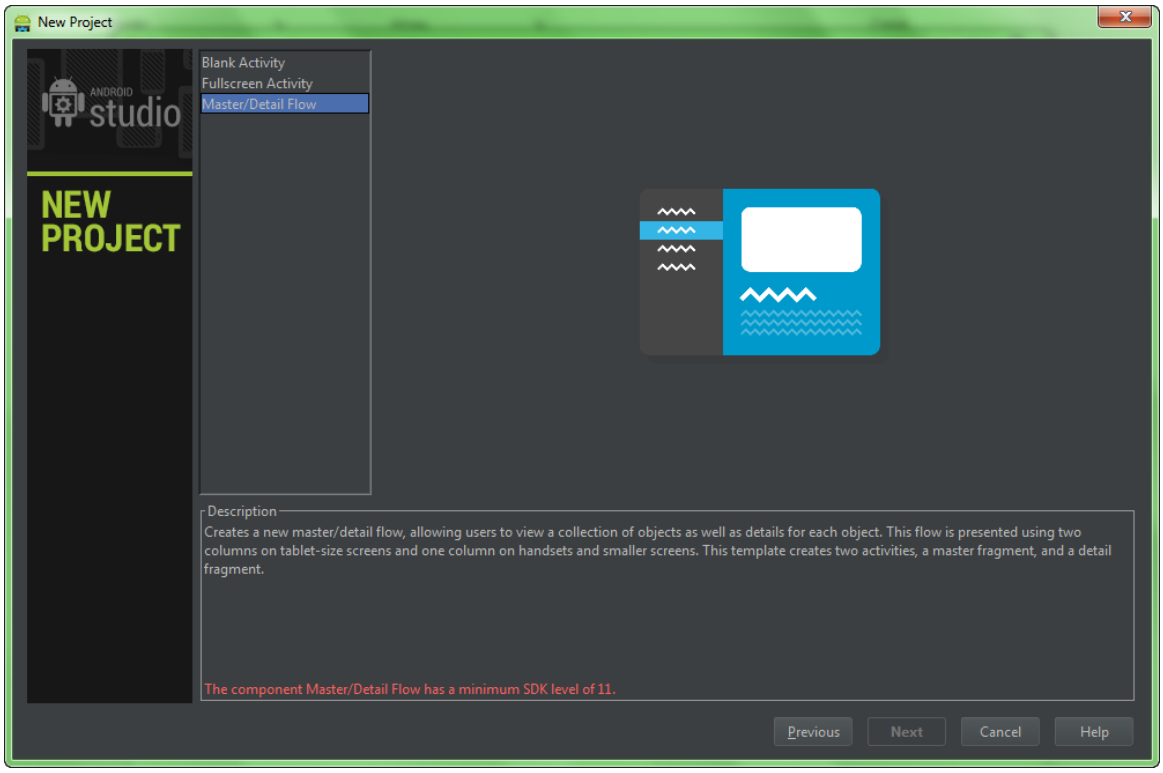


Рисунок 2.7 – Окно выбора базовой Activity

Имеется выбор из трёх вариантов:

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

Далее указываем имя нашей стартовой Activity и имя её layout'а как показано на рисунке 2.8, а также имя первого фрагмента, поскольку мы выбрали поддержку фрагментов при создании проекта. Также можно выбрать тип навигации по Activity.

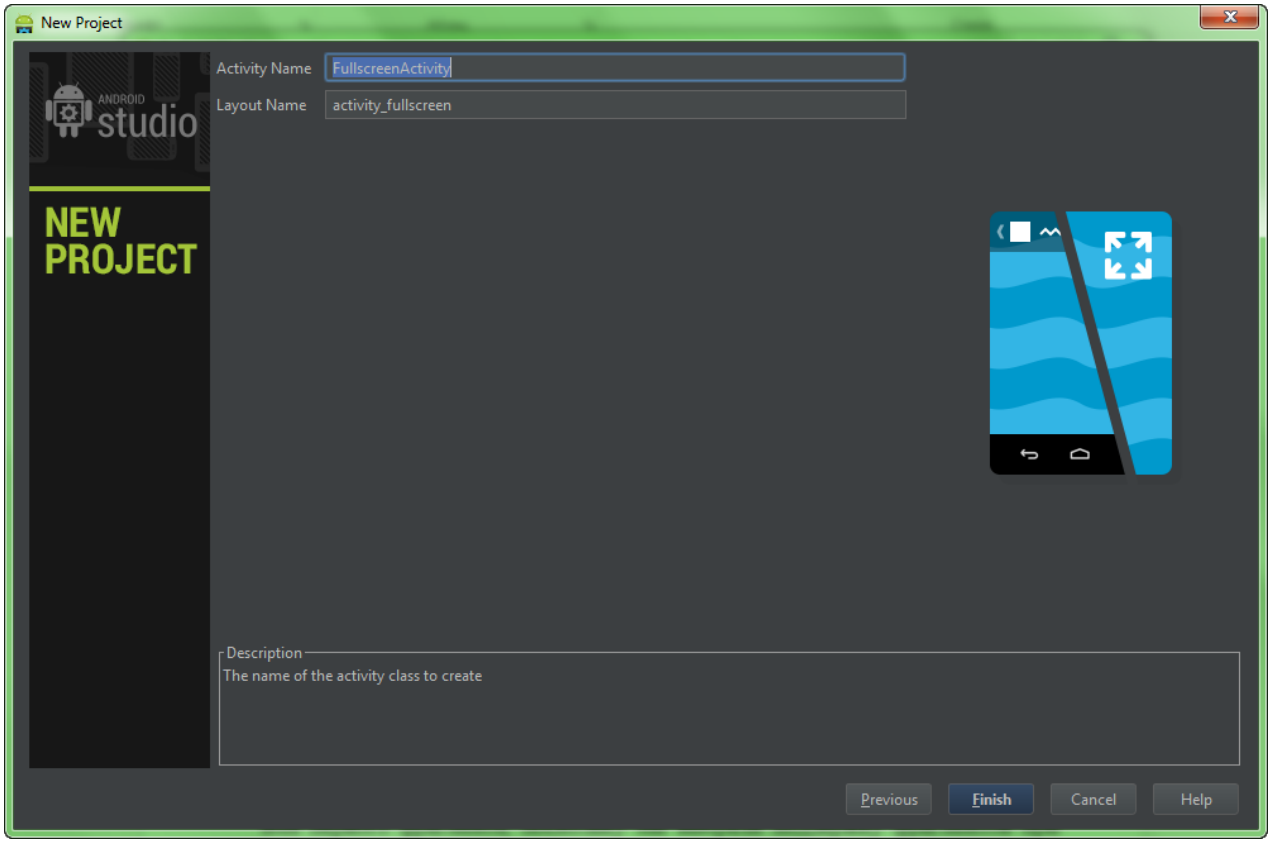


Рисунок 2.8 – Ввод имени нашей стартовой Activity и имени её layout'а

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

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

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

На рисунке 2.9 изображено основное рабочее окно среды разработки.

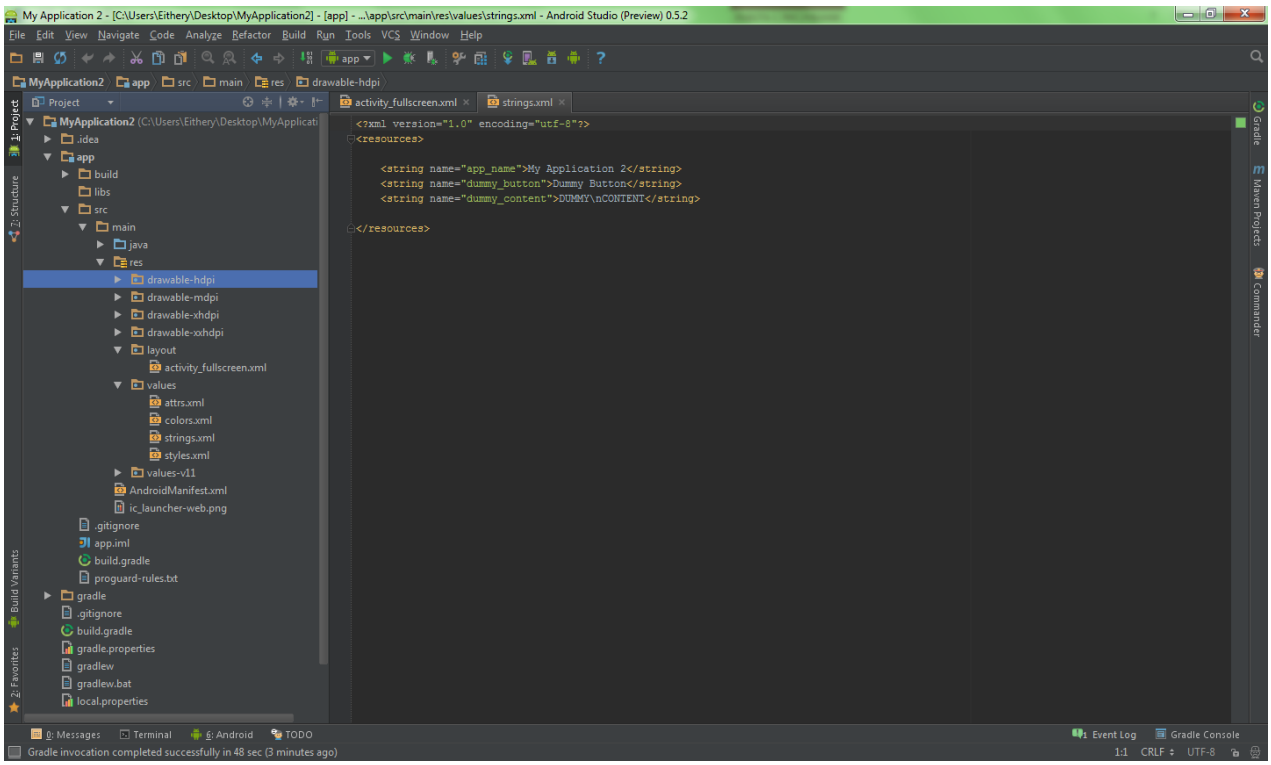


Рисунок 2.9 – Основное окно IDE

Xml редактор, изображённый на рисунке 2.10

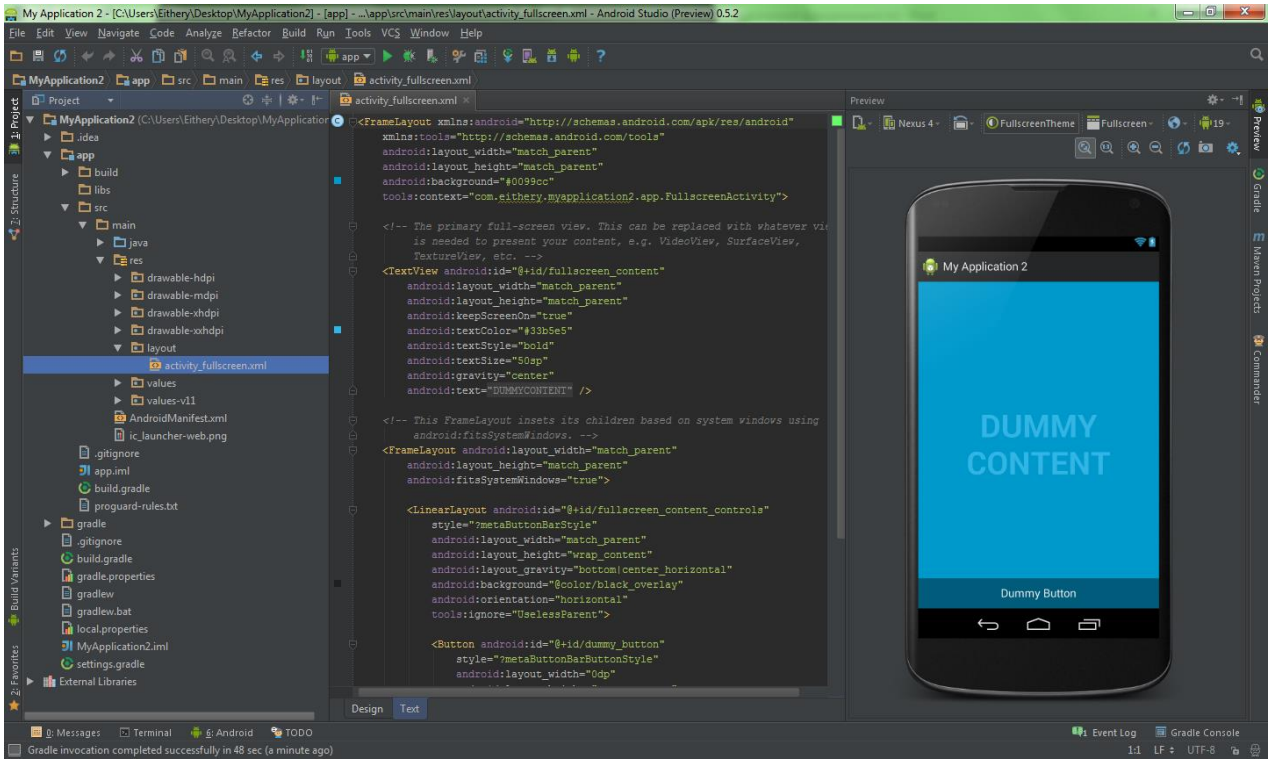


Рисунок 2.10 – Пример окна xml ректора

Рассмотрев рисунок 2.10 можно отметить следующее:

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

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

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

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

Улучшенная интеграция с Android компонентами.

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

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

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

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

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

– Package – пакет нашего приложения.

*Создание компонентов Android*

Рисунок 2.11 демонстрирует окно для создания нового компонента приложения.

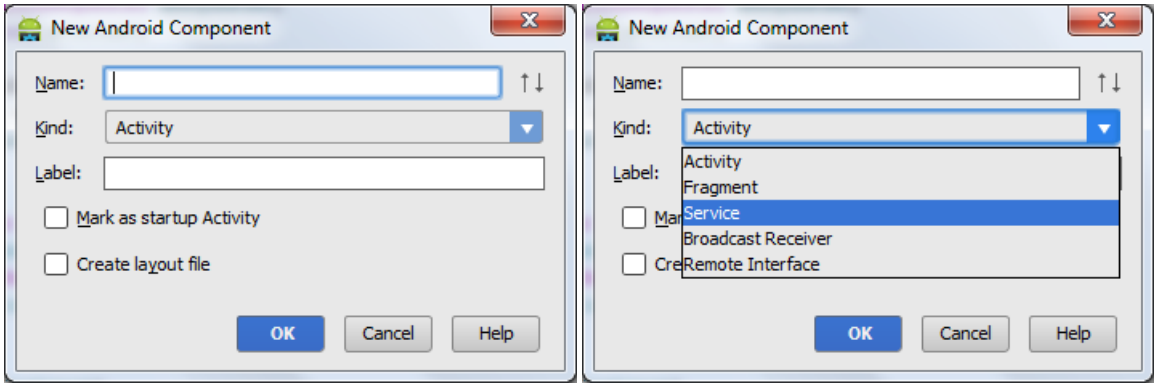


Рисунок 2.11 – Окно создания нового компонента

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

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

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

HTML File – собственно создает html файл.

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

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

– Более интеллектуальный анализ кода.

– Встроенное подключение Android Sources.

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

# 3 Этапы разработки приложения

## 3.1 Проектирование

SoundPro – это мультимедийное приложение, позволяющее создавать и редактировать собственные рингтоны прямо на телефоне. Так же имеется функция редактировать уже имеющиеся музыкальные файлы. Поддерживающиеся файлы должны иметь форматы WAV, MP3, AAC и 3GPP/AMR. Данное приложение имеет простой и удобный интерфейс. Внешний вид приложения должен сочетаться с остальными элементами оформления пользовательского интерфейса операционной системы, передовая задуманный разработчиками платформы опыт использования. Дизайн ПП должен строго следовать установленным нормам и требованиям платформ держателя (в данном случае Google).

Используя все вышеописанные нормы, разрабатывается концепция интерфейса программы SoundPro.

Основной интерфейс программы представлен в виде двух основных частей: главное меню и меню редактирования музыкальных файлов.

Интерфейс главного меню программы представлен пользователям экранным табло на котором размещен список музыкальных файлов находящихся на нашем смартфоне (Рисунок 3.1).

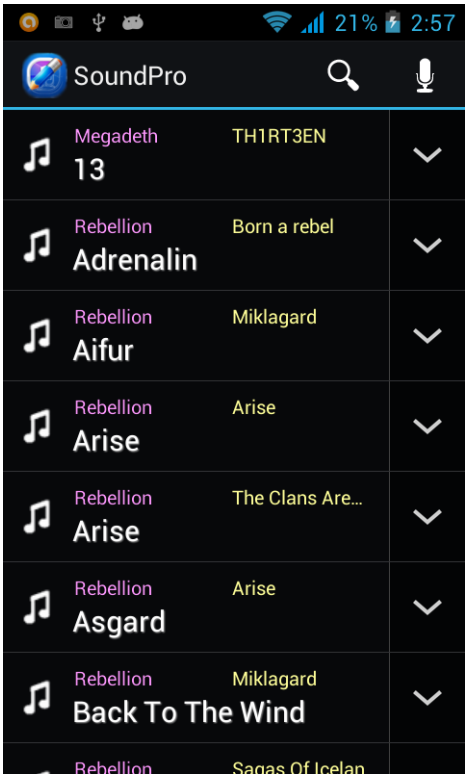


Рисунок 3.1 – Главное меню приложения

Имеется возможность быстрого поиска по альбому, исполнителю или названию музыкального файла, представленная на рисунке 3.2. Данная функция незаменима и удобна в современных программах, т.к. значительно уменьшает время на поиск нужного файла. Так же в SoundPro имеется функция записи собственного файла при помощи микрофона или гарнитуры для дальнейшего редактирования. Все управление в программе производится с помощью пальцев руки.

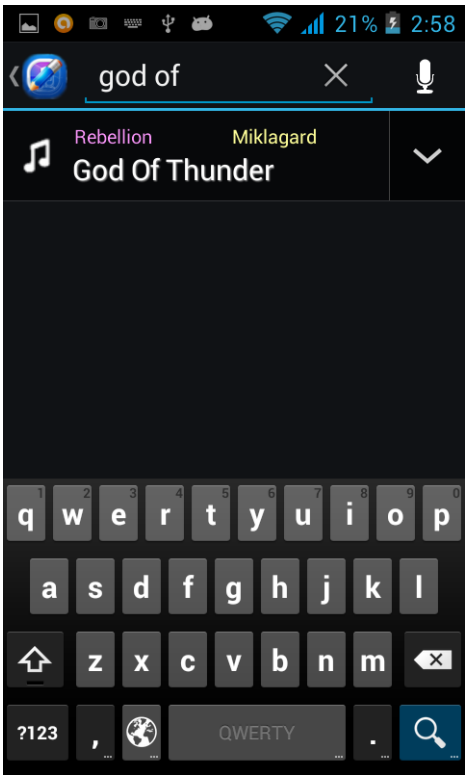


Рисунок 3.2 – Поиск файла по названию

Для начала редактирования звукового файла, необходимо нажать пальцем на его названии, после чего появится мню правки (Рисунок 3.3), в котором, собственно, и происходит весь процесс. При помощи пользовательского интерфейса можно устанавливать точки начала и конца создаваемого рингтона. Так же при редактировании имеется функция прослушивания как обрезаемого кусочка, так и полного музыкального файла. Для этого необходимо нажать на кнопку «Play» или просто коснуться пальцем по музыкальной дорожке. В интерфейсе есть множество вспомогательных функций, которые значительно облегчают работу в данной программе. Показатели начального и конечного маркеров. Сброс – позволяет сбросить выбираемые вами маркеры начала и конца обрезаемого файла на стандартные настройки (0:00 – 0:15).

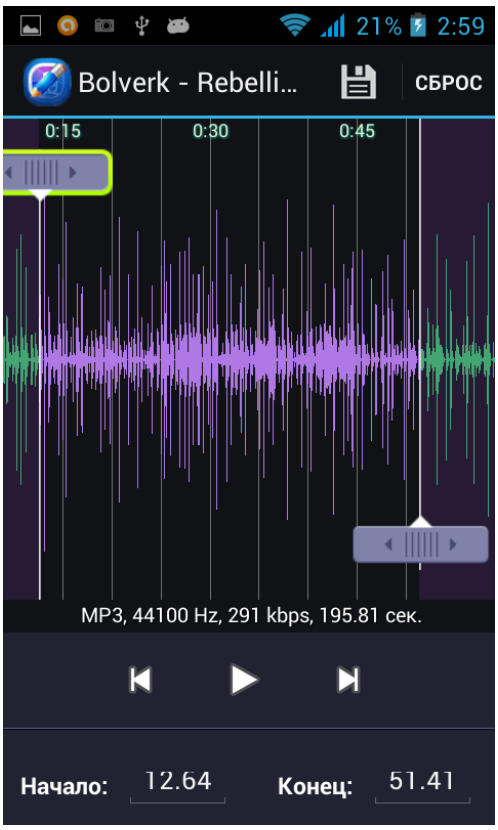


Рисунок 3.3 – Меню редактирования файла

После редактирования звукового файла его можно сохранить в память смартфона, нажав на функциональную кнопку в виде дискеты.

При сохранении нового файла его можно установить как музыку, рингтон, будильник или напоминание (Рисунок 3.4).

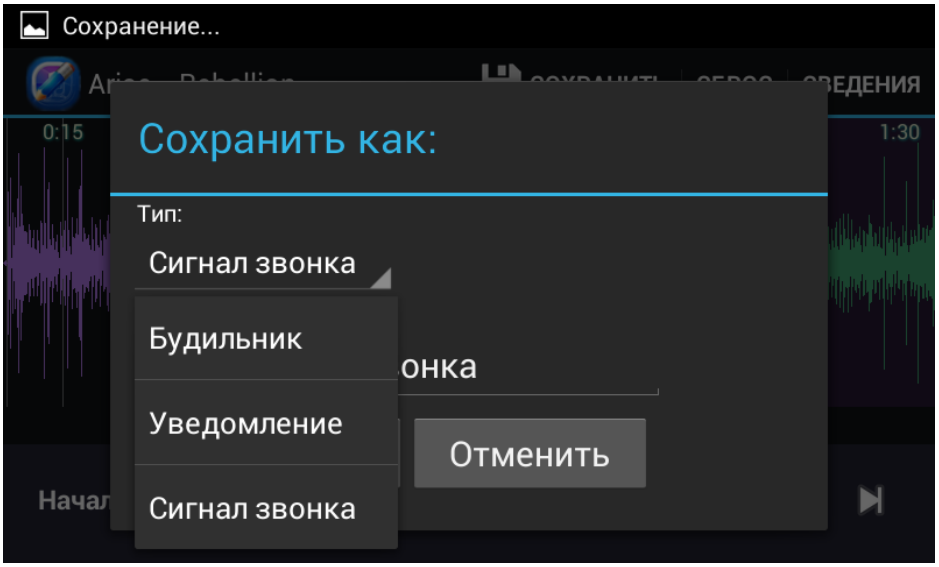


Рисунок 3.4 – Меню сохранения файла

В разделе «О программе» находятся подсказки для пользователя, а так же информация о создателях приложения и версии программы (Рисунок 3.5).

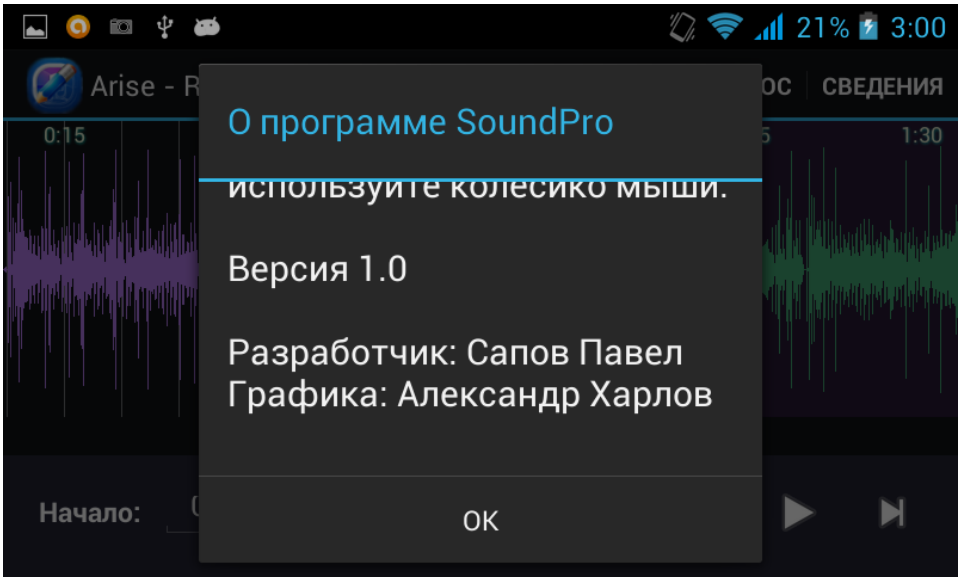


Рисунок 3.5 – О программе

### 3.2 Разработка приложения

Проект состоит из нескольких папок и файлов. Основные из них:

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

– *gen* – файлы сгенерированные самой Java;

– *res* – файлы ресурсов. Содержит несколько подкаталогов:

1 res/drawable-*dpi – в этих пяти 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. Также есть схожие с ним папки values-w820dp, values-v11 (для планшетов Android 3.0), values-14 (для Android 4) предназначенные для определенных видов устройств.

При разработке приложения были нарисованы отдельные компоненты приложения в программе Adobe Illustrator CC:

– иконка приложения;

– маркеры начала и конца обрезанного файла;

– иконки кнопок справки, сброса, сохранения, добавления нового файла, записи файла и т.д.

### 3.2.1 Создание и реализация форм

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

```
     <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
         android:layout_width="fill_parent"
         android:layout_height="fill_parent"
         android:orientation="vertical">

       <LinearLayout
         style="@style/ChooseContactBackground"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:gravity="center_vertical">

         <ImageView
          android:src="@drawable/search"
          android:layout_marginLeft="8dip"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content" />

         <EditText android:id="@+id/search_filter"
          android:textSize="18sp"
          android:layout_margin="8dip"
          android:layout_width="0px"
          android:layout_height="wrap_content"
          android:layout_weight="1">

         </EditText>

       </LinearLayout>

       <ListView android:id="@+id/android:list"
          android:layout_width="fill_parent"
          android:layout_height="0px"
          android:layout_weight="1">
          <requestFocus />
       </ListView>

     </LinearLayout>
```

Так же при создании интерфейса использовался графический редактор дизайна. Пример созданного файла в графическом редакторе изображён на рисунке 3.6.

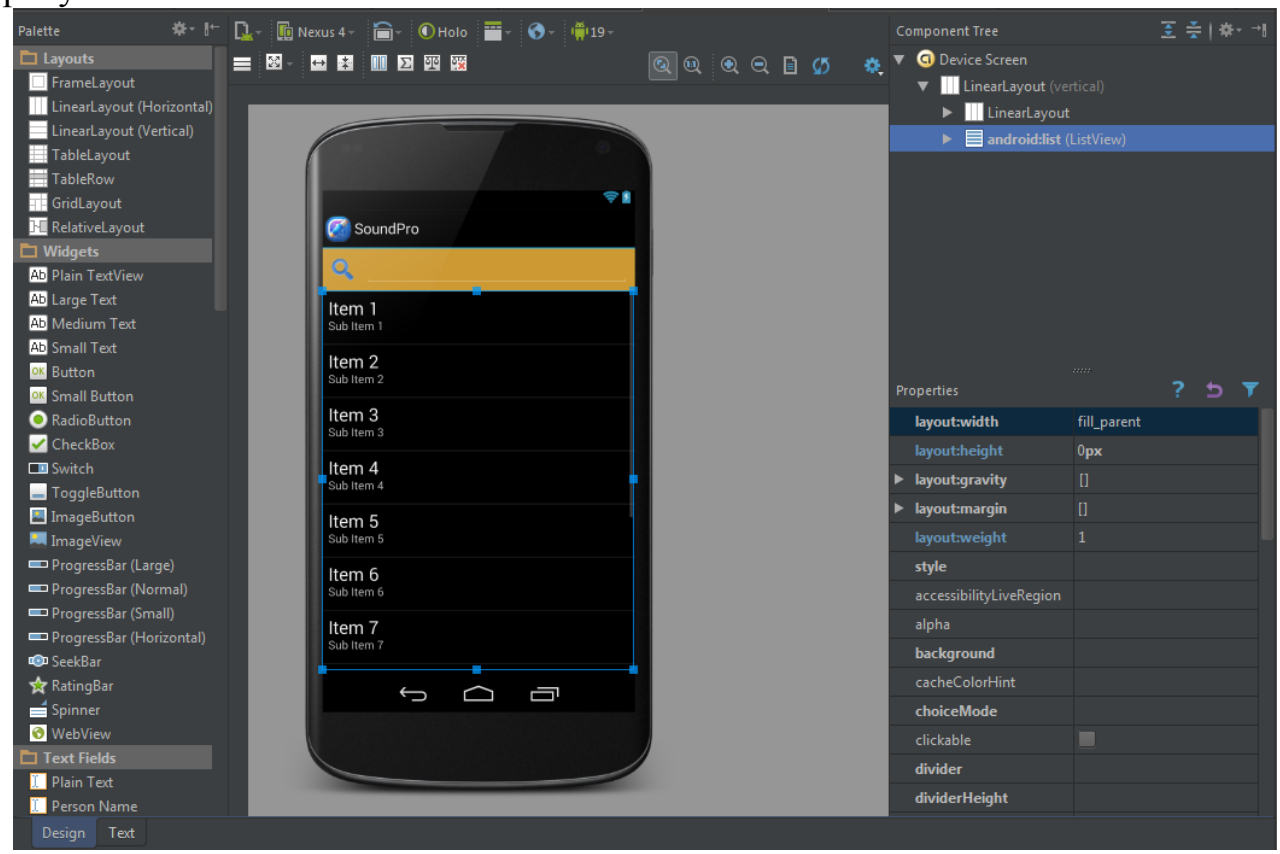


Рисунок 3.6 – Графическое изображение меню выбора файла

Форма отображения каждого музыкального файла реализовывалось в файле media_select_row.xml.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

      <ImageView android:id="@+id/row_icon"
         android:gravity="center_vertical"
         android:scaleType="center"
         android:layout_width="40dip"
         android:layout_height="fill_parent"
         android:layout_marginLeft="4dip"/>

        <LinearLayout
            android:orientation="vertical"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dip"
```

```
        android:layout_marginBottom="8dip"
        android:layout_width="0px"
        android:layout_weight="1"
        android:layout_marginLeft="4dip"
        android:layout_marginRight="4dip">

         <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="wrap_content">

          <TextView android:id="@+id/row_artist"
             android:textColor="#ff99ff"
             android:textSize="12sp"
             android:singleLine="true"
             android:layout_marginLeft="1dip"
             android:layout_width="0px"
             android:layout_height="wrap_content"
             android:layout_weight="1"/>

          <TextView android:id="@+id/row_album"
             android:textColor="#ffff99"
             android:textSize="12sp"
             android:singleLine="true"
             android:layout_marginLeft="8dip"
             android:layout_width="0px"
             android:layout_height="wrap_content"
             android:layout_weight="1"/>

        </LinearLayout>

        <TextView android:id="@+id/row_title"
            android:textColor="#ffffff"
            android:textSize="18sp"
            android:singleLine="true"
            android:shadowColor="#999999"
            android:shadowDx="1"
            android:shadowDy="1"
            android:shadowRadius="1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"/>

     </LinearLayout>
    <LinearLayout style="@style/VerticalDividerForList" />

    <ImageView android:id="@+id/row_options_button"
        android:src="@drawable/navigation_expand"
        android:background="@drawable/button_options_bkgnd"
        android:gravity="center_vertical"
        android:scaleType="center"
        android:layout_height="fill_parent"
        android:layout_width="52dip"
```

```
            android:layout_gravity="center_vertical"
     android:contentDescription="@string/button_options"/>
     </LinearLayout>
```

Создание форм меню обработки файла.

```
     <?xml version="1.0" encoding="utf-8"?>

     <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
         android:layout_width="fill_parent"
         android:layout_height="fill_parent"
         android:orientation="vertical">

         <AbsoluteLayout
            android:layout_width="wrap_content"
            android:layout_height="0px"
            android:layout_weight="1">

           <!-- ImageView android:id="@+id/bkgnd"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:scaleType="centerCrop"
            android:src="@drawable/bkgnd" -->

           <com.soundpro.WaveformView android:id="@+id/waveform"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" />

           <com.soundpro.MarkerView android:id="@+id/startmarker"
             android:nextFocusDown="@+id/endmarker"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:src="@drawable/marker_left"
             android:contentDescription="@string/start_marker" />

           <com.soundpro.MarkerView android:id="@+id/endmarker"
             android:nextFocusUp="@+id/startmarker"
             android:nextFocusDown="@+id/info"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:src="@drawable/marker_right"
             android:contentDescription="@string/end_marker" />

         </AbsoluteLayout>

         <TextView android:id="@+id/info"
            android:nextFocusUp="@+id/endmarker"
            android:nextFocusDown="@+id/play"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
```

```
    style="@style/AudioFileInfoOverlayText"
    android:gravity="center" />

<LinearLayout
 style="@style/ToolbarBackground"
 android:layout_width="fill_parent"
 android:layout_height="62dip"
 android:gravity="center" >

  <ImageButton android:id="@+id/rew"
   android:layout_width="64dip"
   android:layout_height="52dip"
   android:layout_marginTop="6dip"
   android:layout_marginBottom="6dip"
   style="@android:style/MediaButton"
   android:contentDescription="@string/rewind"
   android:src="@android:drawable/ic_media_previous" />

  <ImageButton android:id="@+id/play"
   android:layout_width="71dip"
   android:layout_height="52dip"
   android:layout_marginTop="6dip"
   android:layout_marginBottom="6dip"
   style="@android:style/MediaButton"
   android:contentDescription="@string/play"
   android:src="@android:drawable/ic_media_play" />

  <ImageButton android:id="@+id/ffwd"
   android:layout_width="64dip"
   android:layout_height="52dip"
   android:layout_marginRight="5dip"
   android:layout_marginTop="6dip"
   android:layout_marginBottom="6dip"
   style="@android:style/MediaButton"
   android:contentDescription="@string/ffwd"
   android:src="@android:drawable/ic_media_next" />

</LinearLayout>

<LinearLayout style="@style/HorizontalDividerTop" />
<LinearLayout style="@style/HorizontalDividerBottom" />

<LinearLayout
 style="@style/ToolbarBackground"
 android:layout_width="fill_parent"
 android:layout_height="62dip"
 android:gravity="center_vertical" >

  <LinearLayout
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:layout_marginTop="6dip"
```

```
                 android:gravity="center"
                 android:orientation="horizontal">

                <TextView
                   android:id="@+id/mark_start"
                   android:text="@string/start_label"
                   android:textColor="#ffffffff"
                   android:textSize="14sp"
                   android:textStyle="bold"
                   android:clickable="true"
                   android:gravity="center"
                   android:layout_width="wrap_content"
                   android:layout_height="31dip" />

                <EditText android:id="@+id/starttext"
                   android:textSize="16sp"
                   android:inputType="number|numberDecimal"
                android:layout_marginLeft="10dip"
                android:layout_marginRight="30dip"
                   android:layout_width="70dip"
                   android:layout_height="31dip"
                   android:gravity="center"
                   android:contentDescription="@string/start_label"
/>

                <TextView
                   android:id="@+id/mark_end"
                   android:text="@string/end_label"
                   android:textColor="#ffffffff"
                   android:textSize="14sp"
                   android:textStyle="bold"
                   android:clickable="true"
                   android:gravity="center"
                   android:layout_width="wrap_content"
                   android:layout_height="31dip" />

                <EditText android:id="@+id/endtext"
                   android:textSize="16sp"
                   android:inputType="number|numberDecimal"
                   android:layout_width="70dip"
                   android:layout_height="31dip"
                android:layout_marginLeft="10dip"
                   android:gravity="center"
                   android:contentDescription="@string/end_label" />

             </LinearLayout>

          </LinearLayout>

       </LinearLayout>
```

Создание форм для сохранения файла.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="300sp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dip"
        android:layout_marginRight="10dip"
        android:orientation="vertical">

        <TextView
           android:text="@string/ringtone_type_label"
           android:textColor="#ffffffff"
           android:textSize="12sp"
           android:layout_marginLeft="15dip"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />

        <Spinner android:id="@+id/ringtone_type"
           android:layout_marginLeft="10dip"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />

        <TextView
           android:text="@string/ringtone_name_label"
           android:textColor="#ffffffff"
           android:textSize="12sp"
           android:layout_marginLeft="15dip"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content" />

        <EditText android:id="@+id/filename"
           android:layout_marginLeft="10dip"
           android:layout_width="280sp"
           android:layout_height="wrap_content" />

        <LinearLayout
           android:layout_marginTop="5dip"
           android:layout_marginLeft="10dip"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:gravity="center_horizontal"
            android:weightSum="1">

          <Button android:id="@+id/save"
             android:text="@string/file_save_button_save"
             android:layout_width="100sp"
             android:layout_height="wrap_content"
             android:layout_marginBottom="10dip"
              android:layout_weight="0.31" />
```

```
      <Button android:id="@+id/cancel"
         android:text="@string/file_save_button_cancel"
         android:layout_width="100sp"
         android:layout_height="wrap_content"
         android:layout_marginBottom="10dip"
          android:layout_weight="0.31" />

    </LinearLayout>

</LinearLayout>
```

Добавление форм интерфейса в файле styles.xml.

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

  <style name="AudioFileInfoOverlayText">
    <item name="android:paddingLeft">4px</item>
    <item name="android:paddingBottom">4px</item>
    <item name="android:textColor">#ffffffff</item>
    <item name="android:textSize">12sp</item>
    <item name="android:shadowColor">#000000</item>
    <item name="android:shadowDx">1</item>
    <item name="android:shadowDy">1</item>
    <item name="android:shadowRadius">1</item>
    <item name="android:focusable">true</item>
  </style>

  <style name="ToolbarBackground">
    <item name="android:background">#222233</item>
  </style>

  <style name="ChooseContactBackground">
    <item name="android:background">#cc9933</item>
  </style>

  <style name="VerticalDividerForList">
    <item name="android:layout_width">1px</item>
    <item name="android:layout_height">fill_parent</item>
    <item name="android:layout_marginLeft">4px</item>
    <item name="android:background">#444444</item>
  </style>

  <style name="VerticalDividerLeft">
    <item name="android:layout_width">1px</item>
    <item name="android:layout_height">fill_parent</item>
    <item name="android:layout_marginLeft">4px</item>
    <item name="android:background">#666677</item>
  </style>
  <style name="VerticalDividerRight">
```

```
    <item name="android:layout_width">1px</item>
    <item name="android:layout_height">fill_parent</item>
    <item name="android:layout_marginRight">4px</item>
    <item name="android:background">#000022</item>
  </style>

  <style name="HorizontalDividerTop">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">1px</item>
    <item name="android:background">#666677</item>
  </style>
  <style name="HorizontalDividerBottom">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">1px</item>
    <item name="android:background">#000022</item>
  </style>

</resources>
```

### 3.2.2 Создание и реализация классов

В процессе разработки были созданы и реализованы множество классов:

- AfterSaveActionDialog – подтверждение сохранения файла;
- FileSaveDialog – сохранение файла;
- MarkerView – реализация маркеров;
- SoundProEditActivity – редактирование файла;
- SoundProSelectActivity – выбор файла;
- SongMetadataReader – чтение музыкальных файлов;
- WaveformView – реализация формы сигнала файла и несколько незначительных классов.

Несколько примеров реализации классов:

– Класс сохранения файла FileSaveDialog.java.

```
package com.soundpro;

import android.app.Dialog;
import android.content.Context;
import android.content.res.Resources;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
```

```
import java.util.ArrayList;
import java.util.HashMap;

public class FileSaveDialog extends Dialog {

    // File kinds - these should correspond to the order in
which
    // they're presented in the spinner control
    public static final int FILE_KIND_MUSIC = 0;
    public static final int FILE_KIND_ALARM = 1;
    public static final int FILE_KIND_NOTIFICATION = 2;
    public static final int FILE_KIND_RINGTONE = 3;

    private Spinner mTypeSpinner;
    private EditText mFilename;
    private Message mResponse;
    private String mOriginalName;
    private ArrayList<String> mTypeArray;
    private int mPreviousSelection;

    /**
     * Return a human-readable name for a kind (music, alarm,
ringtone, ...).
     * These won't be displayed on-screen (just in logs) so
they shouldn't
     * be translated.
     */
    public static String KindToName(int kind) {
        switch(kind) {
        default:
            return "Unknown";
        case FILE_KIND_MUSIC:
            return "Music";
        case FILE_KIND_ALARM:
            return "Alarm";
        case FILE_KIND_NOTIFICATION:
            return "Notification";
        case FILE_KIND_RINGTONE:
            return "Ringtone";
        }
    }

    public FileSaveDialog(Context context,
                          Resources resources,
                          String originalName,
                          Message response) {
        super(context);

        // Inflate our UI from its XML layout description.
        setContentView(R.layout.file_save);
```

```
setTitle(resources.getString(R.string.file_save_title));

            mTypeArray = new ArrayList<String>();

mTypeArray.add(resources.getString(R.string.type_music));

mTypeArray.add(resources.getString(R.string.type_alarm));

mTypeArray.add(resources.getString(R.string.type_notification));

mTypeArray.add(resources.getString(R.string.type_ringtone));

            mFilename = (EditText)findViewById(R.id.filename);
            mOriginalName = originalName;

            ArrayAdapter<String>        adapter        =        new
ArrayAdapter<String>(
                context,     android.R.layout.simple_spinner_item,
mTypeArray);
            adapter.setDropDownViewResource(
                android.R.layout.simple_spinner_dropdown_item);
            mTypeSpinner                =                (Spinner)
findViewById(R.id.ringtone_type);
            mTypeSpinner.setAdapter(adapter);
            mTypeSpinner.setSelection(FILE_KIND_RINGTONE);
            mPreviousSelection = FILE_KIND_RINGTONE;

            setFilenameEditBoxFromName(false);

            mTypeSpinner.setOnItemSelectedListener(new
OnItemSelectedListener() {
                    public     void     onItemSelected(AdapterView
parent,
                                                View v,
                                                int position,
                                                long id) {
                        setFilenameEditBoxFromName(true);
                    }
                    public    void    onNothingSelected(AdapterView
parent) {
                    }
                });

            Button save = (Button)findViewById(R.id.save);
            save.setOnClickListener(saveListener);
            Button cancel = (Button)findViewById(R.id.cancel);
            cancel.setOnClickListener(cancelListener);
            mResponse = response;
        }
```

```
        private      void      setFilenameEditBoxFromName(boolean
onlyIfNotEdited) {
            if (onlyIfNotEdited) {
                CharSequence currentText = mFilename.getText();
                String expectedText = mOriginalName + " " +
                    mTypeArray.get(mPreviousSelection);

                if (!expectedText.contentEquals(currentText)) {
                    return;
                }
            }

            int                  newSelection                  =
mTypeSpinner.getSelectedItemPosition();
            String newSuffix = mTypeArray.get(newSelection);
            mFilename.setText(mOriginalName + " " + newSuffix);
            mPreviousSelection                                 =
mTypeSpinner.getSelectedItemPosition();
        }

        private   View.OnClickListener   saveListener   =   new
View.OnClickListener() {
                public void onClick(View view) {
                    mResponse.obj = mFilename.getText();
                    mResponse.arg1                             =
mTypeSpinner.getSelectedItemPosition();
                    mResponse.sendToTarget();
                    dismiss();
                }
            };

        private   View.OnClickListener   cancelListener   =   new
View.OnClickListener() {
                public void onClick(View view) {
                    dismiss();
                }
            };
    }
```

– Создание класса для чтения музыкальных файлов SongMetadataReader.java.

```
package com.soundpro;

import android.app.Activity;
import android.database.Cursor;
import android.net.Uri;
import android.provider.MediaStore;

import java.util.HashMap;
```

```
     public class SongMetadataReader {
         public            Uri            GENRES_URI            =
MediaStore.Audio.Genres.EXTERNAL_CONTENT_URI;
         public Activity mActivity = null;
         public String mFilename = "";
         public String mTitle = "";
         public String mArtist = "";
         public String mAlbum = "";
         public String mGenre = "";
         public int mYear = -1;

         SongMetadataReader(Activity activity, String filename) {
             mActivity = activity;
             mFilename = filename;
             mTitle = getBasename(filename);
             try {
                 ReadMetadata();
             } catch (Exception e) {
             }
         }

         private void ReadMetadata() {
             String GENRE_ID = MediaStore.Audio.Genres._ID;
             String GENRE_NAME = MediaStore.Audio.Genres.NAME;
             String AUDIO_ID = MediaStore.Audio.Media._ID;

             // Get a map from genre ids to names
             HashMap<String,    String>    genreIdMap    =    new
HashMap<String, String>();
             Cursor c = mActivity.managedQuery(
                 GENRES_URI,
                 new String[] { GENRE_ID, GENRE_NAME },
                 null, null, null);
             for        (c.moveToFirst();        !c.isAfterLast();
c.moveToNext()) {
                 genreIdMap.put(c.getString(0), c.getString(1));
             }
             mGenre = "";
             for (String genreId : genreIdMap.keySet()) {
                 c = mActivity.managedQuery(
                     makeGenreUri(genreId),
                     new String[] { MediaStore.Audio.Media.DATA },
                     MediaStore.Audio.Media.DATA  +  "  LIKE  \""  +
mFilename + "\"",
                     null, null);
                 if (c.getCount() != 0) {
                     mGenre = genreIdMap.get(genreId);
                     break;
                 }
                 c = null;
             }
```

```
            Uri                         uri                         =
MediaStore.Audio.Media.getContentUriForPath(mFilename);
            c = mActivity.managedQuery(
                uri,
                new String[] {
                    MediaStore.Audio.Media._ID,
                    MediaStore.Audio.Media.TITLE,
                    MediaStore.Audio.Media.ARTIST,
                    MediaStore.Audio.Media.ALBUM,
                    MediaStore.Audio.Media.YEAR,
                    MediaStore.Audio.Media.DATA },
                MediaStore.Audio.Media.DATA   +   "   LIKE   \""   +
mFilename + "\"",
                null, null);
            if (c.getCount() == 0) {
                mTitle = getBasename(mFilename);
                mArtist = "";
                mAlbum = "";
                mYear = -1;
                return;
            }
            c.moveToFirst();
            mTitle            =            getStringFromColumn(c,
MediaStore.Audio.Media.TITLE);
            if (mTitle == null || mTitle.length() == 0) {
                mTitle = getBasename(mFilename);
            }
            mArtist            =            getStringFromColumn(c,
MediaStore.Audio.Media.ARTIST);
            mAlbum            =            getStringFromColumn(c,
MediaStore.Audio.Media.ALBUM);
            mYear            =            getIntegerFromColumn(c,
MediaStore.Audio.Media.YEAR);
        }

        private Uri makeGenreUri(String genreId) {
            String                  CONTENTDIR                  =
MediaStore.Audio.Genres.Members.CONTENT_DIRECTORY;
            return Uri.parse(
                new StringBuilder()
                .append(GENRES_URI.toString())
                .append("/")
                .append(genreId)
                .append("/")
                .append(CONTENTDIR)
                .toString());
        }

        private   String   getStringFromColumn(Cursor   c,   String
columnName) {
            int index = c.getColumnIndexOrThrow(columnName);
```

```
            String value = c.getString(index);
            if (value != null && value.length() > 0) {
                return value;
            } else {
                return null;
            }
        }

        private   int   getIntegerFromColumn(Cursor   c,   String
columnName) {
            int index = c.getColumnIndexOrThrow(columnName);
            Integer value = c.getInt(index);
            if (value != null) {
                return value;
            } else {
                return -1;
            }
        }

        private String getBasename(String filename) {
            return  filename.substring(filename.lastIndexOf('/')  +
1,
                                        filename.lastIndexOf('.'));
        }
    }
```

## 4 Технико-экономическое обоснование

### 4.1 Цель проекта

Целью данного дипломного проекта является создание мобильного приложения MusicPro для операционной системы Android, являющегося мультимедийным приложением для создания и редактирования музыки, соответствующего всем «гайдлайнам» Android OS. Разработка приложения производится в среде Android Studio на языке Java. Графические элементы приложения создаются в Adobe Illustrator CC.

### 4.2 Трудовые ресурсы, используемые в работе

В работе над данным проектом задействованы:
- дизайнер – создание интерфейса приложения;
- программист – разработка алгоритмов и программирование.

Количество сотрудников, задействованных в проекте, и их месячная заработная плата представлено 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 с учетом НДС, тенге |
|---|---|---|---|
| Ноутбук HP Pavilion 15-e082er | Intel Core i5 3230M, 6 Gb DDR3, 1000 Gb HDD, Radeon HD 7670M | 1 | 112 810 |

*Продолжение таблицы 4.2*

| | | | |
|---|---|---|---|
| Ноутбук HP Pavilion 15-n060er | Intel Core i5-4200U, 8 Gb DDR3, 1000 Gb HDD, GeForce GT 740M | 1 | 126 110 |
| Смартфон Prestigio Multiphone 4055 DUO | 4 Gb ROM, 512 Mb RAM, MediaTek MT6577T | 1 | 33 810 |
| Итого | | 3 | 272730 |

### 4.4 Программное обеспечение, используемое в работе

При разработке приложения используется следующее программное обеспечение:

– Windows 7 – операционная система;
– Adobe Illustrator CC – векторный графический редактор;
– 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 с учетом НДС, тенге |
|---|---|---|
| Windows 7 | 2 | Предустановлено, цена ПО входит в стоимость ноутбука |
| Adobe Illustrator CC | 1 | 37 593 (Годовая подписка) |
| Android Studio | 1 | Бесплатно |
| Java Development Kit | 1 | Бесплатно |
| Итого | | 37 593 |

### 4.5 Сроки реализации проекта

Процесс проектирования и разработки приложения включает в себя 5 этапов:

– постановка задачи и сбор данных;
– разработка дизайна интерфейса приложения;
– разработка форм приложения;
– тестирование на работоспособность;

– оформление и сдача отчета.

В таблице 4.4 приведен график реализации проекта.

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 «гайдлайнов» | | | | | | | | |
| 2 этап | Разработка интерфейса | | | | | | | | |
| | Создание необходимых графических элементов | | | | | | | | |
| 3 этап | Разработка форм приложения | | | | | | | | |
| 4 этап | Тестирование приложения | | | | | | | | |
| | Отладка приложения | | | | | | | | |
| 5 этап | Оформление и сдача отчета | | | | | | | | |

## 4.6 Расчет затрат и стоимости работ по реализации проекта

Разработка мобильного приложения требует большого количества интеллектуальных затрат сотрудников, выполняющих работу, а также необходимых технических средств для ее реализации. Все это требует финансовых вложений, на основе которых высчитывается конечная стоимость проекта.

Затраты на разработку данного приложения вычисляются по формуле

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

где ФОТ – фонд оплаты труда;

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

### 4.6.1 Расчет фонда оплаты труда

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

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

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

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

Среднедневной заработок

$$D = \frac{ЗПм}{Др} \quad (4.3)$$

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

Дизайнер

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

Программист

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

Заработная плата за один час работы сотрудника рассчитывается по формуле

$$Н = \frac{D}{Чр} \quad (4.4)$$

где D – средний дневной заработок работника;

Чр – количество часов рабочего дня (8 часов).

Дизайнер

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

Программист

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

Длительность цикла в днях по каждому виду работ определяется по формуле

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

где Т – 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.

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

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

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

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

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

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

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

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

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

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

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

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

Таблица 4.5 – Сводные результаты расчета затрат на основную заработную плату

| Наименование этапов работ | Исполнитель | Трудоемкость | | Длительность цикла, дни | 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 «гайдлайнов» | 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

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

$З_{доп} = 137128 * 0,1 = 13712,8$ тенге

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

$ФОТ = 137128 + 13712,8 = 150840,8$ тенге

**4.6.2 Расчет затрат по социальному налогу**

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

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

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

$$ПО = ФОТ * 10\% \quad (4.8)$$

$ПО = 150840,8 * 0,1 = 15084,08$ тенге.

Таким образом социальный налог составит

$С_н = (150840,8 - 15084,08) * 0,11 = 14933,23$ тенге

**4.6.3 Расчет амортизационных отчислений**

Амортизационные отчисления рассчитываются по формуле

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

где $H_A$ – норма амортизации;

$C_{пер}$- 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

$$A_1 = \frac{40*272730*40}{100*12*21} = 17316 \text{ тенге}$$

На программное обеспечение

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

Суммарные затраты на амортизацию рассчитываются по формуле

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

A = 17316 + 2387 = 19703 тенге

### 4.6.4 Расчет затрат на электроэнергию

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

Для расчета затрат на электроэнергию используется формула 4.11.

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

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

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

Расходы электроэнергии на оборудование рассчитываются по формуле 4.12.

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

где W – потребляемая оборудованием мощность, кВт;

T – время работы, часы;

S – тариф (1кВт/час = 16,9 тенге);

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

Потребляемая мощность HP Pavilion 15-e082er составляет 90 Вт.

Потребляемая мощность HP Pavilion 15-n060er составляет 65 Вт.

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

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

Таким образом общая сумма затрат на электроэнергию для оборудования:

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

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

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

И составляют

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

Суммарные затраты на электроэнергию составляют

Э = 754,4 + 37,7 = 792,1 тенге

**4.6.5 Расчет накладных расходов**

Накладные расходы рассчитываются в размере 50% от всех затрат.

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

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

**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 |

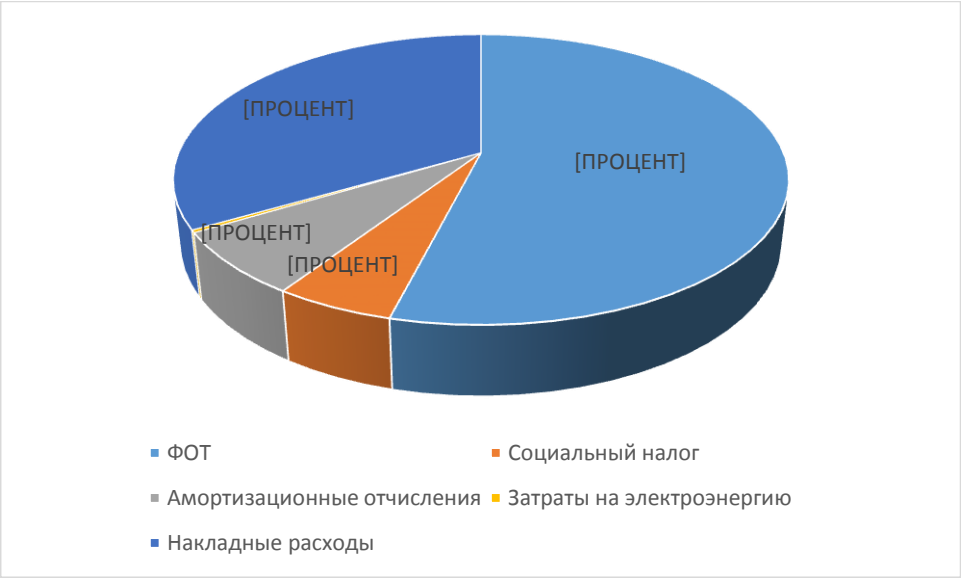


Рисунок ы4.1 – Затраты на разработку приложения

### 4.6.7 Цена реализации проекта

Цена проекта складывается из себестоимости и желаемого чистого дохода.

$$Ц_п = С + П \tag{4.15}$$

Где С – стоимость приложения;

П – чистый доход.

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

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

где Р – рентабельность.

$$Ц_п = 279403{,}69 * \left(1 + \frac{25}{100}\right) = 349254{,}61 \text{ тенге}$$

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

$$Ц_р = Ц_п + НДС \tag{4.17}$$

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

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

**Выводы**

Итоговая стоимость разработки приложения SoundPro для Android OS составила 573126,84 тенге, в которую заложены все возможные затраты при разработки программного продукта.

Наибольшую долю в стоимости проекта составляют затраты на оплату труда сотрудников, более 50%, задействованных в процессе данной разработки.

Разработка мобильного приложения являет дорогим проектом, требующих больших интеллектуальных и финансовых затрат. Однако все вложенные средства легко окупаются при условии высокого качества продукта и активной рекламной кампании, позволяя получить прибыль до 500% и более по истечению определенного времени.

## 5 Безопасность жизнедеятельности

### 5.1 Характеристика помещения и факторы, действующие на разработчика в процессе труда. Рабочее место

Помещение, в котором находится рабочее место разработчика, имеет следующие характеристики:

- длина помещения – 6 метров;
- ширина помещения – 2 метра;
- высота помещения – 3 метра;
- число окон – 1;
- число рабочих мест – 3;
- освещение: естественное (через боковое окно) и общее искусственное.

Рабочее место – это система функционально и пространственно-организованных технических средств и предметов труда, обеспечивающая благоприятные условия для успешного решения человеком-оператором поставленной перед ним задачи.

Рационально организованное рабочее место позволяет повысить производительность труда на 8-20% и минимизировать вредное воздействие компьютера на здоровье.

Развитию утомляемости на производстве способствует неправильная эргономическая организация рабочего места, нерациональные зоны размещения оборудования по высоте от пола, по фронту от оси симметрии и т.д., поэтому далее будут рассмотрены эргономические требования к рабочему месту.

Способы организации рабочего места зависят от характера решаемых задач, от используемого оборудования, от конкретной рабочей деятельности человека.

Габаритные и компоновочные параметры рабочего места определяются антропологическими характеристиками человека и нормированы в соответствующем документе – ГОСТ 21889-76.

#### 5.1.1 Правильная поза при работе с ПК

Большую часть своего рабочего времени разработчик проводит сидя. Такое положение позволяет минимизировать затраты энергии, максимально уменьшить утомление во время работы.

Данная поза наиболее оптимальна для продолжительной работы с учетом некоторых аспектов:

- спина наклонена на несколько градусов назад (такая поза позволяет разгрузить позвоночник);
- руки свободно опущены на подлокотники кресла;

– локти и запястья расслаблены, кисти имеют общую ось с предплечьями: не сгибаются и не разгибаются;

– ноги твердо стоят на полу или на специальной подставке.

### 5.1.2 Состав рабочего места разработчика

Рабочее место разработчика складывается из:

– пространства, занимаемого оборудованием;

– пространства необходимого для технического обслуживания или ремонта;

– зоны проходов, обеспечивающей нормальное функционирование оборудования;

– сенсомоторного пространства (части пространства рабочего места, в которой осуществляется двигательная и сенсорная работа человека).

Из необходимых для работы устройств:

– устройства отображения информации (монитор);

– стол;

– кресло;

– устройства ввода информации (мышь, клавиатура и т. д.);

– устройства вывода информации (принтер, плоттер).

### 5.1.3 Эргономические требования к рабочему месту

Зрительный комфорт, при работе с монитором, в основном определяется следующими факторами:

– размерами знаков;

– расстояние между знаками по горизонтали: 0,25 высоты знака;

– расстояние между строками: 0,5-1,0 высоты знака;

– количеством знаков в строке: 4-80;

– максимально допустимым количеством строк для цветного изображения: не более 25;

– освещенности и равномерности яркости между окружающими условиями и различными участками рабочего места.

При проектировании стола следует учитывать следующее:

– высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;

– нижняя часть стола должна быть сконструирована так, чтобы разработчик мог удобно сидеть, не был вынужден поджимать ноги;

– поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения разработчика;

– конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей, личных вещей).

В соответствии с требованиями для оборудования рабочего места определенными в ГОСТ 21889-76 выбираем следующие параметры стола:

– высота рабочей поверхности стола 700 мм;
– высота пространства для ног 650 мм;
– возможность размещения документов справа и слева;
– расстояние от глаз до клавиатуры 400 мм;
– расстояние от глаз до документов 500 мм.

Кресло оператора должно быть устойчивым. Его конструкция, размеры, форма, наклон сиденья и спинки должны позволять сидеть, выпрямившись, поддерживая тяжесть верхней части туловища не напряжением мышц спины, а путем опоры на спинку.

Сиденье должно иметь некоторый наклон назад (на 5-6 градусов), обеспечивающий устойчивость позы, спинка кресла должна иметь вогнутую форму.

– высота сиденья над уровнем пола 450 мм;
– поверхность сиденья мягкая с закругленным передним краем;
– расстояние от сиденья до нижнего края рабочей поверхности не менее 150 мм.

Габаритные размеры рабочего места должны быть достаточными для:

– размещения работающего человека с учетом его рабочих движений и перемещений;
– расположения средств управления в пределах максимальной и минимальной границ моторного пространства;
– оптимального обзора визуальной информации;
– смены рабочей позы и рабочего положения;
– свободного доступа к оборудованию при ремонте и наладке;
– рационального размещения основных и вспомогательных средств труда;
– ведения записей, работы с документами и приборами.

Как правило, монитор должен располагаться – примерно на расстоянии вытянутой руки – оптимальное расстояние – 450-500мм.

Плоскость экрана должна быть перпендикулярной линии взора, а высота расположения такой, чтобы при выпрямленной спине пользователя и отрегулированном кресле направление взгляда было бы горизонтальным или на 10-15 градусов ниже центра монитора.

Устройства ввода-вывода, такие как принтер, плоттер, сканер и т.д., рекомендуется располагать справа от оператора в зоне максимальной досягаемости. Шумящие устройства следует выносить за пределы рабочей зоны.

### 5.1.4 Расположение рабочего места в помещении

Чаще всего рабочие места сотрудников, работающих с ПК, располагают подальше от окон и таким образом, чтобы оконные проемы находились сбоку. Если экран дисплея обращен к оконному проему, необходимы защитные экраны. Окна рекомендуется снабжать светорассеивающими шторами, регулируемыми жалюзи или солнцезащитной пленкой с металлизированным покрытием.

Монитор, документы, клавиатура должны быть расположены так, чтобы перепад яркостей их поверхностей, зависящий от их расположения относительно источников света, не превышал 1:10 при рекомендуемом значении 1:3. При яркости изображения на экране 50-100 кд/м (номинальное значение) освещенность документа должна составлять 300-500 лк. Должны быть исключены слепящие яркости, блики и отображения от стекла экрана.

Для исключения засветки экранов дисплеев прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора и стене с окнами.

Для обеспечения оптимальных условий работы операторов дисплейных устройств необходима определенная отделка помещений: должны использоваться диффузно – отражающиеся материалы с коэффициентом отражения для потолка – 0,7 – 0,8; для стен – 0,5 – 0,6; для пола – 0,3 – 0,5.

Поверхность пола в помещениях эксплуатации компьютеров должна быть ровной, без выбоин, нескользкой, удобной для очистки и для влажной уборки, обладать антистатическими свойствами.

Схемы размещения рабочих мест должны учитывать расстояния между рабочими столами с мониторами (в направлении тыла поверхности одного монитора и экрана другого монитора), которое должно быть не мене 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.

### 5.1.5 Требования к микроклимату в рабочей зоне помещений. Категория работы

Работа с компьютером относится к категории легких физических работ (категории 1а и 16), производимых сидя и не требующих систематического физического напряжения.

В соответствии с требованиями СанПиН 2.2.2.542 – 96 в помещениях, в которых работа на ПК является основной, должны обеспечиваться оптимальные величины показателей микроклимата, приведенные в таблице 5.1.

Таблица 5.1 Оптимальные нормы показателей микроклимата в рабочей зоне помещений с ПК

| Период года | Категория работ | Температура воздуха, °С | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
|---|---|---|---|---|
| Холодный | легкая 1а | 22-24 | 40-60 | 0,1 |
| | легкая 16 | 21-23 | 40-60 | 0.1 |

| | | | | |
|---|---|---|---|---|
| Теплый | легкая 1а | 23-25 | 40-60 | 0,1 |
| | легкая 16 | 22-24 | 40-60 | 0,2 |

Воздух, поступающий в помещение, должен быть не ниже 19°С и очищен от пыли и микроорганизмов.

Наиболее эффективным мероприятием, обеспечивающим в помещении автоматическое поддержание оптимальных параметров микроклимата в нормативных пределах и требуемую чистоту воздуха, является применение кондиционирования.

При необходимости повышения влажности воздуха в помещениях следует использовать увлажнители воздуха, заправляемые ежедневно дистиллированной или прокипяченной водой.

Уровни положительных и отрицательных аэроионов в воздухе помещении с ПК должны соответствовать нормам, приведенным в таблице 5.2.

Т а б л и ц а 5 . 2 Уровни ионизации воздуха помещений при работе на ПК

| Уровни | Число ионов в 1 $см^3$ воздуха | |
|---|---|---|
| | п+ | п- |
| Минимально необходимые | 400 | 600 |
| Оптимальные | 1500-3000 | 30000-50000 |
| Максимально допустимые | 50000 | 50000 |

Для улучшения в помещении качественного состава воздуха и обеспечения в нем требуемого содержания аэроионов помещения с ПК каждый час должны быть проветрены. При нарушении аэроионного режима в помещении следует применять ионизаторы воздуха.

## 5.2 Освещение рабочего места

Важным условием высокопроизводительного труда разработчика является *рациональное освещение* его рабочего места. Объясняется это тем, что посредством зрительного аппарата человек получает около 90% информации из окружающего мира, ее поступление во многом зависит от качества освещенности.

*Неблагоприятными факторами* для профессиональной деятельности разработчика, негативно влияющими на зрение, являются:

– пониженный уровень освещенности, приводящий к перенапряжению глаз и, как следствие к быстрому утомлению.

– чрезмерно высокая освещенность также является причиной быстрой утомляемости, приводя к раздражению и рези в глазах;

– неправильное направление света способствует появлению резких теней или сильных бликов, заметно быстрее утомляющих глаза.

Для освещения помещений и рабочих мест с ПК должно применяться естественное, искусственное и совмещенное освещение.

Естественное освещение должно осуществляться через светопроемы, ориентированные преимущественно на север и северо-восток, и обеспечивать коэффициент естественной освещенности (КЕО) не ниже 1.2% в зонах с устойчивым снежным покровом и не ниже 1,5% на остальной территории.

Рабочие места с ПК по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева.

Искусственное освещение в помещениях, в которых эксплуатируются ПК, должно осуществляться системой общего равномерного освещения.

Хорошо проникающий в помещение естественный свет оказывает благоприятное воздействие на психику человека, вызывая положительные эмоции, обеспечивая хорошие гигиенические условия работы. За счет естественного освещения стимулируется обмен веществ, кровообращение, дыхание, деятельность центральной нервной системы, что в свою очередь, обеспечивает высокую производительность труда.

### 5.2.1 Определение системы освещения

Выбирая систему освещения, необходимо учитывать, что более эффективной является система комбинированного освещения, но система общего освещения более гигиенична, т.к. обеспечивает большую равномерность освещенности рабочих поверхностей. Используя локализованное общее освещение, можно наиболее просто добиться высоких уровней освещенностей на рабочих местах без значительных затрат. При повышенных требованиях к освещенности отдельных рабочих мест используют комбинированную систему освещения.

Искусственное освещение различных помещений осуществляется с помощью электрических источников света – ламп накаливания и люминесцентных ламп, так называемых ламп дневного света. Для освещения производственных помещений широкое применение находят люминесцентные лампы, что объясняется рядом причин.

– применение данного типа ламп позволяет получить в 1,5 – 2 раза большую освещенность при одинаковом расходе электрической энергии по сравнению с лампами накаливания (экономичность);

– свет люминесцентной лампы мягкий, по своему спектру наиболее близкий к дневному естественному, при почти полном отсутствии теней, что позволяет лучше различать цвета и оттенки;

– также, люминесцентные лампы имеют более длительный срок службы, превышающий срок службы лампы накаливания в 10-12 раз.

К недостатком данного вида ламп можно отнести:

– высокую установочную стоимость;

– зависимость светового потока от температуры окружающей среды;

– существенная пульсация светового потока.

### 5.2.2 Расчет искусственного освещения

Найдем необходимое число ламп при помощи метода коэффициента использования.

Расчёт системы общего освещения производится методом коэффициента использования светового потока, который выражается отношением светового потока, падающего на расчётную поверхность, к суммарному потоку всех ламп. Его величина зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемой коэффициентами отражения стен и потолка.

Необходимый световой поток лампы в каждом светильнике рассчитывается по формуле

$$Fo = \frac{E * k * S * Z}{\eta} \qquad (5.1)$$

где $E$ – заданная минимальная освещённость, лк (E = 500);

k – коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (для люмин. ламп – 1,5);

$S$ – освещаемая площадь, м$^2$ (S = 12) ;

$Z$ – отношение средней освещённости к минимальной (обычно принимается равным 1.1-1.2 , для люмин. ламп – 1,1);

$\eta$ – коэффициент использования светового потока в долях единицы (отношение светового потока, падающего на расчётную поверхность, к суммарному потоку всех ламп).

Коэффициент использования $\eta$ зависит от типа светильника, от коэффициентов отражения потолка $\rho_п$, стен $\rho_с$, расчётной поверхности $\rho_р$, индекса помещения, который рассчитывается по формуле

$$i = \frac{S}{h * (a + b)} \qquad (5.2)$$

где $h$ – высота светильника над рабочей поверхностью;

$a$ – длина помещения;

$b$ – ширина помещения.

Найдем $h$ по формуле

$$h = H - h_p - h_c = 3 - 0{,}7 - 0 = 2{,}3 \text{ (м)}$$

где $H$ – высота помещения, м (H = 3);

$h_p$ – высота рабочей поверхности от пола, м ($h_p$ = 0,7);

$h_c$ – высота свеса светильника от основного потолка, м ($h_c$ = 0).

$$i = \frac{6*2}{2,3*(6+4)} = \frac{12}{18.4} = 0,65$$

Для светлого фона примем: $\rho_п = 70$, $\rho_с = 50$, $\rho_р = 10 \Rightarrow \eta = 36\ \%$.

$$Fл = \frac{Fo}{N} \qquad (5.3)$$

где Fл – световой поток одной лампы;

Fo – общий световой поток;

N – число ламп;

$\eta$ – коэффициент использования светового потока.

$$Fo = \frac{500*1,5*12*1,1}{0,36} = 27500\ лм$$

Количество ламп рассчитаем из выраженной формулы

$$N = \frac{Fo}{Fл} \qquad (5.4)$$

Для освещения выбираем люминесцентные лампы типа ЛХБ65, световой поток которых $Fл$ = 4400 Лм.

$$N = \frac{27500}{4400} = 6,25 \Rightarrow 6$$

Число светильников выбирается в зависимости от размеров освещаемого помещения, при этом количество светильников должно быть таким, чтобы отношение расстояния между ними к высоте их подвеса над поверхностью было равно 1,5 ÷ 2.

При выборе осветительных приборов используем светильники типа ЛСПО 2. Каждый светильник комплектуется двумя лампами. Размещаются светильники тремя рядами, по два в каждом ряду.

Допускается отклонение (ε) светового потока выбранной лампы от расчётного от – 10 % до + 20 %.

$$E_{факт} = \frac{Ф_Л * N * N_{л.св} * \eta}{S * z * k} \qquad (5.5)$$

$$E_{факт} = \frac{4400*6*2*0,36}{20*1,1*1,5} = 576\ лк$$

Отличие от нормированного уровня

$$\frac{E_{факт} - E_{норм}}{E_{норм}} * 100\% \qquad (5.6)$$

$$\frac{576 - 500}{500} * 100\% = 15{,}2\%$$

Для исключения засветки экранов дисплеев прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора и стене с окнами. Такое размещение светильников позволяет производить их последовательное включение в зависимости от величины естественной освещённости и исключает раздражение глаз чередующимися полосами света и тени, возникающее при поперечном расположении светильников.

Электрическая мощность всей осветительной системы вычисляется по формуле

$$P_{общ} = P_1 * N \qquad (5.7)$$

где $P_1$ – мощность одной лампы = 65 Вт;

$N$ – число ламп = 6.

$$P_{общ} = 65 \cdot 6 = 390 \text{ Вт}$$

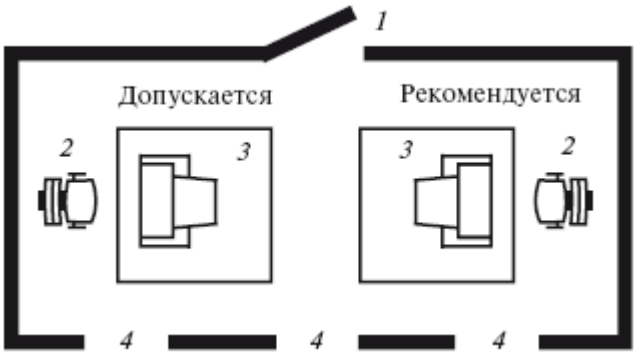


Рисунок 5.1 – План помещения

1 – Дверь; 2 – Рабочее кресло; 3 – Рабочий стол; 4 – Окна.

**Вывод**

В результате анализа условий труда на рабочем месте разработчика были выявлены два основных фактора, влияющие на организм человека в процессе работы. Этими факторами являются освещение и правильное кондиционирование.

Правильно спроектированное и выбранное производственное освещение улучшает условия зрительной работы, снижает утомляемость, способствует повышению производительности труда, благотворно влияет на производственную среду, оказывая положительное психологическое воздействие на сотрудника, повышает безопасность труда, и снижает травматизм.

Также учтены все факторы по соблюдению в помещении максимально благоприятного микроклимата и расположения рабочих мест сотрудников.

## Заключение

Разработка программного обеспечения – интересная и сложная задача, выполнение которой может быть более чем оправдывающей себя. Она требует логического мышления и структурного подхода. Она включает в себя множество задач, не связанных напрямую с изучением языков кода. В ней всегда есть новые знания, которые можно приобрести, новые требования, требующие исследования и новые технологии, с которыми стоит экспериментировать. Наградой является вид работающей финальной версии приложения и осознание своего вклада в достижение этой цели. Программное обеспечение определяет форму современного мира, и те, кто работает над его созданием, могут внести свой вклад в образ будущего.

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

Также было разработано приложение для соответствующей платформы, которое, при желании, может быть перенесено на другие мобильные операционные системы, использующие архитектуру процессора ARM. Данное приложение отвечает всем стандартам и требованиям платформ держателя.

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

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

## Список используемой литературы


1. Сайт https://www.youtube.com/user/freshgamer10/videos
2. Сайт http://startandroid.ru/ru/
3. Сайт http://developer.android.com/intl/ru/index.html
4. Бадагуев Б.Т. Документация по охране труда в организации. – М.: Альфа-пресс, 2010.
5. Рето М., Android 2. Программирование приложений для планшетных компьютеров и смартфонов. – М. Эксмо, 2011.
6. Дюсебаев М.К., Бегимбетова А.С. Методические указания к выпускной работе (для студентов всех форм обучения специальностей 050719 – Радиотехника электроника и телекоммуникации, 050704 – Вычислительная техника и программное обеспечение). – Алматы: АИЭС, 2008.
7. СНиП РК 2.04-05-2002 «Естественное и искусственное освещение. Общие требования. – Астана, 2002.
8. Коробко, В.И. Охрана труда: Учебное пособие для студентов вузов / В.И. Коробко. – М.: ЮНИТИ-ДАНА, 2013. – 239 с.
9. Абдимуратов Ж.С, Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. Бакалавриат. – Алматы: АИЭС, 2009.
10. Абдимуратов Ж. С., Маманбаева С. Е. Расчет производственного помещения. – Алматы: АУЭС, 2009.
11. Айдарханова М., Основы экономической теории. – М.: Фолиант, 2010.
12. Махотина М.В., Симоненко В.И. Экономика в схемах: Учебное пособие. – М.: Эксмо, 2011.
13. Носова С.С., Экономическая теория. – М.: Кнорус, 2010.
14. Артамонова В.С., Иванова С.А. Экономическая теория. – СПб.: Питер, 2010.
15. Камаева В.Д., Лобачевой Е.И. Экономическая теория: Учебник.– М.: Юрайт, 2010.
16. Выпускная работа бакалавров. Экономический раздел: методичка/ Под ред. Бабич А.А., Казыкен Б.Б., Сагира А.А. – Алматы: АУЭС, 2009.