МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра компьютерных технологий

«Допущен к защите»
Заведующий кафедрой Курпайбаев З.К.
профессор, к.ф.-м.н
(Ф.И.О., ученая степень, звание)
_____ «____» _____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка мобильного приложения для обработки и анализа открытых данных

Специальность вычислительная техника и программное обеспечение

Выполнил (а) Толанбаева Г.А. _____
(Фамилия и инициалы) группа

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

Консультанты:

по экономической части:
Бекмуратова А.И., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «28» 04 2016 г.
(подпись)

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

по применению вычислительной техники:
Шайхин Б.М. к.ф.-м.н. доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «25» 05 2016 г.
(подпись)

Нормоконтролер: Шайхин Б.М. к.ф.-м.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «25» 05 2016 г.
(подпись)

Рецензент: _____
(Фамилия и инициалы, ученая степень, звание)
_____ «1» 06 2016 г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет аэрокосмических и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Толыбаева Гаухартас Алгабасқызы
(фамилия, имя, отчество)

Тема проекта Разработка мобильного приложения для обработки и анализа открытых данных

утверждена приказом ректора № 21 от «10» марта 2016 г.
Срок сдачи законченной работы « ___ » _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Разработать мобильное приложение для обработки и анализа открытых данных

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1) выполнить анализ предметной области;
2) проектирование приложения;
3) программирование приложения;
4) разработать дизайн;

Перечень графического материала (с точным указанием обязательных чертежей)

1 рисунок в главе 1
0 рисунков в главе 2
12 рисунков в главе 3
5 рисунков в главе 4
1 рисунок в разделе экономика
2 рисунка в разделе БЖД

Рекомендуемая основная литература

Рето М. Android 2. Программирование приложений для планшетных компьютеров и смартфонов

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Приходько М.Г.	01.04 – 28.04.16	
Эконом. часть	Бекишева А.Ч.	17.03 – 28.04.16	
Глава 1	Шайхин Б.М.	05.01 – 27.01.	
Глава 2	Шайхин Б.М.	01.02 – 29.02.	
Глава 3	Шайхин Б.М.	01.03 – 30.03	
Глава 4	Шайхин Б.М.	01.04 – 30.04.	
Нормаконтроль	Шайхин Б.М.	25.05.2016	

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1.	Глава 1	05.01. - 27.01.	
2.	Глава 2	01.02. - 29.02	
3.	Глава 3	01.03 - 30.03.	
4.	Глава 4	01.04 - 30.04	
5.	БЖД	01.04. - 28.04.16.	
6.	Экономика	17.03. - 28.04.16.	

Дата выдачи задания « 1 » сентября 20 15 г.

Заведующий кафедрой _____ Кулалбаев З.Ж.
(подпись) (Фамилия и инициалы)

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

Задание принял к исполнению
студент _____ Толанбаева Г.Ф.
(подпись) (Фамилия и инициалы)

Аннотация

В данном дипломном проекта описана разработка мобильного приложения для операционной системы Android, которая создана для обработки и анализа открытых данных. Также раскрыто определение открытых данных и выбрана определенная область - это обработка погодных условий для города Алматы. Это приложение полностью соответствует требованиям.

Также был определён анализ условий труда для разработки данного приложения и составлено экономическое обоснование проекта, которое подтверждает экономическую целесообразность.

Аңдатпа

Бұл дипломдық жобада ашық мәліметтерді өңдеу және талдау жасайтын, Android операциялық жүйесі үшін арналған мобильдік бағдарламаның дамуы қарастырылды. Оған қоса, ашық мәліметтердін анықтамасы жазылды және белгілі саласы - ол Алматы қаласының ауа-райының мәліметтерін өңдеу тандалды. Бұл бағдарлама бар талаптарға сай.

Сонымен қатар, бұл бағдарламаны дамуы үшін еңбек жағдайларына талдау анықталды және жобаның экономикалық тиімділігін бекітетін экономикалық негіздемесі құрастырылды.

Annotation

In this diploma project described development of mobile application for the operating system Android, which was created for process and analysis of open data. Also uncovered definition of open data and chose certain area, that is process of weather conditions for Almaty city. This application satisfies the requirements.

Also determined the analysis of working conditions for the development of the application and made the economic justification of the project, which confirmed its economic viability.

Содержание

Введение

Сегодня "умные телефоны" дошли до пика популярности среди всех возрастов, таким образом они стали лидерами продаж, обогнав обычные компьютеры и другие мобильные устройства. Возникает вопрос: чем же смартфоны привлекают современное человека? Дословный перевод слова смартфон(англ. smartphone) — "умный телефон", в действительности смартфоны обладают свойствами персонального компьютера и покоряют своими размерами, в итоге у пользователя есть карманный коммуникатор с дополнительной функциональностью и возможностью изменения настроек задач «под себя». Смартфон, благодаря своей функциональности и обширному выбору мобильных приложений, полностью отвечает требованиям. Что в свою очередь открыло перед программистами просторы для выражения своих способностей и, что не мало важно, возможность заработать – это разработка мобильных приложении под android, windows phone, iOS. Так, современный пользователь желает оставаться онлайн при помощи созданных мобильных приложений. Но стоит отметить, что всё это приобрело актуальность вместе с появлением мобильного интернета. В настоящее время программисты разработали и реализовали мобильные приложения, позволяющее решать различные задачи, к примеру, одни помогают упростить маршрут, иные регулируют количество потребляемых калорий или физических нагрузок. Также есть приложения позволяющие свободно перемещать информацию и получать доступ у нужной, преимуществом других является доступ и быстрая проверка электронной почты. В основе каждой из этих приложений лежит определенная задача или система задач, которые позволяют вам быстро решить проблемы, сэкономить время и достичь комфортного уровня жизни.

Сегодня, с проникновением интернета и информационных систем в жизнь современного пользователя, всё чаще упоминается термин “открытые данные”, которые я попытаюсь раскрыть. В широком смысле слова этот термин лежит идея о том, что предоставляются определённые данные для широких масс в машиночитаемом виде и могут использоваться ими в дальнейшей публикации без ограничений авторского права, так сказать при содействии автора. Проще говоря, открытые данные – это способ публикации имеющейся информации под свободной лицензией в формате, пригодном для последующей обработки и анализа. Сегодня актуальной является публикация открытых данных в области государственной и административной деятельности, так как это делает работу государства более прозрачной и понятной для обычного гражданина, к тому же эта доступность информации государственных органов даёт возможность оценивать и контролировать их деятельность, что в результате – повышает лояльность граждан к власти. Следует также отметить, что открытые данные в машиночитаемом виде

можно использовать при создании различных электронных сервисов, полезных для жизнедеятельности граждан.

Актуальная же выгода в использовании открытых данных колоссальна как и для государства, так и для бизнеса и для конечного пользователя. Чем больше полезной информации будет в открытом доступе, тем быстрее будут создаваться и развиваться инновационные общедоступные сервисы. А чем больше инноваций – тем быстрее экономический рост.

Перспективным направлением сейчас является приложения, разработанные для Android, нужно помнить, что эта ОС разработана не только для коммуникаторов, но и для планшетов, проигрывателей, нетбуков и даже часов. Это говорит об огромной ее функциональности и возможности настройки для реализации множества задач, которые стоят перед разнообразными пользовательскими устройствами. Относительная «молодость» ОС Android и ее востребованность предоставляет создателям мобильных приложений достаточно широкий спектр направлений, в которых может вестись разработка приложений для Android. К тому же, разработчиком под Android может стать гражданин каждой страны. Нужно лишь, заплатить с помощью карточки 20$ и вы становитесь зарегистрированным разработчиком и имеете возможность публиковать бесплатные приложения на Android Market.

Таким образом, учитывая данные особенности Android и дальнейшие перспективы развития программирования мобильных приложении, можно с уверенностью сказать, что разработка мобильного приложения для обработки и анализа открытых данных, а точнее обработка и анализирование погодных условий, предоставленных как открытые данные будет полезна для жителей Алматы, а при дальнейшей разработке и для всего мира. Также с течением времени спрос на программистов-разработчиков будет неуклонно расти. Этим и обусловлена актуальность данной работы.

Целью данной работы было изучение операционной системы android, разработка мобильного приложения для обработки и анализа погодных условий, которое облегчает жизнь населения Алматы.

1. Принципы разработки

1.1 Выбор технологии разработки приложения

Разработка мобильных приложений для мобильных устройств (смартфонов, планшетов) развивается очень быстро. Большинство владельцев бизнесов осознали, что присутствие на мобильных платформах является обязательной составляющей результативной маркетинговой стратегии, к тому же на сегодняшний день почти каждый человек пользуется гаджетами. Таким образом, сформировалось несколько важных решений для успешного старта на рынке мобильных приложений. Одним из них будет выбор правильного способа создания приложения, к тому же сегодня огромный выбор технологий разработки. Разработка приложения состоит из нескольких больших шагов. Первоначально возникает главная идея и формируется список целей, которых приложения должно достигнуть. Цели могут быть маркетинговыми и техническими. Например: привлечь новых и повысить лояльность среди существующих пользователей, упростить процесс ведения клиента от первого знакомства до покупки и т.д., оптимизировать и улучшить интерфейс для мобильных приложений на мобильном устройстве, определить основные цели пользователей-клиентов, что приведет к увеличению количества потенциальной аудитории. Таким образом именно бизнес-задачи должны руководить выбором технологии, а не наоборот. Когда базовые задачи ясны, можно и определиться с техническим воспроизведением. В любом деле есть приоритеты и жертвы, к примеру, если в приоритет брать скорость внесения изменений и охват аудитории, то требуется пожертвовать другими аспектами (возможно, скорость работы) и выбрать технологию, удовлетворяющую поставленной задаче. Либо, наоборот, требуется достичь максимальной адаптированности приложения под определенную платформу и обеспечить максимальную скорость работы.

Приблизительный список параметров для выбора технологии:

– время разработки;

– наличие специалистов;

– удобство разработки и отладки;

– документаци и тех. поддержка;

– скорость работы;

– юзабилити;

– охват платформ

1.2 Операционная система Android

Android – одна из операционных систем нового поколения, созданных для работы с программным обеспечением современных мобильных девайсов. На сегодняшний момент Windows Mobile, Apple iPhone и Palm Pre предлагают очень мощные и простые мобильные приложения в использовании среды разработки. Но в отличие от Android это запатентованные операционные системы, в которых в некоторых случаях приоритет отдается встроенному программному обеспечению нежели приложениям, созданных сторонними программистами. Таким образом, Android выбирают те, кто хочет настроить систему "под себя". Кроме того, невозможно взаимодействия приложений с данными телефона, и эти ос ограничивают или руководят процессом распространения сторонних приложений, созданных для данных платформ. Android же открывает просторы для программистов и новые возможности для их мобильных приложений, к примеру, даёт открытую среду разработки, построенную на открытом ядре Linux. У всех приложений есть доступ к программным средствам девайса, для чего используются специальные серии API-библиотек, к тому же, 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. Пользователь может удалять или изменять встроенные ПО на альтернативные сторонние разработки, будь то номеронабиратель или Рабочий стол.

Так, Android можно определить, 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 оптимизированно для работы на мобильных устройствах;

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

– среду исполнения для приложений, отличающуюся небольшим размером для эффективности использования ее на смартфонах, включающую виртуальную машину Dalvik и библиотеки ядра, отвечающие за функционал Android;

– 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.

Стоит отметить, что открытая архитектура Android позволяет исправлять любые ошибки в пользовательском интерфейсе или дизайне встроенных приложений путем написания расширений или замещений ошибок. Android предоставляет шанс производить собственные интерфейсы для смарфонов, а также мобильные приложения с функционалом и дизайном, настроенные под ваши потребности.

1.2.1 Встроенные приложения Android

В рамках проекта Android Open Source Project (AOSP) (Проект открытых исходных кодов для Android) разработаны определенные программы, которыми оснащены нынешние смарфоны с операционной системой Android.

Перечислим основные из них:

– e-mail-клиент;

– работа SMS;

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

– браузер на базе WebKit;

– музыкальный плеер и галерея фотографий;

– калькулятор;

– «Рабочий стол»;

– будильник.

Во основном Android включает также следующее лицензионное ПО от Google:

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

– 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.

Информация, к которой имеют доступ многие эти приложения, например адресная книга, открыты и для приложений сторонних разработчиков. К тому же, приложения могут обрабатывать такие события,

как входящий звонок или получение SMS. Интерфейс мобильных приложений, которые есть на новых смарфонах под управлением Android, может сильно меняться в зависимости от производителя программного обеспечения и/или оператора, дистрибьютора.

Открытость платформы Android означает, что производители комплектного оборудования (OEM) могут варьировать пользовательский интерфейс и программы на любом девайсе под управлением Android. К примеру, некоторые производители разработали свои собственные интерфейсы на базе Android, например Sense от HTC, MotoBlur от Motorola и интерфейс от Sony Ericsson. Нужно отметить, что для всех совместимых устройств платформа и среда разработки остаются неизменными независимо от производителя или оператора. Естественно, пользовательский внешний вид может иметь изменения, однако приложения будут работать полностью одинаково на всех совместимых с Android устройствах.

1.2.2 Основные характеристики среды разработки для платформы Android

Одним из главных преимуществ Android как среды разработки мобильных приложений является ее интерфейс программирования приложений, называемый коротко API. Для начала стоит рассмотреть понятие "API".

API (интерфейс программирования приложений, интерфейс прикладного программирования) (англ. application programming interface) — определенные классы, функции, процедуры, структуры и константы, собранные воедино, предлагаемых приложением (библиотекой, сервисом) или операционной системой для будущего использования в программных продуктах, который используется программистами-разработчиками при написании всевозможных приложений и программ.

Основные характеристики Android:

— отсутствие трат на получение лицензии, распространение и разработку, а также каких-либо механизмов сертификации готовых программных продуктов;

– свободный доступ к Wi-Fi-устройствам;

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

– комплексный API для работы с навигационными службами, к примеру GPS;

– есть контроль над мультимедийными устройствами, включающие проигрывание или запись информации с камеры и микрофона;

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

– библиотеки для работы с Bluetooth со способностью 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;

– браузер на базе WebKit с открытыми исходными кодами и поддержкой HTML5;

– поддержка приложений, использующих функционал работы с картами в своем пользовательском интерфейсе;

– графика с аппаратным ускорением, которая подстроена под мобильные девайсы, которая к тому же содержит библиотеку для работы с векторной 2D- графикой и поддержку трехмерной графики с использованием OpenGL ES 2.0;

– мультимедия-библиотеки для произведения и записи аудио, видеофайлов или изображений;

– локализация для дальнейшей работы с динамическими ресурсами;

– некое количество программных компонентов для повтора в использовании компонентов и замещения встроенных приложений.

1.3 Архитектура Android приложений

Разработка ОС Android началась в 2003 молодой компанией Android Inc, в 2005 году эта компания была куплена Google. Я считаю, что главные особенности архитектуры Android были определены именно в этот период. Но это заслуга не только Android Inc, архитектурные концепции и финансовые ресурсы Google оказали огромное влияние на нынешнюю архитектуру Android. К примеру, 2003-2005 года отличились повышенным вниманием к AJAX приложениям, что оказало основополагающее влияние на архитектуру Android: отталкиваясь от многих пунктов, можно сказать, что она ближе предрасположена к архитектуре типичного AJAX приложения, нежели к рабочему GUI приложению, написанному на Java, C#, C++, VB.

Связь с архитектурой Eclipse заметна в выборе принципа создания GUI, который больше тянет на SWT, нежели на Swing.

В нормах оформления кода Android есть «венгерская нотация», которая появилась в стенах MS. Я считаю, что тот, кто написал эти нормы, ранее занимался созданием программ под Windows.

ОС Android содержит 3 уровня:

– В основе лежит модифицированная и урезанная версия Linux;

– над уровнем Linux находится уровень инфраструктуры приложения, содержащий виртуальную машину Dalvik, веб-браузер, базу данных SQLite, некие инфраструктурные «костыли» и Java API;

– уровень написанных в Google Android-приложений; в общем, это увеличение уровня инфраструктуры, так как программист- разработчик имеет

возможность использовать эти приложения или их части как блоки для собственных дальнейших разработок.

1.3.1 Java классы

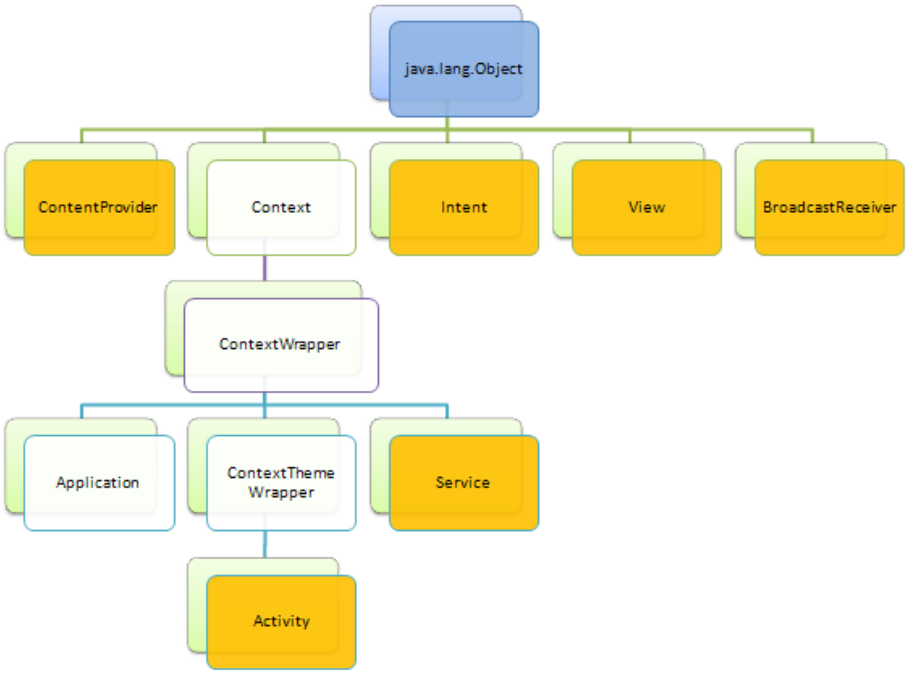

Рис.1.1 - иерархия основных классов из Android SDK

На вышеприведенной диаграмме показана основное количество классов, на самом деле их намного больше. ContentProvider, Intent, View, BroadCastReciever, Service, Activity — классы с которыми программист-разработчик взаимодействует напрямую, то есть, наследуются от них. View — основной класс для всех GUI виджетов. Внешний вид Android-приложения — дерево экземпляров наследников от данного класса. Мы имеем возможность создавать заданное дерево программным путем, но практика показывает, что это неправильный способ. GUI интерфейс задаётся при взаимодействии с XML (файлы слоёв, layout files), и в процессе реализации сразу становится деревом данных объектов.

Отношением между Activity и GUI интерфейсом является отношением один к одному; в основном, каждый Activity обладает только одним взаимосвязанным с ним слоем GUI интерфейса, и так же имеется способ наоборот. Так же Activity владеет жизненным циклом.

Продолжение жизненного цикла Activity может находиться в одном из ниже заданных трёх состояний:

— Активно и реализуется —интерфейс находится на переднем плане, то есть технически находится на вершине стека Activity;

— Приостановлено — при условии, что этот интерфейс пользователя потерял фокус, но при этом всё ещё видим. В таком состоянии никакой код не выполняется.

— Завершено — пользовательский интерфейс невидим. В таком состоянии код не выполняется.

Реализация кода Activity происходит только тогда, когда необходимый пользовательский интерфейс видим и имеет фокус. Конечно, гарантии, что объект Activity и связанные с ним объекты находятся в памяти отсутствуют, когда Activity приостановлено или завершено .

Класс ContentProvider и его подклассы - model в архитектуре MVVM. Во многих практических случаях это обёртка над базой данных SQLite с немного причудливым способом доступа на основе URI. Теоретически, никто не мешает разработчику создать ContentProvider на основе чего-то ещё, кроме базы данных. Тем не менее, существующий метод query() контент-провайдера возвращает объект Cursor, который крайне похож на JDBC ResultSet интерфейсом и тем, как он работает. Поэтому вряд ли кто-то усомнится, что настоящее назначение контент-провайдеров — инкапсулировать базу данных.

По-моему, команда Android пришла к такому дизайну соединив две хорошие, но не слишком совместимые идеи. То есть, кажется, что базовая идея контент-провайдеров базируется на архитектуре AJAX приложений. AJAX приложения обычно используют архитектуру MVVM, где модель представлена как URI на стороне сервера (тем не менее, это изменилось с появлением HTML5, который позволяет хранить данные локально). Таким образом, тот факт, что контент-провайдеры запрашиваются с помощью URI и создают расширение с помощью типов MIME указывает на то, что в основе лежит AJAX. К тому же стоит напомнить, что разработчики из Google Inc. разработали и реализовали огромное количество AJAX приложений, к примеру Gmail, Google Docs и т.п., поэтому я считаю, что идеи заимствовать из архитектуры AJAX вполне допустимы и реальны.

Одни из тяжело классифицируемых - класс Service и его подклассы. Я думаю, что Service — это вид Model, которая обслуживает немного отличающиеся виды использования, нежели ContentProvider. По мне так, архитектурный дизайн Android Service навеян сервисами OSGI. Полагаю, что сервисы были созданы Google-ом как решение логической проблемы, возникшей из-за модели потоков Android. Рассмотрим это: Activity активен и реализует работу только, когда его пользовательский внешний вид находится на переднем плане. Как только интерфейс следующего Activity закрывает собой текущее, последнее останавливается, даже если оно что-то обрабатывало. Предположим, что вам нужно выполнить некую операцию, даже если процесс, которые её выполняет, не на переднем плане? То с помощью Activity вы не можете этого сделать. Вы не сможете это сделать и с помощью ContentProvider, поскольку у них нет собственного жизненного цикла, и они могут выполняться только пока Activity, использующее его,

активно. И тут появляются сервисы. Они могут выполняться даже когда процесс, в котором они работают, не на переднем плане. Так, если вы разрабатываете Activity, выполняющее растянутую во времени операцию, которая должна завершиться даже работая в фоне, вы должны создать Service, реализующий эту операцию, и запустить его из Activity.

Фактически, разработчик под Android не ограничен одним только расширением классов из Android SDK, он может писать собственные классы по своему личному желанию. Но все они будут только помощью («helper classes») для классов из Andoird SDK.

1.3.2. Манифест Android

Ещё одной базовой частью в структуре создания Android-приложения является манифест Android , внедрение которого была заимствована манифестами плагинов для Eclipse. Манифест Android - это XML файл, который обладает несколькими функциями, котрые были заданы Google-ом:

- задаёт идентификатор, то есть имя Java-пакета приложения;
- определяет набор компонентов мобильного приложения, таких как активити, сервисы, броадкаст-ресиверы и контент-провайдеры, затем определяя их идентификаторы, создаваемые каждый из компонентов и воспризводит возможности;
- заранее узнает о процессах, содержащих компоненты данного приложения;
- задаёт разрешения, которые мобильное приложение должно иметь, для дальнейшего доступа к закрытым частям API и работы с оастальными программами;
- еще задаются разрешения, требующиеся для доступа к компонентам приложения;
- перечисление классов Instrumentation, что дают доступ к файлам и необходимые данные в процессе деятельности мобильного приложения. Эти объявления существуют и в манифесте до того, как приложение создается и производится тест работы;
- задаёт меньший шаг Android API, требуемый программой.

1.3.3. SQLite

Одно из отличительных черт является то, что платформа Android имеет встроенный инструментарий для управления базой данных sqlite3. SQLite - это продукт, который успел завоевать признание во всем мире и получить множество наград. Нужно отметить, что iOS также использует базы данных SQLite. К тому же недавно и Microsoft присоединилась к данному решению и телефоны Windows Phone 8 также работают с базой данных SQLite.

SQLite - это программа с открытыми исходными кодами, которая поддерживает нормы возможности обычной SQL: синтаксис, транзакции и другое. Преимуществом является то, что занимает маленькое место - это

около 250 кб. SQLite уже встроен в любое Android-устройств, то есть его не нужно устанавливать отдельно.

SQLite поддерживает типы text (аналог String в Java), integer (аналог long в Java) и real (аналог double в Java). Остальные типы следует конвертировать, прежде чем сохранять в базе данных. SQLite сама по себе не проверяет типы данных, то есть вы можете с легкостью записать целое число в колонку, предназначенную для строк, и также наоборот. Еще здесь нет типа, работающего с датами, можно только применять строковые значения, вот к примеру, как 2013-03-28 (28 марта 2013 года). Для даты со временем полагается использовать формат 2013-03-27T07:58. В этих случаях можно использовать некоторые функции SQLite для добавления дней, установки начала месяца и т.д. SQLite не поддерживает часовые пояса, тип boolean, нужно использовать числа 0 для false и 1 для true. Стоит отметить, что здесь не используется популярный в MySQL тип varchar(n), ограничивающий длину строки.

Для начала работы с базой данных необходимо применить функционал для создания или обновления базы данных. Так как сама база данных SQLite представляет собой файл, то фактически при работе с базой данных, вы работаете с файлом, поэтому операции чтения и записи могут быть достаточно медленными. Поэтому рекомендуется работать с асинхронными операциями, к примеру, при помощи класса AsyncTask.

Когда ваше мобильное приложение создаёт базу данных, она сохраняется в каталоге DATA/data/имя_пакета/databases/имя_базы.db. Метод Environment.getDataDirectory() возвращает путь к каталогу DATA. Главными пакетами для работы с базой данных являются android.database и android.database.sqlite.

Стоит отметить, что база данных SQLite доступна только приложению, которым оно создано. Если вы хотите дать доступ к данным другим приложениям, вам рекомендуется использовать контент-провайдеры (ContentProvider).

1.3.4.Ресурсы

Большинство современных приложений и программ используют ресурсы в разработках, моё Android приложение также использует эту функцию. Следующие типы ресурсов применяются ими:

- изображения;
- слои GUI (XML файлы);
- объявления меню (XML файлы);
- текстовые строки.

В Java ресурсы определяются строками, которые могут содержать, к примеру, путь и имя файла, содержащего изображение, или ID данной строки. Проблема заключена в том, что ошибки в таких ссылках не обнаруживаются в ходе трансляции кода.

Я рассмотрела следующий пример, где файл mybutton.png содержит картинку для кнопки и программист совершает маленькую ошибку и пишет mybuton.png, ссылаясь на ресурс из кода. В результате, код пытается использовать несуществующий ресурс, но компиляция будет успешна. Ошибка может быть обнаружена только в ходе тестирования, а может и не быть обнаружена вовсе.

Разработчики Google нашли хорошее решение этой проблемы. Так, в начале сборки Android-приложения работает особенный Java-класс с именем R, который содержит несколько static final наборов данных. В свою очередь, каждый такой набор данных является ссылкой на отдельный ресурс. Эти ссылки служат в кодинге мобильного приложения для связи с ресурсами. В результате, каждая ошибка в ссылке на ресурсы появляется в ходе работы компиляции.

1.3.5.Файлы

Типы файлов, что используются Android-приложением:

- файлы «общего назначения»;
- файлы базы данных;
- файлы Opaque Binary Blob (OBB);
- кэш файлы.

Но в конечно итоге, все они являются файлами, принадлежащими Linux, что можно раскрывать их как отдельные типы файлов только в случае работы над ними различными API, также при отдельном хранении в различных папках. К тому же они отделяются от файлов, что хранятся во внутреннем или внешнем хранилище.

2. Определение предметной области

2.1 Раскрытие понятия "открытые данные"

Открытые данные (на англ. open data) — принцип, содержащий в себе идею о том, что определённые данные или разработанная информация должны иметь открытый доступ в машиночитаемом виде и в дальнейшей публикации без ограничений авторского права и каких-либо систем контроля. Таким образом, это информация, которую каждый пользователь сети может свободно использовать и распространять. Допустимы лишь рекомендации об указывании источника и дальнейшее распространение на исходных условиях. Получить данные без ограничений авторского права можно с помощью свободных лицензий Creative Commons. Если какой-либо набор данных не является общественным достоянием, либо не связан лицензией, дающей права на свободное повторное использование, то такой набор данных не считается открытым, даже если он выложен в машиночитаемом виде в Интернет.

Цели распространения открытых данных схожи с другими «открытыми» движениями, такие как открытое программное обеспечение (open source), открытый контент (open content) и открытый доступ (open access).

Рост популярности идеи об открытых данных с 2009 связан, прежде всего, с запуском правительственных инициатив, таких как egov.kz. Теперь граждане Республики Казахстан могут иметь свободный доступ к опубликованному набору данных в машиночитаемом виде, что обеспечивает прозрачность деятельности администрации страны. Актуальностью данной системы является обеспечение сбора актуальной информации в различных сферах государства: статистические данные, топографические сведения, сведения социального характера. Что в итоге ведёт обеспечение заинтересованности населения и в дальнейшей популяризации наборов данных. Одной из главных функций данного сайта является обеспечение удобства использования сведений гражданами РК. В основном среди населения открытые данные часто ассоциируются с нетекстовыми материалами такими как карты, геномы, химические компоненты, математические и научные формулы, медицинские данные, данные о биологическом разнообразии. Проблемы в основном происходят по причине что эти данные могут быть коммерчески ценными или могут быть собраны в некие ценные продукты. Доступ к данным, и их последующее использование, руководятся организациями — государственными или частными. Контроль может быть производиться через ограничения, лицензии, копирайт, патенты и требования оплаты для доступа и повторного использования, коротко говоря всеми возможными требованиями. Сторонники идеи «открытых данных» выступают за то, что данные ограничения идут против общественного блага и данные должны быть доступны без ограничений или оплаты. Также важным является чтобы данные были доступны без последующих запросов на разрешение, хотя и способы повторного использования, такие как создание продуктов на базе данных, могут контролироваться лицензией.

Государственные данные во всех сферах представляют один из важных интересов для граждан страны и многочисленные некоммерческие организации и отдельные активисты борются за открытость государственной информации в машиночитаемой форме. Многие национальные правительства в рамках стратегий «открытого государства» создали веб-сайты для распространения части данных, обрабатываемых в области государственного управления.

Полностью раскрыть определение открытости можно раскрыв детали:

— доступность: стоимость открытых данных не должны превышать настоящей стоимости и иметь полный доступ, в основном, через интернет. При этом формат открытых данных должен иметь возможность для удобного чтения и дальнейшего видоизменения;

— повторное использование и свободное распространение: данные не должны ограничиваться какими-либо лицензиями и требованиями, должны быть даны при условиях, что позволяют их последующее использование и распространение, в тому же - при соединении с другими открытыми данными;

— всеобщее участие: каждый пользователь открытыми данными может пользоваться и дальше распространять их. Что значит, что нельзя запрещать какие-либо области в применении, людьми или группами.

2.2 Назначение мобильного приложения

На сегодняшний день очень сложно представить современный компьютер или смартфон без погодного гаджета, ведь именно с помощью этих маленьких и неприметных программ пользователи данных устройств могут быстро и абсолютно бесплатно узнать последние данные о погоде на тот или иной промежуток времени. Так например , даже самый простой виджет прогноза погоды для windows имеет соответствующий набор инструментов для показа прогноза погоды в удобном виде. На самом деле, буквально несколько лет назад было тяжело довериться прогнозам наших синоптиков даже на завтрашний день. Сейчас же данные о погоде стали гораздо точнее, к примеру компания Яндекс даже обещает сообщить о погодных условиях с точностью до дома вашего проживания в своем приложении о погоде. То есть сегодня любой пользователь может узнать информацию о погоде, которая когда-то казалось нереальной, сейчас же является вполне стандартной: доступны данные о влажности, давлении, скорости и направлении ветра, времени восхода и захода солнца — все это наглядно и по часам.

Говоря о приложении представленное компанией Яндекс, она предоставляет все эти описанные функции и для осуществления этого использовала технологию Meteum, то есть благодаря ей мы и обязаны таким точным сведениям. Кстати, у нас есть возможность улучшить работу сервиса — для этого в самом приложении есть специальный раздел, позволяющий отправить Яндексу информацию о погоде там, где вы находитесь в данный момент. Также в приложении есть возможность добавлять интересующие вас места, таким образом вы всегда будете обладателем актуальных сведений о погоде в выбранных вами регионах.

На сегодня почти каждый человек имеет смартфон и хочет знать прогноз погоды не только на завтрашний день, но и на несколько дней, на неделю вперед. В данный момент, благодарю популярности мобильного интернета и разнообразию выбора мобильных приложений, погоду можно узнать. К тому же, помимо этого сейчас доступны данные о скорости ветра, его направлении, атмосферном давлении. Такие приложения пользуются популярностью у метеозависимых людей, которые чаще всех стремятся узнать прогноз погоды на ближайшее время. Это помогает им контролировать свое самочувствие, принять необходимые меры в случае, если погода резко изменится. Погода может изменяться очень быстро, поэтому перед важными мероприятиями, поездками и так далее узнать ее просто необходимо. Так, вы сможете избежать непредвиденных обстоятельств и планировать свои дела.

Для пользователей-водителей прогноз погоды также очень важен. Если у него есть планы и надо поехать куда-то далеко, то он обязательно смотрит

прогноз на ближайшее время. Ведь основополагающим является выбор правильной резины, также требуется залить омыватель, к тому же быть готовым к природным катаклизмам, таким как туман, и другим природным изменениям.

Для новобрачных погода тоже играет большую роль. Ведь дождь, снег, град, сильный ветер могут помешать провести торжество качественно. Если заранее быть готовым к погодным изменениям, то свадьба пройдет «на ура».

Также для путешественников погодные изменения играют большую роль, собираясь в поход на дальнее расстояние, человек обязательно должен подготовиться в резкой смене температур, взять теплые вещи, палатку, все необходимое. Планируя отдых летом, люди тоже учитывают погоду на весь сезон, время отпуска.

2.3 Цель и особенности написания мобильного приложения

Sunshine - приложение, которое отображает текущую погоду, а также прогнозы на ближайшее время, включая влажность воздуха, скорость ветра, давление, при этом определяя местоположение пользователя. Удобное приложение прогноза погоды на платформе Android будет полезно всем, кто является обладателем смартфона Android и хочет всегда быть в курсе актуальных прогнозов погоды на ближайшее время. Еще одной особенностью является возможность поделиться приложением с друзьями в используемых социальных сетях.

Я считаю, что приложение отличается достаточно простым интерфейсом, особенно в век минимализма. Вам нужно лишь скачать приложение, открыть его, в свою очередь оно связанное с GoogleMap определит вашу локацию, для которой обработает и проанализирует погоду в интернете с OpenWeatherMap.com, после чего Вы сможете увидеть и выбрать интересующий вас период. Sunshine использует открытые даные, взятые в Интернете и выдает детальнейший прогноз на ближайшие время, и точные характеристики погодных условий. Вы можете иметь доступ к Интернет в первый сеанс и просмотреть погоду, затем проанализированная информация сохраняется в встроенной базе данный Android Studio, после чего Вы сможете просмотреть эту информацию и без доступа в Интернет. Интерфейс данного мобильного приложения обладает гибкими настройками и простатой в использовании.

3 Инструменты разработки приложения

3.1 Инструменты разработки кода приложения

Для программирования будущего мобильного приложения в данном проекте будет использована среда для разработки Android Studio, а языком будет служить Java. Таким образом, для работы в Android Studio необходимо наличие установленного Java SE Development Kit (JDK) на текущей системе.

Рекомендуемая версия JDK не ниже 7и установить его можно следуя по ссылке.

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

Java SE Development Kit - это бесплатный инструментарий для реализации, который является бесплатным. Так, после перехода по отмеченной выше ссылке, вам требуется выбрать версию, которая соответствует вашей операционной системе, в моём случае Windows. После загрузки нужного файла его нужно произвести и по инструкции установить данный инструментарий. Затем следует загрузить установочный файл Android Studio с официального сайта Android для программистов-разработчиков, которые доступны по данному адресу:

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

Пройдя по данной ссылке необходимо выбрать требуемую версию, которая соответствует вашей операционной системе и после окончания загрузки нужного файла, установить программу по заданной инструкции. Также стоит отметить, что загрузка Android NDK для работы в Android Studio не требует необходимости, так как все утилиты уже включены в набор поставки приложения. После запуска приложения, вас будет ожидать экран приветствия изображенный на рисунке 3.1.

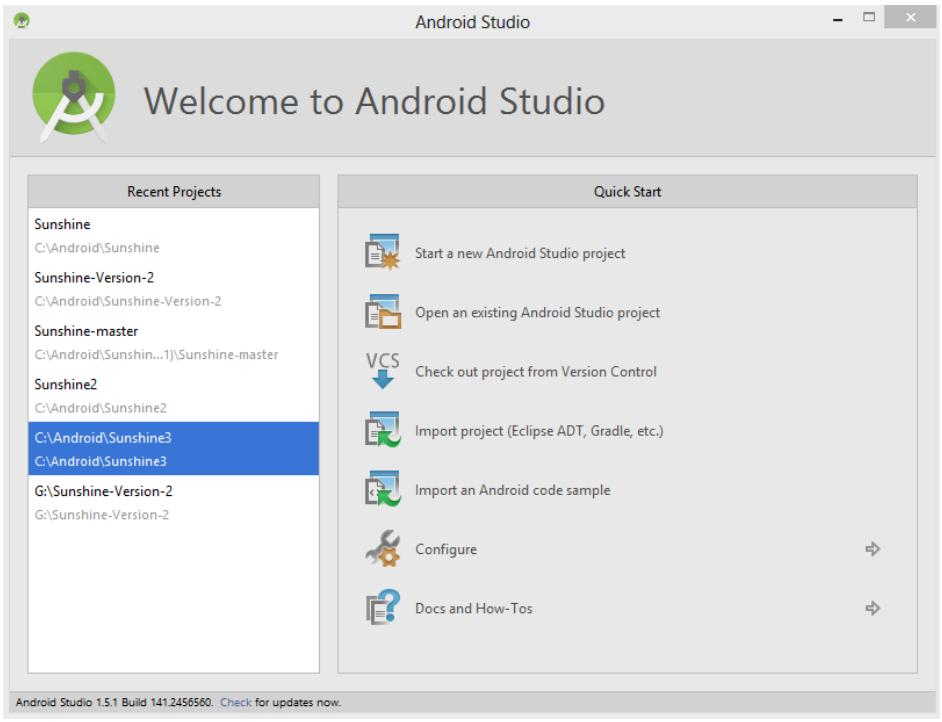

Рисунок 3.1 – Экран приветствия Android Studio

Здесь вы сможете:

- создать новый Android Studio проект;
- импортировать уже существующий проект;
- импортировать пример Android кода;

– открыть уже существующий Android Studio проект;
– открыть любой недавний проект, и 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 3.2.

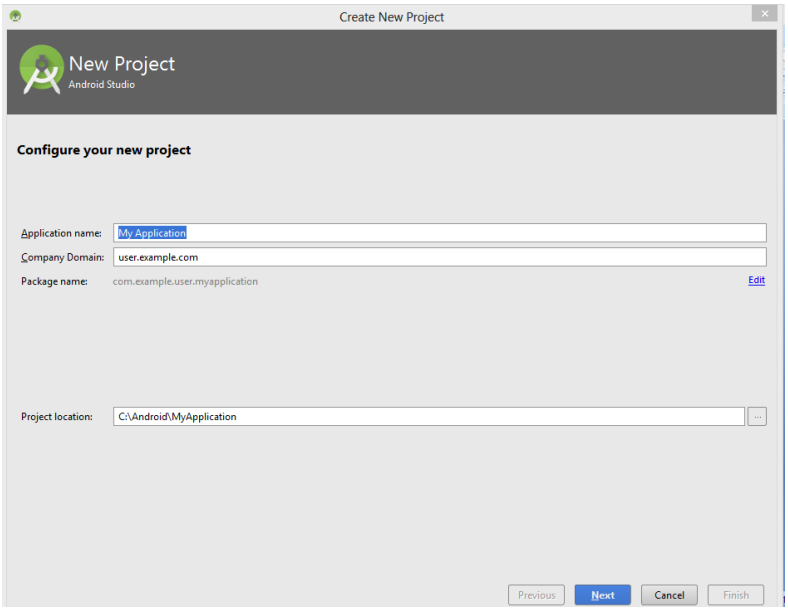

Рисунок 3.2 – Окно создания проекта Android Studio

На этом экране вы можете задать имя приложения, название модуля, название файлы для приложения, а также место расположения каталога с данными проекта. 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 изображено на рисунке 3.3. Этот редактор устроит надобности практически всех разработчиков, так как позволяет создать быстро и качественно значки приложения для разных плотностей экрана и сразу же увидеть результат.

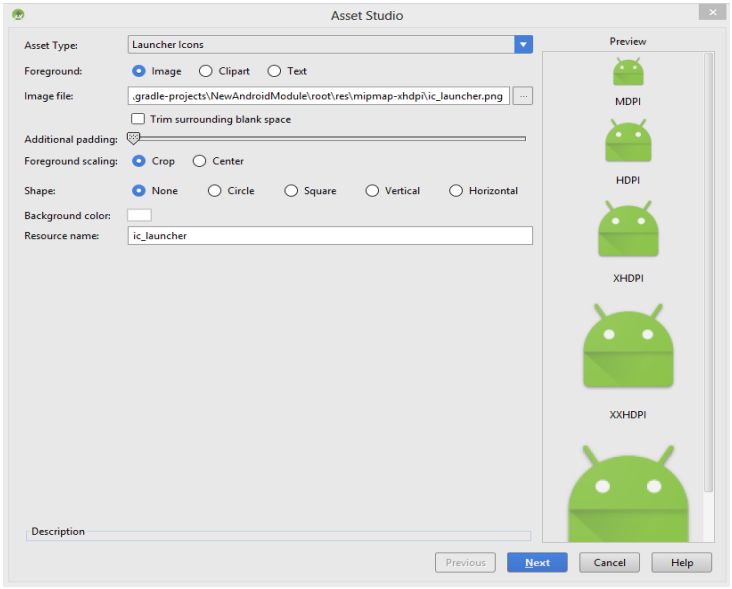

Рисунок 3.3 – Окно создания значка приложения

После завершения этапа создания значка, вы переходите на экран выбора базовой activity, изображенный на рисунке 3.4.

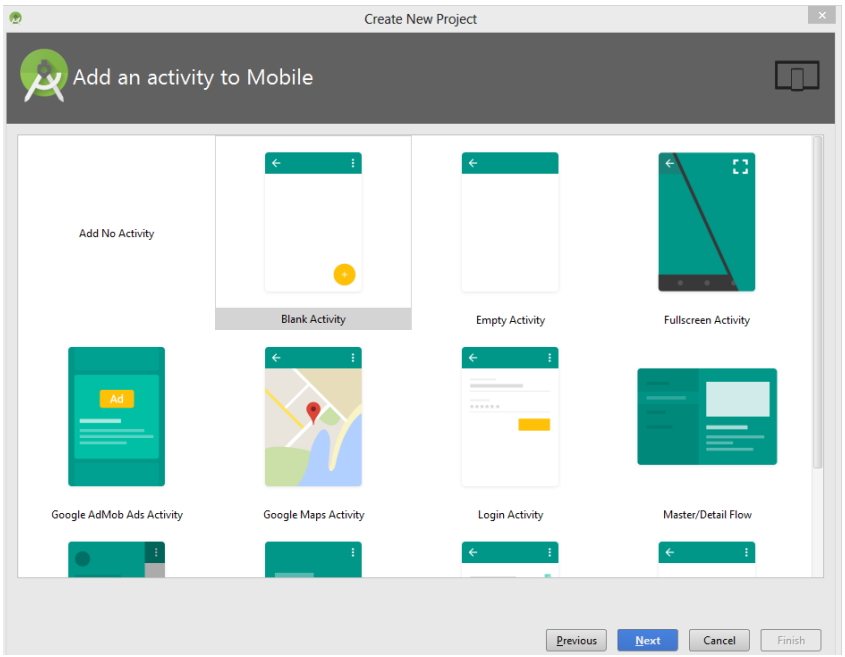

Рисунок 3.4 – Окно выбора базовой Activity

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 3.5., а также имя первого фрагмента, поскольку мы выбрали поддержку фрагментов при создании проекта. Также можно выбрать тип навигации по Activity.

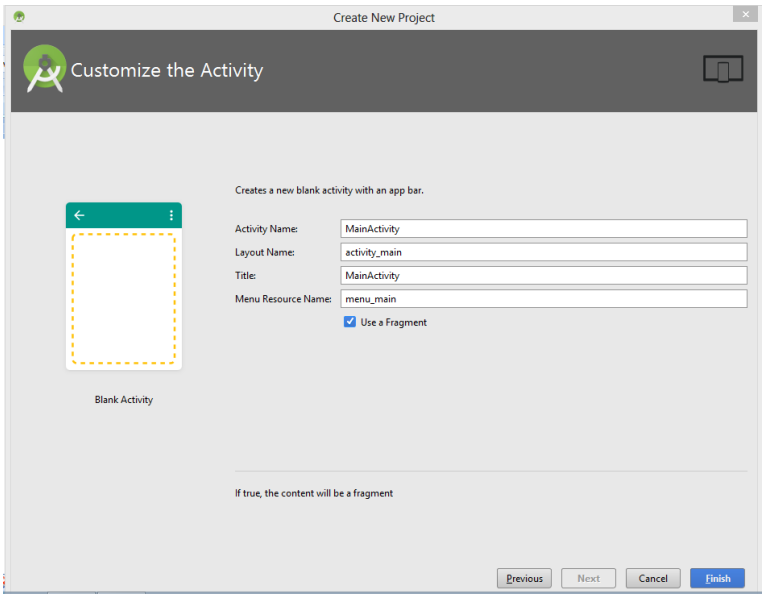

Рисунок 3.5 – Ввод имени нашей стартовой Activity и имени её layout’а

После чего мы переносимся в окно самой IDE. Изначально мы должны были обратить внимание на то, что это структура проекта. Она не такая как в Эклипсе. Так же здесь есть обычные 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 даёт нам управлять зависимостями в проекте и обеспечивает подключение внешних библиотек. Хочу отметить, что при импорте проектов с обычной структурой и эклипс проектов– всё так же отлично работает и компилируется. На рисунке 3.6 изображено основное рабочее окно среды разработки.

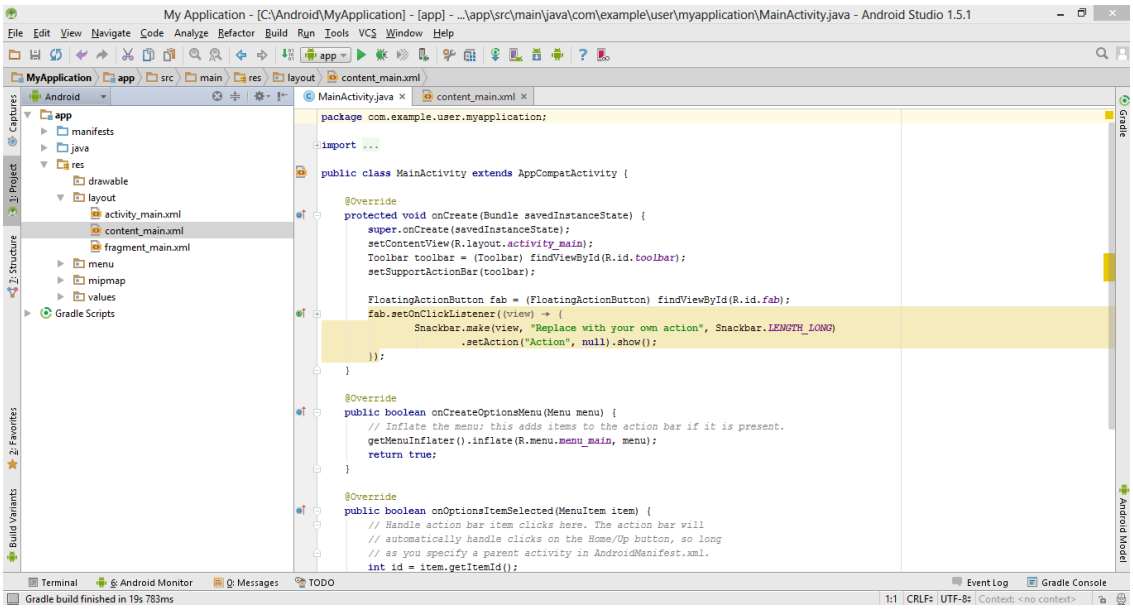

Рисунок 3.6 – Основное окно IDE

Xml редактор, изображённый на рисунке 3.7

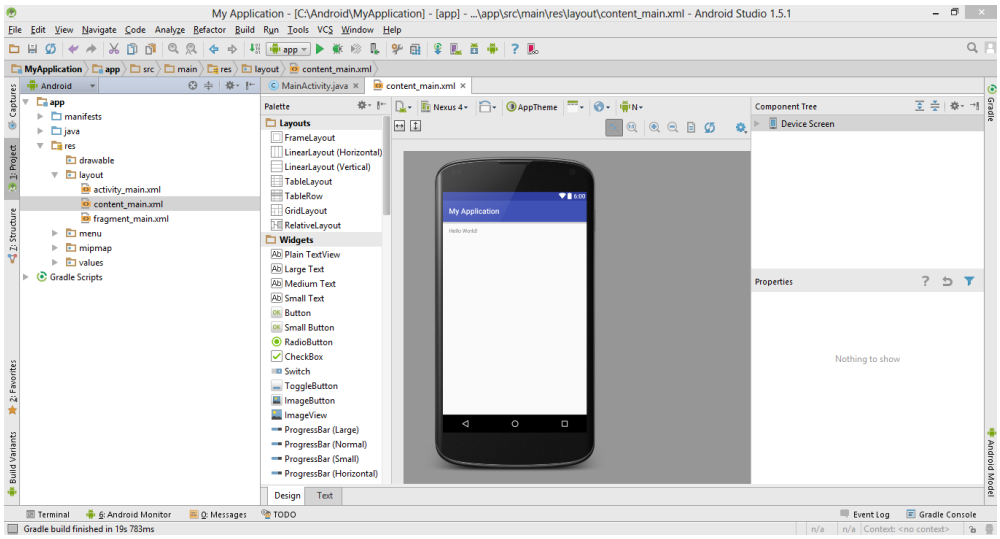

Рисунок 3.7 – Пример окна xml ректора

3.2 Инструменты разработки интерфейса приложения

На самом деле Android не задает каких-либо требований программистам и разработчикам мобильных приложений, о том как именно должны выглядеть мобильные приложения и о их работе. Google является владельцем Android и сразу же задал тренд, что не имеет претензий и особенных требований к внешнему виду. Хотя и существует некий список UI рекомендаций, но он в большинстве своём акцентируются на маленьких вещах, таких как меню или значки.

Со старта платформы имелось столько разных типов внешнего вида приложений, интерфейс имел особенность разнообразия, из-за отсутствия норм и требований. С течением времени платформа Android приобрела некий авторитет на рынке, в результате чего на сегодняшний день сформировывается определенный интерфейс, что я считаю к лучшему. Большинство функций и характеристик Android интерфейса обобщились, есть те, что определили дорогу в библиотеках Android SDK. Таким образом, я предполагаю, что в скором будущем юзеры захотят общую работу от приложений и добьются этого. Также будут свои элементы управления и взаимодействия работы, что заинтегрируются в Android.

В ходе данной дипломной работы я хочу определить, какова деятельность Android UIs на более высшем шаге. Я хочу свести вместе принципы интерфейса и работу Android UIs, чтобы вы наглядно увидели, что я подразумеваю под тем, как должны выглядеть Android приложения.

Новая версия Android (4.0) выпущена недавно, но за такое маленькое время создание произвело крупные пользовательские улучшения Android платформы, еще и самый крупный до этого момента. Так эти изменения влияют на будущий интерфейс и работу Android приложения. Кое-какие обновления встроены вновь для старых версий. Я думаю, что в скором будущем мы будем обозревать большой путь эволюционирования интерфейса ICS, но у нас есть около 200.000.000 девайсов с версиями от 2.1 до

2.3. Существует некий принцип интерфейса, называемый “панель управления”, что имеет спрос среди множества мобильных приложений. К примеру, если содержится одна и более функций в вашем приложении, есть шанс на лучший функционал, где обеспечивается быстрый доступ.

3.2.1. Общий экран приложения

На самом деле Activity проявляется во многих образах, но некоторые особенности стали очень распространяться, что в результате помогло пользователям научиться распознавать и ждать их. Панель активностей в верхней части экрана очень часто используется.

– в верхнем левом углу расположен значок приложения или возврата на главный экран. При нажатии пользователь должен вернуться на главную страницу приложения. Стоит отметить, что новые панели действий не возвращают пользователей на главную страницу, вместо этого совершается переход на один уровень вверх в иерархии меню;

– в средней части панели действий будет расположено название экрана и цвета торговой марки или цветах текущего раздела приложения;

– в правом верхнем углу экрана расположены иконки для наиболее важных действий, которые могут быть выполнены на этом экране. При этом данная часть экрана должна содержать только действия, связанные с содержимым текущего экрана. Функция поиска стала исключением этого правила.

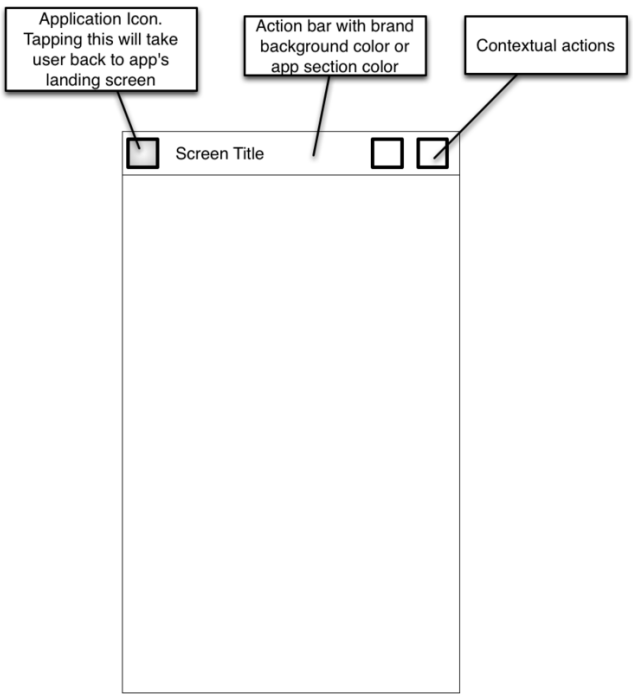

Рис3.1. - Проект ActionBarSherlock (Jake Wharton) облегчяет внедрение панелей действий.

3.2.2. Списки

Списки являются одними из компонентов в широком диапазоне востребованности внешнего интерфейса Android-приложений. Особенностью списков является то, что если неизвестен объем отображаемых данных списков. Но при этом у списков имеется минусы в использовании. Последующий элемент списка должен иметь небольшой размер, то есть должен присутствовать удобное просматривание объема списка. При этом, большое объем данных при маленьком месте для информации является трудной работой в использовании и взаимодействии списков и их поиска. Существуют некоторые директивы деятельности списков Android приложений. Владельцы смартфонов взяли за привычку определенные элементы и функции. Когда список приложения соответствует директивам, то юзерам легче управлять и пользоваться приложениями.

3.2.3. Элементы списка и чекбоксы

Список элементов имеет в своём составе текст и графические элементы. Также есть кнопки на каждом элементе в списке, что используются при нажатии для запуска операций над ними. Удобно было бы расположить чекбокы слева в своём элементе в листе элементов.

Преимущества расположения чекбоксов:

— главное преимущество, конечно, это удобство для глаз, так как мы привыкли к расположению с краю; этот аспект есть во всех программах;

— также чекбокс с левой стороны - эта область обширнее для нажатия;

— это некая видимая подсказка, пользователю легче пролистать лист элементов, то есть где завершается один элемент, там открывается следующий.

3.2.4. Вторичные элементы управления

Некоторым элементам необходимо больше пространства и возможностей взаимодействия, чем простой выбор (чекбокс) или навигация (нажатие). Наиболее распространенным является момент использования этого элемента, управление рейтингом или закрепление элемента в закладки. Единственном местом для вторичного элемента управления является правый край элемента. При любом другом мест произойдут проблемы с расположением. Aldiko и Google Mail являются отличными примерами мобильных приложений, которые владеют приятными списками. Aldiko решили поставить флажки с правой стороны, что визуально улучшает и делает пользовательский интерфейс несбалансированным.

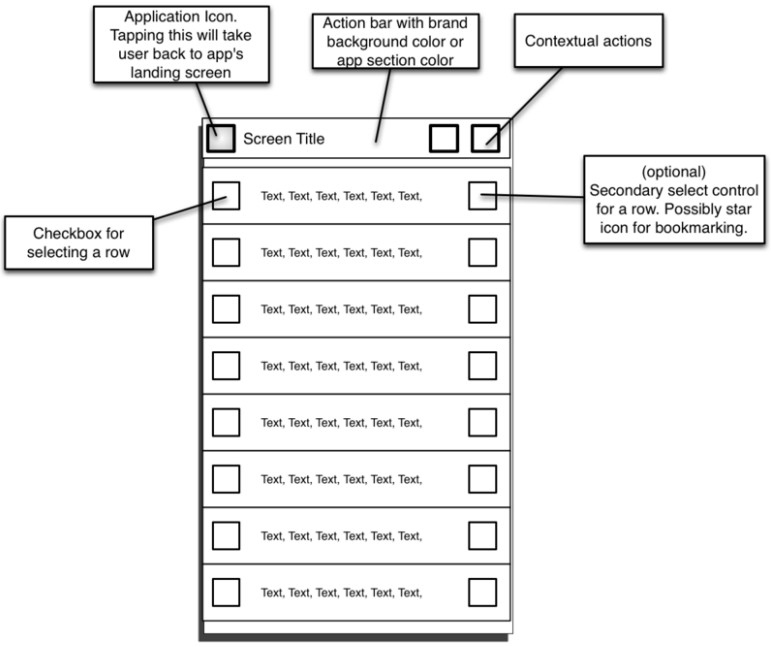

Рис.3.2. - расположение вторичных элементов управления

3.2.5. Бесконечные списки

Еще одними основополагающими в принципе внешнего вида мобильных приложений являются списки, которые в большинстве своём имеют кое-какие элементы, что открываются через интернет. Если загрузка длится долго, что в результате списки не заполняются вовремя. Если юзер дошёл до конца списков, мобильное приложение обязано загружать последующие элементы списков. Так, индикатор обращается к пользователю и объясняет, что эти последующие загруженные элементы переходят в концу списка. К этому относятся также несколько видов загрузки анимации. Анимация дает понять пользователям, что идёт подгрузка данных. Например, Android Market и Twitter автоматически загружают несколько элементов при достижении конца списка.

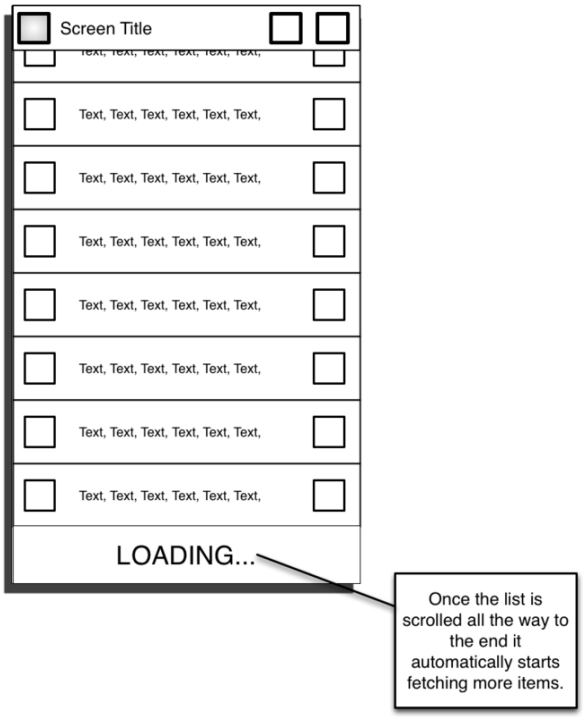

Рис.3.3. - пример загрузки бесконечных списков

3.2.6. Действия над несколькими элементами

Выбор пользователем нескольких элементов позволяется, если список содержит элемент управления чекбокс. Тогда юзер имеет возможность производить несколько выделенных действий. В результате, добавлили панели внизу экрана, где есть чекбоксы для будущих действий сразу над выделенными элементами. Особенностью является то, что анимация с четким скольжением придаёт интерфейсу гибкость и изысканность. Когда снимается конечное выделение или действие выполнено, то панель должна автоматически исчезать. Aldiko и GMail являются отличными примерами, что владеют отличную плавную анимацию. Aldiko показывает юзеру количество выбранных элементов. Это одно из очень хороших аспектов этих приложений.

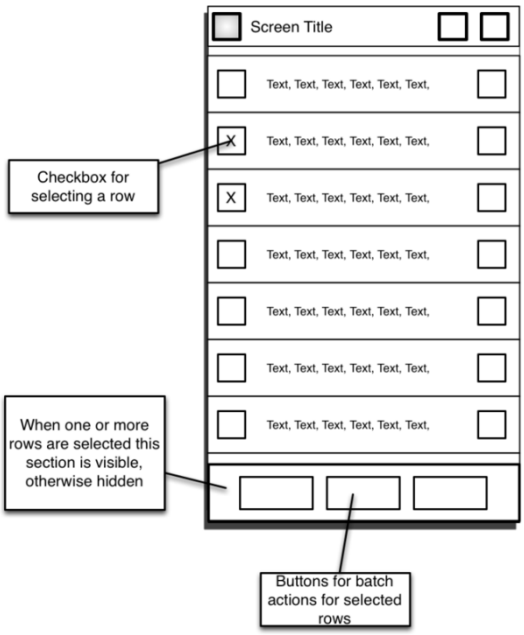

Рис.3.4.- пример интерфейса с действиями над несколькими элементами

3.2.7. Вкладки

Большинство мобильных приложений в различной форме используют вкладки, чтобы пользователи могли свободно перемещаться между страницами. Android версии Honeycomb (3,0) и Ice Cream Sandwich (4,0) внесли немного изменений в способе работы и во внешнем вид вкладок. Я считаю, что выбор нововведенных вкладок в действующих мобильных приложениях необходим. Так тема введения изменений вкладок переходит в новые ожидания пользователей вашего приложения, что есть функция перемещения среди вкладок. Так отличными примерами мобильных приложений, что имеют вышеописанную функцию, а точнее слайдинг для навигации среди вкладок являются Android Market и Google+.

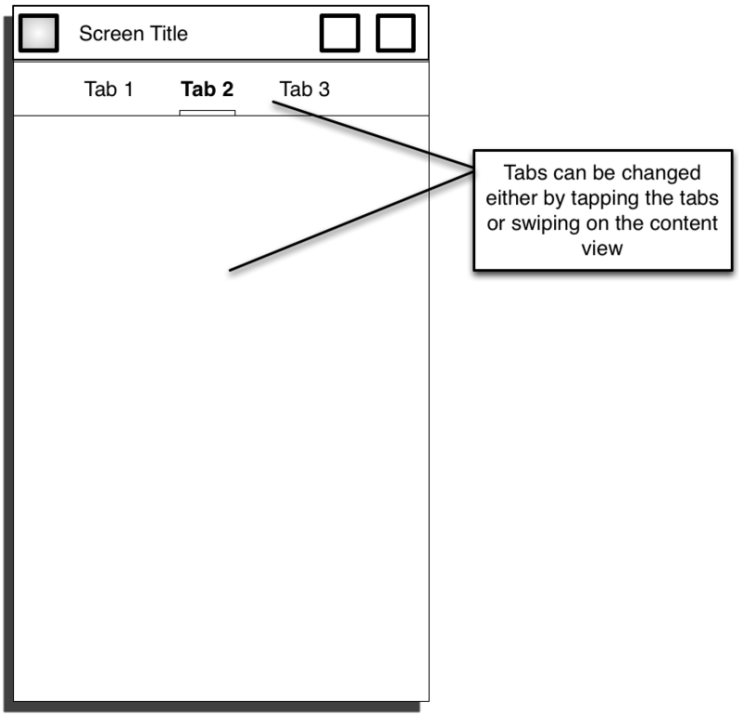

Рис.3.5. - использование вкладок в мобильных приложениях

На сегодняшний день, Android быстро набирает темп и становится популярной платформой, и имеет уже зрелые взгляды и опыт, в результате чего интерфейс и работа мобильных приложений постепенно вливается в новое русло и становиться похожими друг на друга, таким образом, и пользователи начинают ожидать определенное взаимодействие с пользовательским интерфейсом. Естественно пока нет особенных требований и направлений, но видение из далека на популярные приложения определяет наше сознание, что мы должны воспроизводить в будущем в разработке интерфейса и поведения мобильного приложения.

4 Этапы разработки приложения

4.1 Проектирование

Sunshine – это мобильное приложение, в основе которого лежат определения "открытые данные", "API", которое обрабатывает, анализирует и, редактируя, выводит результат в виде погодных условий и их характеристик на ближайшее время. Так же имеется функция как "поделиться", вы можете поделиться приложением в других социальных сетях. Так же есть локация, приложение выводит прогноз погоды для города Алматы. Моё мобильное приложение имеет очень простой и удобный интерфейс. Внешний вид приложения должен соответствовать всем элементам оформления пользовательского интерфейса операционной системы, являясь мостом и передавая опыт использования, что задуман разработчиками платформы. Дизайн должен строго соответствовать установленным нормам и требованиям платформ обладателя (в данном случае Google). Используя все

вышеописанные нормы, я разработала интерфейс приложения Sunshine. Интерфейс мобильного приложения представлен в виде главного toolbar-a, который имеет два перехода: главное меню и меню настроек и обновления. Интерфейс меню приложения представлен пользователям экранным табло на котором размещен список текущего и ближайших дней с обработанным прогнозом погоды находящихся на нашем смартфоне (Рисунок 4.1).

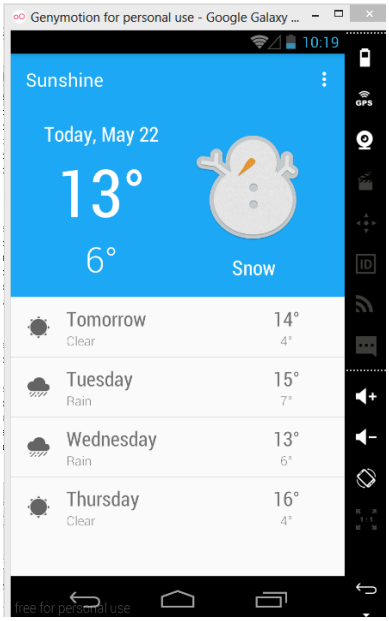

Рисунок 4.1 – Главное меню приложения на эмуляторе Android

Можно переходить по вкладкам на каждый нужный день, где описаны характеристики текущей погоды: градусы дня и ночи, влажность, скорость ветра, давление на рисунке 4.2.

Рисунок 4.2 – Переход по вкладкам

В век развития мобильного интернета и социальных сетей, просто необходима кнопка "Поделиться", переходя по которой можно поделиться с друзьями новым приложением в используемой социальной сети. Данная функция просто незаменима и удобна в современных программах, так как значительно уменьшает время на поиск нужного файла, что изображена на рисунке 4.3

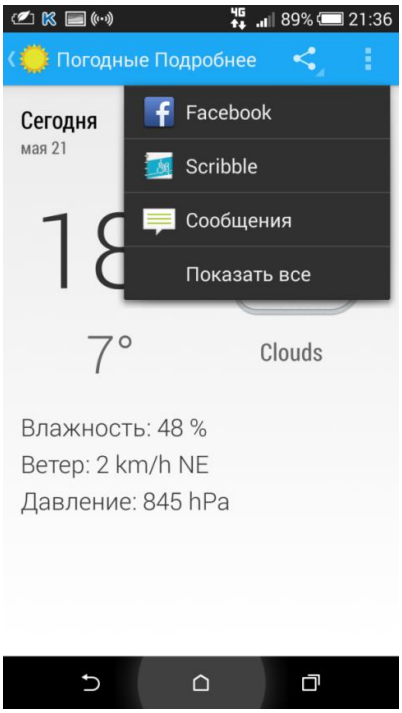

Рисунок 4.3 – Функция "Поделиться"

Как выглядит иконка приложения на рисунке 4.4

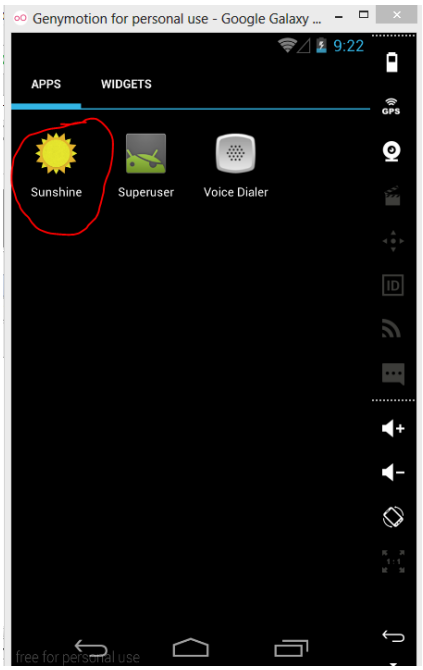

Рисунок 4.4 – Иконка приложения

В разделе «Settings» находятся основные требуемые настройки для среднестатистического пользователя, где можно выбрать город, пока только Алматы, но я надеюсь в дальнейшей разработке можно увеличить количество городов. Так же пользователь может выбрать единицу измерения температуры, которая удобна для него. А так же подсказки для пользователя, и информация о приложении (Рисунок 4.5).

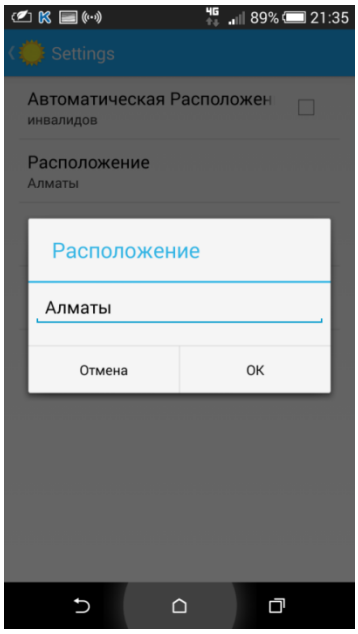

Рисунок 4.5 – Настройки в приложении

4.2 Разработка приложения

Данный проект содержит несколько папок и файлов.

Основные из них:

- src – исходный код на Java, где находится главный файл для работы. Здесь же и будут находиться и создаваться новые классы;
- gen – файлы сгенерированные самой Java;
- res – файлы ресурсов. Содержит несколько подкаталогов:
    1. res/drawable-*dpi – в этих пяти папках лежат ресурсы, для разных расширений экрана. При входе в каждую из них, можно увидеть файл ic_launcher.png, что является иконкой вашего приложения. В папке drawable-ldpi нет файлов, потому что данная папка для устаревших телефонов, которые уже не поддерживаются;
    2. res/layout – в данной папке содержатся xml-файлы, которые определяют интерфейс форм и элементов форм. После завершения создания проекта там уже имеются файлы activity_main.xml и fragment_main.xml. res/menu – здесь находятся ресурсы для меню;
    3. res/values – здесь находятся разные строковые ресурсы, так же ресурсы цветов, тем, стилей и измерений, которые мы используем в дипломном проекте. К тому же есть похожие с ним папки values-

w820dp, values-v11 (для планшетов Android 3.0), values-14 (для Android 4) предназначенные для определенных видов девайса.

4.2.1 Создание и реализация форм

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

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="kz.learndroid.app.sunshine.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

</android.support.design.widget.CoordinatorLayout>
```

Форма отображения менюбара реализовывалось в файле activity_detail.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="kz.learndroid.app.sunshine.DetailActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />
```

```
    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_detail" />

</android.support.design.widget.CoordinatorLayout>
```

4.2.2 Создание и реализация классов

В процессе разработки были созданы и реализованы множество классов:

- DetailActivity – подтверждение сохранения файла;
- DetailActivityFragment – сохранение файла;
- FetchWeatherTask– реализация маркеров;
- ForecastAdapter– редактирование файла;
- ForecastFragment – выбор файла;
- MainAtivity – чтение погодных файлов;
- Utility – реализация формы сигнала файла и несколько незначительных классов.
- WeatherContract– работа с базой данных;
- WeatherDbHelper–работа с базой даных;

```
package kz.learndroid.app.sunshine.data;
import android.provider.BaseColumns;
import android.text.format.Time;
public class
        WeatherContract {
    public static long normalizeDate(long startDate) {
        // normalize the start date to the beginning of the (UTC) day
        Time time = new Time();
        time.set(startDate);
        int julianDay = Time.getJulianDay(startDate, time.gmtoff);
        return time.setJulianDay(julianDay);
    }
    public static final class LocationEntry implements BaseColumns {
        public static final String TABLE_NAME = "location";
        public static final String COLUMN_LOCATION_SETTING =
"location_setting";
        public static final String COLUMN_CITY_NAME = "city_name";
        public static final String COLUMN_COORD_LAT = "coord_lat";
        public static final String COLUMN_COORD_LONG = "coord_long";
    }
    public static final class WeatherEntry implements BaseColumns {
        public static final String TABLE_NAME = "weather";
        public static final String COLUMN_LOC_KEY = "location_id";
        public static final String COLUMN_DATE = "date";
        public static final String COLUMN_WEATHER_ID = "weather_id";
        public static final String COLUMN_SHORT_DESC = "short_desc";
        public static final String COLUMN_MIN_TEMP = "min";
        public static final String COLUMN_MAX_TEMP = "max";
        public static final String COLUMN_HUMIDITY = "humidity";
        public static final String COLUMN_PRESSURE = "pressure";
        public static final String COLUMN_WIND_SPEED = "wind";
        public static final String COLUMN_DEGREES = "degrees";
    }
}

package kz.learndroid.app.sunshine.data;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
```

```
import android.database.sqlite.SQLiteOpenHelper;
import kz.learndroid.app.sunshine.data.WeatherContract.LocationEntry;
import kz.learndroid.app.sunshine.data.WeatherContract.WeatherEntry;
public class WeatherDbHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 2;
    static final String DATABASE_NAME = "weather.db";
    public WeatherDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        final String SQL_CREATE_LOCATION_TABLE = "CREATE TABLE " +
LocationEntry.TABLE_NAME + " (" +
                LocationEntry._ID + " INTEGER PRIMARY KEY," +
                LocationEntry.COLUMN_LOCATION_SETTING + " TEXT UNIQUE
NOT NULL, " +
                LocationEntry.COLUMN_CITY_NAME + " TEXT NOT NULL, " +
                LocationEntry.COLUMN_COORD_LAT + " REAL NOT NULL, " +
                LocationEntry.COLUMN_COORD_LONG + " REAL NOT NULL " +
                " );";
        final String SQL_CREATE_WEATHER_TABLE = "CREATE TABLE " +
WeatherEntry.TABLE_NAME + " (" +
                WeatherEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
+
                WeatherEntry.COLUMN_LOC_KEY + " INTEGER NOT NULL, " +
                WeatherEntry.COLUMN_DATE + " INTEGER NOT NULL, " +
                WeatherEntry.COLUMN_SHORT_DESC + " TEXT NOT NULL, " +
                WeatherEntry.COLUMN_WEATHER_ID + " INTEGER NOT NULL," +
                WeatherEntry.COLUMN_MIN_TEMP + " REAL NOT NULL, " +
                WeatherEntry.COLUMN_MAX_TEMP + " REAL NOT NULL, " +
                WeatherEntry.COLUMN_HUMIDITY + " REAL NOT NULL, " +
                WeatherEntry.COLUMN_PRESSURE + " REAL NOT NULL, " +
                WeatherEntry.COLUMN_WIND_SPEED + " REAL NOT NULL, " +
                WeatherEntry.COLUMN_DEGREES + " REAL NOT NULL, " +
                // Set up the location column as a foreign key to
location table.
                " FOREIGN KEY (" + WeatherEntry.COLUMN_LOC_KEY + ")
REFERENCES " +
                LocationEntry.TABLE_NAME + " (" + LocationEntry._ID +
"), " +

per day
               REPLACE strategy
                " UNIQUE (" + WeatherEntry.COLUMN_DATE + ", " +
                WeatherEntry.COLUMN_LOC_KEY + ") ON CONFLICT REPLACE);";
        sqLiteDatabase.execSQL(SQL_CREATE_LOCATION_TABLE);
        sqLiteDatabase.execSQL(SQL_CREATE_WEATHER_TABLE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int oldVersion,
int newVersion) {

        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " +
LocationEntry.TABLE_NAME);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " +
WeatherEntry.TABLE_NAME);
        onCreate(sqLiteDatabase);
    }
}
```

```
package kz.learndroid.app.sunshine;
import android.content.ContentUris;
import android.content.ContentValues;
```

```
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.AsyncTask;
import android.support.v4.app.FragmentActivity;
import android.text.format.Time;
import android.util.Log;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.SimpleDateFormat;
import kz.learndroid.app.sunshine.data.WeatherContract;
import kz.learndroid.app.sunshine.data.WeatherDbHelper;
public class FetchWeatherTask extends AsyncTask<String, Void, String> {
    Context mContext;
    public FetchWeatherTask(Context context) {
        mContext = context;
    }
    long addLocation(String locationSetting, String cityName, double lat,
double lon) {
        long locationId;

        WeatherDbHelper dbHelper = new WeatherDbHelper(mContext);
        SQLiteDatabase db = dbHelper.getReadableDatabase();

        Cursor locationCursor = db.query(
                WeatherContract.LocationEntry.TABLE_NAME,
                new String[]{WeatherContract.LocationEntry._ID},
                WeatherContract.LocationEntry.COLUMN_LOCATION_SETTING + " =
?",
                new String[]{locationSetting},
                null,
                null,
                null
        );
        if (locationCursor.moveToFirst()) {
            int locationIdIndex =
locationCursor.getColumnIndex(WeatherContract.LocationEntry._ID);
            locationId = locationCursor.getLong(locationIdIndex);
        } else {
            ContentValues locationValues = new ContentValues();

locationValues.put(WeatherContract.LocationEntry.COLUMN_CITY_NAME, cityName);

locationValues.put(WeatherContract.LocationEntry.COLUMN_LOCATION_SETTING,
locationSetting);
            locationValues.put(WeatherContract.LocationEntry.COLUMN_COORD_LAT,
lat);

locationValues.put(WeatherContract.LocationEntry.COLUMN_COORD_LONG, lon);

            db = dbHelper.getWritableDatabase();
            locationId = db.insert(
                    WeatherContract.LocationEntry.TABLE_NAME,
                    null,
                    locationValues
```

```
            );
        }
        locationCursor.close();
        return locationId;
    }
    @Override
    protected String doInBackground(String... params) {
        Log.d("AsyncTask", "Запущен");
        String city = params[0];
        if(city == null) {
            return null;
        }
        HttpURLConnection urlConnection = null;
        BufferedReader reader = null;
        // Will contain the raw JSON response as a string.
        String forecastJsonStr = null;
        try {
            final String FORECAST_BASE_URL =
"http://api.openweathermap.org/data/2.5/forecast/daily?";
            final String QUERY_PARAM = "q";
            final String FORMAT_PARAM = "mode";
            final String UNITS_PARAM = "units";
            final String DAYS_PARAM = "cnt";
            final String APPID_PARAM = "APPID";
            Uri builtUri = Uri.parse(FORECAST_BASE_URL).buildUpon()
                    .appendQueryParameter(QUERY_PARAM, city)
                    .appendQueryParameter(FORMAT_PARAM, "json")
                    .appendQueryParameter(UNITS_PARAM, "metric")
                    .appendQueryParameter(DAYS_PARAM, "7")
                    .appendQueryParameter(APPID_PARAM,
"ffd2bf2379705399609ab1651de867b9")
                    .build();
            URL url = new URL(builtUri.toString());
            Log.v("Наш URL", builtUri.toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.connect();
            InputStream inputStream = urlConnection.getInputStream();
            StringBuffer buffer = new StringBuffer();
            if (inputStream == null) {
                // Nothing to do.
                return null;
            }
            reader = new BufferedReader(new InputStreamReader(inputStream));

            String line;
            while ((line = reader.readLine()) != null) {
                buffer.append(line + "\n");
            }
            if (buffer.length() == 0) {
                // Stream was empty.  No point in parsing.
                return null;
            }
            forecastJsonStr = buffer.toString();
            Log.d("Наша погода", forecastJsonStr);
            getWeatherDataFromJson(forecastJsonStr, city);
        } catch (IOException e) {
            //Log.e("FragmentException", e.getMessage());
            e.printStackTrace();
            return null;
        } catch (JSONException e) {
            e.printStackTrace();
        } finally {
```

```
            if (urlConnection != null) {
                urlConnection.disconnect();
            }
            if (reader != null) {
                try {
                    reader.close();
                } catch (final IOException e) {
                    Log.e("ForecastFragment", "Error closing stream", e);
                }           }
        }
        return "Ok";
    }
    private String getReadableDateString(long time){
    SimpleDateFormat shortenedDateFormat = new SimpleDateFormat("EEE MMM dd");
        return shortenedDateFormat.format(time);
    }
        private String formatHighLows(double high, double low) {
        long roundedHigh = Math.round(high);
        long roundedLow = Math.round(low);
        String highLowStr = roundedHigh + "/" + roundedLow;
        return highLowStr;
    }
    private void getWeatherDataFromJson(String forecastJsonStr, String
locationSetting)
            throws JSONException {
        final String OWM_LIST = "list";
        final String OWM_WEATHER = "weather";
        final String OWM_TEMPERATURE = "temp";
        final String OWM_MAX = "max";
        final String OWM_MIN = "min";
        final String OWM_DESCRIPTION = "main";
        final String OWM_WEATHER_ID = "id";
        final String OWM_PRESSURE = "pressure";
        final String OWM_HUMIDITY = "humidity";
        final String OWM_WINDSPEED = "speed";
        final String OWM_WIND_DIRECTION = "deg";
        final String OWM_CITY = "city";
        final String OWM_CITY_NAME = "name";
        final String OWM_COORD = "coord";
        final String OWM_LATITUDE = "lat";
        final String OWM_LONGITUDE = "lon";
        JSONObject forecastJson = new JSONObject(forecastJsonStr);
        JSONArray weatherArray = forecastJson.getJSONArray(OWM_LIST);
        JSONObject cityJson = forecastJson.getJSONObject(OWM_CITY);
        String cityName = cityJson.getString(OWM_CITY_NAME);
        JSONObject cityCoord = cityJson.getJSONObject(OWM_COORD);
        double cityLatitude = cityCoord.getDouble(OWM_LATITUDE);
        double cityLongitude = cityCoord.getDouble(OWM_LONGITUDE);
        long locationId = addLocation(locationSetting, cityName, cityLatitude,
cityLongitude);
        Time dayTime = new Time();
        dayTime.setToNow();
              int julianStartDay =
Time.getJulianDay(System.currentTimeMillis(), dayTime.gmtoff);
        dayTime = new Time();
        WeatherDbHelper dbHelper = new WeatherDbHelper(mContext);
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        for(int i = 0; i < weatherArray.length(); i++) {
            long dateTime;
            double pressure;
            int humidity;
            double windSpeed;
            double windDirection;
```

```
            double high;
            double low;
            String description;
            int weatherId;
            JSONObject dayForecast = weatherArray.getJSONObject(i);
            pressure = dayForecast.getDouble(OWM_PRESSURE);
            humidity = dayForecast.getInt(OWM_HUMIDITY);
            windSpeed = dayForecast.getDouble(OWM_WINDSPEED);
            windDirection = dayForecast.getDouble(OWM_WIND_DIRECTION);
            dateTime = dayTime.setJulianDay(julianStartDay + i);
            //day = getReadableDateString(dateTime);
            JSONObject weatherObject =
dayForecast.getJSONArray(OWM_WEATHER).getJSONObject(0);
            description = weatherObject.getString(OWM_DESCRIPTION);
            weatherId = weatherObject.getInt(OWM_WEATHER_ID);

            // Temperatures are in a child object called "temp".  Try not to
name variables
            // "temp" when working with temperature.  It confuses everybody.
            JSONObject temperatureObject =
dayForecast.getJSONObject(OWM_TEMPERATURE);
            high = temperatureObject.getDouble(OWM_MAX);
            low = temperatureObject.getDouble(OWM_MIN);

            ContentValues weatherValues = new ContentValues();

            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_LOC_KEY,
locationId);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_DATE,
dateTime);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_HUMIDITY,
humidity);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_PRESSURE,
pressure);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_WIND_SPEED,
windSpeed);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_DEGREES,
windDirection);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_MAX_TEMP,
high);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_MIN_TEMP,
low);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_SHORT_DESC,
description);
            weatherValues.put(WeatherContract.WeatherEntry.COLUMN_WEATHER_ID,
weatherId);

            long insertId = db.insert(
                    WeatherContract.WeatherEntry.TABLE_NAME,
                    null,
                    weatherValues
            );

            Log.d("ID новой погоды", insertId + "");
        }
    }

}
import android.accounts.AbstractAccountAuthenticator;
import android.accounts.Account;
import android.accounts.AccountAuthenticatorResponse;
import android.accounts.NetworkErrorException;
import android.content.Context;
```

```
import android.os.Bundle;
public class SunshineAuthenticator extends AbstractAccountAuthenticator {
    public SunshineAuthenticator(Context context) {
        super(context);
    }
    @Override
    public Bundle editProperties(
            AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }
        @Override
    public Bundle addAccount(
            AccountAuthenticatorResponse r,
            String s,
            String s2,
            String[] strings,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }
       @Override
    public Bundle confirmCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            Bundle bundle) throws NetworkErrorException {
        return null;
    }
        public Bundle getAuthToken(
            AccountAuthenticatorResponse r,
            Account account,
            String s,
            Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }    @Override
    public String getAuthTokenLabel(String s) {
        throw new UnsupportedOperationException();
    }
    @Override
    public Bundle updateCredentials(
            AccountAuthenticatorResponse r,
            Account account,
            String s, Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }
    @Override
    public Bundle hasFeatures(
            AccountAuthenticatorResponse r,
            Account account, String[] strings) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }
}
```

5 Экономическое обоснование проекта

5.1Описание работы и обоснование необходимости

Целью данной дипломной работы является создание и разработка мобильного приложения на платформе Android, обрабатывающего и анализирующего открытые данные в виде характеристики погодных условий.

Таким образом, данное мобильное приложение делает некие операции над открытыми данными, предоставляет погодные условия. Так же оно имеет

возможность сохранять полученные данные после последнего сеанса, что в результате может работать без подключения к интернету – это новшество в современном мире и в мире мобильных технологий, которые постоянно меняются. Приложение позволяет человеку в любой момент узнать погодные условия, их характеристику, так же погоду на ближайшее время.

5.2 Трудовые ресурсы, используемые в работе.

В данном проекте используется интеллектуальный труд, расходы которого выше, чем у физического труда.

В разработке приняли участие:

– дизайнер - разработка интерфейса;
– программист - разработка алгоритмов и программирование;
– Руководитель – оформление и проверка отчета.

Количество сотрудников задействованных в разработке представлено в таблице 5.1.

Таблица 5.1 – Сотрудники и их заработная плата

Должность	Количество человек	Заработная плата в месяц, тенге
Инженер-разработчик	1	90000
Дизайнер	1	60000
Руководитель проекта	1	100000
Итого	3	250000

5.3 Оборудование, используемое в работе

Оборудование, используемое при разработке сайта представлено в таблице 5.2.

Таблица 5.2 – Перечень оборудования, необходимого для разработки

Наименование изделий	Характеристика	Количество единиц	Цена за единицу, тенге	Общая сумма,тенге
Ноутбук	Acer Aspire E1-572G Intel Core i7-5500U 8Gb DDR3 ATM Radeon HD 8750M 2GB HDD 1000Gb 15.6" LED LCD	1	110000	110000
Смартфон	HTC one mini2	1	70000	70000
Итого				180000

Цены на оборудование представлены без учета НДС.

5.4 Программное обеспечение, используемое в работе

При разработке мобильного приложения было использовано следующее программное обеспечение:

- Windows 8 – операционная система;
- Android Studio – платформа для Android-приложений;
- Genymotion – эмулятор Android;
- Mockingbot – инструментарий для прототипа приложения.

Программное обеспечение, использованное при разработке приложения, представлено в таблице 5.3.

Таблица 5.3 – перечень программного обеспечения, необходимого для разработки мобильного приложения

Программное обеспечение	Стоимость, тенге
Windows 8	22000
Android Studio	бесплатно
Genymotion	бесплатно
Mockingbot	бесплатно
Итого	22000

Цены на ПО приведены без учета НДС.

5.5 Сроки реализации проекта

Процесс разработки и сроки реализации Android-приложения состоит из 6 этапов и включает в себя:

- создание прототипа приложения;
- проектирование и разработка приложения;
- программирование;
- разработка дизайна;
- тестирование созданного приложения;
- оформление отчетов.

5.6 Расчет стоимости работы по разработке

Расчет стоимости работы по разработке - это наиболее важная часть экономического анализа, так как на основе этого расчета определяются затраты рабочего времени на разработку проекта на каждом этапе, а также трудовые затраты.

Затраты на разработку данного проекта определяются по формуле

$$С = ФОТ + О_с + А + Э + С_{пр} + Н \quad (5.1)$$

где ФОТ - фонд оплаты труда;

$О_с$ - социальный налог;

А - амортизационные отчисления;

Э - затраты на электроэнергию;

$С_{пр}$ - прочие расходы;

Н - накладные расходы.

5.6.1 Расчет затрат на оплату труда

Затраты на оплату труда персонала, задействованного в разработке проекта, рассчитываются по формуле:

$$ФОТ = З_{осн} + З_{доп} \quad (5.2)$$

где $З_{осн}$ -основная заработная плата;

$З_{доп}$ - дополнительная заработная плата.

Труд инженера-разработчика, дизайнера и руководителя принят условно, на договорной основе в размере 90000,60000 и 100000 тенге соответственно.

На этапах разработки программного продукта участники разработки сайта задействованы неравноценно, для этого необходимо рассчитать средний дневной заработок, а затем и общий размер заработной платы, в зависимости от их фактического участия.

Средний дневной заработок каждого работника рассчитывается по формуле:

$$D = \frac{ЗПм}{Др} \quad (5.3)$$

где $ЗП_м$– ежемесячный размер заработной платы;

$Д_р$ – количество рабочих дней в месяце (это 26 дней – шестидневная рабочая неделя).

1) Для инженера-разработчика

$$D = \frac{90000}{26} = 3461{,}5 \text{ тенге/день}$$

2) Для дизайнера

$$D = \frac{60000}{26} = 2307{,}6 \text{ тенге/день}$$

3) Для руководителя проекта

$$D = \frac{100000}{26} = 3846{,}15 \text{ тенге}$$

Заработная плата за один час вычисляется по формуле:

$$Н = \frac{ЗПм}{Др * Чр} \quad (5.4)$$

где $ЗП_м$– ежемесячный размер заработной платы;

$Д_р$ – количество рабочих дней в месяце.

$Ч_р$ – Количество часов рабочего дня ($Ч_р$ = 8)

1) Для инженера-разработчика

$$Н = \frac{90000}{26*8} = 432,7 \text{ тенге/час}$$

2) Для дизайнера

$$Н = \frac{60000}{26*8} = 288,5 \text{ тенге/час}$$

3) Для руководителя

$$Н = \frac{100000}{26*8} = 480,8 \text{ тенге/час}$$

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

$$t_n = \frac{T}{q_n * z * K} \quad (5.5)$$

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_nокругляем в большую сторону до целых дней.

$t_1 = \frac{24}{1*8*1,1} \approx 3$ - инженер-разработчик, постановка задачи;

Для каждого вида работ произведены аналогичные расчеты.

$$t_n = 4+5+4+6+7+7+7+10+2+2+3+3 = 60 \text{ дней}$$

Так, для произведения всех работ необходимо 60 дней, что составляют 2месяца.

Сводные данные по расчету заработной платы персонала, задействованного в разработке проекта приведены в таблице 5.5.

Таблица 5.5 – Сводные данные по расчету основной заработной платы персонала задействованного в разработке проекта.

Наименование этапов	Исполнитель	Трудоемкость, норма-час	Длительность цикла, дни	Заработная плата за час работы, тенге	Сумма заработной платы, тенге
Постановка задачи	Инженер-разработчик	32	4	432,7	13846,4
Разработка прототипа мобильного приложения	Инженер-разработчик	40	5	432,7	17308
Подбор и изучение литературы	Инженер-разработчик	32	4	432,7	13846,4
Разработка дизайна мобильного приложения	Дизайнер	48	6	288,5	13848
Проектирование приложения	Инженер-разработчик	56	7	432,7	24231,2
Разработка приложения	Инженер-разработчик	56	7	432,7	24231,2
Создание дизайна на Android Studio	Дизайнер	56	7	288,5	16156
Программирование приложения	Инженер-разработчик	80	10	432,7	34616
Тестирование ПО	Инженер-разработчик	16	2	432,7	6923,2
Отладка ПО	Инженер-разработчик	16	2	432,7	6923,2
Оформление НИР	Руководитель	24	3	480,8	11539,2
Проверка и сдача отчета	Руководитель	24	3	480,8	11539,2
Итого		480	60	4423,2	195008,0

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

$$З_{доп} = З_{осн} * 0,1 \quad (5.6)$$

и составит:

$З_{доп}$ = 195008* 0,1 = 19500,8 тенге;

Таким образом, затраты на оплату труда согласно произведенным расчетам и в соответствии с формулой 4.2 составит:

ФОТ = 195008,0+ 19500,8 =214508,8 тенге

5.6.2 Расчет затрат по социальному налогу

Социальный налог составляет 11% (ст. 358 п. 1 НК РК) от дохода работника, и рассчитывается по формуле

$$О_с = (ФОТ - ПО) * 11\% \quad (5.7)$$

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

$$ПО = ФОТ * 10\% \quad (5.8)$$

ПО = 214508,8 * 0,1 = 21450,88 тенге.

Таким образом, в соответствии с произведенными расчетами и согласно формуле 4.7размер отчислений на социальные нужды составит:

$О_с$ = (214508,8 - 21450,88) * 0,11= 21236,4 тенге

5.6.3 Расчет амортизационных отчислений

Амортизационные отчисления рассчитываются по формуле

$$A_i = \frac{H_A * C_{ПЕР} * N}{100 * n} \quad (5.9)$$

где $Н_А$ – норма амортизации;

$С_{ПЕР}$ – первоначальная стоимость оборудования;

N – количество дней на выполнение работ;

n – количество рабочих дней в году.

Норма амортизации на компьютерную технику составляет 40% от стоимости всего оборудования, на программное обеспечение - 15%.

Таким образом, амортизационный отчисления по используемому оборудованию, в соответствии с формулой 4.9 составят

- на ноутбук

$$A_1 = \frac{40 * 110000 * 60}{100 * 365} = 7232{,}87 \text{ тенге}$$

- на многофункциональное устройство

$$A_2 = \frac{40 * 70000 * 4}{100 * 365} = 306{,}8 \text{ тенге}$$

- на программное обеспечение

$$A_3 = \frac{15 * 22000 * 60}{100 * 365} = 542{,}46 \text{ тенге}$$

$$A_i = 7232{,}87 + 306{,}8 + 542{,}46 = 8082{,}13 \text{ тенге}$$

Сводные результаты расчета амортизационных отчислений представлены в таблице 5.6.

Таблица 5.6 – Сводные данные по расчету затрат на амортизацию

Наименование оборудования	Количество	Норма амортизации, %	Сумма амортизации, тенге
Ноутбук	1	40	7232,87
Многофункциональное устройство	1	40	306,8
Программное обеспечение	-	15	542,46
Итого			8082,13

5.6.4 Расчет затрат на электроэнергию

Нужно учитывать, что в процессе производства используется электрооборудование, поэтому необходимо рассчитать затраты на электроэнергию. Затраты на электроэнергию для производственных нужд включают в себя расходы электроэнергии на оборудование и дополнительные нужды. И рассчитываются по формуле

$$Э = З_{эл.эн.об.} + З_{доп.} \quad (5.10)$$

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

$З_{доп.}$ – затраты на электроэнергию для дополнительных нужд.

Расходы электроэнергии для оборудования рассчитываются по формуле

$$З_{эл.эн.об.} = W*T*S*К_{исп} \quad (5.11)$$

где W – потребляемая мощность, Вт;

T – время работы, часы;

S – тариф(1кВт = 16,02 тенге);

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

$З_{эл.эн.об.(ноутбук)}$ = 0,7 * 480 * 16,02 * 0,9 = 4844,5 тенге

$З_{эл.эн.об.(МФУ)}$ = 0,9 * 32 * 16,02 * 0,9 = 415,24 тенге

Общая сумма затрат на электроэнергию основного оборудования согласно формуле 5.11 составляет

$З_{эл.эн.об.}$ = 4844,5+ 415,24 = 5259,7 тенге

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

$$З_{доп} = З_{эл.эн.об.}*5\% \quad (5.12)$$

и составляют

$З_{доп}$ = 5259,7* 0,05 = 263 тенге

Таким образом, суммарные затраты на электроэнергию, согласно формуле 4.11 составляют

Э = 5259,7+ 263 = 5522,7 тенге

Сводные результаты расчета затрат на электроэнергию представлены в таблице 5.7

Таблица 5.7 – Сводные данные о затратах на электроэнергию

Наименование приборов	Количество	Потребляемая мощность, Вт	Число рабочих дней	Коэффициент использования	Время работы оборудования, часы	Сумма затрат, тенге
Ноутбук	1	0,7	60	0,9	480	5259,7
МФУ	1	0,9	4	0,9	32	263
Итого						5522,7

5.6.5 Расчет накладных и прочих расходов

Прочие расходы включают в себя:

1) расходы на интернет на 60 дней(2месяца). Стоимость интернета за один месяц составляет 4500 тенге. Получаем:

4500 * 2 = 9000 тенге

2) расходы на канцелярские товары:
- упаковка бумаги формата А4 – 1000 тенге;
- упаковка ручек – 300 тенге;
- 3 карандаша – 150 тенге;
- ластик – 50 тенге;
- линейка - 100 тенге;
- стикеры – 200 тенге.

Общая стоимость канцелярских товаров

1000 + 300 + 150 + 50 +100 +200 = 1800 тенге.

3) аренда помещения на 60 дней(2месяца).

Площадь помещения:

$$S = 6м * 6м = 36 м^2 \quad (5.13)$$

Стоимость одного квадратного метра равна 1500 тенге. Получаем:

36 * 1500 * 2 = 108000 тенге.

Прочие расходы составляют

$C_{ПР}$ = 9000 + 1800 + 108000 = 118800 тенге.

Накладные расходы составляют 20% от всех затрат и рассчитываются по формуле

$$Н = (ФОТ + О_с + А + Э + С_{ПР}) * 20\% \quad (5.14)$$

Н = (214508,8 + 21236,4 + 8082,13+ 5522,7 + 118800) * 0,2 =

=73630,006 тенге.

5.6.6 Расчет стоимости по всем статьям затрат и определение структуры затрат

В соответствии с формулой 4.1 суммарные затраты по разработке моего проекта составляют:

С = 214508,8 + 21236,4 + 8082,13+ 5522,7 + 118800+ 73630,006 =
= 147260,012 тенге

Смета затрат по разработке мобильного приложения в таблице 4.8 и на рисунке 5.1

Таблица 4.8 – Суммарные данные по стоимости разработки проекта

Наименование статьи затрат	Сумма, тенге
Фонд оплаты труда	214508,8
Социальный налог	21236,4
Амортизация	8082,13
Затраты на электроэнергию	5522,7
Прочие расходы	118800
Накладные расходы	73630,006
Итого	147260,012

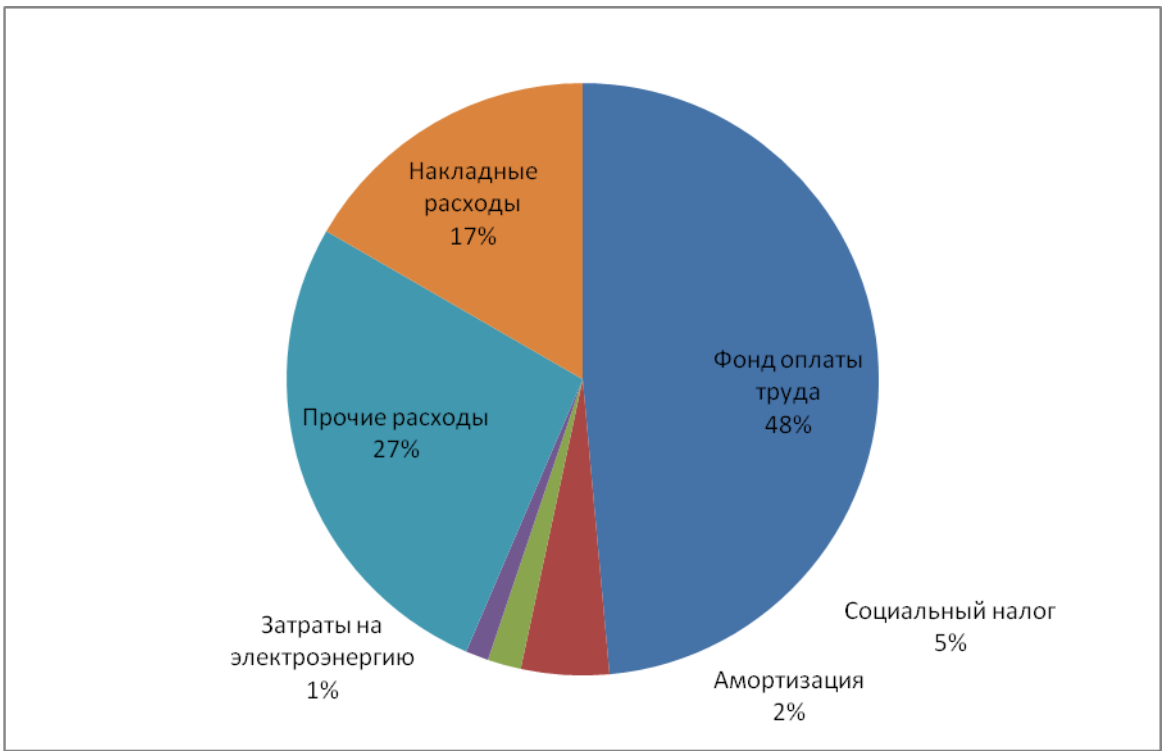

Рисунок 5.1 – Структура затрат по разработке мобильного приложения

5.7. Цена интеллектуального труда

Цена реализации проекта складывается из стоимости и чистого дохода

$$Ц = С + П \quad (5.15)$$

где С – стоимость продукта;

П – чистый доход.

При определении первоначальной цены следует задаться желаемым уровнем рентабельности (здесь 20%) реализации программных продуктов

$$Ц_П = С * (1 + \frac{Р}{100}) \quad (5.16)$$

где Р – рентабельность.

$$Ц_П = 147260{,}012 * (1 + \frac{20}{100}) = 176712{,}01 \text{ тенге}$$

Цена реализации проекта рассчитывается по формуле

$$Ц_Р = Ц_П + НДС \quad (5.17)$$

где НДС – налог на добавленную стоимость по ставке 12%

$$НДС = Ц_П * 12\% \quad (5.18)$$

НДС = 176712,01 * 0,12 = 21205,4тенге

В соответствии с формулой 4.16 цена реализации проекта составит

$$Ц_Р = 176712,01 + 21205,4 = 197917,41 \text{ тенге}$$

Вывод

В итоге на разработку мобильного приложения для обработки и анализа открытых данных, в виде погодных условий города Алматы для ОС Android тратится 197917,41 тенге, которые составляют всевозможные затраты для разработки и реализации проекта.

При создании подобных продуктов внимание уделяется разработчикам, которые будут осуществлять проект, так как наибольшую долю расходов в стоимости проекта составляют затраты на оплату труда сотрудников, что составляет 48% от всех затрат.

Сегодня на рынке мобильных технологий преобладает такие понятия как, "открытые данные", "API", "open source", что являются актуальной темой для продвижения и создания более новых и современных программ. Для меня одной из главных задач было раскрыть эту тему и выбрать область открытых данных, которая имеет потребность среди населения города. Таким образом, разработала мобильное приложение на основе открытых данных, которое обрабатывая и анализируя их, выдаёт в более качественном и удобном виде в мобильном приложении. Приложение поможет пользователям иметь доступ к точному прогнозу погоды, а также влажность, ветер, скорость и т.д. При этом если пользователь не имеет доступа в Интернет, он может зайти в приложение и увидеть прогноз погоды, который он обновлял в предыдущем сеансе, поскольку эти данные сохраняются в базе данных.

Проектирование и разработка мобильного приложения являет одним из востребованных и дорогих проектов, которые требуют интеллектуального и финансового участия. Но при этом, все расходу могут окупиться благодаря качеству и актуальности проекта среди пользователей.

6 Безопасность жизнедеятельности

6.1 Влияние электромагнитных волн на работу оператора

Слабые электромагнитные поля (ЭМП) мощность, которых примерно достигает сотые возможно даже тысячные доли Ватт высокой частоты для обычного человека грозят тем, что сила таких полей схожа с силой излучений организма человека при обычной работе всех систем и органов в его теле. В результате этого взаимодействия собственное поле человека искажается, провоцируя развитие различных заболеваний, преимущественно в наиболее ослабленных звеньях организма.

Одно из отрицательных свойств электромагнитных сигналов в том, что они могут сохраняться с течением времени в человеке. У среднестатистического человека, по сфере деятельности огромная потребность в использовании различной оргтехникой – компьютеры, телефоны (в т.ч. мобильные), в результате чего выявлено понижение иммунитета, частые стрессы и депрессии, понижение сексуальной активности, повышенная утомляемость.

Большинство специалистов выделяют нервную систему, как одну из наиболее уязвимых частей организма человека. Конструкция воздействия устроена так — электромагнитные поля нарушают пропускаемость клеточных мембран для ионов кальция, что приводит к неправильному функционированию нервной системы. Разнообразие отклонений, вызванных электромагнитными сигналами весьма широк — в итоге экспериментов выявились перемены в электроэнцефалографии головного мозга, в замедлении реакции, в ухудшении памяти, в депрессивных проявлениях.

Экспериментальные исследования в области подверженности иммунной системы ЭМИ показывают, что то у животных, облученных ЭМП, появляются изменения в характере инфекционной болезни — то есть течение болезненного процесса проходит с тяжестью. На основании чего можно предположить, что ЭМИ имеет ухудшающее влияние и нарушает работу иммуногенеза. Это связывают с появлением аутоиммунитета.

Нельзя не сказать о нарушениях в области сердечно-сосудистой системы, которая проявляются в виде неустойчивости пульса и артериального давления. Выделяются стадии перемен в среде периферической крови.

Голова, щитовидная железа, печень, сердце половая органы — это только основные опасные области воздействия ЭМИ, которые очевидны. Конечно, организм каждого человека индивидуален и действительная картина выглядит по-разному. Но в той или иной степени эти органы поражаются у всех пользователей техникой в разное время.

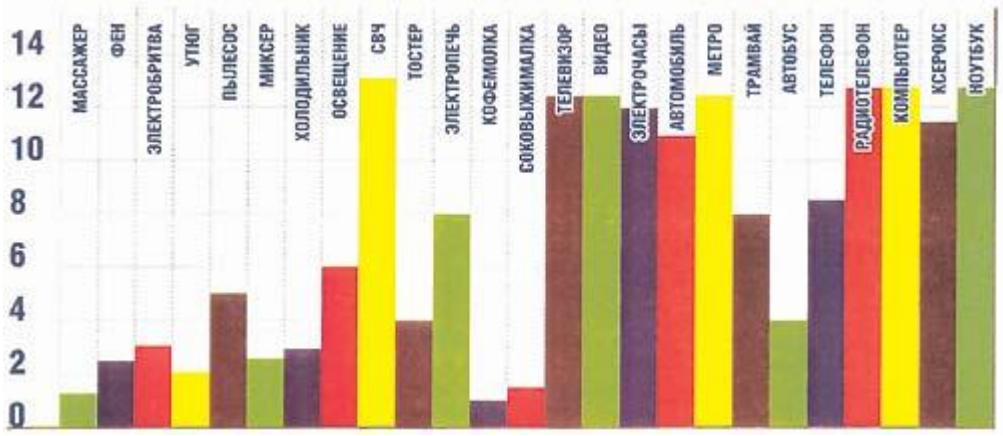

Рис. 6.1 - Влияние электромагнитного излучения различных бытовых приборов, мкВт/кв.см (плотность потока мощности)

Табл. 6.1 - процентная доля работников, сообщивших о симптомах болезни

СИМПТОМЫ ВОЗДЕЙСТВИЯ КОМПЬЮТЕРА	ПРОЦЕНТ ОПЕРАТОРОВ, СООБЩИВШИХ О СИМПТОМАХ			
	Неполная смена Работа за дисплеями до 1 года	Полная смена Работа за дисплеями до 1 года	Работа за дисплеями более 1 года	Работа за дисплеями более 2-х лет
Головная боль и боль в глазах	8%	35%	51%	76%
Утомление, головокружение	5%	32%	41%	69%
Нарушение ночного сна	____	8%	15%	50%
Сонливость в течение дня	11%	22%	48%	76%
Изменение настроения	8%	24%	27%	50%
Повышенная раздражительность	3%	11%	22%	51%
Депрессия	3%	16%	22%	50%
Снижение интеллектуальных способностей, ухудшение памяти	____	3%	12%	40%
Натяжение кожи лба и головы	3%	5%	13%	19%
Выпадение волос	_____	_____	3%	5%
Боль в мышцах	11%	14%	21%	32%
Боль в области сердца, неровное сердцебиение, отдышка	_____	5%	7%	32%
Снижение половой активности	12%	18%	34%	64%

6.2 Организация рабочего места оператора ЭВМ

Создание рабочих мест для работников ЭВМ является важной эргономической деятельностью в области вычислительной техники. Таким образом, планирование рабочего места для безопасного и эффективного

труда выполняется на основании ГОСТ 12.1.005-88 ССБТ «Общие санитарно-гигиенические требования к воздуху рабочей зоны». От правильной планировки рабочего места зависит снижение либо искоренение большой части опасных факторов, влияющих на пользователей ЭВМ.

Технический отдел состоит из двух служащих: главный технический специалист и оператор работающий на ПК. Время технического специалиста - 3 раза в неделю по 8 часов. Оператор же работает с ЭВМ каждый день.

Есть определенные эргономические требования для безопасной обстановки окружающей рабочее место оператора ЭВМ - требования к освещенности, уровню шума, температуре окружающей среды и влажности.

Уровень освещенности должен быть обеспечен с учетом создания необходимого контраста между экраном дисплея и окружающей обстановкой, особенностей выполняемых работ и требований пользователя. Необходимый уровень освещенности поверхности стола равен около 300 - 500 лк. Также в зависимости от решаемых задач, рабочее место оператора ЭВМ может быть оснащено индивидуальным источником освещения.

Говоря о шуме, создаваемом каким-либо устройством в зоне рабочего места оператора должен иметь ограничения на уровне, не ведущему к потере внимания оператором и не мешающем восприятию голоса. На рабочих местах, где ведется повышенное внимание или общение голосом, максимальный уровень шума ограничен 55 дБ, а для обычных рабочих мест - 60 дБ. К тому же должен учитываться частотный спектр шума и возраст работников, так как молодыми служащими (в особенности, молодыми женщинами) воспринимаются высокочастотные шумы, которые неслышимы пожилыми людьми.

В помещениях при работе с ЭВМ должны быть следующие климатические условия:

Холодный период года:

- оптимальная температура 24 С˚, допустимая температура 26 С˚;
- относительная влажность 45 %, допустимая влажность 60%;
- скорость движение воздуха относительная и допустимая 0,05 м/с.

Тёплый период года:

- оптимальная температура 23 С˚, допустимая температура 25 С˚;
- относительная влажность 50 %, допустимая влажность 55%;
- скорость движение воздуха относительная 0,1 м/с и допустимая 0,1м/с.

Не всем известен вред от использования устройств ввода информации, клавиатура и мышь. Есть немалое количество заболеваний, образующихся за счет деятельности на клавиатуре, которые объединены под общим названием - повреждения, вызванные повторяющимися нагрузками (RSI). Эти проблемы действительно актуальны, тем более в странах, где преобладает работа за компьютерами во многих областях человеческой деятельности. Но пользователи компьютеров могут уменьшить риск заболевания, сделав

несколько шаги по улучшению эргономичности своего рабочего места и переходу на правильную деятельность, в которую входят состояние сотрудника и тщательное распределение рабочего времени, возможен график.

Таким образом, рабочее место оператора ЭВМ должно соответствовать всем следующим требованиям, к тому же существуют дополнительные требования к состоянию работающего и геометрическим параметрам рабочего места сотрудника :

- угол между плечом и предплечьем должен быть от 70^0 до 135^0;
- угол между торсом и бедром должен быть от 90^0 до 100^0;
- угол между верхней и нижней частью ноги должен быть от 60^0 до 100^0;
- ступни должны полностью быть на полу;
- кресло должно быть регулируемым по высоте и наклону спинки;
- кресло должно быть на колесах, чтобы была возможность менять позицию;
- для обычного человека, высота стола, в среднем, 65-85см, как показано на рисунке;
- при работе с клавиатурой, руки должны двигать кисти по клавиатуре, вместо того, чтобы держать кисти неподвижно и бить по клавишам пальцами;
- когда печатаешь, не обязательно сильно бить по клавишам, нужно просто плавно нажимать;
- при необходимости нажатия двух клавиш одновременно, лучше использовать обе руки.

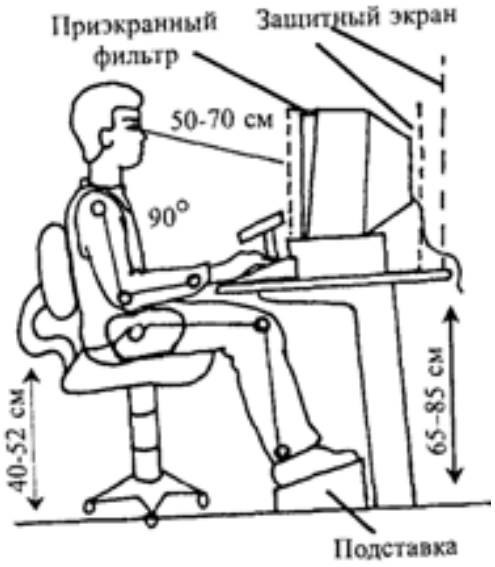

Рис.6. 2 - рабочее место оператора ЭВМ с соответствующими требованиями

Обычный пользователь компьютером выполняет работу средней тяжести с плотностью воспринимаемых сигналов 75 - 175 в час. Для существования безопасных условий труда предусматривается анализ правильной планировки умственного труда и выбор рационального труда и отдыха. Для высокой работоспособности должен быть определенный график

работы. То есть распределенный труд формируют стереотип, который помогает организму работника экономно расходовать энергию. Данный ритм задается для каждого организма по-разному, который изображен на рисунке 6.3.

Рис. 6.3 - График рациональной организации умственного труда

Так же стоит с особенным вниманием отнестись к правильному положению тела работника за рабочим столом в течении рабочего дня.

Задача для уменьшения вредного воздействия на пользователя является создание перерывов для отдыха продолжительностью 50 минут при 8-часовой рабочей системе. К примеру, создано такое расписание перерывов на отдых:

- 2 часа рабочего времени - 10 минут перерыва;
- 2 часа рабочего времени - 30 минут перерыв на обед;
- 2 часа рабочего времени - 10 минут перерыва;
- 2 часа рабочего времени - конец рабочего дня.

Следует во время перерывов выполнять упражнения, содержащие в себя шаги, приседания, повороты головой, сгибания и разгибание рук, наклоны и повороты туловища, для снятия утомления с туловища и ног, для улучшения кровообращения в области спины и живота, для предотвращения отечности, застоя крови и лимфообращения.

В заключении всего вышесказанного, можно сказать, что защита от излучений компьютера и вреда за работой начинается с пользователя данного компьютера. То есть разумный подход к его эксплуатации и дозированное время работы перед ярко мерцающим экраном.

6.3 Анализ условий труда

6.3.1 Рабочее место, виды работ

Параметры:

- выполняемая работа относится к категории легких работ (категория Iа), выполняемых в сидячем положении (ГОСТ 12.2.032-78);

- высота рабочей поверхности: 725 мм, высота сиденья: 420 мм (ГОСТ 12.2.032-78), данные ГОСТа указаны в таблице 6.1;

- размер различаемых в процессе работы объектов: 1 мм, расстояние от объекта до глаз работника: 500 мм - разряд зрительной работы:V (СНиП РК 2.04.-0.5-2002) данные СНиПа указаны в таблице 6.2;

- число работников: 1.

Т а б л и ц а 6 . 1 - Разряд зрительной работы

Наименование работ	Класс работ	Пол работника	Высота рабочей поверхности	Высота сиденья
Легкие работы (конторская работа)	Класс Iа (в сидячем положении)	Мужской, женский	725 мм	420 мм

Т а б л и ц а 6 . 2 - Виды работ

Размер минимального различаемого объекта	Расстояние от объекта до глаз работника	Разряд зрительной работы
1 - 10 мм	500 мм	V

6.3.2.Расчет помещения

Параметры:

- здание - университет, расположеный в городе Алматы (черта города). Здание пятиэтажное;

- рабочее помещение находится на третьем этаже здания;

- размеры рабочего помещения (комнаты): длина $l = 5м$, ширина $s = 3м$, высота $h = 3м$;

- остекление помещения - двойное (одно окно размером 2000х1800мм) без стального переплетения;

- внутренняя отделка стен - светлая;

- помещение по зрительным условиям работы относится к IV разряду (размер различаемых при работе предметов от 1 до 10 мм и выше) (ГОСТ 12.1.028-80);

- режим работы (продолжительность рабочего дня): 9^{00} - 18^{00};

План помещения представлен в приложении А.

План рабочего помещения:

- здание относится к I степени огнестойкости (СНиП РК 2.02-05-2002) таблица 7.3;

- общая площадь помещения 15 м^2. Площадь, занимаемая оборудованием и мебелью 3,5 м^2.

Т а б л и ц а 6 . 3 - Конструктивная характеристика зданий в зависимости от их степени огнестойкости

Степень огнестойкости	Конструктивные характеристики
I	Здания с несущими и ограждающими конструкциями из естественных или искусственных материалов, бетона или железобетона с применением листовых о плитных негорючих материалов

6.4 Расчет освещения, системы кондиционирования

6.4.1. Расчет освещенности

Определим площадь боковых световых проемов помещения кабинета, необходимую для создания нормальной освещенности на рабочем месте .

Помещение имеет размеры: длина $l = 5м$, ширина $s = 3м$, высота $h = 3м$. Высота рабочей поверхности над уровнем пола - 0,725 м, окно начинаются с высоты 1 м, высота окна 1,8 м. Рабочее помещение находится в IV часовом поясе - город Алматы. Напротив здания университета здание на расстоянии в 1,5 раза больше чем само здание.

Рабочее место находится в 1,5 м от наружной стены помещения, где спроектированы окна. Минимальная освещенность будет в месте на расстояние 3 м от оконного проема.

Общую площадь окон S_0, м^2 определим по формуле

$$100 \cdot \frac{S_0}{S_n} = \frac{e_n \cdot \eta_0}{\tau_0 \cdot r_1} \cdot k_{ЗД} \cdot k_З, \quad (6.1)$$

$$S_0 = \frac{S_n \cdot e_n \cdot \eta_0 \cdot k_{ЗД} \cdot k_З}{100 \cdot \tau_0 \cdot r_1} \quad (6.2)$$

где S_n - площадь помещения, м^2;

e_n - нормированное значение КЕО;

$k_З = 1,2$ - коэффициент запаса;

τ_0 - общий коэффициент светопропускания;

η_0 - световая характеристика окон;

$k_{ЗД}$ - коэффициент, учитывающий затенение окон противостоящими зданиями;

r_1 - коэффициент, учитывающий повышение КЕО при боковом освещении благодаря свету, отраженному от поверхностей помещения и подстилающего слоя, прилегающего к зданию.

Нормативное значение КЕО

$$e_n^{IV} = e_n \cdot m \cdot c \quad (6.3)$$

где m=0,9;

c=0,75 - для IV часового пояса;

e_n=1,2 для работ средней точности IV подразряда.

$e_n^{IV} = 1{,}2 \cdot 0{,}9 \cdot 0{,}75 = 0.81$

Общий коэффициент светопропускания равный

$$\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 \qquad (6.4)$$

$\tau_0 = 0{,}8 \cdot 0{,}6 \cdot 0{,}8 \cdot 1 = 0{,}38$

Отношение длины L=5 м комнаты к глубине B=3 м наиболее удаленной точки от окна равно

$$\frac{L}{B} = \frac{5}{3} = 1{,}67$$

Высота от уровня рабочей поверхности до верха окна

h_1=2,8-0,725=2,075 м

Определим отношение ширины помещения к высоте от уровня рабочей поверхности до верха окна равно

$$\frac{B}{h_1} = \frac{3}{2.075} = 1{,}45$$

Отсюда $\eta_0 = 10{,}5$

Отношение

$$\frac{l}{B} = \frac{3}{3} = 1$$

Средний коэффициент отражения в помещении $\rho_{CP} = 0{,}5$, принимаем одностороннее боковое освещение. Тогда r_1=1,5

Найдем $k_{ЗД}$ затеняющиее здание находится на расстоянии в соотношении р: Н=2, то $k_{ЗД}$=1,1

Вычислим общую площадь окон

$$S_0 = \frac{15 \cdot 0{,}81 \cdot 10{,}5 \cdot 1{,}1 \cdot 1{,}2}{100 \cdot 0{,}38 \cdot 1{,}5} = 2.95 м^2$$

Из расчета видно, что предусмотренные оконные проемы соответствуют нормативам естественного освещения.

6.4.2 Расчет системы кондиционирования

Ниже рассчитан расчет системы кондиционирования в кабинете. Кондиционирование обеспечит соответствие климата в рабочем помещении нормативам.

Количество приточного воздуха $L_{ПР}, \frac{м^3}{ч}$ определяем по формуле

$$L_{ПР} = \frac{Q_{ИЗБ}}{c \cdot p_{ПР} \cdot (t_{ВЫТ} - t_{ПР})} \quad (6.5)$$

где $Q_{ИЗБ}$ - избыточное выделение явной теплоты, $\frac{кДж}{ч}$;

c - удельная теплоемкость воздуха при постоянном давлении, равная $1\frac{кДж}{кг \cdot ^0 C}$;

$p_{ПР}$ - плотность поступающего в помещение воздуха, равная $1{,}2\frac{кг}{м^3}$;

$t_{ВЫТ}$ - температура удаляемого из помещения за пределы рабочей или обслуживаемой зоны, ^0C;

$t_{ПР}$ - температура приточного воздуха, ^0C;

Температура удаляемого из помещения воздуха $t_{ВЫТ}, ^0C$, определяется по формуле

$$t_{ВЫТ} = t_{РЗ} + \Delta t \cdot (H - z) \quad (6.6)$$

где $t_{РЗ}$ - температура в рабочей зоне, которая не должна превышать допустимую по нормам $(t_{РЗ} \le t_{ДОП}), ^0C$. Поскольку расчет производится для теплого периода года, то примем $t_{РЗ} = 22^0C$;

Δt - температурный градиент по высоте помещения $(\Delta t = 0{,}5 - 1{,}5)^0C$;

Н - расстояние от пола до центра вытяжных проемов (кондиционера), м. Внутренняя часть кондиционера расположена на высоте Н=2,5 м;

z- высота рабочей зоны, м.

$$t_{ВЫТ} = 22 + 1{,}2 \cdot (2{,}5 - 3) = 21{,}4^0C$$

Температура приточного воздуха ($t_{ПР}$) при наличии при наличии избытка явной теплоты должна быть на 5-7 ^{0}С ниже температуры воздуха в рабочей зоне.

$$t_{ПР} = 22 - 7 = 15^0C$$

Величину избыточного выделения явной теплоты $O_{ИЗБ}$ находят на основании баланса теплоты в помещении по формуле

$$O_{ИЗБ} = \sum Q - \sum Q_{УХ} \qquad (6.7)$$

где $\sum Q$ - суммарное количество поступающей в помещение явной теплоты;

$\sum Q_{УХ}$ - суммарное количество уходящей из помещения теплоты (за счет теплопотерь ограждениями, нагрева поступающего в помещение воздуха и т. п.).

Главными ресурсами избыточного тепла являются светильники, промышленные печи, электроустановки, люди и др. При этом нельзя забывать о теплопоступления от солнечной радиации.

Тепловыделения от искусственного освещения Q_2, рассчитываются, предопределяя, что на практике вся потребляемая энергия в итоге переходит в тепло, по формуле

$$Q_2 = N \qquad (6.8)$$

где N - расходуемая мощность светильников, Вт.

$$Q_2 = 5 \times 65 = 325 Вт$$

Тепловыделения от людей Q_3 определяют по формуле

$$Q_3 = n \cdot q_Ч \qquad (6.9)$$

где n - число работающих;

$q_ч$ - количество тепла, выделяемое одним человеком, Вт.

Т а б л и ц а 6 . 4 - Количество тепла, выделяемое одним человеком в зависимости от категории работ и температуры окружающей среды

Категория работ	Тепло, Вт			
	Полное		Явное	
	при 10^0С	при 35^0С	при 10^0С	при 35^0С
Легкая	180	145	150	5

$$Q_3 = 1 \cdot 145 = 145 Вт$$

Количество тепла, поступающего в помещение от солнечной радиации $Q_{ОСТ.РАД}$, определяют по формуле

$$Q_{ОСТ.РАД} = F_{ОСТ} \cdot q_{ОСТ} \cdot A_{ОСТ} \quad (6.10)$$

Для покрытий

$$Q_{П.РАД} = F_n \cdot q_n \cdot k_n \quad (6.11)$$

где $F_{ОСТ}$ и F_n - площадь поверхности и покрытия, м^2;

$q_{ОСТ}$ и q_n - теплопоступления через 1 м^2 поверхности остекления и поверхности покрытия, при коэффициенте теплопередачи, равном $1\frac{Вт}{м^2 \cdot {}^0C}, \frac{Вт}{м^2}$;

$A_{ОСТ}$ - коэффициент остекления;

k_n - коэффициент теплопередачи покрытия, $\frac{Вт}{м^2 \cdot {}^0C}$;

Значение $q_{ОСТ}$ в зависимости от географической ориентации поверхности и характеристики окон или фонарей принимается в пределах 70-210, а коэффициента $A_{ОСТ}$ в зависимости от вида остекления и его солнцезащитных свойств - в пределах 0,25 - 1,25, средние значения теплопоступления от солнечной радиации через покрытие в зависимости от географической широты и вида покрытия принимают в пределах 6 - 24.

$$F_{ОСТ} = 2 \cdot 1{,}8 = 3{,}6 м^2$$

Окно в рабочем помещении смотрит на юг, поэтому примем значение $q_{ОСТ}$ равным $170\frac{Вт}{м^2 \cdot {}^0C}$. Поскольку остекление образовано светлым стеклом и без стального переплетения, то примем $A_{ОСТ}$=0,35.

$$Q_{ОСТ.РАД} = 3{,}6 \cdot 170 \cdot 0{,}35 = 214{,}2 Вт$$

Среднее значение теплопоступления для примем равным $Q_{П.РАД} = 18 Вт$.

Количество потери тепла из рабочего кабинета $Q_{УХ}$, кВт через стены двери, окна насчитывается примерно по следующей формуле

$$Q_{УХ} = \frac{\lambda \cdot S \cdot (t_{ВЫТ} - t_{ПР})}{\delta} \quad (6.12)$$

где λ - теплопроводность стен, $\frac{Вт}{м \cdot {}^0C}$;

S - площадь, м^2;

δ - толщина стен, м.

Стены рабочего места сделаны из тяжелого бетона М600, теплопроводность которого равна $1{,}2\frac{Вт}{м\cdot^0 C}$. Толщина стен $\delta = 0{,}5м$.

$$Q_{УХ} = \frac{1{,}2 \cdot 15 \cdot (21{,}4 - 15)}{0{,}5} = 230{,}4Вт$$

Определим сумму общего числа теплоты, что поступает в кабинет

$$\sum Q = Q_2 + Q_3 + Q_{ОСТ.РАД} + Q_{п.РАД} \qquad (6.13)$$

$$\sum Q = 325 + 145_3 + 214{,}2 + 18 = 702{,}2Вт$$

Подставляя значения, вычислим избыточное выделение явной теплоты

$$O_{ИЗБ} = 702{,}2 - 230{,}4 = 471{,}8Вт$$

Также определим количество приточного воздуха

$$L_{ПР} = \frac{471{,}8}{1 \cdot 1{,}2 \cdot (24{,}1 - 15)} = 43{,}63\frac{м^3}{ч}$$

Таким образом, ориентируясь на расчетах в разделе БЖД решила, что необходим кондиционер в рабочем помещении: Polaris PS-7 - сплит-система настенного типа, японского производства, что имеет в комплекте японский компрессор и качественными комплектующими для долгой и бесперебойной работы.

Данный экземпляр владеет саморегулируемыми створками, которые отвечают за полную циркуляцию воздуха в рабочем кабинете.

- режим охлаждения - 2200 Вт;
- режим обогрева - 2400Вт;
- расход воздуха - до 400 м3/час;
- уровень шума < 40 Дб;
- габариты внутреннего блока - 785х260х175 мм, вес - 8,5 кг (внутренний блок), 27 кг (внешний блок);
- рассчитан на помещение площадью до 24 м^2.

Заключение

Разработка мобильного приложения на платформе Android была для меня огромным опытом и очень сложной задачей, выполнение которой может быть более чем оправдывающей себя. Работа требовала сильного логического мышления и структурного подхода. Она содержит в себе задачи, не связанные напрямую с изучением языков программирования. Я получила огромный поток новых знаний, который в основном заключался в новизне, что требовал исследования и новые технологии, с которыми стоит экспериментировать. Наградой за весь труд является вид исправно работающей финальной версии мобильного приложения и осознание своего вклада в осуществление этой цели. Сегодня программное обеспечение задаёт образ современного мира, и те, кто работает над его реализацией, могут внести свой вклад в форму будущего.

В течении всей работы над данным дипломным проектом были изучены основные принципы разработки мобильных приложений, а также требований и нормы, предъявляемых к ним. Изучены принципы построения и работы приложения для операционной системы Android, работа с базой данный SQLite. Кроме этого, работала над определение "открытые данные", только входящим в современный мир, эта тема является актуальной в реалиях современного мира разработчиков и бизнесменов. А именно проанализировав эту тему, выбрала область являющуюся необходимой среди пользователей - это прогноз погоды. В принципе разработанное приложение отвечает всем стандартам и требованиям платформ держателя(Google), которые я разобрала в дипломном проекте. Мной была рассчитана общая стоимость продукта, в основе которой суммировала различные виды затрат, которые возможны при производстве. Вышеприведенный расчет соответствует любому виду реализации продукта, как на разработка для заказчика, так и самостоятельная разработка, для последующей публикации в интернет магазине мобильных приложений. В ходе исследования подобных продуктов на рынке, установила, что стоимость реализации соответствует качеству реализуемого мобильного приложения и есть возможность получить большой объем прибыли. В разделе «Безопасность жизнедеятельности» я рассчитала условия для обеспечения необходимых всем установленным нормам, рабочего места для сотрудников, разрабатывающих проект. Также рассчитала нужное искусственное освещение, которое требуется для комфортной работы без нанесения вреда здоровью работников.

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. Сайт https://habrahabr.ru/
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. Носова С.С., Экономическая теория. – М.: Кнорус, 2010.
13. Артамонова В.С., Иванова С.А. Экономическая теория. – СПб.: Питер, 2010.
14. Камаева В.Д., Лобачевой Е.И. Экономическая теория: Учебник.– М.: Юрайт, 2010.

Приложение А

Листинг программы

```
package kz.sunshine;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;

import kz.aibol.sunshine.sync.SunshineSyncAdapter;

public class MainActivity extends AppCompatActivity implements
ForecastFragment.Callback {

    private final String LOG_TAG = MainActivity.class.getSimpleName();
    private static final String DETAILFRAGMENT_TAG = "DFTAG";

    private String mLocation;
    private boolean mTwoPane;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        mLocation = Utility.getPreferredLocation(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        toolbar.setLogo(R.drawable.ic_logo);
        toolbar.setTitle("");
        setSupportActionBar(toolbar);

        if (findViewById(R.id.weather_detail_container) != null) {
            mTwoPane = true;
            if (savedInstanceState == null) {
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.weather_detail_container, new
DetailFragment(), DETAILFRAGMENT_TAG)
                        .commit();
            }
        } else {
            mTwoPane = false;
            getSupportActionBar().setElevation(0f);
        }
        ForecastFragment forecastFragment = (ForecastFragment)
getSupportFragmentManager().findFragmentById(R.id.fragment_forecast);
        forecastFragment.setUseTodayLayout(!mTwoPane);

        SunshineSyncAdapter.initializeSyncAdapter(this);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
         int id = item.getItemId();

               if (id == R.id.action_settings) {
            startActivity(new Intent(this, SettingsActivity.class));
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onResume() {
        super.onResume();
        String location = Utility.getPreferredLocation(this);
        if (location != null && !mLocation.equals(location)) {
            ForecastFragment ff = (ForecastFragment)
getSupportFragmentManager().findFragmentById(R.id.fragment_forecast);
            if (ff != null) {
                ff.onLocationChanged();
            }
            DetailFragment df = (DetailFragment)
getSupportFragmentManager().findFragmentByTag(DETAILFRAGMENT_TAG);
            if (df != null) {
                df.onLocationChanged(location);
            }
            mLocation = location;
        }
    }

    @Override
    public void onItemSelected(Uri contentUri) {
        if(mTwoPane) {
            Bundle args = new Bundle();
            args.putParcelable(DetailFragment.DETAIL_URI, contentUri);

            DetailFragment fragment = new DetailFragment();
            fragment.setArguments(args);

            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.weather_detail_container, fragment,
DETAILFRAGMENT_TAG)
                    .commit();
        } else {
            Intent intent = new Intent(this, DetailActivity.class)
                    .setData(contentUri);
            startActivity(intent);
        }
    }
package kz.sunshine;
import android.annotation.TargetApi;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
public class SettingsActivity extends AppCompatPreferenceActivity {
    private static Preference.OnPreferenceChangeListener
sBindPreferenceSummaryToValueListener = new
Preference.OnPreferenceChangeListener() {
```

```
        @Override
        public boolean onPreferenceChange(Preference preference, Object value)
{
            String stringValue = value.toString();

            if (preference instanceof ListPreference) {
                ListPreference listPreference = (ListPreference) preference;
                int index = listPreference.findIndexOfValue(stringValue);
                preference.setSummary(
                        index >= 0
                                ? listPreference.getEntries()[index]
                                : null);

            } else {

               preference.setSummary(stringValue);
            }
            return true;
        }
    };
      private static void bindPreferenceSummaryToValue(Preference preference)
{
        // Set the listener to watch for value changes.

preference.setOnPreferenceChangeListener(sBindPreferenceSummaryToValueListener
);
        sBindPreferenceSummaryToValueListener.onPreferenceChange(preference,
                PreferenceManager
                        .getDefaultSharedPreferences(preference.getContext())
                        .getString(preference.getKey(), ""));
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setupActionBar();
        addPreferencesFromResource(R.xml.pref_general);


bindPreferenceSummaryToValue(findPreference(getString(R.string.pref_location_k
ey)));

bindPreferenceSummaryToValue(findPreference(getString(R.string.pref_units_key)
));
    }
    private void setupActionBar() {
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            // Show the Up button in the action bar.
            actionBar.setDisplayHomeAsUpEnabled(true);
        }
    }

    @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
    @Nullable
    @Override
    public Intent getParentActivityIntent() {
        return
super.getParentActivityIntent().addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    }
}
public class ForecastFragment extends Fragment implements
LoaderManager.LoaderCallbacks<Cursor> {
```

```
private final String LOG_TAG = ForecastFragment.class.getSimpleName();

private static final String[] FORECAST_COLUMN = {
        WeatherContract.WeatherEntry.TABLE_NAME + "." +
                WeatherContract.WeatherEntry._ID,
        WeatherContract.WeatherEntry.COLUMN_DATE,
        WeatherContract.WeatherEntry.COLUMN_SHORT_DESC,
        WeatherContract.WeatherEntry.COLUMN_MAX_TEMP,
        WeatherContract.WeatherEntry.COLUMN_MIN_TEMP,
        WeatherContract.LocationEntry.COLUMN_LOCATION_SETTING,
        WeatherContract.WeatherEntry.COLUMN_WEATHER_ID,
        WeatherContract.LocationEntry.COLUMN_COORD_LAT,
        WeatherContract.LocationEntry.COLUMN_COORD_LONG
};

static final int COL_WEATHER_ID = 0;
static final int COL_WEATHER_DATE = 1;
static final int COL_WEATHER_DESC = 2;
static final int COL_WEATHER_MAX_TEMP = 3;
static final int COL_WEATHER_MIN_TEMP = 4;
static final int COL_LOCATION_SETTING = 5;
static final int COL_WEATHER_CONDITION_ID = 6;
static final int COL_COORD_LAT = 7;
static final int COL_COORD_LONG = 8;

private static final int FORECAST_LOADER = 0;
private boolean mUseTodayLayout;

ForecastAdapter mForecastAdapter;

private ListView mListView;
private int mPosition = ListView.INVALID_POSITION;

private static final String SELECTED_KEY = "selected_position";

public ForecastFragment() {
}

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

    setHasOptionsMenu(true);
}

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    inflater.inflate(R.menu.forecastfragment, menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_map:
            openPreferredLocationInMap();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}

@Override
```

```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             final Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.fragment_main, container,
false);

        mForecastAdapter = new ForecastAdapter(getActivity(), null, 0);
        mForecastAdapter.setUseTodayLayout(mUseTodayLayout);

        mListView = (ListView) rootView.findViewById(R.id.listview_forecast);
        View emptyView = rootView.findViewById(R.id.listview_forecast_empty);
        mListView.setEmptyView(emptyView);
        mListView.setAdapter(mForecastAdapter);
        mListView.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int
i, long l) {
                Cursor cursor = (Cursor) adapterView.getItemAtPosition(i);
                if (cursor != null) {
                    String locationSetting =
Utility.getPreferredLocation(getActivity());
                    ((Callback) getActivity())

.onItemSelected(WeatherContract.WeatherEntry.buildWeatherLocationWithDate(
                                    locationSetting,
cursor.getLong(COL_WEATHER_DATE)
                            ));
                }
                mPosition = i;
            }
        });

        if(savedInstanceState != null &&
savedInstanceState.containsKey(SELECTED_KEY)) {
            mPosition = savedInstanceState.getInt(SELECTED_KEY);
        }

        return rootView;
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        if(mPosition != ListView.INVALID_POSITION) {
            outState.putInt(SELECTED_KEY, mPosition);
        }
        super.onSaveInstanceState(outState);
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        getLoaderManager().initLoader(FORECAST_LOADER, null, this);
        super.onActivityCreated(savedInstanceState);
    }

    public void onLocationChanged() {
        updateWeather();
        getLoaderManager().restartLoader(FORECAST_LOADER, null, this);
    }

    private void updateWeather() {
        SunshineSyncAdapter.syncImmediately(getActivity());
    }
```

```
    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
        String locationSetting = Utility.getPreferredLocation(getActivity());

        String sortOrder = WeatherContract.WeatherEntry.COLUMN_DATE + " ASC";
        Uri weatherForLocationUri =
WeatherContract.WeatherEntry.buildWeatherLocationWithStartDate(locationSetting
,
                System.currentTimeMillis());

        return new CursorLoader(getActivity(),
                weatherForLocationUri,
                FORECAST_COLUMN,
                null,
                null,
                sortOrder);
    }

    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
        mForecastAdapter.swapCursor(data);
        if(mPosition != ListView.INVALID_POSITION) {
            mListView.smoothScrollToPosition(mPosition);
        }
        updateEmptyView();
    }

    public void setUseTodayLayout(boolean useTodayLayout) {
        mUseTodayLayout = useTodayLayout;
        if(mForecastAdapter != null) {
            mForecastAdapter.setUseTodayLayout(mUseTodayLayout);
        }
    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {
        mForecastAdapter.swapCursor(null);
    }public interface Callback {
         public void onItemSelected(Uri dateUri);
    }

    private void openPreferredLocationInMap() {
   public class ForecastAdapter extends CursorAdapter {

    private boolean mUseTodayLoayout;

    private final int VIEW_TYPE_TODAY = 0;
    private final int VIEW_TYPE_FUTURE_DAY = 1;
    private static final int VIEW_TYPE_COUNT = 2;

    public ForecastAdapter(Context context, Cursor c, int flags) {
        super(context, c, flags);
    }

    public void setUseTodayLayout(boolean useTodayLayout) {
        mUseTodayLoayout = useTodayLayout;
    }

    @Override
    public int getViewTypeCount() {
        return VIEW_TYPE_COUNT;
    }
```

```
    @Override
    public int getItemViewType(int position) {
        return (position == 0 && mUseTodayLoayout) ? VIEW_TYPE_TODAY :
VIEW_TYPE_FUTURE_DAY;
    }

    /*
        Remember that these views are reused as needed.
     */
    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        int viewType = getItemViewType(cursor.getPosition());
        int layoutId = -1;

        if (viewType == VIEW_TYPE_TODAY) {
            layoutId = R.layout.list_item_forecast_today;
        } else if (viewType == VIEW_TYPE_FUTURE_DAY) {
            layoutId = R.layout.list_item_forecast;
        }

        View view = LayoutInflater.from(context).inflate(layoutId, parent,
false);

        ViewHolder viewHolder = new ViewHolder(view);
        view.setTag(viewHolder);

        return view;
    }    @Override
    public void bindView(View view, Context context, Cursor cursor) {
                ViewHolder viewHolder = (ViewHolder) view.getTag();

              int viewType = getItemViewType(cursor.getPosition());
        switch (viewType) {
            case VIEW_TYPE_TODAY:

viewHolder.iconView.setImageResource(Utility.getArtResourceForWeatherCondition
(

cursor.getInt(ForecastFragment.COL_WEATHER_CONDITION_ID)
                ));
                break;
            case VIEW_TYPE_FUTURE_DAY:

viewHolder.iconView.setImageResource(Utility.getIconResourceForWeatherConditio
n(

cursor.getInt(ForecastFragment.COL_WEATHER_CONDITION_ID)
                ));
                break;
        }

        long date = cursor.getLong(ForecastFragment.COL_WEATHER_DATE);
        viewHolder.dateView.setText(Utility.getFriendlyDayString(context,
date));

        String description =
cursor.getString(ForecastFragment.COL_WEATHER_DESC);
        viewHolder.descriptionView.setText(description);
        viewHolder.iconView.setContentDescription(description);
        boolean isMetric = Utility.isMetric(context);
        double high = cursor.getDouble(ForecastFragment.COL_WEATHER_MAX_TEMP);
        viewHolder.highTempView.setText(Utility.formatTemperature(context,
high, isMetric));
```

```
        double low = cursor.getDouble(ForecastFragment.COL_WEATHER_MIN_TEMP);
        viewHolder.lowTempView.setText(Utility.formatTemperature(context, low,
isMetric));
    }

    public static class ViewHolder {
        public final ImageView iconView;
        public final TextView dateView;
        public final TextView descriptionView;
        public final TextView highTempView;
        public final TextView lowTempView;

        public ViewHolder(View view) {
            iconView = (ImageView) view.findViewById(R.id.list_item_icon);
            dateView = (TextView)
view.findViewById(R.id.list_item_date_textview);
            descriptionView = (TextView)
view.findViewById(R.id.list_item_forecast_textview);
            highTempView = (TextView)
view.findViewById(R.id.list_item_high_textview);
            lowTempView = (TextView)
view.findViewById(R.id.list_item_low_textview);
        }
    }
public class DetailFragment extends Fragment implements
LoaderManager.LoaderCallbacks<Cursor> {

    private static final int DETAIL_LOADER = 0;

    private static final String[] DETAIL_COLUMNS = {
            WeatherContract.WeatherEntry.TABLE_NAME + "." +
WeatherContract.WeatherEntry._ID,
            WeatherContract.WeatherEntry.COLUMN_DATE,
            WeatherContract.WeatherEntry.COLUMN_SHORT_DESC,
            WeatherContract.WeatherEntry.COLUMN_MAX_TEMP,
            WeatherContract.WeatherEntry.COLUMN_MIN_TEMP,
            WeatherContract.WeatherEntry.COLUMN_HUMIDITY,
            WeatherContract.WeatherEntry.COLUMN_PRESSURE,
            WeatherContract.WeatherEntry.COLUMN_WIND_SPEED,
            WeatherContract.WeatherEntry.COLUMN_DEGREES,
            WeatherContract.WeatherEntry.COLUMN_WEATHER_ID,
            WeatherContract.LocationEntry.COLUMN_LOCATION_SETTING
    };
    public static final int COL_WEATHER_ID = 0;
    public static final int COL_WEATHER_DATE = 1;
    public static final int COL_WEATHER_DESC = 2;
    public static final int COL_WEATHER_MAX_TEMP = 3;
    public static final int COL_WEATHER_MIN_TEMP = 4;
    public static final int COL_WEATHER_HUMIDITY = 5;
    public static final int COL_WEATHER_PRESSURE = 6;
    public static final int COL_WEATHER_WIND_SPEED = 7;
    public static final int COL_WEATHER_DEGREES = 8;
    public static final int COL_WEATHER_CONDITION_ID = 9;

    private ImageView mIconView;
    private TextView mFriendlyDateView;
    private TextView mDateView;
    private TextView mDescriptionView;
    private TextView mHighTempView;
    private TextView mLowTempView;
    private TextView mHumidityView;
    private TextView mWindView;
```

```
    private TextView mPressureView;

    private static final String LOG_TAG =
DetailFragment.class.getSimpleName();
    static final String DETAIL_URI = "URI";
    private static final String FORECAST_SHARE_HASHTAG = "#SunshineApp";
    private String mForecastStr;
    private Uri mUri;
    private ShareActionProvider mShareActionProvider;
    public DetailFragment() {
        setHasOptionsMenu(true);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        Bundle arguments = getArguments();
        if (arguments != null) {
            mUri = arguments.getParcelable(DetailFragment.DETAIL_URI);
        }
        View rootView = inflater.inflate(R.layout.fragment_detail, container,
false);
        mIconView = (ImageView) rootView.findViewById(R.id.detail_icon);
        mDateView = (TextView)
rootView.findViewById(R.id.detail_date_textview);
        mFriendlyDateView = (TextView)
rootView.findViewById(R.id.detail_day_textview);
        mDescriptionView = (TextView)
rootView.findViewById(R.id.detail_forecast_textview);
        mHighTempView = (TextView)
rootView.findViewById(R.id.detail_high_textview);
        mLowTempView = (TextView)
rootView.findViewById(R.id.detail_low_textview);
        mHumidityView = (TextView)
rootView.findViewById(R.id.detail_humidity_textview);
        mWindView = (TextView)
rootView.findViewById(R.id.detail_wind_textview);
        mPressureView = (TextView)
rootView.findViewById(R.id.detail_pressure_textview);

        return rootView;
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.detailfragment, menu);

        MenuItem item = menu.findItem(R.id.action_share);

        mShareActionProvider = (ShareActionProvider)
MenuItemCompat.getActionProvider(item);

        if (mForecastStr != null) {
            mShareActionProvider.setShareIntent(createShareForecastIntent());
        }
    }
    private Intent createShareForecastIntent() {
        Intent shareIntent = new Intent(Intent.ACTION_SEND);
        shareIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
        shareIntent.setType("text/plain");
        shareIntent.putExtra(Intent.EXTRA_TEXT, mForecastStr +
FORECAST_SHARE_HASHTAG);
        return shareIntent;
    }
```

```
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        getLoaderManager().initLoader(DETAIL_LOADER, null, this);
    }
    void onLocationChanged(String newLocation) {
        Uri uri = mUri;
        if (null != uri) {
            long date = WeatherContract.WeatherEntry.getDateFromUri(uri);
            Uri updatedUri =
WeatherContract.WeatherEntry.buildWeatherLocationWithDate(newLocation, date);
            mUri = updatedUri;
            getLoaderManager().restartLoader(DETAIL_LOADER, null, this);
        }
    }
    @Override
    public Loader<Cursor> onCreateLoader(int id, Bundle args) {
        if (null != mUri) {
            return new CursorLoader(
                    getActivity(),
                    mUri,
                    DETAIL_COLUMNS,
                    null,
                    null,
                    null
            );
        }
        return null;
    }
    @Override
    public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
        if (data != null && data.moveToFirst()) {
            int weatherId = data.getInt(COL_WEATHER_CONDITION_ID);

mIconView.setImageResource(Utility.getArtResourceForWeatherCondition(weatherId
));
            long date = data.getLong(COL_WEATHER_DATE);
            String friendlyDateText = Utility.getDayName(getActivity(), date);
            String dateText = Utility.getFormattedMonthDay(getActivity(),
date);
            mFriendlyDateView.setText(friendlyDateText);
            mDateView.setText(dateText);
            String description = data.getString(COL_WEATHER_DESC);
            mDescriptionView.setText(description);


            mIconView.setContentDescription(description);
            boolean isMetric = Utility.isMetric(getActivity());
            double high = data.getDouble(COL_WEATHER_MAX_TEMP);
            String highString = Utility.formatTemperature(getActivity(), high,
isMetric);
            mHighTempView.setText(highString);
            double low = data.getDouble(COL_WEATHER_MIN_TEMP);
            String lowString = Utility.formatTemperature(getActivity(), low,
isMetric);
            mLowTempView.setText(lowString);

            float humidity = data.getFloat(COL_WEATHER_HUMIDITY);

mHumidityView.setText(getActivity().getString(R.string.format_humidity,
humidity));
            float windSpeedStr = data.getFloat(COL_WEATHER_WIND_SPEED);
            float windDirStr = data.getFloat(COL_WEATHER_DEGREES);
```

```
            mWindView.setText(Utility.getFormattedWind(getActivity(),
windSpeedStr, windDirStr));
            float pressure = data.getFloat(COL_WEATHER_PRESSURE);

mPressureView.setText(getActivity().getString(R.string.format_pressure,
pressure));
            mForecastStr = String.format("%s - %s - %s/%s", dateText,
description, high, low);

            if (mShareActionProvider != null) {

mShareActionProvider.setShareIntent(createShareForecastIntent());
            }
        }
    }

    @Override
    public void onLoaderReset(Loader<Cursor> loader) {
    }
}
 if ( null != mForecastAdapter ) {
            Cursor c = mForecastAdapter.getCursor();
            if ( null != c ) {
                c.moveToPosition(0);
                String posLat = c.getString(COL_COORD_LAT);
                String posLong = c.getString(COL_COORD_LONG);
                Uri geoLocation = Uri.parse("geo:" + posLat + "," + posLong);

                Intent intent = new Intent(Intent.ACTION_VIEW);
                intent.setData(geoLocation);

                if (intent.resolveActivity(getActivity().getPackageManager())
!= null) {
                    startActivity(intent);
                } else {
                    Log.d(LOG_TAG, "Couldn't call " + geoLocation.toString() +
", no receiving apps installed!");
                }
            }

        }
    }

    private void updateEmptyView() {
        if(mForecastAdapter.getCount() == 0) {
            TextView tv = (TextView)
getView().findViewById(R.id.listview_forecast_empty);;
            if(tv != null) {
                int message = R.string.empty_forecast_list;
                if(!Utility.isNetworkAvailable(getActivity())) {
                    message = R.string.empty_forecast_list_no_network;
                }
                tv.setText(message);
            }
        }
    }

}
  public static final String DATE_FORMAT = "yyyyMMdd";
    public static String getFriendlyDayString(Context context, long
dateInMillis) {
                Time time = new Time();
        time.setToNow();
```

```
        long currentTime = System.currentTimeMillis();
        int julianDay = Time.getJulianDay(dateInMillis, time.gmtoff);
        int currentJulianDay = Time.getJulianDay(currentTime, time.gmtoff);
        if (julianDay == currentJulianDay) {
            String today = context.getString(R.string.today);
            int formatId = R.string.format_full_friendly_date;
            return String.format(context.getString(
                    formatId,
                    today,
                    getFormattedMonthDay(context, dateInMillis)));
        } else if (julianDay < currentJulianDay + 7) {
            return getDayName(context, dateInMillis);
        } else {
                        SimpleDateFormat shortenedDateFormat = new
SimpleDateFormat("EEE MMM dd");
            return shortenedDateFormat.format(dateInMillis);
        }
    }    public static String getDayName(Context context, long dateInMillis) {
        Time t = new Time();
        t.setToNow();
        int julianDay = Time.getJulianDay(dateInMillis, t.gmtoff);
        int currentJulianDay = Time.getJulianDay(System.currentTimeMillis(),
t.gmtoff);
        if (julianDay == currentJulianDay) {
            return context.getString(R.string.today);
        } else if (julianDay == currentJulianDay + 1) {
            return context.getString(R.string.tomorrow);
        } else {
            Time time = new Time();
            time.setToNow();

            SimpleDateFormat dayFormat = new SimpleDateFormat("EEEE");
            return dayFormat.format(dateInMillis);
        }
    }    public static String getFormattedMonthDay(Context context, long
dateInMillis) {
        Time time = new Time();
        time.setToNow();
        SimpleDateFormat dbDateFormat = new
SimpleDateFormat(Utility.DATE_FORMAT);
        SimpleDateFormat monthDayFormat = new SimpleDateFormat("MMMM dd");
        String monthDayString = monthDayFormat.format(dateInMillis);
        return monthDayString;
    }

    public static String getFormattedWind(Context context, float windSpeed,
float degrees) {
        int windFormat;
        if (Utility.isMetric(context)) {
            windFormat = R.string.format_wind_kmh;
        } else {
            windFormat = R.string.format_wind_mph;
            windSpeed = .621371192237334f * windSpeed;
        }        String direction = "Unknown";
        if (degrees >= 337.5 || degrees < 22.5) {
            direction = "N";
        } else if (degrees >= 22.5 && degrees < 67.5) {
            direction = "NE";
        } else if (degrees >= 67.5 && degrees < 112.5) {
            direction = "E";
        } else if (degrees >= 112.5 && degrees < 157.5) {
            direction = "SE";
        } else if (degrees >= 157.5 && degrees < 202.5) {
```

```
            direction = "S";
        } else if (degrees >= 202.5 && degrees < 247.5) {
            direction = "SW";
        } else if (degrees >= 247.5 && degrees < 292.5) {
            direction = "W";
        } else if (degrees >= 292.5 && degrees < 337.5) {
            direction = "NW";
        }
        return String.format(context.getString(windFormat), windSpeed,
direction);
    }

    public static String getPreferredLocation(Context context) {
        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(context);
        return prefs.getString(context.getString(R.string.pref_location_key),
                context.getString(R.string.pref_location_default));
    }

    public static boolean isMetric(Context context) {
        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(context);
        return prefs.getString(context.getString(R.string.pref_units_key),
                context.getString(R.string.pref_units_metric))
                .equals(context.getString(R.string.pref_units_metric));
    }

    public static String formatTemperature(Context context, double
temperature, boolean isMetric) {
        double temp;
        if (!isMetric) {
            temp = 9 * temperature / 5 + 32;
        } else {
            temp = temperature;
        }
        return context.getString(R.string.format_temperature, temp);
    }         if (weatherId >= 200 && weatherId <= 232) {
            return R.drawable.ic_storm;
        } else if (weatherId >= 300 && weatherId <= 321) {
            return R.drawable.ic_light_rain;
        } else if (weatherId >= 500 && weatherId <= 504) {
            return R.drawable.ic_rain;
        } else if (weatherId == 511) {
            return R.drawable.ic_snow;
        } else if (weatherId >= 520 && weatherId <= 531) {
            return R.drawable.ic_rain;
        } else if (weatherId >= 600 && weatherId <= 622) {
            return R.drawable.ic_snow;
        } else if (weatherId >= 701 && weatherId <= 761) {
            return R.drawable.ic_fog;
        } else if (weatherId == 761 || weatherId == 781) {
            return R.drawable.ic_storm;
        } else if (weatherId == 800) {
            return R.drawable.ic_clear;
        } else if (weatherId == 801) {
            return R.drawable.ic_light_clouds;
        } else if (weatherId >= 802 && weatherId <= 804) {
            return R.drawable.ic_cloudy;
        }
        return -1;
    }
    public static int getArtResourceForWeatherCondition(int weatherId) {
                if (weatherId >= 200 && weatherId <= 232) {
```

```
            return R.drawable.art_storm;
        } else if (weatherId >= 300 && weatherId <= 321) {
            return R.drawable.art_light_rain;
        } else if (weatherId >= 500 && weatherId <= 504) {
            return R.drawable.art_rain;
        } else if (weatherId == 511) {
            return R.drawable.art_snow;
        } else if (weatherId >= 520 && weatherId <= 531) {
            return R.drawable.art_rain;
        } else if (weatherId >= 600 && weatherId <= 622) {
            return R.drawable.art_rain;
        } else if (weatherId >= 701 && weatherId <= 761) {
            return R.drawable.art_fog;
        } else if (weatherId == 761 || weatherId == 781) {
            return R.drawable.art_storm;
        } else if (weatherId == 800) {
            return R.drawable.art_clear;
        } else if (weatherId == 801) {
            return R.drawable.art_light_clouds;
        } else if (weatherId >= 802 && weatherId <= 804) {
            return R.drawable.art_clouds;
        }
        return -1;
    }

    static String formatDate(long dateInMillis) {
        Date date = new Date(dateInMillis);
        return DateFormat.getDateInstance().format(date);
    }
    static boolean isNetworkAvailable(Context context) {
        ConnectivityManager cm =
                (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
        return activeNetwork != null &&
activeNetwork.isConnectedOrConnecting();    }
}
```