Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра дискретной математики и
информационных технологий

Разработка персонифицированной системы сбора и обработки метеорологических данных

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 521 группы
направления 09.03.01 «Информатика и вычислительная техника»
факультета компьютерных наук и информационных технологий
Гетманцева Романа Вячеславовича

Научный руководитель
к. ф.-м.н., доцент _____ А.Д. Панферов
подпись, дата

Зав. кафедрой
к. ф.-м.н., доцент _____ Л.Б. Тяпаев
подпись, дата

Саратов 2018

Введение. В настоящее время практически каждый человек использует в своей повседневной жизни смартфоны или планшеты, которые в ходе своего развития стали своеобразной заменой персонального компьютера. Эволюция «карманных» гаджетов является одним из основных направлений IT – индустрии, а разработка программного обеспечения для этих устройств приносит огромную прибыль разработчикам, что с каждым годом увеличивает статистические показатели рынков мобильной электроники и программного обеспечения с каждым годом.

Актуальность данной работы заключается в демонстрации полного цикла разработки приложений с архитектурой клиент - сервер для современных мобильных устройств с использованием актуальных инструментов и технологий. Эта задача реализована на примере приложения, обеспечивающего формирование архива метеоданных по настройкам клиента. Пользователю предоставляется как возможность стандартного просмотра текущих метеоданных, так и работа с сохранёнными архивными данными.

Основной целевой аудиторией приложения являются пользователи, которым необходима история данных о погоде в конкретных точках. Дополнительно предусматривается возможность сохранения ограниченной функциональности в режиме временного отсутствия доступа к сети интернет.

Преимуществами приложения должны стать:

1. Ежедневное сохранение отчетов на сервера, что обеспечит своевременное пополнение архивных данных в большем объёме.

2. Сохранение выбранных отчетов в телефон для работы с приложением и его данными офлайн.

Целью работы является изучение и демонстрация современных технологий создания приложений для операционной системы Android. Она достигается посредством решения следующих задач:

1. Выбор поддерживаемой версии Android для соблюдения баланса между простотой разработки, которой способствуют более поздние версии ОС, и охватом аудитории пользователей, многие из которых могут использовать

устаревшие версии ОС.

2. Определение оптимального набора функциональных возможностей для создания конкурентоспособного приложения.

3. Детальное определение средств разработки.

4. Загрузка данных в Android приложение.

5. Тестирование Android приложения.

Основное содержание работы. Основным языком программирования, используемым в проекте, является Java. Java – старый и проверенный временем объектно-ориентированный язык программирования. Данный ЯП был создан компанией Sun Microsystems в 1995 году. В данный момент правами на Java обладает компания Oracle. Ключевой особенностью Java является его кроссплатформенность. Код написанный на Java единожды может исполнятся на множестве различных устройств: от микроконтроллеров до искусственных спутников земли. Добиться этого получилось за счёт того, что приложения Java транслируются в специальный байт-код, который в свою очередь исполняется на JVM (java virtual machine). Реализовав для любой отдельно взятой платформы JVM можно получить возможность запускать на ней практически все программы, написанные на Java.[1]

Язык Java является основным языком программирования под Android. Байт-код исполняемый на виртуальных машинах Android (Dalvik для версий 4.4 и ниже, ART для версии 4.4 и выше) нестандартный и содержится в файлах с расширением .dex. Для компиляции данного кода используется дополнительный инструмент – SDK (Software Development Kit), разрабатываемый компанией Google.

SDK – комплект средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ.

Android SDK включает в себя разнообразные библиотеки, документацию и инструменты, которые помогают разрабатывать мобильные приложения для платформы Android.

Spring – это свободно распространяемый фреймворк, созданный Родом Джонсоном (Rod Johnson) и описанный в его книге «Expert One-on-One: J2EE Design and Development». Он был создан с целью устранить сложности разработки корпоративных приложений и сделать возможным использование простых компонентов JavaBean для достижения всего того, что ранее было возможным только с использованием EJB. Однако область применения Spring не ограничивается разработкой программных компонентов, выполняющихся на стороне сервера. Любое Java-приложение может использовать преимущества фреймворка в плане простоты, тестируемости и слабой связанности.[2]

В своем устремлении на сложности, связанные с разработкой на языке Java, фреймворк Spring использует четыре ключевые стратегии:

- легковесность и ненасильственность благодаря применению простых Java-объектов (POJO);
- слабое связывание посредством внедрения зависимостей(DI) и ориентированности на интерфейсы;
- декларативное программирование через аспекты(AOP) и общепринятые соглашения;
- уменьшение объема типового кода через аспекты и шаблоны.

Spring Boot позволяет легко создавать полноценные, производственного класса Spring-приложения, про которые можно сказать - "просто запусти". Туда включили Spring-платформу и сторонние библиотеки, чтобы его запуск требовал минимум усилий. Большинству Spring Boot приложениям требуется совсем маленькая Spring-конфигурация.

Java Persistence API (JPA) — спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных.

Существует несколько реализаций этого интерфейса, одна из самых популярных использует для этого Hibernate. JPA реализует концепцию ORM.

ORM (Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.

ORM избавляет программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки. Кроме того, большинство современных реализаций ORM позволяют программисту при необходимости самому жёстко задать код SQL-запросов, который будет использоваться при тех или иных действиях (сохранение в базу данных, загрузка, поиск и т. д.) с постоянным объектом.

Spring Data JPA реализует слой доступа к данным. Spring Data JPA призвано значительно улучшить реализацию слоя доступа к данным, сократив усилия на написания boilerplate кода. Как разработчик, вы пишете интерфейс репозитория, включая собственные методы поиска, а Spring обеспечивает их автоматическую реализацию.

На этапе разработки серверной части приложения, в качестве RDBMS выбрана in-memory база данных H2. Такой выбор обусловлен простотой развертывания и использования. Простота развертывания заключается в том, что для того, чтобы база данных стала доступна достаточно добавить maven зависимость. Простота использования, в свою очередь, предполагает, что на раннем этапе разработки база данных создается при старте приложения и уничтожается после остановки; на позднем этапе данные сохраняются в файл. Так же, благодаря Spring Data и при правильном описании сущностей, отсутствует необходимость заботиться о структуре и наполнении базы данных.

В дальнейшем необходимо отказаться от реляционной модели хранения и выбрать в качестве СУБД MongoDB. Данный выбор обусловлен следующими причинами:

1. данные связанны лишь в виде «ключ – значение». Где ключом выступают координаты, а значением – метеоданные.

2. для одних координат, будет храниться все большее количество метеоданных.

3. благодаря хранению данных в BSON(binary JSON) формате, структуру данных легко менять «на лету», если в процессе эксплуатации программного комплекса возникнет необходимость добавить новые значения к хранимым данным.

Для создания приложения первоочередной задачей является реализация rest-сервиса получения прогнозов погоды с стороннего API. На этапе прототипа выбрано API «Yahoo weather». В дальнейшем планируется так же получать данные с API «Яндекс.Погода»(после окончания разработки данного API), openweathermap.org и рассчитывать медианные значения показателей.

Далее данные о погоде необходимо загрузить с внешних API и сохранить в базу данных. Так же данные необходимо обновлять по расписанию раз в день.

После генерации пустого Android проекта в Android Studio, первое что требуется сделать разработчику – создать первую Activity (Активность), описать её в файле конфигурации – androidmanifest.xml и описать файл макета для неё.

Activity ассоциируется с отдельным экраном или окном приложения, переключение между которыми происходит в виде перехода от одной activity к другой. Каждое приложение запускается в виде отдельного процесса, что позволяет системе раздавать приоритеты выполнения этим процессам. При запуске пользователем приложения, оно получает от системы высокий приоритет. Благодаря механизму приоритетов системе удаётся добиться работы приложений одновременно, например, при работе с одним приложением не блокировать входящие телефонные звонки. После прекращения работы с

приложением система переводит его в разряд низкоприоритетного и при необходимости освобождает используемые им ресурсы и закрывает его.[3]

В рамках данного проекта были созданы следующие активности:

- MainActivity – activity – контейнер, содержащий один фрагмент, унаследован от класса AppCompatActivity.
- ByNameFragment – Предоставляет пользователю доступ к поиску метеоданных по названию города.
- ByCordFragment и ThemesListActivity – Предоставляет доступ к поиску метеоданных по координатам.
- HystoryFragment – предоставляет доступ к, сохранённым в мобильном устройстве, метеоданным.

Ресурс в приложении Android может представлять из себя любой файл, хранящий в себе сведения необходимые для работы приложения.

Для отрисовки интерфейсов используются файлы разметки, они представлены в виде .xml документов, в которых описано расположение элементов интерфейса на экране и их свойства. Так, например, в листинге кода

Помимо ресурсов разметки слоёв, в этом приложении использовались и другие ресурсы:

- Графические ресурсы, линий разметки и элементов дизайна графического интерфейса приложения.
- Строковые ресурсы, для хранения значения строковых констант.
- Ресурсы стилей, определяющие внешний вид некоторых элементов интерфейса приложения.
- Ресурсы цветов, хранящие RGB константы.

Фрагменты используются в основном для организации отображения нескольких блоков графического интерфейса на одном экране. Данная концепция применима на больших экранах планшетов или телевизоров на ОС Android, как показано на рисунке 1.

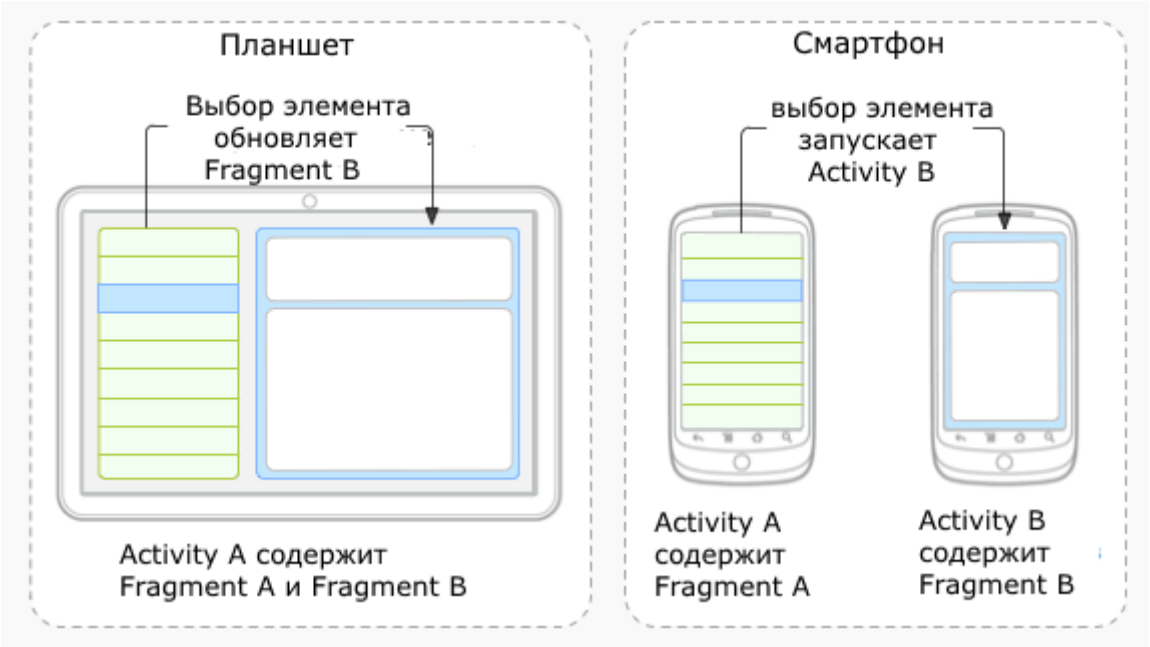

Рисунок 1 - Демонстрация концепции использования фрагментов.

Фрагмент может существовать только в контексте Activity и имеет свой жизненный цикл. Activity может содержать несколько фрагментов.

Приложения Android строятся на базе архитектуры MVC (Model-View-Controller) «Модель-Представление-Контроллер». Согласно канонам MVC каждый объект в приложении должен быть объектом модели, представления или объектом контроллера.

Объект модели содержит данные приложения и «бизнес-логику». Классы модели обычно проектируются для моделирования сущностей, с которыми имеет дело приложение,. Объекты модели ничего не знают о пользовательском интерфейсе, их единственной целью является хранение и управление данными.

Объекты представления умеют отображать себя на экране и реагировать на ввод пользователя, например касание. Android предоставляет широкий выбор настраиваемых классов представлений. Как пример можно привести: AppCompatTextView – поле для отображения текста, AppCompatImageView – изображение, AppCompatButton – кнопка и так далее.

Объекты контроллера связывают объекты представления и модели. Они содержат логику приложения. Контроллеры реагируют на различные события, инициируемые объектами представления, управляют потоками данных между

объектами модели и уровнем представления. В рамках текущего приложения это все фрагменты и activity.

Механизм создания базы данных в Android имеет несколько особенностей. В частности для её открытия и взаимодействия с ней, требуется создать класс, расширяющий возможности SQLiteOpenHelper. В нашем случае это класс DBHelper, в котором реализованный следующие методы:

onCreate() – срабатывает при первом запуске приложения, в нём создаются таблицы и заполняются первоначальными данными.

onUpdate() – метод, срабатывающий в случае если отличаются версии баз данных, и необходимо выполнить обновление базы до более новой версии.

readQuarries() – метод считывающий из файлов, сформированных в редакторе базы данных, sql запросы, и возвращает их как массив строк.

Для взаимодействия с созданной и заполненной базой данных применяется класс DBManager.

Вызывая методы данного класса в соответствующий контроллерах и происходит взаимодействие приложения с базой данных.

В нашем приложении из обоих экранов метеоданные на определенный день сохраняются на мобильном устройстве в результате нажатия на соответствующую кнопку. Сохранение происходит в базу данных и в дальнейшем эти метеоданные доступны на закладке “History”.

Для лучшего доступа только к действительно нужным пользователю метеоданным, а так же для того чтобы не «засорять» память мобильного устройства ненужными данными, была реализована возможность удалить не актуальные метеоданные. Так же, как и кнопка сохранения, кнопка удаления появляется после длительного нажатия на строку с метеоданными. При этом метеоданные всегда можно заново скачать на устройство с сервера. Удаление любых данных из базы данных на сервере не предполагается.

В результате проделанной работы был разработан распределенный прикладной комплекс, состоящий из серверного микросервиса, предоставляющего удобный интерфейс для получения как актуальных, так и

историчных метеоданных и Android приложения, позволяющего хранить данные непосредственно на устройстве пользователя, что открывает доступ к метеоданным даже в условиях отсутствия подключения к сети интернет.

Заключение. В результате выполнения представленного исследования были изучены многие технологии, используемые в разработке полноценных корпоративных Java приложений, а так же в разработке под ОС Android.

Продемонстрирован полный цикл разработки комплексного приложения, начиная от постановки задачи, определения оптимальной функциональности и до полноценного тестирования на реальных данных с достаточно большим их объёмом.

В ходе работы были созданы и серверное, и Android приложения, предоставляющие пользователю функционал для работы с метеоданными. Созданное приложение быстро работает даже на относительно маломощных устройствах, при этом обладая приятным внешним видом и функциональностью. У приложения есть и коммерческий потенциал, при внедрении в него механизмов монетизации и последующей публикации в магазине приложений оно, несомненно, найдёт своих пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 wikipedia – электронная энциклопедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ – Загл. с экрана. – яз. рус.

2 Javasrc-code.net – сайт о программировании [Электронный ресурс] – Режим доступа: http://src-code.net. – Загл. с экрана. – яз. англ.

3 Харди, Б., Филлипс, Б., Стюарт, К. Android программирование для провессифаналов: Пер. с англ. / ООО издательство Питер. – 2-е изд. – СПб.: Питер, 2016. – 640 с.