«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.01 – Информатика и вычислительная техника
Профиль	Вычислительные машины, комплексы, системы и сети
Факультет	КТИ
Кафедра	ВТ

К защите допустить

Зав. кафедрой, д. т. н., профессор М. С. Куприянов

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: ПРИЛОЖЕНИЕ ДЛЯ ПОИСКА IT-МЕРОПРИЯТИЙ НА БАЗЕ ОС ANDROID

Студент _____ М. Ф. Аллаяров
подпись

Руководитель _____ М. Г. Павловский
подпись

Консультанты к. т. н., доцент _____ И. С. Зуев
подпись

_____ С. В. Латынцева
подпись

Санкт-Петербург

2018

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой ВТ, д. т. н., профессор
_____ М. С. Куприянов
«___»_____2018 г.

Студент М. Ф. Аллаяров **Группа** 4305

Тема работы: Приложение для поиска IT-мероприятий на базе ОС Android

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ»

Исходные данные (технические требования): интегрированная среда разработки Android Studio, операционная система Android 5.0, программный интерфейс приложения TimePad API, библиотека JSOUP.

Содержание ВКР: Область мобильной разработки. Выбор средств и инструментов для разработки мобильного приложения. Разработка мобильного приложения. Анализ эффективности разработанных решений.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал, реферат, презентация.

Дополнительные разделы: экономическое обоснование ВКР.

Дата выдачи задания
«___»_____2018 г.

Дата представления ВКР к защите
«___»_____2018 г.

Студент _____ М. Ф. Аллаяров

Руководитель _____ М. Г. Павловский

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой ВТ, д. т. н., профессор

_____ М. С. Куприянов

«___»_____2018 г.

Студент М. Ф. Аллаяров Группа 4305

Тема работы: Приложение для поиска IT-мероприятий на базе ОС Android

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	01.03 – 15.03
2	Рынок мобильной разработки приложений для поиска IT-мероприятий	16.03 – 24.03
3	Выбор средств и инструментов для разработки мобильного приложения	25.03 – 31.03
4	Разработка мобильного приложения	01.04 – 30.04
5	Анализ эффективности разработанных решений	01.05 – 07.05
6	Экономическое обоснование ВКР	08.05 – 15.05
7	Оформление пояснительной записки	16.05 – 31.05
8	Предварительное рассмотрение работы	05.06
9	Представление работы к защите	13.06

Студент _____ М. Ф. Аллаяров

Руководитель _____ М. Г. Павловский

РЕФЕРАТ

Пояснительная записка 70 страниц, 25 рисунков, 8 таблиц, 26 источников, 4 приложения.

IT-МЕРОПРИЯТИЯ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС, ANDROID, ANDROID STUDIO, JSOUP, СБОР ИНФОРМАЦИИ

Объектом исследования являются сервисы для получения информации об IT-мероприятиях, способы получения информации.

Цель работы – разработка мобильного приложения для поиска IT-мероприятий в сети Интернет на базе ОС Android, которое позволит пользователю централизованно, быстро и в удобном виде получать нужную ему информацию.

В работе рассмотрены различные способы получения информации из различных источников, методы их агрегирования. Описан алгоритм, решающий задачу сбора и унификации информации. Итогом является разработанное приложение для ОС Android, с помощью которого можно просматривать, искать, хранить мероприятия, подписываться на интересующие темы.

Дополнительным разделом данной работы выступает экономическое обоснование разработки. В нём были рассчитаны полные затраты на проект.

ABSTRACT

Graduation work «Searching the IT-events application based on Android OS» is devoted to the development of a mobile application which will allow the user to receive the information necessary to him about IT-events centrally, quickly and in a convenient way.

The paper considers various ways of obtaining information from various sources, methods of their aggregation. There is the described algorithm that solves the problem of collecting and unifying information. The result is the developed application for Android OS that allows to view, search, store events, subscribe to interesting topics.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API – программный интерфейс приложения, набор готовых классов, функций и структур, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах

DOM – программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов

IDE – интегрированная среда разработки, комплекс программных средств для разработки ПО

HTML – стандартизированный язык разметки документов в Интернет-сети

HTTP – протокол прикладного уровня передачи данных

JSON – текстовый формат обмена данными, основанный на JavaScript

JVM – виртуальная машина Java

ORM - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных»

SDK – набор средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ

UI – пользовательский интерфейс

ОС – операционная система

ПО – программное обеспечение

ПК – персональный компьютер

ЦП – центральный процессор

ВВЕДЕНИЕ

Одними из главных черт характера, которые продвигают человечество вперед, являются любознательность и желание учиться чему-то новому. Непрерывный доступ к информации и возможность получить ее в любое время играют важную роль в развитии новых навыков и приобретении знаний. Также очень важным делиться накопленными знаниями с окружающими, ведь как известно, ни одно великое изобретение не было создано в одиночку.

В настоящее время разработчики программного обеспечения решают эти проблемы с помощью организационных мероприятий: лекции, семинары, конференции, хакатоны, вебинары, а также другие образовательные мероприятия.

Однако в современном мире подобных мероприятий огромное количество, но публикуются они на разных площадках, которые нередко имеют свои собственные приложения. Поэтому разработчику достаточно проблематично быстро найти и отследить нужную ему информацию. Очень неудобно посещать несколько сайтов и иметь несколько разных приложений для разных источников для того, чтобы отследить все надвигающиеся события.

Так как влияние мобильных устройств в текущий момент велико, и они обеспечивают непрерывный доступ к данным, то очевидным решением этой проблемы является создание агрегата – мобильного приложения, которое само будет работать с несколькими выбранными источниками одновременно.

Необходимо учесть, что пользователю не интересно только просматривать списки мероприятий. Для него представляет интерес поиск событий, сохранение интересующих, подписка на оповещение по выбранным темам. Так как на рынке в свободном доступе нет ни одного такого сервиса, то разрабатываемое приложение будет наделено данным функционалом. В связи с этим, разработку является актуальной.

Целью бакалаврской работы является разработка мобильного приложения для поиска IT-мероприятий в сети Интернет на базе ОС Android, которое

позволит пользователю централизованно, быстро и в удобном виде получать нужную ему информацию.

Для достижения цели были поставлены следующие задачи:

- разработать универсальную модель данных для мероприятий;
- разработать гибкую архитектуру приложения, позволяющую легко добавлять и изменять источники данных;
- предоставить пользователю возможность просматривать ленту событий, вести поиск, переходить на страницу источника мероприятия, работать с избранным;
- предоставить пользователю возможность тонкой настройки ленты событий.

В данной работе объектом исследования являются сервисы для получения информации об IT-мероприятиях, способы получения информации. Предметом исследования является разработка приложения для поиска IT-мероприятий в сети Интернет на базе ОС Android.

Первый раздел является обзором существующих инструментов, аналогов, в нем ставятся требования для разработки.

Второй раздел посвящен выбору технологий и инструментов для решения поставленных задач.

Третий раздел посвящен процессу разработки приложения.

Четвертый раздел содержит в себе анализ получившегося решения, сравнение способов получения данных по различным характеристикам.

Пятый раздел является экономическим обоснованием разработки.

Заключение содержит основные выводы, сделанные в процессе работы.

1 Рынок мобильной разработки приложений для поиска IT-мероприятий

В данном разделе произведен сбор и анализ материала по теме ВКР. Он включает в себя задачи поиска источников информации, их анализ, переработку и осмысление. По результатам исследования сделан вывод.

1.1 Мобильные ОС

Мобильная операционная система — это операционная система для различных мобильных устройств: смартфоны, планшеты, КПК. Мобильные ОС соединяют в себе функциональность ОС для ПК и такие функции для мобильных устройств как сенсорный экран, сотовая связь, Bluetooth, Wi-Fi и так далее.

Из всех мобильных устройств, смартфоны занимают существенную долю потребительского рынка – около 60% людей на планете имеет свой личный смартфон [1]. Логично, что доля мобильных приложений занимает весомое место на рынке.

Чтобы написать мобильное приложение, разработчик должен в первую очередь изучить мобильный рынок: какие мобильные ОС популярны в текущий момент, их версии, какие есть инструменты для разработки под конкретные ОС.

В настоящее время, самыми популярными мобильными ОС среди пользователей являются Android – 75.66% и iOS – 19.23%. Остальные мобильные ОС не занимают существенной доли рынка [2]. График статистики использования мобильных ОС представлен на рисунке 1.1.

Подробнее рассмотрим ОС Android и iOS.

1.1.1 ОС Android

ОС Android широко известна как ОС для смартфонов, но, помимо этого, она также распространяется на планшетах, электронных книгах, цифровых проигрывателях, наручных часах и других устройствах. В основе лежит ядро Linux [3] и собственная реализация виртуальной машины Java от Google.

Изначально разрабатывалась компанией Android Inc., которая потом была выкуплена корпорацией Google в 2005 году. ОС Android позволяет создавать приложения на языке Java, управляющие устройством с помощью библиотек, разработанных Google. Android Native Development Kit позволяет портировать библиотеки и компоненты приложений, написанные на Си и других языках.

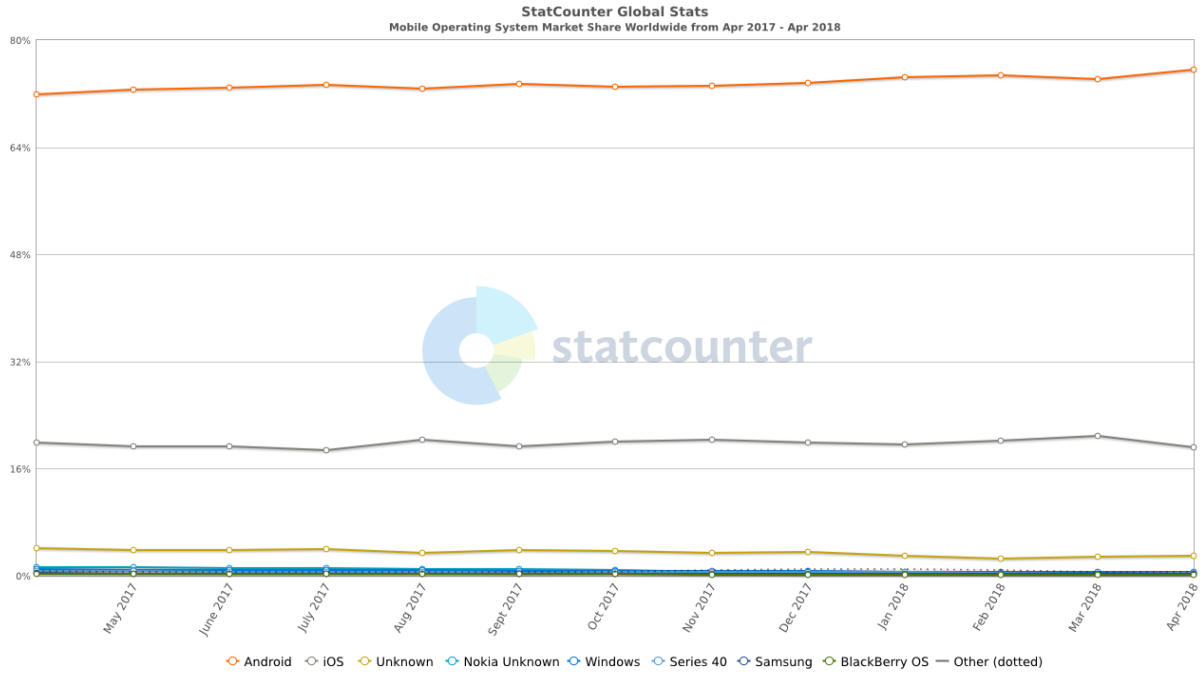

Рисунок 1.1 – Статистика использования мобильных ОС

С 2008 года приложения на ОС Android распространяются с помощью онлайн-магазина приложений — Android Market. По соглашению, разработчики получают 70 % прибыли, операторы сотовой связи — 30 % [4].

В марте 2012 года компания Google объединила мультимедийные сервисы «Книги», «Android Market», «Музыка» и другие в единый сервис Google Play. Интернет-магазин Google Play работает в 190 странах и насчитывает более 700 тысяч приложений, а за время работы сервиса набралось около 25 млрд скачиваний.

В отличие от обычных Linux приложений, приложения Android подчиняются дополнительным правилам:

- обмен данными между приложениями - Content Providers;

- доступ к таким ресурсам, как файлы XML, PNG, JPEG - Resource Manager;
- доступ к строке состояния - Notification Manager;
- управление активными приложениями - Activity Manager.

SDK (Software Development Kit) – свободный для скачивания инструментарий для разработки, предлагаемый Google, который предназначен для x86-машин под операционными системами Linux, macOS, Windows. Для разработки требуется JDK 5 или более новый.

В 2009 году был опубликован Android Native Development Kit (NDK) [5] — пакет инструментариев и библиотек, позволяющий реализовать часть приложения на языке C/C++. NDK рекомендуется использовать для разработки участков кода, критичных к скорости.

1.1.2 iOS

iOS (ранее iPhone OS) – это мобильная ОС, которая была создана компанией Apple эксклюзивно для их аппаратного обеспечения. Она используется на многих мобильных устройствах компании, включая iPhone, iPad, iPod Touch и так далее. Как было сказано ранее, является второй самой популярной мобильной ОС, уступая Android. Была представлена в 2007 году для iPhone, позже и для других устройств Apple.

В iOS используется ядро XNU, основой которого является микроядро Mach, содержащее программный код, разработанный компанией Apple, а также код из ОС NeXTSTEP и FreeBSD.

Приложения распространяются через App Store. По соглашению, разработчики могут устанавливать любую цену, превышающую минимальную установленную, за их приложения, из которой они будут получать 70%. Кроме того, они могут распространять своё приложение бесплатно, в этом случае они должны платить только членские взносы, которые составляют 99 долларов в год [6].

Для разработки приложений на iOS используется набор программного обеспечения iOS SDK (ранее iPhone SDK), разработанный компанией Apple. Все пользователи Mac ПК могут бесплатно скачать SDK, однако пользователи Windows ПК не имеют такой возможности.

SDK содержит в себе набор инструментов, позволяющий разработчикам получить доступ к различным функциям и сервисам iOS устройств. Для того, чтобы протестировать приложения, получить техническую поддержку, распространять приложения через App Store, необходимо оформить подписку на Apple Developer Program.

1.1.3 Выводы

В связи с тем, что ОС Android популярнее, чем iOS, было принято решение вести разработку на Android ОС, чтобы приложение могло покрыть большую аудиторию.

1.2 Способы сбора информации из различных источников

Целью бакалаврской работы является разработка мобильного приложения для поиска IT-мероприятий в сети Интернет на базе ОС Android. Чтобы решить эту задачу, нужно уметь собирать информацию из нескольких источников данных. Для этого существует несколько подходов.

1. Сбор информации с помощью других приложений.

ОС Android позволяет одним приложениям «подписываться» на другие, чтобы получать информацию о случившихся событиях, например, что произошел входящий звонок или пришло новое оповещение. Однако для этого, один разработчик должен явно знать структуру другого приложения, как оно работает внутри, явно указать это приложение в манифесте своего приложения, предоставить права доступа на прослушивание событий, генерировать события и оповещать приложение.

2. Использование API различных сайтов.

API – набор готовых классов, процедур, функций, который предоставляет приложение или ОС для использования во внешней среде. Таким образом, он определяет функционал, который предоставляет приложение, абстрагируясь от подробностей реализации этой самой функциональности. В свою очередь, различные API определяют свои правила использования своих сервисов. Они могут потребовать зарегистрировать приложение, используя специально сгенерированный ключ, установить библиотеку для работы, использовать свои собственные токены для общения между сервисами. Поэтому важно понимать, что подбор правильного API помимо функционала включает себя еще и ограничения по работе с API.

3. Синтаксический анализ сайтов.

Для этого можно использовать синтаксический анализатор – программу (часть программы), которая будет сопоставлять линейную последовательности лексем естественного или формального языка с его формальной грамматикой. В случае синтаксического анализа сайта, можно легко обходить DOM-дерево, извлекая из него информацию, ведя поиск. Однако стоит учитывать, что все эти операции не так удобны в использовании и требуют значительных затрат ресурсов, в отличие от работы с API.

1.3 Существующие на рынке приложения для поиска IT-мероприятий

Чтобы лучше понять текущую ситуацию на рынке приложений для поиска IT-мероприятий, были выделены следующие критерии для анализа:

- возможность поиска по именам, местам;
- возможность подписки на оповещения на различные темы;
- наличие ленты новостей;
- использование нескольких источников данных;
- возможность регулировать количество событий в ленте новостей;

- возможность настройки источников данных;
- высокая скорость работы.

Так как идея данного приложение не является новой, на рынке уже существуют приложения, предоставляющие пользователю просматривать ленту событий, вести поиск, подписываться на интересующие события. Такими приложениями являются:

- IT Events Belarus [7];
- IT-Мероприятия [8].

Анализ функционала приложений будет произведен в следующих пунктах.

1.3.1 IT Events Belarus

IT Events Belarus – это приложение, созданное Maria Sitkina для отслеживания всех IT-мероприятий Беларуси. Имеет весьма приятный и понятный интерфейс, отмечается высокая скорость работы, поиск ведется по названию и месту события. Однако, в приложении нет возможности подписаться на какие-либо оповещения, используется один источник данных, соответственно, нельзя регулировать источники, нет функционала для регулирования количества мероприятий в ленте.

Скриншот приложения приведен на рисунке 1.2.

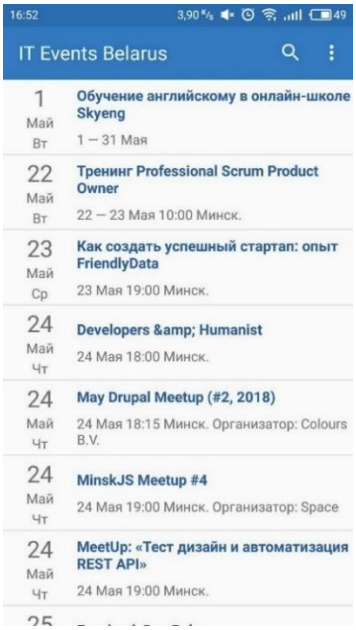

Рисунок 1.2 – Приложение IT Events Belarus

1.3.2 IT-Мероприятия

Приложение IT-Мероприятия было создано LiteSoftTeam. Имеется возможность отфильтровывать события по городу и по категориям, в ленте новостей есть превью, есть возможность перейти на источник события. К сожалению, отсутствует возможность создать свой фильтр или отредактировать существующий, нет функционала для поиска мероприятий, какой-либо настройки ленты новостей, все данные приходят из одного источника. Есть возможность получать оповещения, однако, только для определенных событий. Для этого необходимо перейти к описанию события и нажать соответствующую кнопку, что нельзя назвать удобным. Также в приложении присутствует огромное количество рекламы, которую нельзя отключить.

Скриншот приложения приведен на рисунке 1.3.

Рисунок 1.3 – Приложение IT-Мероприятия

1.3.3 Выводы

Рассмотренные приложения решают проблему поиска IT-мероприятий, однако, не в полном объеме. Они представляют собой мобильную версию платформ, с которых получают информацию о событиях.

Данное исследование помогло сделать вывод, что в текущий момент на рынке не существует приложения, которое удовлетворяет всем вышеперечисленным критериям в полном объеме.

1.4 Требования, выдвигаемые при разработке приложения

Данный пункт содержит различные требования, которые необходимо соблюсти при разработке мобильного приложения по теме ВКР.

1.4.1 Требования к функциональным характеристикам

Приложение должно предоставлять следующие возможности:

- поиск IT-мероприятий по имени или по метаданным мероприятия: место, источник данных, совпадения в описании;
- сохранение мероприятий в избранное с возможностью удаления;
- просмотр мероприятий в виде ленты новостей;
- регулировка количества отображаемых мероприятий в ленте;
- переход на страницу источника мероприятия в браузере из приложения;
- изменение источников данных для мероприятий;
- загрузка данных из разных источников;
- подписка на интересующую тему с целью получения оповещений о новых мероприятиях.

1.4.2 Требования к надёжности

Приложение должно учитывать возможность возникновения ошибок и уметь обрабатывать их. Должна быть реализована возможность корректного считывания данных и загрузки результатов. Приложение должно быть отказоустойчивым, не вызывать сбоев в работе ОС.

1.4.3 Условия эксплуатации

Приложение будет использоваться в нормальных условиях при отсутствии повышенных нагрузок и вредоносных внешний воздействий со стороны аппаратных и программных средств.

1.4.4 Требование к информационной и программной совместимости

Выходная информация приложения должна быть удобна для визуального восприятия. Приложение должно быть выполнено на языке программирования высокого уровня и быть совместимым с операционной системой Android. Обязательными требованиями при разработке кода приложения являются использование следующих конструкций языка:

- закрытые и открытые члены классов;
- наследование;
- динамическое создание объектов.
- конструкторы с параметрами;
- абстрактные базовые классы;
- абстрактные методы.

1.4.5 Требования к нефункциональным характеристикам

- Приложение должно не испытывать трудностей с производительностью;
- приложение должно одинаково интерпретировать разные входные данные с различных сайтов по единому шаблону;
- приложение должно иметь отзывчивый и интуитивно понятный интерфейс;
- в условиях высокого качества Интернет-соединения приложение должно производить поиск, анализ и выдачу результата за время, меньшее чем 5 секунд, в условиях низкого качества Интернет-соединения за время, меньшее чем 10 секунд.

2 Выбор средств и инструментов для разработки мобильного приложения

В данном разделе будут рассмотрены технологии и их аналоги, позволяющие решить поставленные задачи на ОС Android, будет сделан их выбор на основе выделенных критериев.

2.1 Анализ основных составляющих

Необходимо определить программное обеспечение, которое понадобится для реализации поставленной задачи.

Выделим несколько основных составляющих:

- операционная система. На данном этапе необходимо обосновать выбор мобильной ОС, определить приоритетную версию ОС, под управлением которой будет разрабатываться и отлаживаться проектируемое приложение;
- язык разработки. Необходимо выбрать язык разработки для мобильной ОС, который поможет решить поставленную задачу с наибольшей скоростью и с наименьшими усилиями;
- среда разработки. Для ведения эффективной разработки и поддержки всех возможностей необходимо пользоваться современными средами разработки. Они избавят от забот по настройке и сборке проекта, дав возможность сконцентрировать усилия непосредственного на разработке;
- источники данных. Необходимо выбрать несколько способов получения данных из разных источников для того, чтобы экспериментально сравнить их характеристики: влияние на производительность приложения, нагрузка на сеть Интернет.

2.2 Выбор операционной системы

В пункте «1.1 Мобильные ОС» были рассмотрены лидирующие ОС для смартфонов. Были выделены критерии, по которым решено было выбрать ОС Android как платформу для разработки.

1. Доступность SDK.

Для разработки на ОС Android не требуется Mac ПК или какое-либо другое специальное оборудование. Все инструменты являются бесплатными, работают на всех ОС ПК.

2. Лояльная ценовая политика.

В отличие ежегодной подписки в App Store в размере 99$, в Google Play для публикации приложений нужно внести разовую оплату за соглашение разработчика в размере 25$.

Стоит учесть, что версии ОС Android различаются функционалом, который доступен для использования и разработки. Однако, Google придерживается политики обратной совместимости: новые версии ОС поддерживают приложения, разработанные на предыдущих версиях ОС.

На рисунке 2.1 представлена информация по проценту использования различных версий Android на 07.05.2018 по данным, предоставленных с официального сайта Google по Android-разработке [9].

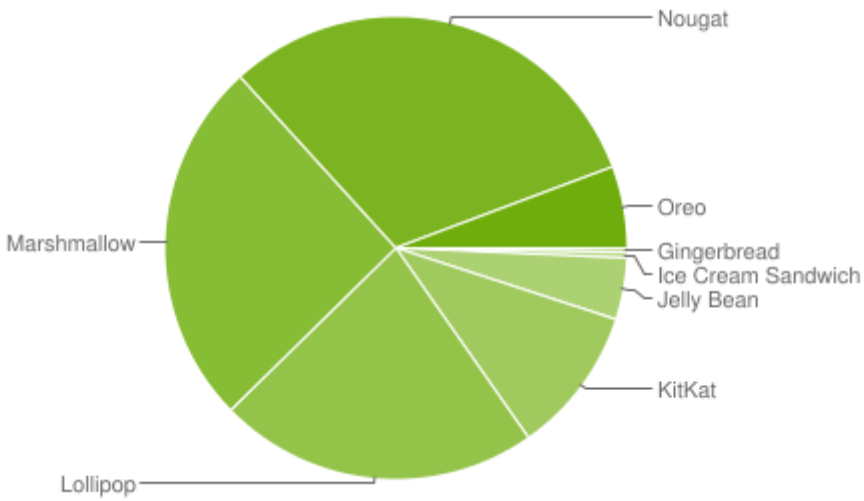

Рисунок 2.1 – Процент использования различных версий ОС Android

Как видно из рисунка, приоритетными версиями являются Lolipop (5.0), Marshmallow (6.0), Nougat (7.0). Так как все новые версии ОС Android поддерживают функционал предыдущих, выбор сделан за Lolipop (5.0). Таким образом, приложение будет работать на ~85% устройств.

2.3 Выбор языка программирования для разработки

Приложения на ОС Android работают на JVM, поэтому необходимо выбрать удобный и современный JVM-язык, который решает поставленные задачи.

Список современных JVM языков:

- Clojure - функциональный язык, диалект Lisp;
- Groovy - сценарный язык;
- JRuby - реализация Ruby;
- Jython - реализация Python;
- Kotlin - объектно-ориентированный язык для индустриальной разработки;
- Scala- объектно-ориентированный и функциональный язык;
- Java.

Выбор стоит производить исходя из легкости написания кода, производительности на слабых устройствах, целей языка и многих других факторов.

Перед тем как приступить к окончательному выбору, необходимо выделить четкий набор факторов, которые необходимо учитывать в первую очередь. Приведем здесь перечень наиболее часто используемых при выборе:

- сложность используемого языка программирования;
- затраты на обучение персонала;
- производительность на слабых устройствах;
- объем занимаемого namespace функций.

На основе анализа были выбраны Kotlin и Java, т.к. они являются в первую очередь языками ООП и активно поддерживаются в разработке на ОС Android.

По следующим причинам было решено выбрать язык Kotlin вместо Java.

1. Современность

Язык Kotlin предоставляет такие возможности современных языков программирования, как система безопасного Nullability, изменяемые\неизменяемые свойства, объекты и так далее [10].

2. Совместимость

Языки Java и Kotlin обратно совместимы: код на одном из этих языков будет запускаться на другом.

3. Сообщество

Команда разработчиков JetBrains с радостью принимает идеи от сообщества, сообщения об ошибках, также сам язык является открытым проектом на GitHub [11].

4. Поддержка

Все библиотеки языка Kotlin работают на любом уровне Android API, официально поддерживается Google [12].

5. Читаемость

Код на языке Kotlin читается гораздо проще, чем код на языке Java, требуется меньше времени для понимая кода.

6. Интеграция с Android Studio

Так как язык Kotlin и IDE Android Studio разрабатываются командой разработчиков JetBrains, Kotlin имеет великолепную интеграцию с Android Studio.

2.4 Выбор среды разработки

Популярными средами разработки для ОС Android являются Eclipse и Android Studio.

Было решено выбрать Android Studio 3.0 для разработки по следующим причинам.

1. Официальная поддержка Google

Как известно, Google отказался поддерживать инструменты Eclipse для разработки на ОС Android в пользу Android Studio [13].

2. Высокая интеграция с языком Kotlin

Как было сказано в предыдущем пункте, язык Kotlin и Android Studio имеют великолепную интеграцию.

3. Большое количество удобных инструментов

Android Studio имеет много удобных в использовании инструментов, которые упрощают разработку и поиск ошибок, умеет генерировать множество файлов: иконки, файлы макетов, автоматически создавать файлы для разных версий по выбранному критерию. Также существует инструмент для подробного изучения различных характеристик приложения – Android Profiler [14].

2.5 Выбор источников данных

В пункте «1.2 Способы сбора информации из различных источников» были рассмотрены три основных способа получения данных: сбор информации с помощью других установленных приложений, использование API и синтаксический анализ сайтов. Так как первый способ сложный в использовании, требует много ручной работы, обладает высокой связностью, было решено не использовать его в этой работе.

Для остальных пунктов необходимо провести анализ и выбор соответствующего инструмента.

2.5.1 Выбор API

Для решения поставленной задачи, выбранный API должен подходить по нескольким критериям:

- возможность искать только IT-мероприятия;

- отсутствие необходимости регистрации для использования;
- встроенная возможность поиска и сортировки;
- возможность получить всю требуемую информацию о мероприятии: название, время, место, ссылка на источник, превью и т.д.;
- возможность связаться с командой поддержки в случае каких-либо проблем, связанных с использованием API;
- приемлемые ограничения со стороны API.

Был произведен поиск соответствующего API. Всем предъявленным критериям удовлетворял TimePad API [15]. Сайт TimePad [16] является сервисом для организации событий и продажи электронных билетов.

2.5.2 Синтаксический анализ

Для того, чтобы воспользоваться этим способом получения информации, необходимо провести анализ существующих синтаксических анализаторов, сделать выбор анализатора и источника данных для него.

Так как создание своего синтаксического анализатора – достаточно сложная и долгая задача, было принято решение использовать готовые реализации.

Существует множество синтаксических анализаторов для HTML. В таблице 2.1 приведены популярные анализаторы, которые имеют реализацию на языке Java [17].

Таблица 2.1 – Популярные анализаторы, поддерживающие язык Java

Имя	Дата последнего обновления	Возможности			
		Анализ HTML	Анализ HTML5	Очистка HTML	Обновление HTML
HtmlUnit	2016-05-27	Да	Неизвестно	Нет	Нет
HtmlCleaner	2015-08-24	Нет	Нет	Да	Неизвестно
Jaunt API	2013-08-01	Да	Неизвестно	Да	Нет
JSOUP	2018-04-15	Да	Да	Да	Да
JTidy	2012-10-09	Нет	Неизвестно	Неизвестно	Неизвестно
TagSoup	2011-07-07	Нет	Неизвестно	Неизвестно	Неизвестно
Validator.nu HTML Parser	2012-06-05	Да	Да	Неизвестно	Неизвестно

Под возможностью «очистка HTML» подразумевается «дезинфекция» (создание стандартной совместимой веб-страницы, уменьшение спама и так далее) и «очистка» (удаление излишних презентационных тегов, удаление кода XSS) HTML-кода.

Под возможностью «Обновление HTML» подразумевается обновление HTML4.X до XHTML или HTML5, преобразовывая устаревшие теги (например, CENTER) в допустимые (например, DIV со стилем = "text-align: center;").

Исходя из таблицы выше, в качестве синтаксического анализатора решено было выбрать JSOUP [18] – Java библиотека для работы с HTML. Эта библиотека позволяет:

- получать и разбирать HTML из URL, файла или строки;
- находить и извлекать информацию используя DOM-навигацию или CSS-селекторы;
- манипулировать HTML элементами, атрибутами, текстом;
- формировать HTML.

В процессе работы, JSOUP создает синтаксического дерево, которым можно в дальнейшем манипулировать.

Также проведен поиск и анализ различных сайтов, которые могли бы выступать в качестве источника данных для синтаксического анализатора. Многие из них имели весьма сложный и запутанный интерфейс, другие не имели достаточного количества данных, необходимых для создания мероприятия в приложении. Поэтому в результате анализа был выбран сайт IT-events [19].

Его преимущества:

- сайт посвящен только IT-мероприятиям по самым различным тематикам. Таким образом, отпадает необходимость фильтровать данные по типу мероприятия;
- информация на сайте расположена в порядке убывания, начиная с самых свежих новостей. Это очень удобно, так как не придется вручную сортировать данные, обходить все события, чтобы получить самые свежие.
- на сайте можно получить всю интересующую информацию: название, место, время, источник, превью и так далее;
- сайт не требует регистрации для использования;
- сайт имеет интуитивно понятную верстку, унифицированный графический интерфейс.

Благодаря совокупности всего вышесказанного, сайт является лучшим кандидатом для проведения синтаксического анализа.

2.5.3 Вывод

Несмотря на огромное количество источников данных в сети Интернет, лишь малое количество подходит для решения поставленных задач. Требуется весьма много времени для анализа и поиска подходящих источников.

3 Разработка мобильного приложения

В данном разделе будет рассмотрен процесс создания архитектуры приложения, разработанные решения и алгоритмы, будут приведены результаты выполненной работы.

3.1 Стандартная модель данных

Приложение будет считывать данные из нескольких различных источников, поэтому важно привести их к общему виду, стандартизовать набор минимальных данных. Исходя из этого, были выделены следующие общие поля, которые необходимы для составления мероприятия:

- название;
- время проведения;
- место проведения;
- источник данных.

Однако следует учесть, что событие в ленте новостей в приложении также должно иметь превью, должна быть возможность сохранить событие в избранное, открыть полное описание в браузере. Таким образом, необходимо иметь следующие дополнительные поля:

- URL-адрес превью;
- ссылка на полное описание события;
- переменная булевого типа для отслеживания того, сохранено ли событие в избранное.

Таким образом, была создана следующая модель данных, представленная на рисунке 3.1. Исходный код модели приведен в приложении А.

Благодаря встроенным механизмам языка Kotlin, которые позволяют задавать полям значения по умолчанию, решается проблема излишнего написания кода, излишних проверок при инициализации объекта и так далее.

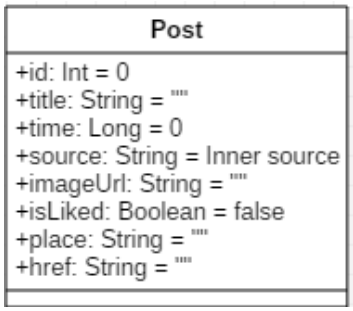

Рисунок 3.1 – Стандартная модель данных

Стоит отметить, что поле id необходимо для того, чтобы сохранять объект мероприятия в базу данных. Подробная информация об используемой базе данных будет рассмотрена в следующих пунктах.

3.2 Архитектура приложения

Построим use-case диаграмму для описания структуры приложения, показанную на рисунке 3.2.

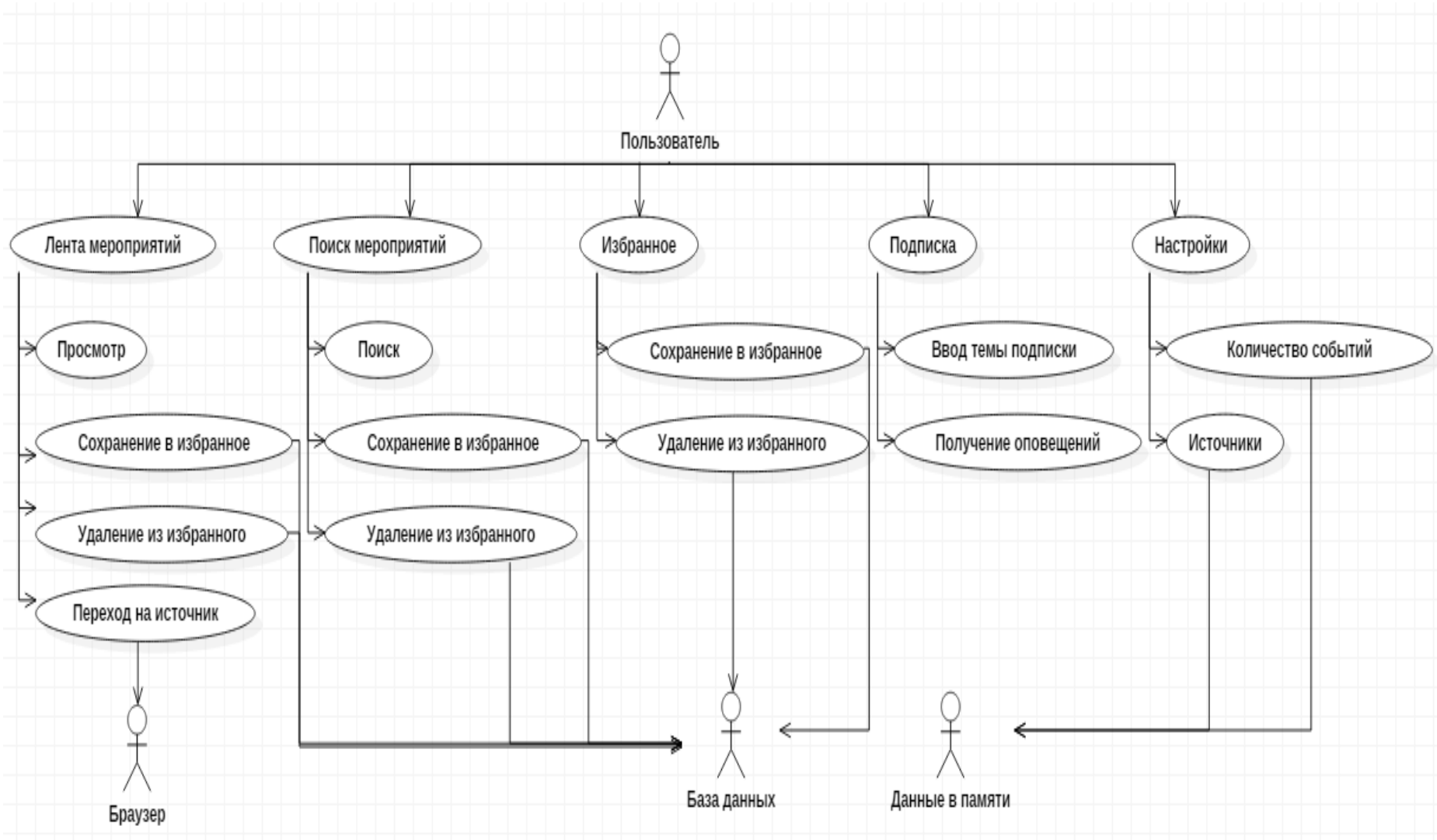

Рисунок 3.2 – Диаграмма прецедентов

Проанализируем полученную диаграмму. Всего существует пять основных экранов: лента мероприятий, поиск, избранное, подписка и настройки. Экран настроек является особым, так как в нем нельзя просматривать события.

Экраны имеют схожий функционал. Они позволяют просматривать информацию в виде ленты событий, сохранять в избранное и удалять из него. Можно сделать вывод, что необходимо выделить общий абстрактный класс, который будет ответственным за создание шаблона ленты мероприятий, ее настройку, задавать слушателей для различных событий, управлять анимацией загрузки данных, показа данных и так далее.

Android ОС имеет несколько видов памяти [20]:

- внутренне файловое хранилище: хранение приватных файлов приложения в файловой системе устройства;
- внешнее файловое хранилище: хранение файлов в общем внешнем хранилище. Обычно используется для общих файлов пользователя, таких как фотографии;
- общие предпочтения (ориг. Shared preferences): хранение приватной простой информации в памяти приложения в виде пар ключ-значение;
- базы данных: хранение структурированной информации в приватной базе данных.

Из диаграммы видно, что приложение работает как с базой данных, так и данными в памяти, а точнее с механизмом Shared Preferences. Он очень удобен в использовании, так как для сохранения настроек не требуется много данных, они не должны быть обязательно структурированы, они являются простыми типами. Также использование механизма Shared Preferences для настроек приложения рекомендуется для использования разработчиками [21].

Анализ и выбор базы данных, подходящей для решения поставленной задачи, будет сделан в следующих пунктах.

Настройки работают со свойствами лент каждого из экранов. Из этого следует, что экран настроек будет изменять свойства некоторого класса, который будет ответственным за предоставление общих параметров, таких как количество событий и источники событий, другим экранам.

Будет правильнее выделить механизм получения данных в отдельные классы, которые будут взаимодействовать с общим классом-агрегатором этих данных. Это позволит построить более гибкую архитектуру, в будущем будет легче вносить изменения в код.

На диаграмме представлено «получение оповещений». Оповещения должны приходить даже тогда, когда приложение находится в выключенном состоянии. Для этого можно использовать механизм служб и оповещений, которые предоставляет Android ОС.

3.2.1 Вывод

Таким образом, можно сделать следующие выводы по результатам анализа:

- для экранов необходимо создать абстрактный класс, который будет ответственным за создание шаблона ленты новостей, регистрацию и отписку слушателей событий загрузки новых данных, анимацию загрузки;
- требуется создать интерфейс, который будут реализовывать все классы, занимающиеся сбором информации из конкретного источника;
- необходимо создать класс, который будет работать с общими настройками, получать информацию от классов, которые занимаются получением данных из своего источника, собирать эту информацию и рассылать событие с результатом об успешном или неудачном сборе;

- за создание, отправку и контроль оповещений о получении новых событий по теме, на которую подписан пользователь, должен быть ответственным отдельный класс.

3.3 Создание абстрактного класса для экранов

Большинство экранов обладает схожей архитектурой и задачами: они должны отображать ленту мероприятий, показывать анимации, регистрировать слушателей и так далее. Значит, можно выделить этот функционал в абстрактный класс, который будет их родителем.

Таким образом, был разработан абстрактный класс BaseFragment, представленный на рисунке 3.3. Исходный код класса приведен в приложении Б.

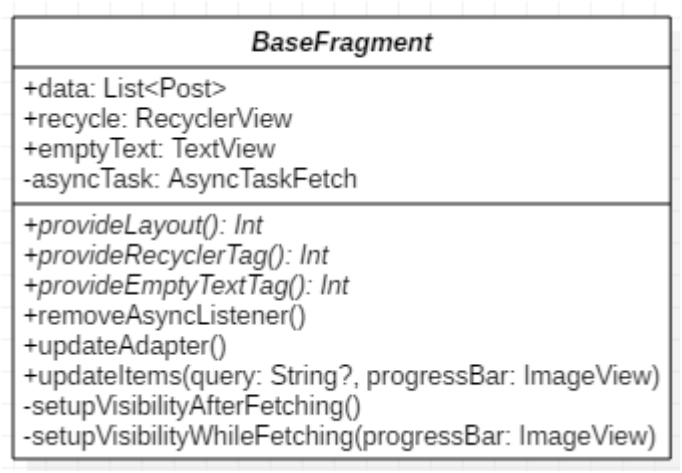

Рисунок 3.3 – Диаграмма абстрактного класса BaseFragment

Этот класс задает базовый графический интерфейс, содержащий в себе:

- сообщение о том, что нет данных для отображения в ленте мероприятий;
- анимацию загрузки данных;
- ленту мероприятий, реализованную с помощью RecyclerView.

Класс BaseFragment управляет:

- видимостью элементов, анимации;
- базовым поведением ленты мероприятий;

- слушателями для AsyncTaskFetch.

Он является абстрактным, так как для создания графического интерфейса необходимо предоставить идентификатор макета интерфейса для заполнения, идентификатор ленты мероприятий, идентификатор объекта, ответственного за сообщение о том, что нет данных для отображения.

Подробнее о механизмах работы RecyclerView, AsyncTaskFetch, загрузки и предоставления данных будет рассказано в следующих пунктах.

3.4 Анализ и выбор базы данных

Для использования базы данных в Android ОС рассматривались два подхода: SQLite или ORM. Каждый из них имеет свои достоинства и недостатки.

Сильные стороны использования SQLite:

- встроенный в IDE механизм для просмотра и редактирования информации в базе данных;
- не требует зависимостей, так как является встроенным в Android ОС решением;
- разработчик может создать именно такую схему данных, которая ему требуется;
- разработчик имеет полный контроль над запросами к базе данных, так как они являются собственноручно написанными.

Слабые стороны использования SQLite:

- требуется написать очень много шаблонного кода;
- необходимо создать класс ContentProvider, являющийся еще одним слоем;
- ручное обновление и версионирование схемы данных;
- SQL – это язык программирования, чтобы его изучить потребуется некоторое время;
- SQL запросы могут быть очень длинными, сложными и большими;

- тяжело тестировать.

Так как приложение использует простые схемы данных, простые запросы к базе, а оставшиеся плюсы не являются критичными, то не остается причин выбрать подход SQLite.

Рассмотрим ORM подход. Преимущества перед SQLite:

- избавляет от большого количества шаблонного кода;
- решает множество задач автоматически: версионирование и обновление схемы данных, составление запросов, создание схемы данных и так далее.

Таким образом, ORM подход является более предпочтительным, нежели SQLite. Однако необходимо провести анализ, выбрать и изучить библиотеку для работы с базой данных.

В качестве ORM базы данных была выбрана dbFlow по следующим причинам [22].

1. Интеграция с Kotlin

БД dbFlow поддерживает работу с Kotlin «из коробки», то есть не требует установки дополнительного ПО.

2. Расширяемость

БД dbFlow не имеет никаких ограничений на наследование классов таблиц. Можно расширить классы, которые не являются моделями для базы данных, и использовать их как таблицы.

3. Скорость

БД dbFlow имеет поддержку ленивой загрузки данных, кэширования данных, что увеличивает производительность и ускоряет работу.

4. Поддержка аннотаций

БД dbFlow построена на возможности Java генерировать код при помощи аннотаций. Таким образом, это избавляет от необходимости написания шаблонного кода, является очень удобным способом работы с базой данных.

5. Запросы в стиле SQLite

Запросы в dbFlow сделаны таким образом, чтобы быть максимально похожим на язык запросов SQLite. Поэтому разработчику, знакомым с SQLite, будет легче начать работу с этой библиотекой.

6. Хорошая документация

Документация на официальном сайте написана понятным доступным языком, содержит множество содержательных примеров и ответы на различные вопросы.

3.5 Агрегирование данных

Под агрегированием данных понимается процесс сбора, обработки и представления информации в конечном виде. Для того, чтобы агрегировать данные, был разработан следующий алгоритм:

1. При старте приложения, загружаются основные настройки.
2. В процессе инициализации экранов приложения, создаются шаблоны ленты мероприятий.
3. Экраны запрашивают данные, чтобы заполнить шаблоны ленты мероприятий.
4. В зависимости от запроса, собираются различные данные. В процессе учитываются загруженные настройки.
5. Завершение сбора данных вызывает срабатывания слушателей, которые инициируют анимации и процесс заполнения шаблона ленты данными.

На рисунке 3.4 представлена наглядная схема алгоритма.

Рисунок 3.4 – Наглядная схема алгоритма

Рассмотрим более подробно каждый этап алгоритма.

3.5.1 Старт приложения, загрузка основных настроек

Исходя из выводов по результатам анализа архитектуры приложения, был разработан класс DataCenter, показанный на рисунке 3.5. Исходный код класса приведен в приложении В.

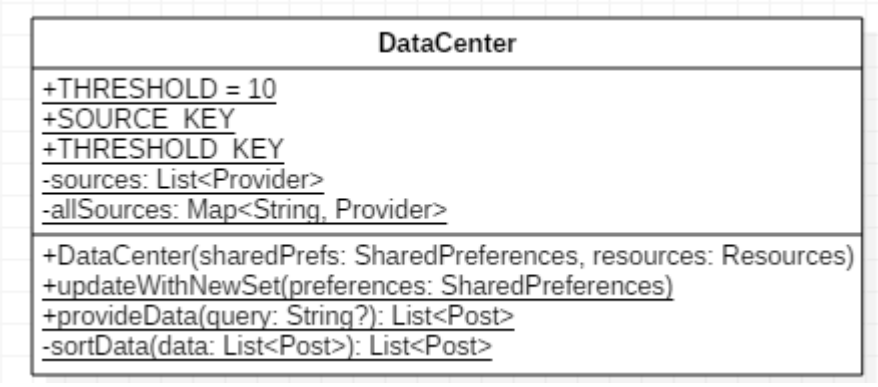

Рисунок 3.5 – Диаграмма класса DataCenter

Он инициализируется при старте приложения и является ответственным за предоставление доступа к различным настройкам, за обработку запроса получения данных.

- поле THRESHOLD показывает, с каким количеством мероприятий из каждого источника необходимо вести работу;
- поле allSources содержит в себе соответствие между именем источника данных в Shared Preferences и объектом, который работает с этим источником;
- поле sources хранит в себе список выбранных источников данных;
- функция updateWithNewSet обновляет список sources в зависимости от выбранных пользователем источников данных;
- функция provideData отвечает за сбор информации о мероприятиях из источников данных. Подробнее о том, как работает данная функция, будет рассказано в следующих пунктах.

Работа с настройками приложения ведется в отдельных окнах, которые показано на рисунках 3.6 и 3.7.

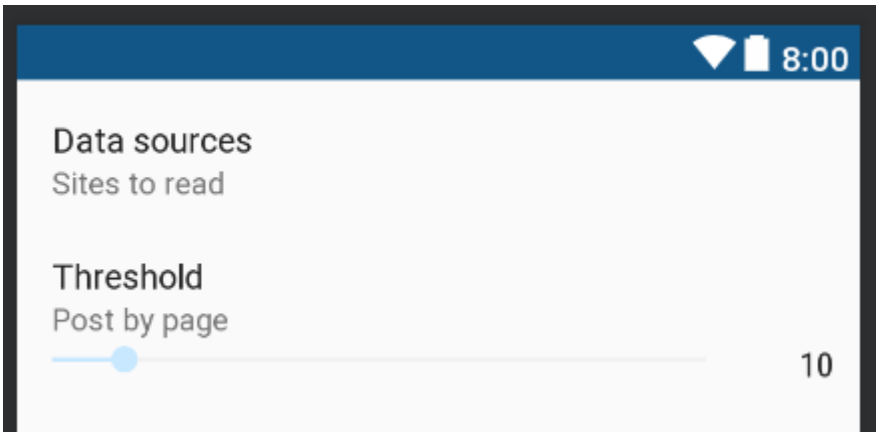

Рисунок 3.6 – Окно настроек

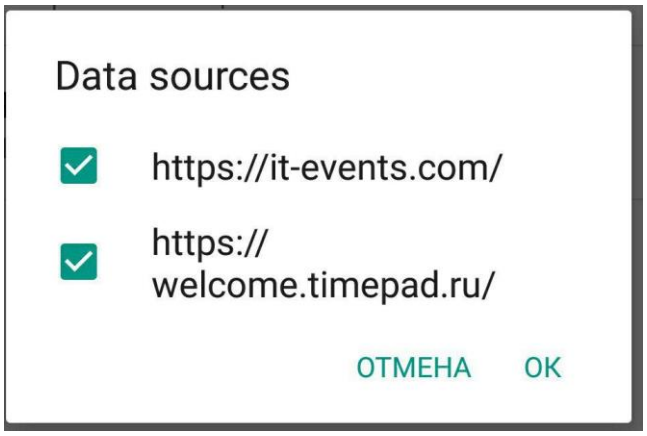

Рисунок 3.7 – Окно выбора источника данных

После изменения каких-либо настроек, происходит анализ изменений, меняются соответствующие поля в классе DataCenter.

3.5.2 Создание шаблона ленты мероприятий

Задача создания некоторого прокручиваемого списка элементов и его отображением в ОС Android легко решается благодаря компоненту RecyclerView [23].

В модели RecyclerView несколько различных компонентов работают вместе для того, чтобы отобразить данные на экран. Общий контейнер для пользовательского интерфейса – это объект RecyclerView, который добавляется в макет экрана. Он сам заполняется шаблонами отображения данных, которые предоставляет ему менеджер макета, который указывается в коде, например LinearyLayoutManager или GridLayoutManager, которые соответственно располагают объекты в виде линейки или таблицы.

Сами же макеты для отображения данных представлены объектами view holder. Эти объекты являются сущностями класса, который определяется с помощью расширения RecyclerView.ViewHolder класса. Каждый объект view holder отвечает за отображение одного элемента в списке. RecyclerView создает только столько объектов view holder, сколько требуется для отображения данных, которые умещаются на экран, плюс еще пару для более плавного отображения. Когда пользователь прокручивает список, RecyclerView переиспользует view holder, которые не видны на экране, и перезаполняет их теми данными, которые прокручиваются в текущий момент.

Объекты view holder управляются адаптером, который создается с помощью расширения класса RecyclerView.Adapter. Он создает объекты view holder при необходимости. Также адаптер соединяет эти объекты с данными, которые они должны отображать. Это делается путем назначения объекту view holder позиции и вызовом метода onBindViewHolder, который использует номер позиции чтобы определить, какие данные нужно передать объекту.

Модель RecyclerView выполняет большую оптимизационную работу:

- подготовка следующих ближайших элементов, которых нет на экране, к отображению;
- создание объектов view holder только по требованию и переиспользование тех объектов, которых не видно на экране;
- динамическое связывание данных с элементами, которые ответственны за их отображение.

Чтобы реализовать RecyclerView, должны быть созданы следующие компоненты:

- RecyclerView, который должен быть добавлен в макет экрана;
- макет для элемента списка;
- адаптер, который содержит данные и связывает их со списком.

Компонент RecyclerView был добавлен во все макеты экранов, которые занимаются отображением списков мероприятий. Как было сказано в пункте «3.3 Создание абстрактного класса для экранов» все экраны наследуются от абстрактного класса BaseFragment, который регистрирует данный компонент в процессе инициализации.

Далее был разработан макет элемента списка с учетом унифицированной модели данных. Результат представлен на рисунке 3.8. Исходный код макета представлен в приложении Г.

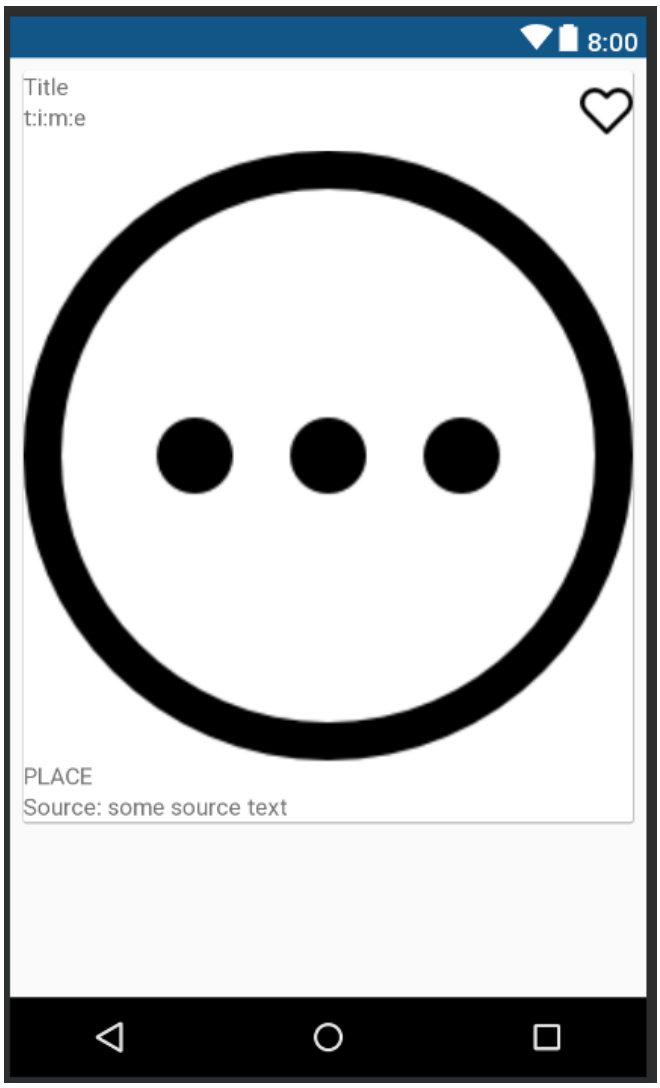

Рисунок 3.8 – Макет элемента списка

После были созданы адаптер PostAdapter, который определяет объекты view holder и данные, которые они должны отобразить, и PostHolder, который связывает данные и элементы UI. Подробнее об этом процессе будет рассказано в следующих пунктах.

В результате работы получилась следующая лента событий, показанная на рисунке 3.9.

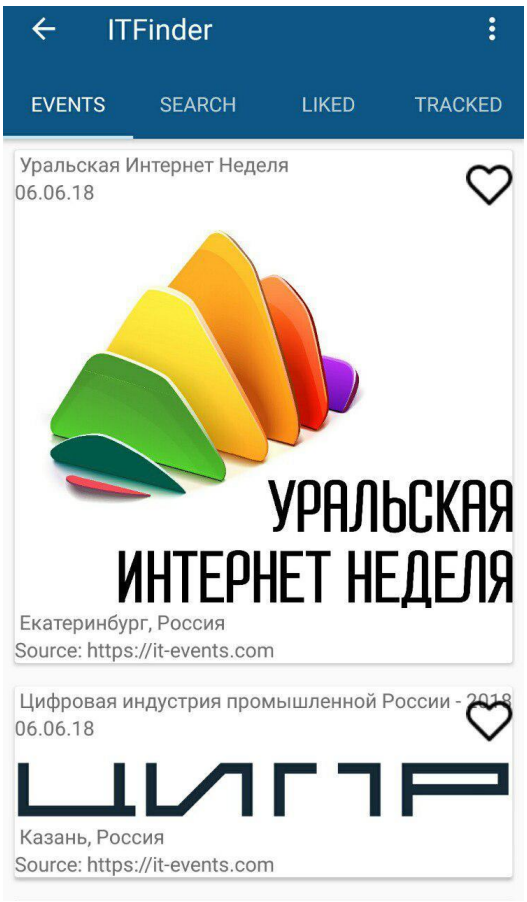

Рисунок 3.9 – Лента событий

3.5.3 Запрос и сбор данных

В ОС Android существует абстрактный класс, который дает возможность выполнять тяжелые по времени выполнения задачи в фоновом режиме, не прерывая тем самым поток пользовательского интерфейса – AsyncTask.

Приложение на Android изначально работает по однопоточной модели - запускается и работает на одном потоке. Из-за этого задачи, выполнение которых занимает больше времени, создают «зависания» интерфейса приложения, делая его нечувствительным к различным событиям. Использование асинхронных задач, реализованных с помощью класса AsyncTask, помогает избежать данную проблему. С помощью этого класса можно выполнять трудоемкие задачи в фоновом режиме в выделенном потоке и передавать результаты своей работы обратно в поток пользовательского интерфейса.

Основные методы, используемые в классе AsyncTask, определены ниже:

- doInBackground – этот метод содержит код, который необходимо выполнить в фоновом режиме. Существует возможность отправлять результаты выполнения в поток пользовательского интерфейса методом publishProgress. Чтобы уведомить, что фоновая обработка завершена, используется оператор return;
- onPreExecute – этот метод содержит код, который выполняется до начала выполнения функции фоновой обработки;
- onPostExecute – этот метод вызывается после завершения обработки метода doInBackground. Результат doInBackground будет передан этому методу;
- onProgressUpdate – этот метод получает обновления прогресса из метода doInBackground, который отправляется с помощью метода publishProgress. Метод может использовать эту информацию для обновления элементов пользовательского интерфейса.

Асинхронные задачи в разработанном приложении используются для выполнения запроса по сбору и предварительной обработки списков мероприятий. Это обусловлено тем, что скорость и отзывчивость различных источников данных и их обработчиков может быть разной и непостоянной.

В классе BaseFragment содержится метод updateItems, который создает новую асинхронную задачу AsyncTaskFetch, которая в свою очередь наследует AsyncTask. Для этой задачи устанавливается слушатель DataPreparedListener, о котором будет рассказано в следующем пункте, после чего она начинает свое выполнение – в фоновом режиме вызывается метод DataCenter.provideData. Как было сказано в пункте «3.5.1 Старт приложения, загрузка основных настроек», эта функция отвечает за сбор информации о мероприятиях с источников данных.

Сначала создается пустой список для сохранения результатов, далее анализируется строка query.

1. Если строка является null, то это значит, что требуется получить N новых мероприятий. Для всех источников данных в sources вызывается метод fetchPosts. Подробнее об этом методе будет рассказано ниже.
2. Если строка совпадает со специальной строкой LIKED_QUERY, то это значит, что запрос пришел из экрана избранных событий. В этом случае, выполняется запрос к базе данных на получение всех сохраненных избранных мероприятий.
3. Если строка пустая, то ничего не происходит.
4. Все остальные случаи означают, что необходимо найти N свежих мероприятий по данной строке. Для всех источников данных в sources вызывается метод searchPosts. Подробнее об этом методе будет рассказано ниже.

Все результаты выполнения сохраняются в список, который был создан в начале работы метода. Он сортируется по убыванию, начиная от самых свежих мероприятий, после чего метод возвращает данный список. После этого класс AsyncTaskFetch вызывает метод onPostExecute, который инициализирует процесс заполнения ленты мероприятий данными.

Стоит сказать, что класс DataCenter оперирует источниками данных посредством интерфейса Provider, представленном на рисунке 3.10.

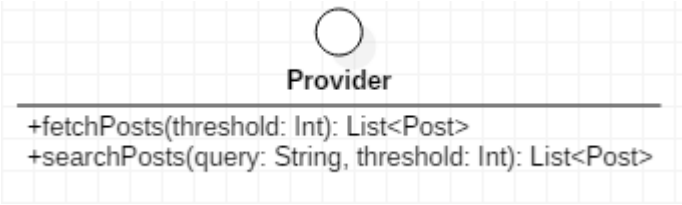

Рисунок 3.10 – Диаграмма интерфейса Provider

Он содержит два метода:

- fetchPosts - принимает переменную типа Int, которая показывает, какое максимальное количество мероприятий должно быть обработано. Возвращает список самых свежих мероприятий из источника данных.

- searchPosts - принимает два параметра – строку, по которой будет вестись поиск (имя, место или источник данных) и переменную типа Int, которая показывает, какое максимальное количество мероприятий должно быть обработано. Возвращает список всех найденных мероприятий, начиная от самых свежих.

3.5.4 Заполнение ленты мероприятий

В предыдущем пункте было сказано, что в процессе создания асинхронной задачи, для нее устанавливается слушатель DataPreparedListener. Это интерфейс, который содержит лишь один метод retrieveNewData(data: List<Post>), принимающий список событий. Метод onPostExecute вызывает этот метод, передавая в него список данных, полученный в результате выполнения метода doInBackground. Это вызывает срабатывания слушателя того экрана, который инициировал выполнение асинхронной задачи.

Слушатель вызывает метод updateAdapter абстрактного класса BaseFragment. Этот метод создает новый PostAdapter с новыми данными и прикрепляет его к RecyclerView. Это инициирует процесс связывания данных с компонентами интерфейса. В процессе связывания:

- макет мероприятия заполняется такими данными, как название, время, источник данных, место;
- проверяется, является ли мероприятие добавленным в избранное. В зависимости от результата кнопка «Добавить в избранное» в виде сердца подсвечивается красным или остается без цвета;
- на кнопку «Добавить в избранное» добавляется слушатель, который удаляет мероприятие из избранного или добавляет в избранное посредством запросов к базе данных;
- производится асинхронная загрузка превью мероприятия, после чего оно отображается в макете;

- на изображение добавляется слушатель, который по нажатию на изображение открывает исходную страницу мероприятия в браузере.

Для решения проблемы асинхронной загрузки, подготовки и отображения превью в процессе работы над приложением было испытано два способа:

- использование механизмов Looper, Handler системы ОС Android, утилитных классов;
- использование библиотеки Picasso для ОС Android.

Первый метод оказался слишком громоздким, сложно реализуемым, требующим большого количества шаблонного кода. Также пришлось самостоятельно реализовывать кэширование изображений, обработку изображения, изменение размеров и так далее. Результат оказался неудовлетворительным, так как требовал больших аппаратных затрат.

Второй метод оказался простым и компактным. Библиотека Picasso по умолчанию поддерживает загрузку изображений из указанного URL-адреса, использование шаблонных изображений в случае ошибок, кэширование, определение нужных размеров изображения и его подготовку, отображение полученных результатов в нужном шаблоне и формате.

В итоге, вместо трех классов и достаточно большого количества кода в разных местах, был реализован легко читаемый метод loadImageInto(post: Post), который принимал данные о мероприятии и асинхронно загружал, подготавливал и отображал превью. Код функции представлен на рисунке 3.11.

```
private fun loadImageInto(post: Post) {
Picasso
        .get()
        .load(post.imageUrl)
        .error(R.drawable.error_256)
        .placeholder(R.drawable.placeholder_256)
        .into(image)
}
```

Рисунок 3.11 – Функция loadImageInto(post: Post)

3.6 Использование TimePad API

Как было описано ранее, класс DataCenter запрашивает данные у всех источников, указанных в списке sources, которые реализуют интерфейс Provider. Для работы с TimePad API был создан класс TimePadProvider, показанный на рисунке 3.12.

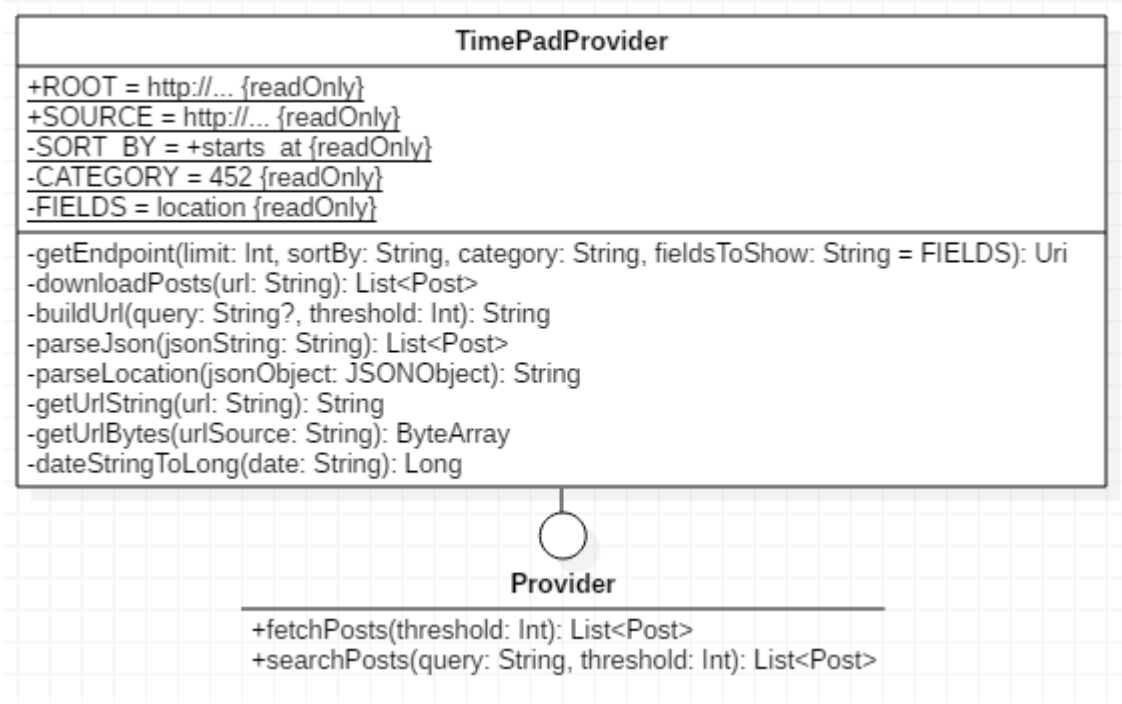

Рисунок 3.12 – Диаграмма класса TimePadProvider

Методы fetchPosts и searchPosts последовательно вызывают методы buildUrl и downloadPosts, после чего возвращают полученные результаты.

Метод buildUrl подготавливает запрос к API: настраивает лимит мероприятий, которые необходимо получить, настраивает параметры таким образом, чтобы в ответе на запрос события располагались от свежих к старым, анализирует строку query: она не является null, то к запросу также добавляется параметр keywords, который говорит о том, что нужно найти только те события, которые содержат в названии или описании события строку query [24].

Алгоритм работы метода downloadPosts:

- вызов метода getUrlString, который открывает HTTP-соединение по указанному адресу, считывает данные и возвращает их в формате JSON.
- вызов метода parseJSON. Этот метод анализирует объект JSON, собирая из него информацию о времени мероприятия, названии, ссылке на источник и так далее. В процессе вызываются вспомогательные для анализа методы. В результате возвращается список мероприятий, полученных с использованием TimePad API.

Таким образом, TimePadProvider собирает самую актуальную информацию для поступивших от DataCenter запросов.

Для создания данного класса, потребовалось прочитать документацию к API, поработать с различными запросами, проанализировать JSON ответы. Эти задачи являются не слишком трудоемкими, большую часть работы берет на себя API, что, несомненно, является плюсом.

3.7 Использование JSOUP

Для анализа сайта IT-Events с использованием библиотеки JSOUP, был разработан класс ItEventsComProvider, который реализует интерфейс Provider. Он показан на рисунке 3.13.

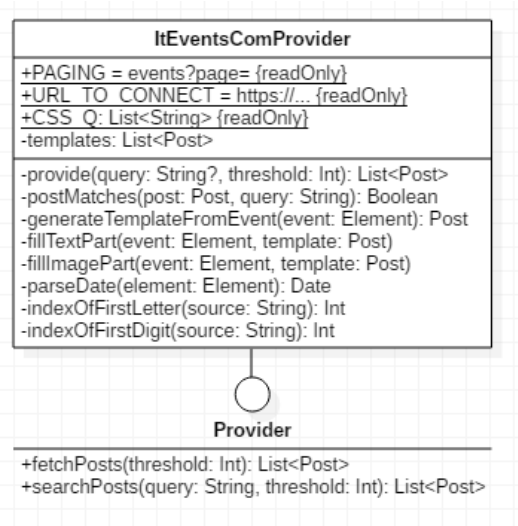

Рисунок 3.13 – Диаграмма класса ItEventsComProvider

Методы fetchPosts и searchPosts вызывают метод provide и возвращают результаты, полученные этим методом.

Алгоритм работы метода provide.

1. Подготовка начальных данных – задание первой страницы для анализа, пустого списка мероприятий.
2. Цикл while:
   - построение DOM-дерева указанного сайта с помощью метода Jsoup.connect(URL_TO_CONNECT + PAGING + (++page)).get(), где URL_TO_CONNECT – адрес сайта, PAGING – параметр, указывающий какую страницу открыть;
   - нахождение с помощью последовательных вызовов метода select(String cssQuery) на построенном дереве элемента, в котором хранятся мероприятия. Этот метод возвращает элементы, которые совпадают указанным CSS запросом. Поэтому был проведен анализ исходной страницы сайта для того, чтобы выделить соответствующие элементы.
   - найденный элемент анализируется, один за другим извлекаются данные о мероприятиях, хранящихся в нем. Это происходит с помощью вызовов метода select, множества вызовов вспомогательных функций, проверок, ручного анализа и формирования данных.
   - после извлечения данных и формирования объекта Post, анализируется строка query. Если она не null, то в случае, если метод postMatches вернет true, мероприятие будет добавлено в результирующий список. Функция postMatches проверяет, содержится ли в названии, месте или источнике строка query.
   - цикл завершает работу, если на новой странице нет элементов с мероприятиями для анализа, если набралось достаточное количество мероприятий или если произошла какая-либо исключительная ситуация.

3. Функция возвращает список мероприятий, собранный в цикле, описанном выше.

Таким образом, ItEventsComProvider с помощью библиотеки JSOUP собирает требуемую для DataCenter информацию для мероприятий.

Для разработки этого класса потребовалось изучить исходный код страницы сайта, проанализировать различные CSS атрибуты, чтобы однозначно найти нужны элементы в процессе выполнения кода, разработать метод анализа всех требуемых страниц, разработать множество вспомогательных методов, провести ручной анализ. Эти задачи являются весьма трудоемкими по сравнению с аналогичными задачами при использовании TimePad API.

3.8 Разработка функции подписки и оповещений

Как было сказано в анализе архитектуры приложения, за подписку и оповещения должен быть ответственным отдельный класс. Также стоит учесть, что приложение может быть выключено. Для решения таких задач Android ОС предоставляет механизм служб [25].

Служба является компонентом приложения, который может выполнять длительные операции в фоновом режиме и не содержит пользовательского интерфейса. Другой компонент приложения может запустить службу, которая продолжит работу в фоновом режиме даже в том случае, когда пользователь перейдет в другое приложение. Кроме того, компонент может привязаться к службе для взаимодействия с ней и даже выполнять межпроцессное взаимодействие.

Был создан класс TrackService, расширяющий службу IntentService, который устанавливает команду Android ОС каждые 15 минут создавать запрос к TrackService. Этот запрос обрабатывается с помощью метода onHandleIntent:

- сначала проверяется, существует ли Интернет-соединение и активно ли оно. В случае, если это не так, ничего не происходит;

- с помощью механизма Shared Preferences выгружаются наименование отслеживаемой темы query и наименование самого свежего мероприятия последнего запроса по этой теме lastResult;
- TrackService обращается к DataCenter с запросом на получение всех мероприятий по запросу query. Если самое свежее мероприятие совпадает с lastResult, то это означает, что с момента последнего обращения ничего нового не произошло. В противном случае, создается оповещение о новом мероприятии по указанной теме, результат lastResult перезаписывается.

Экран отслеживания интересующей темы показан на рисунке 3.14, пример оповещения показан на рисунке 3.15.

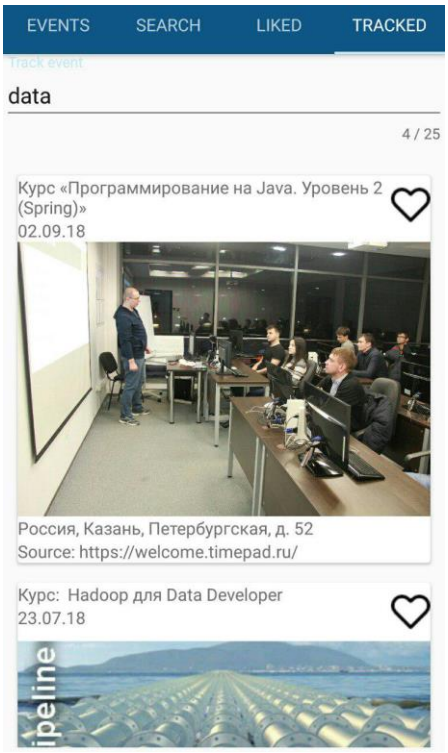

Рисунок 3.14 – Экран подписки на интересующую тему

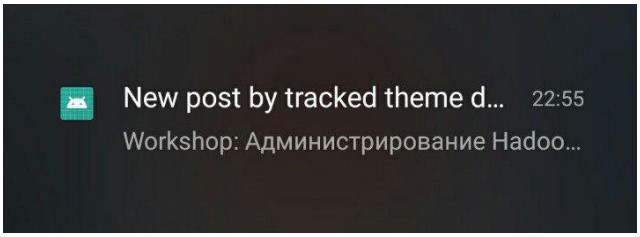

Рис. 3.15 – Оповещение о новом мероприятии

4 Анализ эффективности разработанных решений

В данном разделе сравниваются решения с использованием TimePad API и JSOUP по различным характеристикам. Делается выводы об эффективности использования этих методов.

4.1 Технические характеристики базового оборудования

В качестве базовой платформы для анализа представлен телефон Meizu MX6 со следующими техническими характеристиками [26]:

- процессор: MediaTek Helio X20 MT6797, 10 ядер (4x Cortex-A53, 1.4 ГГц + 4x Cortex-A53 1,9 ГГц + 2x Cortex-A72 2.3 ГГц);
- графический процессор: ARM Mali-T880 MP4, 780 МГц;
- оперативная память: 4 Гб;
- операционная система: FlyMe 6.0 (Android 6.0 Marshmallow);
- аккумулятор: 3060 мА/ч, литий-полимерный.

Интернет-подключение обеспечивает WiFi со скоростью до 54Мбит/сек.

Для обеспечения чистоты тестирования, все остальные приложения и их фоновые процессы отключались перед запуском самого приложения.

Профилирование велось при помощи инструмента Android Profiler. С помощью него были построены графики нагрузки на ЦП, на память и на сеть. Они были построены для количества событий в размере 10, 30 и 50.

4.2 Анализ решений с использованием функции просмотра всех событий

Первая группа тестирований проходила с использованием функции просмотра всех событий. В качестве длительности тестирования было принято время, равное 10 секундам после запуска.

Результаты работы представлены на рисунках 4.1, 4.2, 4.3.

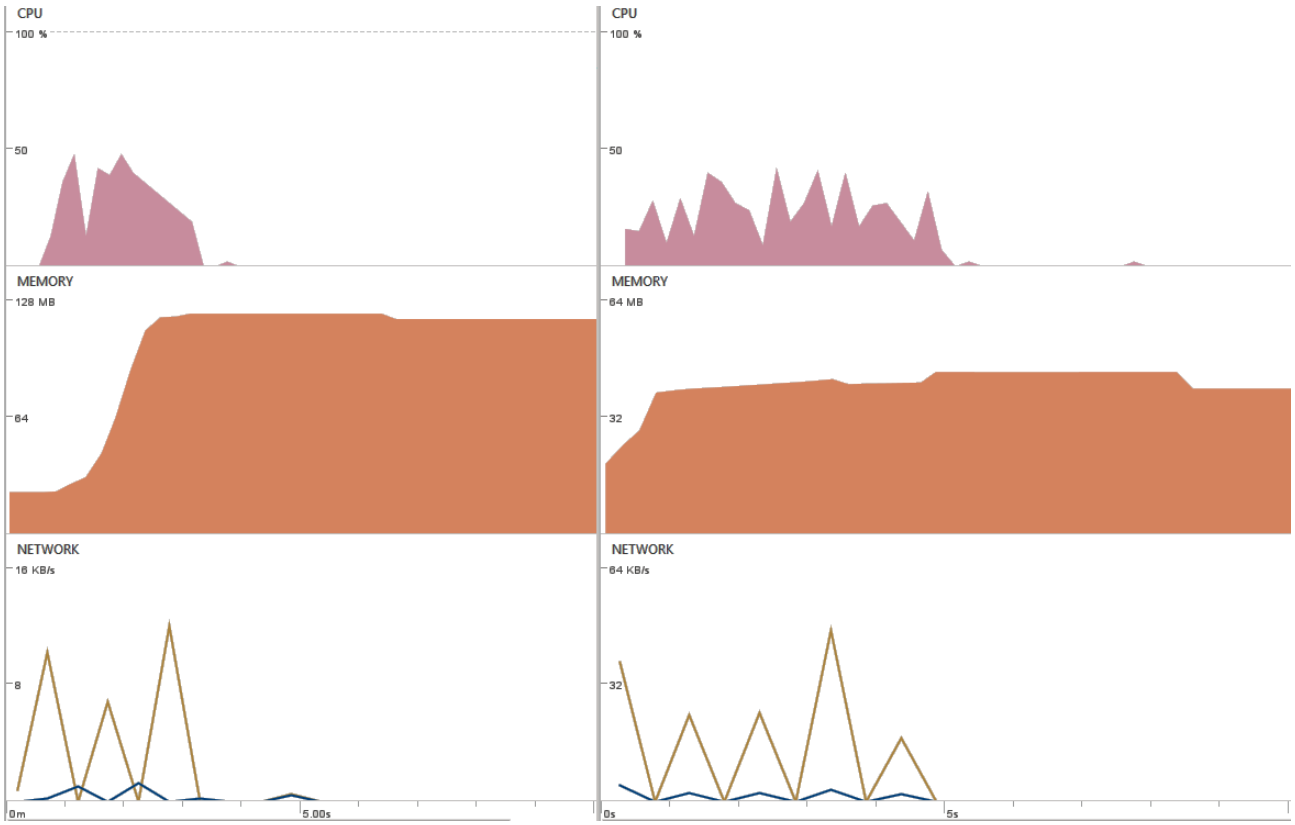

Рисунок 4.1 – Графики нагрузки при 10 элементах

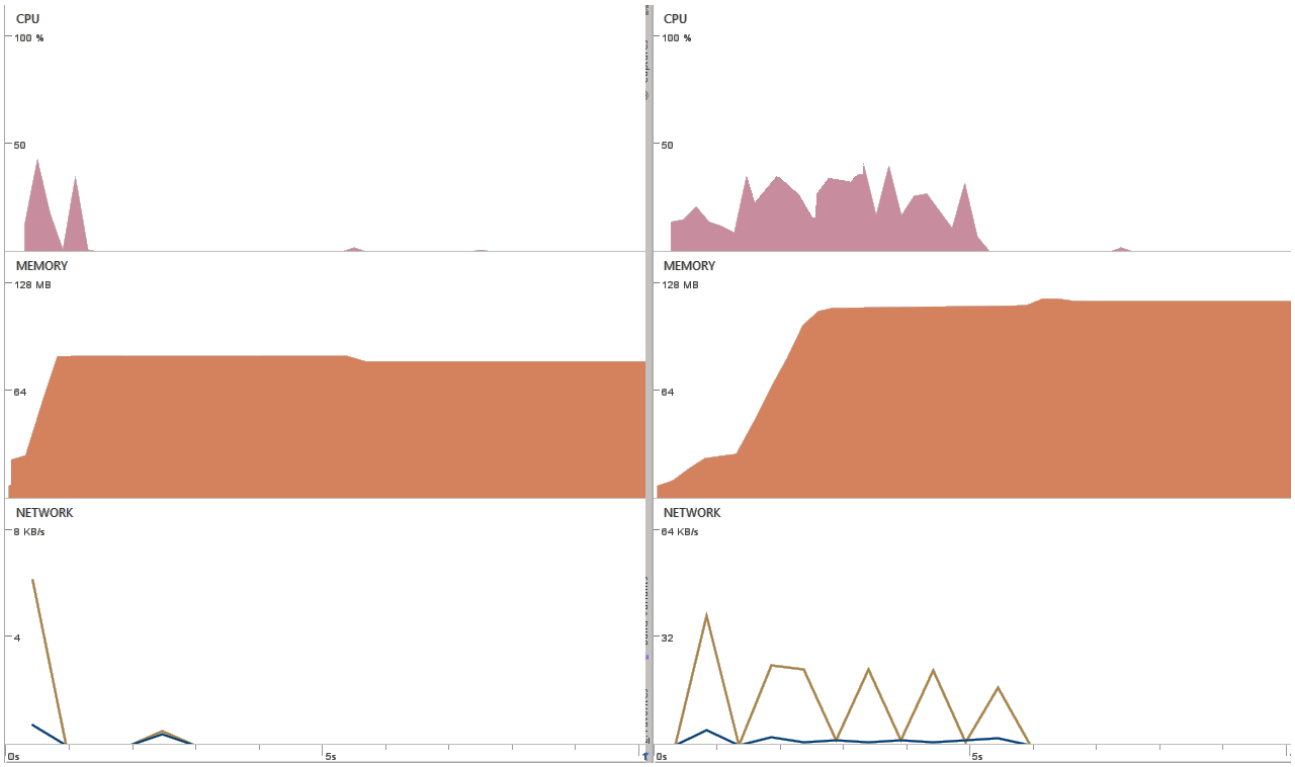

Рисунок 4.2 – Графики нагрузки при 30 элементах

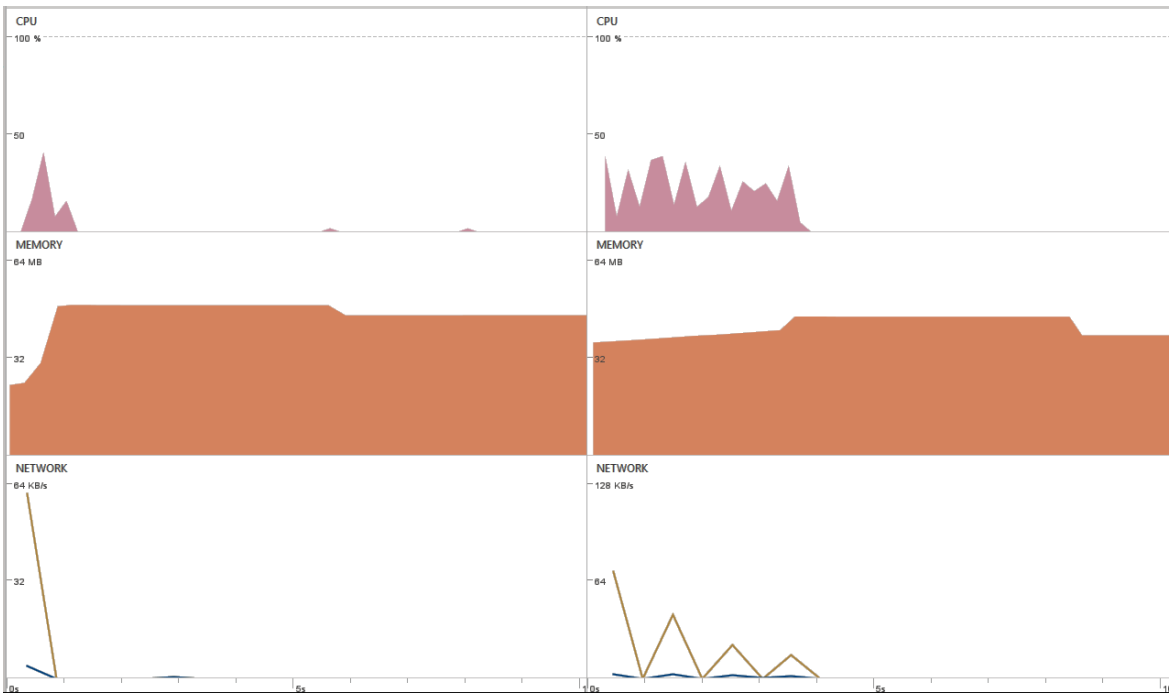

Рисунок 4.3 – Графики нагрузки при 50 элементах

На всех графиках слева изображена нагрузка при использовании TimePad API, справа при использовании JSOUP.

Из графиков видно следующее:

- нагрузка на CPU при использовании TimePad API длится меньше и является менее интенсивной, чем при использовании JSOUP;
- пиковая величина нагрузки на сеть при использовании TimePad API растет с количеством требуемых элементов, однако, длительность всегда остается короткой, отмечается низкая интенсивность. Это связано с тем, что приложение делает лишь один запрос на получение данных, после чего принимает данные в формате JSON;
- пиковая величина нагрузки на сеть при использовании JSOUP начинается с достаточной большой по сравнению с TimePad API. Также она является интенсивной и длительной. Это связано с тем, что необходимо постоянное соединение с сайтом анализа;

- нагрузка на память ведет себя непредсказуемо. Возможно, это связано с тем, что размеры и форматы превью у источника TimePad API и сайтом, который анализирует JSOUP, различаются.

Таким образом, можно сделать вывод, что библиотека JSOUP требует гораздо больше аппаратных ресурсов для работы, чем использование TimePad API.

Однако, по времени работы при просмотре всех событий, JSOUP не отстает: при использовании обоих подходов время до появления первых событий составило примерно 1.2 секунды.

4.3 Анализ решений с использованием функции поиска мероприятий

Вторая группа тестирований проходила с использованием функции поиска мероприятий. В качестве длительности тестирования было принято время, равное 10 секундам после отправки запроса на поиск.

Результаты работы представлены на рисунках 4.4, 4.5, 4.6.

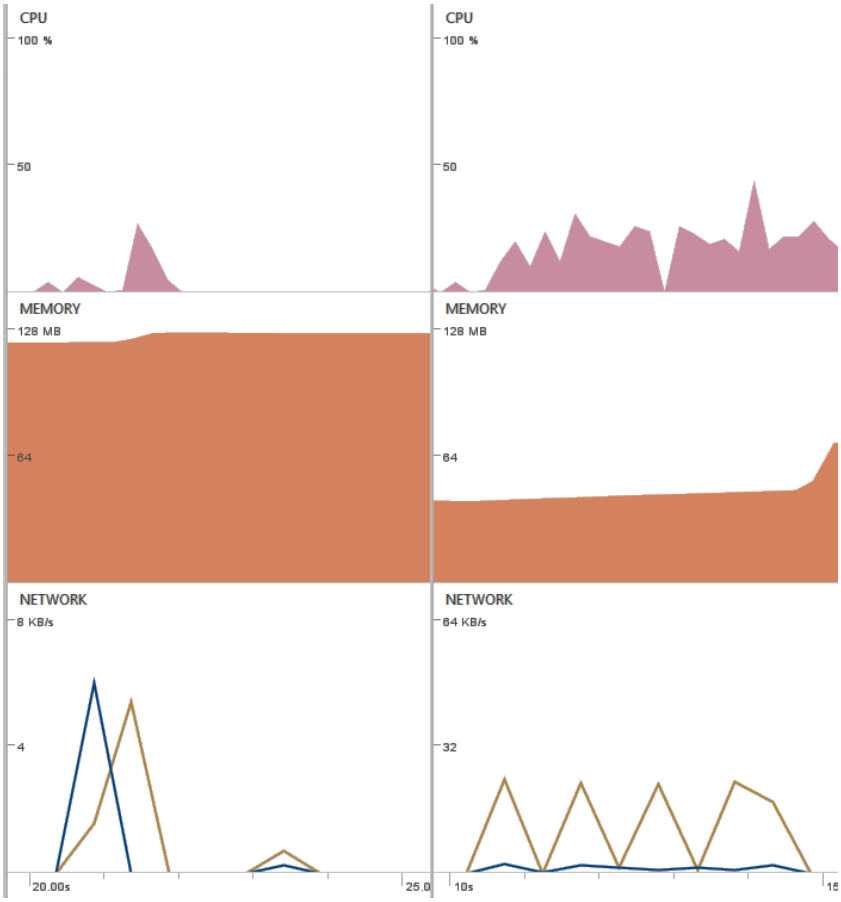

Рисунок 4.4 – Графики нагрузки при 10 элементах

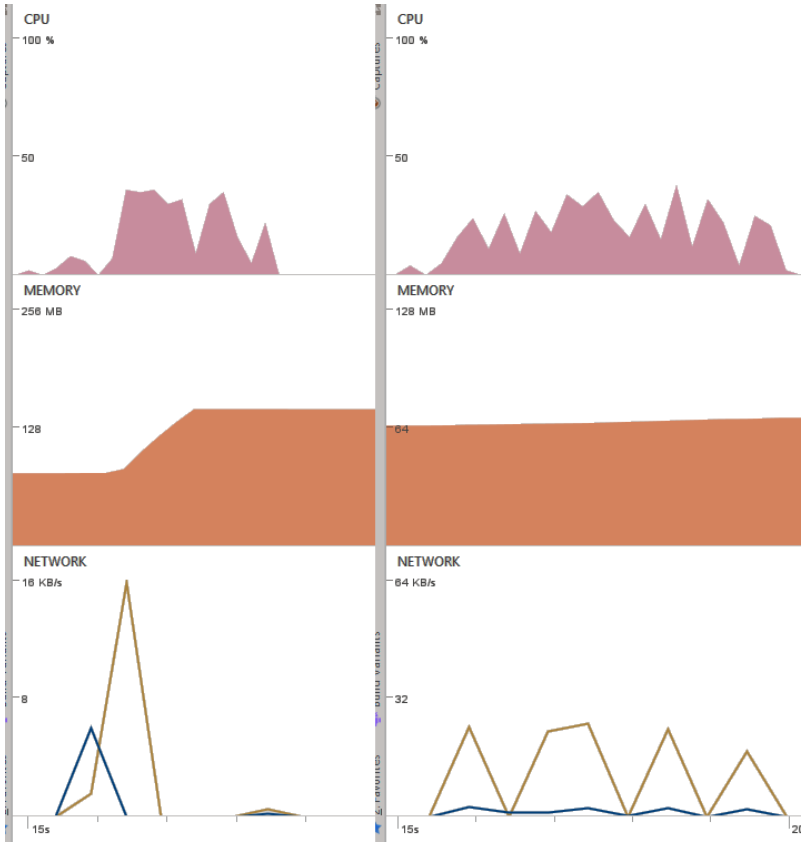

Рисунок 4.5 – Графики нагрузки при 30 элементах

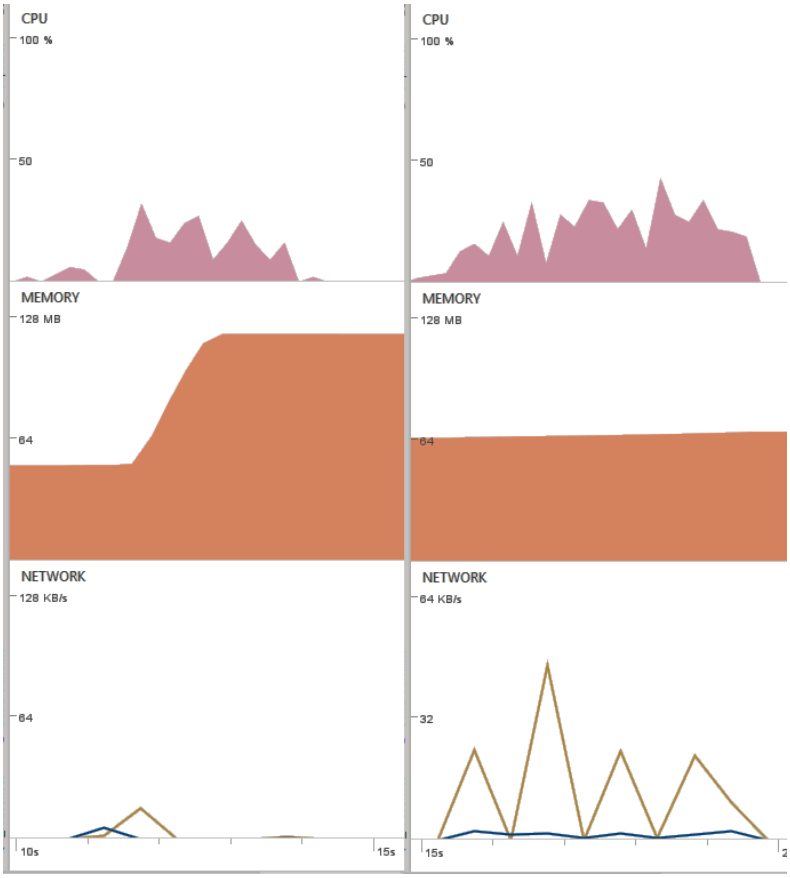

Рисунок 4.6 – Графики нагрузки при 50 элементах

На всех графиках слева изображена нагрузка при использовании TimePad API, справа при использовании JSOUP.

Из графиков видно следующее:

- при возрастании количества элементов при использовании TimePad API, интенсивность нагрузки на ЦП увеличивается и становится близкой к графику интенсивности нагрузки при использовании JSOUP;
- как и в предыдущем случае, нагрузка на сеть гораздо выше и интенсивнее при использовании JSOUP.

Также замерялось среднее время ответа на запрос при использовании разных подходов. Для TimePad API оно составило 0.92 секунды, для JSOUP 1.47 секунд. Это связано с тем, что JSOUP анализирует все страницы сайта до тех пор, пока не наберется нужное количество мероприятий для ответа.

4.4 Выводы

В данном разделе были протестированы два подхода для получения информации из различных источников – с использованием TimePad API и JSOUP. Оба подхода выполняли свою задачу за очень короткий промежуток времени. Однако, подход с использованием JSOUP оказался хуже, так как оказывал большую нагрузку на сеть и ЦП, в то время как подход с использованием TimePad API справлялся с задачей быстрее и требовал меньше аппаратных затрат.

5 Технико-экономическое обоснование

В данном разделе будет проведено технико-экономическое обоснование, рассчитана себестоимость программного продукта с точки зрения вложения в него финансовых ресурсов.

5.1 Концепция

Выпускная квалификационная работа посвящена разработке приложения для поиска IT-мероприятий на базе ОС Android.

Приложение предназначено для распространения через магазин приложений Google Play.

5.2 Краткое техническое описание

Разработанное приложение будет использоваться для просмотра новых мероприятий, поиска, сохранения в избранное, подписки на оповещение о новых событиях.

5.3 Экономическая значимость

Разработанное программное обеспечение планируется поставлять на бесплатной основе через магазин приложений Google Play с возможностью покупки платных сервисов по подписке.

5.4 Расчет полных затрат при разработке ПО

Расчет начинается с расчета затрат на каждом этапе работ. Детализированный план-график выполнения работ представлен в таблице 5.1, затраты по каждому этапу работ представлены в таблице 5.2.

Таблица 5.1 – Детализированный план-график выполнения работ

№	Наименование работы	Длительность работы, час		
		Tmin	Tmax	To
1	Разработка ТЗ	4	8	5,6
2	Анализ ТЗ	1	3	1,8
3	Установка необходимого ПО	1	2	1,4
4	Разработка основной архитектуры приложения	4	12	7,2
5	Ручной разбор источника данных с использованием JSOUP	4	8	5,6
6	Автоматизированное получение данных с использованием API	4	8	5,6
7	Разработка дополнительного функционала (поиск, сохранение в избранное, подписка на обновления)	24	32	27,2
8	Разработка графического интерфейса	16	24	19,2
9	Соединение графического интерфейса и функциональной части	8	16	11,2
10	Добавление возможностей тонкой настройки	4	8	5,6
11	Тестирование приложения и исправление ошибок	12	24	16,8
12	Оформление пояснительной записки	40	60	48
ИТОГО		122	205	155,2

Таблица 5.2 – Затраты по этапам работ

№	Наименование работы	Исполнитель	Трудоемкость То, час	Ставка, руб/час
1	Разработка ТЗ	Разработчик	5,6	178,5
2	Анализ ТЗ	Разработчик	1,8	178,5
3	Установка необходимого ПО	Разработчик	1,4	178,5
4	Разработка основной архитектуры приложения	Разработчик	7,2	178,5
5	Ручной разбор источника данных с использованием JSOP	Разработчик	5,6	178,5
6	Автоматизированное получение данных с использованием API	Разработчик	5,6	178,5
7	Разработка дополнительного функционала (поиск, сохранение в избранное, подписка на обновления)	Разработчик	27,2	178,5
8	Разработка графического интерфейса	Разработчик	19,2	178,5
9	Соединение графического интерфейса и функциональной части	Разработчик	11,2	178,5
10	Добавление возможностей тонкой настройки	Разработчик	5,6	178,5
11	Тестирование приложения и исправление ошибок	Разработчик	16,8	178,5
12	Оформление пояснительной записки	Разработчик	48	178,5

Далее проводится расчет заработной платы заказчика. Результаты приведены в таблице 5.3.

Таблица 5.3 – Заработная плата заказчика

№	Этапы и содержание выполняемых работ	Исполнитель	Трудоемкость То, час	Ставка, руб/час
1	Разработка, анализ и уточнение ТЗ	Руководитель	4	250
2	Поиск и анализ существующих методов получения данных	Руководитель	4	250
3	Анализ предметной области	Руководитель	8	250
4	Рекомендации по разработке функционала	Руководитель	20	250

На основе данных о трудоемкости выполняемых работ и ставке в день соответствующих исполнителей определим расходы на заработную плату исполнителей и отчисления на страховые взносы на обязательное пенсионное, социальное и медицинское страхование.

Расходы на основную заработную плату исполнителей определяются по формуле

$$З_{\text{осн.з/пл}} = \sum_{i=1}^{k} T_i \cdot C_i,$$

где $З_{\text{осн.з/пл}}$ – расходы на основную заработную плату исполнителей (руб.); k – количество исполнителей; T_i – время, затраченное i-м исполнителем на проведение исследования (дни или часы); C_i – ставка i-го исполнителя (руб./день или руб./час).

$$З_{\text{осн.з/пл}} = 155{,}2 * 178{,}5 + 36 * 250 = 27703{,}2 + 9000 = 36703{,}2$$

Расходы на дополнительную заработную плату исполнителей определяются по формуле

$$З_{\text{доп.з/пл}} = З_{\text{осн.з/пл}} \cdot \frac{H_{\text{доп}}}{100},$$

где $З_{доп.з/пл}$ – расходы на дополнительную заработную плату исполнителей (руб.); $З_{осн.з/пл}$ – расходы на основную заработную плату исполнителей (руб.); $H_{доп}$ – норматив дополнительной заработной платы (%). При выполнении расчетов в ВКР норматив дополнительной заработной платы принимаем равным 14%.

$$З_{доп.з/пл} = 36703{,}2 * 0{,}14 = 5138{,}4 \text{ руб.}$$

Отчисления на страховые взносы на обязательное социальное, пенсионное и медицинское страхование с основной и дополнительной заработной платы исполнителей определяются по формуле:

$$З_{соц} = (З_{осн.з/пл} + З_{доп.з/пл}) \cdot \frac{H_{соц}}{100}$$

где $З_{соц}$- отчисления на социальные нужды с заработной платы (руб.); $З_{осн.з/пл}$- расходы на основную заработную плату исполнителей (руб.); $З_{доп.з/пл}$ – расходы на дополнительную заработную плату исполнителей (руб.); $H_{соц}$ – норматив отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование (30% на 14.05.2016).

$$З_{соц} = (36703{,}2 + 5138{,}4) * 0{,}3 = 12552{,}5 \text{ руб.}$$

Работа не подразумевала использование стендов, но выполнение любой работы связано с материальными затратами. На статью материалы относятся расходы на основные и вспомогательные материалы, покупные полуфабрикаты и комплектующие изделия, использованные при выполнении разработки. Стоимость вышеперечисленного определяется из величины их расхода, действующих цен и транспортно-заготовительных расходов. Затраты по статье “материалы” приведены в таблице 5.4.

Таблица 5.4 – Затраты по статье «материалы»

Материалы	Единица измерения	Количество	Цена, руб.	Сумма, руб.
Бумага писчая А4	Пачка	1	299	299
Картридж для принтера	шт.	1	1 900	1 900

Покупные комплектующие изделия:

$$З_n = \sum_{l=1}^{L} N_l Ц_l (1 + \frac{H_{т.з}}{100})$$

где $З_n$ – затраты на покупные комплектующие изделия (руб.); N_l – количество l-тых комплектующих изделий входящих в единицу продукции (шт.); $Ц_l$ – цена приобретения единицы l-го комплектующего (руб./шт.); $Н_{т.з}$ – норма транспортно-заготовительных расходов (%). При выполнении расчетов в ВКР норму транспортно-заготовительных расходов ($Н_{т.з}$) принимаем равной 10%.

$$З_n = (299 + 1900) * (1 + 0.1) = \ 2418{,}9 \text{ руб.}$$

При выполнении ВКР дополнительное спецоборудование и услуги сторонних организаций не требуются. Командировки специалистов, проводящих разработку программного обеспечения, не предусмотрены.

Прочие прямые расходы

Смета по статье “прочие прямые расходы”, включающая затраты, связанные с получением специальной научно-технической литературы, а также платежи за использование средств связи и коммуникации приведены в таблице 5.5.

Таблица 5.5 – Затраты по статье «прочие прямые расходы»

Наименование	Единица измерения	Количество	Цена, руб.	Сумма, руб.
Средства связи и коммуникации				
Доступ в интернет	мес.	4	500	2000
Научно-техническая литература				
Билл Филлипс, К. Стюарт, Кристин Марсикано «Android. Программирование для профессионалов», Санкт-Петербург «Питер», 2017	шт.	1	1089	1089
Итого				3089

Амортизационные отчисления

Для проведения разработки программы используются основные средства, поэтому необходимо учесть и включить в затраты амортизационные отчисления по этим средствам.

Амортизационные отчисления по основному средству I за год определяются как:

$$A_i = Ц_{п.н.i.} \cdot \frac{H_{ai}}{100},$$

гдеA_i- амортизационные отчисления за год по i-му основному средству (руб.); $Ц_{п.н.i.}$ – первоначальная стоимость i-го основного средства (руб.); H_{ai} – годовая норма амортизации i-го основного средства (%). Величина амортизационных отчислений по i-му основному средству, используемому студентом при работе над ВКР, определяется по формуле:

$$A_{i\text{ВКР}} = A_i \cdot \frac{T_{i\text{ВКР}}}{12},$$

где $A_{i\text{ВКР}}$ - амортизационные отчисления по i-му основному средству, используемому студентом в работе над ВКР (руб.); A_i – амортизационные отчисления за год по i-му основному средству (руб.); $T_{i\text{ВКР}}$ – время, в течение которого студент использует i-ое основное средство (мес.). Амортизационные отчисления приведены в таблице 5.6.

Таблица 5.6 – Амортизационные отчисления

№	Средство	Стоимость	Норма амортизации	Отчисления, руб.
1	Персональный ноутбук	45000	50,00%	7500
2	Meizu M6	16000	33,33%	3111,11
Итого				10611,11 руб.

Накладные расходы

В эту статью включаются расходы на управление и хозяйственное обслуживание проектной команды.

$$C_{HP} = \frac{C_{ЗО} \cdot H_{HP}}{100},$$

Где $H_{НР}$ – норматив накладных расходов, составляет 34%.

$$C_{HP} = \frac{56\,300 \cdot 20}{100} = 11260$$

Смета затрат на ВКР.

Смета затрат на ВКР, основанная затратах, по статьям расходов, приведенных выше, представлена в таблице 5.7.

Таблица 5.7 – Смета затрат на ВКРа

№ п/п	Наименование статьи	Сумма, руб.
1.	Расходы на оплату труда	41841,6
2.	Отчисления на социальные нужды	5138,4
3.	Материалы	2418,9
4.	Затраты по работам, выполняемым сторонними организациями	0
5.	Прочие прямые расходы	3089
6.	Амортизационные отчисления	10611,11
7.	Накладные расходы	88062,72
8.	Спецоборудование	0
ИТОГО затрат: 151161,73 руб.		

5.5 Выводы по разделу

В данном разделе была рассчитана себестоимость программного продукта с точки зрения вложения в него финансовых ресурсов.

Были посчитаны величины основной и дополнительной заработной платы, социальных отчислений участников, принимающих участие в проекте, материальные затраты на оборудование рабочих мест.

Исходя из расчета, затраты на разработку данного проекта составляют 151161,73 руб. Эта сумма может послужить базой для дальнейшего ценообразования при коммерческом распространении продукта.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы были рассмотрены современные мобильные операционные системы, способы сбора информации из различных источников. Были установлены критерии для приложения для поиска IT-мероприятий, рассмотрены существующие аналоги, на основе чего был сделан вывод, что в текущее время на рынке нет приложения, которое бы удовлетворяло всем поставленным критериям в полном объеме.

В работе была предложена архитектура и алгоритм для решения поставленной задачи. Были исследованы различные способы получения информации из различных источников, разработаны решения для проблемы извлечения и поиска данных. В работе был проведен анализ эффективности полученных решений.

В результате было создано мобильное приложение для поиска IT-мероприятий в сети Интернет на базе ОС Android, которое позволяет пользователю централизованно, быстро и в удобном виде получать нужную ему информацию.

Дальнейшее развитие данной работы заключается в добавлении новых источников данных. Улучшение разработанных алгоритмов для поиска и извлечения данных. Более глубокая работа над пользовательским интерфейсом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фирма HIS Markit. Исследование количества пользователей смартфонов и прогнозирование роста. – http://news.ihsmarkit.com/press-release/technology/more-six-billion-smartphones-2020-ihs-markit-says
2. Проект Statscounter. Статистика использования различных ОС по всему миру. – http://gs.statcounter.com/os-market-share/mobile/worldwide
3. Официальный сайт по Android разработке. Описание системы Android. – http://developer.android.com/guide/basics/what-is-android.html
4. Официальный блог разработчиков Android. Анонсирование Интернет-магазина Play Market и условий публикации приложений. – http://android-developers.blogspot.com/2008/10/android-market-now-available-for-users.html
5. Официальный сайт по Android разработке. Описание NDK. – http://developer.android.com/sdk/ndk/index.html
6. Фирма Apple. Описание программы для iOS разработчиков. – https://developer.apple.com/programs/ios/
7. Магазин Google Play. Приложение IT Events Belarus. – https://play.google.com/store/apps/details?id=com.masha.android.eventsclient
8. Магазин Google Play. Приложение IT Мероприятия. – https://play.google.com/store/apps/details?id=ru.srcblog.javavirys.doumeetups
9. Официальный сайт по Android разработке. Статистика использования различных версий Android ОС. – https://developer.android.com/about/dashboards/
10. Документация языка Kotlin. Возможности языка Kotlin. – https://kotlinlang.org/docs/reference/null-safety.html
11. Компания GitHub. Страница исходного кода языка Kotlin. – https://github.com/JetBrains/kotlin
12. Официальный сайт по Android разработке. Описание поддержки языка Kotlin. – https://developer.android.com/kotlin/

13. Сайт InfoQ. Информации о завершении поддержки инструмента Android Eclipse Tools. – https://www.infoq.com/news/2015/06/google-android-eclipse
14. Официальный сайт по Android разработке. Информация об инструменте Android Profiler. – https://developer.android.com/studio/profile/android-profiler
15. Документация по использованию TimePad API для разработчиков. – http://dev.timepad.ru/
16. Сайт компании TimePad. – https://welcome.timepad.ru/company
17. Сравнение HTML анализаторов. – https://en.wikipedia.org/wiki/Comparison_of_HTML_parsers
18. Официальный сайт библиотеки JSOUP. – https://jsoup.org/
19. Сайт IT-Events. – https://it-events.com/
20. Официальный сайт по Android разработке. Информация о типах памяти в Android. – https://developer.android.com/guide/topics/data/data-storage
21. Официальный сайт по Android разработке. Рекомендации по реализации настроек приложения. – https://developer.android.com/guide/topics/ui/settings
22. Документация по использованию библиотеки dbFlow. – https://agrosner.gitbooks.io/dbflow/content/
23. Официальный сайт по Android разработке. Информация о компоненте RecyclerView. – https://developer.android.com/guide/topics/ui/layout/recyclerview
24. Документация по использованию TimePad API для разработчиков. Описание параметров запроса. – http://dev.timepad.ru/api/get-v1-events/
25. Официальный сайт по Android разработке. Описание механизма служб. – https://developer.android.com/guide/components/services
26. Сайт DeviceSpecifications. Технические характеристики телефона Meizu MX6. – https://www.devicespecifications.com/ru/model/0ebd3d48

ПРИЛОЖЕНИЕ А

Исходный код стандартной модели данных

Ниже представлен код, описывающий стандартную модель данных. Аннотации над полями необходимы для работы с ORM базой данных.

```
/**
 * Сущность представляет собой одно IT-мероприятие.
 */
@Table(name = "posts", database = AppDatabase::class)
data class Post(
        @PrimaryKey(autoincrement = true)
        var id: Int = 0,
        @Column
        var title: String = "",
        @Column
        var time: Long = 0,
        @Column
        var source: String = "Inner source",
        @Column
        var imageUrl: String = "",
        var isLiked: Boolean = false,
        @Column
        var place: String = "",
        @Column
        var href: String = "") {

    override fun toString(): String {
        return "Post(title='$title', time=$time, source='$source',
            imageUrl='$imageUrl', isLiked=$isLiked, place='$place',
                  href='$href')"
    }
```

ПРИЛОЖЕНИЕ Б

Исходный код класса BaseFragment

Ниже представлен код абстрактного класса BaseFragment, который определяет базовую модель поведения экранов.

```
/**
 * Класс представляет собой базовое поведение экранов.
 */
abstract class BaseFragment : Fragment() {

    /**
     * Идентификатор макета.
     */
    abstract fun provideLayout(): Int
    /**
     * Идентификатор RecyclerView.
     */
    abstract fun provideRecyclerTag(): Int
    /**
     * Идентификатор сообщения об отсутствии данных.
     */
    abstract fun provideEmptyTextTag(): Int

    /**
     * Список мероприятий.
     */
    var data: List<Post> = ArrayList()
    lateinit var recycle: RecyclerView
    lateinit var emptyText: TextView
    private lateinit var asyncTask: AsyncTaskFetch


    /**
     * Создание объекта базового экрана.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        retainInstance = true
        // setup()
    }
```

```
/**
 * Создание макета базового экрана,
 * присвоение сущностей,
 * настройка ленты мероприятий.
 */
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
  savedInstanceState: Bundle?): View? {
    val rootView = inflater.inflate(provideLayout(), null)
    emptyText = rootView.findViewById(provideEmptyTextTag())
    recycle = rootView.findViewById(provideRecyclerTag())
    val layoutManager = LinearLayoutManager(this.activity)
    val scrollPosition = (recycle.layoutManager as?
        LinearLayoutManager)?.findFirstCompletelyVisibleItemPosition() ?: 0

    recycle.layoutManager = layoutManager
    recycle.smoothScrollToPosition(scrollPosition)
    updateAdapter()

    return rootView
}

/**
 * Обновление источников данных.
 */
fun updateAdapter() {
    if (isAdded) {
        val adapter = PostAdapter(data, activity)
        recycle.adapter = adapter
        setupVisibilityAfterFetching()
    }
}

/**
 * Управление анимацией.
 */
private fun setupVisibilityAfterFetching() {
    if (data.isEmpty()) {
        recycle.visibility = View.GONE
        emptyText.visibility = View.VISIBLE
    } else {
        recycle.visibility = View.VISIBLE
        emptyText.visibility = View.GONE
    }
}

/**
 * Запрос на получение новых данных.
 */
fun updateItems(query: String?, progressBar: ImageView) {
    setupVisibilityWhileFetching(progressBar)
    asyncTask = AsyncTaskFetch(query)
    asyncTask.setListener(object : DataPreparedListener {
        override fun retrieveNewData(data: List<Post>) {
            this@BaseFragment.data = data
            updateAdapter()
            progressBar.visibility = View.GONE
```

```
            }
        })
        asyncTask.execute()
    }

    /**
     * Управление анимацией.
     */
    private fun setupVisibilityWhileFetching(progressBar: ImageView) {
        progressBar.visibility = View.VISIBLE
        emptyText.visibility = View.GONE
        recycle.visibility = View.GONE
    }

    /**
     * Удаление слушателя.
     */
    fun removeAsyncListener() {
        asyncTask.setListener(null)
    }
}
```

ПРИЛОЖЕНИЕ В

Исходный код класса DataCenter

Ниже представлен код класса BaseFragment, который является ответственным за сбор, обработку и предоставление информации из зарегистрированных источников данных.

```
/**
 * Этот класс управляет сбором и передачей данных.
 */
class DataCenter(sharedPrefs: SharedPreferences, resources: Resources) {

    companion object {
        var THRESHOLD = 10 // количество обрабатываемых мероприятий
        lateinit var SOURCE_KEY: String // обозначение ключа источников данных
        lateinit var THRESHOLD_KEY: String // обозначение ключа количества
            обрабатываемых мероприятий
        private val sources = ArrayList<Provider>() // используемые источники
            данных
        private lateinit var allSources: Map<String, Provider> // все
            существующие источники данных

        /**
         * Обновление используемых источников данных.
         */
        fun updateWithNewSet(preferences: SharedPreferences) {
            sources.clear()
            val selections = preferences.getStringSet(SOURCE_KEY, null)
            for (name in selections) {
                sources.add(allSources[name]!!)
            }
        }

        /**
         * Сбор информации о мероприятиях из источноков.
         */
        fun provideData(query: String?): List<Post> {
            val data = ArrayList<Post>()

            when (query) {
                null -> {
                    for (provider in sources) {
                        data.addAll(provider.fetchPosts(THRESHOLD))
                    }
                }

                LikedFragment.LIKED_QUERY -> {
                    data.addAll(PostRepository.getAll())
                }

                "" -> {
                    val pass = Unit
                }

                else -> {
```

```
                    for (provider in sources) {
                        data.addAll(provider.searchPosts(query, THRESHOLD))
                    }
                }
            }

            return sortData(data)
        }

        private fun sortData(data: List<Post>): List<Post> =
            data.sortedByDescending { post -> post.time }
    }

    /**
     * Инициализация значений переменных.
     */
    init {
        allSources = mapOf(
                resources.getString(R.string.it_events) to
                        ItEventsComProvider(),
                resources.getString(R.string.timepad) to TimePadProvider()
        )
        SOURCE_KEY = resources.getString(R.string.data_source_key)
        THRESHOLD_KEY = resources.getString(R.string.seek_bar_key)
        THRESHOLD = sharedPrefs.getInt(THRESHOLD_KEY, 10)
        updateWithNewSet(sharedPrefs)
    }
}
```

ПРИЛОЖЕНИЕ Г

XML-код описывающий внешний вид элемента списка

Ниже представлен XML-код элемента списка, описывающий его графическое представление с помощью разметки.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <android.support.v7.widget.CardView
        android:id="@+id/post_card_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="8dp">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <RelativeLayout
                android:id="@+id/post_header_layout"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:gravity="top">

                <TextView
                    android:id="@+id/post_title"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignParentStart="true"
                    android:text="Title" />

                <TextView
                    android:id="@+id/post_time"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignParentStart="true"
                    android:layout_below="@+id/post_title"
                    android:text="t:i:m:e" />

                <ToggleButton
                    android:id="@+id/post_like"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_alignParentEnd="true"
                    android:layout_alignParentTop="true"
                    android:drawableEnd="@drawable/like_button"
```

```
                android:background="@null"
                android:textOn=""
                android:textOff=""
                android:focusable="false"
                android:focusableInTouchMode="false" />
        </RelativeLayout>

        <ImageView
            android:id="@+id/post_image"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/post_header_layout"
            android:adjustViewBounds="true"
            android:clickable="false"
            android:cropToPadding="false"
            android:foregroundGravity="center_vertical"
            android:scaleType="centerCrop"
            app:srcCompat="@drawable/placeholder_256"/>


        <TextView
            android:id="@+id/post_place"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:text="PLACE"
            android:layout_below="@id/post_image"/>

        <TextView
            android:id="@+id/post_source"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:text="Source: "
            android:layout_below="@id/post_place"/>

        <TextView
            android:id="@+id/post_source_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="some source text"
            android:layout_toEndOf="@id/post_source"
            android:layout_below="@id/post_place"/>

    </RelativeLayout>
</android.support.v7.widget.CardView>
</LinearLayout>
```