

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

СОГЛАСОВАНО

Руководитель,
профессор департамента
программной инженерии
факультета компьютерных наук,
Д. Т. Н.

_____ Д. В. Александров

«29» ноября 2021 г.

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Программная инженерия», профессор
департамента программной
инженерии, канд. техн. наук

_____ В. В. Шилов

«__» _____ 2021 г.

**Выпускная квалификационная работа
(проектно-исследовательская)**

на тему: **«Android-приложение для организации мероприятий»**

по направлению подготовки 09.03.04 «Программная инженерия»

ВЫПОЛНИЛ

студент группы БПИ172
образовательной программы
09.03.04 «Программная инженерия»

_____ А.Л. Червинский

«30» ноября 2021 г.

Реферат

Практически все организации, нацеленные на получение дополнительного дохода, рано или поздно приходят к выводу о необходимости мобильной реализации их продуктов, ведь аудитория смартфонов является одной из самых многочисленных в мире. Данные организации желают с минимальными затратами разместить свой продукт на специализированной платформе, где пользователи смартфонов смогут его найти и использовать. В этом документе описывается одна из таких платформ в области организации мероприятий «Android-приложение для организации мероприятий». Помимо предоставления функций для создания событий, данное приложение также предоставляет возможности для проведения мероприятий, которые способны снизить нагрузку на организаторов и улучшить взаимодействие с пользователем.

Данная Работа содержит 54 страниц, 4 главы, 44 рисунка, 1 таблицу, 14 источников, 6 приложений

Ключевые слова - android; событие; организация; конференция; встреча; мероприятие; собрание

abstract

Almost all organizations aimed at increasing their sales in our time, sooner or later set out to capture the audience of mobile phones, because of its large amount. Put their product on a platform where smartphone users can find it and use it. This paper describes one of these applications-an Android applications for organizing events. In addition to providing functionality for organizing these events, this application also provides functionality for conducting events, which can reduce the burden on event organizers and improve the user experience

This work contains 54 pages, 4 chapters, 44 pictures, 1 table, 14 references, 6 appendices

Keywords—android; event; organizing; conference; meetup (keywords)

СОДЕРЖАНИЕ

Введение	6
Глава 1. Обоснование необходимости разработки мобильного приложения.....	8
1.1 Анализ конкурентов разрабатываемого приложения	8
Выводы по главе	11
Глава 2. Принципы работы и архитектурные особенности	12
2.1 Основные функции приложения.....	12
2.2 Описание сценариев использования приложения	13
2.3. Выбор архитектуры мобильного приложения	16
2.3.1 Архитектура верхнего уровня	16
2.3.2 Архитектура модулей	17
2.3.3 Слой presentation	19
2.3.4 Слой Data	19
2.3.5 Слой Domain.....	20
2.3.6 Навигация в приложении	21
Выводы по главе	21
Глава 3. Реализация приложения	22
3.1 Язык программирования и основные средства разработки.....	22
3.2 Выбор минимального версии операционной системы	22
3.3 Особенности реализации паттерна MVVM.....	24
3.4 Используемые фреймворки для реализации приложения.....	26
3.5 Особенности организации работы с данными.....	27
3.6 Инъекция зависимостей.....	27
3.7 Использование библиотеки Chat UI Kit.....	28
3.8 Реализация адаптера для списка событий	29
3.9 Модели для работы с сервером.....	30
3.10 Особенности взаимодействия с серверной частью.....	31
Глава 4. Описание пользовательского интерфейса.....	34
Список использованных источников.....	54

Основные определения, термины и сокращения

Android – операционная система для мобильных устройств от компании Google.

Клиент-серверное приложение – архитектура взаимодействия приложения и сервера, при которой вся логика поведения приложения разделена на две части: клиентскую и серверную.

Мероприятие – сущность, обладающая набором некоторых событий, имеет название, время начала и окончания, окончания и другие свойства.

Событие – сущность, которая является вложенной для конференций и других событий

Спикер – участник конференции, являющийся организатором определенного события

Тариф – сущность, описывающая билеты на определенную конференцию.

REST API – метод взаимодействия между клиентом и сервером посредством HTTP запросов.

JSON – вид представления данных, в рамках проекта используется для организации взаимодействия с сервером.

Kotlin – язык разработки от компании JetBrains. С 2018 года является официальным языком разработки под операционную систему android.

Jwt токен – генерируемый на сервере набор символов для идентификации пользователя.

Навигационный граф – это двунаправленный граф, определяющий переходы между различными разделами приложения

WebSocket – протокол организации взаимодействия с сервером. Предназначен для реализации двустороннего взаимодействия клиента и сервера в режиме реального времени.

Mock объект – объект имитация, имеющая ограниченный функционал, который необходим для тестирования других объектов, обычно наследуется от родительского класса или интерфейса начального объекта.

Карутина – Облегченный поток, но она не привязана к нативному потоку и не требует переключения контекста на процессор, что дает ей возможность превосходить базовые потоки по производительности.

Введение

В последние годы, рынок мобильных устройств развивается стремительными темпами. Согласно исследованию, проведенному компанией Statista в 2021 году количество активных пользователей смартфонов превысило 3.8 миллиарда человек, что на 8 процентов выше, чем в прошлом году [1]. Подобная популярность смартфонов в основном обусловлена их функциональными возможностями. А развитием функциональных возможностей по средствам приложений предоставляемых на специализированных торговых площадках занимаются компании желающие получить дополнительный доход, выпуская при этом все больше и больше различных мобильных приложений, покрывая при этом все больше различных областей в которых пользователи желали получить мобильную реализацию того или иного продукта. За счет этого рынок мобильных приложений развивается и растет, и все больше различных компаний желают получить мобильные приложения для реализации своих продуктов.

Данная тенденция не могла обойти стороной и предметную область моей дипломной работы – организация мероприятий. Организаторы, которые желают провести какое-либо мероприятие хотят иметь возможность разместить их на какой-либо площадке, и чтобы к ним был доступ через современный смартфон, где люди могли бы узнать об этих мероприятиях, просмотреть информацию о них и записаться прямо внутри приложения. Однако рынок подобных площадок невелик и все представленные там приложения в основном представляют слишком ориентированные на конкретные группы пользователей площадки или же представляют из себя решения, которые разрабатывались для проведения единственного или очень редко повторяемого мероприятия. Поэтому я задался целью создать качественное мобильное приложение, предоставляющее возможность создавать мероприятия при помощи универсального конструктора мероприятий, а также предоставляющее функциональность для проведения этих самых мероприятий.

Данная работа посвящена разработке приложения для мобильной платформы Android. Целью работы является разработка android-приложения для организации и проведения мероприятий, а также доработка существующего сервера. Для достижения этой цели требуется решить следующие задачи:

- Проанализировать существующие решения в сфере приложений для организации мероприятий
- Реализовать адаптируемый интерфейс приложения, учитывая существующую версию приложения на платформе IOS
- Реализовать бизнес-слой приложения в соответствии с функциональными требованиями
- Добавить необходимый недостающий функционал на сервер
- Настроить клиент-серверное взаимодействие
- Произвести тестирование продукта
- Разработать техническую документацию

Данная работа состоит из 4 глав:

В первой главе были рассмотрены существующие на данный момент на торговой площадке android-приложений Google play аналоги. Для каждого были выделены основные преимущества и недостатки относительно рассматриваемого программного продукта. А затем результаты анализа были занесены в результирующую таблицу.

Во второй главе были выделены основные сценарии использования приложения, были обоснованы выборы тех или иных инструментов реализации приложения, а также была описана архитектура разрабатываемого программного продукта.

В третьей главе были рассмотрены детали реализации программного продукта

В четвертой главе были рассмотрены элементы пользовательского интерфейса, а также связанная с ними бизнес- логика, были приведены иллюстрации основных компонентов с подробными комментариями.

Глава 1. Обоснование необходимости разработки мобильного приложения

1.1 Анализ конкурентов разрабатываемого приложения

После проведенного анализа магазина мобильных приложений для платформы android – Google play, было выявлено, что основные конкуренты разрабатываемого мобильного приложения в основном представляют из себя ориентированные на одно конкретное мероприятие или слишком ориентированные на конкретные группы пользователей. Однако среди всего множество приложений есть и те, которые имеют схожий функционал с разрабатываемым программным продуктом. Для выявления основных преимуществ данного приложения было решено провести сравнительный анализ. Результаты анализа приведены ниже

Eventbrite

Страница мероприятий очень похожа на разрабатываемый программный продукт, однако гибкость и удобство редактора мероприятий заметно хуже.

Преимущества:

1. Есть возможность создавать мероприятия;
2. Есть возможность проверить билет по QR;
3. Есть лента мероприятий;
4. Есть авторизация через Google

Недостатки:

1. Нет возможности создать персональное расписание;
2. Нет вложенных событий;
3. Неудобный поиск;
4. Отсутствие привязки к выступающим;
5. Локализовано только на английском языке;

Eventzilla

Приложение разделено на 2 части, одна предназначена для организаторов, а другая для пользователей. Подобное разделение позволяет заметно улучшить возможности по созданию мероприятий.

Преимущества:

1. Подробная статистика для организаторов;
2. бесплатное;
3. очень качественный редактор;

Недостатки:

1. Отсутствует возможность работы с персональным расписанием;
2. Отсутствует лента;

Meetup

Данное приложение похоже на разрабатываемый программный продукт. Очень качественно реализована работа с аккаунтами.

Преимущества:

1. Есть Возможность входа через google play и facebook
2. Есть возможность работать с персональным расписанием

Недостатки:

1. Отсутствует разделение на категории
2. Нет возможности генерировать билеты

Eventee

Данное приложение ориентировано именно на работу с персональным расписанием, а создание публичных мероприятий вторичная функция.

Преимущества:

1. Очень качественный конструктор персонального расписания;
2. Есть возможность общения внутри приложения;
3. Есть возможность оставить отзыв на мероприятие;
4. Генерация опросов внутри приложения;
5. Интеграция с умными часами;
6. Полностью бесплатное.

Недостатки:

1. Нет возможности генерации билетов
2. Представляет из себя именно агрегатор, в котором отсутствует возможность создавать мероприятия

Whoova

Создавалось в основном для работы с конференциями, из-за чего имеет адаптированный под конференции дизайн. Очень качественное приложение, обладающее огромным функционалом для генерации и обслуживания мероприятий.

Преимущества:

1. Есть возможность общения внутри приложения;
2. Есть возможность защиты мероприятия от нежелательного доступа при помощи пароля;
3. Качественное наполнение конференций дополнительными файлами с сортировкой по типу;

Недостатки:

1. Нет возможности создавать конференции из мобильного приложения
2. Платное
3. Нет локализации

Explory

Данное приложение очень похоже на разрабатываемое приложение, также имеет ряд интересных интерфейсных решений, которые в дальнейшем можно интегрировать в разрабатываемое приложение.

Преимущества:

1. Отличный дизайн;
2. Есть возможность добавлять категории в любых количествах к конференции
3. Есть возможность просмотра мероприятий на карте;

Недостатки:

1. Есть ошибки во время работы
2. Нет возможности создания вложенных событий

EventMobi

Представляет из себя приложения для организации мероприятий. Однако данное приложение имеет возможность получения доступа к мероприятиям только через код, который вводится при входе в приложение.

Преимущества:

1. Лаконичный дизайн;
2. Быстрая установка;

Недостатки:

1. Имеет сильно ограниченный функционал

Кроме всех перечисленных преимуществ, разрабатываемое приложение будет иметь чат для общения внутри каждого отдельного мероприятия. Обобщенные результаты проведенного анализа приведены в таблице ниже:

Имя	Создание мероприятий	Лента	Персональное расписание	Чат	Работа с аккаунтами	Монетизация
Eventbrite	+	+	+/-	-	+/-	Платно
Eventzilla	+	-	-	-	-	Бесплатно
Meetup	+	-	-	-	+/-	Подписка
Eventee	-	+	+/-	+/-	+/-	Бесплатно
Whoova	-	+	+-	+	+	Платно + комиссия
Explory	+	+	+/-	-	+	Платно
EventMobi	+	-	+/-	-	-	Бесплатно
Application	+	+	+	+	+	Бесплатно

Выводы по главе

В данной главе были рассмотрены и проанализированы основные конкуренты разрабатываемого программного продукта, выявлены преимущества и недостатки и определены основные функциональные возможности для достижения конкурентно-способности на торговой площадке для android приложений Google Play.

Глава 2. Принципы работы и архитектурные особенности

2.1 Основные функции приложения

- Приложение предоставляет регистрации авторизацию для организаторов и участников;
- Приложение поддерживает «демонстрационный режим»: работу без необходимости авторизации в системе с ограничением определенных функций.
- Приложение предоставляет ленту мероприятий с возможностью фильтрации по категории и местоположению;
- Приложение предоставляет поиск по пользователям и мероприятиям;
- Приложение предоставляет возможность поделиться ссылкой на мероприятие или пользователя для владельцев устройств на платформе android;
- Приложение предоставляет возможность создавать мероприятия, события и тарифы;
- Приложение предоставляет возможность просмотра статистики по регистрациям на мероприятия;
- Приложение предоставляет возможность доступа к аккаунту авторизованного пользователя;
- Приложение предоставляет возможность просмотра страниц пользователей;
- Приложение предоставляет возможность зарегистрироваться на мероприятия;
- Приложение предоставляет возможность сохранения билетов в специализированный раздел приложения;
- Приложение предоставляет возможность работать с персональным расписанием;
- Приложение предоставляет возможность писать в чат конференций.

2.2 Описание сценариев использования приложения

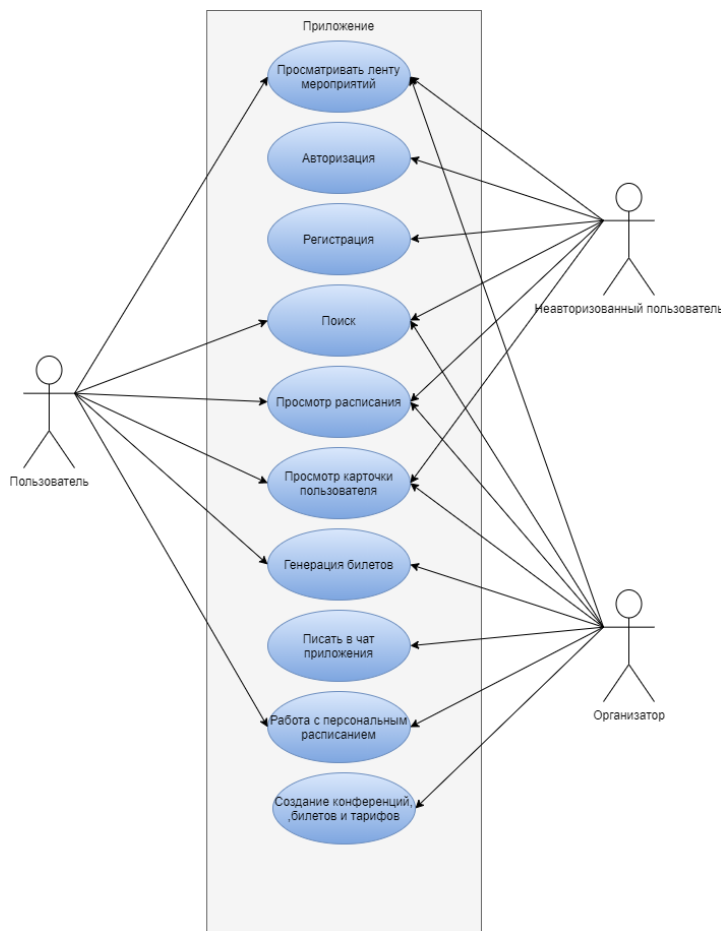


Рисунок 1 use case диаграмма

Использование приложения в демонстрационном режиме. После первого входа пользователю откроется весь доступный функционал без регистрации. При этом пользователь не сможет писать в чат или добавлять мероприятия в личное расписание, кроме того, он не сможет зарегистрироваться на мероприятия.

Авторизация/Регистрация. После того, как пользователь решит, что ему необходимо осуществить авторизацию или регистрацию, он сможет это сделать в специализированном разделе приложения

Просмотр ленты мероприятий. После нажатия кнопки “Афиша” из любой части приложения пользователю откроется лента мероприятий, со всеми публичными мероприятиями. После нажатия на любое из них пользователю отобразится карточка с дополнительной информацией о мероприятии. Кроме того, в данном разделе приложения доступна фильтрация по местоположению или категории.

Поиск конференций/событий/людей. После нажатия на иконку “лупы” в правом-верхнем углу пользователю откроется поиск, где он сможет выбрать в какой категории ему искать конференции/события/люди. После осуществления поиска, пользователь может нажать на один из найденных элементов, после нажатия пользователю откроется страница с описанием выбранного элемента.

Просмотр расписания мероприятия. После нажатия на кнопку расписания внутри карточки мероприятия пользователю откроется список всех событий данного мероприятия. Пользователь также может нажать на любое событие и посмотреть дополнительную информацию о нем.

Просмотр карточки пользователя. После нажатия на кнопку спикер/организатор пользователю откроется соответствующий человек, где будет возможность посмотреть дополнительную информацию о нем и способы связи с данным пользователем.

Генерация билетов. Когда пользователь находится в конференции он может зарегистрироваться на мероприятие, при этом будет сгенерирован билет и добавлен на вкладку билеты.

Написать в чат приложения. После входа в карточку мероприятия у зарегистрированных пользователей будет возможность писать в чат данного мероприятия.

Работа с персональным расписанием. После входа в аккаунт пользователи смогут просматривать и редактировать личное расписание.

Создание ссылки на мероприятие или человека. Находясь на карточке мероприятия или человека есть возможность создать ссылку, которую можно разместить на внешнем ресурсе.

Регистрация на мероприятие. У авторизованных пользователей есть возможность зарегистрироваться на мероприятие

Просмотр и печать билетов. После осуществления регистрации на мероприятие у человека появляется возможность просмотра и печати своих билетов в специализированной части приложения.

2.3. Выбор архитектуры мобильного приложения

2.3.1 Архитектура верхнего уровня

Поскольку приложение уже на этапе планирования казалось довольно большим, было решено разделить его на модули-фичи. Подобное разделение сильно улучшает читаемость кода, помогло упростить работу с навигационными графами [14] и немного уменьшило занимаемую память за счет использования технологии *dynamic delivery* в модулях. Данная технология позволяет загружать не все приложение, а только те модули, которые наиболее вероятно будут использоваться пользователем. А далее уже, если пользователь зайдет в не загруженную часть приложения, она будет установлена автоматически с небольшой задержкой и после уже будет находится в памяти устройства.

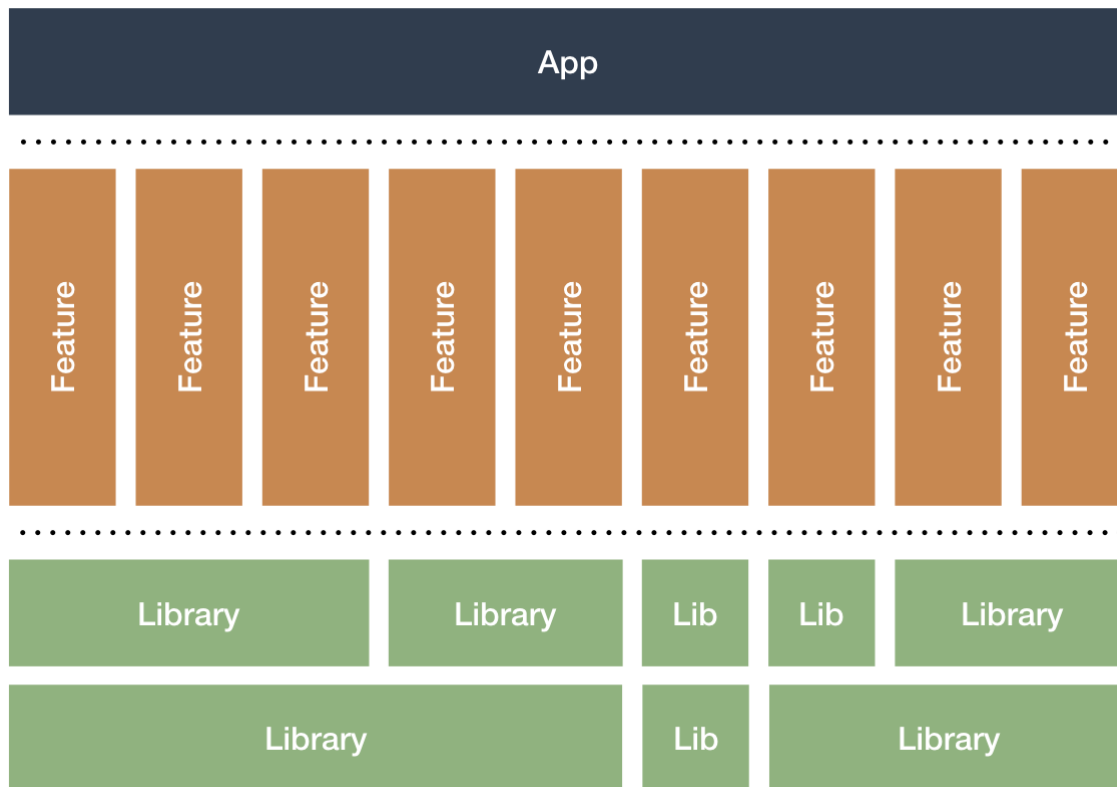


Рисунок 2 Архитектура верхнего уровня

В приложении это выглядит следующим образом:

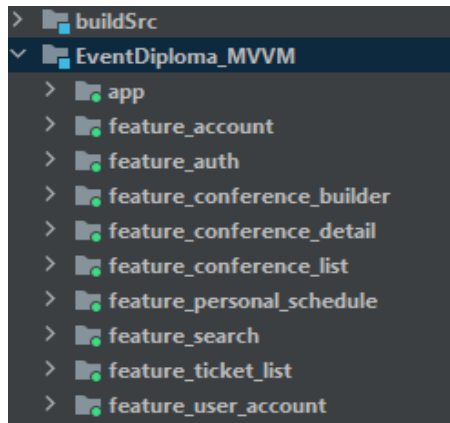


Рисунок 3 Строение приложения

2.3.2 Архитектура модулей

Каждый модуль приложения выполнен по принципам Clean Architecture.[5] Данный подход предполагает разделение модулей на слои, каждый из которых слабо зависит от других, что дает следующие преимущества:

- Упрощает тестирование
- Независимость UI (замена интерфейсных элементов не влияет на остальные модули)
- Независимость от репозитория (замена REST API и БД методов не влияет на остальные модули)
- Независимость от внешних сервисов и фреймворков

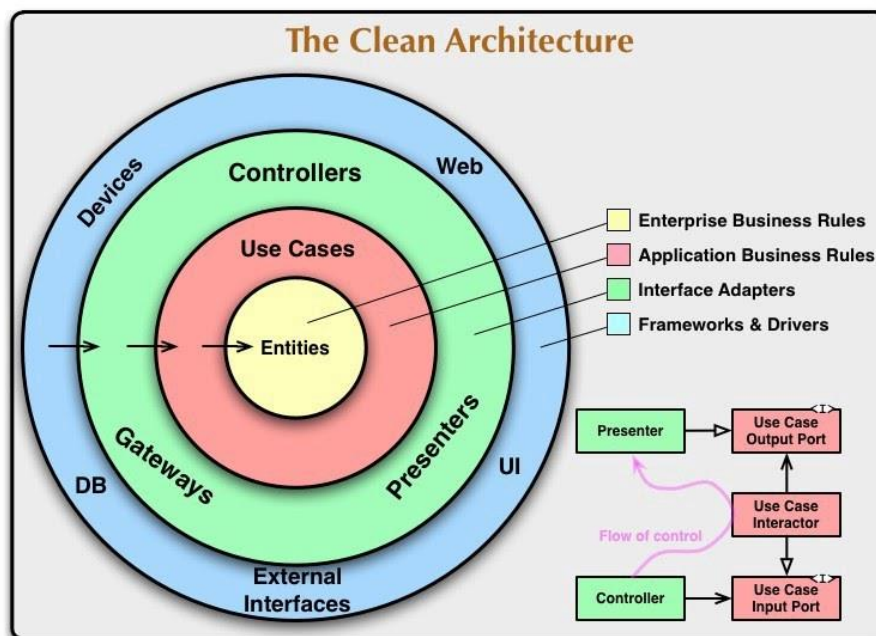


Рисунок 4 Clean Architecture диаграмма

Также данное архитектурное решение имеет реализацию специально для android. В ней всего 3 слоя: presentation, domain data.

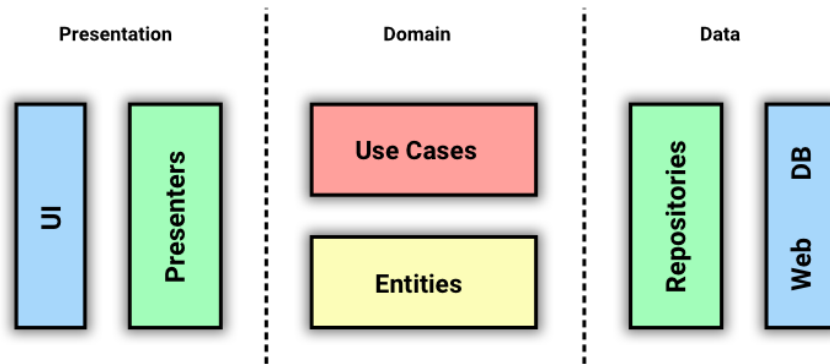


Рисунок 5 Clean Architecture android

Снизу приведена UML диаграмма классов в контексте чистой архитектуры для экрана списка конференций, остальные экраны имеют примерно такую же схему.

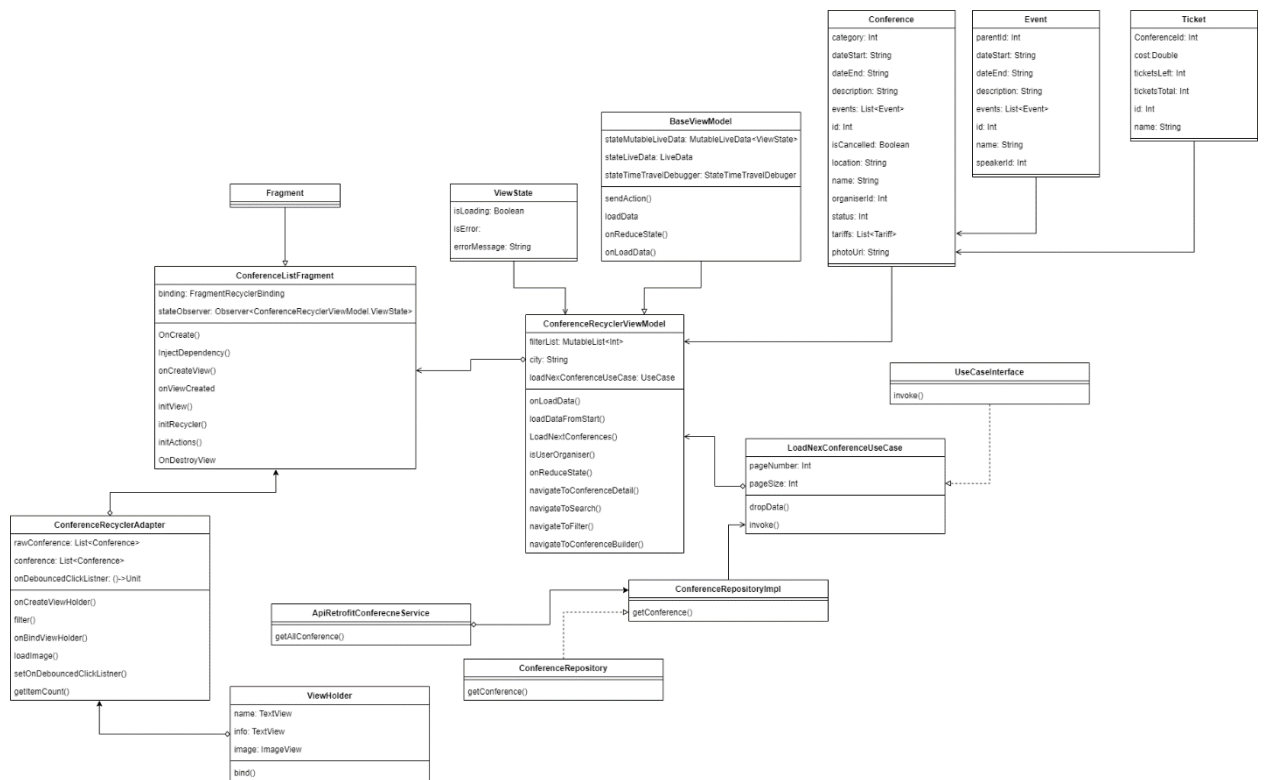


Рисунок 6 Диаграмма классов

2.3.3 Слой presentation

Для реализации presentation слоя была выбрана архитектура MVVM (model, view, view-model) [6], она является логичным продолжением MVP, но в отличие от MVP в данной архитектуре обратная связь между слоями осуществляется при помощи паттерна observer [7], это еще больше уменьшает зависимость между компонентами, улучшая тестируемость кода и уменьшая затраты на устранение ошибок. Кроме того, именно эту архитектуру советуют для использования в Google, как следствие выпуская большое количество вспомогательных библиотек для реализации данного паттерна.

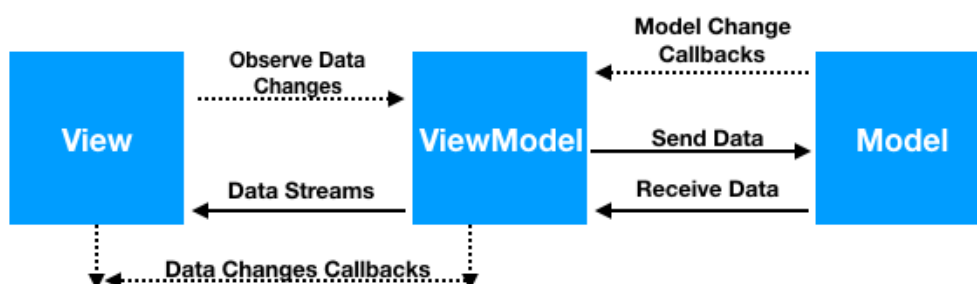


Рисунок 7 MVVM диаграмма

2.3.4 Слой Data

Главной целью данного слоя является связь с удаленным сервером или базой данных на устройстве. Данный слой имеет следующие пакеты:

model – содержит все модели, использующиеся для работы с сервером и базой данных

network – содержит классы для работы с сервером

database – содержит классы для работы с базой данных

repositoryImpl – содержит реализацию интерфейсов из слоя domain, каждый класс соответствует паттерну repository [11] и имеет следующую структуру:

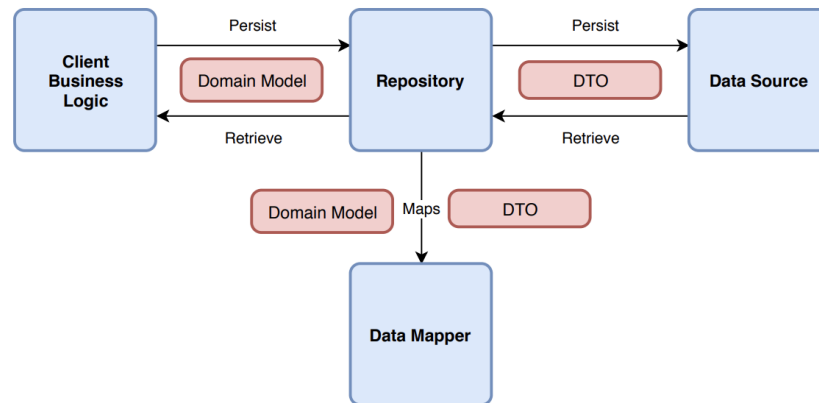


Рисунок 8 Паттерн Repository

Подобное разделение данного слоя свойственно для чистой архитектуры, это позволяет добавлять новые функциональности или менять старые без необходимости менять что-либо в других слоях. Также вся работа с данными осуществляется посредством паттерна репозиторий, классы реализации, которого доступны для слоя domain.

2.3.5 Слой Domain

Данный слой отвечает за всю бизнес-логику модуля, он содержит следующие пакеты:

Model – содержит модели бизнес- логики приложения

Repository – содержит интерфейсы для реализации в слое data.

UseCase – содержит бизнес-логику модуля.

Данный слой, как и слой Data является слабо связанным с другими слоями чистой архитектуры. Что позволяет довольно легко менять бизнес-логику приложения без необходимости изменения других слоев. Кроме того, подобное вынесение бизнес- логики вместе с внедрением зависимостей сильно упрощает тестирование.

2.3.6 Навигация в приложении

Также важным архитектурным решением является использование паттерна Single Activity app. Данный паттерн предполагает использование одной activity [2] с контейнером для фрагментов(fragment) [3] на все приложение. Это позволяет интегрировать navigation component [4] для каждого фрагмента. При помощи этого подхода возможно быстро и просто осуществлять навигацию по приложению при помощи навигационных графов. При таком подходе activity всегда остается на экране, а меняются фрагменты внутри нее.

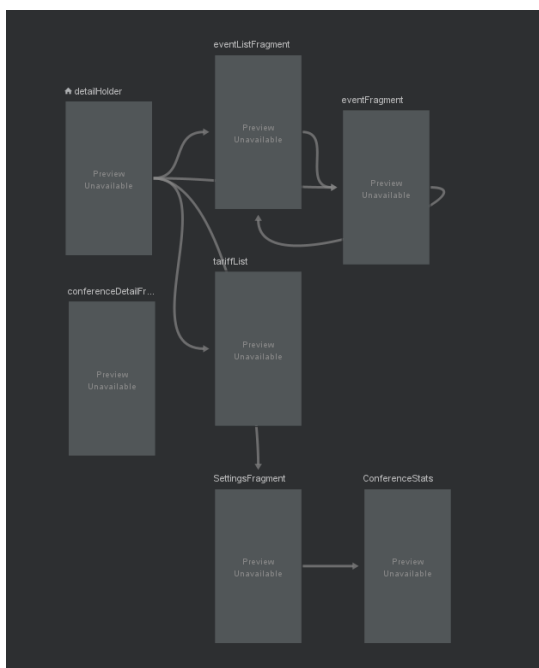


Рисунок 9 Пример навигационного графа

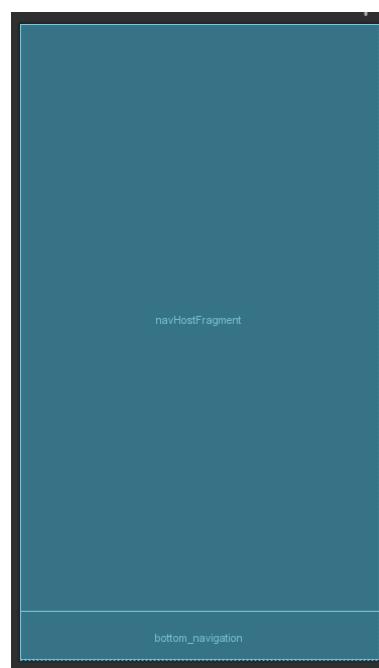


Рисунок 10 Структура Activity

Кроме того, так как в данном приложении используется разделение на модули, данная реализация навигации позволила выделить навигационные графы для каждого модуля отдельно, что позволило заметно упростить работу с ними. При этом при вызове модуля, его граф загружается динамически, и программа начинает использовать новый граф для навигации.

Выводы по главе

В данной главе были описаны основные функции приложения и приведена архитектура программного продукта.

Глава 3. Реализация приложения

3.1 Язык программирования и основные средства разработки

В качестве языка программирования был выбран Kotlin, разработанный компанией JetBrains. В 2018 году Google сделала его официальным языком для разработки под Android, с этих пор все новые и большинство старых приложений переходят на него. В качестве среды разработки была выбрана IDE Android Studio, которая является стандартом для разработки Android приложений. Для запуска отладочного сервера использовалась IntelliJ IDEA и pgAdmin [12] для работы с локальной базой данных.

3.2 Выбор минимальной версии операционной системы

Крайне важным этапом проектирования будущего приложения является выбор минимальной версии операционной системы, которую будет поддерживать приложение. Для этих целей компания Google предоставляет статистику количества пользователей для каждой версии, при помощи нее довольно легко определить сколько пользователей могут использовать приложение.

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	
4.2 Jelly Bean	17	99,8%
4.3 Jelly Bean	18	99,5%
4.4 KitKat	19	99,4%
5.0 Lollipop	21	98,0%
5.1 Lollipop	22	97,3%
6.0 Marshmallow	23	94,1%
7.0 Nougat	24	89,0%
7.1 Nougat	25	85,6%
8.0 Oreo	26	82,7%
8.1 Oreo	27	78,7%
9.0 Pie	28	69,0%
10. Q	29	50,8%
11. R	30	24,3%

Рисунок 10 Распределение людей по версиям

Известно, что все компоненты, которые будут использоваться в приложении стабильно работают с версии 4.4, однако использование версии ниже 5 приводит к необходимости использовать legacy библиотеки, поэтому было решено использовать версию android 5.0, как видно из таблицы в таком случае потери составят около 1.4 процента, однако польза от данного решения перекрывает данный недостаток. Поэтому для реализации была выбрана версия 5.0.

3.3 Особенности реализация паттерна MVVM

Связь между View и ViewModel. Для реализации связи между фрагментами и ViewModel было решено разработать базовый класс, который будет наследником ViewModel предоставляемого стандартной библиотекой. Данный класс имеет много различных полей, в частности для связи с View данный класс имеет дженерик объект stateLiveData, являющийся наследником класса LiveData, который предоставляется базовой библиотекой. На изменения данного объекта возможно подписаться, при этом за счет использования дженериков можно использовать любой возможный тип возвращаемого значения, что делает данную реализацию универсальной. Ниже приведен пример создания класса ViewModel.

```
class ConferenceStatsViewModel: BaseViewModel<ConferenceStatsViewModel.ViewState,  
    ConferenceStatsViewModel.Action>(ViewState()) {
```

Рисунок 12 Пример создания класса ViewModel

В большинстве случаев в качестве типов для объекта LiveData передаются подобные, приведенному ниже data классы, у которых могут различаться поля. Именно на изменения объектов этих классов подписываются фрагменты.

```
data class ViewState(  
    val isLoading: Boolean = true,  
    val isError: Boolean = false,  
    val errorMessage: String = "",  
    val conference: Conference? = null  
) : BaseViewState
```

Рисунок 13 Пример класса LiveData

Связь между ViewModel и UseCase. Как и для связи между View и ViewModel, для реализации взаимодействия, используется базовый класс, в котором в качестве второго дженерика передается тип события, которое генерируется при получении ответа от бизнес-слоя. Сами события в большинстве случаев выглядят следующим образом:

```
interface Action : BaseAction {  
    class LoadSuccess(val conference: Conference) : Action  
    class Failure(val message: String) : Action  
}
```

Рисунок 14 Пример интерфейса события

Реализация отправки осуществляется за счет метода из базового класса `sendAction` при этом вызывается `OnReduceState`. Пример обработки ответа от объекта бизнес-слоя приведен ниже:

```
fun loadConference(){
    viewModelScope.launch { this: CoroutineScope
        loadConferenceByIdUseCase(conferenceId).also { result ->
            val action = when (result) {
                is LoadConferenceByIdUseCase.Result.Success ->
                    Action.LoadSuccess(result.data)
                is LoadConferenceByIdUseCase.Result.Error ->
                    Action.Failure("Ошибка подключения")
                else -> Action.Failure("Ошибка подключения")
            }
            sendAction(action)
        }
    }
}
```

Рисунок 15 Пример обработки ответа от бизнес-слоя

Полученное событие бизнес-слоя обрабатывается, после чего изменяется объект `LiveData`, а фрагмент получает оповещение об этом. Базовая реализация метода обработки события бизнес-слоя приведена ниже:

```
override fun onReduceState(viewAction: Action): ViewState = when (viewAction) {
    is Action.LoadSuccess -> {
        state.copy(
            isLoading = false,
            isError = false,
            conference = viewAction.conference
        )
    }
    is Action.Failure -> {
        val message = viewAction.message
        state.copy(
            isLoading = false,
            isError = true,
            errorMessage = message
        )
    }
}
```

Рисунок 16 Определение действий для фрагмента

3.4 Используемые фреймворки для реализации приложения

navigation component – предоставляет инструменты для упрощения работы с менеджером фрагментов, используемым для навигации, также улучшает понятность последовательности экранов для разработчика.

Room – данный фреймворк предоставляет возможность создавать и работать с базой данных на устройстве.

Google gms, places – библиотеки предоставляемые Google, дают возможность работы с google api, в частности в данном проекте используется для работы с картами и загрузки предсказаний для текстовых полей, ассоциированных с картами.

Stomp client – предоставляет удобный клиент для реализации чата, работает по протоколу Stomp.

rxKotlin – библиотека для реализации принципов реактивного программирования для JVM.

Chat UI kit – предоставляет библиотеку для реализации интерфейсных элементов чата.

Google materials – представляет из себя набор готовых компонентов для реализации пользовательского интерфейса. Был выбран, так как является стандартом google и имеет нативную поддержку на платформе android.

Dagger2 – представляет собой библиотеку, которая помогает реализовать паттерн Внедрение зависимости (Dependency Injection), является стандартом для android разработки.

Retrofit – REST клиент для Android. Помогает более качественно организовать взаимодействие с сервером.

Espresso – Для осуществления ui тестирования во время разработки приложения используется фреймворк espresso

Picasso – Данный фреймворк предоставляет инструменты для загрузки и редактирования изображений, также предоставляет возможности для интеграции в интерфейсные элементы устройства.

SockJS – Представляет из себя набор готовых элементов для реализации STOMP протокола на Spring-сервере

3.5 Особенности организации работы с данными

Вся работа с данными в приложении осуществляется асинхронно. Это необходимо для того, чтобы пользовательский интерфейс не зависал до окончания загрузки. При этом результат каждого асинхронного запроса обрабатывается и в случае возникновения ошибок, информация доносится до пользователя через основной поток.

Одной из особенностей использования мобильных приложений заключается в том, что существует вероятность того, что в любой момент возможна потеря соединения с сервером, при этом теряется возможность использования приложения. Для решения этой проблемы было решено использовать внутреннюю базу данных Room. Для корректной работы был выбран следующий алгоритм действий для каждой сущности, которую необходимо сохранять на устройстве:

Проверка возможности загрузки данных. На данном этапе приложение осуществляет асинхронную попытку загрузки данных с сервера, если приложение не имеет возможности загрузить данные, оно загружает их из базы и передает пользователю.

Если на прошлом этапе загрузка осуществилась успешно, приложение осуществляет сохранение новых данных в базу, кроме того, если сервер загрузил данные, которые уже находятся на устройстве, и они различаются, то произойдет перезапись данных на новые, однако данная программа не сохраняет изображения, загруженные с сервера в базу. Это сделано специально для экономии занимаемой базой места.

Далее приложение отправляет данные в presentation слой, где данные становятся видны пользователю и появляется возможность взаимодействовать с ними.

Кроме того, в приложении используются конфиденциальные данные пользователей, которые требуют более безопасного хранения. Для их сохранности используется фреймворк `sql cypher`, [8] который позволяет добавить дополнительное шифрование данных пользователя перед сохранением в базу данных Room [13], что значительно повышает затраты необходимые для получения несанкционированного доступа к данным пользователя.

Также в данном приложении используются данные содержащие в себе изображения. Работа с такими данными в приложении организована специальным образом. Для экономии времени загрузки запросы для получения данных с изображениями разделяются и выполняются последовательно в разных карутинах. Сначала загружаются текстовые данные и передаются в интерфейс, после чего осуществляется загрузка изображений.

3.6 Инъекция зависимостей

В подобных по объему, разработанному приложению, программных продуктов используется большое количество различных экранов, библиотек и моделей. При этом разработчику программного продукта приходится создавать много различных объектов, каждый из объектов может зависеть от других. По итогу, это способно замедлить разработку программного продукта и увеличить затраты на тестирование.

Для решения этих проблемы было решено использовать средства для внедрения зависимостей. Из всего их множества была выбрана, проверенная временем, способная корректно работать в многомодульном проекте, библиотека Dagger 2. Она позволяет

полностью автоматически на основе аннотаций и приведенных в специализированных разделах программы, описаниях генерировать инициализации для объектов и приводить их в нужные разделы программы.

Dagger работает на основе графа зависимостей, который dagger генерирует на основе интерфейсов и классов, помеченных аннотациями приведенными ниже:

3. **Component.** Данной аннотацией помечается интерфейс, в котором приводится метод связи между объектом нуждающемся в инъекции зависимостей и библиотекой.
4. **Module.** Данной аннотацией помечается класс, представляющий из себя фабрику для объектов.
5. **Provides.** Данной аннотацией помечаются методы внутри фабрики, предоставляющие готовые объекты определенного в них типа.
6. **Inject.** Данной аннотацией помечается объект, нуждающийся в инъекции в него зависимости

Как писалось ранее Dagger способен уменьшить трудозатраты на написание программного кода, для иллюстрации данного свойства приводится инициализация двух одинаковых объектов: `loadConferenceByIdUseCase` и `loadConferenceByIdUseCaseNonDagger`, первый использует для инициализации dagger, а второй инициализируется без него.

```
@Inject
lateinit var loadConferenceByIdUseCase: LoadConferenceByIdUseCase

val retrofit = Retrofit.Builder()
    .baseUrl(ServerConstants.LOCAL_SERVER)
    .addConverterFactory(GsonConverterFactory.create(GsonBuilder().create()))
    .build()

var loadConferenceByIdUseCaseNonDagger =
    LoadConferenceByIdUseCase(
        ConferenceDetailRepositoryImpl(
            retrofit.create(ApiRetrofitConferenceDetailService::class.java)
        )
    )
```

Рисунок 17 Сравнение «С dagger и без»

Как видно из кода количество необходимого для инициализации кода выше. Также важно отметить, что на любом этапе инициализации вложенных объектов, возможна замена настоящего объекта на его mock версию, что сильно упрощает тестирование.

3.7 Использование библиотеки Chat UI Kit

Как было описано в функциях приложения в данном программном продукте используется чат. Так как проектирование пользовательского интерфейса очень трудоемкая задача было

решено использовать готовую библиотеку. Из всего множества наиболее подходящей на момент анализа, по функционалу для данного приложения, оказалась Chat UI Kit.

Для работы данной библиотеки необходимо разместить в пользовательском интерфейсе предоставляемый библиотекой контейнер для списка сообщений, после чего внутри Activity привязать к данному контейнеру специальный адаптер сообщений, предоставляемый библиотекой, в который необходимо передать созданные разработчиком контейнеры для сообщений. Далее уже вся работа с чатом производится по средством адаптера, через который возможно добавлять новые сообщения на пользовательский интерфейс или удалять старые.

Процесс создания контейнеров для сообщений схож с разработкой фрагментов. У каждого контейнера есть его интерфейсное представление и ассоциированный с ним класс имеющий доступ к интерфейсу. Каждый такой класс наследуется от ChatMessageIncomingViewHolder или ChatMessageOutcomingViewHolder, в зависимости от отправителя сообщения. Каждый из них имеет доступ к ассоциированным с ними UI элементам чата. Для того, чтобы данный класс корректно проинициализировал сообщение необходимо в методе OnBind(message) записать все необходимые поля сообщения внутрь view, при этом нет необходимости инициализировать элементы, которые имеют названия и тип схожий с базовым, как например текст сообщения в данной реализации. Пример работы для исходящего сообщения приведен ниже:

```
private val beautyTimeFormatter = SimpleDateFormat( pattern: "HH:mm", Locale.getDefault())
override fun onBind(message: ChatMessage) {
    super.onBind(message)
    itemView.findViewById<TextView>(R.id.messageTimeOut).text = beautyTimeFormatter.format(message.createdAt)
}
```

Рисунок 18 Пример инициализации UI сообщения

3.8 Реализация адаптера для списка событий

В android приложениях для представления связанного контента в списках необходимы адаптеры контента. Они способны привести контент, представляемый в виде массива или списка, к представлению в котором пользователь сможет увидеть его в виде интерфейсного элемента и взаимодействовать с ним. Обычно реализация адаптеров для списков - это тривиальная задача для android разработчика, однако для реализации списка событий необходим нестандартный адаптер. В данном адаптере необходимо использовать сразу несколько различных вариантов ячеек и при этом использование тех или иных ячеек зависит от ассоциированного с ними контента.

Для реализации такого адаптера было принято решение разделить всю его логику на два класса: первый представляет связь с UI элементами – класс представления, а второй приводит список к нужному виду и передает контент из него по средством запроса через индекс – класс логики.

Алгоритм приведения списка класса логики, следующий: изначально создается пустой список промежуточных данных имеющий, помимо основного контента, дополнительные поля для титула и типа. Далее программа проходит по первоначальному списку и в

зависимости от того изменилась ли дата событий: создает объект типом титула для нового дня или объект с типом события.

После того, как промежуточный список сгенерирован, класс представления адаптера запрашивает у класса логики количество элементов. Далее данный класс будет запрашивать по индексу весь список из промежуточных данных и на основе типа будет генерировать UI элементы.

3.9 Модели для работы с сервером

Для взаимодействия с сервером используются следующие структуры данных представляющие из себя data классы:

AuthInfo – данная структура представляет из себя ответ сервера на запрос авторизации, в случае удачной авторизации именно через эту структуру передается Jwt токен пользователя.

Conference – представляет из себя структуру отдельного мероприятия, содержит все поля характеризующие мероприятия в том числе тарифы и события.

Event – представляет из себя структуру отдельного события.

LoginRequestBody – модель для описания тела запроса на авторизацию. Включает логин и хешированный пароль

Tariff – модель, описывающая отдельный тариф мероприятия.

User – описывает все поля пользователя, для безопасности в данную структуру не включен пароль и логин пользователя.

Ticket – Представляет из себя модель билета

ChatMessage- содержит все данные о сообщении, отправленном пользователем

3.10 Особенности взаимодействия с серверной частью

Данное приложения является android-частью проекта посвященного организации мероприятий. Для работы всех приложений был написан единый REST API сервер. Для осуществления взаимодействия используется HTTP запросы, а для организации данных JSON. Также в данном приложении есть реализация чата. Для организации чата использовать стандартные HTTP запросы вычислительно затратно, поэтому для уменьшения нагрузки на сервер было решено использовать протокол STOMP[10], который осуществляет работу при помощи WebSocket[9]. Он позволяет подписываться на изменения чатов каждой отдельной конференции и получать сообщения автоматически.

STOMP over WebSocket messages

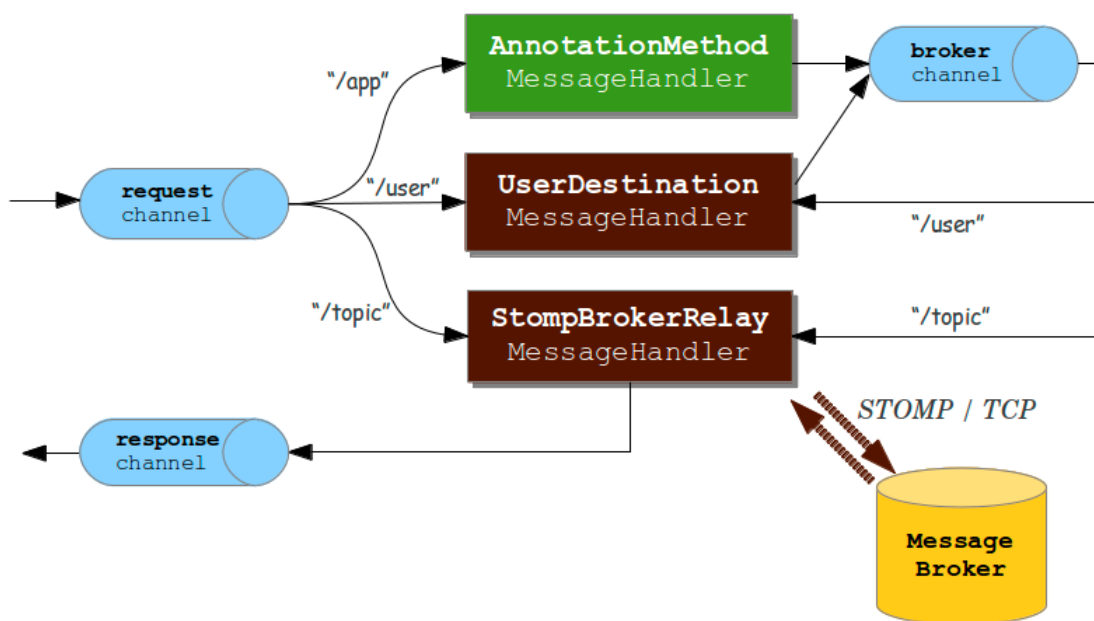


Рисунок 19 Архитектура сервера с использованием STOMP протокола

Описание методов взаимодействия с сервером:

URL	Тип	Параметры	Назначение	Возвращает
/api/reg/organizer	POST	SignUp	регистрация организатора	success, message
/api/login/organizer	POST	Login	авторизация организатора	success, message
/api/org/createConference	POST	Conference	создание конференции	success

/api/org/addEventsToConference	POST	[Event], Conference.ID	добавление новых событий в конференцию	success
/api/org/addTariffsToConference	POST	[Tariff], Conference.ID	добавление новых тарифов в конференцию	success
/api/reg/user	POST	SignUp	регистрация пользователя	success, message
/api/login/user	POST	Login	авторизация пользователя	success, message
/api/usr/info/{username}	GET	User.username	информация о пользователе	User
/api/usr/info/byId/{ID}	GET	User.ID	информация о пользователе	User
/api/usr/eventsWhereSpeaker/{username}	GET	User.username	события определенног о спикера	[Event]
/api/usr/getConferenceEvents/{ID}	GET	Conference.ID	события определенной конференции	[Event]
/api/usr/childrenEvents/{ID}	GET	Event.ID	события внутри события	[Event]
/api/usr/conference/{ID}	GET	Conference.ID	информация о конференции	Conference

/api/usr/event/{Event.ID}	GET	Event.ID	информация о событии	Event
/api/usr/searchConferences/{text}	GET	text	поиск по конференциям	[Conference]
/api/usr/searchEvents/{text}	GET	text	поиск по событиям	[Event]
/api/usr/searchUsers/{text}	GET	text	поиск по пользователям	[User]
/api/usr/findConferencesByCategory/{ID}	GET	Category.ID	выборка из ленты конференций по категории	[Conference]
api/chat/send	POST	ChatMessage, conference.id	Позволяет отправлять сообщение в чат конференции	success message
/chat/\$id/queue/messages	WebSocket	-	Позволяет подписаться на изменения чата определенной конференции	ChatMessage

Выводы по главе

В данной главе были описаны основные технические средства, использованные в приложении, обоснован выбор некоторых библиотек и фреймворков и описаны алгоритмы работы, используемые в данном приложении.

Глава 4. Описание пользовательского интерфейса

4.1 Общие принципы

Каждый элемент интерфейса выполнен с учетом принципов проектирования адаптивных интерфейсных элементов, поэтому данное приложение будет корректно отображаться практически на любом мобильном устройстве под управлением операционной системы android. Также каждый UI элемент вынесен в тему приложения, из которой без особого труда возможно изменить интерфейсные элементы по всему устройству. Кроме того, данное приложение использует только локализуемые параметры для отображения в интерфейсных элементах, что позволяет автоматически переводить приложение на любой добавленный язык, в зависимости от языка устройства. Сейчас приложение может работать с русским и английским языками.

4.2 Демонстрационный режим

При первом входе в приложение, пользователь получит доступ к ограниченным функциональным возможностям, которые можно использовать без необходимости авторизации. В демонстрационном режиме пользователю доступны следующие функции: возможность доступа к ленте конференций, возможность просмотра конференций, доступ к событиям и вложенным событиям, а также возможность поиска конференций и людей. Если пользователь попытается попасть в область приложения, к которой у него нет доступа, в таком случае приложение сообщит об этом, а доступ к разделу будет заблокирован.

4.3 Лента конференций

После запуска приложения пользователь попадет на экран со списком конференций, здесь пользователь может выбрать интересующую его конференцию или воспользоваться фильтрами, переход к которым находится слева вверху. Фильтровать конференции возможно по местоположению и категории, при нажатии кнопки сбросить все фильтры возвращаются в исходное состояние, а пользователя переносит назад к списку конференций.

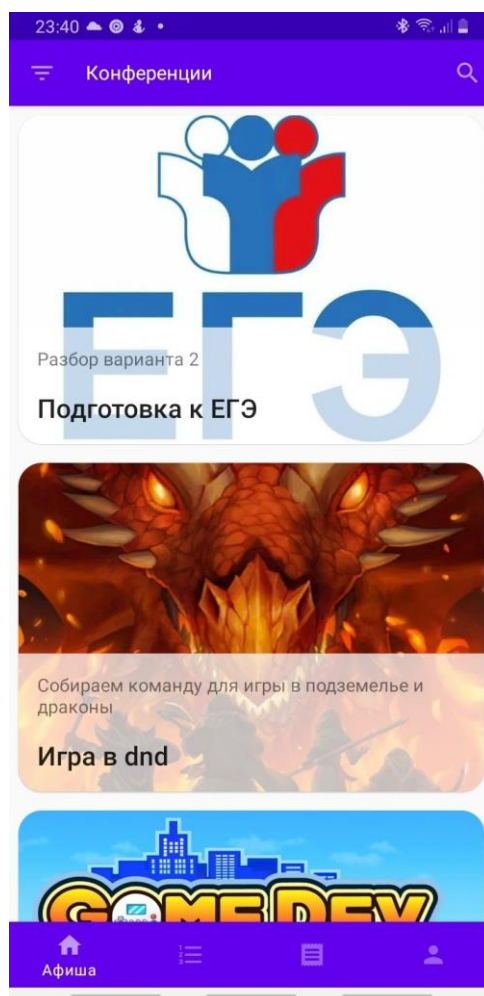


Рисунок 20 Экран Афиши мероприятий

На экране фильтра пользователь может выбрать интересующие его категории, а также выбрать город, в котором необходимо искать конференции. Также на этом экране возможно сбросить выбор нажатием кнопки "сбросить".

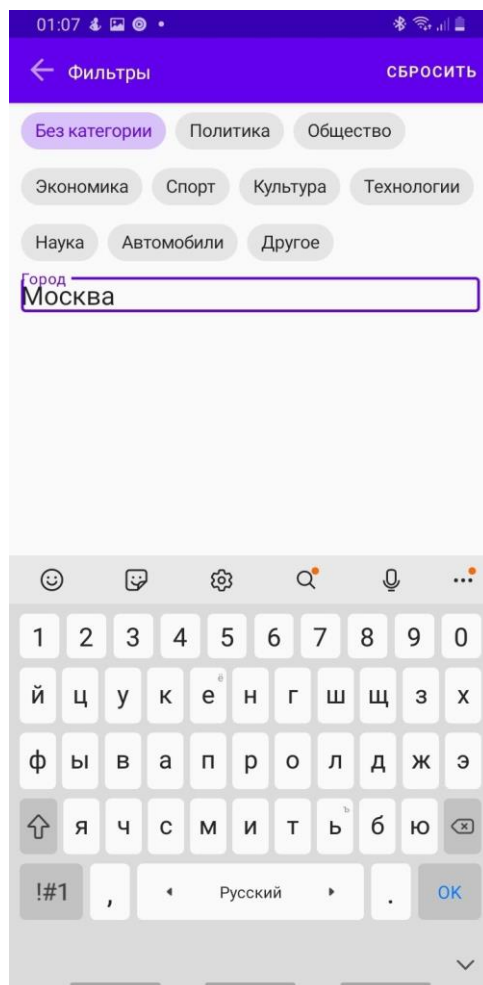


Рисунок 11 Экран фильтров мероприятий

4.4 Авторизация и регистрация

В крайнем правом разделе нижнего меню находится раздел работы с аккаунтом, здесь пользователь может произвести авторизацию или если у него еще нет аккаунта зарегистрироваться. Как только пользователь успешно авторизуется, ему отобразится экран с информацией об аккаунте. Также у пользователя имеется возможность запомнить свой аккаунт, в таком случае пользователь продолжит быть авторизованным даже после перезапуска или авторизоваться как организатор в таком случае пользователю откроется возможность создавать конференции, тарифы и события.

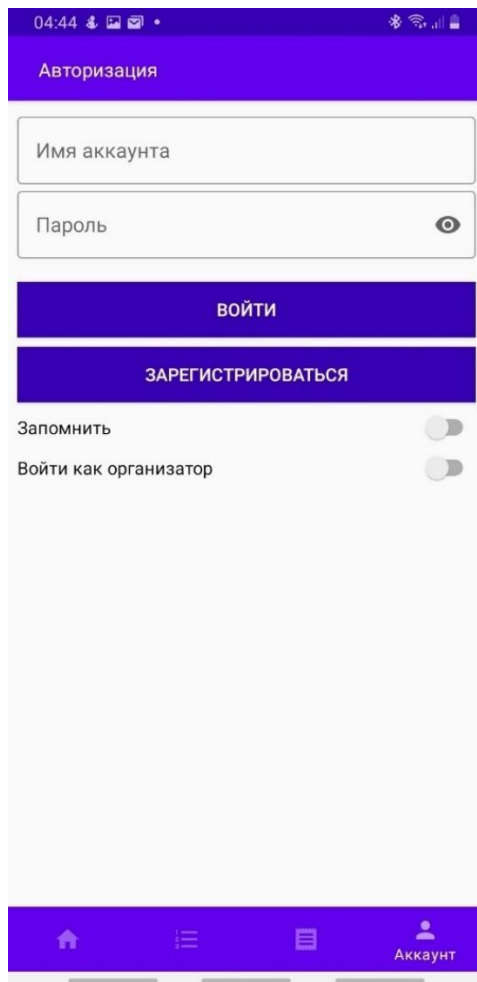


Рисунок 12 Экран авторизации

Если пользователь нажмет кнопку регистрации его перенесет на экран регистрации, где он сможет ввести все поля и зарегистрироваться, также пользователь при помощи специального элемента может переключить тип аккаунта, который он хочет создать.

04:45

← Зарегистрироваться

Имя аккаунта

Пароль

Имя

Фамилия

Электронная почта

Номер телефона

Обо мне

Зарегистрироваться как организатор

ЗАРЕГИСТРИРОВАТЬСЯ

Аккаунт

Рисунок 13 Экран регистрации

4.5 Аккаунт

После того как пользователь авторизовался, его переносит на экран его аккаунта, здесь он может зайти в настройки, нажав на значок шестеренки справа вверху.

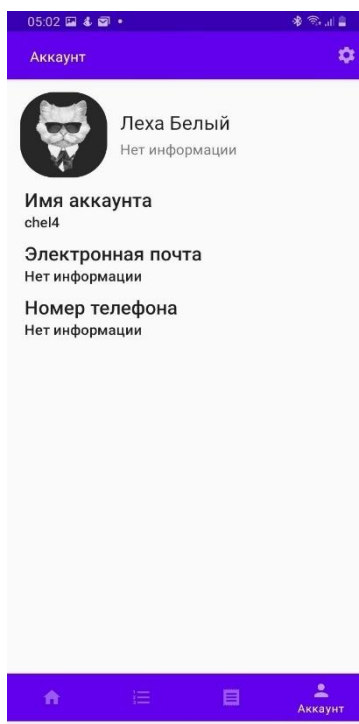


Рисунок 14 Экран аккаунта

Внутри экрана настроек аккаунта пользователь может создать ссылку на свой аккаунт, при помощи которой любой человек на устройстве под управлением android сможет попасть на экран с его аккаунтом.

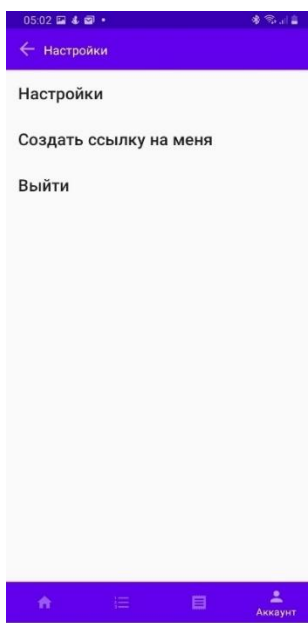
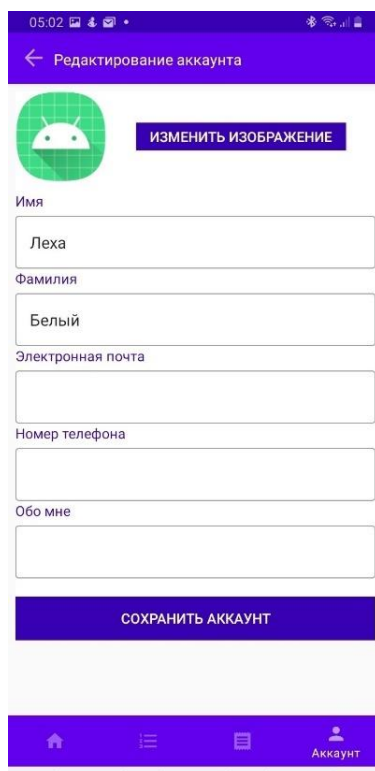


Рисунок 15 Экран настроек аккаунта

На экране редактирования пользователь может изменить информацию о своем аккаунте, и загрузить изображение из памяти устройства.



05:02

← Редактирование аккаунта

ИЗМЕНИТЬ ИЗОБРАЖЕНИЕ

Имя

Леха

Фамилия

Белый

Электронная почта

Номер телефона

Обо мне

СОХРАНИТЬ АККАУНТ

Аккаунт

Рисунок 16 Экран редактирования аккаунта

4.6 Экран поиска

Если пользователь на экране списка конференций нажал на значок лупы, то его перенесет в раздел поиска, где он сможет выбрать одну из двух вкладок, с тем, что именно ему необходимо найти. При этом при нажатии символа обратной стрелки пользователь будет возвращен в раздел поиска конференций, а при нажатии иконки крестика, весь контент из поисковой строки удалится.

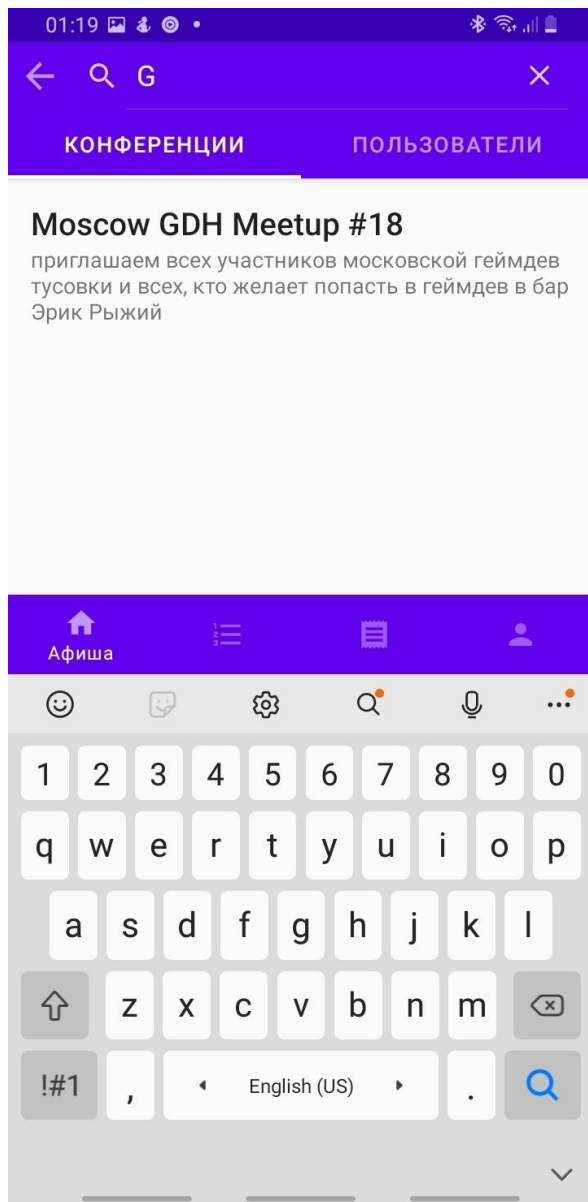


Рисунок 17 Поиск по конференциям

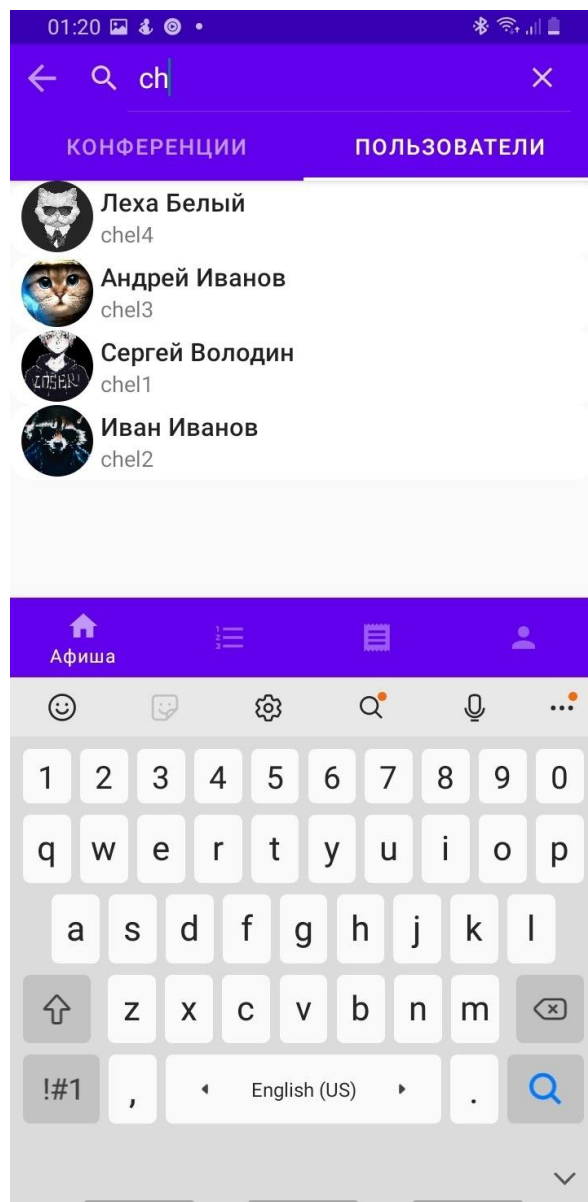


Рисунок 28 Поиск по пользователям

4.7 Экран конференции

После нажатия на любую конференцию пользователю откроется страница с данным мероприятием со всей информацией об этой конференции. При этом у пользователя будет возможность развернуть карту с местоположением и используя кнопки справа снизу на карте получить маршрут до места назначения через google карты.

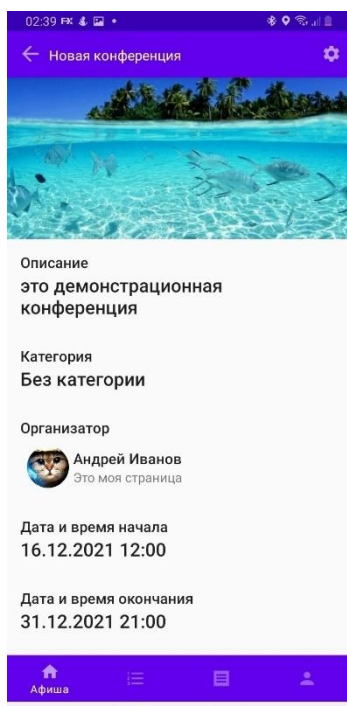


Рисунок 29 Конференция 1



Рисунок 30 Конференция 2

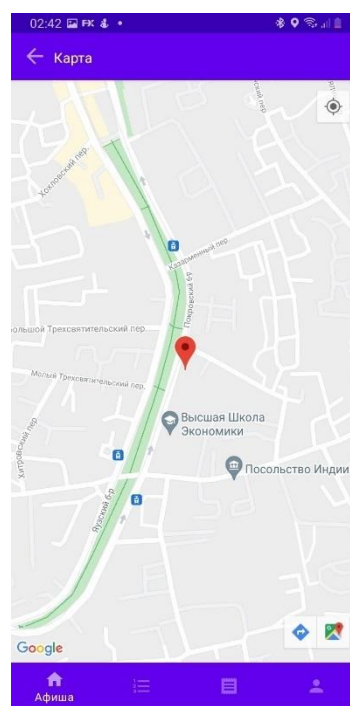


Рисунок 31 Расширенная карта

Также если на этом экране нажать иконку шестеренки справа вверху, пользователю откроется меню данной конференции, где для всех пользователей будет возможность создать пригласительную ссылку.

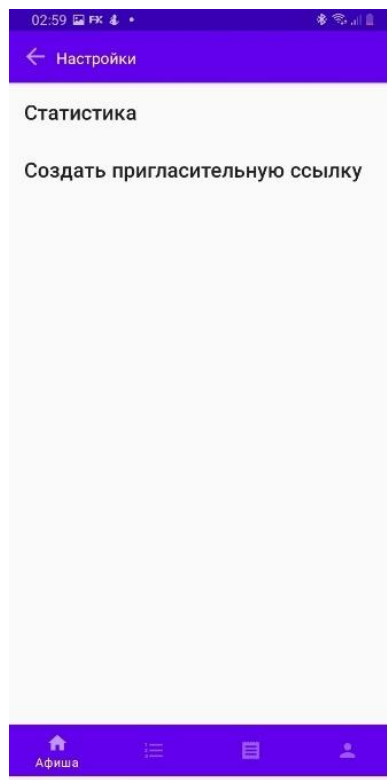


Рисунок 32 Настройки конференции

Для организаторов на экране настроек конференции есть возможность открыть статистику по регистрациям

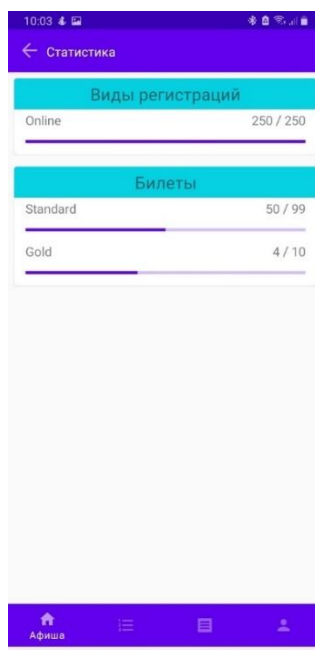


Рисунок 33 Статистика конференции

Если пользователь нажмет на любое событие из списка, то его перенесет на экран данного события, где пользователь сможет при помощи специальной кнопки добавить событие в избранное.

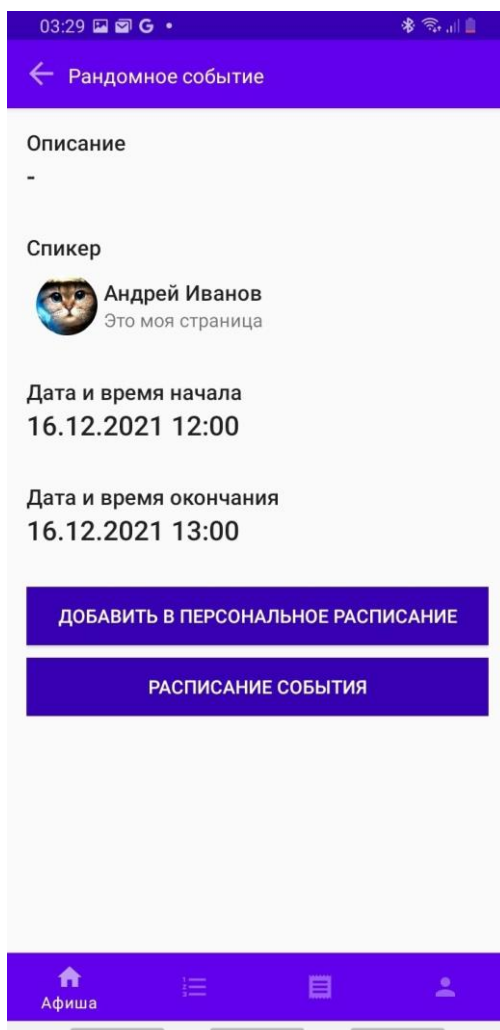


Рисунок 35 Описание события

4.9 Тарифы

Если на экране конференции нажать на зарегистрироваться на мероприятия, то пользователю откроется возможные варианты регистрации.

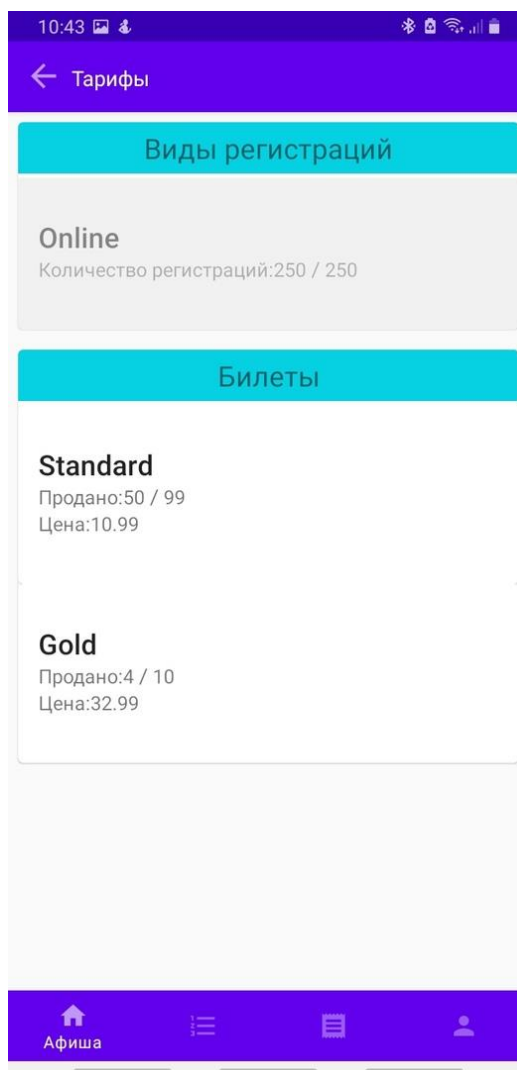


Рисунок 36 Список тарифов

Если на данном этапе выбрать билет, на который еще остались места, то билет добавится в раздел билеты.

4.10 Чат

Если на экране конференции нажать на чат, то откроется чат данного мероприятия. При этом если нажать на кнопку отправки, сообщение отправляется всем пользователям находящимся в данной комнате

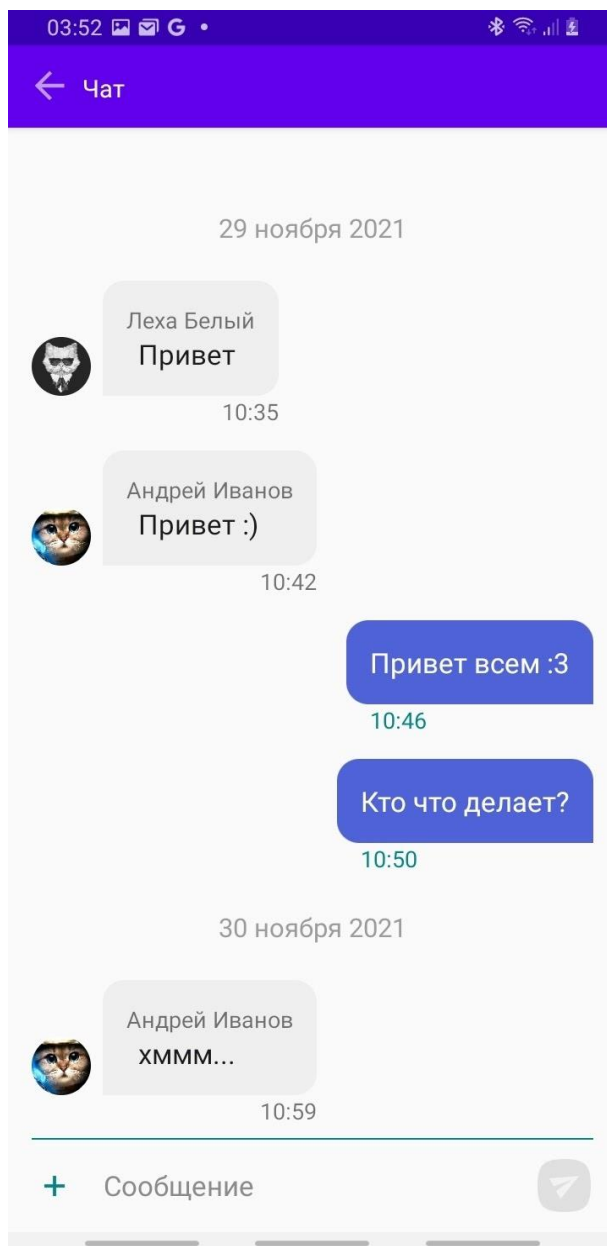


Рисунок 37 Чат

4.11 Персональное расписание(избранное)

Данный экран содержит все актуальные события, которые пользователь добавил к себе, при помощи кнопки добавить в персональное расписание. По функционалу совпадает с расписанием конференций, но только содержит события из разных мероприятий. Когда события становятся неактуальны они автоматически удаляются из персонального расписания.

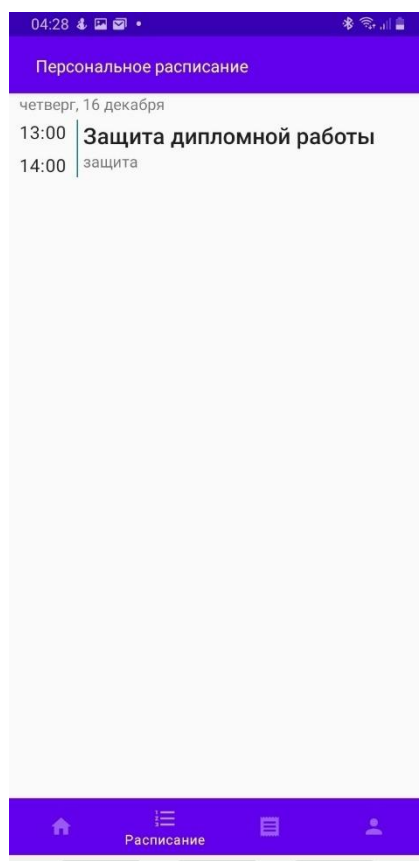


Рисунок 38 Персональное расписание

4.12 Создание конференций, событий и тарифов

Если пользователь является организатором у него появляется возможность создавать мероприятия из раздела списка конференций. Для того чтобы создать конференцию пользователю необходимо нажать на кнопку с изображением плюса снизу справа, после чего у него появится редактор конференций, где пользователь сможет настроить все необходимые параметры.

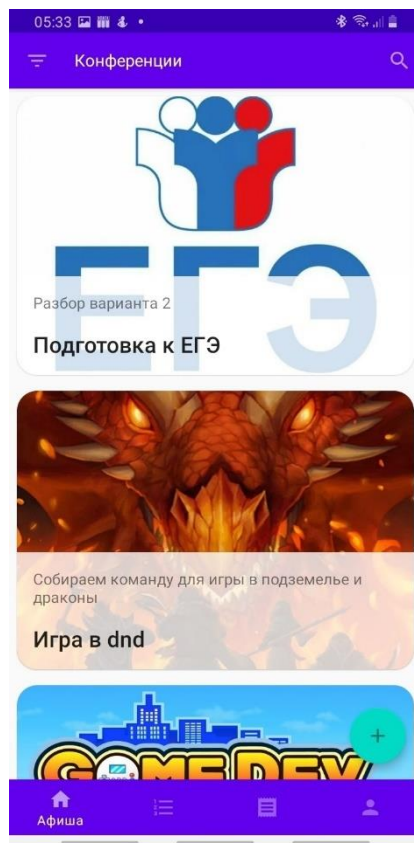


Рисунок 39 Афиша с кнопкой создания конференции

Как только пользователь нажмет кнопку создания конференции, откроется редактор конференций, где пользователь сможет при помощи полей и специальных интерфейсных элементов настроить, конференцию и сохранить ее на сервер.

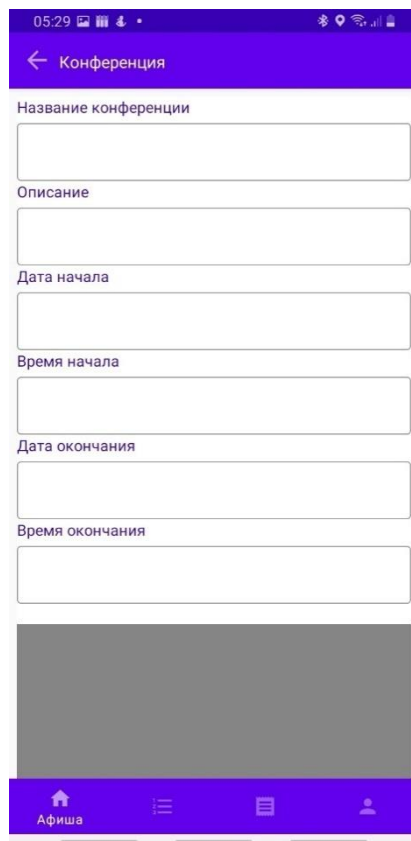


Рисунок 40 Редактор

Везде, где возможно данное приложение использует специализированные интерфейсные элементы для сбора параметров от пользователя. Если пользователь нажмет кнопку выбора даты, то для него откроется интерфейсный элемент позволяющий выбирать дату, при этом он не позволяет выбирать дату раньше текущего дня.

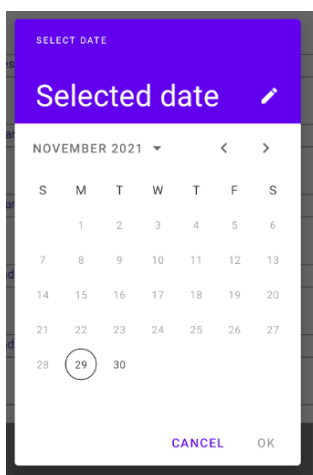


Рисунок 18 Инструмент получения даты

Также для сбора информации о времени используется специализированный элемент. На нем можно выбрать время после чего нажать кнопку подтверждения выбора или отмены.

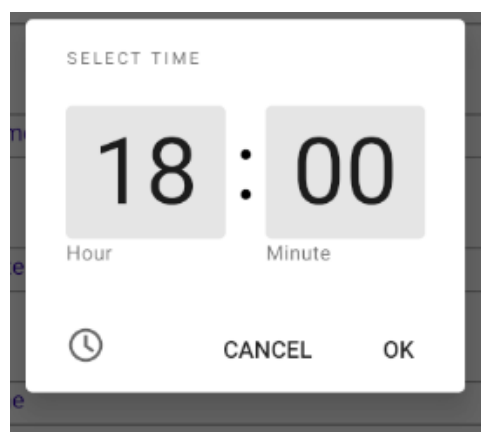


Рисунок 42 Инструмент получения времени

Для сбора информации о местоположении, также используется специализированный элемент. Он работает с API google и подсказывает адрес при этом автоматически отображая наиболее вероятный вариант на карте, переходя к нему при помощи анимации. При этом пользователь вообще может не использовать текстовое поле, а выбрать место прямо на карте, а результат отобразится в текстовом поле.

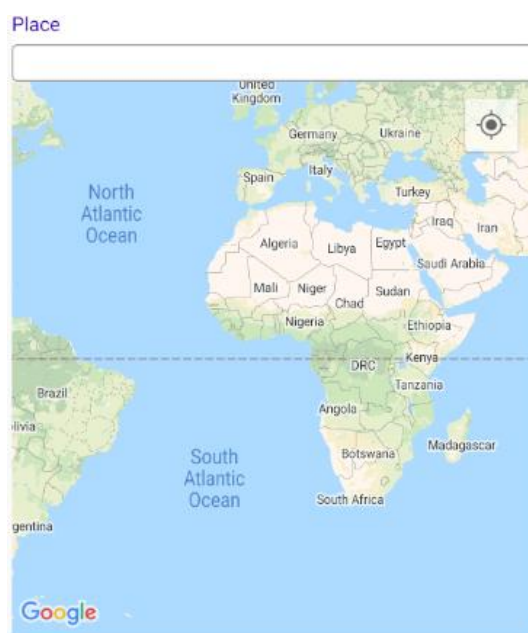


Рисунок 43 Инструмент получения адреса

Кроме того, приложение не дает создавать события имеющие некорректные поля. И при помощи самих элементов через подсказки информирует пользователя об этом.

A screenshot of a mobile application interface. At the top, the text "Дата окончания" (End date) is displayed in a light blue font. Below it is a white rectangular input field with a thin grey border. At the bottom of the input field, the text "Для продолжения задайте значение данного поля" (To continue, specify the value of this field) is written in a small red font.

Рисунок 44 Пример информирования пользователя

Выводы по главе.

В данной главе был описан интерфейс программного продукта, а также были раскрыты основные подходы к реализации интерфейсных элементов для операционной системы android.

Заключение

В рамках данной выпускной квалификационной работы было разработано приложение для организации мероприятий. Основными преимуществами данного программного продукта перед аналогами является возможность использования внутреннего чата для осуществления общения с другими участниками конференции, а также возможность создавать вложенные события для других событий.

Как результат работы над проектом был произведен анализ существующих платформ для организации публичных мероприятий и были выявлены ключевые функциональные особенности, которые способны сделать данный проект конкурентноспособным на площадке android-приложений Google Play.

В результате были изучены и применены множество фреймворков и подходов в разработке android-приложений и разработке серверной части с использованием Spring фреймворка. Как итог получился конкурентно-способный продукт, открытый для дальнейшей проработки и улучшений.

В будущем планируется целый ряд улучшений, нацеленных на повышение конкурентно-способности приложения в частности: доработка интерфейсных элементов, добавления возможности валидации билета по QR коду, создание других заранее заготовленных шаблонов для создания конференций, интеграция с системами оплаты, проработка системы модерации внутри конференций, добавление дополнительных возможностей для проведения мероприятий, таких как создание опросов внутри чата или проведение тестов.

Список использованных источников

- 1) Number of smartphone users Statista [Электронный ресурс] URL: <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/> (Дата обращения: 18.05.2021)
- 2) Activity Android [Электронный ресурс] URL: <https://developer.android.google.com/reference/android/app/Activity> (Дата обращения: 01.06.2021)
- 3) Fragment android [Электронный ресурс] URL: <https://developer.android.com/guide/fragments> (Дата обращения: 02.06.2021)
- 4) Navigation component [Электронный ресурс] URL: <https://developer.android.com/guide/navigation> (Дата обращения: 02.06.2021)
- 5) Clean architecture [Электронный ресурс] URL: <https://www.raywenderlich.com/3595916-clean-architecture-tutorial-for-android-getting-started> (Дата обращения: 05.06.2021)
- 6) MVVM(model-view-viewModel) [Электронный ресурс] URL: <https://developer.android.com/jetpack/guide> (Дата обращения: 07.06.2021)
- 7) Observer паттерн [Электронный ресурс] URL: <https://refactoring.guru/ru/design-patterns/observer> (Дата обращения: 07.06.2021)
- 8) sqlcipher [Электронный ресурс] URL: <https://www.zetetic.net/sqlcipher/> (Дата обращения: 14.06.2021)
- 9) WebSocket [Электронный ресурс] URL: <https://stfalcon.com/ru/blog/post/android-websocket> (Дата обращения: 20.06.2021)
- 10) STOMP (Streaming Text Oriented Message Protocol) [Электронный ресурс] URL: <https://stomp.github.io/websocket> (Дата обращения: 20.06.2021)
- 11) Repository [Электронный ресурс] URL: <https://medium.com/swlh/repository-pattern-in-android-c31d0268118c> (Дата обращения: 08.06.2021)
- 12) pgAdmin [Электронный ресурс] URL: <https://www.pgadmin.org/> (Дата обращения: 10.06.2021)
- 13) Room [Электронный ресурс] URL: <https://developer.android.com/jetpack/androidx/releases/room> (Дата обращения: 19.06.2021)
- 14) Navigation graph [Электронный ресурс] URL: <https://developer.android.com/guide/navigation/navigation-design-graph> (Дата обращения: 19.06.2021)