

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Сервис для покупки билетов и организации мероприятий “TicketEase”

Курсовой проект

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Зав. кафедрой _____ С.Д. Махортов, д.ф.- м.н., доцент ___. ___.20__

Обучающийся _____ А.А. Положенцев, 3 курс, д/о

Обучающийся _____ А.А. Бредихина, 3 курс, д/о

Обучающийся _____ А.В. Щербинина, 3 курс, д/о

Обучающийся _____ П.С. Мишакин, 3 курс, д/о

Руководитель _____ В.С. Тарасов, ст. преподаватель

Руководитель _____ К.В. Зенин, ассистент

Воронеж 2023

Содержание

Введение.....	5
1 Постановка задачи.....	7
1.1 Требования к разрабатываемой системе.....	7
1.1.1 Функциональные требования.....	7
1.1.2 Технические требования.....	7
1.2 Требования к интерфейсу.....	8
1.3 Задачи, решаемые в процессе разработки	8
2 Анализ предметной области	10
2.1 Глоссарий	10
2.2 Анализ рынка площадок по реализации билетов	11
2.3 Обзор аналогов	12
2.3.1 Ponominalu.....	12
2.3.2 Яндекс.Афиша	13
2.3.3 Kassir.ru	13
3 Реализация.....	15
3.1 Средства реализации.....	15
3.2 Входные-выходные данные (IDEF0).....	16
3.3 Диаграмма вариантов использования	16
3.4 Диаграмма состояний	18
3.4.1 Диаграмма состояний для мероприятия	18
3.4.2 Диаграмма состояний для пользователя.....	19
3.5 Диаграмма деятельности	20
3.5.1 Диаграмма деятельности для неавторизованного пользователя.....	20
3.5.2 Диаграмма деятельности для покупателя	21
3.5.3 Диаграмма деятельности для организатора.....	22
3.6 Диаграмма кооперации.....	23
3.6.1 Диаграмма кооперации для неавторизованного пользователя	23
3.6.2 Диаграмма кооперации для покупателя.....	23

3.6.3 Диаграмма кооперации для организатора.....	24
3.7 Диаграмма последовательности	24
3.7.1 Диаграмма последовательности для неавторизованного пользователя.	24
3.7.2 Диаграмма последовательности для покупателя	26
3.7.3 Диаграмма последовательности для организатора.....	27
3.8 Диаграмма развертывания	27
3.9 Архитектура приложения.....	28
3.10 Реализация серверной части приложения	29
3.10.1 Схема базы данных.....	29
3.11 Диаграмма классов.....	30
3.11.1 Диаграмма классов сущностей.....	30
3.11.2 Диаграмма классов интерфейсов транзакций	31
3.11.3 Диаграмма классов реализаций транзакций.....	32
3.11.4 Диаграмма вспомогательных классов для доступа к базе данных	32
3.11.5 Диаграмма классов маршрутов.....	33
3.11.6 Диаграммы классов для хэширования	33
3.11.7 Диаграмма классов токенов	35
3.10.8 Диаграмма служебных классов	35
3.11.9 Слой доступа к данным (data layer).....	38
3.11.10 Слой транзакций.....	39
3.11.11 Слой маршрутизаторов.....	41
3.12 Реализация клиентской части приложения	42
3.12.1 Навигация по приложению	42
3.12.2 Макеты приветственных экранов.....	42
3.12.3 Сценарий для неавторизованного пользователя.....	44
3.13.4 Регистрация покупателя	48
3.12.5 Авторизация покупателя	50
3.12.6 Сценарий для авторизованного покупателя.....	52
3.12.7 Регистрация нового организатора	56
3.12.8 Авторизация организатора.....	59

3.12.9 Создание мероприятия.....	61
3.12.10 Личный кабинет организатора.....	65
3.14 Календарный план.....	70
Заключение	73
Список использованных источников	74

Введение

Невозможно недооценить роль досуга в современном мире. В первую очередь, он позволяет снизить напряжение, вызванное бесчисленными стрессовыми ситуациями, начиная от домашних ссор и заканчивая рутинными задачами на работе.

В последнее время сфера досуга испытывает бурный рост. Однако большинство из её составляющих несет исключительно развлекательный характер, но всё же существует обособленная зона, которая обеспечивает не только чувство наслаждения от посещения, но и позволяет обогатить внутренний мир. Конечно, речь идёт о культурно-массовых мероприятиях. Это и театральные постановки, и картинные выставки, и многое другое. А если посмотреть на культурно-массовые события с другой стороны, с точки зрения художественных коллективов, они позволяют объединить огромное число актёров, сценаристов, композиторов и режиссеров. Таким образом, вклад этих светских мероприятий в развитие искусства огромен. У многих из них многовековая история, так годом рождения театра считается 543 г. до н.э.

Переживая постоянную трансформацию, отвечая требованиям современности, культурно-массовые мероприятия занимают огромную нишу в мировом досуге. В частности, в России в 2021 году театры, подведомственные Минкульту, посетило 25 млн зрителей. Число государственных театров увеличилось с 662 в 2018 году до 679 в 2021 году, а общее число театров превышает 1750. Данная статистика говорит о том, что сфера подобных мероприятий испытывает повышенный интерес со стороны жителей нашей страны.

С развитием компьютерных технологий удобство посещения культурно-массовых мероприятий только повышается. Практически каждый человек имеет под рукой смартфон под управлением ОС Android. Согласно статистике мая 2022 года, данная система установлена на 72.2% носимых устройств, что безусловно говорит о популярности платформы. Учитывая всё

вышесказанное, разработчики мобильных приложений оперативно реагируют на потребности рынка, создавая новые приложения по реализации билетов на культурно-массовые мероприятия.

Целью данной работы является разработка мобильного приложения по реализации билетов на мероприятия малых художественных коллективов и картинных выставок под Android с использованием современного языка программирования Kotlin. Особенностью данного приложения будет совмещение ролей покупателя и организатора в одном приложении. Таким образом, организатор коллектива сможет создать мероприятие на предложенной площадке, а покупатель купить билет на данное мероприятие.

1 Постановка задачи

Целью данного курсового проекта является разработка мобильного приложения по реализации билетов на мероприятия малых художественных коллективов и картинных выставок – приложения “TicketEase”. К разрабатываемому мобильному приложению выдвинуты следующие требования:

- Возможность создания руководителем художественного коллектива мероприятия с указанием места проведения, стоимости, жанра.
- Возможность создания руководителем нескольких мероприятий;
- Возможность руководителем редактирования и отмены мероприятий;
- Возможность просмотра созданных мероприятий;
- Возможность покупки билетов на созданные мероприятия.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому приложению выдвигаются следующие требования:

- Возможность регистрации новых учетных записей для покупателей и руководителей художественных коллективов;
- Возможность авторизации зарегистрированных покупателей и руководителей;
- Вывод ленты событий: для авторизованного покупателя – на основе ранее купленных билетов, для неавторизованного – по дате проведения;
- Осуществление редактирования пользовательских данных;

1.1.2 Технические требования

Приложение должно обеспечивать:

- Авторизацию пользователей посредством персонального логина и пароля, которые хранятся и идентифицируются внешней (по отношению к приложению) системой авторизации;
- Удаление содержимого корзины при удалении учетной записи покупателя;
- Удаление созданных мероприятий при удалении учетной записи руководителя.

1.2 Требования к интерфейсу

Интерфейс должен быть выполнен в единой для всех экранов цветовой гамме, едином стиле. Все надписи должны быть легко читаемы, все элементы управления должны быть выполнены в едином стиле, размере, должны выделяться на фоне содержимого экранов. Интерфейс должен корректно отображаться при изменении размера экрана. Интерфейс должен поддерживать портретную ориентацию экрана.

Интерфейс должен содержать необходимую для пользователя информацию: информацию о самом покупателе, информацию об избранном, корзине, купленных билетах, информацию о коллективе, о созданных мероприятиях. Информация должна находиться в тех местах приложения, где она будет актуальна.

1.3 Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

- Анализ предметной области;
- Анализ аналогов;
- Постановка задачи;
- Разработка макетов интерфейса;
- Определение используемой платформы;
- Построение use-case диаграмм;
- Создание доски Trello и репозитория GitHub;
- Построение UML диаграмм;

- Написание технического задания;
- Реализация слоя доступа к БД;
- Реализация интерфейса;
- Подключение swagger;
- Размещение backend-части и БД на облаке;
- Реализация модуля авторизации и регистрации;
- Описание процесса разработки и результата.

2 Анализ предметной области

2.1 Глоссарий

- **Сервер, серверная часть** – компьютер, обслуживающий другие устройства (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач;
- **Клиент, клиентская сторона** – в данном проекте, мобильное устройство с установленным на него приложением, предоставляет возможности пользователю взаимодействовать со всей системой;
- **Front-end** – клиентская часть приложения. Отвечает за получение информации с программно-аппаратной части и отображение ее на устройстве пользователя. В проекте, это само android приложение;
- **Back-end** – программно-аппаратная часть приложения. Отвечает за функционирование внутренней части приложения;
- **GitHub** – веб-сервис для хостинга ИТ-проектов и их совместной разработки;
- **Авторизованный покупатель** – авторизованный в системе человек, пользующийся покупательским функционалом приложения;
- **Наблюдатель, неавторизованный покупатель** – человек, не имеющий учетной записи, имеет ограниченный доступ к функционалу приложения;
- **Организатор** – авторизованный человек, пользующийся организаторским функционалом приложения;
- **Presentation layer, слой представления** – часть приложения, которая разворачивается на клиенте, в частности смартфоне пользователя;
- **Data layer, слой доступа к данным** – часть приложения, относящаяся к серверной части;
- **Clean architecture** – парадигма проектирования приложений, предложенная Робертом Мартином в 2012 году;

- **MVVM (Model-View-ViewModel)** – паттерн проектирования клиентской стороны. View содержит структурное определение того, что пользователи получат на экранах. ViewModel отвечает за управление ссылками данных. Model отвечает за бизнес-данные;
- **«Material Design»** – дизайн-система для создания интерфейсов программного обеспечения и приложений, разработанная компанией Google;
- **REST API** – стиль архитектуры программного обеспечения для построения масштабируемых веб-приложений;
- **СБП** – система быстрых платежей. Сервис, с помощью которого можно совершать межбанковские переводы по номеру мобильного телефона без комиссии.

2.2 Анализ рынка площадок по реализации билетов

Как было отмечено ранее, сфера культурно-массовых мероприятий вызывает неподдельный интерес у населения РФ. Согласно исследованию, ВЦИОМ с 1992 по 2022 год, аудитория театров выросла на 2-5%, а музеев и выставок в 1.5 раза. Таким образом, виды досуга, связанные с эмоциональным восприятием, живыми впечатлениями, такие как просмотр спектаклей и посещение выставок, не только не утратили своей популярности, но и нарастили её.

Российский рынок отличается быстрой цифровизацией всех отраслей экономики. В первом полугодии 2022-го года интернет-продажи непродовольственных товаров в России увеличилась на 51.5%, не последнюю роль в этом процессе сыграла пандемия COVID-19. Интернет-маркетинг самая конкурентная среда среди всех остальных. Одни площадки, своевременно отреагировав на изменения особенностей покупательского спроса, активно продолжают развиваться, такие как «Яндекс.Афиша» или «Kassir.ru», речь о которых пойдёт позже в обзоре аналогов, другие – наоборот, неправильно построив бизнес-модель, прекращают своё существование или осуществляют слияние с более крупными площадками.

2.3 Обзор аналогов

Для выхода на рынок разрабатываемый проект должен иметь конкурентные преимущества, которые позволяют собрать определённую аудиторию пользователей. Проведём обзор аналогов, который позволит выявить сильные и слабые стороны существующих решений, а также позволит сосредоточиться на функционале разрабатываемого приложения с учётом требований пользователей.

2.3.1 Ponominalu

«Ponominalu» - одна из самых популярных площадок для реализации билетов на различные мероприятия. Принадлежит компании «МТС». Интерфейс сайта представлен на рисунке 1.

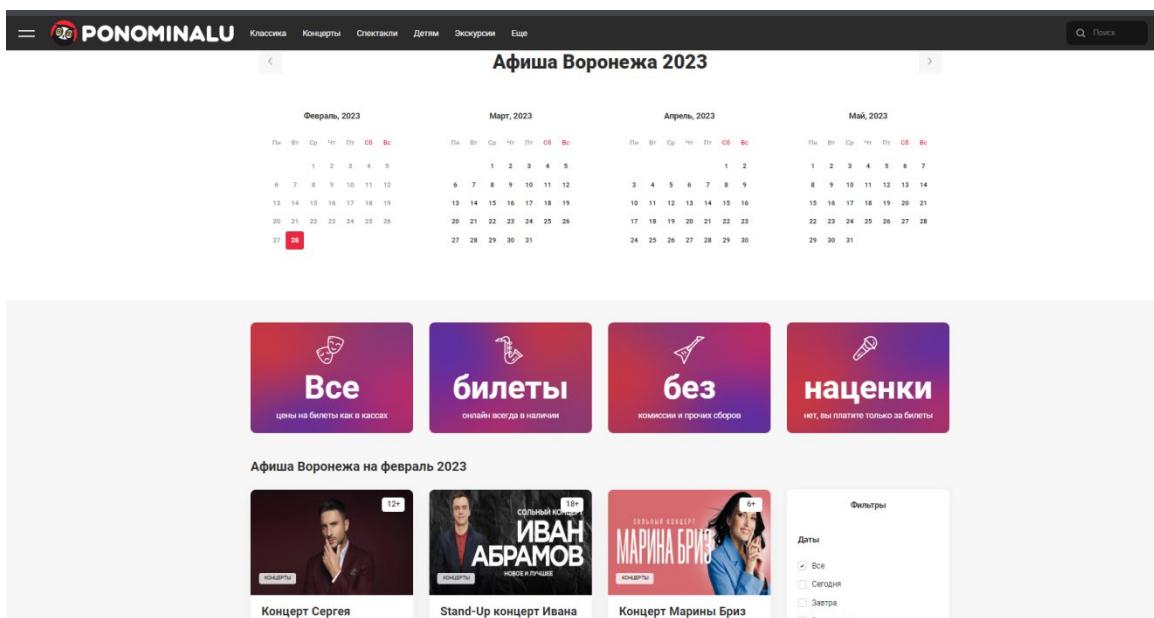


Рисунок 1 - Интерфейс сайта Ponominalu.ru

Функциональные возможности сайта:

- Заказ билетов с возможностью выкупа онлайн или ближайшую кассу;
- Наличие мобильного приложения под IOS.

Недостатки:

- Отсутствие мобильного приложения под ОС Android;
- Отсутствие личного кабинета с историей заказов.

2.3.2 Яндекс.Афиша

«Яндекс.Афиша» - также одна из популярных площадок реализации билетов от известного поискового гиганта – Яндекса. Интерфейс сайта представлен на рисунке 2.

Афиша событий в Воронеже >

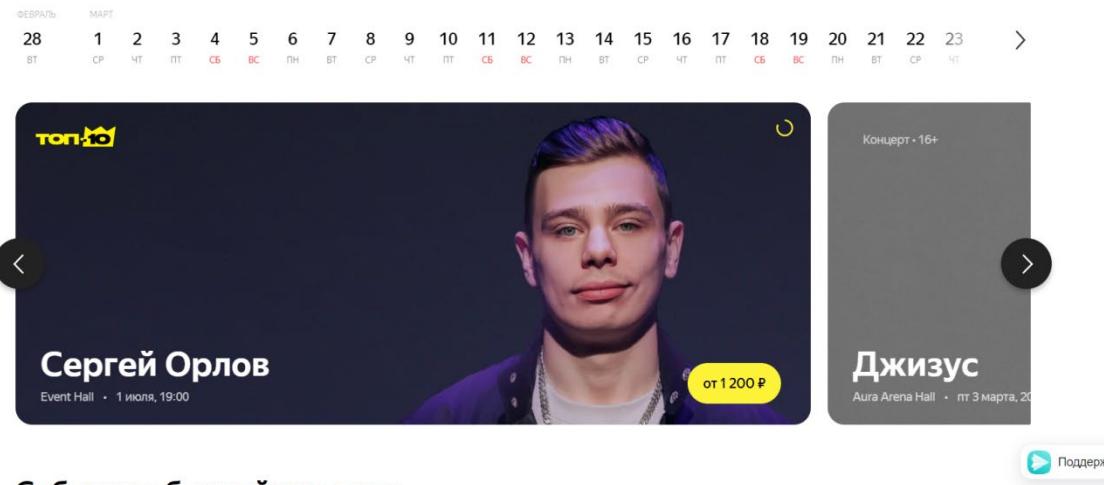


Рисунок 2 - Интерфейс сайта «Яндекс.Афиша»

Функциональные возможности сайта:

- Заказ билетов с возможностью выкупа онлайн;
- Наличие личного кабинета с историей заказов;
- Возможность оформить подарочный сертификат;
- Наличие мобильного приложения под IOS.

Недостатки:

- Отсутствие мобильного приложения под ОС Android;

2.3.3 Kassir.ru

«Kassir.ru» - самая известная площадка по продаже билетов в РФ, недавно совершила слияние с сервисом «Parter.ru». Интерфейс сайта представлен на рисунке 3.

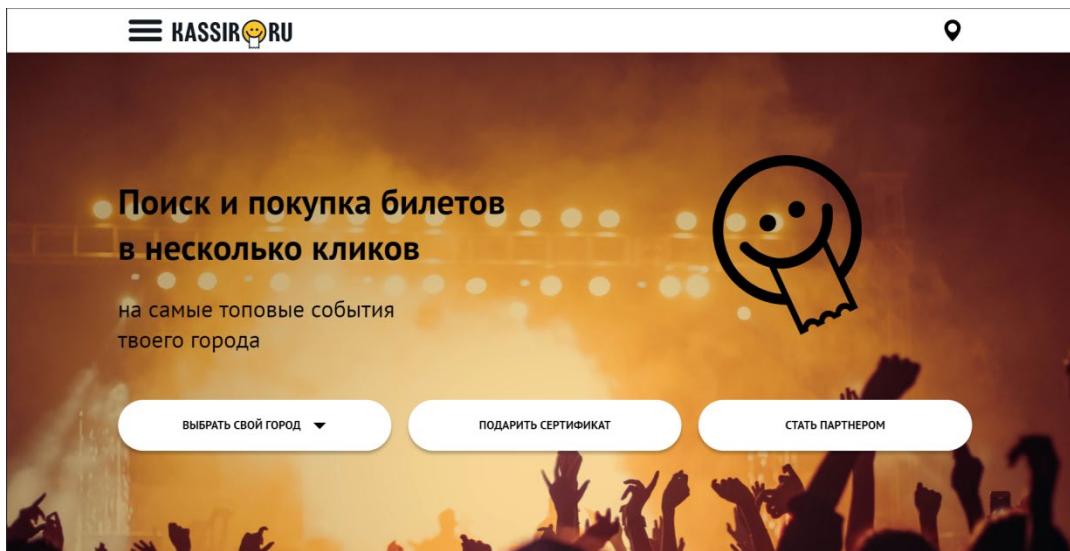


Рисунок 3 - Интерфейс сайта «Kassir.ru»

Функциональные возможности сайта:

- Заказ билетов с возможностью выкупа онлайн;
- Наличие личного кабинета с историей заказов;
- Возможность оформить подарочный сертификат;
- Наличие мобильного приложения под IOS и Android.

Недостатки:

- Не выявлены функциональные недостатки, однако отметим присутствие сервисного сбора в размере 10%.

3 Реализация

3.1 Средства реализации

Для реализации серверной части были выбраны следующие технологии:

- Язык программирования Kotlin (компиляция проводится на JVM (Java 17));
- Фреймворк Ktor;
- СУБД PostgreSQL;
- Библиотека Jackson;
- Фреймворк Kotlin Exposed;
- Система автоматической сборки Gradle.

Для реализации клиентской части были выбраны следующие технологии:

- Фреймворк Jetpack Compose 1.3.3;
- Android sdk;

Стек технологий был выбран ввиду современных тенденций в android-разработке с учётом скорости разработки кода, а также длительной поддержки кода. Согласно опросу, проведённому компанией JetBrains, удовлетворённость от использования языка Kotlin составляет 86%.

3.2 Входные-выходные данные (IDEF0)

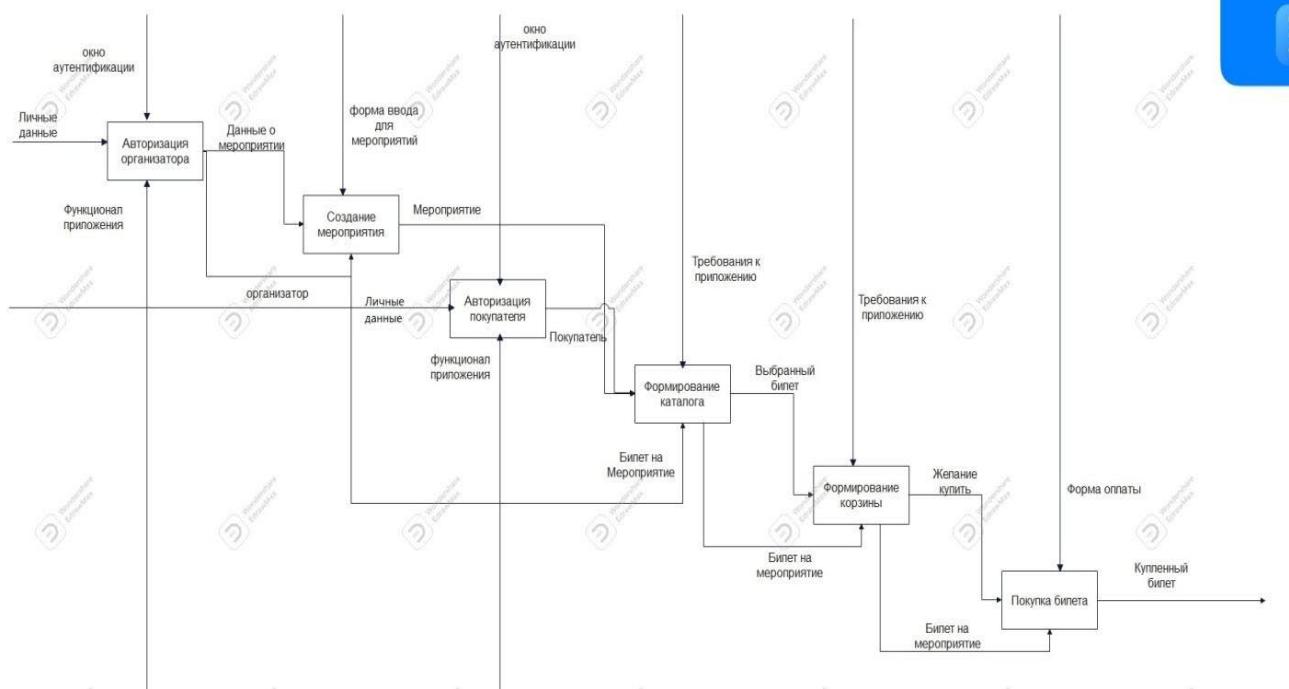


Рисунок 4 - Диаграмма IDEF0

3.3 Диаграмма вариантов использования

Неавторизованным считается пользователь, который решил пользоваться приложением без входа в учетную запись. Ему доступен не весь функционал приложения.

Авторизованный пользователь (покупатель) – пользователь, который имеет доступ ко всему функционалу для покупки билетов, но не может организовать мероприятия.

Авторизованный организатор – пользователь, прошедший особую регистрацию, обладает особыми правами организации мероприятий.

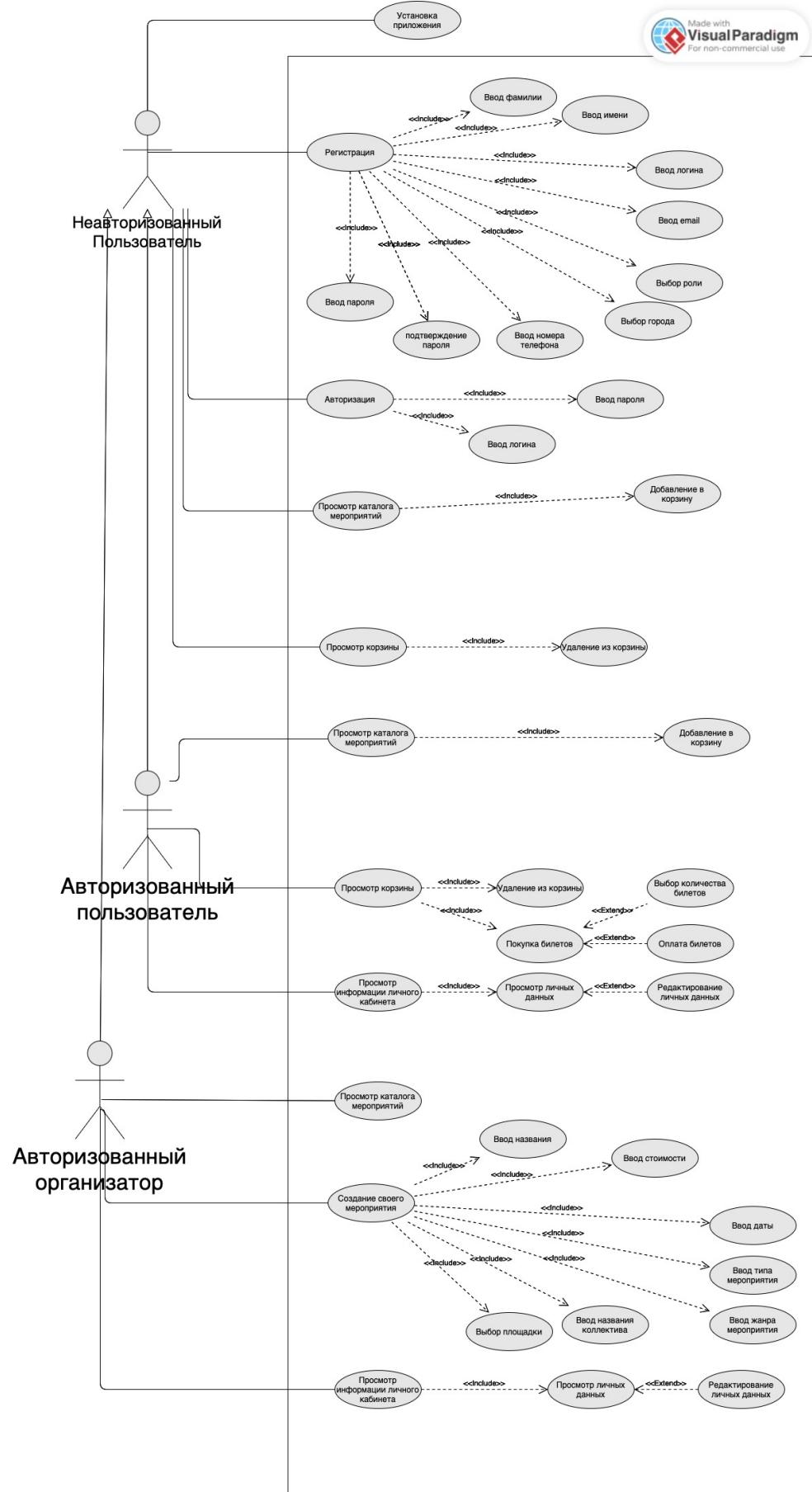


Рисунок 5 - Диаграмма вариантов использования

3.4 Диаграмма состояний

3.4.1 Диаграмма состояний для мероприятия

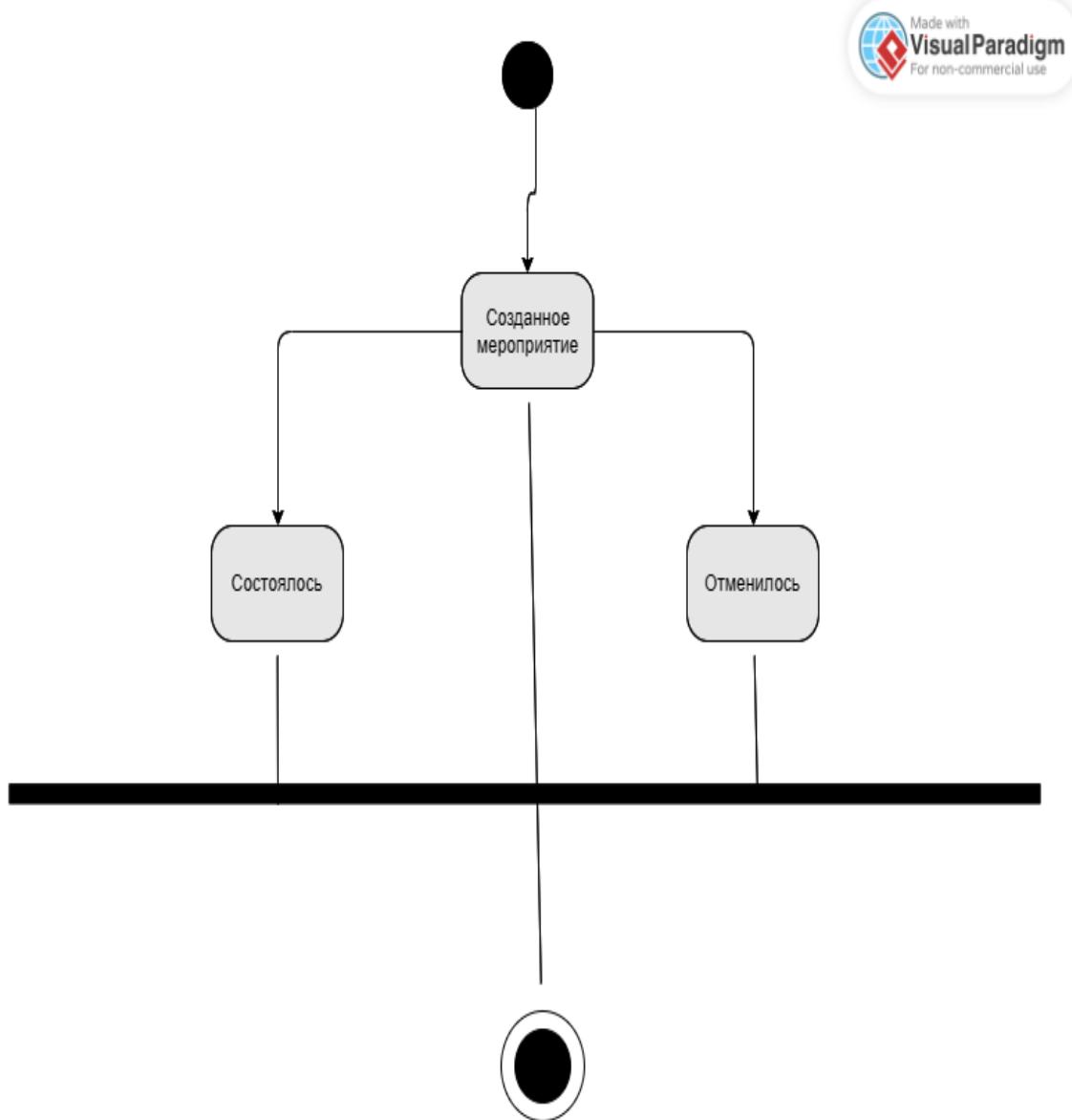


Рисунок 6 - Диаграмма состояний для мероприятия

3.4.2 Диаграмма состояний для пользователя

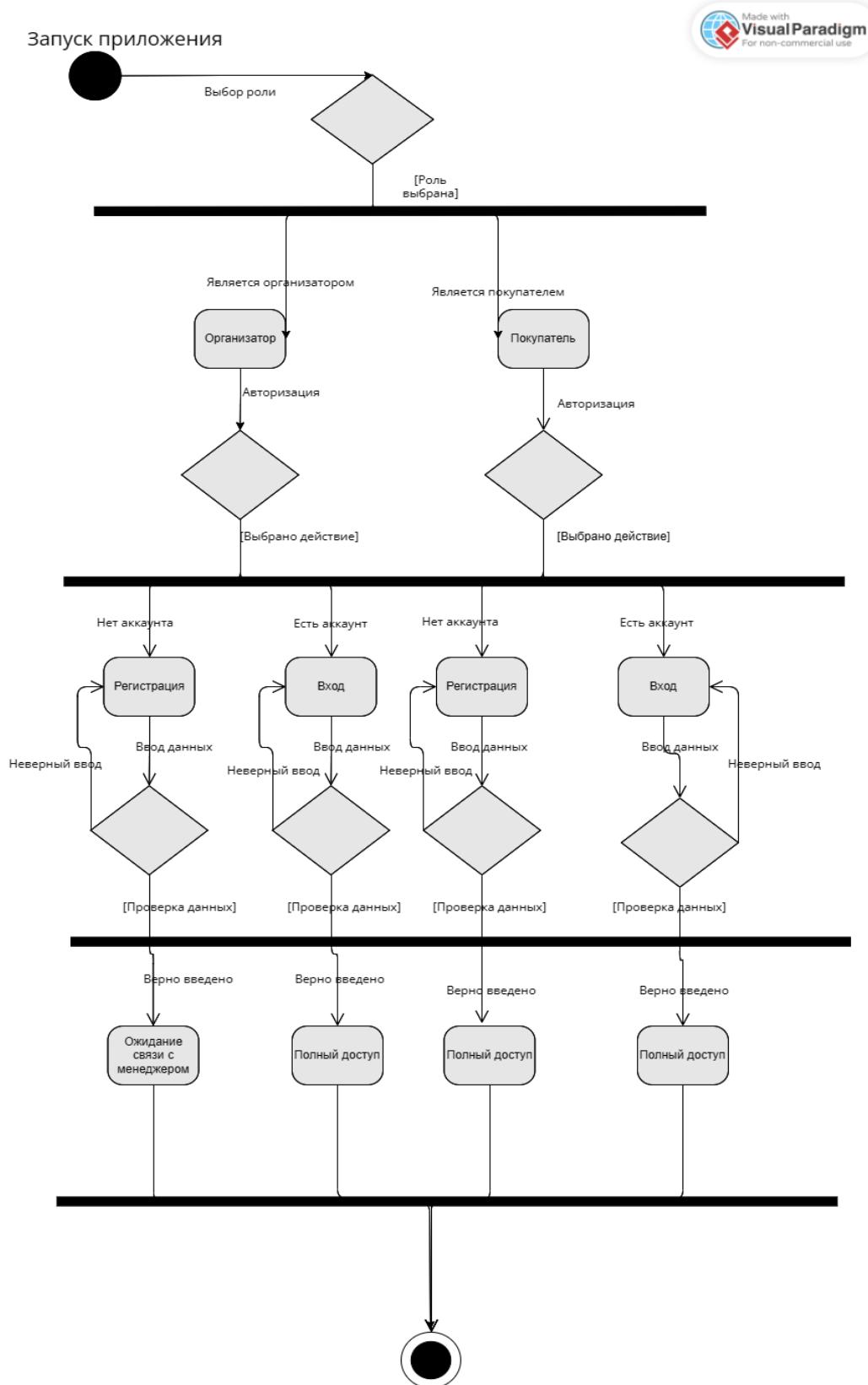


Рисунок 7 - Диаграмма состояний для пользователя

3.5 Диаграмма деятельности

3.5.1 Диаграмма деятельности для неавторизованного пользователя

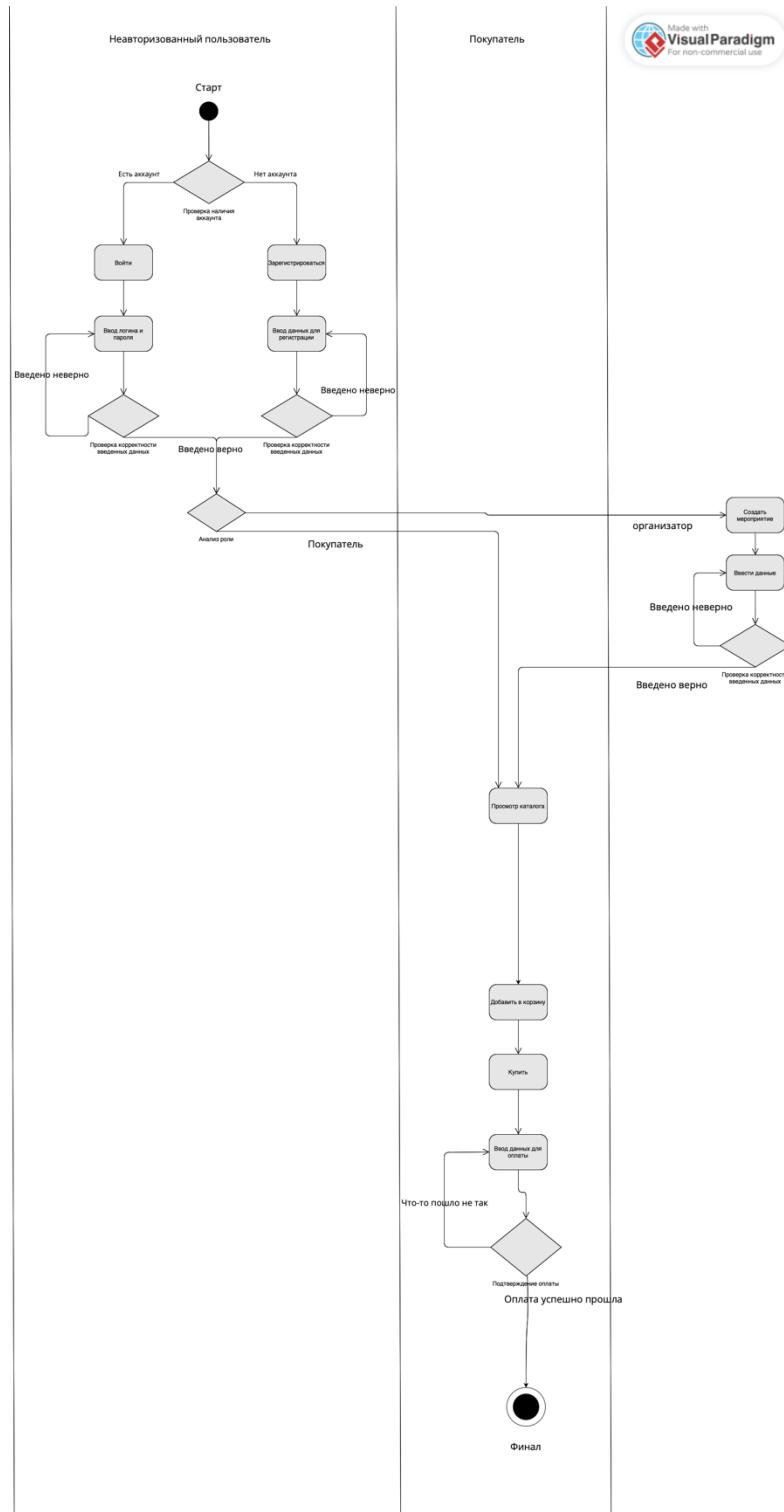


Рисунок 8 - Диаграмма действий для неавторизованного пользователя

3.5.2 Диаграмма деятельности для покупателя

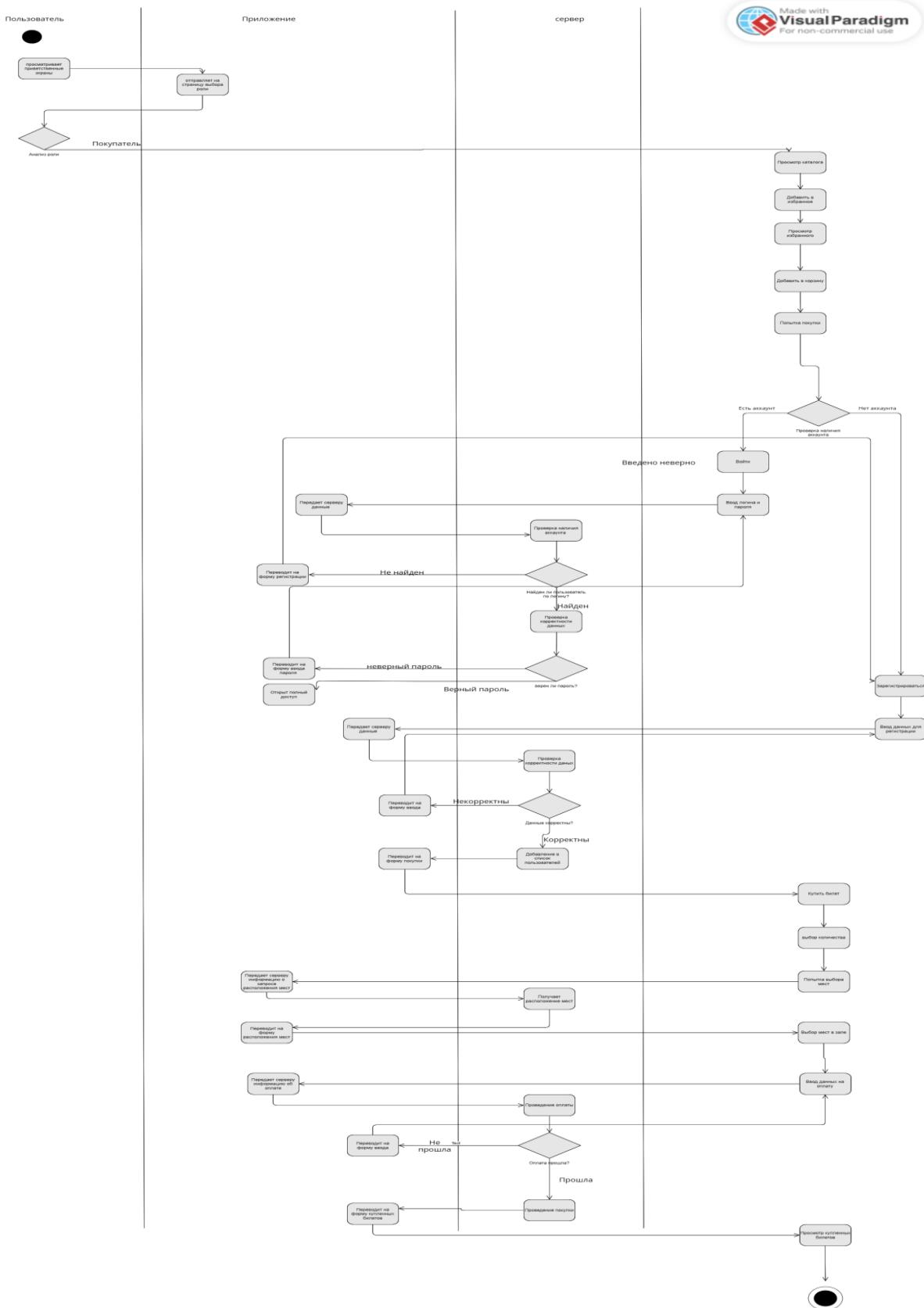


Рисунок 9 - Диаграмма действий для покупателя

3.5.3 Диаграмма деятельности для организатора

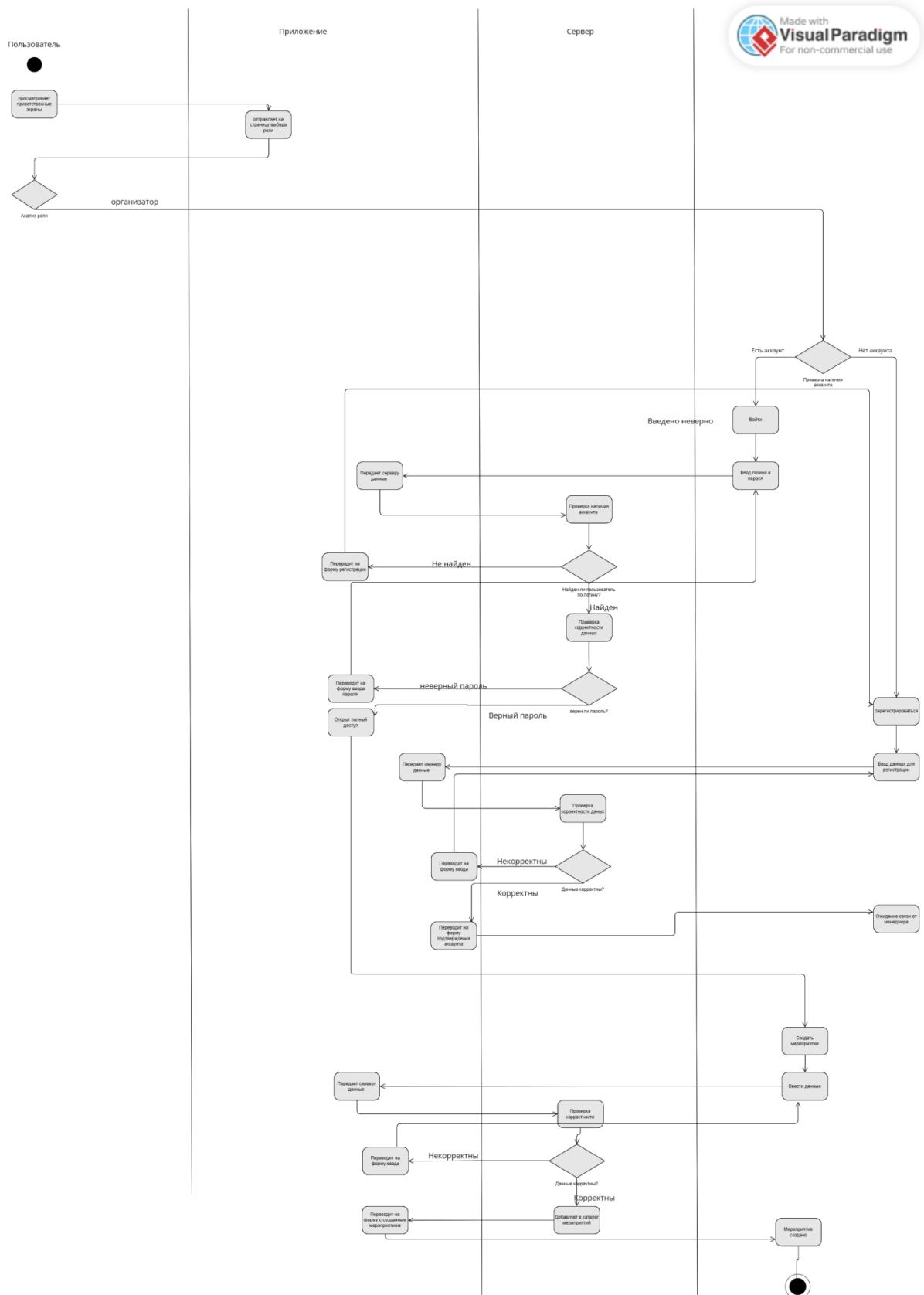


Рисунок 10 - Диаграмма действий для организатора

3.6 Диаграмма кооперации

3.6.1 Диаграмма кооперации для неавторизованного пользователя

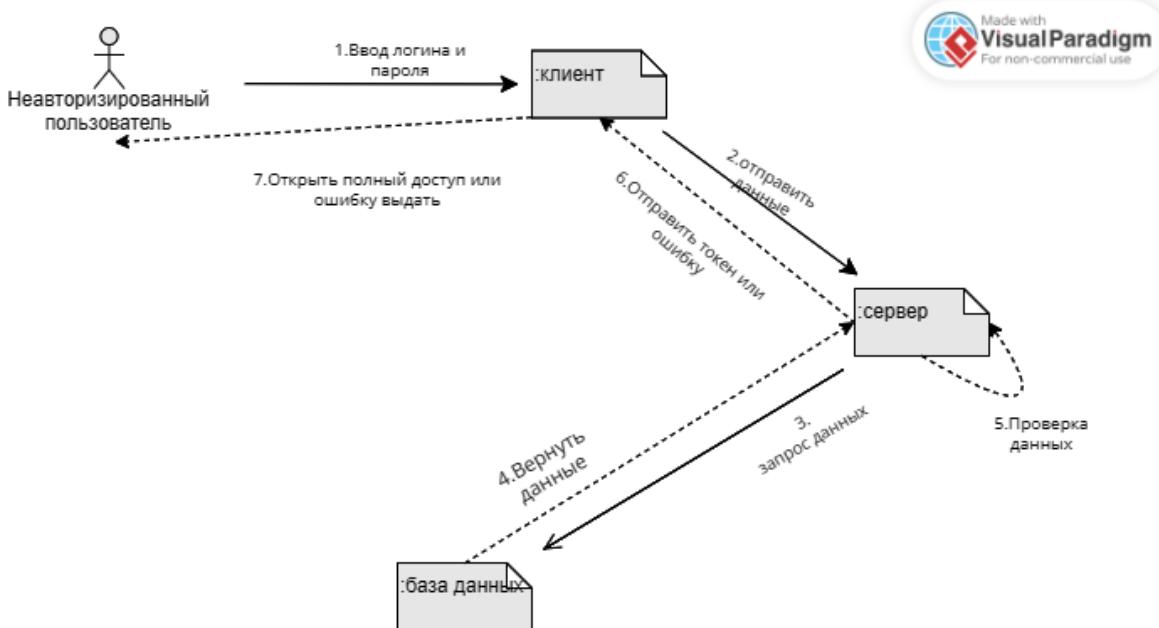


Рисунок 11 - Диаграмма кооперации для неавторизованного пользователя

3.6.2 Диаграмма кооперации для покупателя

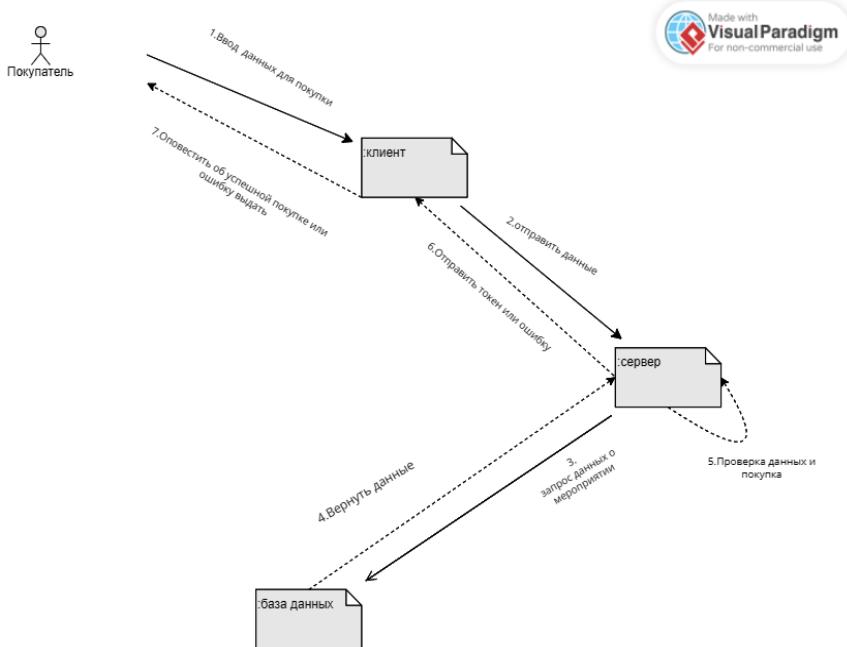


Рисунок 12 - Диаграмма кооперации для покупателя

3.6.3 Диаграмма кооперации для организатора

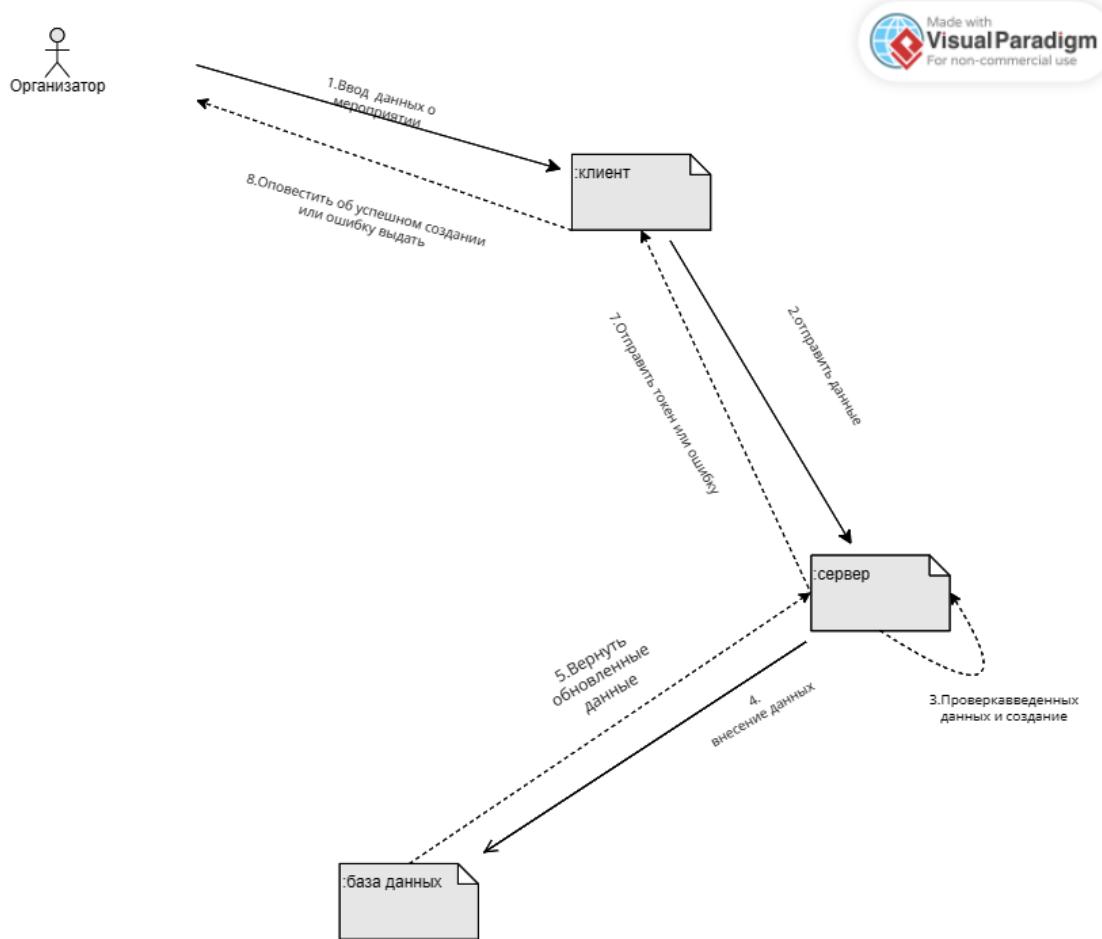


Рисунок 13 - Диаграмма кооперации для организатора

3.7 Диаграмма последовательности

3.7.1 Диаграмма последовательности для неавторизованного пользователя

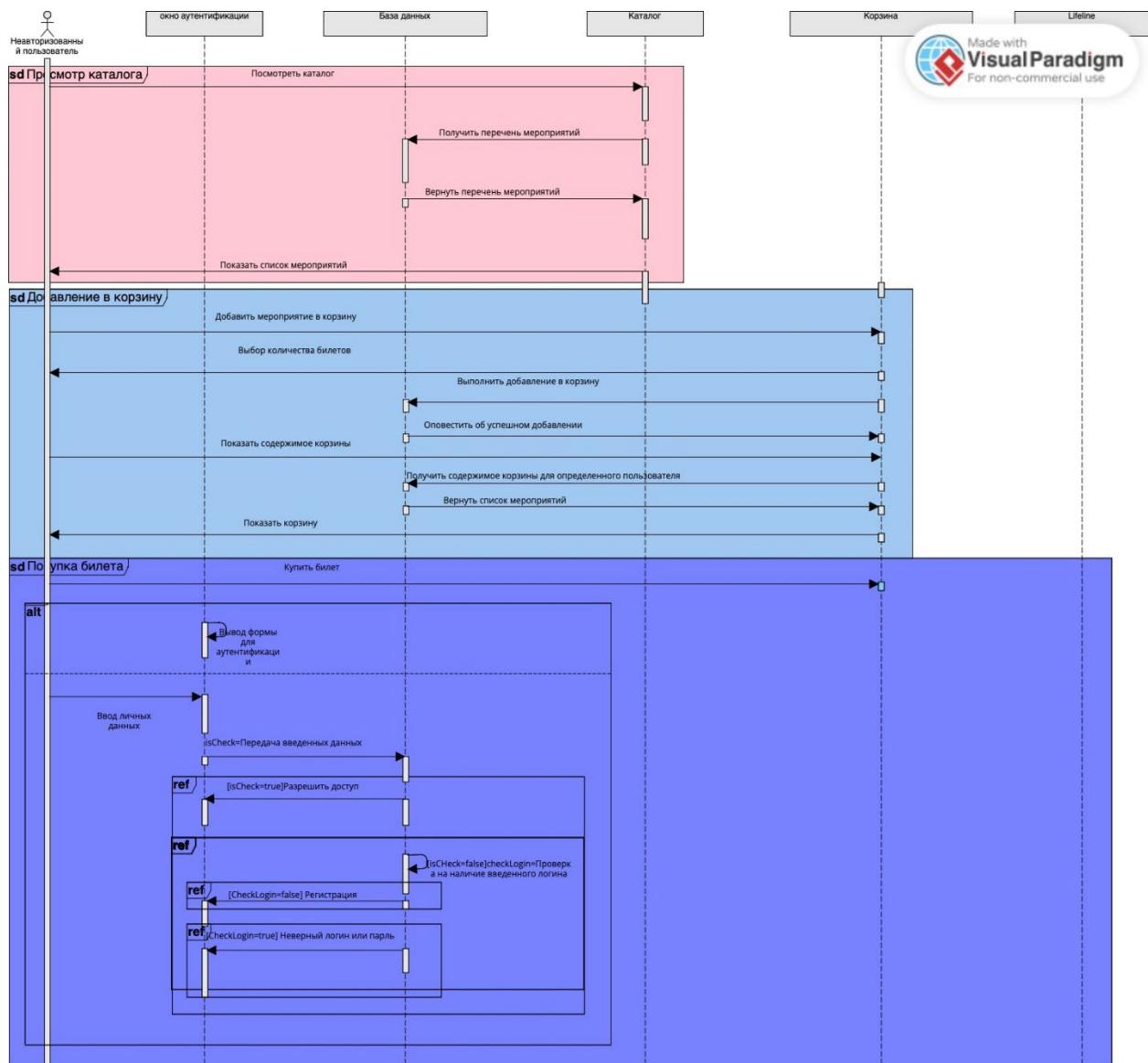


Рисунок 14 - Диаграмма последовательности для неавторизованного пользователя

3.7.2 Диаграмма последовательности для покупателя

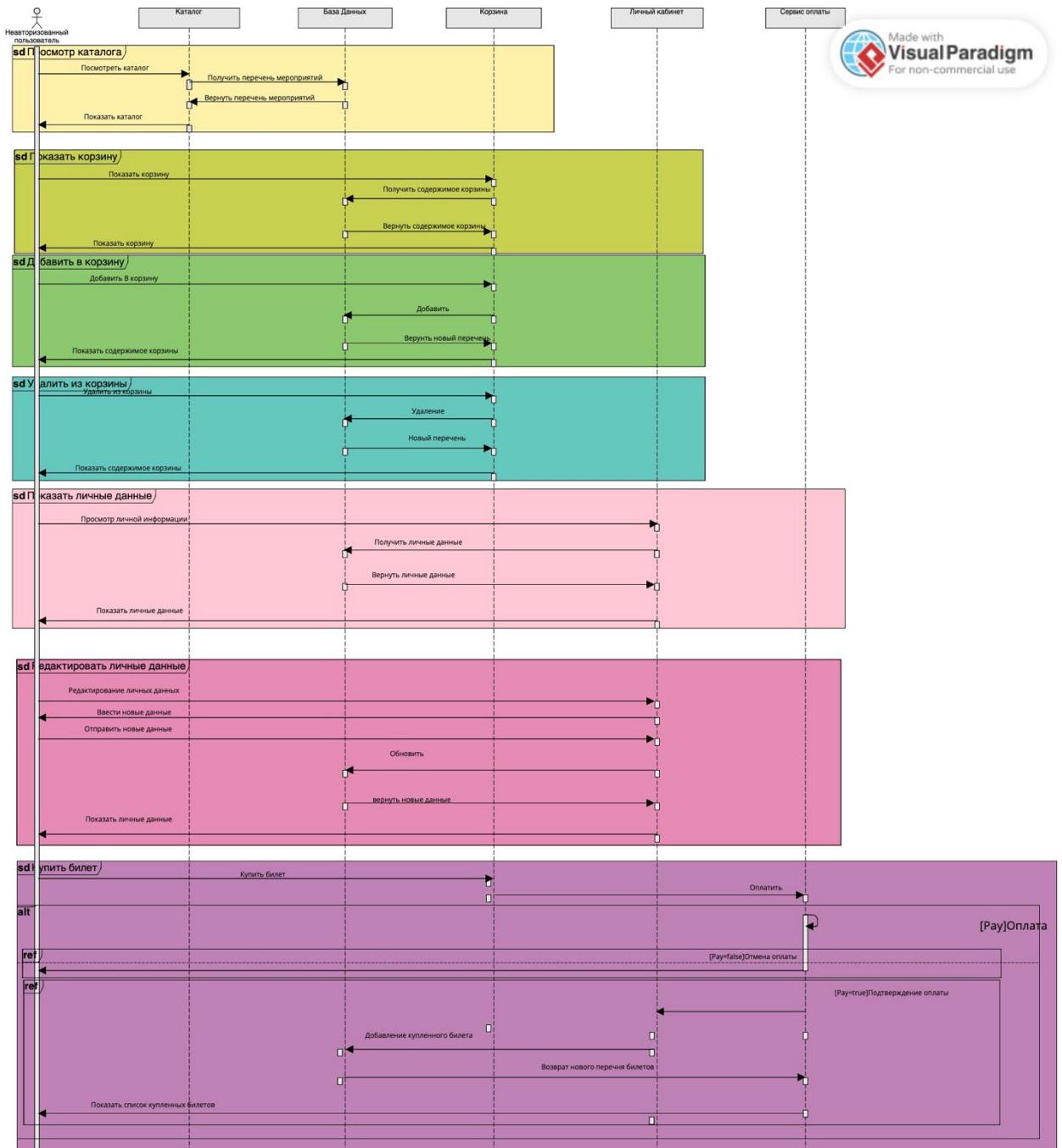


Рисунок 15 - Диаграмма последовательности для покупателя

3.7.3 Диаграмма последовательности для организатора

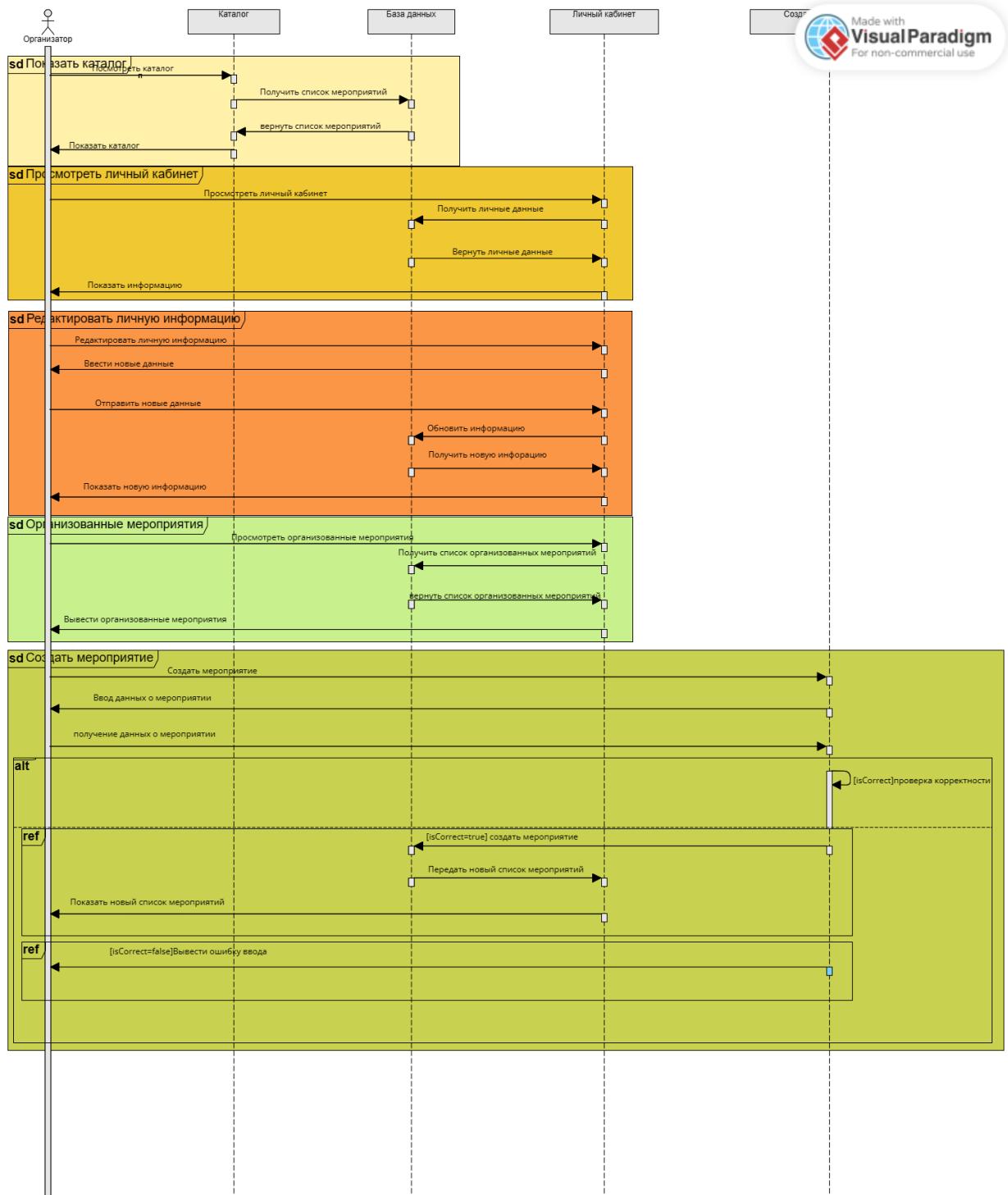


Рисунок 16 - Диаграмма последовательности для организатора

3.8 Диаграмма развертывания

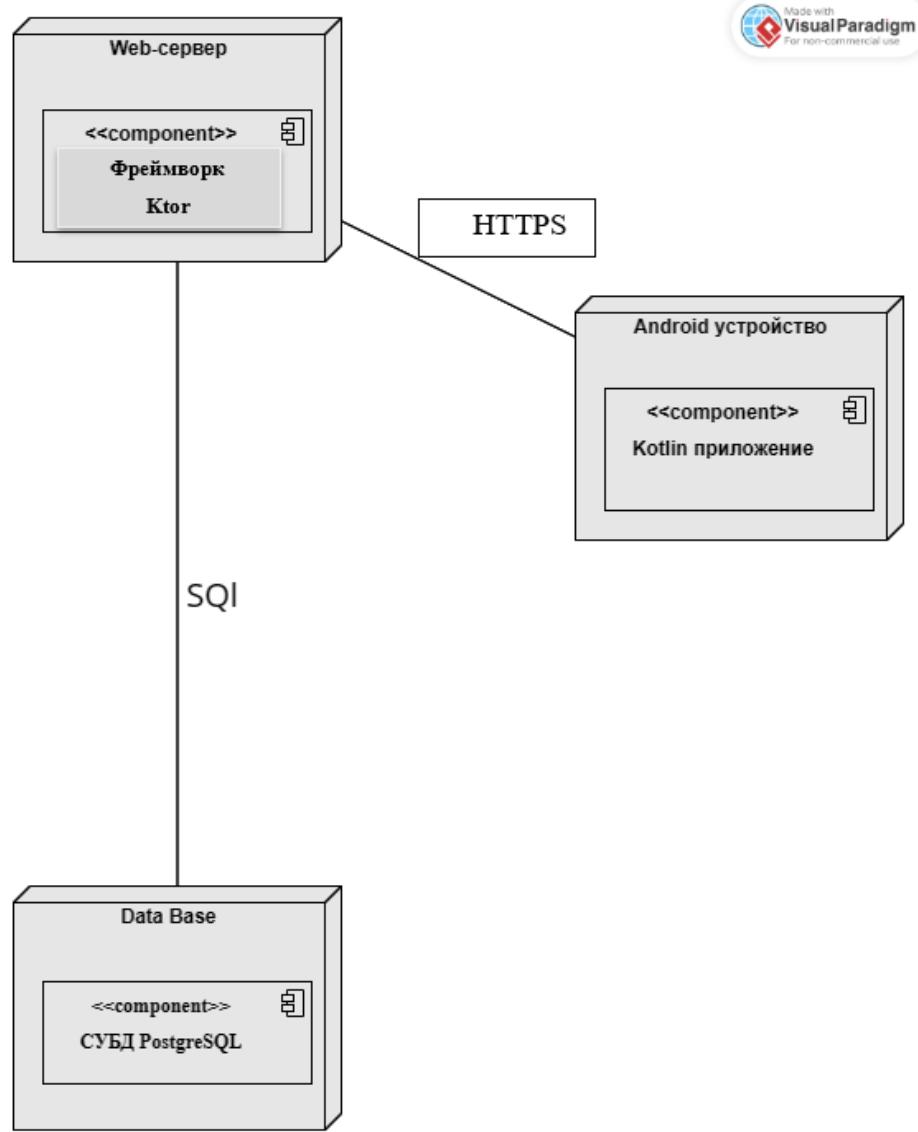


Рисунок 17 - Диаграмма развертывания

3.9 Архитектура приложения

Созданное приложение соответствует шаблону Клиент-Серверного приложения с применением «Clean architecture» и разделением на два слоя: слой представления (presentation layer) с использованием паттерна MVVM – front-end, слой доступа к данным (data layer) – back-end и связью между ними по средству REST API. Схематичное изображение архитектуры проекта продемонстрировано на рисунке 18.

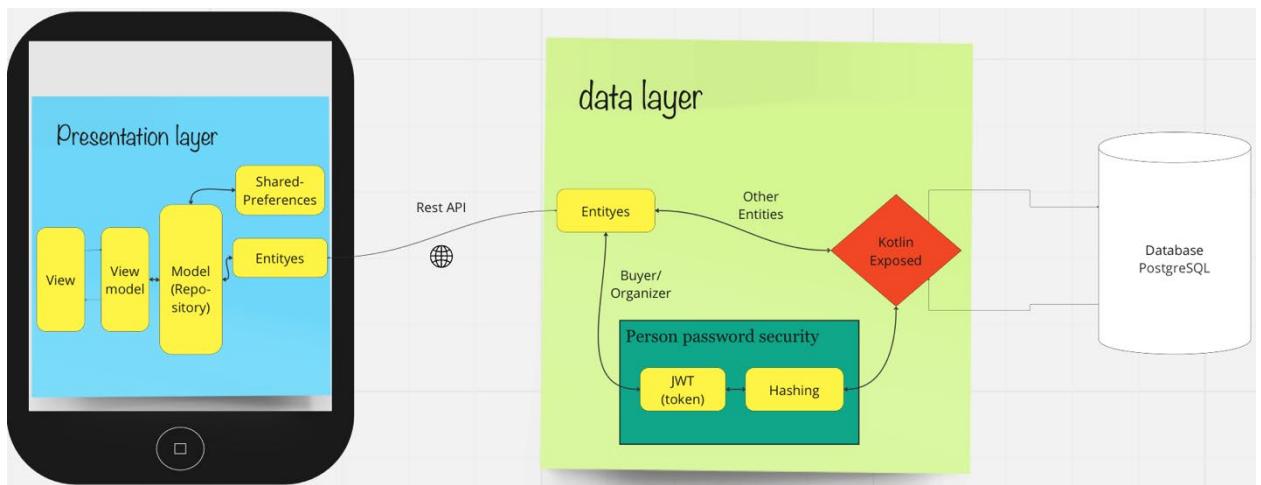


Рисунок 18 - Архитектура проекта

3.10 Реализация серверной части приложения

3.10.1 Схема базы данных

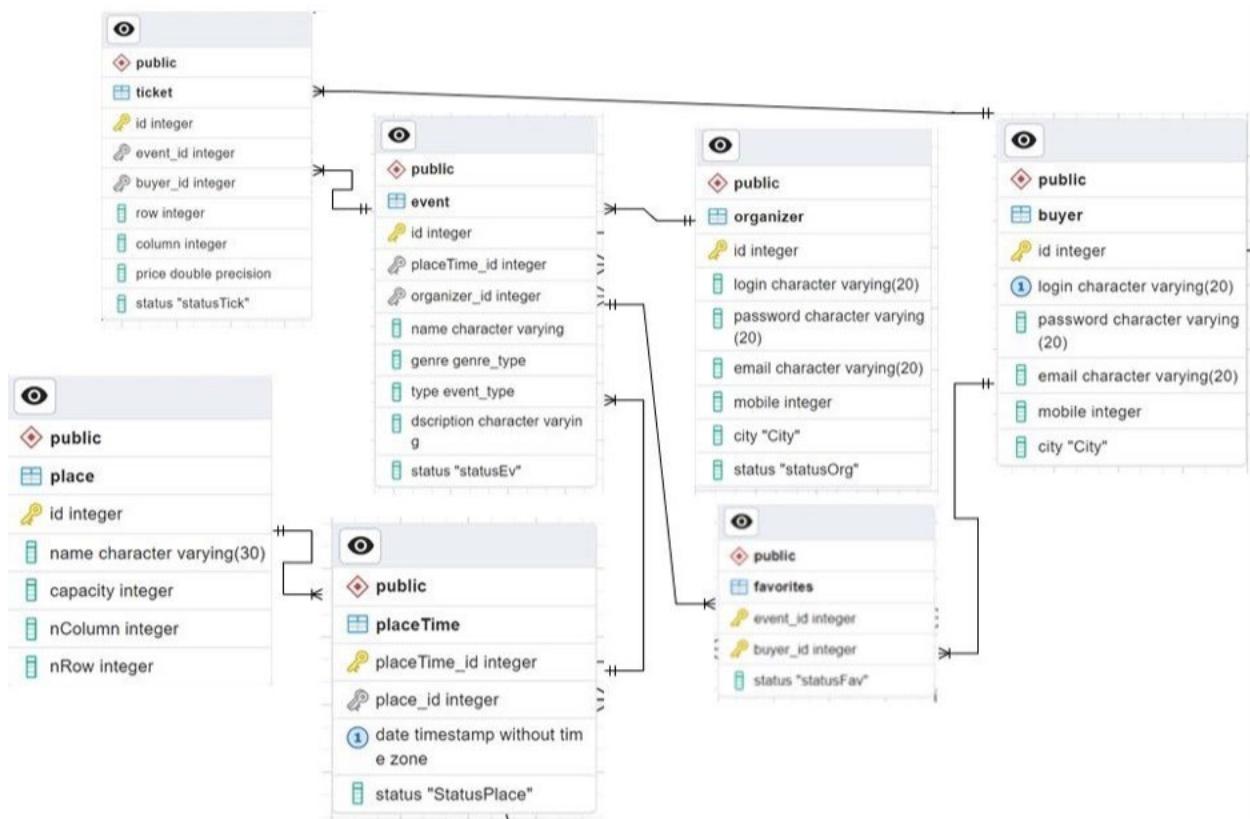


Рисунок 19 - Схема базы данных

Cart – таблица, содержащая информацию о билетах, добавленных в корзину.

Ticket - таблица, содержащая информацию о билетах, которые купил пользователь.

Event - таблица, содержащая информацию о всех мероприятиях.

Organizer - таблица, содержащая информацию о зарегистрированных и прошедших подтверждение организаторах.

Buyer - таблица, содержащая информацию о зарегистрированных покупателях.

Place - таблица, содержащая информацию о площадках, на которых можно организовывать мероприятия.

PlaceTime - таблица, содержащая информацию о занятости площадок.

Favorites - таблица, содержащая информацию о мероприятиях, которые пользователь добавил в избранное.

3.11 Диаграмма классов

3.11.1 Диаграмма классов сущностей

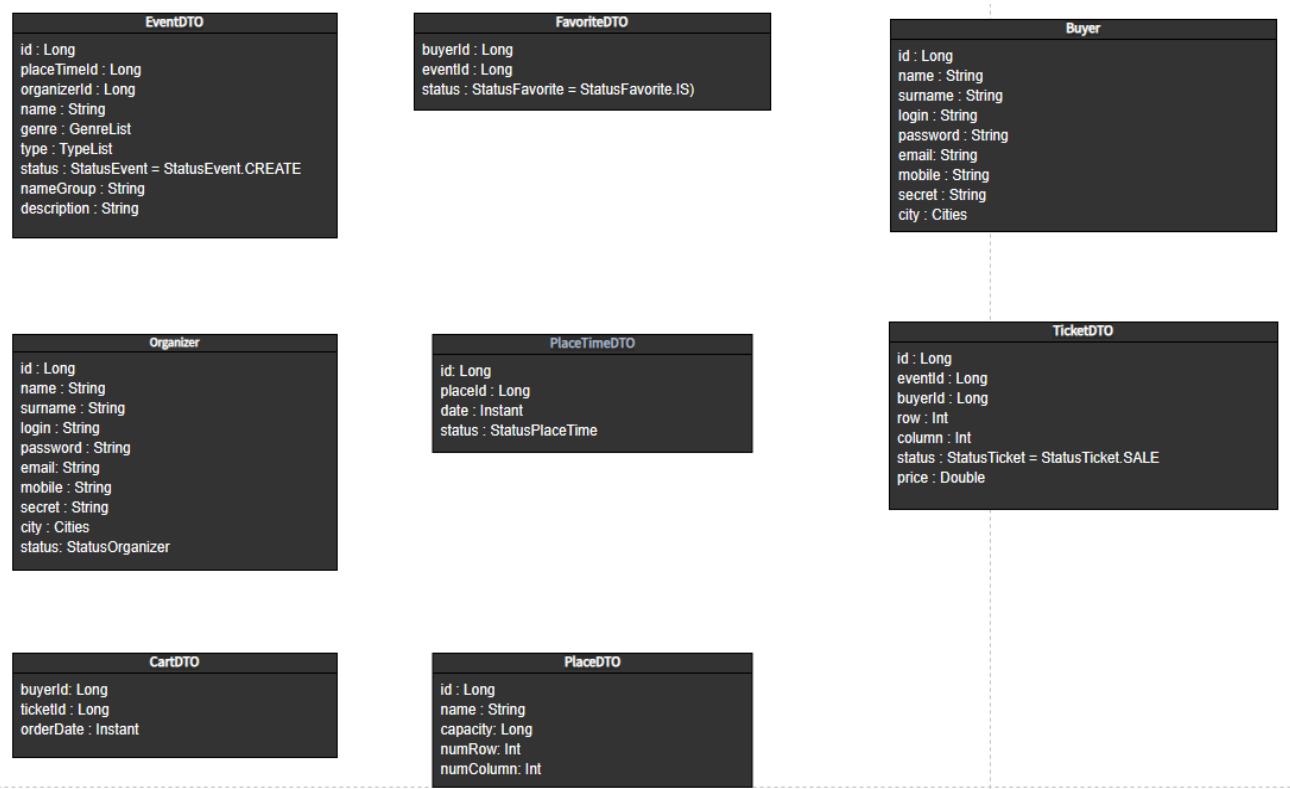


Рисунок 20 - Диаграмма классов сущностей

Поля каждого из этих классов эквивалентны атриутам таблиц в базе данных.

3.11.2 Диаграмма классов интерфейсов транзакций

Все диаграммы классов интерфейсов транзакций должны реализовывать CRUD операции. Диаграмма классов CRUDOperations представлена на рисунке 21.

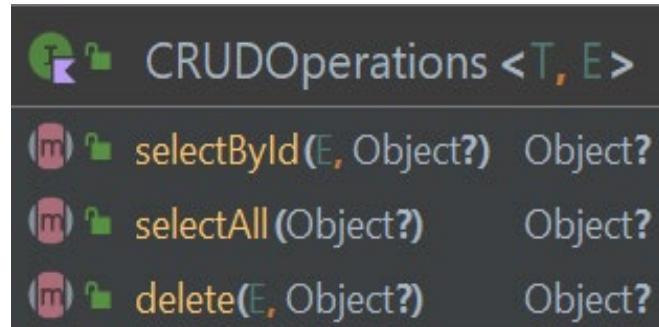


Рисунок 21 - Диаграмма классов CRUDOperations

Диаграммы классов интерфейсов транзакций представлены на рисунке 22.

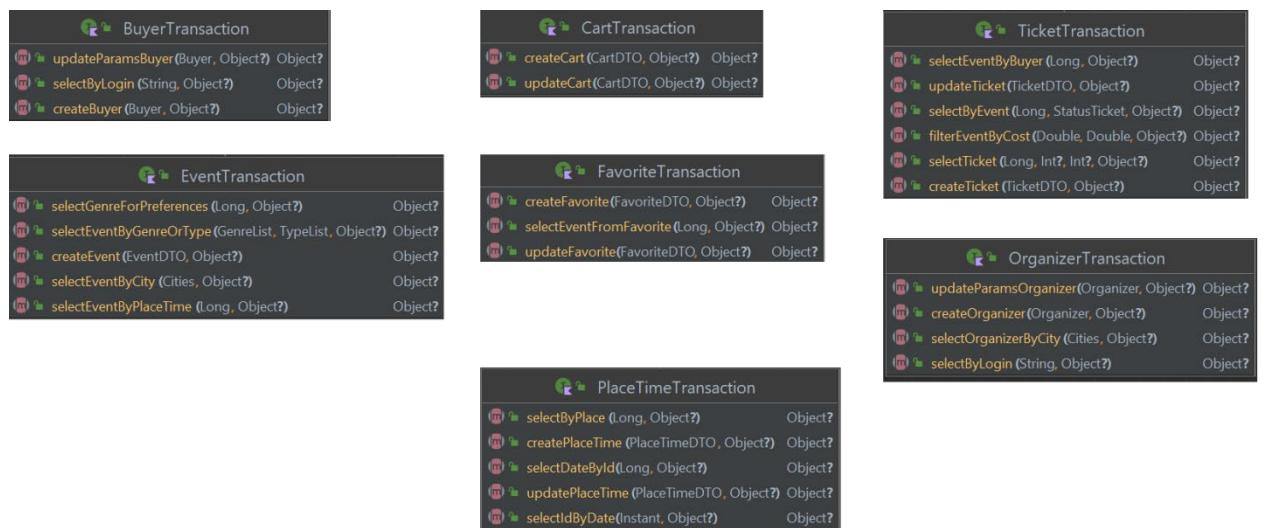


Рисунок 22 - Диаграмма классов интерфейсов транзакций

BuyerTransaction и OrganizerTransaction должны реализовывать еще и изменение параметров пользователя. За это отвечает PersonTransaction, представленный на рисунке 23.



Рисунок 23 - Диаграмма класса PersonTransaction

3.11.3 Диаграмма классов реализаций транзакций

Каждый из этих классов относится к слою доступа к данным, т.е. обеспечивает взаимодействие приложения с базой данных.

Через соответствующие классы с транзакциями (рисунок 24) обеспечивается взаимодействие с таблицами базы данных.

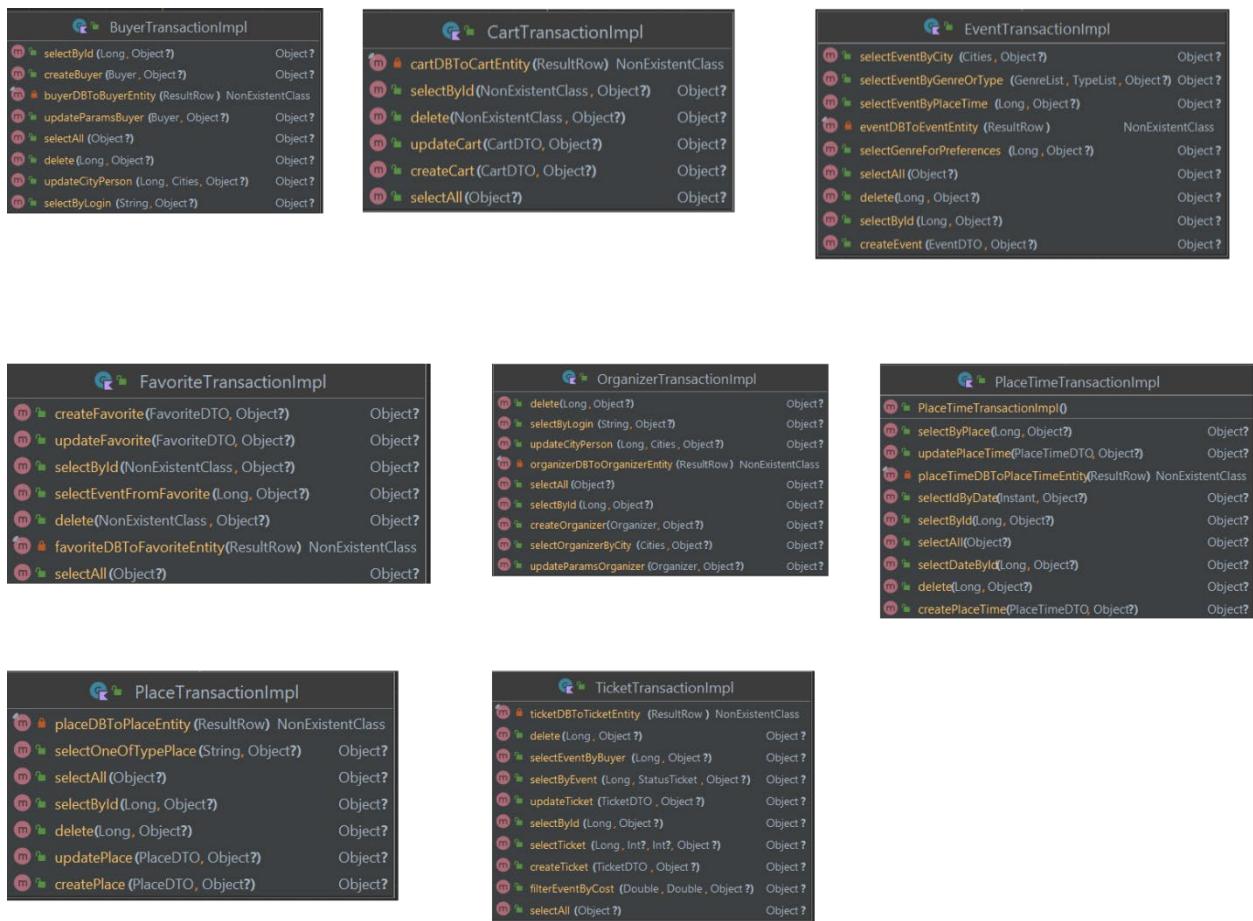


Рисунок 24 - Диаграмма классов реализаций транзакций

Каждый из этих классов является частью слоя бизнес-логики, т.е. выступают связующим звеном между слоем доступа к данным и контроллерами. Все вычисления и алгоритмы находятся в этих классах.

3.11.4 Диаграмма вспомогательных классов для доступа к базе данных



Рисунок 25 - Диаграмма вспомогательных классов для доступа к базе данных

3.11.5 Диаграмма классов маршрутов

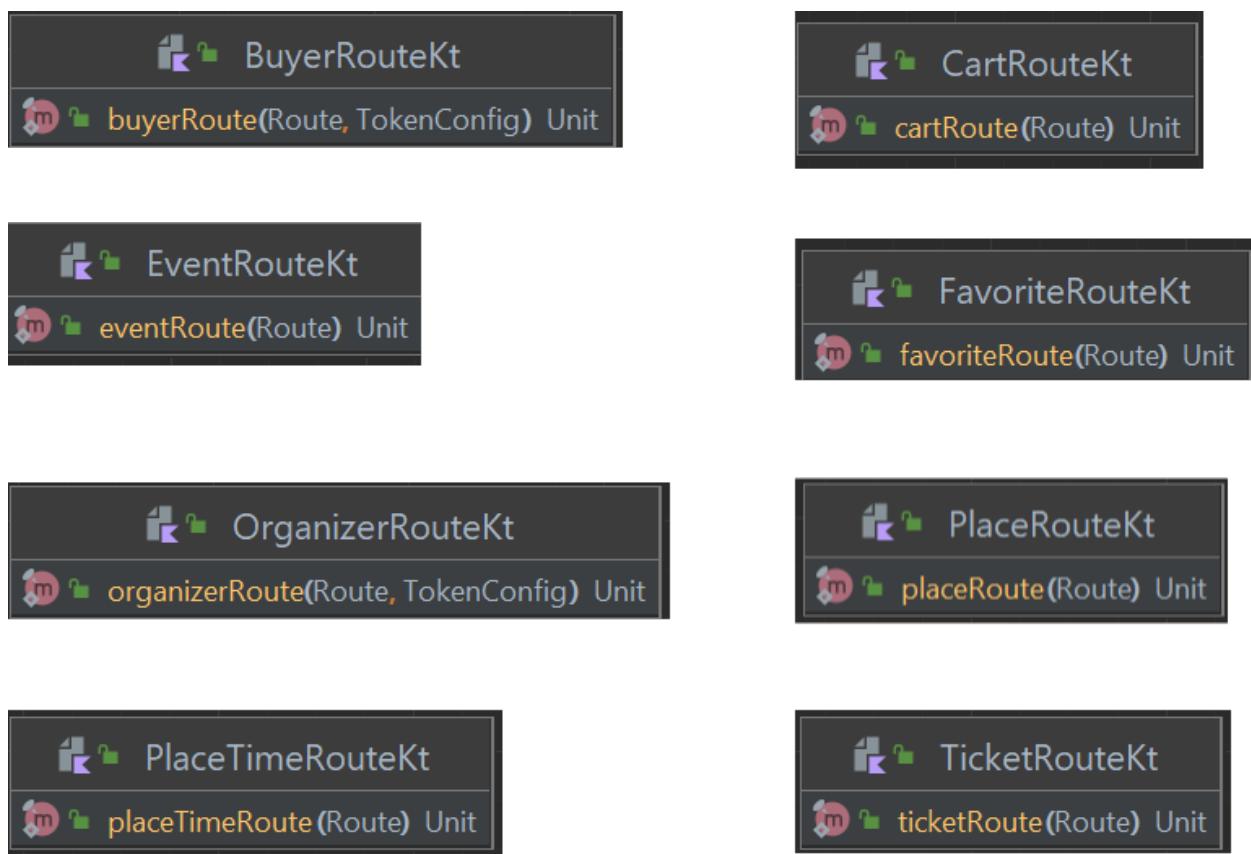


Рисунок 26 - Диаграмма классов маршрутов

Эти классы необходимы для общения с клиентом. Они получают запросы, вызывают методы слоя бизнес-логики и отправляют ответы назад на клиент.

3.11.6 Диаграммы классов для хэширования

С целью защиты личных данных пользователей приложения используется хеширование. Хеширование превращает данные в набор строковых и целочисленных элементов.

Пользователь при регистрации заполняет форму, в которой присутствует поле “пароль”. Перед записью в базу, пароль обрабатывается хэш-функцией. Следовательно, оригинальное значение пароля нигде не используется.

При авторизации пользователь заполняет форму авторизации, в которой требуется ввести логин и пароль. Скрипт-обработчик хэширует пароль, который ввёл пользователь. Скрипт находит запись в базе данных, и считывает значения пароля, который хранится в ней. Пароль из базы и пароль, введённый пользователем, сравниваются, и если они совпадают (в хешированном виде), то пользователя впускают в систему.

Для реализации хеширования используется класс Hash. Диаграмма класса Hash приведена на рисунке 27.

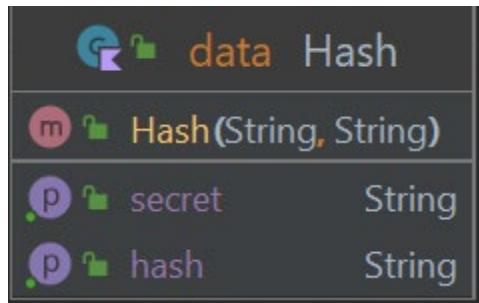


Рисунок 27 - Диаграмма класса Hash

В поле hash хранится зашифрованный пароль. В поле secret хранится исходный пароль, который необходимо хешировать.

Само хеширование реализовано с помощью класса HashServiceImpl, который включает в себя две функции. Диаграмма представлена на рисунке 28.



Рисунок 28 - Диаграмма класса для реализации хэширования
Класса HashServiceImpl реализует интерфейс HashService.

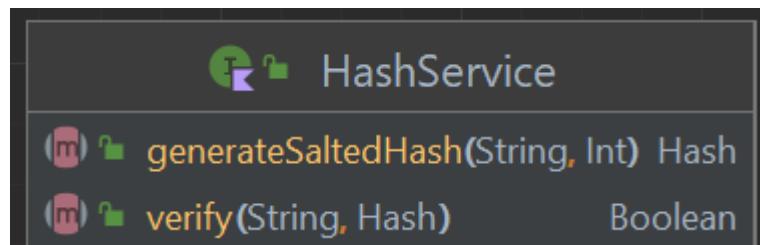


Рисунок 29 - Диаграммы класса HashServiceImpl

3.11.7 Диаграмма классов токенов

Взаимодействие клиента и сервера происходит через токены. Класс JwtTokensevice создает токен.

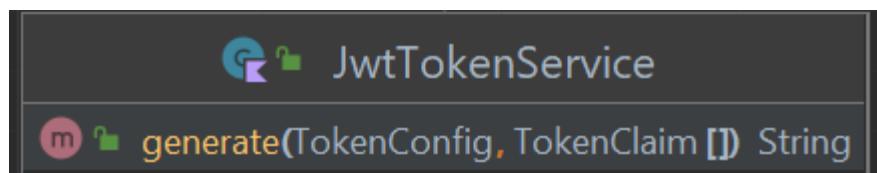


Рисунок 30 - Диаграмма класса JwtTokensevice
Класс JwtTokensevice реализует интерфейс Tokensevice



Рисунок 31 - Диаграмма класса JwtTokensevice

3.10.8 Диаграмма служебных классов

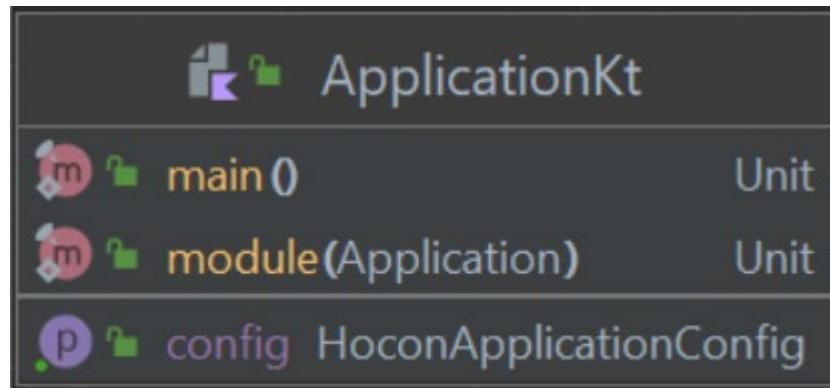


Рисунок 32 - Диаграмма класса Application

Класс Application является точкой входа для приложения, он запускает цепочку загрузки нужных зависимостей и классов, благодаря которым приложение функционирует.

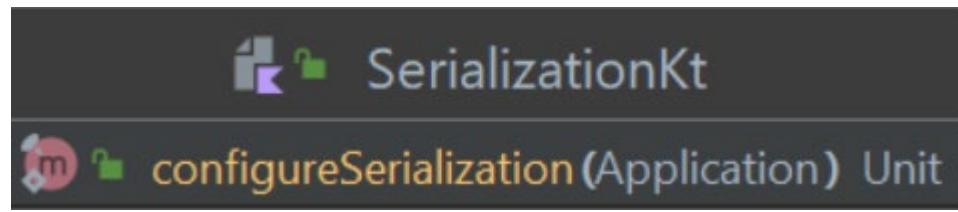


Рисунок 33 - Диаграмма класса Serialization

Класс Serialization используется для настройки библиотеки Jackson.

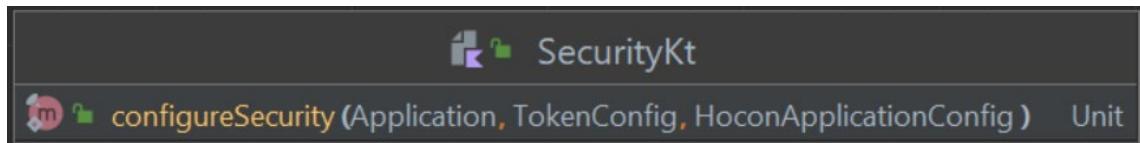


Рисунок 34 - Диаграмма класса Security

Класс Security используется для настройки библиотеки ktor, а именно для настройки безопасности jwt.

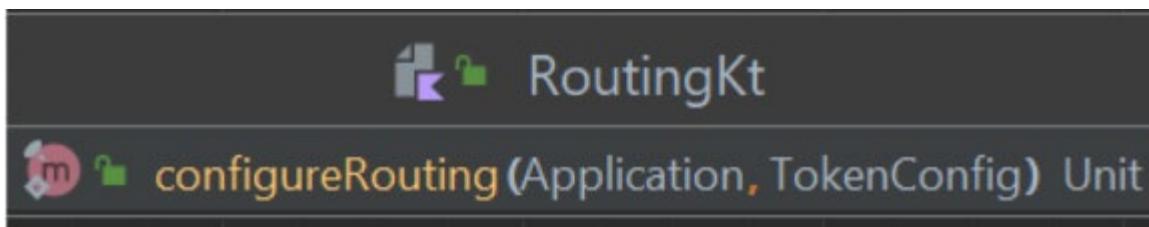


Рисунок 35 - Диаграмма класса Routing

Класс Routing используется для настройки маршрутов для классов:

- placeRoute();
- cartRoute();
- eventRoute();
- favoriteRoute();
- placeTimeRoute();
- ticketRoute();
- roomRoute();
- buyerRoute(tokenConfig);
- organizerRoute(tokenConfig).

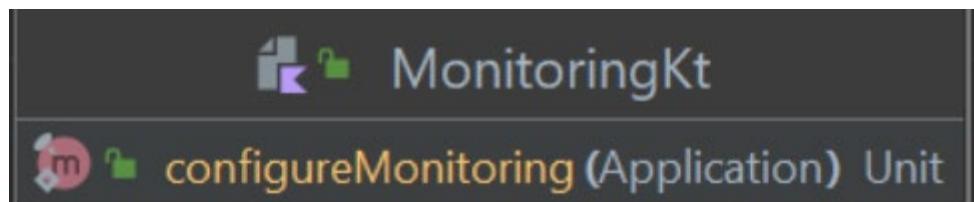


Рисунок 36 - Диаграмма класса Monitoring

Класс Monitoring используется для подключения и настройки библиотеки DropwizardMetrics. DropwizardMetrics позволяет использовать репортеры. Используем репортер SLF4J, который позволяет периодически выдавать отчеты.

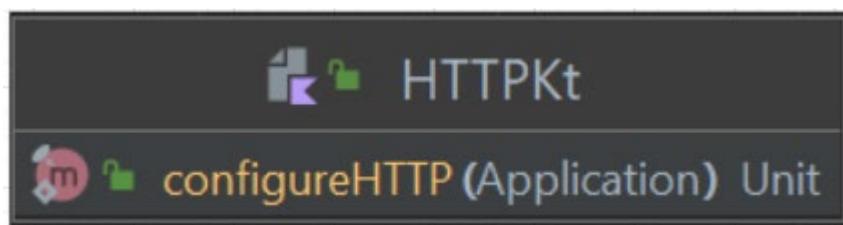


Рисунок 37 - Диаграмма класса HTTP

Класс HTTP используется для настройки и подключения библиотеки CORS. CORS указывает разрешенные методы. В нашем случае, разрешенными являются методы:

- Options
- Put

- Delete
- Post
- Patch

Кроме того, CORS указывает разрешенный хост. В нашем случае, запросы можно принимать с любого хоста, а именно устанавливаем параметр anyHost().

В этом же классе подключаем swagger.

3.11.9 Слой доступа к данным (data layer)

Для каждой сущности из базы данных был реализован DTO (Data Transfer Object) класс. Такой подход позволил комфортно хранить данные в базе данных и оптимизировать их использование для улучшения производительности. Пример DTO для сущности “Мероприятие” приведен на рисунке 38.

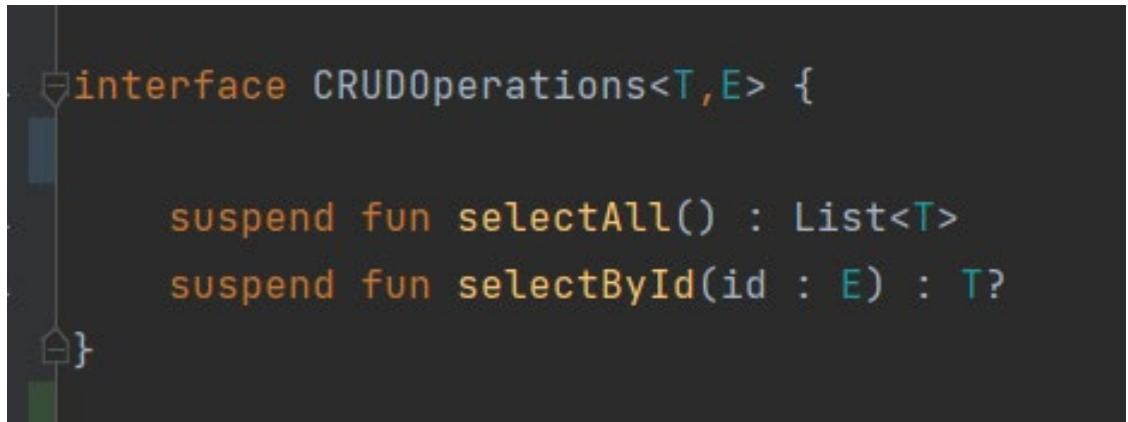
```
@Serializable
data class EventDTO(val id : Long?,
                     val placeTimeId : Long,
                     val organizerId : Long,
                     val name : String,
                     val genre : GenreList,
                     val type : TypeList,
                     val status : StatusEvent = StatusEvent.CREATE,
                     val nameGroup : String? = null,
                     val description : String? = null)

object EventTable : LongIdTable("event"){
    val placeTimeId = long("placeTime_id").references(PlaceTimeTable.id)
    val organizerId = long("organizer_id").references(OrganizerTable.id)
    val name = varchar("name", 100)
    val genre = varchar("genre", 50)
    val type = varchar("type", 50)
    val nameGroup = varchar("name_group", 75).nullable()
    val description = varchar("description", 255).nullable()
    val status = varchar("status", 30)
}
```

Рисунок 38 - Реализация DTO для сущности “Мероприятие”

3.11.10 Слой транзакций

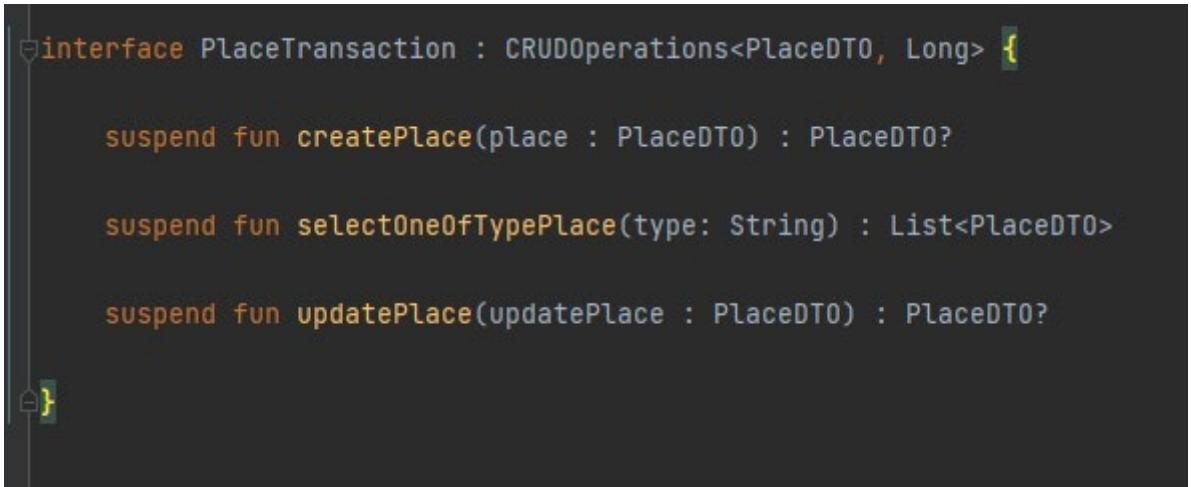
В работе реализован интерфейс CRUD. CRUD – это сокращение от Create (создание), Read (чтение), Update (модификация) и Delete (удаление) – четыре основных операции при работе с базой данных. Использование CRUD упрощает разработку приложения, делая его более масштабируемым и облегчает контроль безопасности, удовлетворяя различные требования доступа.



```
interface CRUOperations<T, E> {  
    suspend fun selectAll(): List<T>  
    suspend fun selectById(id: E): T?  
}
```

Рисунок 39 - Интерфейс CRUD

Для всех классов сущностей созданы интерфейсы-транзакции, которые наследуют CRUD. Например, PlaceTransaction.



```
interface PlaceTransaction : CRUOperations<PlaceDTO, Long> {  
  
    suspend fun createPlace(place: PlaceDTO): PlaceDTO?  
  
    suspend fun selectOneOfTypePlace(type: String): List<PlaceDTO>  
  
    suspend fun updatePlace(updatePlace: PlaceDTO): PlaceDTO?  
}
```

Рисунок 40 - Интерфейс PlaceTransaction

Есть реализация всех интерфейсов-транзакций.

```

class PlaceTransactionImpl : PlaceTransaction {

    private val logger = LoggerFactory.getLogger(javaClass)
    private val place = PlaceTable

    private fun placeDBToPlaceEntity(rs : ResultRow) = PlaceDTO(
        id = rs[place.id].value,
        name = rs[place.name],
        capacity = rs[place.capacity],
        numRow = rs[place.numRow],
        numColumn = rs[place.numColumn]
    )

    override suspend fun createPlace(placeAdd : PlaceDTO) : PlaceDTO? = dbQuery{
        logger.info("Place create transaction is started.")
        val insertStatement = place.insert {
            it[place.name] = placeAdd.name
            it[place.capacity] = placeAdd.capacity
            it[place.numRow] = placeAdd.numRow
            it[place.numColumn] = placeAdd.numColumn
        }
        insertStatement.resultedValues?.singleOrNull()?.let(::placeDBToPlaceEntity)
    }

    override suspend fun selectOneOfTypePlace(type: String): List<PlaceDTO>
    = dbQuery{
        logger.info("Place select by type place transaction is started.")
        if (type == TypeOfPlace.WITH.toString()) place.select{place.numRow neq null; place.numColumn neq null}
            .map(::placeDBToPlaceEntity) else
            place.select{place.numRow eq null; place.numColumn eq null}.map(::placeDBToPlaceEntity)
    }
}

```

Рисунок 41 - Реализация интерфейс PlaceTransaction

Кроме того, приложение поддерживает возможность изменить город у организатора и покупателя. Для этого создан дополнительный интерфейс PersonTransaction.

```

interface PersonTransaction<T> : CRUDOperations<T, Long> {

    suspend fun updateCityPerson(id : Long, city : Cities) : Boolean
}

```

Рисунок 42 - Интерфейс PersonTransaction

Именно его будут наследовать интерфейсы BuyerTransaction и OrganizerTransaction.

```
| interface BuyerTransaction : PersonTransaction<Buyer> {  
  
    suspend fun updateParamsBuyer(buyer: Buyer):Buyer?  
  
    suspend fun createBuyer(buyer: Buyer) : Buyer?  
  
    suspend fun selectByLogin(login : String) : Buyer?  
  
}
```

```
interface OrganizerTransaction : PersonTransaction<Organizer> {  
  
    suspend fun updateParamsOrganizer(organizer: Organizer):Organizer?  
  
    suspend fun createOrganizer(organizer: Organizer) : Organizer?  
  
    suspend fun selectOrganizerByCity(city : Cities) : Query  
  
    suspend fun selectByLogin(login : String) : Organizer?  
  
}
```

Рисунок 43 - Интерфейсы BuyerTransaction и OrganizerTransaction

3.11.11 Слой маршрутизаторов

Маршрутизация — это плагин для упрощения и структурирования обработки запросов страниц. Он предоставляет структурированный способ обработки входящих HTTPS-запросов и генерации соответствующих ответов. С помощью маршрутизации можно определять маршруты, соответствующие определенным URL-адресам и методам запросов, а затем обрабатывать эти запросы.

```

@Suppress("unused")
fun Route.buyerRoute(tokenConfig: TokenConfig){
    val buyerService = BuyerTransactionImpl()
    val tokenService = JwtTokenService()
    val hashService = HashServiceImpl()

    route("/buyers"){
        post{
            call.respond(HttpStatusCode.OK,buyerService.selectAll())
        }
        delete("/{id}") {
            val idFromQuery = call.parameters["id"] ?: kotlin.run {
                throw NotFoundException("Please provide a valid organizer id")
            }
            buyerService.delete(idFromQuery.toLong())
            call.respond("Buyer is deleted.")
        }
        post("/create"){
            val parameters = call.receive<Buyer>()
            val buyer = buyerService.createBuyer(parameters)
            if (buyer == null) call.respond(HttpStatusCode.BadRequest,"Buyer isn't created.") else
                call.respond(
                    HttpStatusCode.OK,BuyerResponse(tokenService.generate(
                        config = tokenConfig,
                        TokenClaim(
                            name = "userId",
                            value = parameters.id.toString()
                        )
                    )))
        }
        put("/{id}/update"){
            val parameters = call.receive<Buyer>()
            val buyer = buyerService.updateParamsBuyer(parameters)
            if (buyer == null) call.respond(HttpStatusCode.BadRequest,"Buyer isn't updated.") else
                call.respond(
                    HttpStatusCode.OK,BuyerResponse(buyer.password))
        }
        put("/signIn") {

```

Рисунок 44 - Пример реализации buyerRoute

При помощи buyerService указывается какой именно класс транзакций будем использовать. В примере это BuyerTransactionImpl().

3.12 Реализация клиентской части приложения

3.12.1 Навигация по приложению

Навигация в приложении осуществляется с помощью кнопок переходов на экраны. Возврат на предыдущий экран возможен при помощи соответствующих иконок или кнопки «Назад» мобильного устройства.

3.12.2 Макеты приветственных экранов

После установки приложения пользователя ожидают приветственные экраны, на которых он увидит направленность и особенности приложения,

которые можно пропустить, нажав на значок, расположенный в правом верхнем углу.

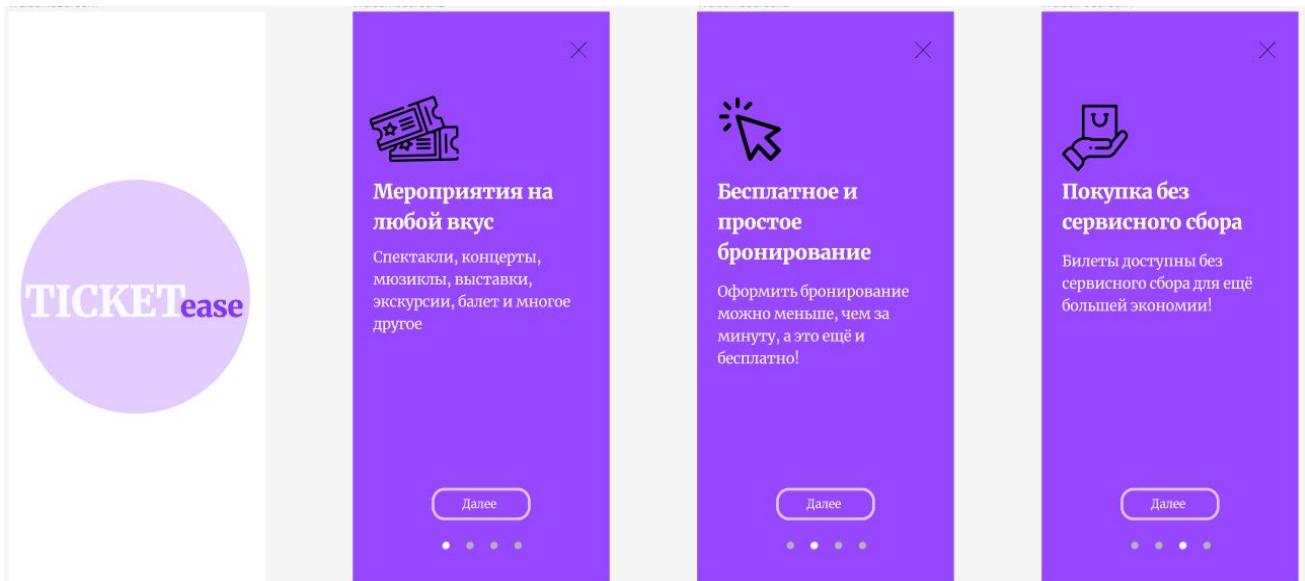


Рисунок 45 - Приветственные экраны

Важные данные, которые пользователь выбирает на данной стадии:

- выбор роли (организатор или покупатель);
- выбор города, который можно будет изменить в любой момент.

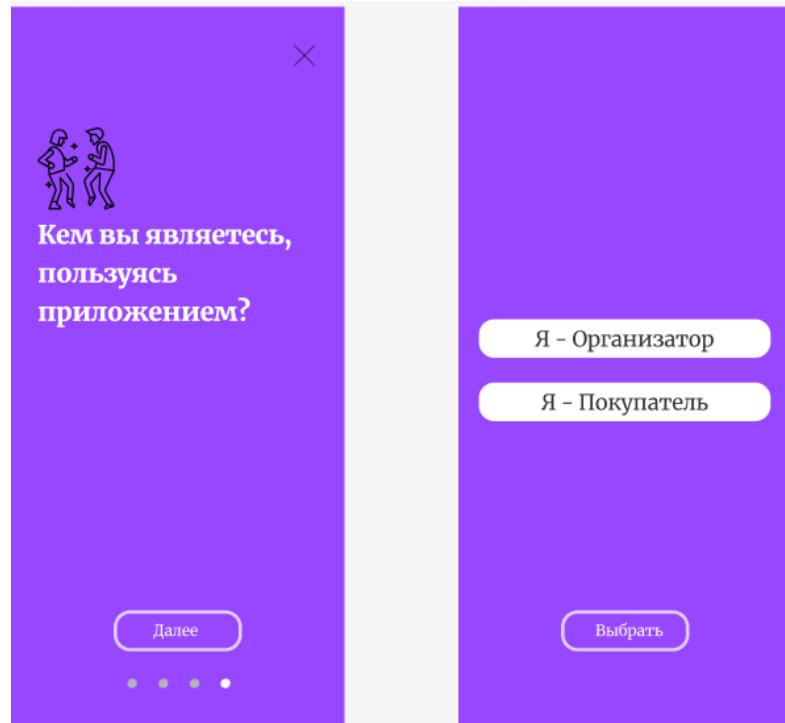


Рисунок 46 - Выбор роли



Рисунок 47 - Выбор города

3.12.3 Сценарий для неавторизованного пользователя

Неавторизованному пользователю доступны следующие экраны: каталог, избранное, предпочтения, корзина, личный кабинет.

Рассмотрим каждый экран подробнее. На данном экране можно просмотреть мероприятия, проходящие в выбранном городе, а также положить билет в корзину или провести фильтрацию по стоимости, дате, типу и жанру мероприятия.

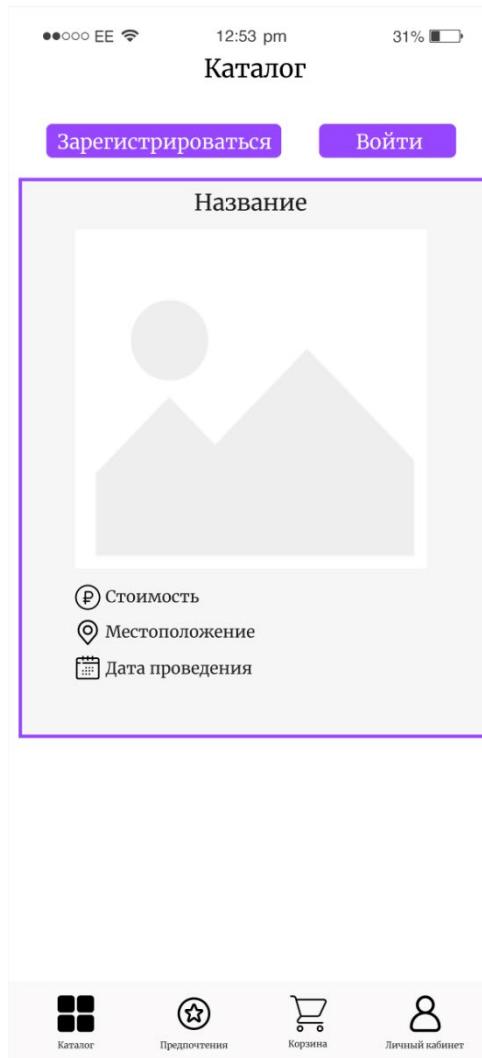


Рисунок 48 - Каталог

В приложении присутствует «Предпочтения». Данный раздел формирует ленту мероприятий по времени проведения мероприятия для неавторизованного пользователя. Также присутствует возможность помещения билета в корзину при нажатии кнопки «Купить».

Предпочтения

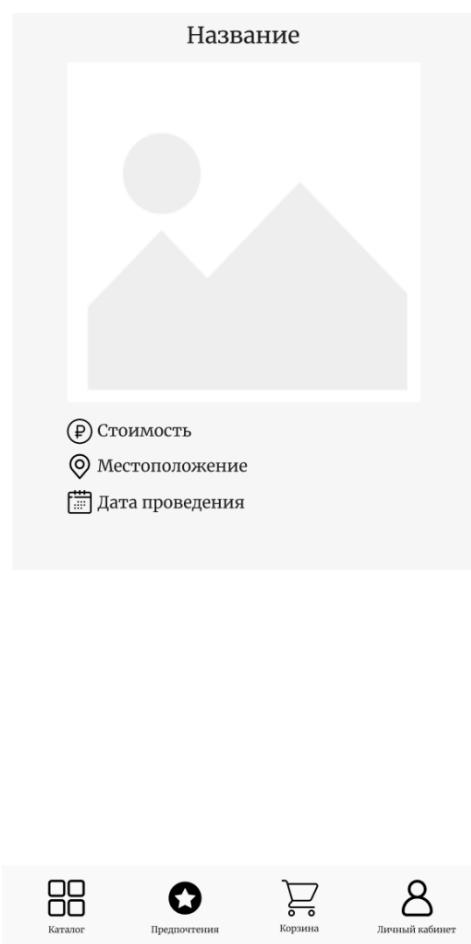


Рисунок 49 - Предпочтения

У неавторизованного пользователя есть возможность посмотреть свою «Корзину» и даже выбрать места в зале или количество билетов в зависимости от типа мероприятия, но купить билет неавторизованный пользователь не может.



Рисунок 50 - Корзина для неавторизованного пользователя

При отсутствии авторизации пользователю предлагается в разделе «Личный кабинет», зарегистрироваться или авторизоваться после выбора кнопки «Войти», также неавторизованный пользователь может изменить город присутствия или задать вопрос команде «TicketEase».

Личный кабинет

Ваш город

Воронеж 



Рисунок 51 - Личный кабинет неавторизованного пользователя

3.13.4 Регистрация покупателя

При регистрации нового покупателя, пользователь должен ввести данные о себе.

Важные данные, которые покупатель вводит на данной стадии:

- «Фамилия»,
- «Имя»,
- «Логин»,
- «E-mail»,
- «Номер телефона»,
- «Пароль»,

- «Подтверждение пароля»;

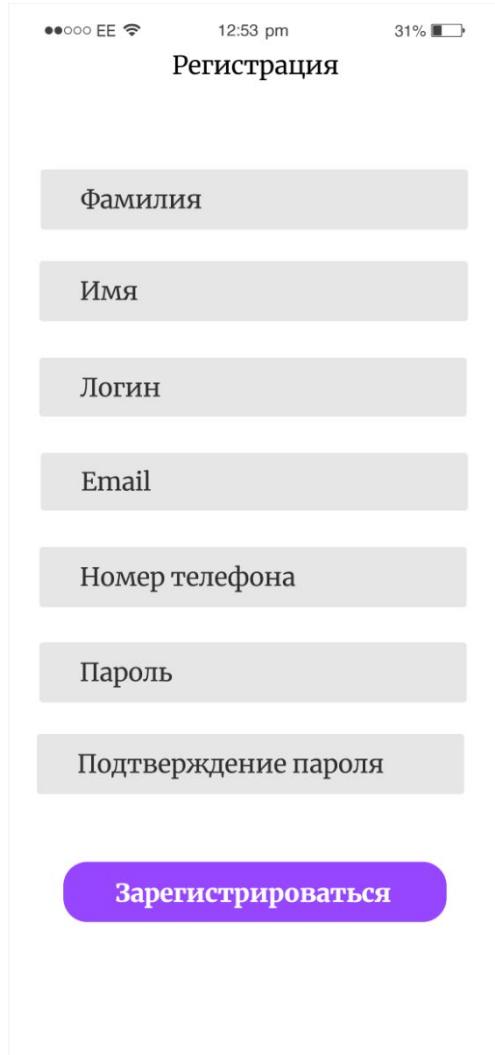


Рисунок 52 - Регистрация покупателя: ввод данных

При ошибочном заполнении появится экран с соответствующим уведомлением.

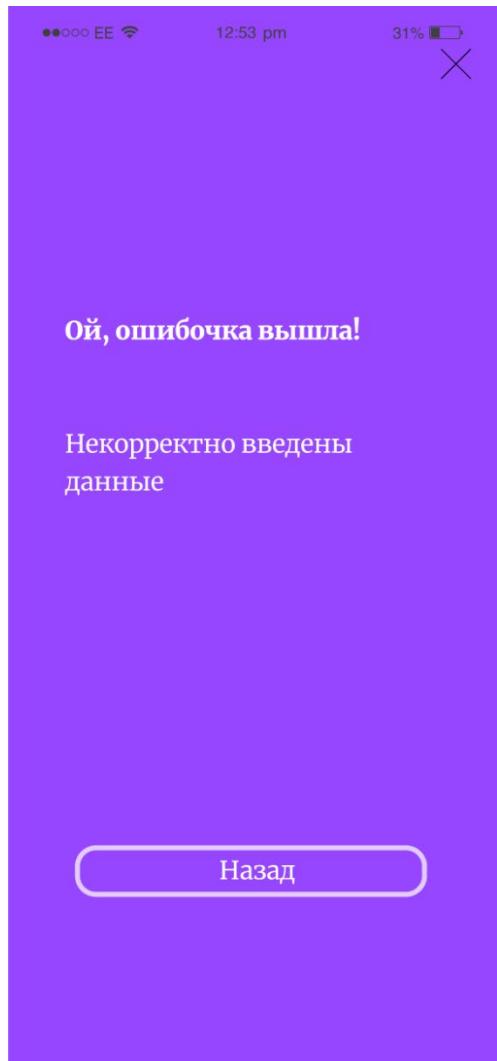


Рисунок 53 - Ошибка при регистрации покупателя

3.12.5 Авторизация покупателя

При наличии у покупателя учётной записи в приложении он может авторизоваться.

Важные данные, которые покупатель вводит на данной стадии:

- «Логин»
- «Пароль»;

Авторизация

Логин

Пароль

Войти

Рисунок 54 - Авторизация покупателя: ввод данных

При отсутствии учётной записи с введёнными параметрами происходит переход на экран с уведомлением с указанием возможной причины не авторизации.

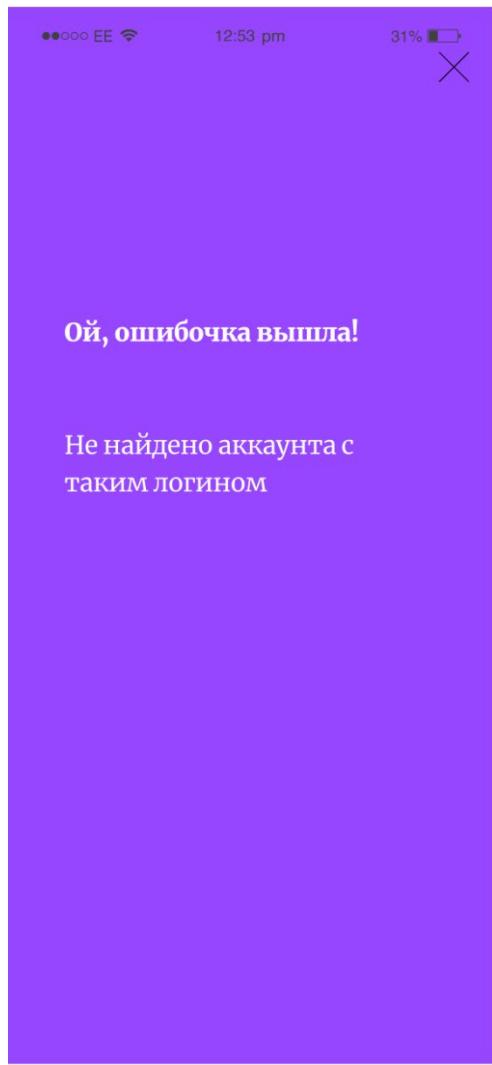


Рисунок 55 - Ошибка при авторизации покупателя

3.12.6 Сценарий для авторизованного покупателя

Авторизованному покупателю доступны, также, как и не авторизованному пользователю, следующие экраны: каталог, предпочтения, корзина, личный кабинет.

Каталог ничем не отличается от экранов, которые видит неавторизованный пользователь.

Все билеты, которые пользователь положил в корзину, он может купить, оплатив через СБП.

●●○○○ EE 12:53 pm 31%

Корзина



Итого ... ₽

Способ оплаты:
СБП

Оформить заказ



Рисунок 56 - Покупка билетов

В случае успешной покупки, приложение оповестит об этом.

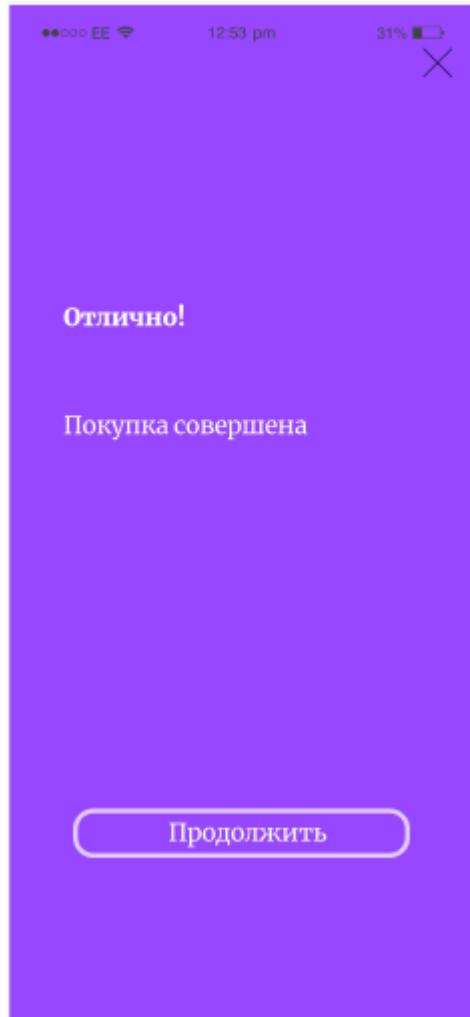


Рисунок 57 - Оповещение о совершении покупки

Раздел «Предпочтения» имеет точно такой же вид, как у неавторизованного пользователя, но данный раздел формирует ленту мероприятий на основе ранее купленных билетов.

Авторизованный пользователь имеет право изменить данные о себе.



Рисунок 58 - Личный кабинет авторизованного пользователя

Покупатель может редактировать личные данные. Данные, которые авторизованный покупатель может изменить:

- Обновить поля: «Фамилия», «Имя», «E-mail», «Номер телефона»;
- Имеет возможность изменить «Город».

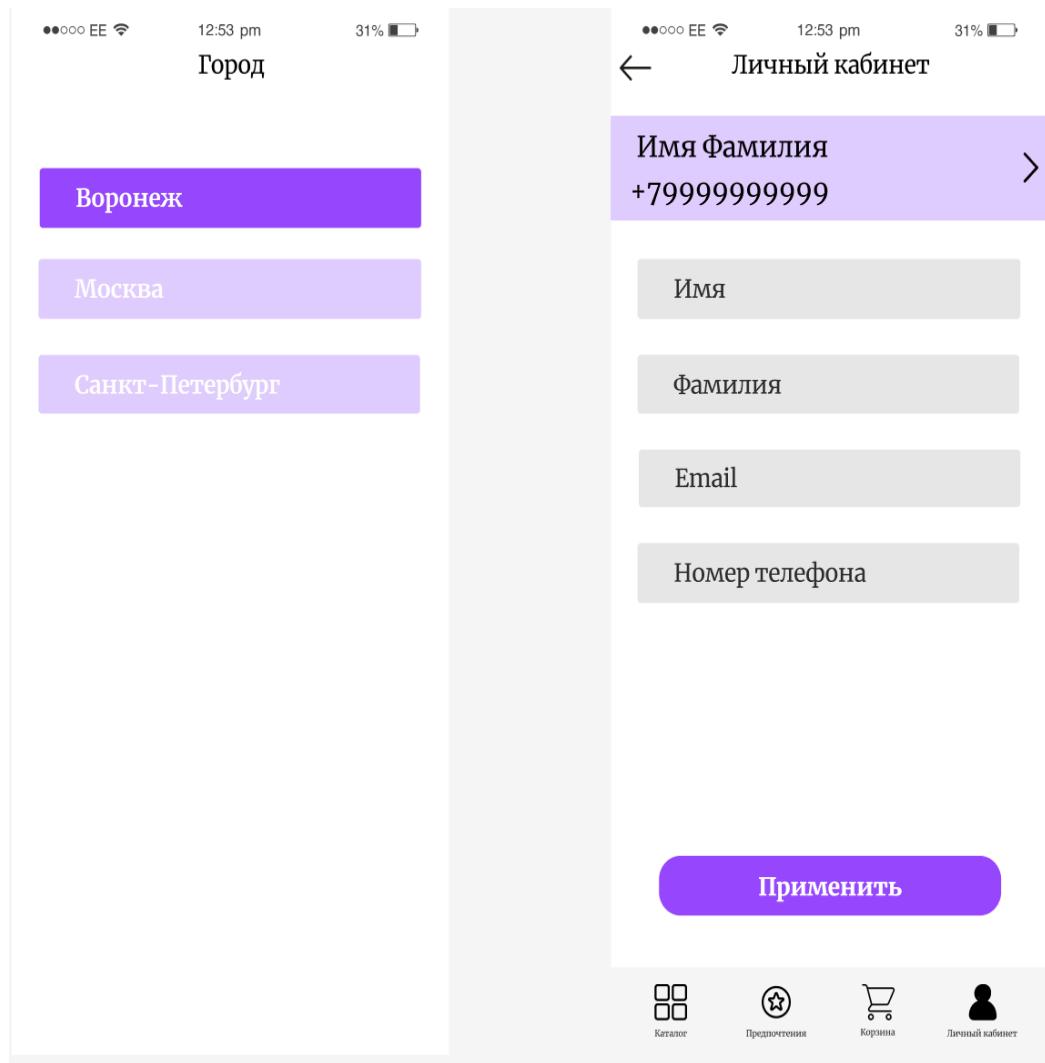


Рисунок 59 - Редактор данных

3.12.7 Регистрация нового организатора

После выбора роли «Я-Организатор» появляется форма регистрации, так как созданный аккаунт организатора проверяется менеджерами компании, чтобы исключить возможность ложного создания аккаунта со стороны покупателя.

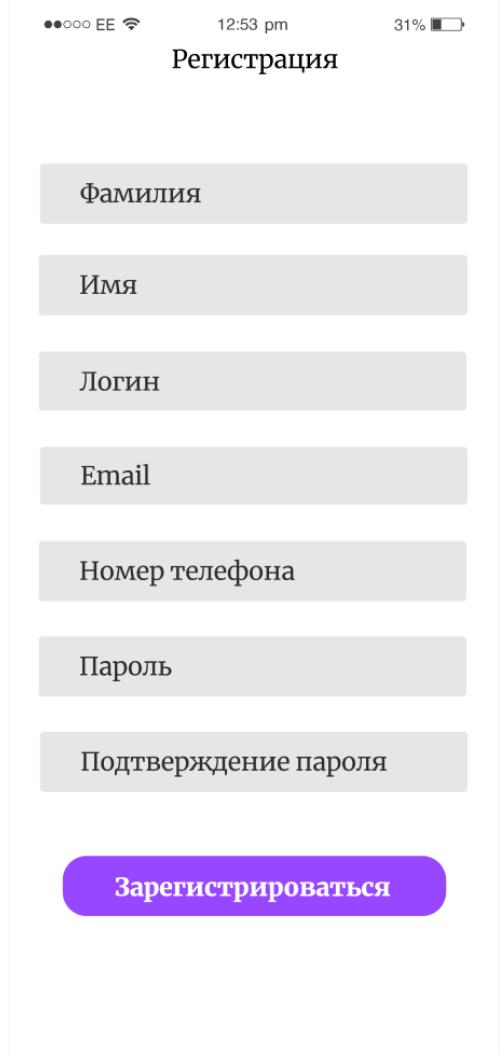


Рисунок 60 - Форма регистрации для организатора

Важные данные, которые организатор на данной стадии:

- Фамилия;
- Имя;
- Логин;
- E-mail;
- Номер телефона;
- Пароль;
- Подтверждение пароля.

При ошибочном заполнении формы появляется экран с уведомлением об ошибке.

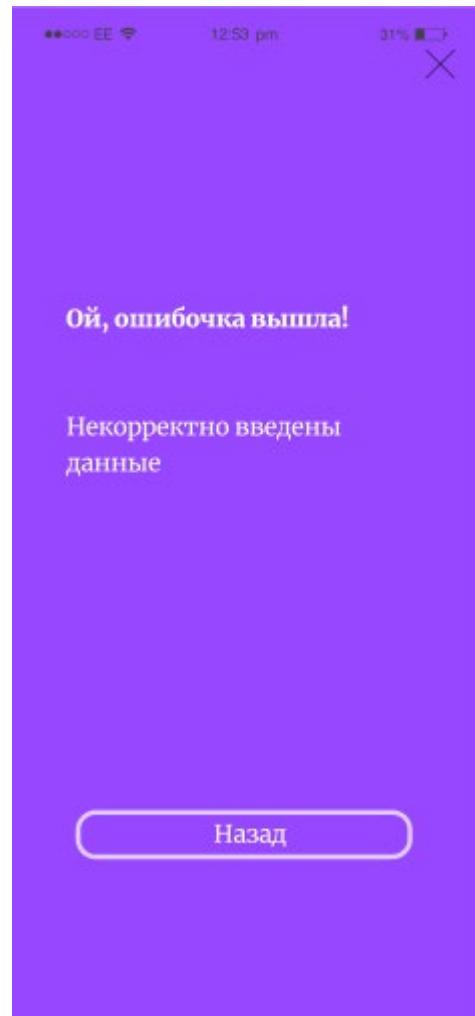


Рисунок 61 - Уведомление при ошибке в форме регистрации

При успешной регистрации появляется экран с уведомлением о том, что с пользователем свяжется наш менеджер и подтвердит учётную запись.

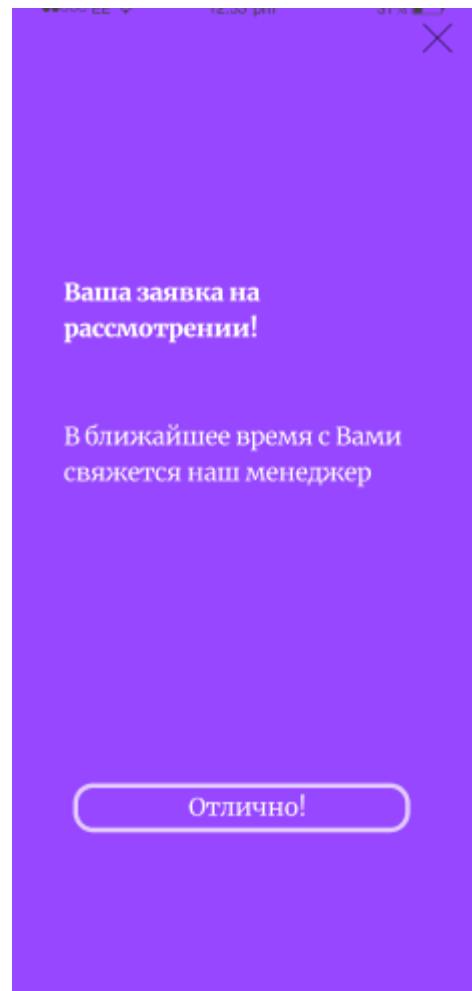


Рисунок 62 - Уведомление при успешной регистрации организатора

3.12.8 Авторизация организатора

При наличии учетной записи организатор может авторизоваться.

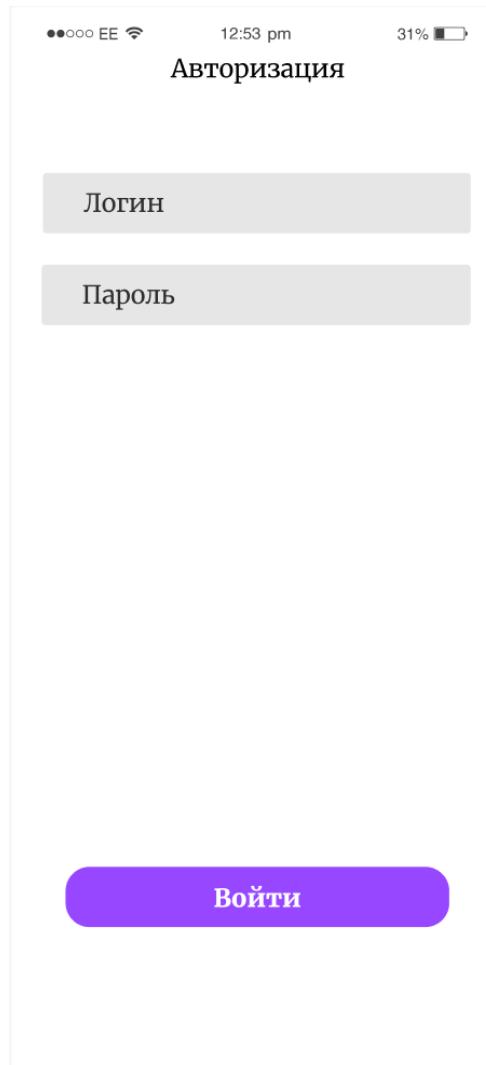


Рисунок 63 - Авторизация регистратора

Важные данные, которые организатор вводит на данной стадии:

- Логин;
- Пароль.

При отсутствии учётной записи с введёнными параметрами происходит переход на экран с уведомлением с указанием возможной причины не авторизации.

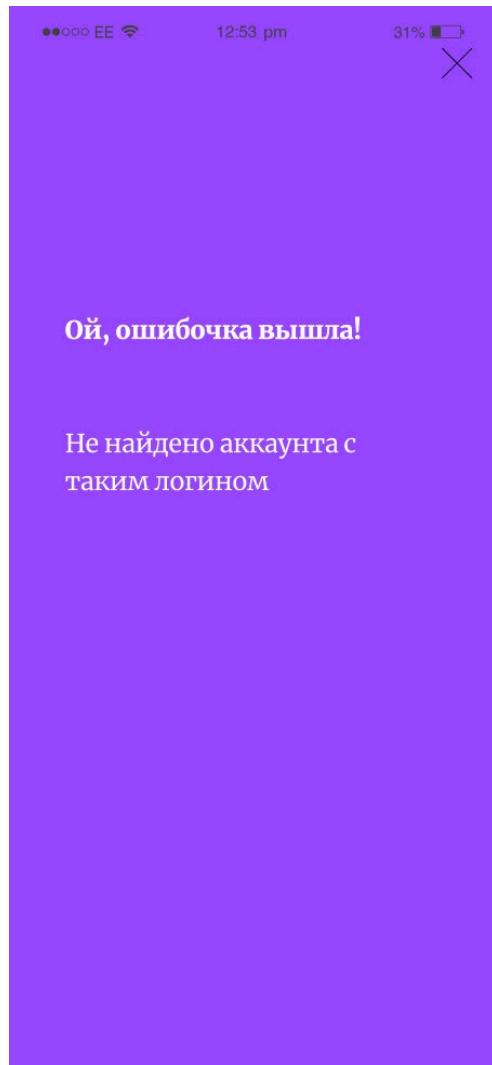


Рисунок 64 - Уведомление об отсутствии аккаунта

3.12.9 Создание мероприятия

Главной функциональной возможностью роли организатора является создание мероприятия, механизм которого представлен на рисунках 65-68.

Важные данные, которые организатор вводит на данной стадии:

- заполняет поля: «Название мероприятия», «Стоимость билета», «Дата мероприятия», «Тип мероприятия», «Жанр», а также заполнить «Название коллектива»;
- Опционально заполняет описание мероприятия;
- Выбирает одну из предложенных площадок;
- Выбирает время, доступное в приложении для данной площадки.

После заполнения всех форм, появляется уведомление об успешном создании мероприятия.

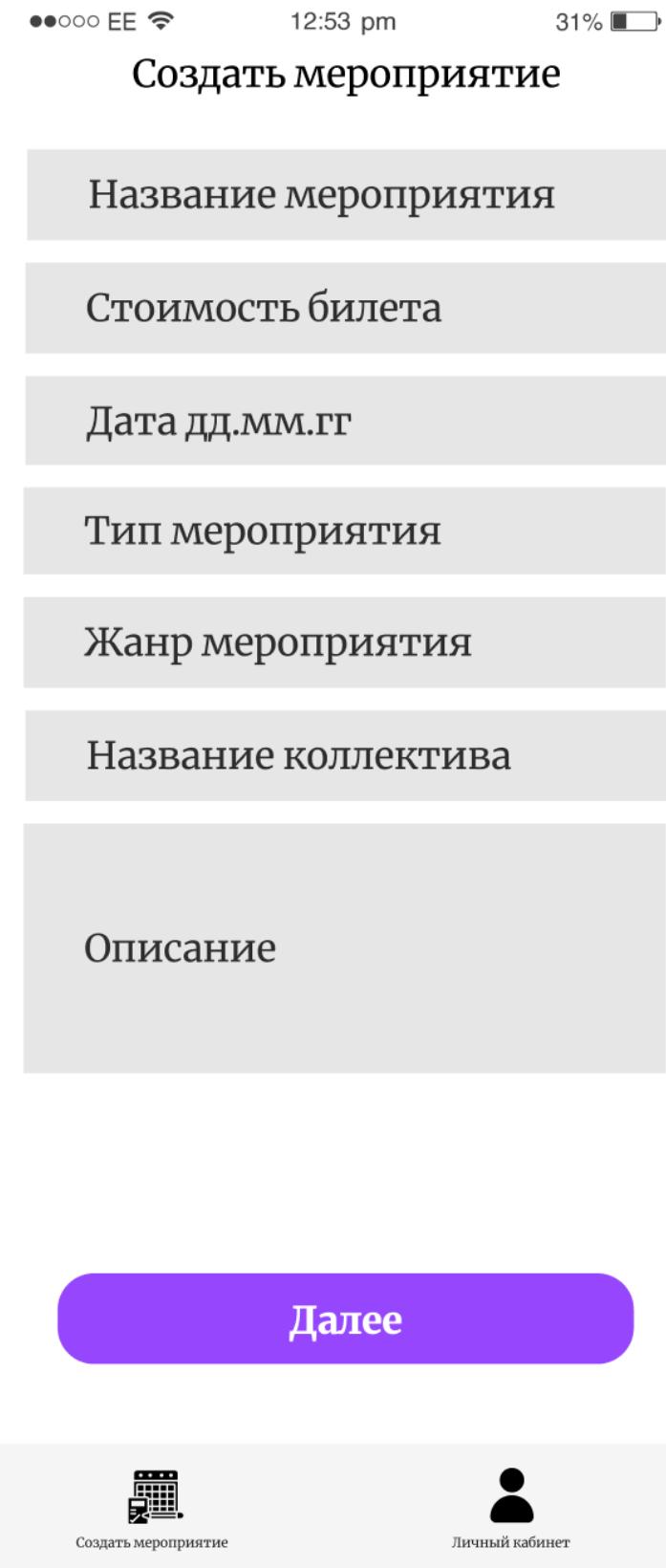


Рисунок 65 - Создание мероприятия: ввод основных данных

Площадка

AURA, проспект Революции, 56, Воронеж

Вместимость зала чел.

ВИНЗАВОД, Кольцовская, 24д, Воронеж

Вместимость зала чел.

Рисунок 66 - Создание мероприятия: выбор площадки

Время проведения

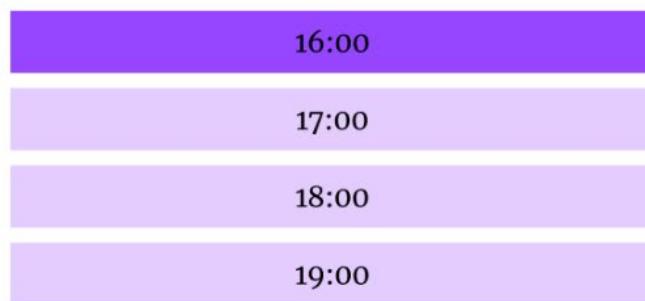


Рисунок 67 - Создание мероприятия: выбор доступного времени



Рисунок 68 - Создание мероприятия: уведомление об успешном создании

3.12.10 Личный кабинет организатора

У организатора есть возможность редактировать свои данные.

Данные, которые организатор может изменить:

- Обновляет поля: «Фамилия», «Имя», «E-mail», «Номер телефона»;
- Так же имеет возможность изменить «Город».

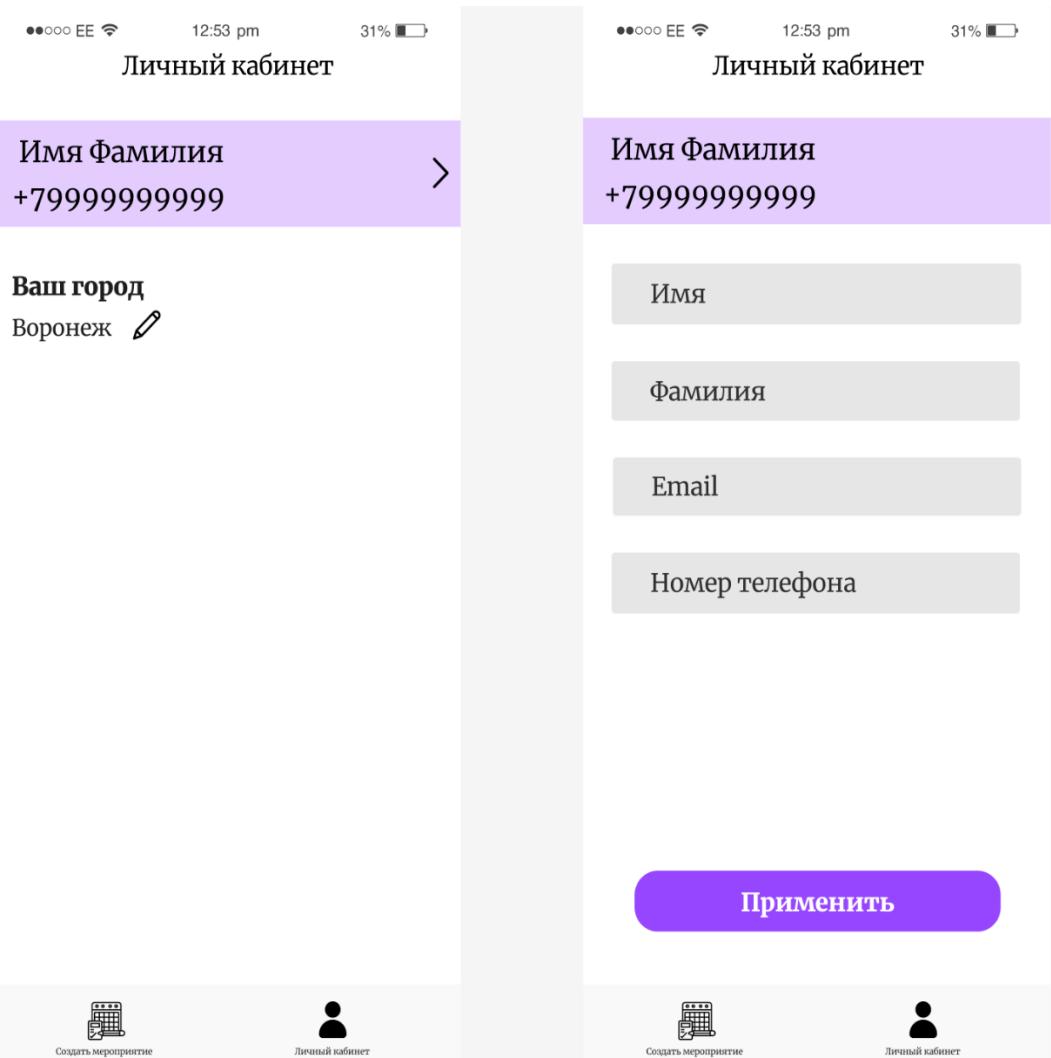


Рисунок 69 – Редактор данных для организатора

3.13 Диаграмма классов

Рассмотрим архитектуру клиентского приложения подробнее. Язык программирования Kotlin, как и другие объектно-ориентированные языки, оперирует объектами, а точнее data классами. Клиентская часть также, как и серверная, связана на использовании сущностей. Приведём диаграмму data классов пакета Person (рисунок 70), по подобному принципу устроены все пакеты с сущностями Event, Place, PlaceTime, Ticket.

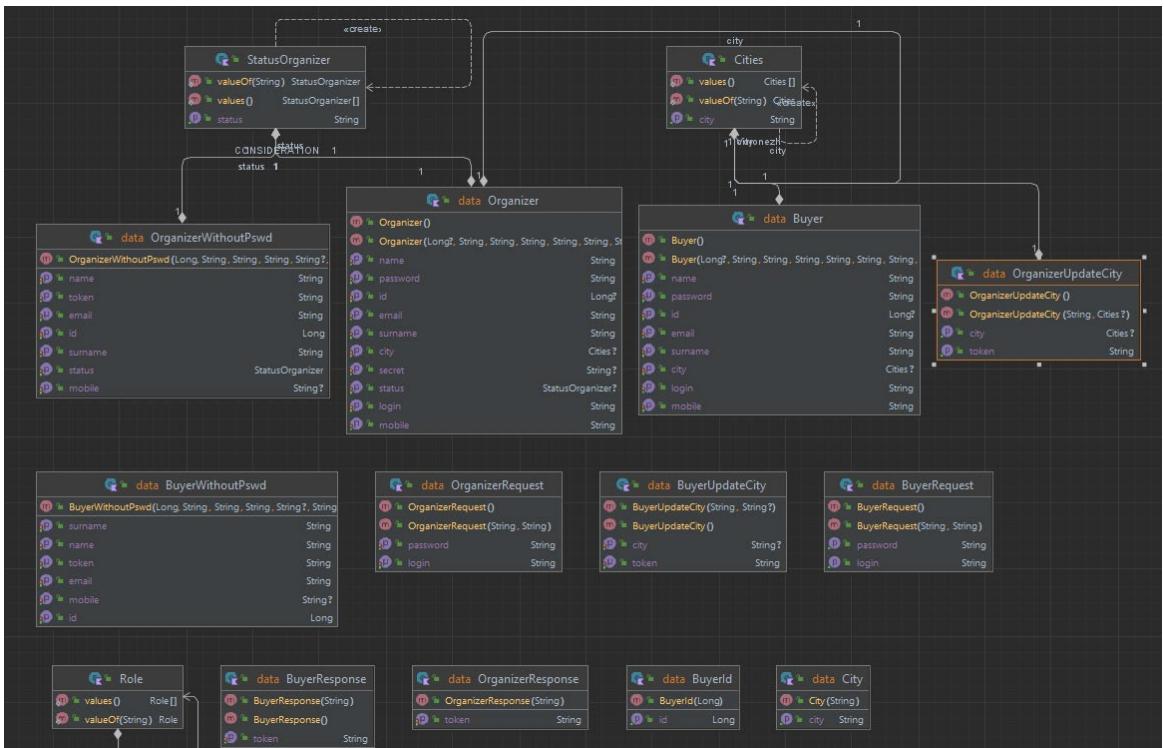


Рисунок 70 – Устройство пакета с сущностями Buyer и Organizer

Далее перейдём к устройству доставки данных от точки входа на клиента со стороны сети до слоя представления – соответствующего экрана приложения. Всё начинается с реализации интерфейса RetrofitAPI (рисунок 71), который позволяет определять соответствующее направление сервера в зависимости от запроса.

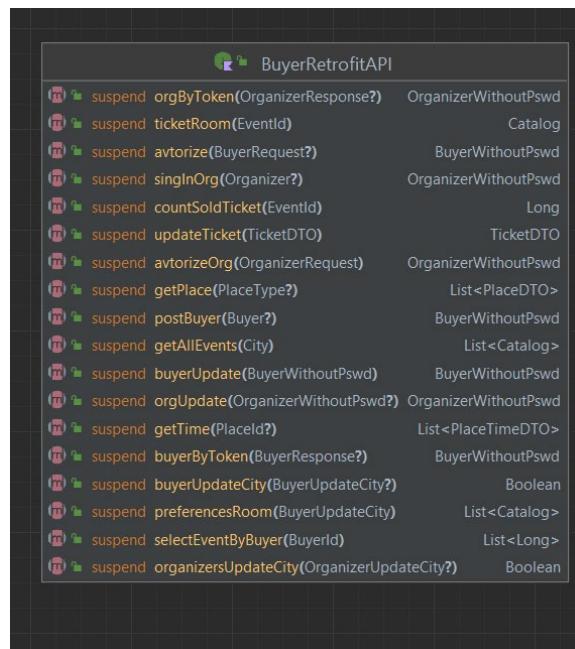


Рисунок 71 – Устройство RetrofitAPI

Далее мы объявляем класс singleton AppModule (рисунок 72), где прописываем поставщиков данных для каждого экрана, в нашем случае поставщиков нет только для экранов с примитивной логикой, подразумевающую наличие не более 1 кнопки возврата. В нашем проекте повсеместно используются аннотации библиотеки Hilt, которая упрощает процедуру ручного внедрения зависимостей в наш проект.

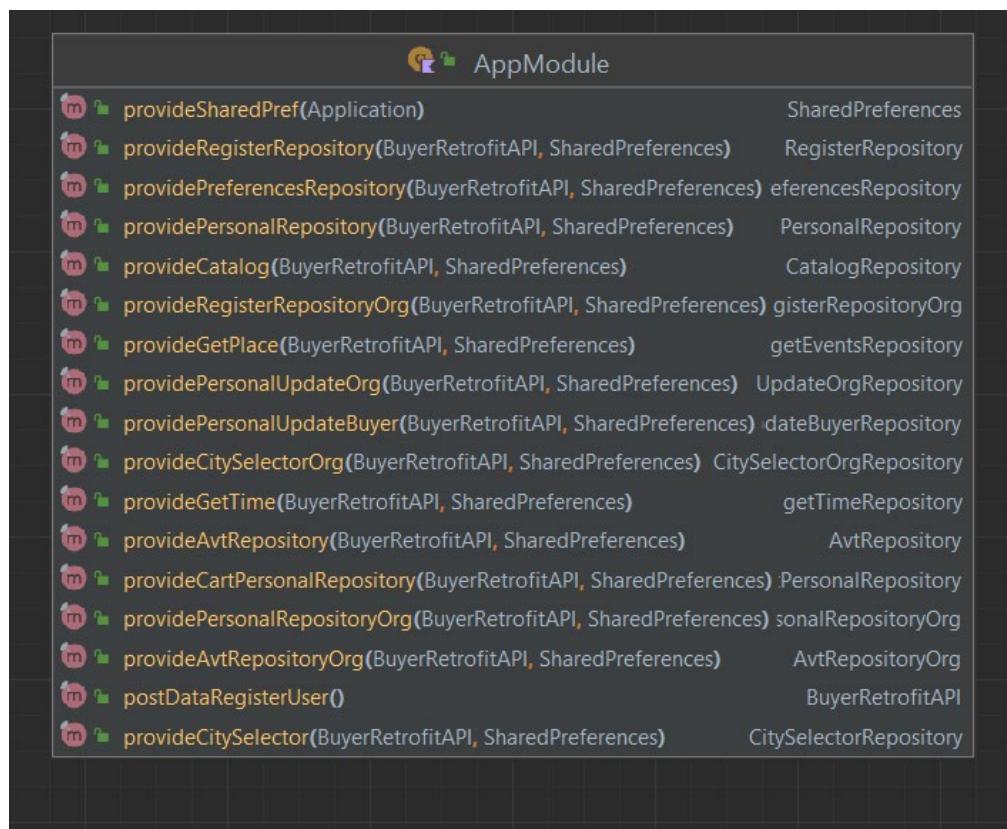


Рисунок 72 – Устройство AppModule

Далее обратимся к реализации паттерна проектирования слоя представления MVVM: ModelView-View-Model. Применение данного подхода полностью соответствует принципам SOLID и позволяет с высокой эффективностью внедрять новую функциональность в приложение.

Рассмотрим реализацию подхода на примере экрана с регистрацией покупателя. Класс RegisterRepository и его производные (рисунок 73) поставляют данные на экран из сети или из внутреннего хранилища или сохраняют данные в SharedPreferences или отправляют их на сервер, и передают их в класс ViewModelRegisterBuyer (рисунок 74), который отвечает

за хранение состояний экрана с регистрацией. Для хранения изменяемых полей экрана был разработан sealed класс (рисунок 75), который контролирует любое изменение полей. После этого на экране при нажатии кнопки проводится соответствующий запрос к серверу через паттерн Repository (классы RegisterRepository) и сохраняется в поле состояния, которое уже вызывается на презентационном классе RegisterBuyer (рисунок 76), где хранится отрисовка экрана и проверка простых ситуаций с неправильным вводом.

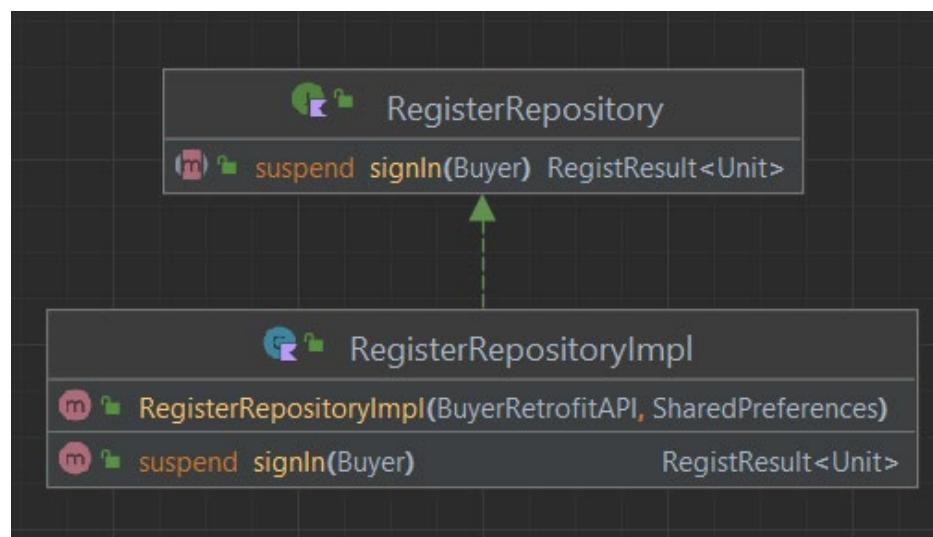


Рисунок 72 – Диаграмма классов RegisterRepository

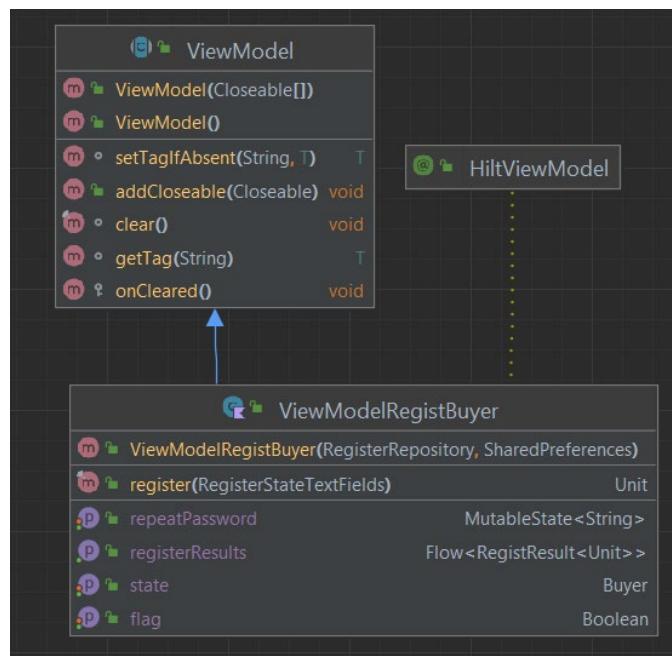


Рисунок 73 – Диаграмма классов ViewModelRegistBuyer

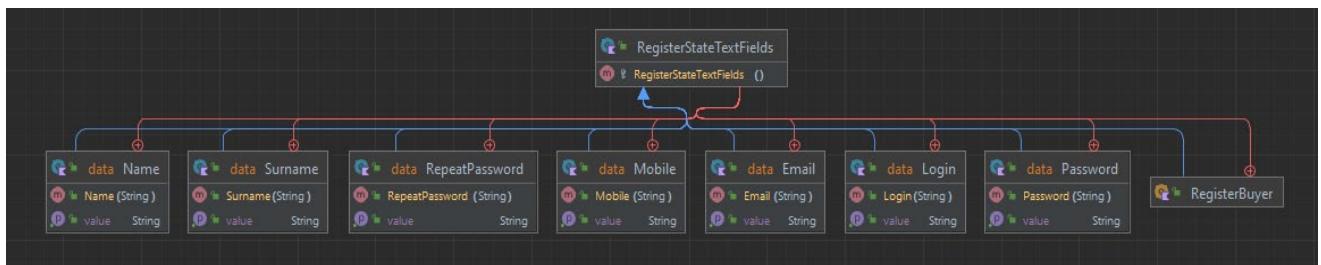


Рисунок 74 – Диаграмма классов sealed class RegisterStateTextFields

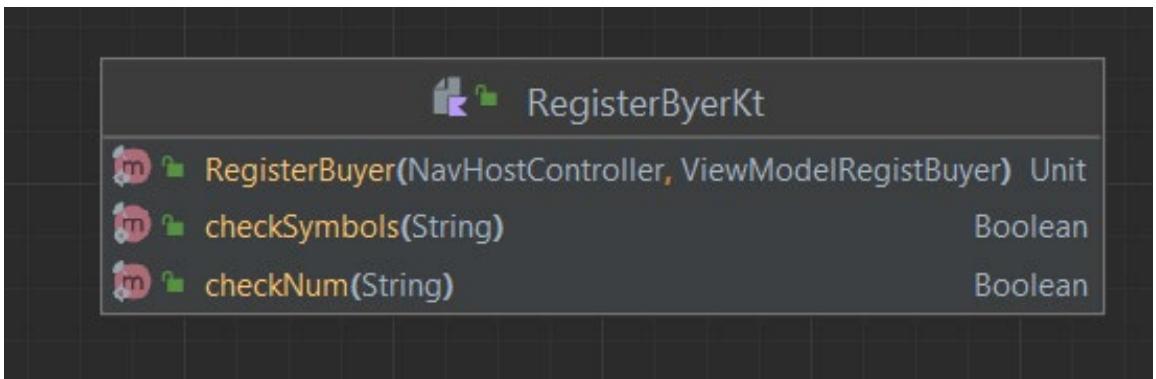


Рисунок 75 – Диаграмма класса представления RegisterBuyer

По данному принципу устроены все экраны приложения.

3.14 Календарный план

При разработке приложения использовался следующий календарный план:

Плановый срок начала работ – Март 2023 г.

Плановый срок окончания работ – Май 2023 г.

Основные этапы работ по созданию системы, их содержание и примерные сроки приведены в таблице 1.

Таблица 1 – Основные этапы разработки приложения со сроками их исполнения.

Этап	Содержание работ	Порядок приёмки документы	Сроки	Ответственный
1.Составление ТЗ	Разработка функциональных и нефункциональных	Утвержденное ТЗ https://docs.google.com/document/d/1wh3F7_nDOtC5FdGBcnFrBjiT3ZxzKr7L/edit	До 24.03.2023	Разработка – Исполнитель Положенцев А.А Согласование – Заказчик.

	требований к системе			
2.Техническое проектирование	Разработка сценариев работы системы	Ссылка на figma.com https://www.figma.com/file/PaynYsGdjVVYR96SojTCNb/TicketEase?type=design&node-id=0-1&t=QRglDz8UccUmdsz3-0	До 24.03.2023	Исполнитель Бредихина А.А
	Разработка дизайн-макета проекта	Предоставление изображений дизайн-макета проекта	До 24.03.2023	
3.Разработка программной части	Разработка серверного модуля, модуля хранения данных	Приемка осуществляется в процессе испытаний	В течение 55 дней с момента утверждения ТЗ	Исполнитель Положенцев А.А, Щербинина А.В
	Разработка статической части приложения			
	Разработка динамической части приложения			
4.Предварительные автономные испытания	Проверка соответствия функциональным требованиям	Согласно ТЗ	В течение 7 дней с момента завершения разработки	Исполнитель Бредихина А.А
	Проверка комплекта документации			
	Доработка и повторные испытания до устранения недостатков			

6.Разработка курсового проекта	Разработка курсового проекта, содержащего аналитическую информацию о проекте на основе ТЗ	В течение всего времени работы над проектом	До 07.06.2023	Исполнитель Щербинина А.В Мишакин П.С
7.Опытная эксплуатация	Эксплуатация с привлечением небольшого количества участников	Ведение соответствующего внутреннего документа	До 07.06.2023	Исполнитель Мишакин П.С
	Доработка и повторные испытания до устранения недостатков			

Заключение

В ходе выполнения данного курсового проекта был выполнен анализ предметной области и аналогов разрабатываемого приложения.

Для разработки приложения были разработаны макеты интерфейса, выбрана платформа приложения, построены UML диаграммы.

Для контроля версий был создан репозиторий GitHub.

При разработке приложения было реализовано следующее:

- Реализация слоя доступа к БД;
- Реализация интерфейса;
- Подключение swagger;
- Размещение backend-части и БД на облаке;
- Реализация модуля авторизации и регистрации;
- Описание процесса разработки и результата.

Разработанное приложение удовлетворяет поставленным требованиям.

Все поставленные задачи были выполнены.

В качестве дальнейшего развития проекта рассматриваются следующие усовершенствования:

- Обслуживание залов с местами для зрителей;
- Авторизация покупателей с помощью google аккаунта;
- Просмотр купленных ранее билетов покупателями;
- Просмотр организованных мероприятий организаторами;
- Удаление не проведённых мероприятий.

Список использованных источников

1. How to Post Data to API using Retrofit in Android using Jetpack Compose? [Электронный ресурс]: статья. — Режим доступа: <https://www.geeksforgeeks.org/how-to-post-data-to-api-using-retrofit-in-android-using-jetpack-compose/>
2. Jetpack Compose Tutorial [Электронный ресурс]: документация. — Режим доступа: <https://developer.android.com/jetpack/compose/tutorial>
3. Веб-токены JSON [Электронный ресурс]: документация. — Режим доступа: <https://ktor.io/docs/jwt.html>
4. Филлипс, Б. Android Программирование для профессионалов [Текст] / Б. Филлипс, Марсиано К., Гарднер Б. ; перевод с англ. С. Черников, Е. Матвеев. - Питер, 2021. – 704 с.: ил. — (Серия «Для профессионалов»). - 1200 экз. – ISBN 978-5-4461-1657-7.
5. Формирование JWT-token [Электронный ресурс]: программа. — Режим доступа: <https://jwt.io/>
6. Сколько россияне тратят на театр [Электронный ресурс]: статья. Режим доступа: <https://journal.tinkoff.ru/theatre-stat/>
7. Тренды культурного досуга: 1992-2022 [Электронный ресурс]: статья. — Режим доступа: <https://wciom.ru/analytical-reviews/analiticheskii-obzor/trendy-kulturnogo-dosuga-1992-2022>