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

Факультет компьютерных наук
Информационные технологии управления

# *Разработка приложения для продажи билетов на мероприятия «CityConcert»*

*Курсовой проект по дисциплине «Технологии программирования»*

09.03.02 Информационные системы и технологии
Кафедра Информационные технологии управления

Преподаватель _____В.С. Тарасов, ст. преподаватель __.__20__

Обучающийся _____М.Н. Быков, 3 курс, д/о
Обучающийся _____И.Д. Наумова, 3 курс, д/о
Обучающийся _____О.С. Дьяконова, 3 курс, д/о
Обучающийся _____И.В. Черников, 3 курс, д/о

Воронеж 2023

## Содержание

# Введение

В современном мире мобильные приложения являются неотъемлемой частью нашей жизни. Они помогают нам во многих сферах, от общения и развлечений до работы и управления финансами. Мобильные приложения позволяют нам быть всегда на связи и получать необходимую информацию в любое время и в любом месте. Одним из времязатратных занятий, которое было облегчено приходом в нашу жизнь смартфонов, стала покупка билетов. Раньше мы были вынуждены преодолевать зачастую не близкий путь к необходимой кассе, отстаивать длинную очередь и надеяться, что, когда придёт наш черёд, долгожданные билеты не закончатся. Разумеется, сейчас все эти трудности в прошлом. Каждый обладатель современного телефона может выбрать билет на любимое мероприятие в несколько кликов – на помощь ему в этом придут специальные мобильные приложения, чья задача – демонстрировать грядущие мероприятия и продавать билеты на них. Конкуренция в данном секторе рынка мобильных приложений крайне высока, что вынуждает разработчиков этих приложений постоянно совершенствовать продукт. Об одном из таких проектов и пойдёт речь в данной курсовой работе.

## 1 Глоссарий

MVC — это шаблон программирования, разделяющий архитектуру приложения на три модуля: модель (Model), представление (View), контроллер (Controller). Он позволяет изменять каждый компонент независимо друг от друга для простой разработки и поддержки веб-приложений.

HTTP — HyperText Transfer Protocol – это широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

XML — eXtensible Markup Language — расширяемый язык разметки) — это язык программирования для создания логической структуры данных, их хранения и передачи в виде, удобном и для компьютера, и для человека.

JSON — JavaScript Object Notation — это формат передачи данных, который используется при взаимодействии веб-сервера и браузера.

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

ORM — Object-Relational Mapping (объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

GET — метод для чтения данных с сайта.

POST — метод для направления запроса, при котором веб-сервер принимает данные, заключённые в тело сообщения, для хранения.

PUT — метод, который служит для изменения или вставки ресурса.

DELETE — метод, который запрашивает первоначальный сервер об удалении ресурса, идентифицируемого запрашиваемым URI (Request-URI).

GraphQL — это язык запросов данных и язык манипулирования данными с открытым исходным кодом для построения веб ориентированных программных интерфейсов.

SOAP — это протокол обмена структурированными сообщениями в распределённой вычислительной среде.

Swagger — это набор инструментов, который позволяет автоматически описывать API на основе его кода.

API — это интерфейс для связи между разными программными продуктами.

Hibernate — это библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения, самая популярная реализация спецификации JPA.

DTO — Data Transfer Object — один из шаблонов проектирования, используется для передачи данных между подсистемами приложения.

Стандарты ANSI SQL — это набор формально определенных рекомендаций языка структурированных запросов (SQL). Американский национальный институт стандартов (ANSI) и Международная организация по стандартизации (ISO) приняли стандарты SQL в 1986 году.

C# — это объектно-ориентированный язык программирования.

.NET — это модульная платформа для разработки программного обеспечения с открытым исходным кодом.

Lombok — это библиотека для сокращения кода в классах и расширения функциональности языка Java.

CRUD — это акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание, чтение, модификация, удаление.

Фреймворк — это набор инструментов, библиотек и правил, который позволяет разработчикам быстро создавать приложения и упрощает процесс разработки. Он представляет собой абстрактную структуру, которая определяет основные компоненты приложения, их взаимодействие и правила работы с ними.

Интерфейс — это средство взаимодействия между пользователем и приложением, которое позволяет пользователю управлять приложением и

получать от него информацию. Интерфейс может быть графическим, текстовым или голосовым, в зависимости от типа приложения и специфики его использования.

Бизнес-логика — это совокупность правил и процессов, которые определяют, как должно работать приложение в соответствии с бизнес-процессами компании. Бизнес-логика описывает логику работы приложения, его функциональность и поведение в различных ситуациях.

Дескриптор — представляет собой протокол, который позволяет перехватывать операции чтения, записи и удаления атрибутов и выполнять дополнительные действия при их выполнении.\

Конфигурация — это особенности конструкции, включая архитектуру, состав и характеристики основных составных частей и вспомогательных (периферийных) средств, а также организацию связей между ними.

Комьюнити — сообщество людей, которых объединяет одна цель, идея, или проблема, над которыми они хотят, могут или соглашаются работать с помощью комьюнити-активности.

Виджет — это графический элемент интерфейса, помогающий пользователю быстрее получить доступ к необходимой информации.

Кроссплатформенность — это способность программного обеспечения работать с несколькими аппаратными платформами или операционными системами.

Java-аннотация — в языке Java специальная форма синтаксических метаданных, которая может быть добавлена в исходный код. Аннотации используются для анализа кода, компиляции или выполнения.

Бин — это объекты, которые являются основой приложения и управляются Spring IoC контейнером.

Шифратор — это логическое устройство, выполняющее логическую функцию — преобразование позиционного n-разрядного кода в m-разрядный двоичный, троичный или k-ичный код.

Имплементация интерфейса — это создание класса, который реализует все методы, определенные в интерфейсе.

Репозиторий — это место, где хранятся и поддерживаются какие-либо данные.

# 2 Анализ предметной области

## 2.1 Анализ рынка

Приложение для продажи билетов на концерты относится к предметной области развлечений и событийной индустрии. Оно нацелено на рынок концертных и музыкальных событий, где пользователи могут искать, выбирать и приобретать билеты на различные концерты и выступления.

Данный рынок является динамичным и популярным. Он включает в себя широкий спектр мероприятий, таких как музыкальные концерты, фестивали, спортивные события, театральные представления и другие развлекательные мероприятия.

В целом, рынок развлечений и событийной индустрии является конкурентным, но также предлагает много возможностей для развития и инноваций в области продажи билетов. Это динамичная сфера, где пользователи всегда в поиске новых и уникальных мероприятий, а приложение для продажи билетов на концерты может предоставить им удобство, выбор и персонализированный опыт.

## 2.2 Определение проблематики

На рынке развлечений и событийной индустрии существуют несколько распространенных проблем, с которыми сталкиваются как организаторы мероприятий, так и покупатели билетов. Некоторые из этих проблем включают:

1. Ограниченная доступность билетов: на популярные мероприятия билеты могут быть распроданы очень быстро, оставляя мало шансов на покупку для заинтересованных покупателей. Это может привести к разочарованию и неудовлетворенным потребностям пользователей.
2. Высокие цены и мошенничество: билеты на некоторые мероприятия могут иметь завышенные цены на вторичном рынке, что делает их недоступными для некоторых покупателей. Кроме того, возможны

случаи мошенничества, когда покупатели приобретают поддельные или недействительные билеты.

3. Отсутствие централизованной информации: поиск информации о различных мероприятиях, таких как расписание, место проведения, артисты и цены билетов, может быть неудобным и времязатратным процессом. Организаторы мероприятий и покупатели нуждаются в централизованной платформе, где они могут найти всю необходимую информацию.
4. Отсутствие персонализации и рекомендаций: покупатели хотят получать персонализированные рекомендации о мероприятиях, основанные на их предпочтениях и интересах. Однако, не всегда доступны инструменты и функции для предоставления таких рекомендаций.
5. Ограниченные возможности обмена и возврата билетов: в случае, если покупатели не могут посетить мероприятие, они могут столкнуться с ограничениями или сложностями при обмене или возврате билетов.

Решение данных проблем может быть связано с разработкой и использованием удобных и надежных приложений для продажи билетов, которые предоставляют широкий выбор мероприятий, информацию, персонализацию, возможности обмена и возврата билетов, а также борьбу с мошенничеством и обеспечение честных цен.

Приложение для продажи билетов на концерты предоставляет удобный и централизованный способ для пользователей находить и приобретать билеты на свои любимые мероприятия. Оно облегчает процесс покупки билетов, предоставляет информацию об артистах, расписании мероприятий и площадках проведения, а также может предлагать персонализированные

рекомендации и дополнительные возможности для зарегистрированных пользователей.

### 2.3 Постановка задачи

Целью данного курсового проекта является разработка мобильного приложения для продажи билетов на различные музыкальные мероприятия. Пользователи данного приложения смогут совершать покупку билетов будучи неавторизованными, либо авторизоваться, пройдя процедуру входа или регистрации. Для авторизованного пользователя доступны дополнительные функции, такие как: просмотр истории заказов, поиск компании на мероприятие, обмен билетов на мероприятие. Для удобства поиска приложение обладает возможностью фильтрации каталога по дате, жанрам, названию мероприятия и артисту.

### 2.4 Основные требования к приложению

Для достижения данных целей приложение должно отвечать следующим требованиям:

1. Приложение должно обладать простым и не перегруженным лишними деталями интерфейсом, с выделяющейся на фоне аналогов цветовой схемой;
2. Приложение должно выполнять ряд основных функциональных задач:
    1) просмотр информации о предстоящих музыкальных мероприятиях,
    2) поиск мероприятий по названию,
    3) покупка билетов на предстоящие музыкальные мероприятия без регистрации,
    4) формирование выборки мероприятий на основе фильтров,

5) поиск компании на выбранное музыкальное мероприятие для зарегистрированных пользователей,

6) обмен билетами с другими посетителями в рамках выбранного мероприятия для зарегистрированных пользователей,

7) просмотр истории заказов в профиле пользователя для зарегистрированных пользователей,

8) регистрация для новых пользователей, открывающая доступ к дополнительным функциям,

9) добавление новых мероприятий и площадок для мероприятий с помощью панели администратора,

10) редактирование существующих мероприятий с уведомлением об поступивших изменениях обладателей билетов с помощью панели администратора.

Для выполнения данных требований необходимо выполнить следующие задачи:

1. Разработка технического задания проекта;
2. Проектирование REST API на языке программирования Java;
3. Реализация ролей:
    1) администратор,
    2) авторизованный пользователь,
    3) неавторизованный пользователь (гость).
4. Реализация функциональных возможностей ролей;
5. Разработка базы данных;
6. Подключение базы данных для хранения данных;
7. Разработка бизнес-логики приложения;

8. Разработка front-end части, включающей в себя:

    1) создание макета дизайна,
    2) реализация мобильного приложения при помощи фреймворка Flutter,
    3) добавление соединения с API сервера,
    4) проведение тестирования проекта.

# 3 Пользовательские истории

## 3.1 Пользовательская история 1

Актёр: Андрей, 22 года, фотограф

Как активный пользователь социальных сетей и общительный человек, Андрей хочет иметь возможность находить единомышленников и новые знакомства, для того чтобы совместно посещать мероприятия, которые ему интересны, и, в перспективе, заводить новых друзей.

## 3.2 Пользовательская история 2

Актёр: Светлана 47 лет, домохозяйка

Как неопытный интернет-пользователь и редкий посетитель концертов, Светлана хочет иметь возможность максимально просто находить необходимое ей мероприятие, не обременяя себя регистрацией и дополнительными функциями, для того чтобы купить билет на интересующее её мероприятие самостоятельно, не прибегая к посторонней помощи.

## 3.3 Пользовательская история 3

Актёр: Пётр, 30 лет, официант.

Петр планирует поездку к родственникам в другой город на пару дней. Но его бюджет строго ограничен, дней для отдыха не так много, а желание устроить родственникам концертную программу огромное.

Ему не хочется листать большое количество страниц с мероприятиями и соотносить их по цене и дате. Он бы хотел сразу сделать выборку событий по необходимым ему параметрам.

### 3.4 Пользовательская история 4

Актёр: Зинаида, 60 лет, пенсионерка

Зинаида частый посетитель различных выступлений. К удивлению, все мероприятия знакомых артистов она уже посетила. Зине хочется на очередной концерт, но с творчеством доступных артистов она не знакома. Женщина считает, что ей бы пригодился сервис для подборки различных событий, которая будет основана на ее интересах, увлечениях, прошлых заказах.

### 3.5 Пользовательская история 5

Актёр: Денис и Анна, 18 лет.

Денис и Аня договорились вместе пойти на концерт их любимого исполнителя. Но по итогу разговора недопоняли друг друга и купили билеты в разные секторы зала. Им хотелось бы сидеть рядом, но перепродавать билеты и покупать их заново неудобно, трудно и времязатратно; тем более если концерт уже завтра или билетов в наличие нет.

### 3.6 Пользовательская история 6

Актёр: Наталья 32 года

Наталья является большим поклонником рок-музыки и постоянно ищет новые концерты для посещения. «Копаться» в длинном списке выступлений, жанры которых ее не интересует, утруждают и сильно выматывают. Девушка хочет иметь возможность фильтрации по жанрам и быстро находить концерты для посещения.

## 4 Продуктовые воронки

### 4.1 Неавторизованный пользователь

1. Открытие сайта - пользователь заходит на сайт, чтобы найти информацию о концертах;
2. Регистрация - пользователь регистрируется на сайте, чтобы:
    1) получить персонализированные рекомендации на концерты,
    2) получить возможность просмотра истории заказов,
    3) получить персонализированные рекомендации на концерты,
    4) получить возможность обмена билетов.
3. Поиск концертов - пользователь ищет концерты по цене, жанру, дате и времени;
4. Выбор билетов - пользователь выбирает количество и тип билетов, которые он хочет приобрести;
5. Оформление заказа - пользователь заполняет форму заказа, указывая свои контактные данные и способ оплаты;
6. Оплата - пользователь производит оплату билетов;
7. Получение билетов - пользователь получает электронные билеты на почту.

### 4.2 Авторизованный пользователь

1. Вход на сайт - пользователь авторизуется на сайте для просмотра информации о мероприятии;
2. Просмотр профиля - пользователь просматривает свой профиль и редактирует информацию о себе;

3. Рекомендации - сайт предлагает пользователю персонализированные рекомендации на концерты, основываясь на его предпочтениях и ранее посещенных мероприятиях;
4. Поиск концертов - пользователь ищет концерты по цене, жанру, дате и времени;
5. Выбор билетов - пользователь выбирает количество и тип билетов, которые он хочет приобрести;
6. Оформление заказа - пользователь заполняет форму заказа, указывая свои контактные данные и способ оплаты;
7. Оплата - пользователь производит оплату билетов;
8. Получение билетов - пользователь получает электронные билеты в приложении;
9. Обмен билетов – пользователь меняет свое место в определенном секторе на другое возможное;
10. Поиск компании – пользователь ищет на конкретное мероприятие возможную компанию или собирает ее сам.

# 5 Обзор аналогов

## 5.1 Kassir.ru

Приложение для покупки билетов на различные мероприятия: концерты, фестивали, спортивные мероприятия, театр, цирк и многое другое. Помимо мобильного приложения существует также и вебсайт. Сервис берет на себя все организационные моменты по продвижению мероприятия и реализации билетов на своей интернет-площадке. На официальном сайте можно посмотреть афишу главных событий в городе и по всей стране с указанием даты и ценового разбега.

Компания появилась на рынке услуг еще в далеком 1999 году. На данный момент онлайн-сервис занимает одну из лидирующих позиций в сегменте. Kassir.ru охватывает не только Россию, но и семь стран ближнего зарубежья.

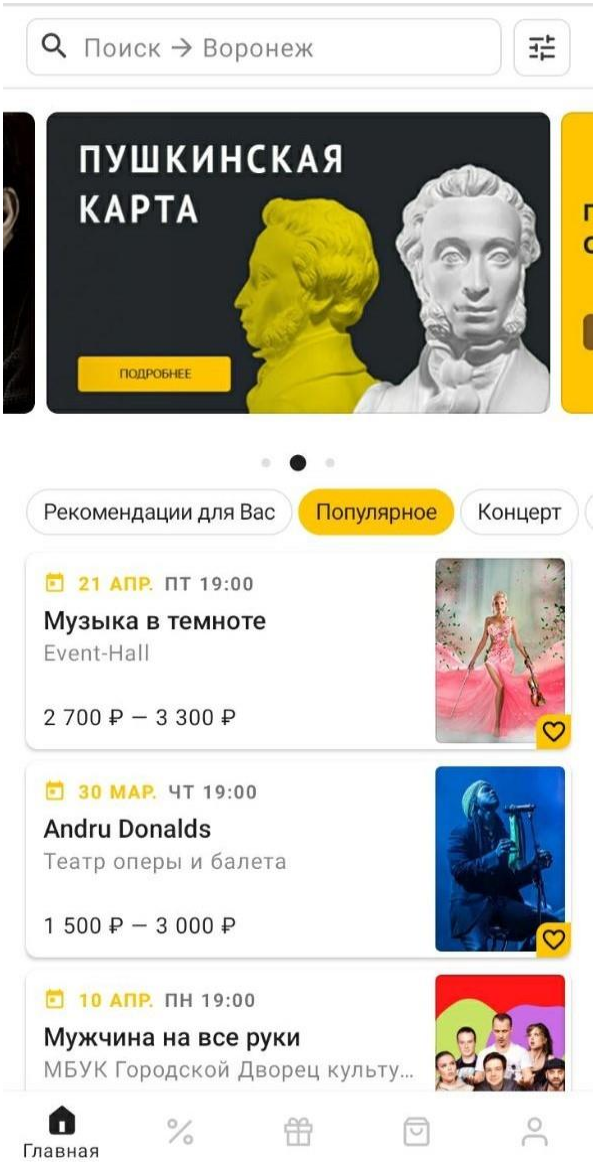


Рисунок 1 - Главная страница

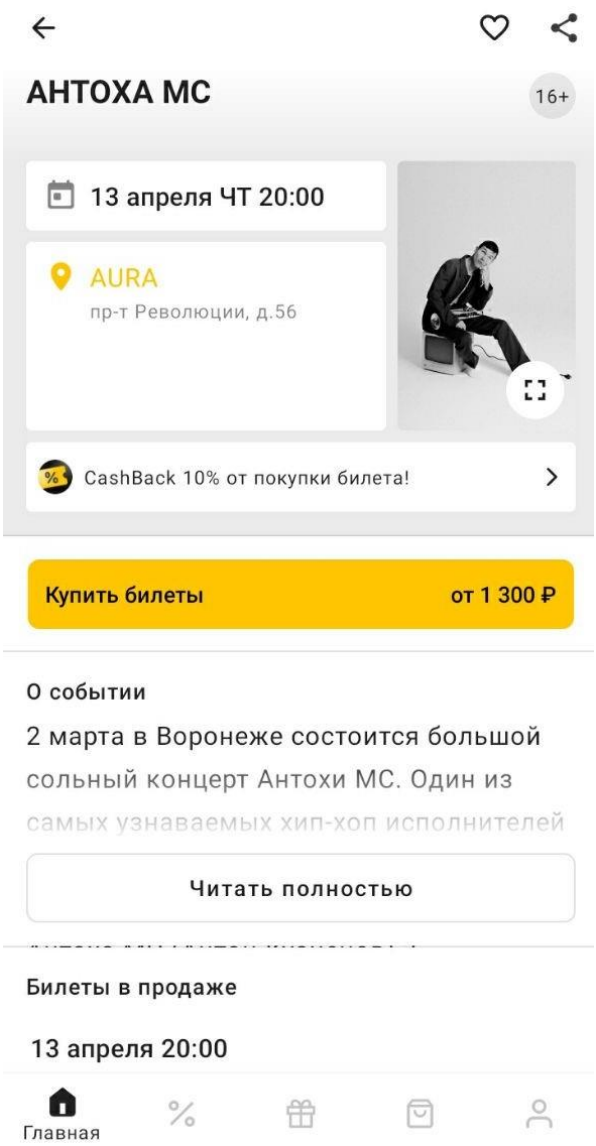


Рисунок 2 - Страница мероприятия

Достоинства:

1. Возможность настроить рекомендации в профиле пользователя;
2. Информативная карточка мероприятия пользователя;
3. Возможность приобретения подарочных сертификатов;
4. Большое количество категорий мероприятий.

Недостатки:

1. Неудобное и контринтуитивное меню фильтров;

2. Малое количество жанровых дескрипторов для мероприятий;

3. Высокий сервисный сбор (10%) при покупке билетов в приложении.

### 5.2 Горбилет

«Горбилет» — сервис по бронированию и продаже билетов со скидкой. Здесь можно найти билеты на концерты, спектакли, экскурсии, выставки и многое другое. Проект был запущен в 2018 году как группа в ВКонтакте, через некоторое время появился веб-сайт и мобильное приложение.

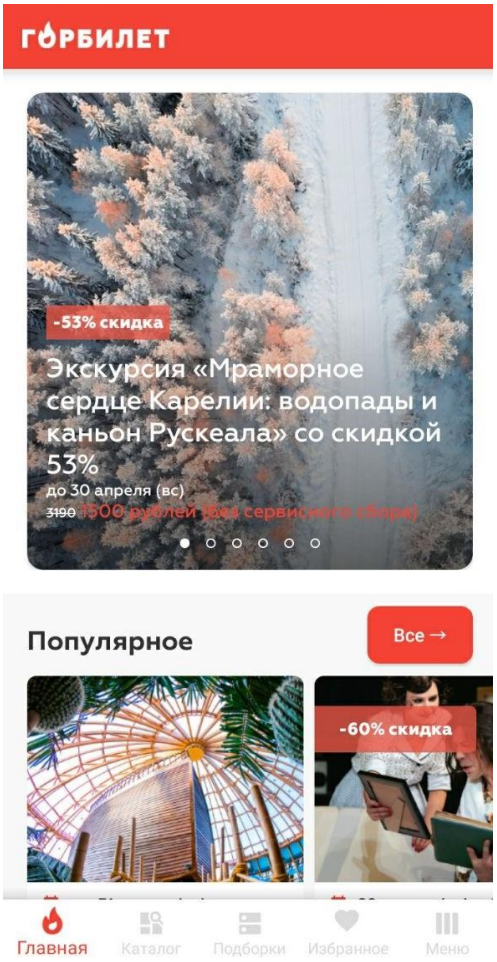


Рисунок 3 - Главная страница


Рисунок 4 - Страница мероприятия

Достоинства:

1. Наличие тематических подборок с мероприятиями;
2. Возможность добавлять мероприятия в избранное;
3. Демонстрация недавно посещённых карточек мероприятий;
4. Подробное описание процесса покупки и оплаты на странице мероприятия.

Недостатки:

1. На выбор доступны только 2 города – Москва и Санкт-Петербург;
2. Огромный размер карточек в каталоге;
3. Отсутствие тёмной темы;

4. Малое количество жанровых дескрипторов для мероприятий;
5. Низкая производительность приложения;
6. Отсутствие информации о площадке.

### 5.3 Суфлёр

«Суфлёр» – это персональный рекомендательный сервис и навигатор по драматическим и музыкальным театрам. В его афише вся театральная Москва и весь театральный Питер. Приложение выпущено в 2018 году разработчиком Souffleur Ltd. На данный момент существует только мобильная версия приложения.

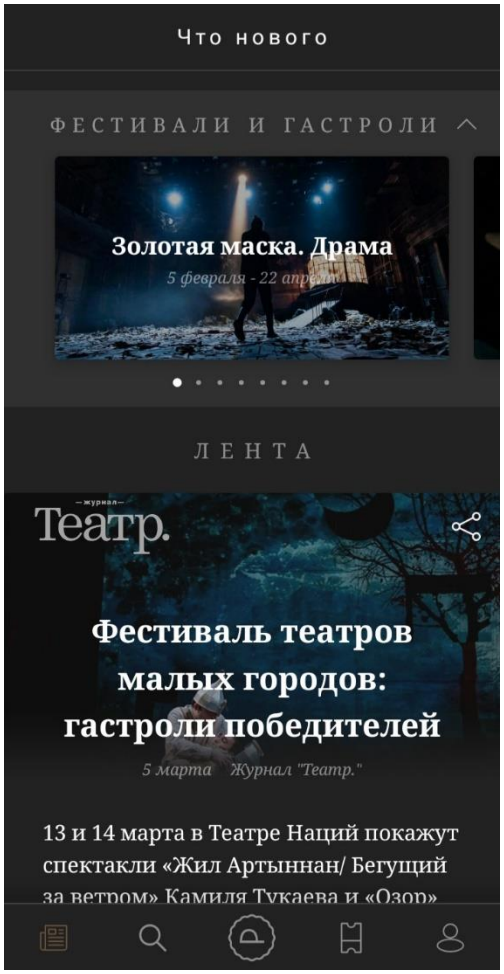


Рисунок 5 - Главная страница

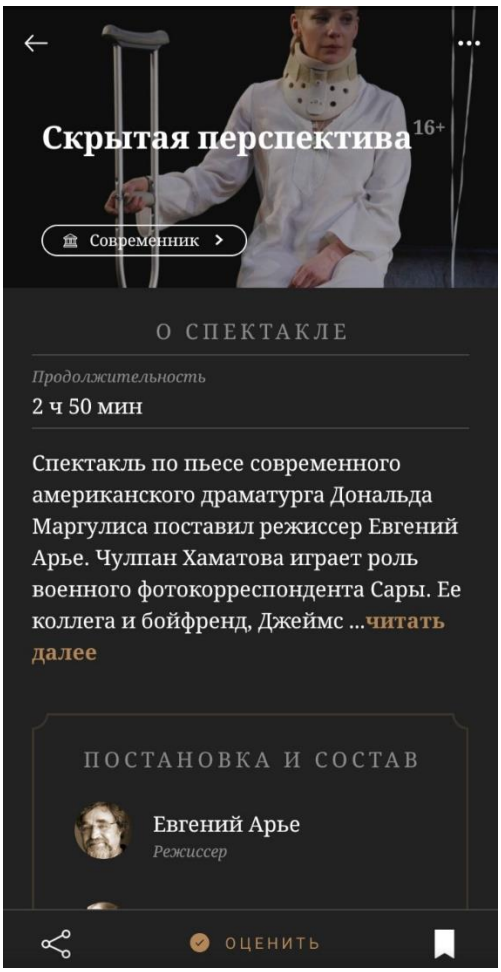


Рисунок 6 - Страница мероприятия

Достоинства:

1. Наличия календаря в качестве фильтра;
2. Рекомендации для новых пользователей на основе трёх вопросов;
3. Возможность оценки мероприятий;
4. Обзоры предстоящих мероприятий.

Недостатки:

1. На выбор доступны только 2 города – Москва и Санкт-Петербург;
2. Малое количество фильтров - только цена и наличие доступных билетов;

3. Выбор мероприятий ограничен театрами;
4. «Отсутствие» каталога в привычном понимании данного термина;
5. Отсутствие веб-версии.

### 5.4 Goldstar

Это мобильное приложение фокусируется на том, чтобы помочь пользователям приобрести билеты на различные мероприятия со скидкой. В список доступных мероприятий входят такие разделы как театр, музыка, комедия, спорт, прочие развлечения (такие как мероприятия в зоопарках и океанариумах).

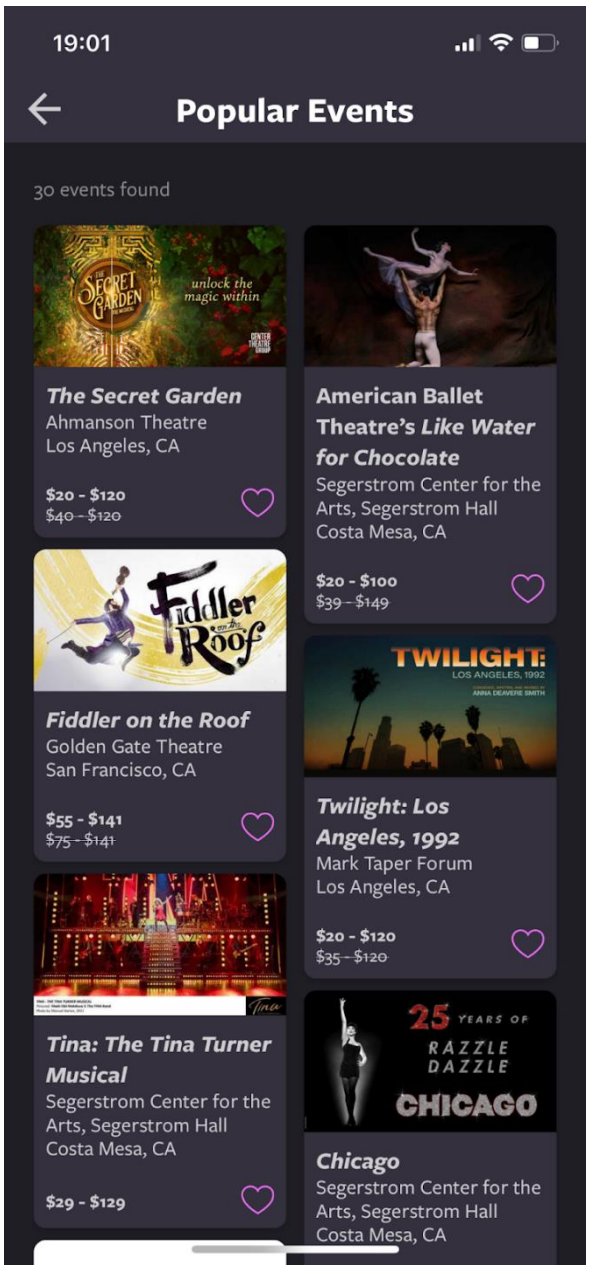


Рисунок 7 - Главная страница

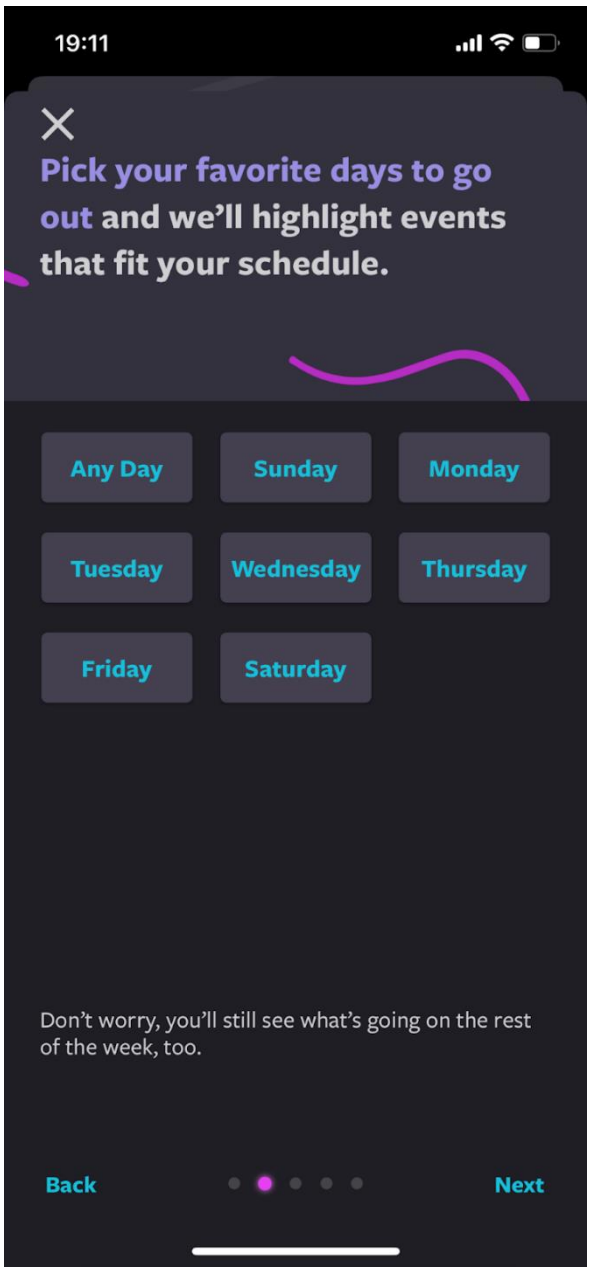


Рисунок 8 - Страница с фильтрами

Достоинства:

1. Возможность указать свои предпочтения (небольшой выбор);
2. Возможность добавить мероприятие в избранное;
3. Возможность построить маршрут до площадки проведения.

Недостатки:

1. Чтобы приобрести билет или добавить в избранное необходимо зарегистрироваться;
2. Небольшое количество доступных мероприятий несмотря на большое количество категорий.

### 5.5 TodayTix

Данное мобильное приложение создано для продажи билетов на театральные представления. Выбрать мероприятие можно более чем в 15 городах мира (в основном в северной Америке и Австралии), и купить билет по эксклюзивной цене.

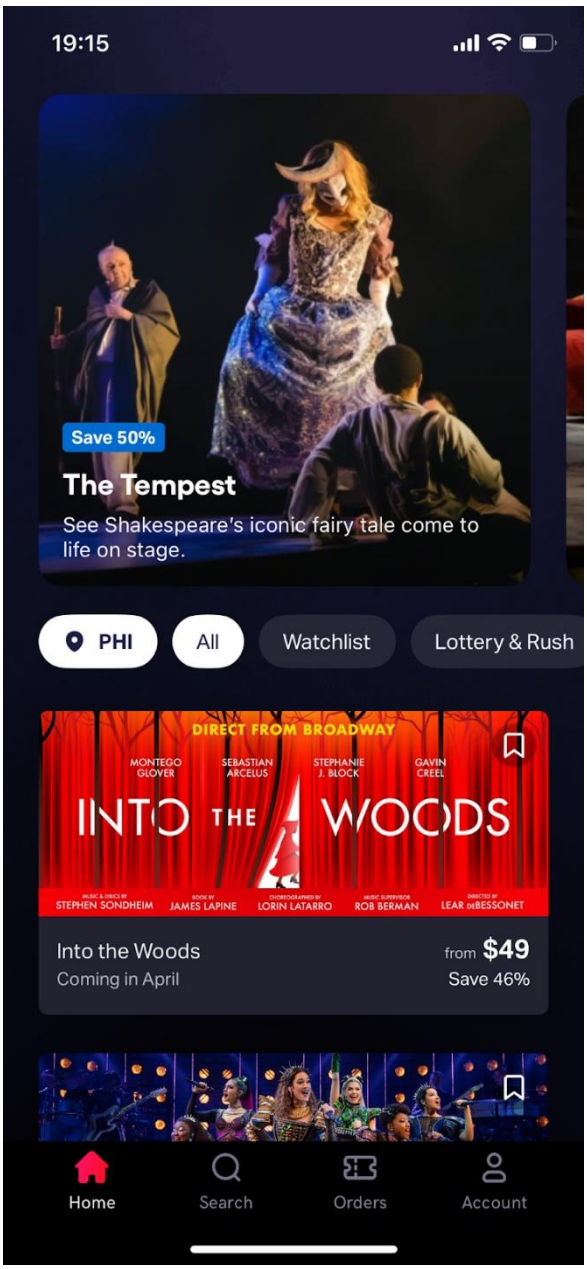


Рисунок 9 - Главная страница

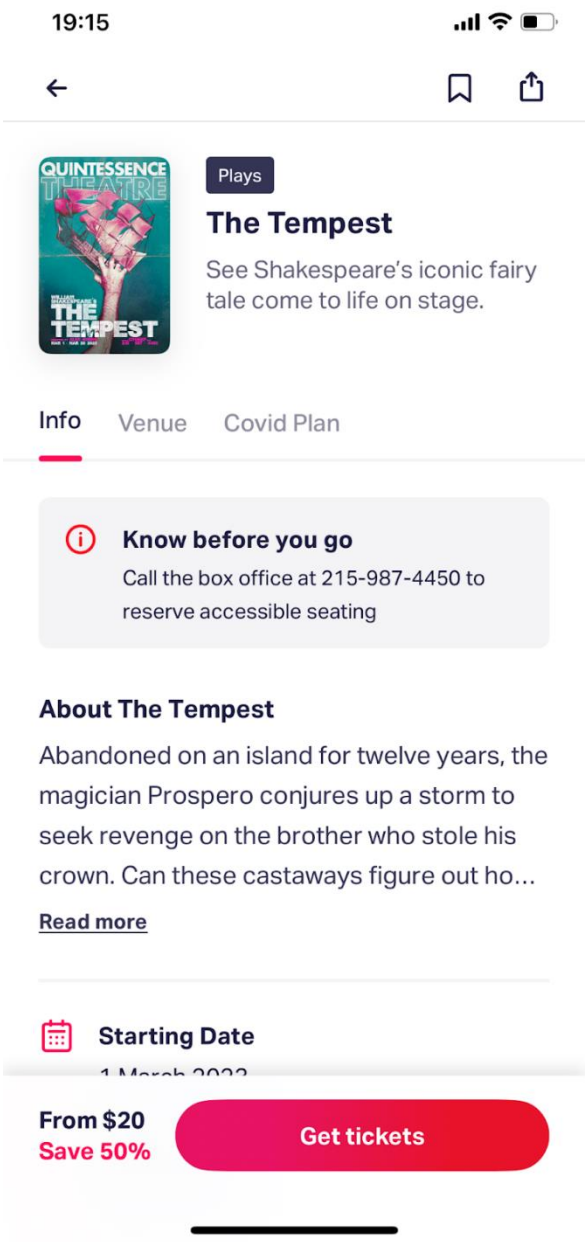


Рисунок 10 - Страница мероприятия

Достоинства:

1. Предоставляет возможность купить скидочную подарочную карту для покупки билета в приложении;
2. Возможность построить маршрут до площадки проведения.

Недостатки:

1. Чтобы приобрести билет надо регистрироваться;
2. Основные страницы выполнены в темных тонах, а при переходе на конкретное мероприятие тема резко меняется на светлую;
3. Отсутствие фильтров.

# 6 Диаграммы

## 6.1 Диаграмма прецедентов (Use-case diagram)

Данная диаграмма демонстрирует пользователей системы и доступные им действия.

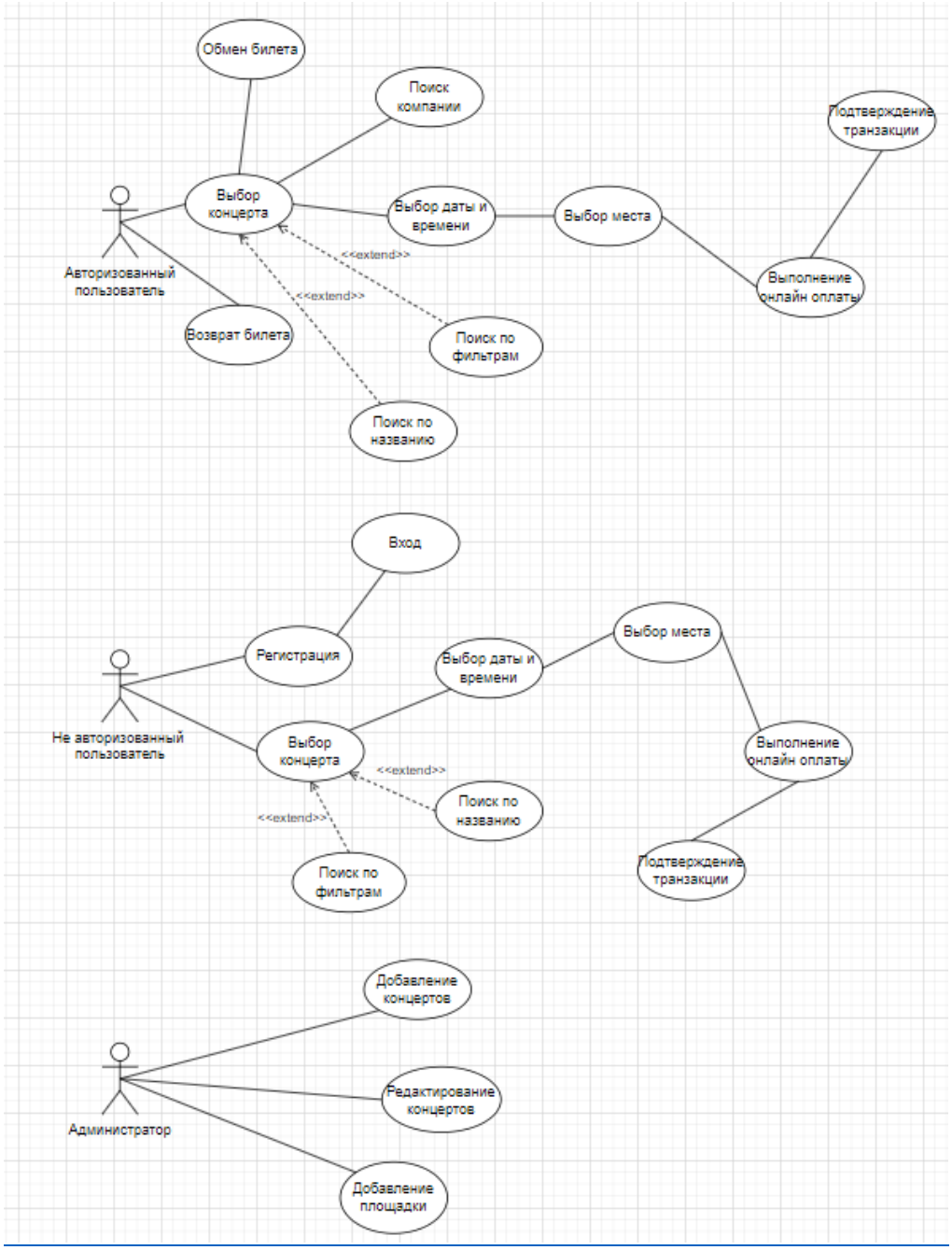


Рисунок 11 - Диаграмма прецедентов

### 6.2 Диаграмма классов (Class diagram)

На данной диаграмме отображены классы системы, их методы и атрибуты с типами данных. Помимо этого, показано взаимодействие между классами посредством связей.

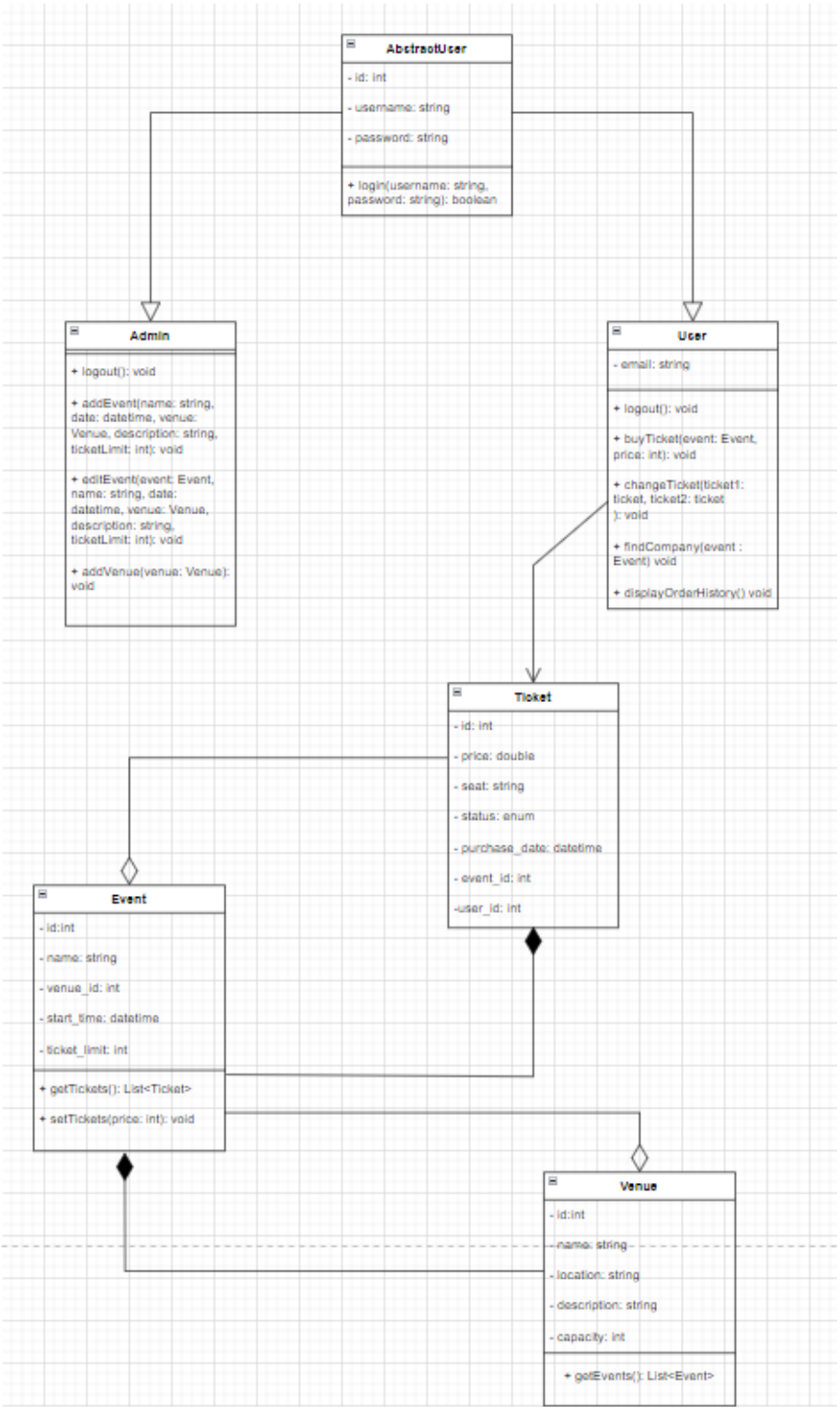


Рисунок 12 - Диаграмма классов

### 6.3 Диаграмма последовательности (Sequence diagram)

Данные диаграммы отражают взаимодействие различных частей системы между собой для выполнения функции, а также показывает последовательность действий, которые проводят к завершению этих функций.

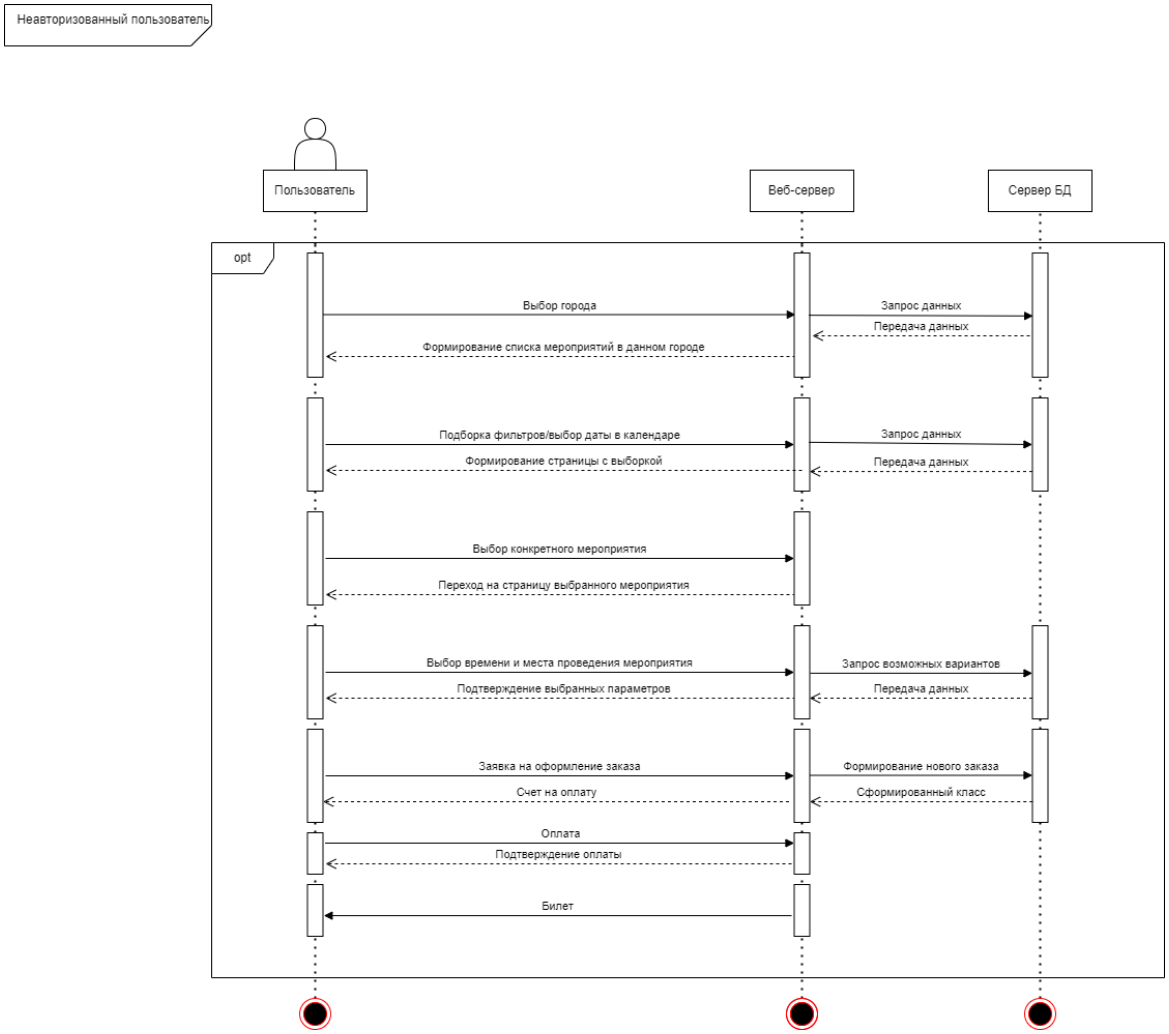


Рисунок 13 - Диаграмма последовательности для неавторизованного пользователя

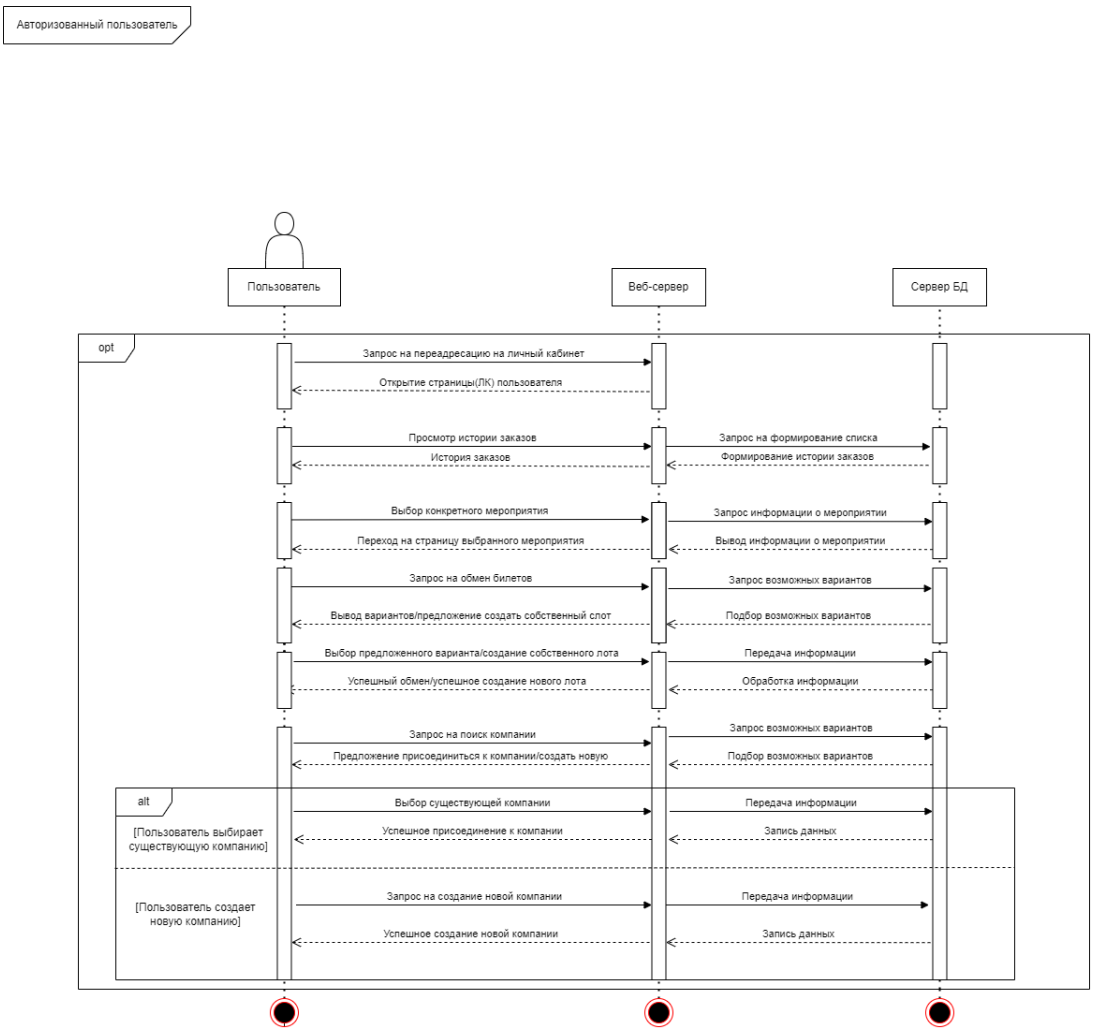


Рисунок 14 - Диаграмма последовательности для авторизованного пользователя

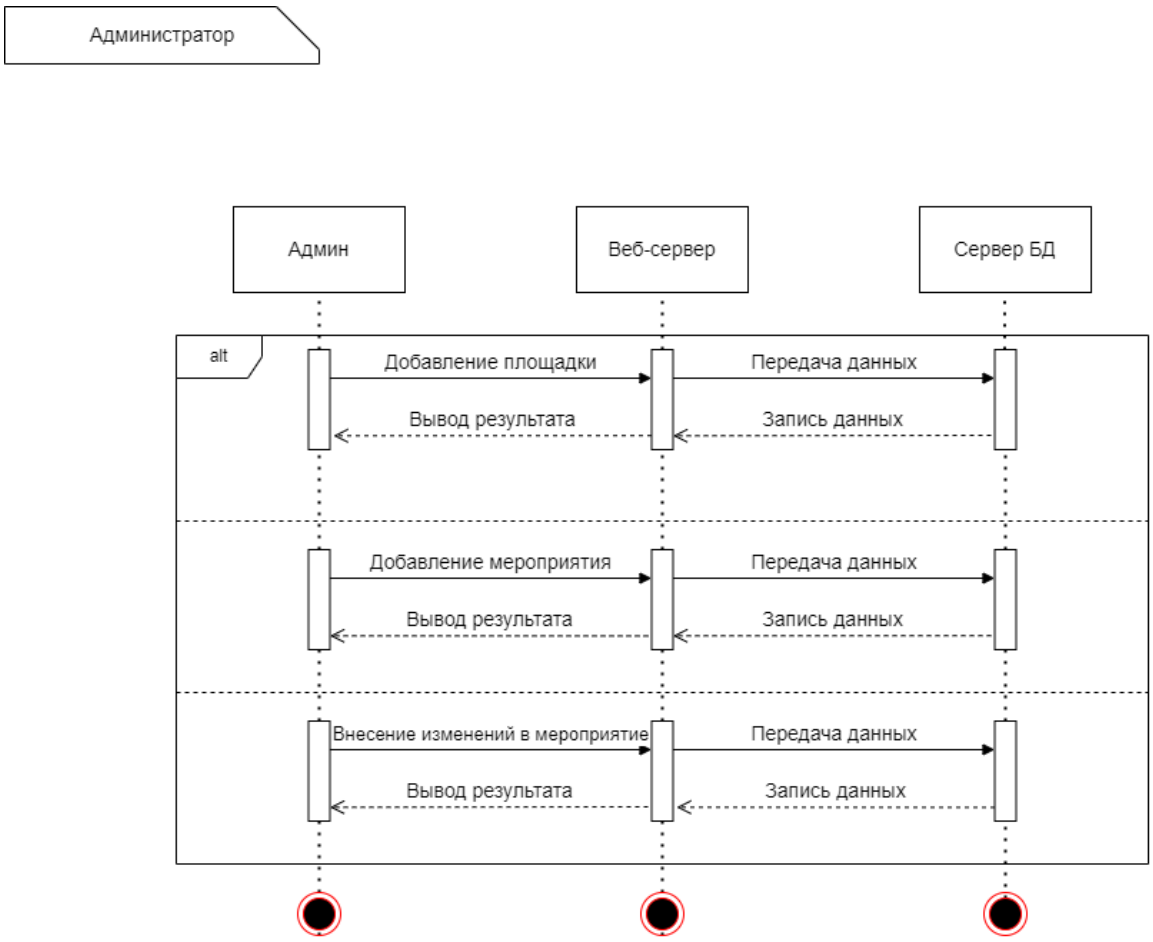


Рисунок 15 - Диаграмма последовательности для администратора

### 6.4 Диаграмма активности (Activity diagram)

Данная диаграмма демонстрирует возможные действия пользователей и их последовательность.

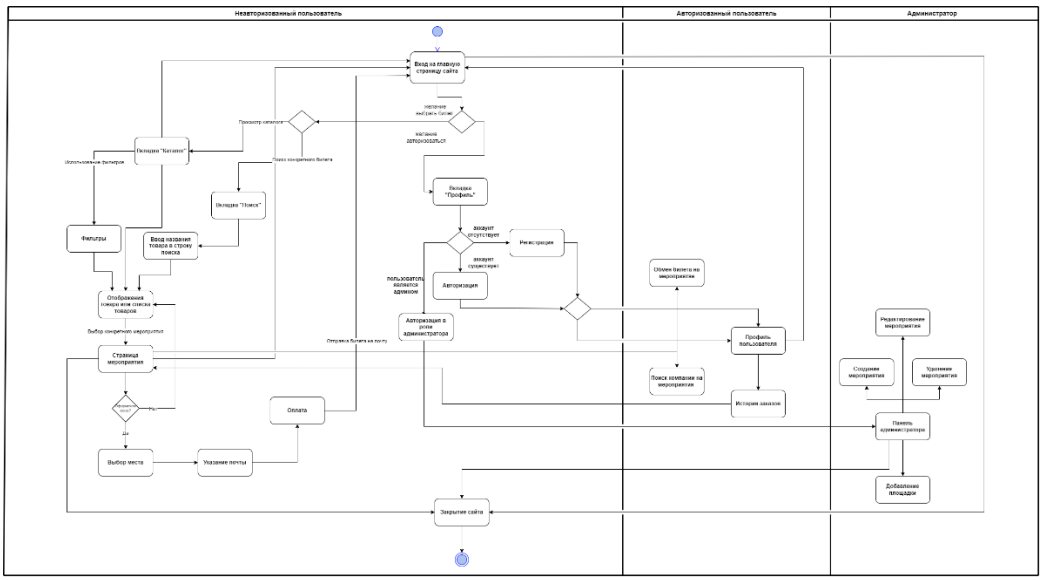


Рисунок 16 - Диаграмма активности

## 6.5 Диаграмма развёртывания (Deployment diagram)

В данной диаграмме отражены основные программные компоненты, способы их развёртывания и общая конфигурация проектируемой программной системы.

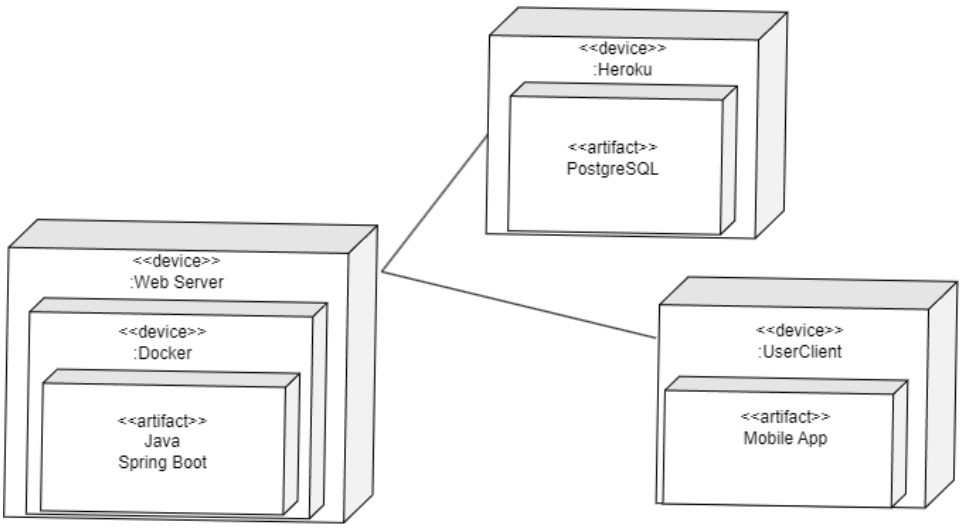


Рисунок 17 - Диаграмма развёртывания

## 6.6 Диаграмма сотрудничества (Collaboration diagram)

В данной диаграмме отображено соединение и взаимодействие объектов между собой в системе.


Рисунок 18 - Диаграмма сотрудничества

## 6.7 Диаграмма объектов (Object diagram)

Диаграмма объектов иллюстрирует сущности, используемые данной системой.

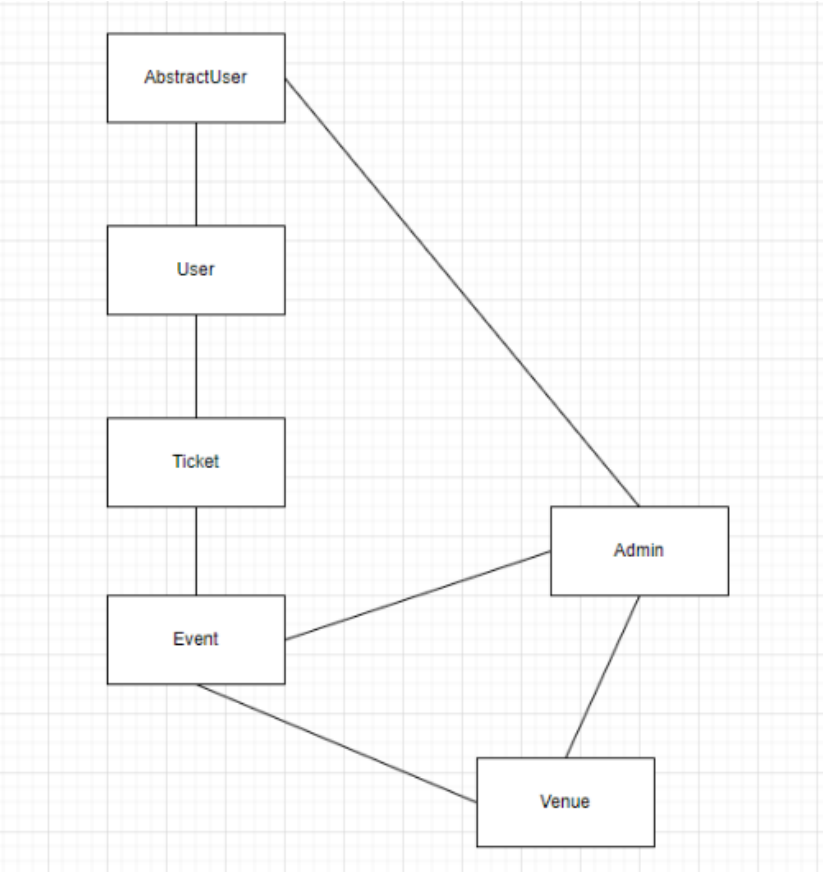


Рисунок 19 - Диаграмма объектов

## 6.8 Диаграмма состояний (Statechart diagram)

Диаграммы состояний отображают разрешенные состояния и переходы, а также события, которые влияют на эти переходы.

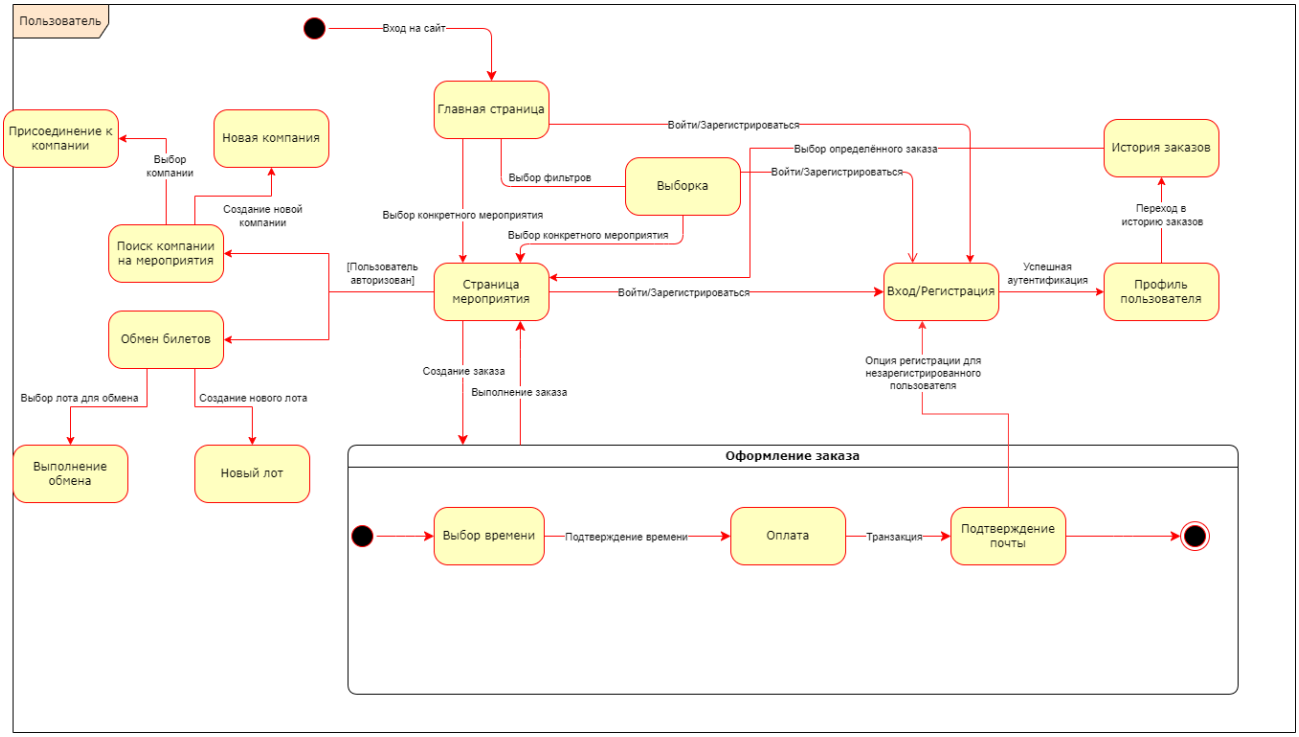


Рисунок 20 - Диаграмма состояний для пользователя

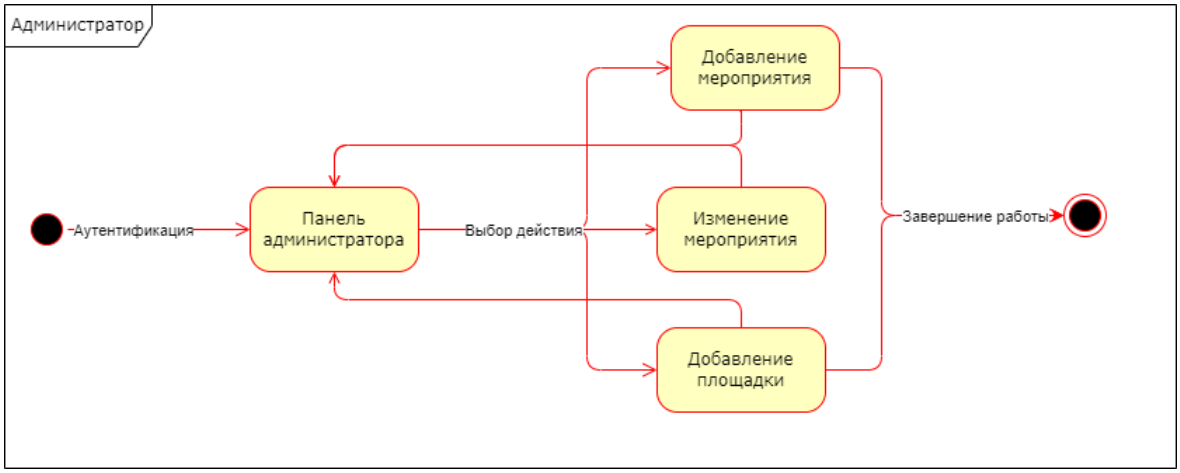


Рисунок 21 - Диаграмма состояний для администратора

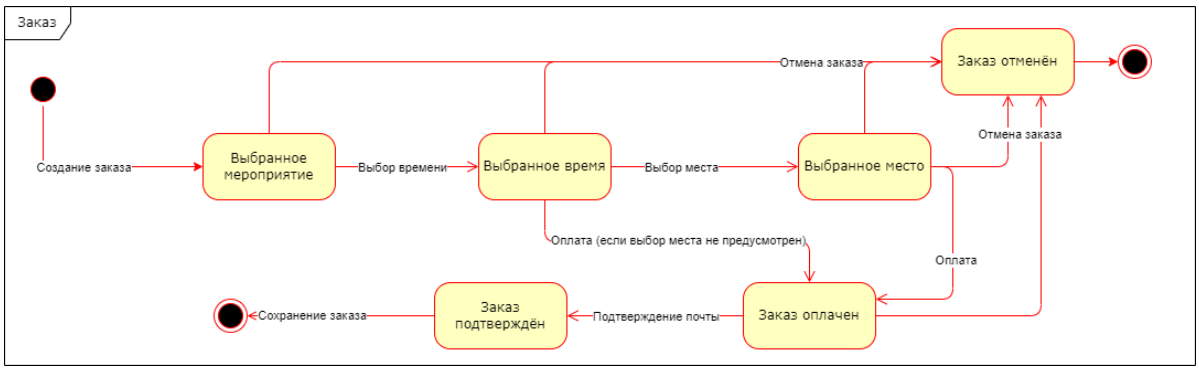


Рисунок 22 - Диаграмма состояний для заказа

### 6.9 IDEF0 Диаграмма

IDEF0 - Методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов.

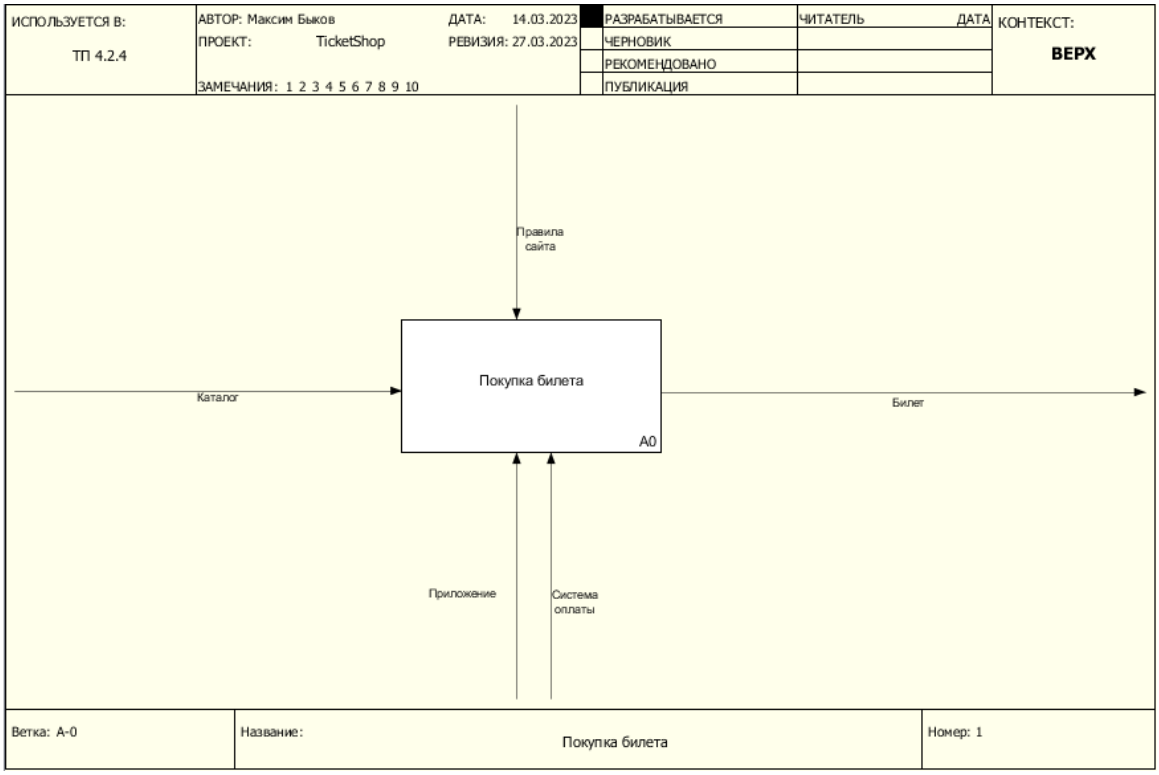


Рисунок 23 - Система приложения (Уровень 0)

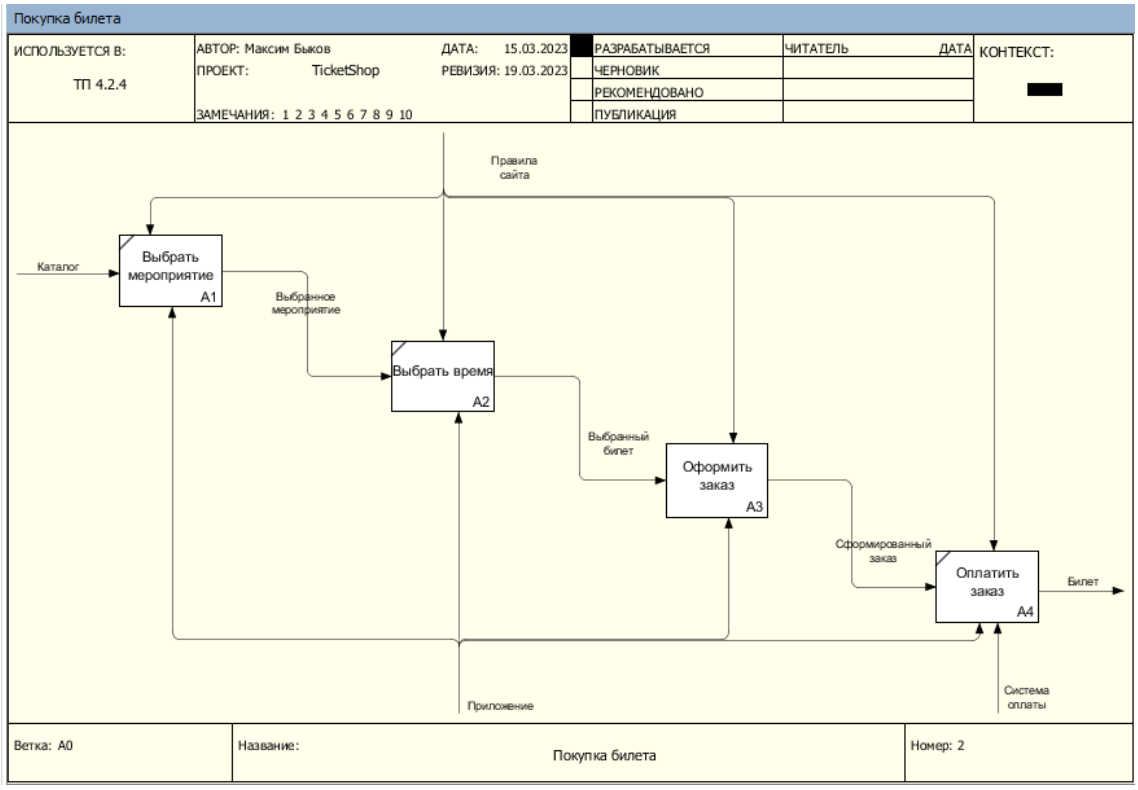


Рисунок 24 - Система приложения (Уровень 1)

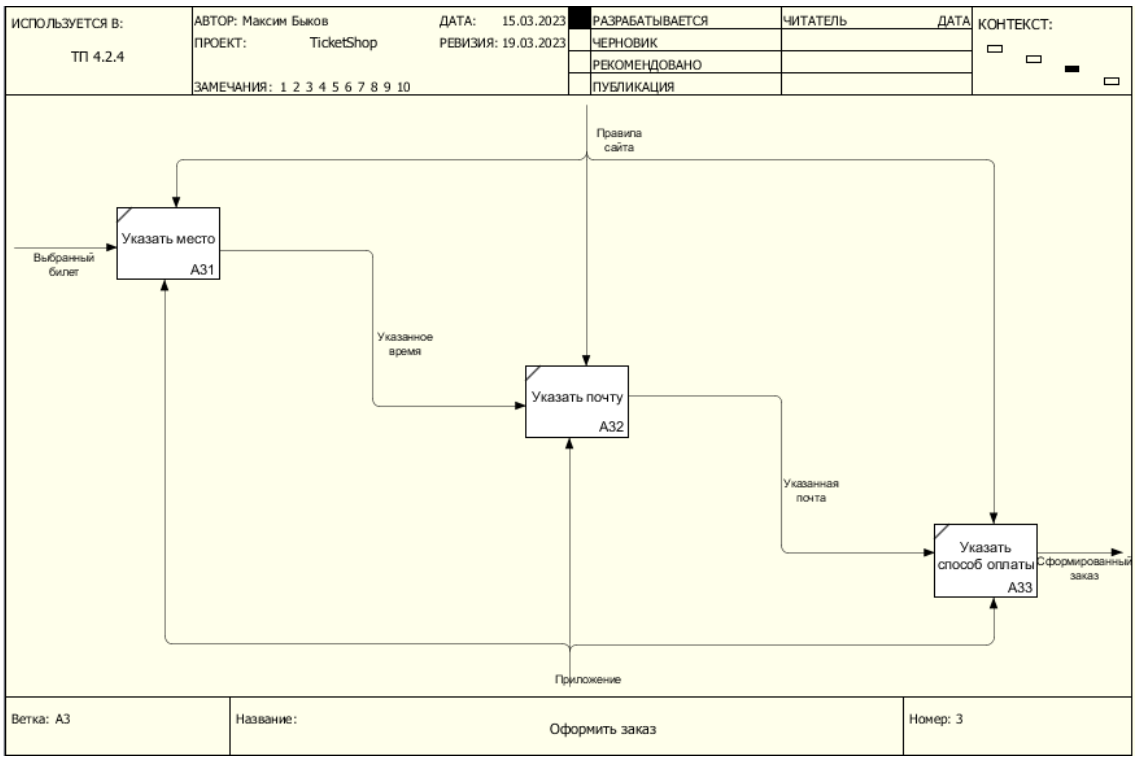


Рисунок 25 - Система приложения (Уровень 2)

# 7 Реализация приложения

## 7.1 Средства реализации

Серверная часть:

1. Java (версия 8 и выше);
2. Spring Boot;
3. Spring Security;
4. СУБД PostgreSQL.

Клиентская часть:

1. Android 12 или новее;
2. Flutter.

Язык программирования Java является отличным выбором для разработки мобильных приложений благодаря своей простоте, безопасности, высокой производительности и множеству инструментов и библиотек.

Spring Boot – фреймворк разработки приложений на Java. Он обеспечивает удобные инструменты и функциональность для создания серверной части приложения, обрабатывая HTTP-запросы, управляя бизнес-логикой и взаимодействуя с базой данных. Он позволяет создавать масштабируемые и гибкие приложения с использованием модульной структуры.[3]

Spring Security – это модуль фреймворка Spring Boot для обеспечения безопасности в приложениях на языке Java. Среди его возможностей стоит выделить: защита от уязвимостей, аутентификация и авторизация, интеграция с другими технологиями, поддержка множества протоколов безопасности.[4]

СУБД PostgreSQL – мощная реляционная база данных с открытым исходным кодом. Она предоставляет надежное хранение данных для приложения и обладает расширенными возможностями, включая поддержку сложных запросов и транзакций. PostgreSQL обеспечивает эффективную работу с данными и обеспечивает надежность и целостность информации.

Flutter – это открытая платформа для разработки мобильных приложений. Flutter предоставляет широкие возможности для создания мобильных приложений, включая работу с анимациями, базами данных, сетевыми запросами и многим другим.[1]

## 7.2 Разбор аналогов технологий разработки

### 7.2.1 Существующие аналоги Java Spring

На рынке существует множество аналогичных фреймворков для разработки серверной части приложений. Далее перечислен перечень фреймворков, которые были рассмотрены при выборе технологического стека для написания бек-энда приложения в качестве альтернативы Spring Boot:

1. Django — это свободный фреймворк для разработки быстрых и безопасных веб-приложений и сайтов на языке Python. Использует шаблон проектирования MVC. Является популярным выбором при реализации серверной части мобильных приложений благодаря простоте освоения и написания кода, а также высокой безопасности серверной части. Ключевым минусом фреймворка является его монолитность по сравнению с Java Spring Boot, который значительно качественнее оптимизирован для работы с проектами данного масштаба.[2]

2. Flask — это облегченный фреймворк для стандартных функций, написанный на Python. Flask считается лучшим веб-фреймворком для создания легковесных веб-приложений и небольших статических сайтов. Главный недостаток - отсутствие ORM, что усложняет взаимодействие с базой данных.[2]

3. Node.js — это кроссплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык JavaScript. Обладает широким спектром инструментов для разработки серверной части приложения, а также обширным комьюнити, поддерживающим развитие фреймворка. Однако Java Spring Boot предоставляет более современный и гибкий подход к разработке приложений, с упором на безопасность и удобство.[2]

### 7.2.2 Преимущества Java Spring Boot

1. Простота использования: Java Spring Boot предлагает удобные и простые инструменты для разработки приложений, позволяя разработчикам быстро начать работу и ускорить процесс разработки;
2. Гибкость и модульность: Java Spring Boot предлагает модульную структуру, которая позволяет разработчикам выбирать только необходимые компоненты и интегрировать их в свои приложения. Это делает фреймворк гибким и масштабируемым;
3. Широкая экосистема: Java Spring Boot имеет обширную экосистему, включая множество сторонних библиотек, инструментов и ресурсов. Это облегчает разработку и расширение приложений с использованием уже существующих компонентов и решений;
4. Активное сообщество: Java Spring Boot имеет большое и активное сообщество разработчиков, которое обеспечивает поддержку, обновления и разработку новых функций. Это обеспечивает надежность и актуальность фреймворка.

### 7.2.3 Существующие аналоги PostgreSQL

На рынке существует несколько аналогичных реляционных баз данных, которые могут быть рассмотрены в качестве альтернативы PostgreSQL. Некоторые из них включают:

1. MySQL: MySQL — это популярная реляционная база данных с открытым исходным кодом. Она обладает хорошей производительностью, широкой поддержкой и распространенностью. Однако PostgreSQL предлагает более расширенный набор функций, включая поддержку сложных запросов, транзакций, полнотекстового поиска, географических данных и многого другого.[7] PostgreSQL также имеет более строгую совместимость со стандартами ANSI SQL;

2. Oracle Database: Oracle Database — это мощная коммерческая реляционная база данных. Она предлагает высокую производительность, масштабируемость и надежность. Однако PostgreSQL является бесплатной и открытой альтернативой, что может быть привлекательным с экономической точки зрения. Кроме того, PostgreSQL имеет более широкую поддержку для различных операционных систем и платформ.[7]

### 7.2.4 Преимущества PostgreSQL

1. Расширенные возможности: PostgreSQL предлагает богатый набор функций, включая поддержку сложных запросов, транзакций, географических данных, полнотекстового поиска, JSON и других расширений. Это делает PostgreSQL мощным инструментом для разработки разнообразных приложений;
2. Открытый исходный код: PostgreSQL является бесплатной и открытой базой данных, что делает его доступным и привлекательным с экономической точки зрения. Он также имеет активное сообщество разработчиков, обеспечивающее поддержку, обновления и разработку новых функций;
3. Платформонезависимость: PostgreSQL поддерживается на различных операционных системах и платформах, включая Linux, Windows, macOS и другие. Это обеспечивает гибкость в выборе окружения развертывания;
4. Хорошая масштабируемость и производительность: PostgreSQL обладает возможностями горизонтального и вертикального масштабирования, что позволяет адаптировать базу данных к растущим потребностям приложения. Он также имеет эффективный планировщик запросов и оптимизатор, обеспечивающий хорошую производительность запросов.

### 7.2.5 Существующие аналоги Flutter

Flutter — это фреймворк для разработки мобильных и веб-приложений, который обладает рядом особенностей.[1] В настоящее время на рынке существуют несколько аналогов Flutter, некоторые из которых включают:

1. React Native: React Native — это фреймворк для разработки мобильных приложений, который использует JavaScript и React для создания переносимых приложений. Он позволяет разработчикам использовать один код для создания приложений для разных платформ. Однако Flutter предлагает более высокую производительность и нативный интерфейс, так как он компилируется в нативный код и не требует моста для связи с нативными компонентами;

2. Xamarin: Xamarin — это платформа разработки мобильных приложений, которая использует C# и .NET для создания приложений для разных платформ. Она позволяет разработчикам использовать общий код и имеет хорошую интеграцию с экосистемой Microsoft. Однако Flutter обладает более дружелюбным интерфейсом разработки, а также более широким сообществом разработчиков.

### 7.2.6 Преимущества Flutter

1. Высокая производительность: Flutter компилируется в нативный код для каждой платформы, что обеспечивает более высокую производительность и быстродействие приложений;

2. Отсутствие моста связи с нативными компонентами также уменьшает задержку и улучшает отзывчивость пользовательского интерфейса;

3. Однородный нативный интерфейс: Flutter позволяет создавать приложения с нативным интерфейсом для каждой платформы, что обеспечивает лучшую интеграцию и пользовательский опыт. Он

также предлагает богатый набор готовых виджетов и стилей, что упрощает создание красивого и современного дизайна интерфейса;

4. Быстрая разработка и горячая перезагрузка: Flutter обладает механизмом горячей перезагрузки, который позволяет разработчикам мгновенно видеть результаты изменений в коде без необходимости перекомпиляции всего приложения. Это сокращает время разработки и повышает производительность разработчиков;
5. Кроссплатформенность: Flutter позволяет создавать приложения для разных платформ, включая iOS, Android, веб и даже настольные приложения. Одним кодом можно охватить несколько платформ, что экономит время и ресурсы разработки;
6. Активное сообщество: Flutter имеет активное и быстрорастущее сообщество разработчиков, что обеспечивает доступ к обучающим материалам, ресурсам и поддержке. Это делает процесс разработки более удобным и эффективным.

### 7.3 Разработка Frontend

Для реализации клиентской части приложения был выбран Flutter — этот фреймворк позволяет разрабатывать кроссплатформенные приложения, что экономит ресурсы разработки, так как одним кодом можно охватить несколько платформ.

Во Flutter также есть ядро Flutter Engine, которое использует собственный растеризатор Skia при рендеринге пользовательского интерфейса. а также принцип "один виджет - одна отрисовка", благодаря которому перерисовывается только минимально необходимая часть, все это это позволяет быстро обновлять пользовательский интерфейс приложения. Ядро также компилирует Dart-код в бинарный код, что способствует повышению скорости работы приложения.

Таким образом, пользователь получает эффективное приложение с привычным для своей платформы внешним видом.

Архитектурный подход, который был выбран для клиентского приложении (Bloc), помогает разделить логику и представление клиента. В блочной архитектуре UI запускает событие, Bloc считывает событие и реализует бизнес-логику, а по завершении генерирует новое состояние для UI.

### 7.4 Разработка Backend

Серверная часть приложения была написана на языке Java с использованием фреймворка Spring Boot. Данный фреймворк обеспечивает разбиение проекта на Maven-модули по слоям приложения. Каждый модуль организован в виде отдельного пакета. Структура приложения и иерархия модулей приведены на рисунке 26.

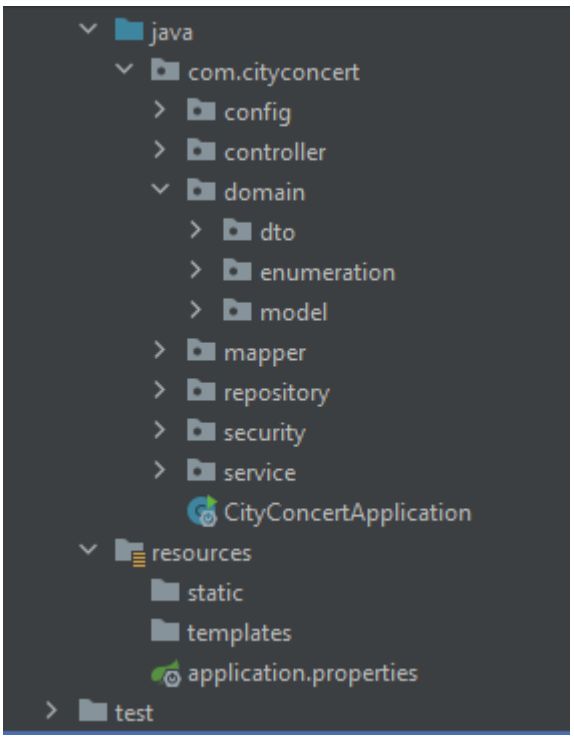


Рисунок 26 - Схема проекта серверной части приложения

При разработке серверной части приложения были реализованы следующие модули:

1. Модуль приложения (Application module). Это основной модуль, в котором находится главный класс приложения с аннотацией @SpringBootApplication, являющийся стартовой точкой приложения. и файл конфигурации application.properties, содержащий параметры для подключения сервера к базе данных и почтовому сервису gmail. Для безопасности приложения и упрощения его развёртывания в различных окружениях ключевые параметры, такие как пароли, имена пользователя, порты и названия базы данных, заменены на переменные окружения (environmental variables).

2. Модуль контроллеров (Controller Module). В этом модуле находятся классы контроллеров, отвечающие за обработку HTTP-запросов и взаимодействие с клиентом (браузером). Контроллеры содержат

аннотации, такие как @RestController и @RequestMapping, для определения эндпоинтов и обработки запросов.

3. Модуль конфигурации (Config Module). В этом модуле находятся классы конфигурации Spring, которые содержат настройки и бины, связанные с приложением. В данном приложении представлен следующими классами:

    1) SecurityConfig, в котором определены и настроены основные компоненты Spring Security для обеспечения безопасности серверной части приложения, в том числе встроенный шифратор паролей BCryptPasswordEncoder, позволяющий хранить пароли в базе данных в зашифрованном виде, и SecurityFilterChain – набор фильтров, определяющий уровень доступа в системе для пользователей с различными ролями,

    2) SpringFoxConfig, в котором происходит настройка Swagger API,

    3) Constants, в котором хранятся необходимые для функционирования системы константы, такие регулярные выражения для паролей, почт и имён пользователей.

4. Модуль моделей (Model Module). Этот модуль содержит классы моделей данных, которые отображают структуру таблиц в базе данных и представляют объекты бизнес-логики. Модели аннотированы аннотациями JPA, такими как @Entity и @Table, а также аннотациями библиотеки Lombok, генерирующими геттеры, сеттеры и конструкторы класса при компиляции. При работе с базой данных использовался фреймворк Hibernate. Он обеспечивает автоматическое отображение классов и свойств на таблицы и столбцы в базе данных. Помимо самих моделей классов также используются DTO – Data Transfer Objects – специальные классы,

используемые для обмена данными между клиентской и серверной частями приложения. Большинство DTO классов представляют из себя модификации существующих моделей, однако существуют и специализированные классы (например, DTO для регистрации нового пользователя).

5. Модуль репозиториев (Repository Module). В этом модуле находятся классы репозиториев, отвечающие за взаимодействие с базой данных. Репозитории используют Spring Data JPA и содержат аннотацию @Repository. Они предоставляют методы для выполнения операций CRUD (создание, чтение, обновление, удаление) и других запросов к базе данных.

6. Модуль сервисов (Service Module). Этот модуль содержит бизнес-логику приложения, которая обрабатывает запросы, полученные от контроллеров. Здесь находятся интерфейсы сервисов, а также имплементации данных интерфейсов – классы, помеченные аннотацией @Service, которые выполняют определенные операции и взаимодействуют с репозиториями.

7. Модуль маппингов (Mapping Module). Этот модуль предназначен для конвертации объектов DTO в объекты моделей и наоборот. Конвертация производится с помощью библиотеки MapStruct, которая самостоятельно генерирует имплементацию мапперов на основе конфигурации, указываемой в соответствующих интерфейсов.

8. Модуль тестирования (Testing Module). Этот модуль отвечает за тестирование приложения с помощью юнит-тестов.

### 7.5 Взаимодействие серверной и клиентской частей

В чем преимущества архитектурного подхода REST API по сравнению с другими архитектурами?

1. Простота и легковесность: REST API основан на простых и широко применяемых протоколах, таких как HTTP. Он не требует использования сложных протоколов или специфических библиотек, что делает его легким в использовании и понимании;

2. Гибкость и масштабируемость: REST API позволяет создавать гибкие и масштабируемые системы. Он разделяет клиентскую и серверную части, что позволяет им работать независимо друг от друга. Сервер может предоставлять данные в различных форматах (например, JSON или XML), что делает его гибким для различных клиентских приложений;

3. Кэширование: REST API поддерживает кэширование данных на стороне клиента, что позволяет улучшить производительность и снизить нагрузку на сервер. Клиент может кэшировать полученные данные и использовать их повторно без необходимости сетевого запроса к серверу.

4. Прозрачность: REST API обеспечивает прозрачность взаимодействия между клиентом и сервером. Клиент и сервер могут обмениваться данными, используя стандартные HTTP методы (GET, POST, PUT, DELETE), что делает коммуникацию понятной и предсказуемой.

5. Широкая поддержка: REST API имеет широкую поддержку в различных языках программирования и платформах. Это делает его доступным для разработчиков с различным опытом и предпочтениями.

Однако, следует отметить, что выбор архитектурного подхода зависит от требований конкретного проекта. В некоторых случаях, другие

архитектурные подходы, такие как GraphQL или SOAP, могут быть более подходящими в зависимости от особенностей проекта и его потребностей.

# 8 Тестирование

## 8.1 Функциональное тестирование

Функциональное тестирование позволяет проверить работоспособность каждой функции приложения, а также выявить возможные ошибки и недочеты.

В функциональном тестировании идет проверка основных функций приложения, таких как поиск концертов, выбор билетов, оформление покупки и другой функционал.

## 8.2 Системное тестирование

Системное тестирование направлено на проверку работы всей системы или приложения в целом, включая различные компоненты и их взаимодействие.

## 8.3 Тестирование пользовательского интерфейса

Тестирование интерфейса мобильного приложения CityConcert направлено на проверку использования пользовательского интерфейса и на выявление возможных ошибок в его работе.

# Заключение

Таким образом, было реализовано мобильное приложение – инструмент для поиска, покупки и продажи билетов на различные мероприятия. Благодаря нему, пользователи смогут посещать различные мероприятия и спокойно обменивать билеты на них, не опасаясь мошенничества. Фильтры позволят настроить поиск под свои предпочтения, а возможность найти компанию на мероприятие сделает посещение события еще более интересным и приятным.

В заключение, разработка мобильного приложения для продажи билетов на концерты является сложным и многогранным проектом, требующим использования различных технологий и фреймворков. Однако, благодаря правильно подобранным инструментам, возможно создание высококачественного и удобного приложения, способного удовлетворить потребности пользователей и обеспечить успешную работу бизнеса. Важно учитывать требования безопасности и защиты данных пользователей, а также предусмотреть возможность масштабирования приложения для обеспечения его стабильной работы в будущем.

В рамках дальнейшего улучшения в приложение может быть добавлена светлая тема, дополнительные языки интерфейса, возможность добавлять мероприятия в «список желаний» или «избранное», возможность фильтровать и искать билеты в разделе «мои билеты». Также может быть добавлено уведомление пользователя о предстоящих мероприятиях.

Для администратора может быть добавлена возможность указать несколько дат проведения для одного мероприятия. А также возможность модерации запросов на обмен билетов и поиск компании.

# Список использованных источников


1. Документация Flutter [Электронный ресурс]. – Режим доступа: URL: https://docs.flutter.dev/ – Заглавие с экрана. – (Дата обращения 01.05.2023).
2. Документация SpringBoot [Электронный ресурс]. – Режим доступа: URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ – Заглавие с экрана. – (Дата обращения 22.04.2023).
3. Документация Swager [Электронный ресурс]. – Режим доступа: URL: https://swagger.io/docs/ - (Дата обращения 10.05.2023).
4. Вигерс К. Разработка требований к программному обеспечению/ Карл Вигерс, Джой Бити. — М.: Изд-во Русская редакция, 2014. — 736 с.
5. The Prosand Cons of Online Booking Systems [Электронный ресурс]. – Режим доступа: URL: https://miyn.app/the-pros-and-cons-of-online-booking-system/ – Заглавие с экрана. – (Дата обращения 21.05.2023).
6. Платное обслуживание населения России [Электронный ресурс]. – Режим доступа:URL: https://rosstat.gov.ru/folder/210/document/13235 – Заглавие с экрана. – (Дата обращения 21.05.2023).
7. Особенности специфики формирования доступности услуг массового спорта [Электронный ресурс]. – Режим доступа: URL: https://apni.ru/article/1095-osobennosti-spetsifiki-formirovaniya-dostupn – Заглавие с экрана. – (Дата обращения 21.05.2023).