

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

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Онлайн менеджер по учету вечеринок “AlcoManager”

Курсовой проект

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Руководитель _____ В.С. Тарасов, ст. преподаватель __.__.20__

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

Обучающийся _____ Д.А. Савельев, 3 курс, д/о

Обучающийся _____ Г.О. Корчагин, 3 курс, д/о

Руководитель _____ К.В. Зенин, ассистент

Воронеж 2022

Содержание

Введение	4
1. Постановка задачи	5
1.1. Требования к разрабатываемой системе	5
1.1.1. Функциональные требования	5
1.1.2. Технические требования	5
1.2. Требования к интерфейсу	5
1.3. Задачи решаемые в процессе разработки.....	6
2. Анализ предметной области.....	7
2.1. Глоссарий.....	7
2.2. Анализ задачи	8
2.3. Входные-выходные данные	9
2.4. Обзор аналогов	10
2.4.1. Diobox.....	10
2.4.2. Doodle.....	11
3. Реализация	12
3.1. Средства реализации.....	12
3.2. Реализация серверной части приложения	13
3.2.1. Схема базы данных.....	13
3.2.2. Диаграмма классов	14
3.2.2.1. Диаграмма классов сущностей	14
3.2.2.2. Диаграмма классов сервисов	15
3.2.2.3. Диаграмма классов контроллеров	16
3.2.2.4. Диаграмма служебных классов.....	17
3.2.3. Архитектура серверной части приложения	19
3.2.4. Слой доступа к данным.....	19
3.2.5. Слой контроллеров.....	19
3.2.6. Слой бизнес логики	20

3.3. Реализация клиентской части приложения.....	21
3.3.1. Макеты интерфейса	21
Заключение	29
Список использованных источников	30

Введение

Проблема распределения времени в современном мире имеет важную роль. Зачастую даже согласовав время мероприятия, мы сталкиваемся с проблемой разности вкусовых предпочтений. Но с развитием цифровых технологий решение данных проблем становится всё легче и легче, ведь даже в больших компаниях можно прийти к общему решению, выше перечисленных проблем.

Целью данной работы является разработка веб-приложения с возможностью согласовывать какие-либо мероприятия и делиться своими предпочтениями.

1. Постановка задачи.

Целью данного проекта является разработка веб-приложения для менеджмента мероприятий и предпочтений, под названием – “AlcoManager”.

1.1. Требования к разрабатываемой системе.

1.1.1. Функциональные требования.

К разрабатываемому приложению выдвигаются следующие требования:

- Возможность авторизации и регистрации новых пользователей;
- Разделение пользователей по ролям;
- Добавление еды и напитков;
- Создание мероприятий;
- Просмотр информации о мероприятиях;
- Изменения предпочтений в еде и напитках.

1.1.2. Технические требования.

Приложение должно обеспечивать:

- Авторизацию пользователей посредством логина и пароля;
- Шифрование логина и пароля при записи в базу данных.

1.2. Требования к интерфейсу.

Список требований к интерфейсу:

- Выполнен в едином стиле и цветовой гамме;
- Все надписи должны быть легко читаемы;
- Все элементы управления должны выделяться на фоне содержимого;
- Должен корректно отображаться при изменении размеров экрана;
- Содержать только необходимую для пользователя информацию.

1.3. Задачи, решаемые в процессе разработки.

Были поставлены следующие задачи:

- Анализ предметной области;
- Анализ аналогов;
- Разработка макетов интерфейса;
- Определение используемой платформы;
- Написание технического задания;
- Реализация сервера;
- Реализация интерфейса;
- Реализация модуля авторизации;
- Описание процесса разработки и результата.

2. Анализ предметной области

2.1. Глоссарий.

- **Проект** – разрабатываемое приложение;
- **Сервер, серверная часть** – компьютер, обслуживающий другие устройства (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач;
- **Клиент, клиентская сторона** – в данном проекте - сайт, который 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.
- **Back-end** – программно-аппаратная часть приложения. Отвечает за функционирование внутренней части приложения;
- **GitHub** – веб-сервис для хостинга IT-проектов и их совместной разработки;
- **Пользователь** – авторизованный в системе человек, пользующийся функционалом приложения;
- **Администратор** – пользователь, у которого есть привилегии;
- **Гость** – человек, не имеющий учетной записи, может только зарегистрироваться или авторизоваться.

2.2. Анализ задачи.

На рисунке 1 продемонстрирована диаграмма UseCase, которая показывает какие сценарии использования приложения доступны пользователю.

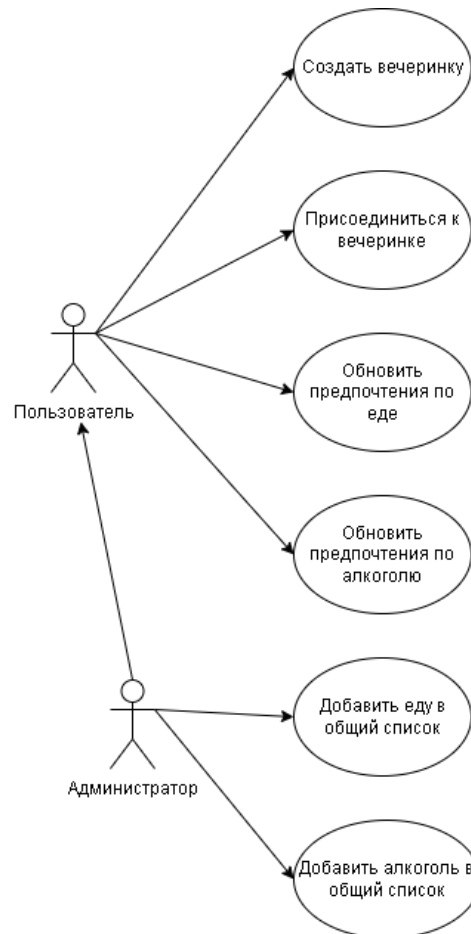


Рисунок 1 - UseCase диаграмма.

Обычный пользователь имеет 4 сценария работы с приложением.

2.3 Входные-выходные данные

Рассмотрим основной бизнес – процесс на примере IDEF0 диаграммы, представленной на рисунке 2. Данная диаграмма представляет собой общее видение процесса работы веб-приложения.

Работу сервиса регулирует законодательство Российской Федерации. Обеспечивает работу приложения БД с алкоголем и едой. На вход в систему поступает Пользователь.



Рисунок 2 – IDEF0 диаграмма.

2.4 Обзор аналогов

Приступая к разработке, приложения необходимо проанализировать проекты, уже существовавшие до него. Рассмотрим их достоинства, недостатки. Насколько они удобны в использовании, содержат ли необходимую функциональность. И на основе этого анализа сделаем выводы какие моменты нужно учесть при разработке своего приложения, а именно каким образом будет построен удобно для пользователя интерфейс, и какую функциональность нужно обеспечить.

2.4.1. Diobox.

На рисунке 3 представлен интерфейс приложения. В нем есть необходимый функционал для управления мероприятием. Можно отправлять приглашения, управлять профилями гостей, назначать «плюс», отслеживать действия гостей и просматривать действия гостей по мере необходимости. Это также упрощает сотрудничество с вашей командой, используя разные роли прав доступа. Он эффективен и прост в использовании. В общем, интегрированное решение для расчета расписания рассадки, создания веб-сайта мероприятий, продажи билетов и приема платежей в нескольких валютах.

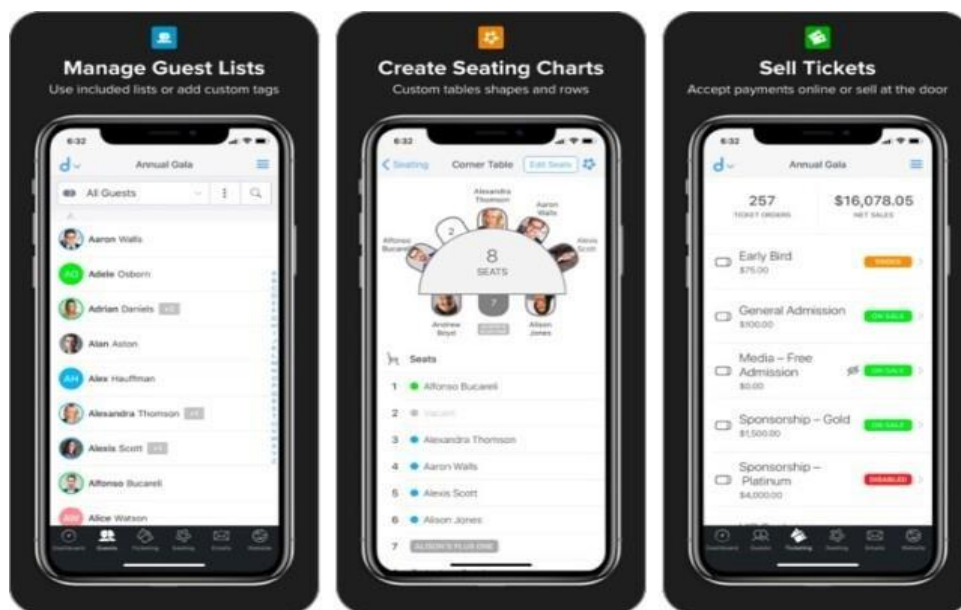


Рисунок 3 - Внешний вид Diobox

Достоинства:

- Можно импортировать список пользователей из контактов;
- Много функций.

Недостатки:

- Неудобный интерфейс;
- Цена (22\$/мес).

2.4.2. Doodle.

Это приложение позволяет координировать доступность участников на вечеринку, чтобы указать день, в который все могут собраться. Одной из наиболее практичных функций является возможность делиться своей повесткой дня с другими пользователями в дополнение к интеграции с Календарем Google.

Достоинства:

- Прост в использовании.

Недостатки:

- Мелкий интерфейс;
- Нестабильная работа.

3. Реализация.

3.1. Средства реализации.

В качестве средств реализации приложения были выбраны:

Сервер:

- ОС MacOS BigSur, Windows 10;
- Frameworks: Spring Boot 2.6.6;
- Библиотеки: Hibernate, Lombok;
- База данных – MySQL;
- Язык разработки - Java 17;
- IntelliJ IDEA 2021.3.2 (Ultimate Edition);
- Система контроля версий – Git;

Клиент:

- ОС MacOS BigSur, Windows 10;
- IntelliJ IDEA 2021.3.2 (Ultimate Edition);
- Система контроля версий – Git.

Для серверной части был выбрана связка Java и Spring Boot так как фреймворк Spring Boot имеет большое количество преимуществ, среди них:

- Большое количество доступной документации;
- Встроенный сервер для развертывания приложения, что существенно облегчает процесс разработки;
- Огромный выбор плагинов, которые легко ставятся и сильно облегчают процесс разработки;
- Автоматическое внедрение зависимостей.

3.2. Реализация серверной части приложения.

3.2.1. Схема базы данных.

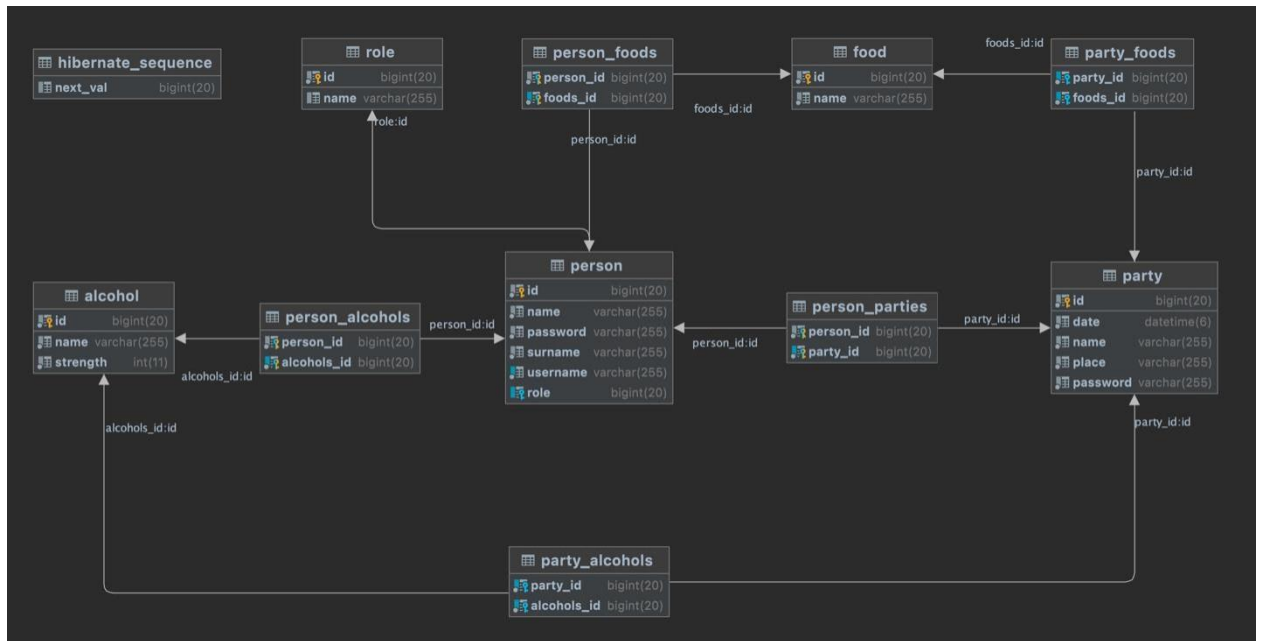


Рисунок 4 – Схема базы данных.

На рисунке 4 представлена схема базы данных. В ней присутствуют все таблицы, используемые в приложении.

3.2.2. Диаграмма классов.

3.2.2.1. Диаграмма классов сущностей.



Рисунок 5 – Диаграмма классов сущностей.

На рисунке 5 представлена диаграмма классов сущностей со всеми полями.

3.2.2.2. Диаграмма классов сервисов.

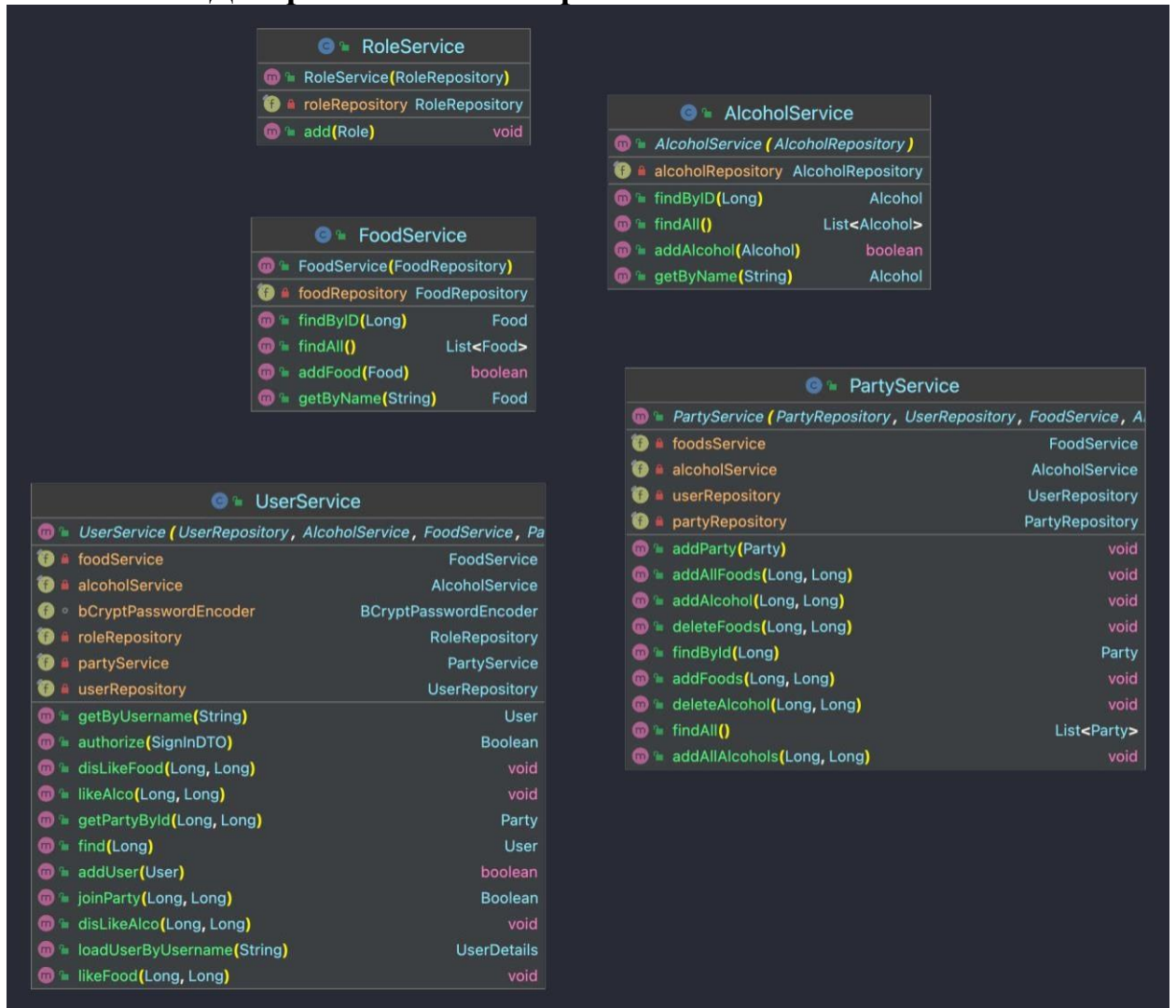


Рисунок 6 – Диаграмма классов сервисов.

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.

3.2.2.3. Диаграмма классов контроллеров.

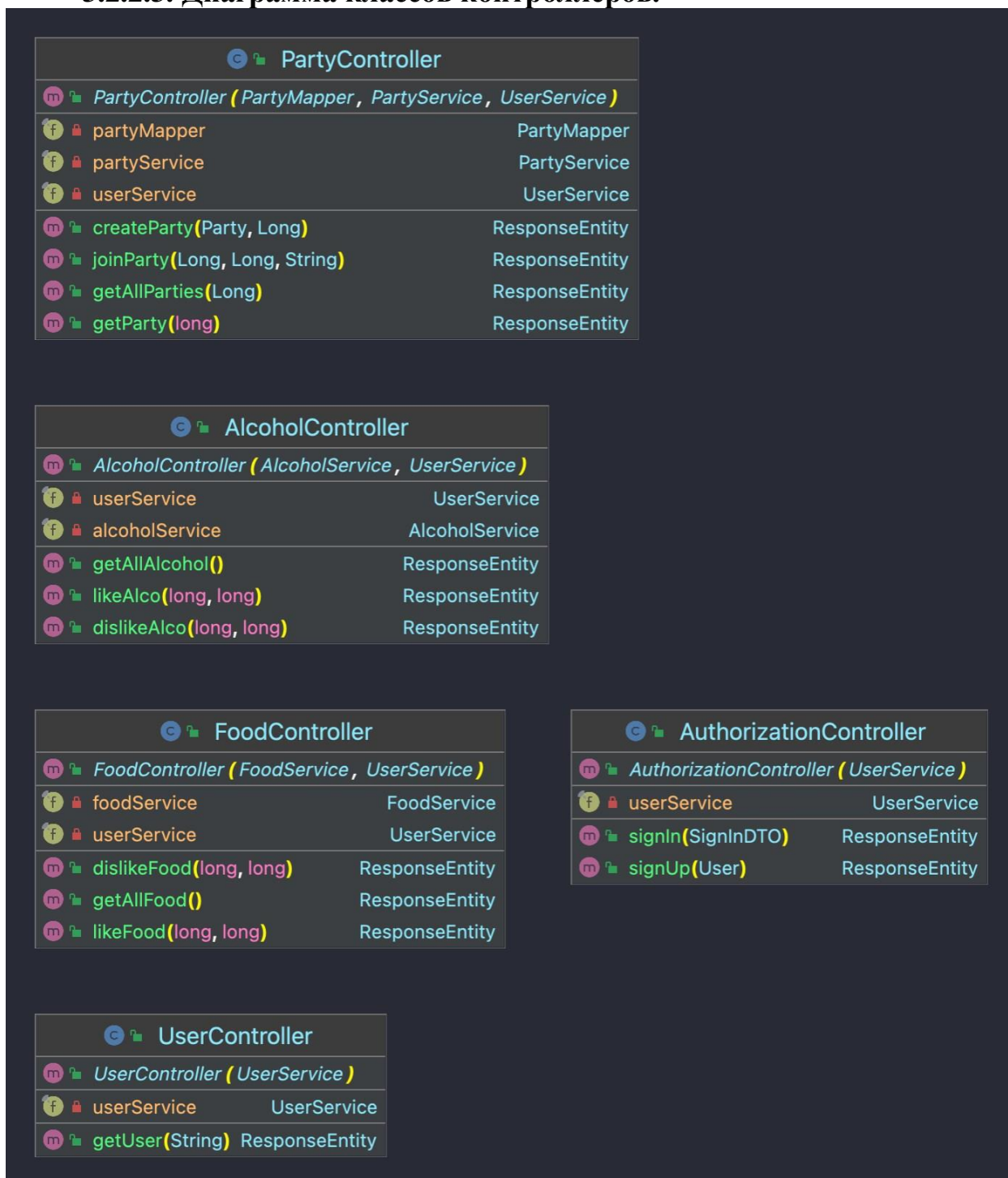


Рисунок 7 – Диаграмма классов контроллеров.

Классы на рисунке 7 необходимы для общения с клиентом. Они получают запросы, вызывают методы слоя бизнес-логики и отправляют ответы на клиент.

3.2.2.4. Диаграмма служебных классов.

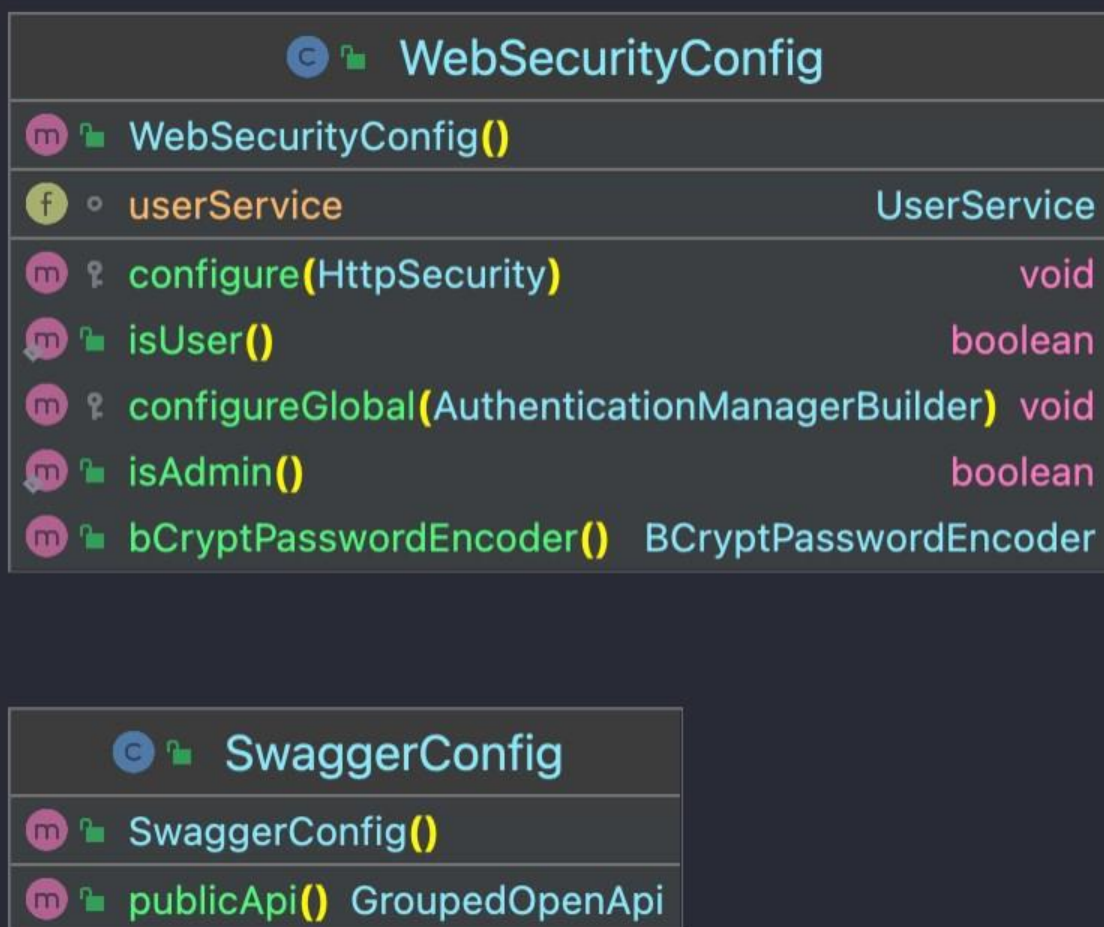


Рисунок 8 – Диаграмма служебных классов.



Рисунок 9 – Диаграмма служебных классов.

На рисунке 8 и 9 представлены диаграммы служебных классов.

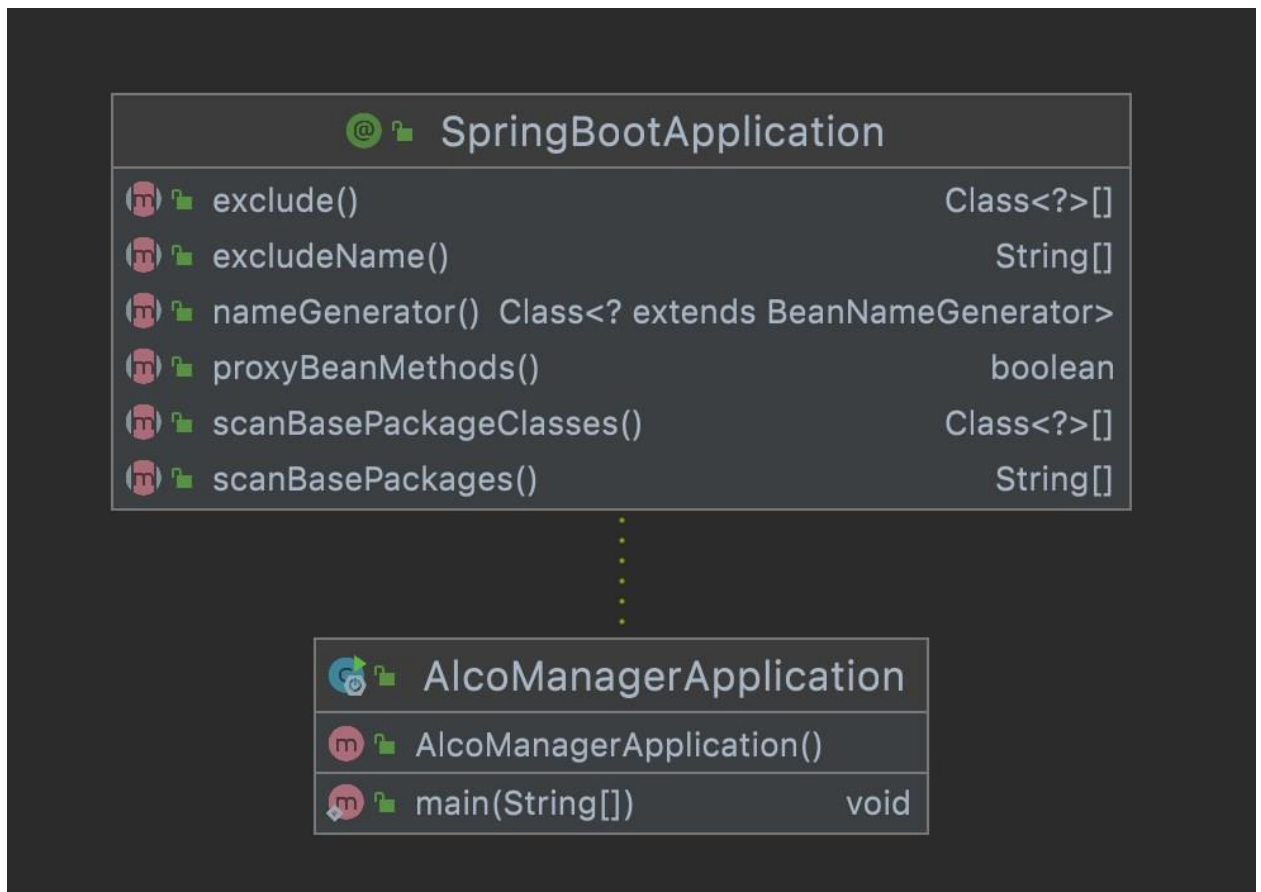


Рисунок 10 – Точка входа.

Класс AlcoManagerApplication является точкой входа для приложения, представлен на рисунке 10, он запускает цепочку загрузки нужных зависимостей и классов, благодаря которым приложение функционирует.

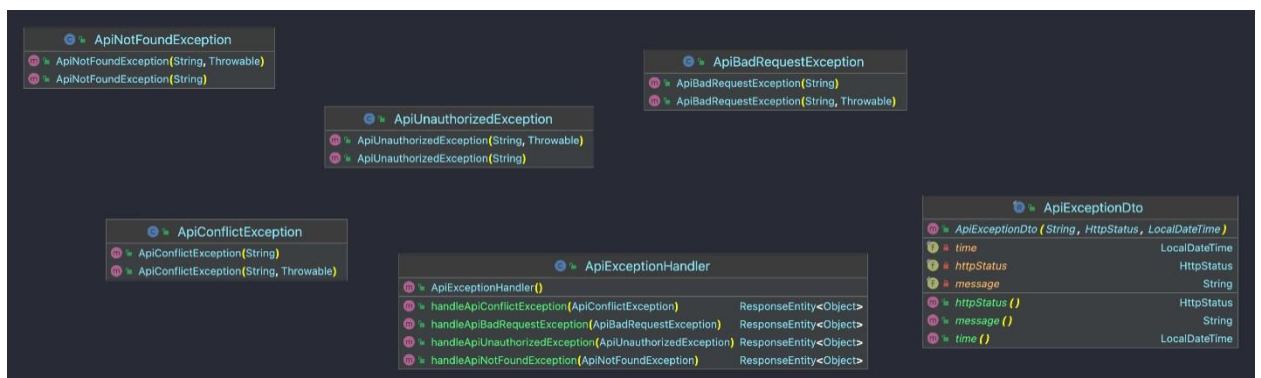


Рисунок 11 – ExceptionHandler.

Класс ApiExceptionHandler представлен на рисунке 11 и является обработчиком исключений.

3.2.3. Архитектура серверной части приложения.

Серверная часть приложения реализована соответственно трехслойной архитектуре веб-приложения с использованием фреймворка Spring boot. Это фреймворк предоставляет возможности для удобной работы с базами данных, сам настраивает внедрение зависимостей.

3.2.4. Слой доступа к данным.

На рисунке 12 представлена реализация класса UserRepository. Для каждой из сущностей был реализован интерфейс-репозиторий, который является наследником JpaRepository. Такой подход позволил существенно уменьшить количество написанного кода, путем использования возможностей Spring Data, а именно больше количество уже реализованных методов для генерации запроса в базу данных и получения ответа, и возможность объявления своих методов, при правильном написании названий которых Spring Data сам сгенерирует запрос в базу.

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
}
```

Рисунок 12 – Реализация интерфейса UserRepository.

3.2.5. Слой контроллеров.

Контроллеры – такие классы, каждый метод из которых обрабатывает запрос с клиента на определенный маппинг и возвращает ответ. Для каждой из сущностей были написаны контроллеры, методы которых отвечают за необходимые приложению действия с этими сущностями.

3.2.6. Слой бизнес-логики.

Вся бизнес-логика реализована в Service слое. Этот слой является промежуточным звеном между контроллерами и работой с базой данных, поэтому чаще всего методы просто передают аргумент, полученный из контроллера, вызвавшего его, в соответствующий метод в репозитории, затем возвращают ответ в контроллер, или собирают объект из полученных аргументов и также передают его в репозиторий, а полученный ответ затем возвращают контроллеру. Такие методы представлены на рисунке 13.

```
public boolean addUser(User user) {  
    if (getByUsername(user.getUsername()) != null) {  
        return false;  
    }  
    Role roleUser = roleRepository.findByName("ROLE_USER");  
    if (roleUser == null) {  
        Role newRole = new Role( id: 0L, name: "ROLE_USER");  
        roleRepository.save(newRole);  
        user.setRole(newRole);  
    } else {  
        user.setRole(roleUser);  
    }  
    user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));  
    userRepository.save(user);  
    return true;  
}
```

Рисунок 13 – Метод addUser.

3.3. Реализация клиентской части приложения.

3.3.1. Макеты интерфейса.

Сразу после запуска приложения пользователю будет показан экран, на котором ему будет предложено войти в свою учетную запись, зарегистрировать новую учетную запись или продолжить работу с приложением без входа в учетную запись.

На рисунке 14 представлен начальный экран приложения.

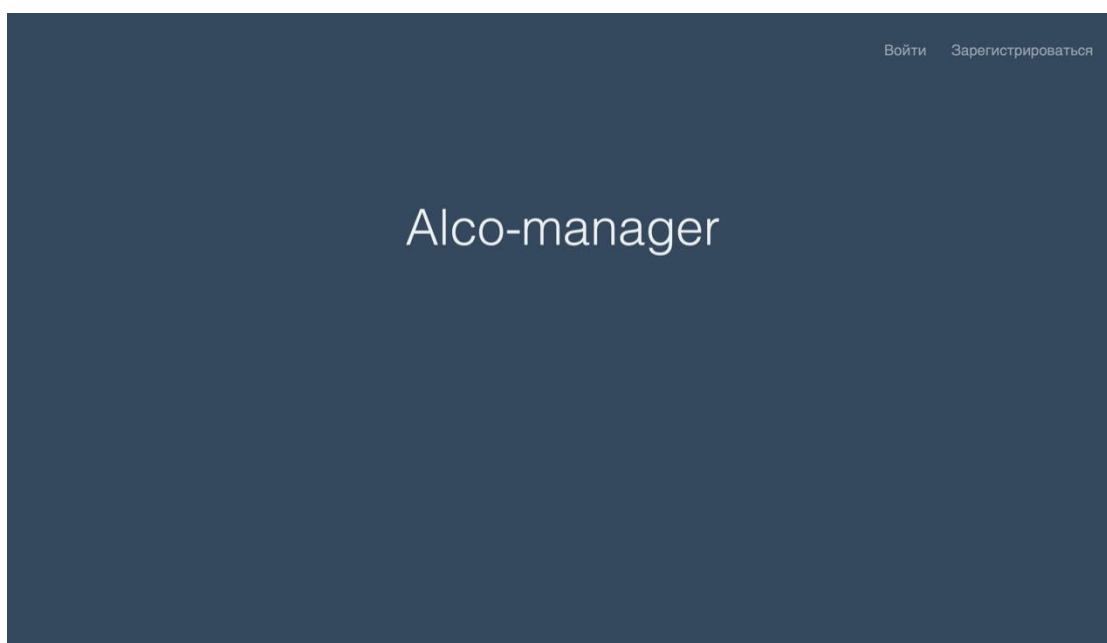


Рисунок 14 – Начальный экран приложения.

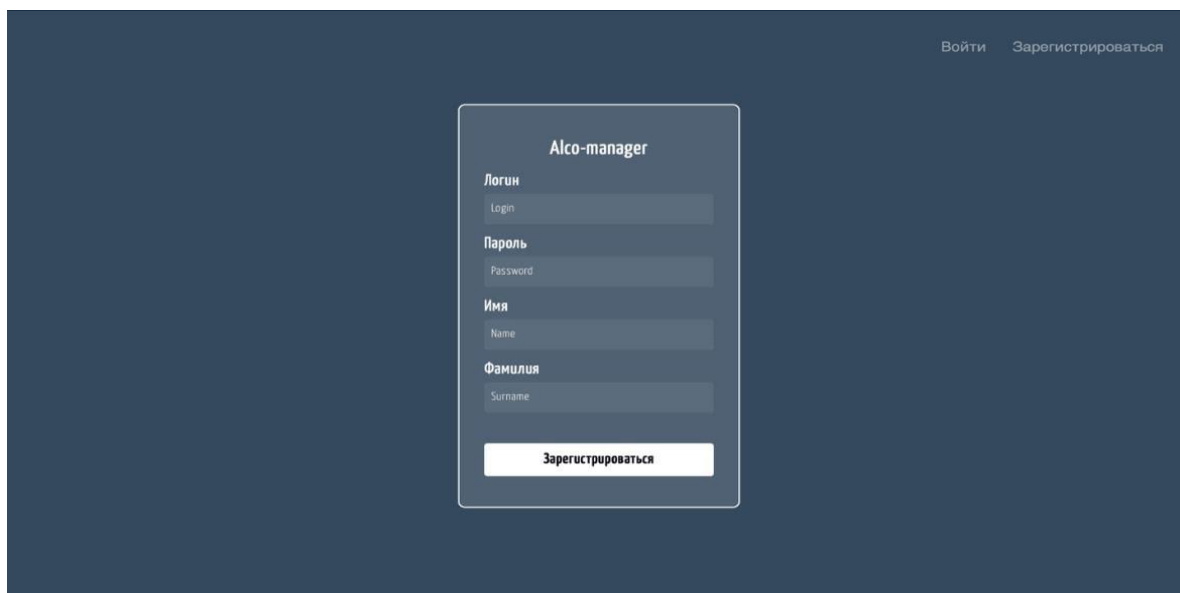


Рисунок 15 – Экран регистрации.

На рисунке 15 представлен экран регистрации приложения. Рассмотрим случай, когда пользователь в первый раз зашел в приложение и еще не зарегистрировал учетную запись.

В этом случае при нажатии на кнопку «Регистрация» пользователь увидит экран регистрации, где ему будет предложено ввести логин, пароль, имя и фамилию. Если будут введены корректные данные, то будет создана новая учетная запись, и он будет направлен на экран входа.

На рисунке 16 представлен экран авторизации. После нажатия, зарегистрированного пользователя, кнопки “Войти”, появляется окно, где ему предлагают ввести свои учётные данные, в виде логина и пароля. Если пользователь ввёл верные учётные данные, то он увидит главный экран своей учётной записи.

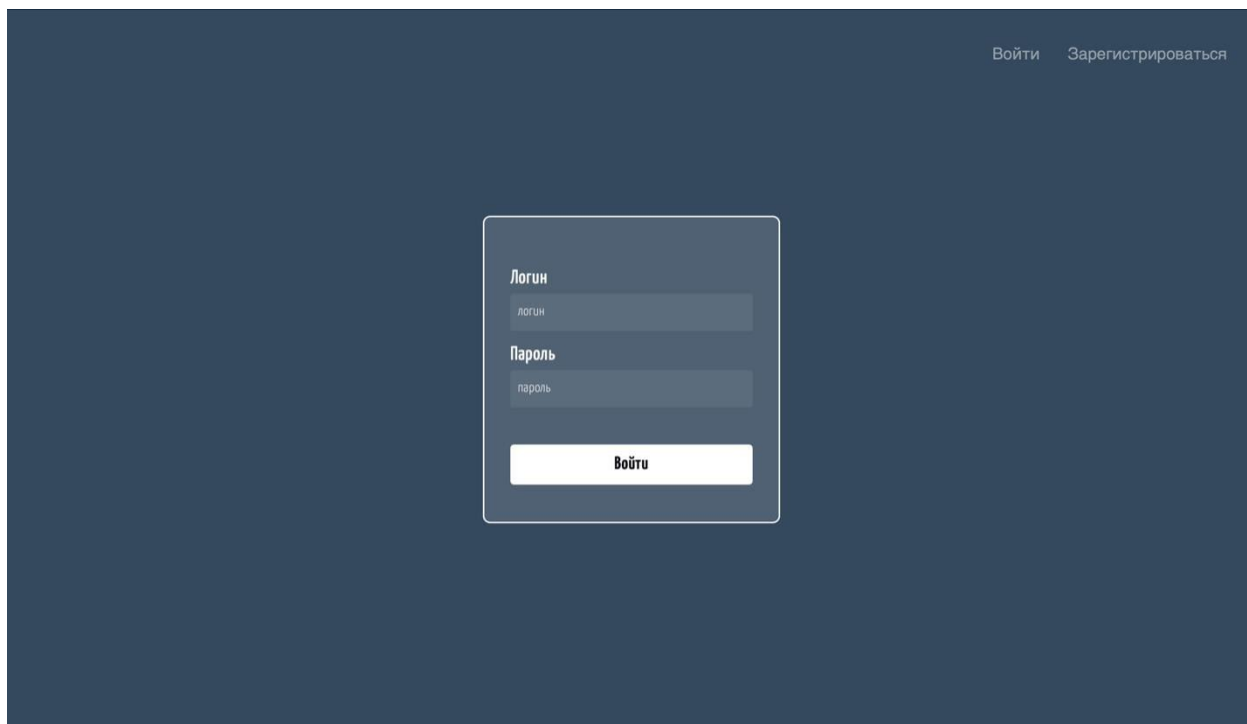


Рисунок 16 – Экран входа в приложение.

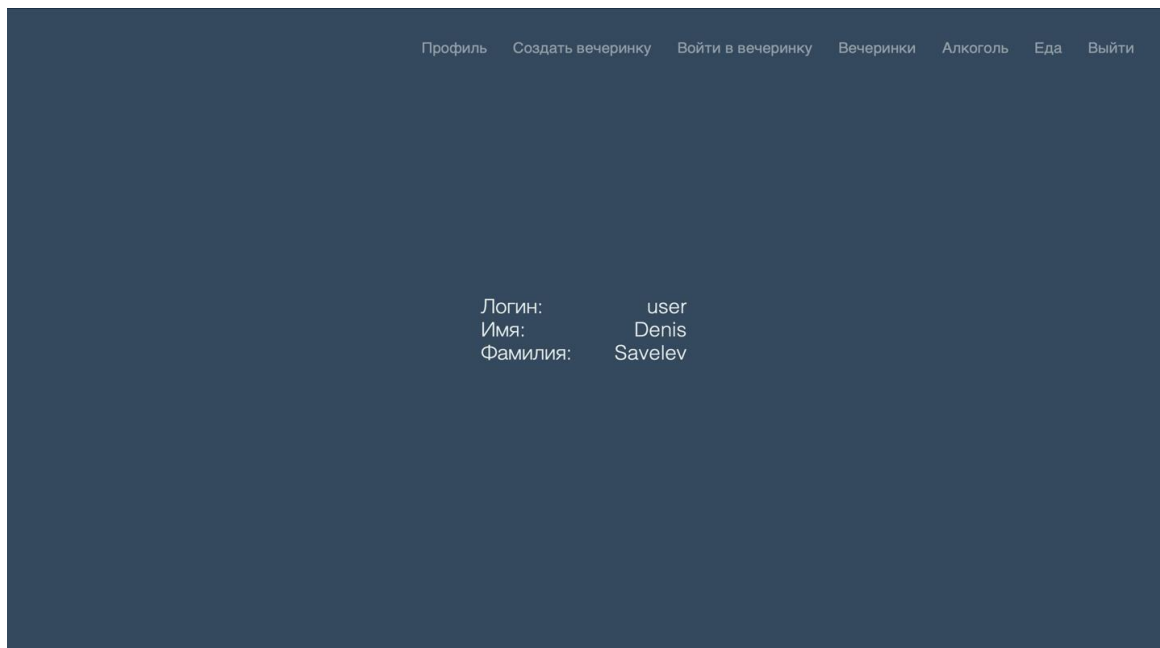


Рисунок 17 – Главный экран приложения для обычного пользователя.

На рисунке 17 представлен главный экран приложения для обычного пользователя.

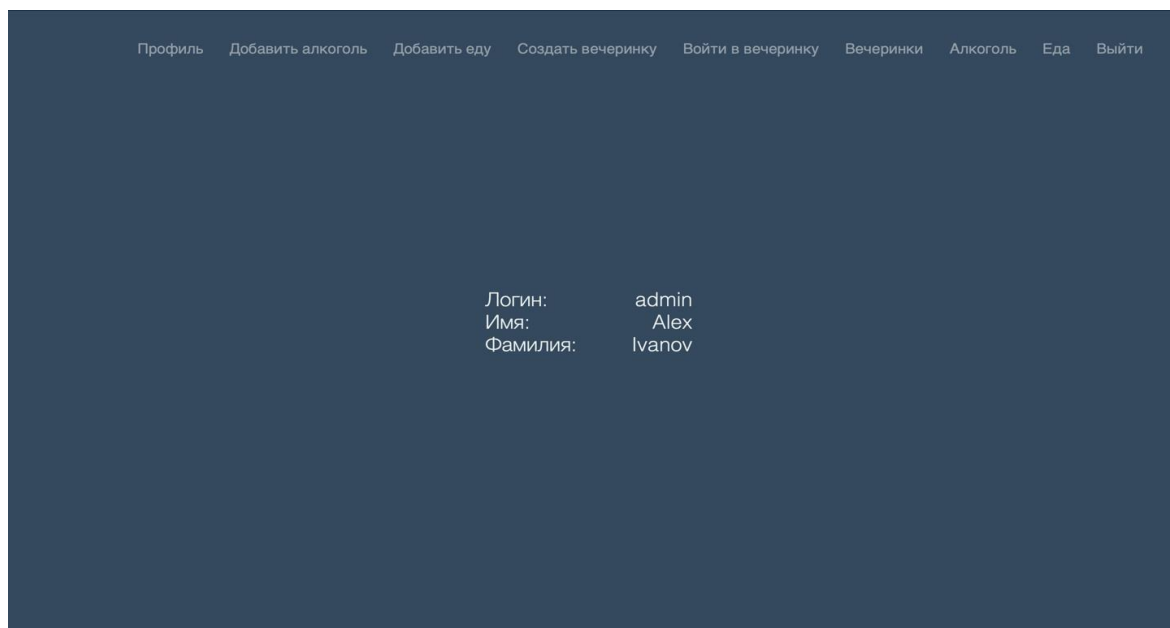
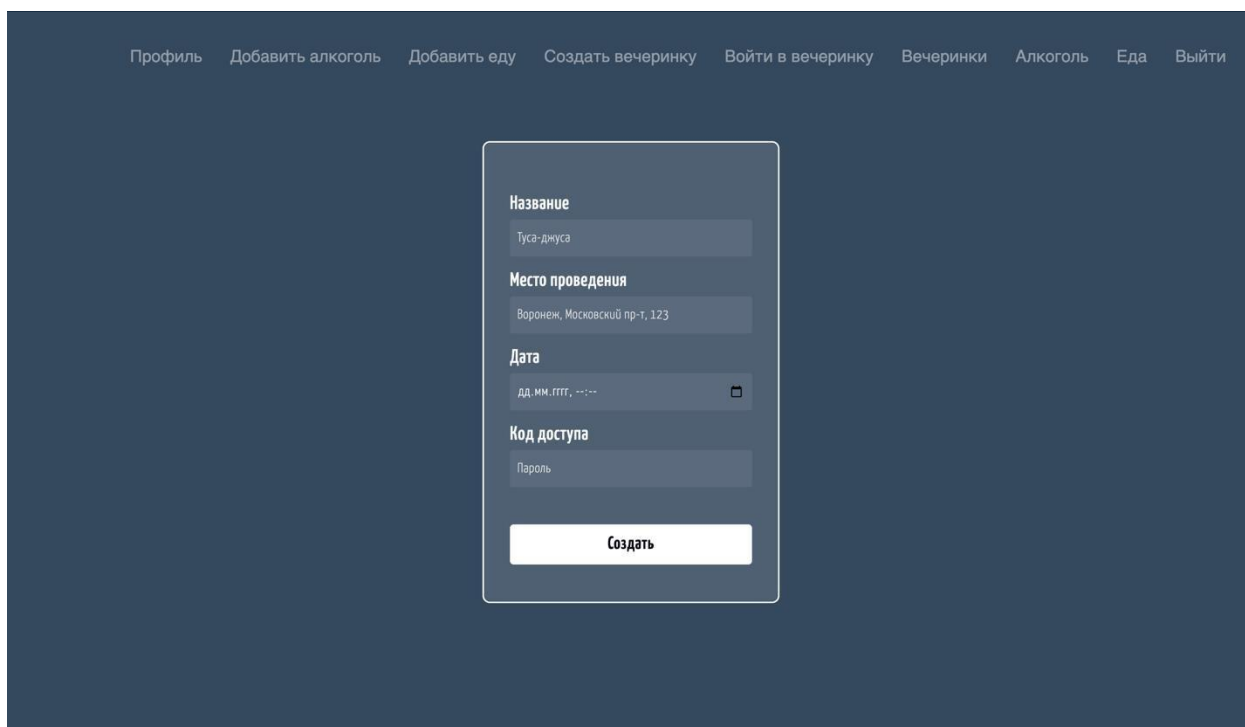


Рисунок 18 – Главный экран приложения для администратора.

На рисунке 18 представлен главный экран приложения для администратора.

Рассмотрим сначала дополнительные возможности администратора – «Добавить алкоголь» и «Добавить еду»

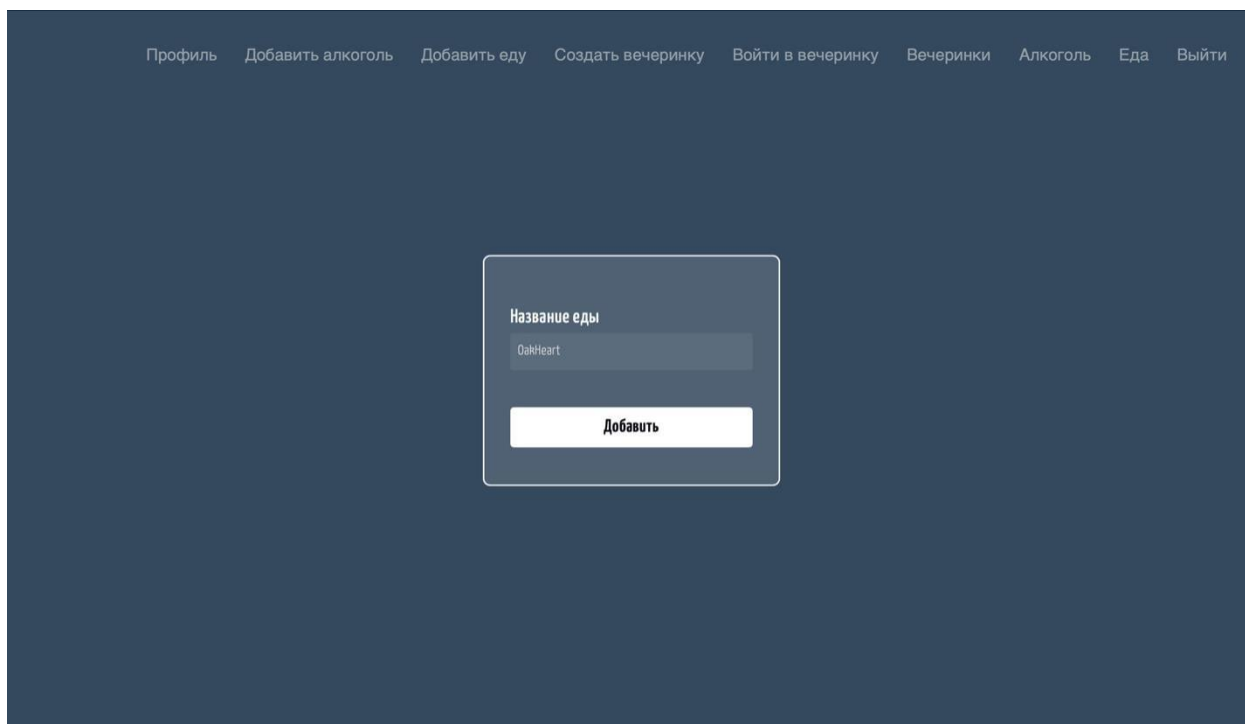
На рисунке 19 представлена страница добавления алкоголя.



The screenshot shows a web application interface with a dark blue background. At the top, there is a navigation bar with links: 'Профиль', 'Добавить алкоголь', 'Добавить еду', 'Создать вечеринку', 'Войти в вечеринку', 'Вечеринки', 'Алкоголь', 'Еда', and 'Выйти'. In the center, there is a white-bordered form for adding alcohol. The form contains the following fields: 'Название' (Name) with the value 'Туса-днуса', 'Место проведения' (Location) with the value 'Воронеж, Московский пр-т, 123', 'Дата' (Date) with a placeholder 'дд.мм.гггг' and a calendar icon, and 'Код доступа' (Access code) with the value 'Пароль'. At the bottom of the form is a white button labeled 'Создать'.

Рисунок 19 – Экран добавления алкоголя.

На рисунке 20 представлена страница добавления еды.



The screenshot shows the same web application interface as Figure 19. The navigation bar is identical. In the center, there is a white-bordered form for adding food. The form contains the following fields: 'Название еды' (Food name) with the value 'OakHeart', and a white button labeled 'Добавить' at the bottom.

Рисунок 20 – Экран добавления еды.

Профиль Добавить алкоголь Добавить еду Создать вечеринку Войти в вечеринку Вечеринки Алкоголь Еда Выйти

Название
Tusa-джуса

Место проведения
Воронеж, Московский пр-т, 123

Дата
дд.мм.yyyy, ---

Код доступа
Пароль

Создать

Рисунок 21 – Экран создания мероприятия.

На рисунке 21 представлена страница на которой, можно создавать мероприятия, указав все нужные данные:

- Название мероприятия;
- Место проведения;
- Дата и код доступа по которому другие пользователи смогут присоединиться к вечеринке.

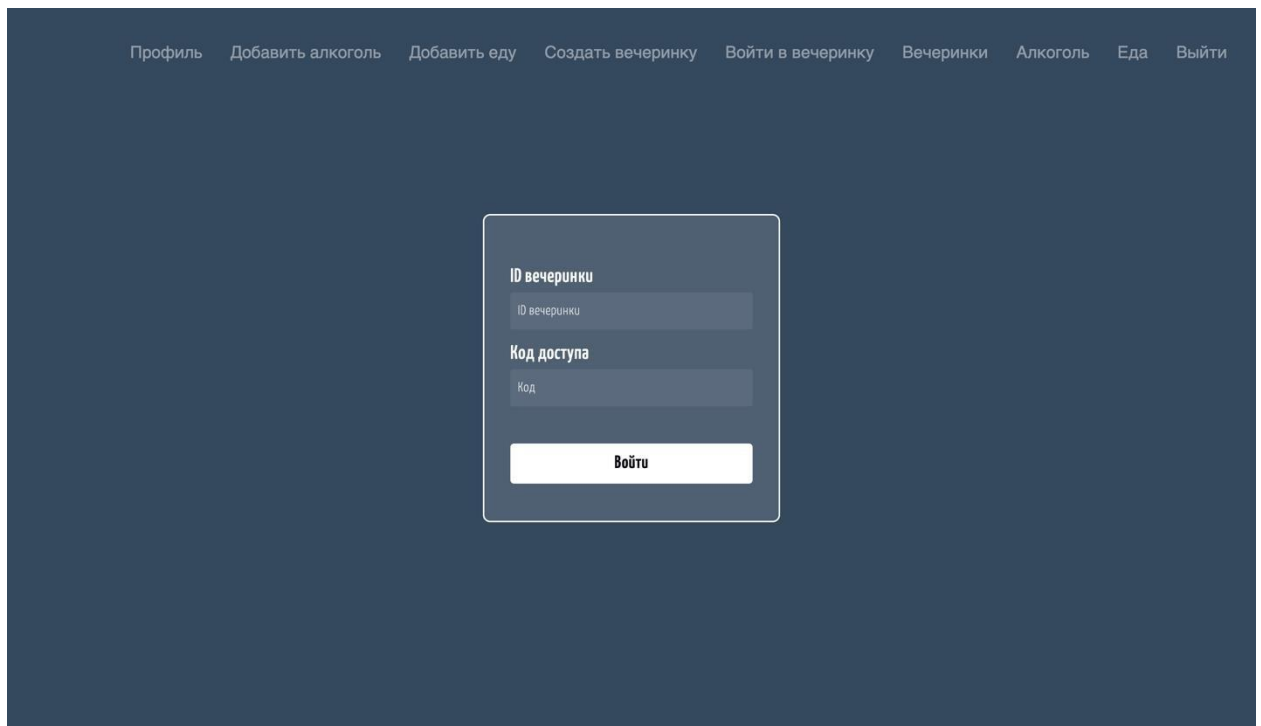


Рисунок 22 – Экран “Войти в мероприятие”.

На рисунке 22 представлена страница входа в мероприятие. После ввода данных пользователь присоединяется к мероприятию., все его предпочтения добавляются в предпочтения мероприятия. Пользователю после успешного присоединения показывается страница мероприятия, к которой он присоединился.

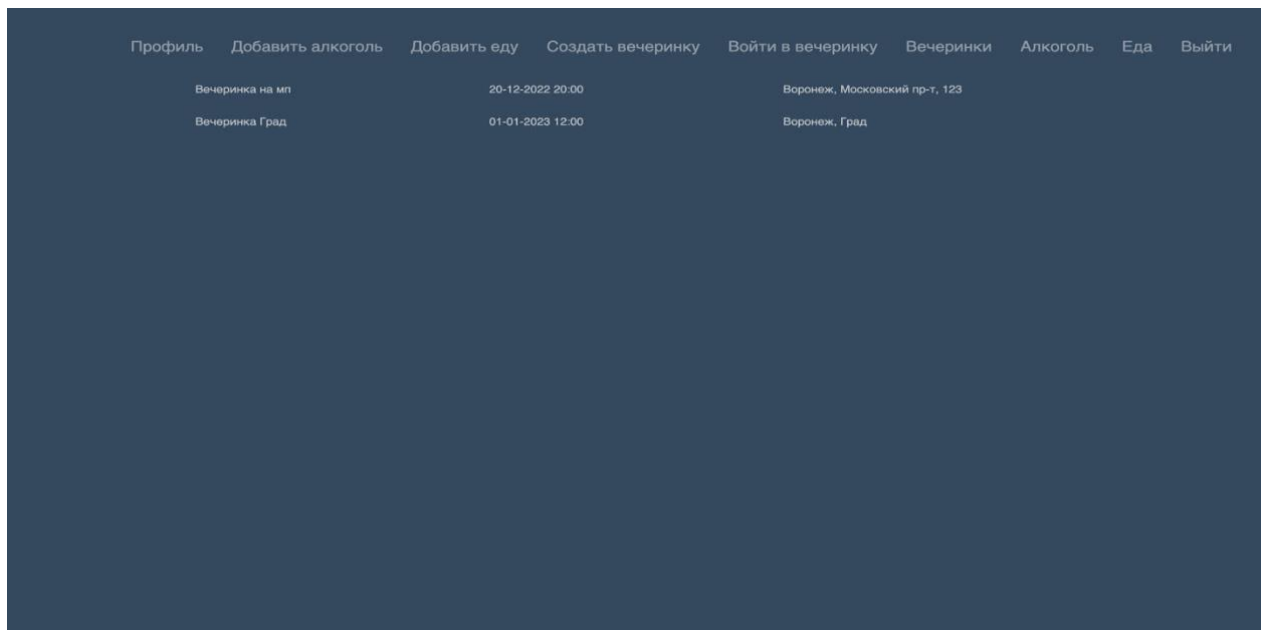


Рисунок 23 – Экран “Мероприятия”.

На рисунке 23 представлена страница текущих мероприятий.

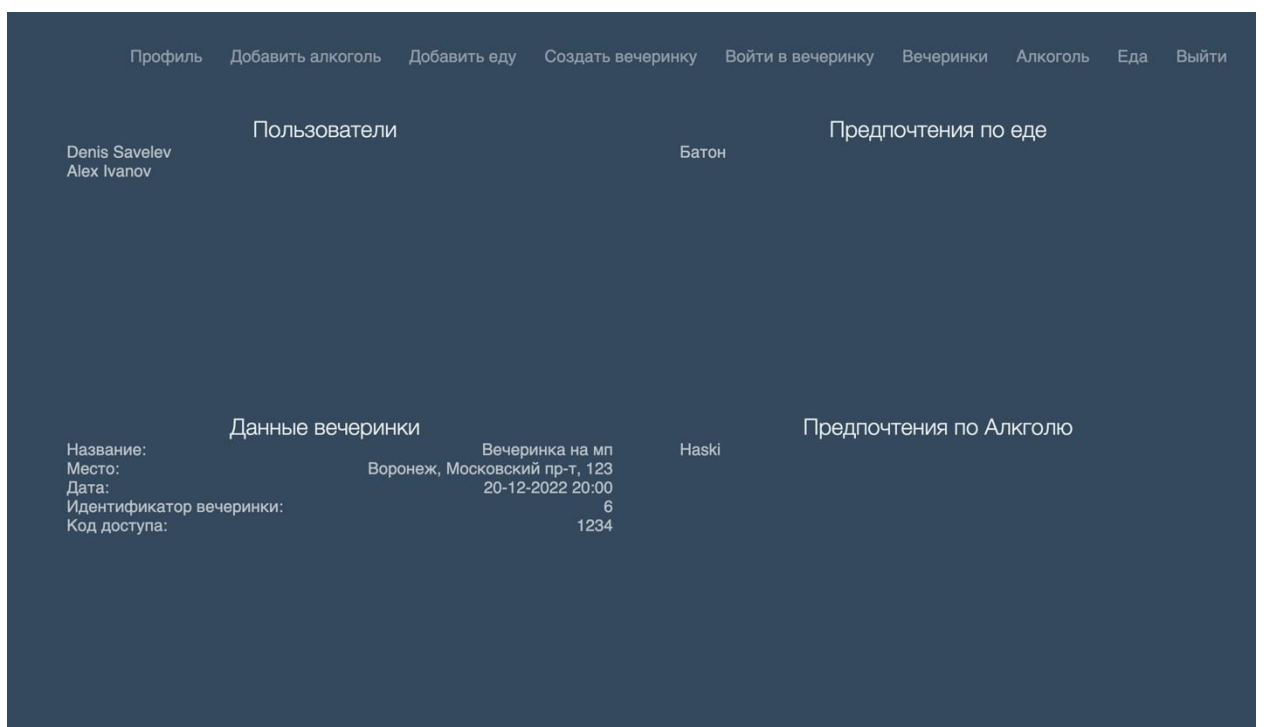


Рисунок 24 – Экран отдельно взятого мероприятия.

На рисунке 24 представлена страница просмотра отдельно взятого мероприятия.

Заключение

В ходе выполнения данного проекта, командой было разработано веб-приложение для отслеживания и создания мероприятий.

Была выполнена следующая работа:

- Произведён анализ предметной области и аналогов разрабатываемого приложения;
- Создана бизнес-логика приложения;
- Составлена схема базы данных;
- Создан макет интерфейса;
- Модуль авторизации;
- Для контроля версии был создан репозиторий GitHub.

В качестве дальнейшего усовершенствования приложения рассматриваются следующие модификации:

- Создания чата пользователей одного мероприятия.
- Возможность авторизации пользователей через различные ресурсы, например, Google, Яндекс, iCloud и т. д.
- Анализ предпочтений пользователя.
- Создания мобильной версии приложения.

Список использованных источников.

1. Документация Spring Data <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.query-methods> (дата обращения: 11.04.2022).
2. Документация Spring Boot <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#documentation.first-step> (дата обращения: 04.05.2022).
3. Документация js <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата обращения: 23.052022).
4. Документация react js <https://reactjs.org/docs/getting-started.html> (дата обращения: 23.052022).
5. Документация Lombook <https://projectlombok.org/features/> (дата обращения: 05.05.2022).
6. Документация react js <https://www.solidjs.com/docs/latest> (дата обращения: 23.052022).