

**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

Зав. кафедрой \_\_\_\_\_ С.Д. Махортов, д.ф.- м.н., доцент \_\_\_\_\_.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 \_\_\_\_\_ В.С. Тарасов, ст. преподаватель

Руководитель \_\_\_\_\_ И.В. Клейменов, 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 .....	7
2.3 Входные-выходные данные .....	9
2.4 Обзор аналогов .....	9
2.4.1 Diobox .....	10
2.4.2 Doodle .....	11
3 Реализация.....	12
3.1 Средства реализации.....	12
3.2 Реализация серверной части приложения .....	12
3.2.1 Схема базы данных .....	12
3.2.2 Диаграмма классов.....	13
3.2.2.1 Диаграмма классов сущностей .....	13
3.2.2.2 Диаграмма классов сервисов .....	14
3.2.2.3 Диаграмма классов контроллеров .....	15
3.2.2.4 Диаграмма служебных классов .....	16
3.2.3 Архитектура серверной части приложения.....	18

3.2.4 Слой доступа к данным .....	19
3.2.5 Слой контроллеров .....	19
3.2.6 Слой бизнес-логики .....	19
3.3 Реализация клиентской части приложения .....	20
3.3.1 Макеты интерфейса .....	20
Заключение .....	29
Список использованных источников .....	30

## **Введение**

Издавна человечество старалось упростить свою жизнь. Большая часть сил была направлена на облегчение задач в различных сферах жизни. Огромным шагом в этом направлении является развитие и удешевление ЭВМ, появление и постоянный прогресс смартфонов. В настоящее время сложно представить жизнь без коммуникации с людьми. Но общение не должно ограничиваться только интернет-пространством. Многие люди хотят встречаться вживую, но собираться какими-то большими компаниями сложно, т.к. бывает очень тяжело собрать всех в нужное время, в нужном месте. И не менее тяжело узнать, кто что предпочитает в напитках и еде.

Целью данной работы является разработка веб приложения с возможностью создавать какие-либо развлекательные встречи и мероприятия, а также делиться своими предпочтениями в напитках и еде.

## **1 Постановка задачи**

Целью данного курсового проекта является разработка веб приложения для менеджмента вечеринок и предпочтений– AlcoManager. К разрабатываемому приложению выдвинуты следующие требования.

### **1.1 Требования к разрабатываемой системе**

#### **1.1.1 Функциональные требования**

К разрабатываемому приложению выдвигаются следующие требования:

- Возможность регистрации новых пользователей;
- Возможность авторизации зарегистрированных пользователей;
- Разделение пользователей по ролям;
- Добавление напитков и еды;
- Создание вечеринок;
- Просмотр информации о вечеринках;
- Изменения предпочтений в еде и напитках;

#### **1.1.2 Технические требования**

Приложение должно обеспечивать:

- Авторизацию пользователей посредством логина и пароля, который шифруется и хранится в зашифрованном виде;

### **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. Информация должна находиться в тех местах приложения, где она будет актуальна, то есть, например, при просмотре определенной

вечеринки не нужно выводить информацию о пользователе или его предпочтениях.

### **1.3 Задачи, решаемые в процессе разработки**

Были поставлены следующие задачи:

- Анализ предметной области;
- Анализ аналогов;
- Постановка задачи;
- Разработка макетов интерфейса;
- Определение используемой платформы;
- Написание технического задания;
- Реализация слоя доступа к БД;
- 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.
- **БД** – база данных.

### 2.2 Анализ задачи

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

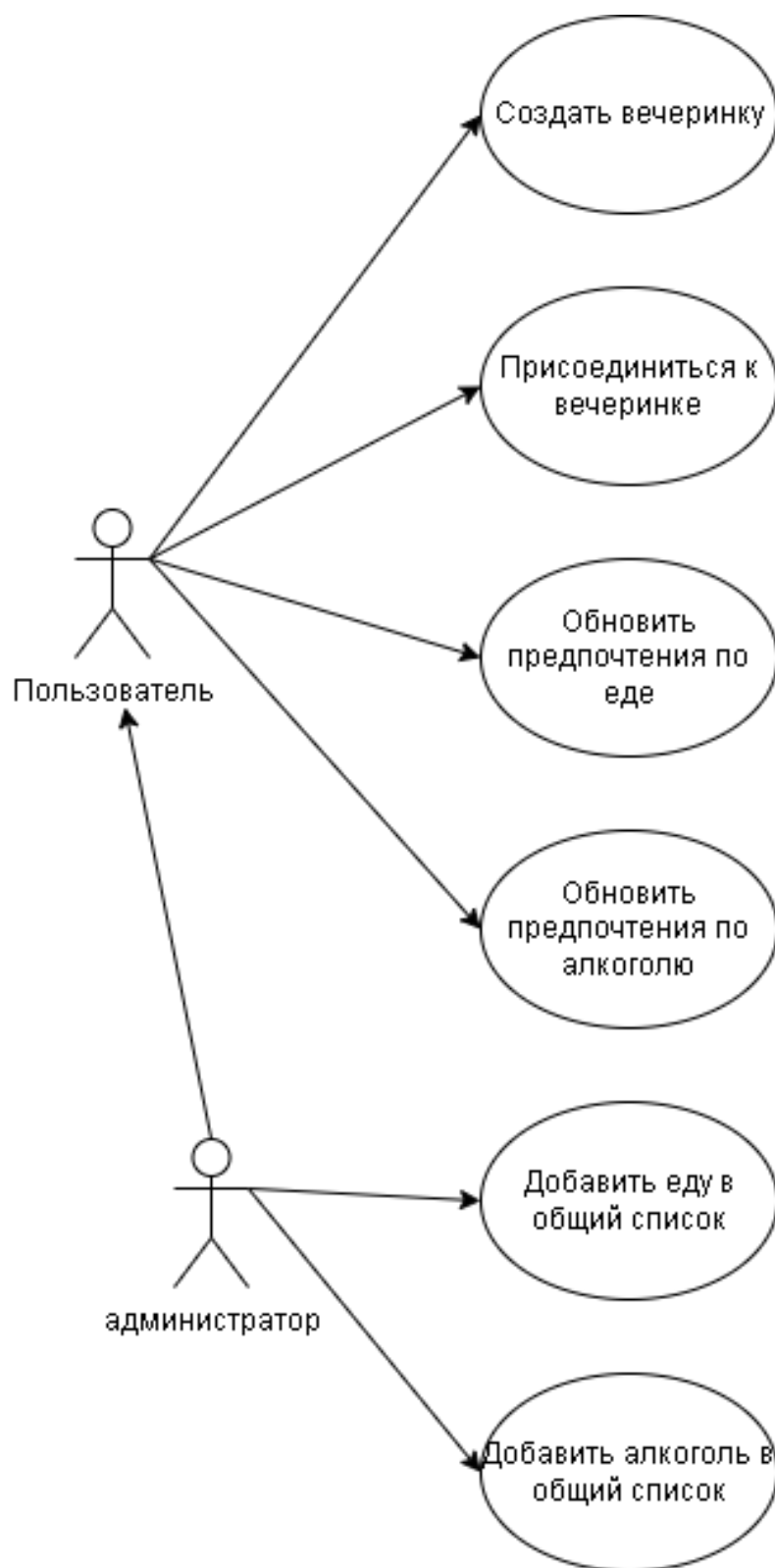


Рисунок 1 - Use case диаграмма

Неавторизованный пользователь имеет только 2 сценария

Обычный пользователь имеет 4 сценария работы с приложением



У администратора имеется 2 сценариев работы, но также он связан с обычным пользователем и у него есть все возможности обычного пользователя.

### 2.3 Входные-выходные данные

Рассмотрим основной бизнес – процесс на примере контекстной 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 - Контекстная 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.

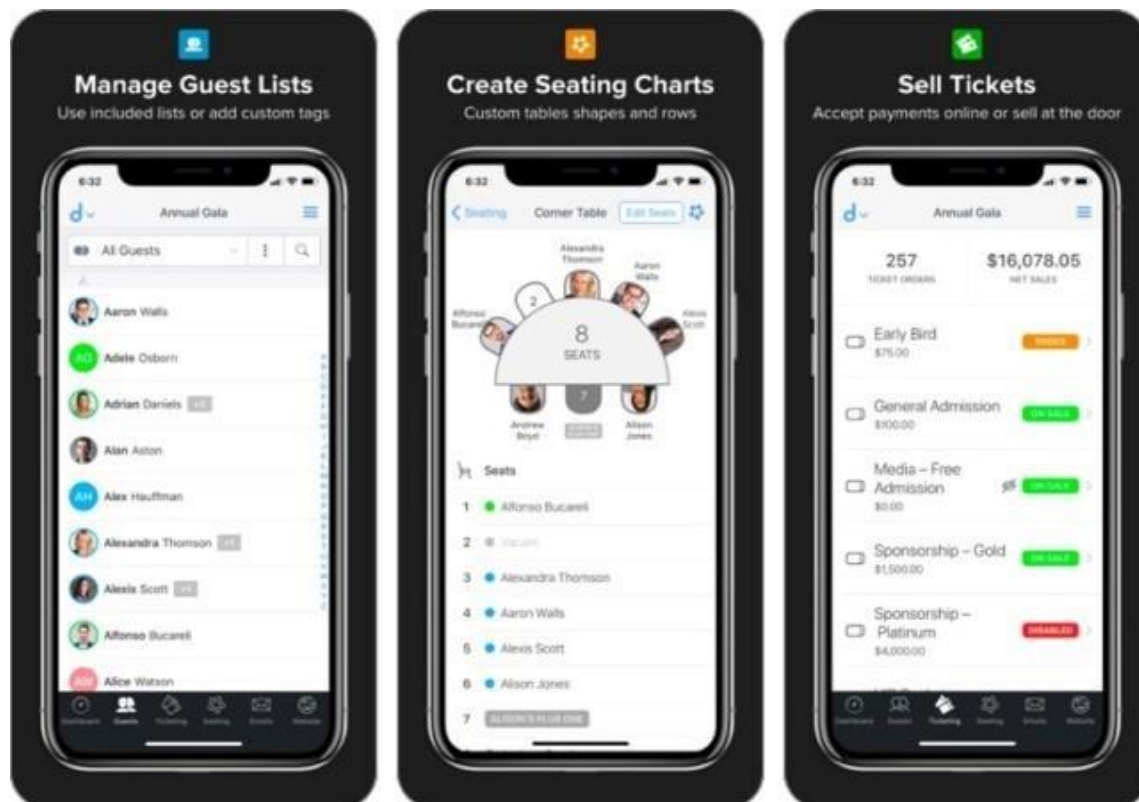


Рисунок 3 - Внешний вид Diobox

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 мобильного приложения были выбраны:

Сервер:

- ОС MacOS BigSur, Windows 10;
- Frameworks: Spring boot 2.6.6, Hibernate, Lombok;
- БД – MySQL;
- Язык разработки - Java v. 17;
- Используемая IDE: IntelliJ IDEA 2021.3.2 (Ultimate Edition);
- Система контроля версий – Git;

Клиент:

- ОС Windows 10;
- Игровой движок Unity 2020.3.32f1;
- Используемая IDE: IntelliJ;
- Система контроля версий – Git;

Для серверной части был выбран стек Java + 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;
- Авто конфигурирование огромного количества компонентов.

#### **3.2 Реализация серверной части приложения**

##### **3.2.1 Схема базы данных**

На рисунке 4 представлена схема базы данных.

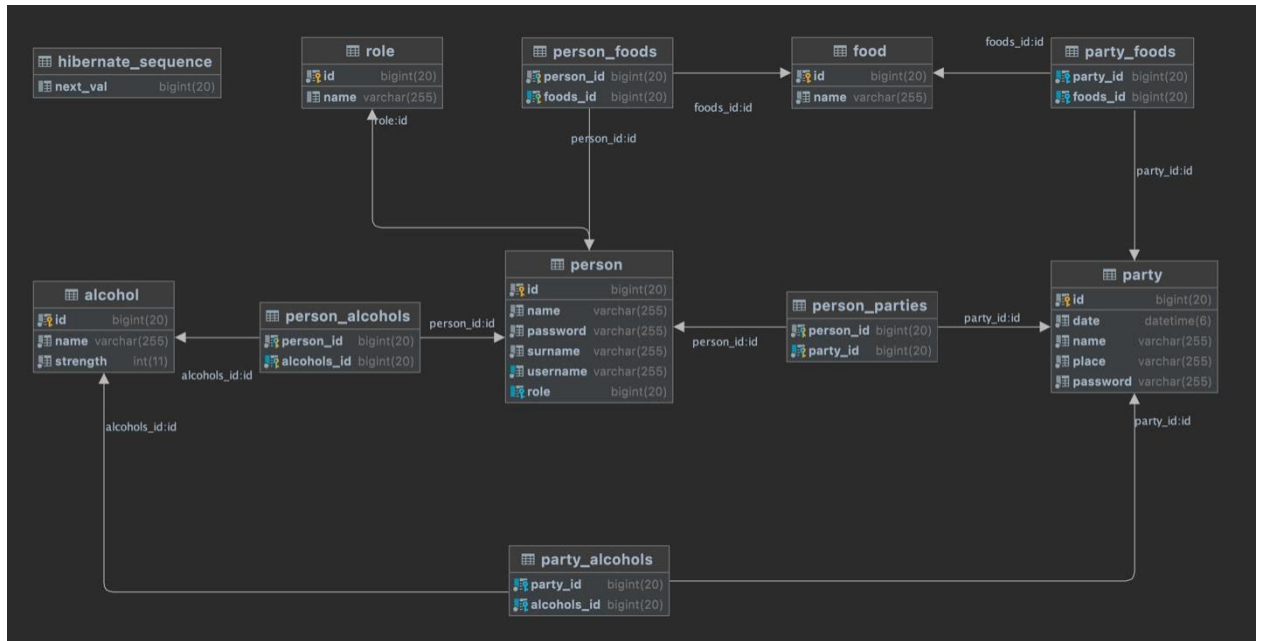


Рисунок 4 - Схема базы данных

### 3.2.2 Диаграмма классов

#### 3.2.2.1 Диаграмма классов сущностей

На рисунке 5 представлена диаграмма классов сущностей.



Рисунок 5 - Диаграмма классов сущностей

Поля каждого из этих классов эквивалентны атрибутам одноименных таблиц в БД.

### 3.2.2.2 Диаграмма классов сервисов

На рисунке 6 представлена диаграмма классов сервисов.

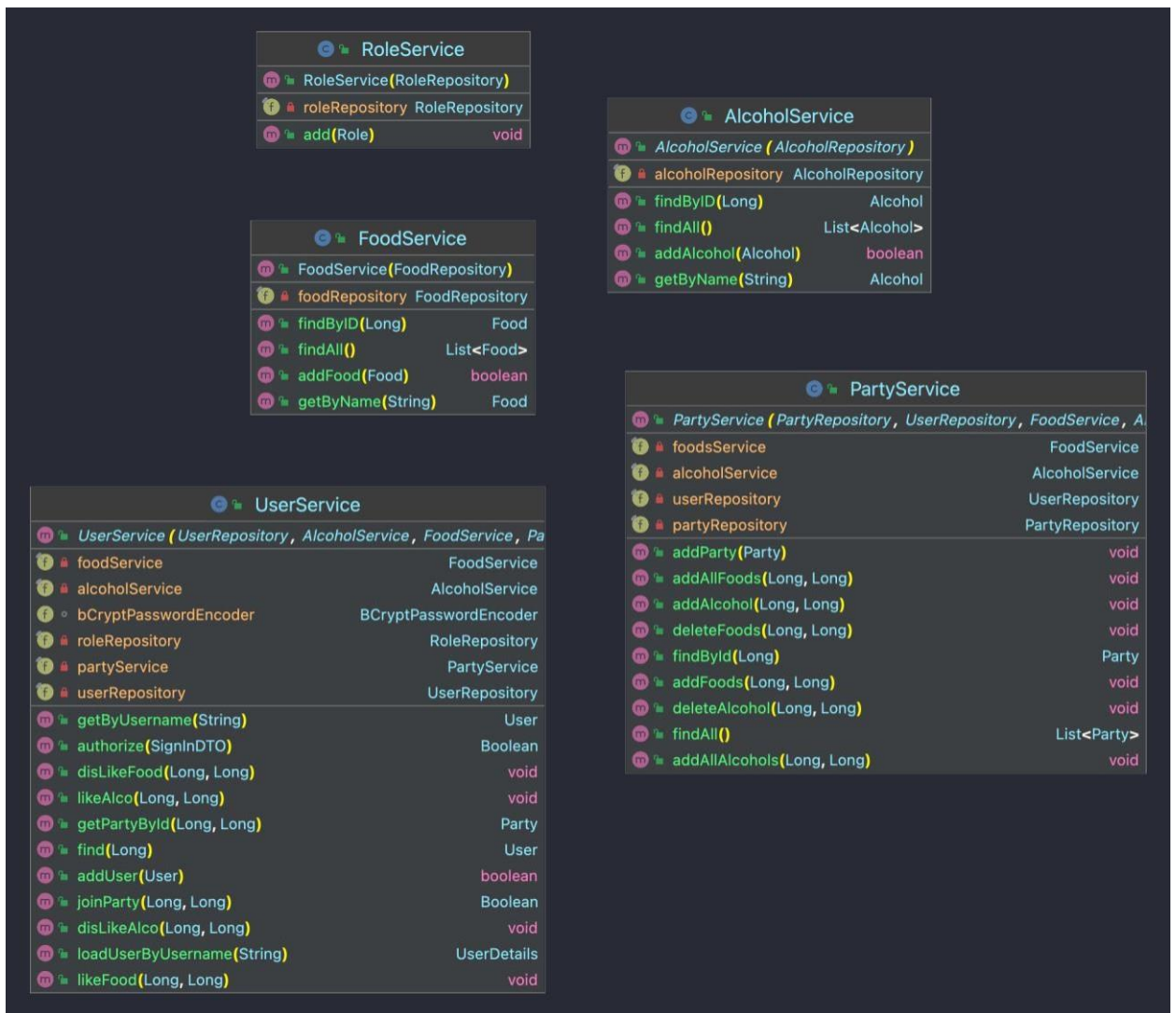


Рисунок 6 - Диаграмма классов сервисов

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

На рисунке 7 представлена диаграмма классов контроллеров.



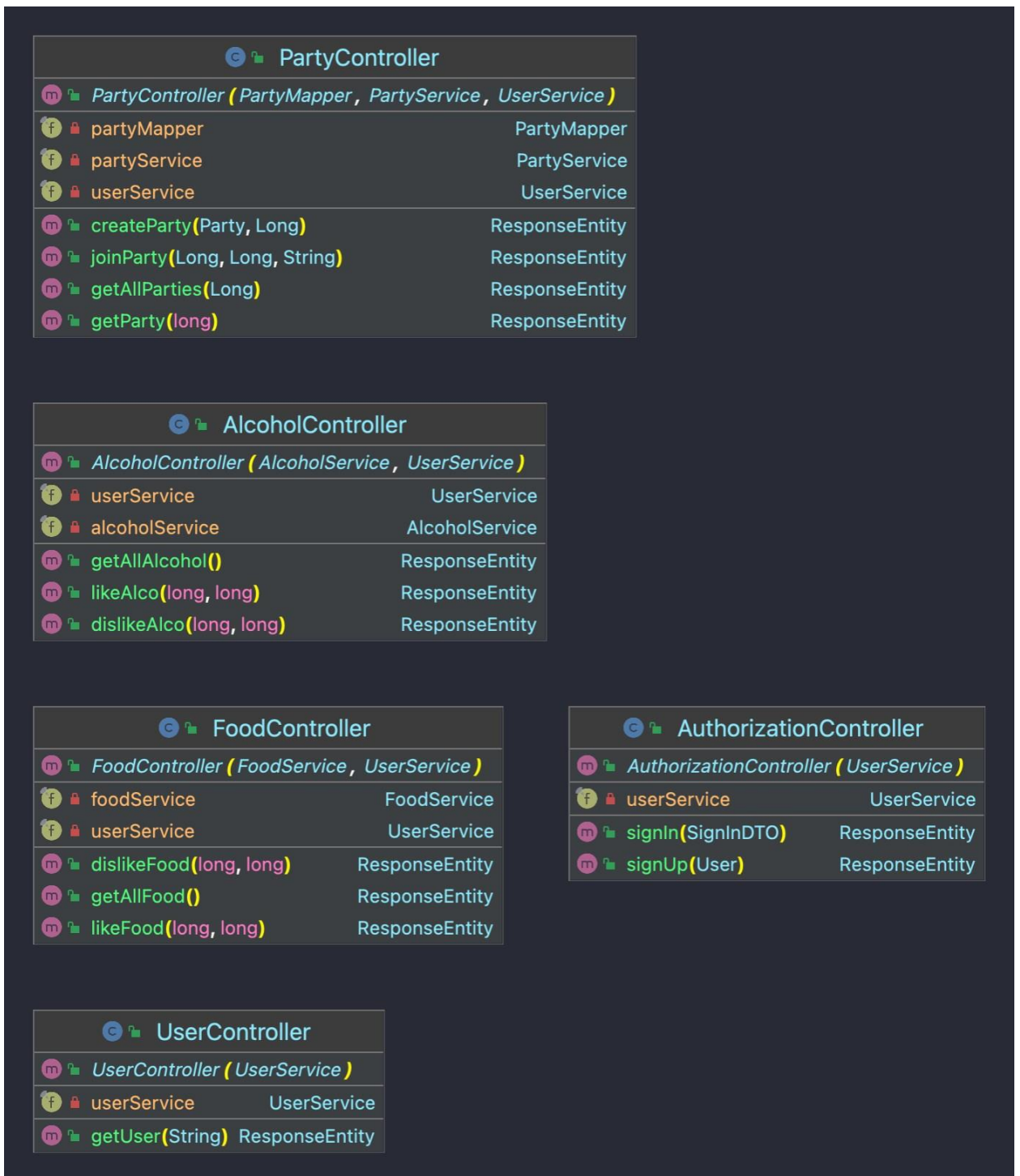


Рисунок 7 - Диаграмма классов контроллеров

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

### 3.2.2.4 Диаграмма служебных классов

На рисунках 8 и 9 представлены диаграммы служебных классов.



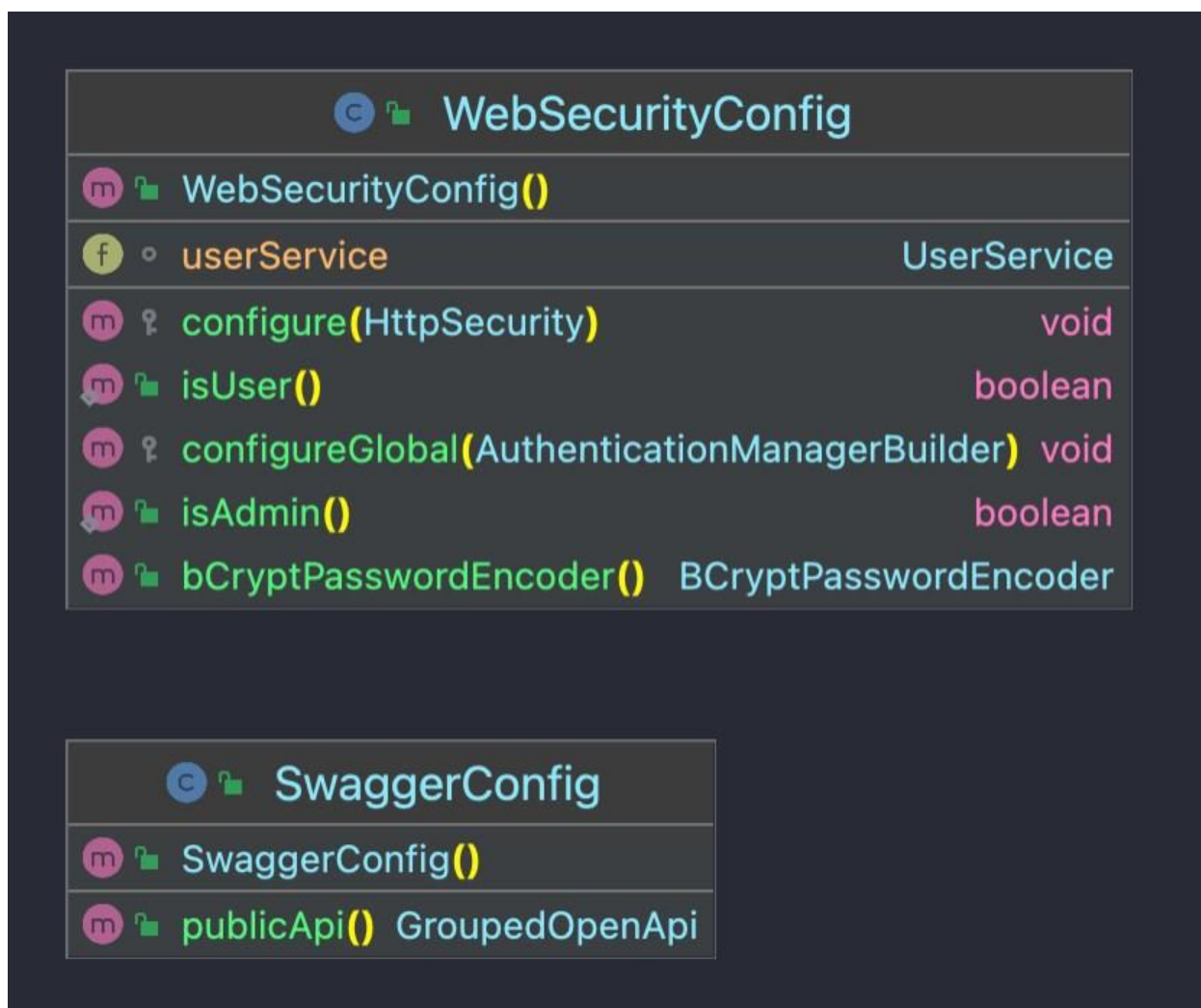


Рисунок 8 - Диаграмма служебных классов



Рисунок 9 - Диаграмма служебных классов

На рисунке 10 представлена точка входа в приложении.

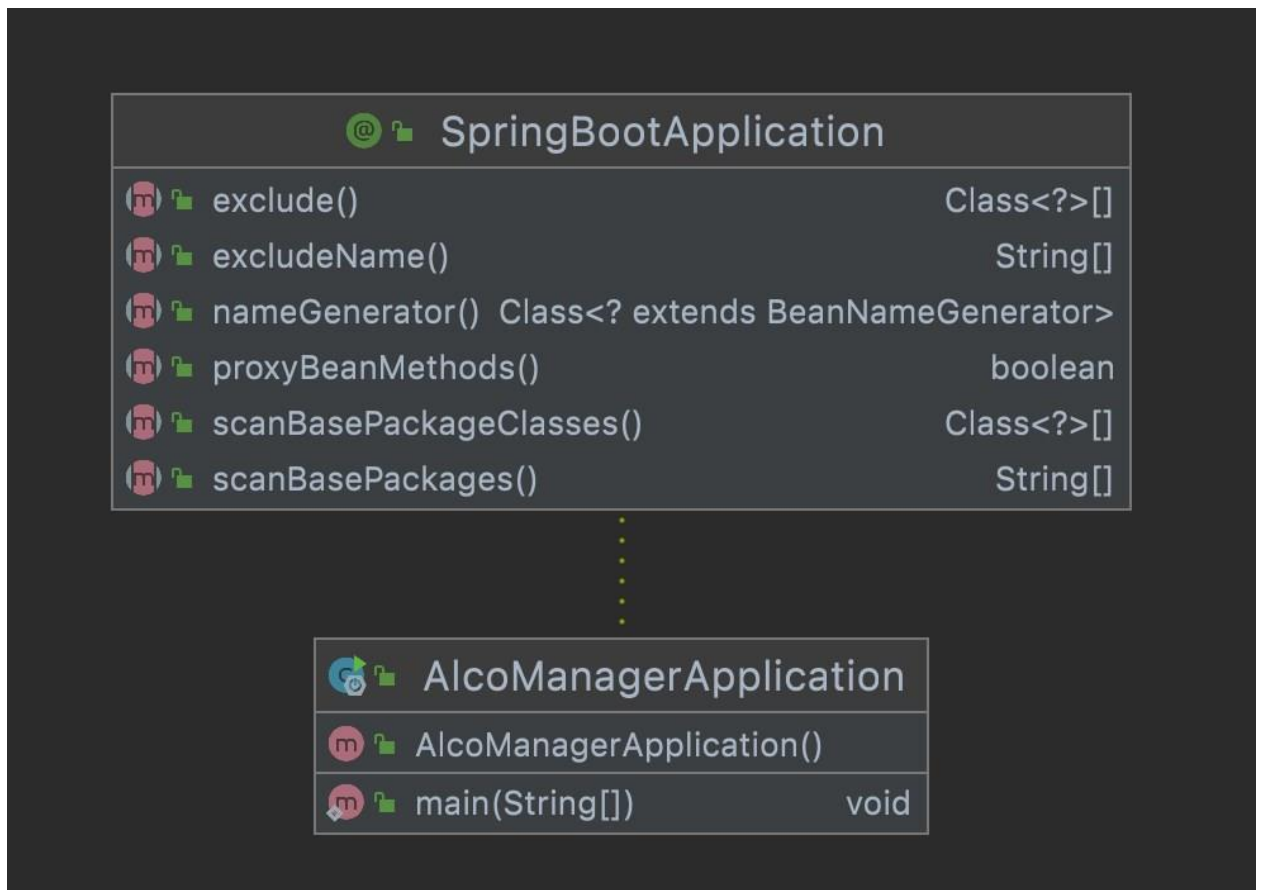


Рисунок 10 - Точка входа

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

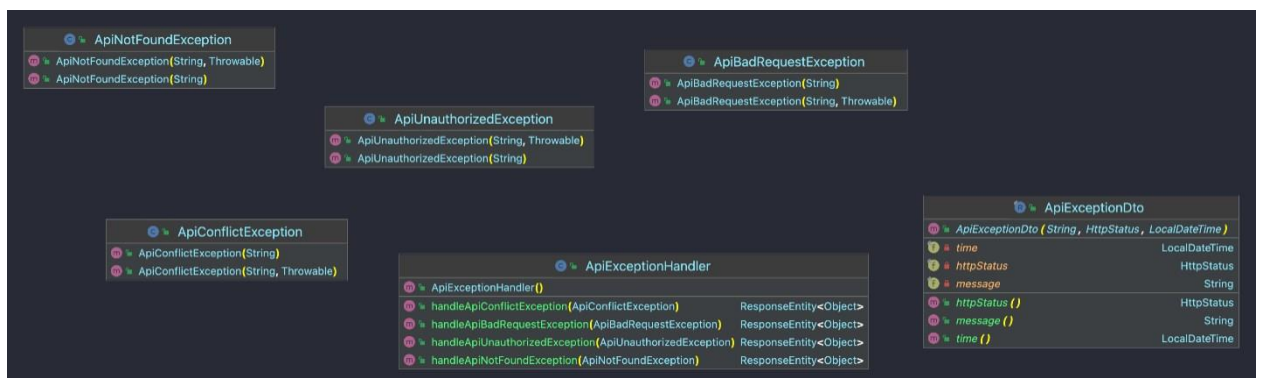


Рисунок 11 - ExceptionHandler

Класс ApiExceptionHandler 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 в

возвращают контроллеру. Такие типичные 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.

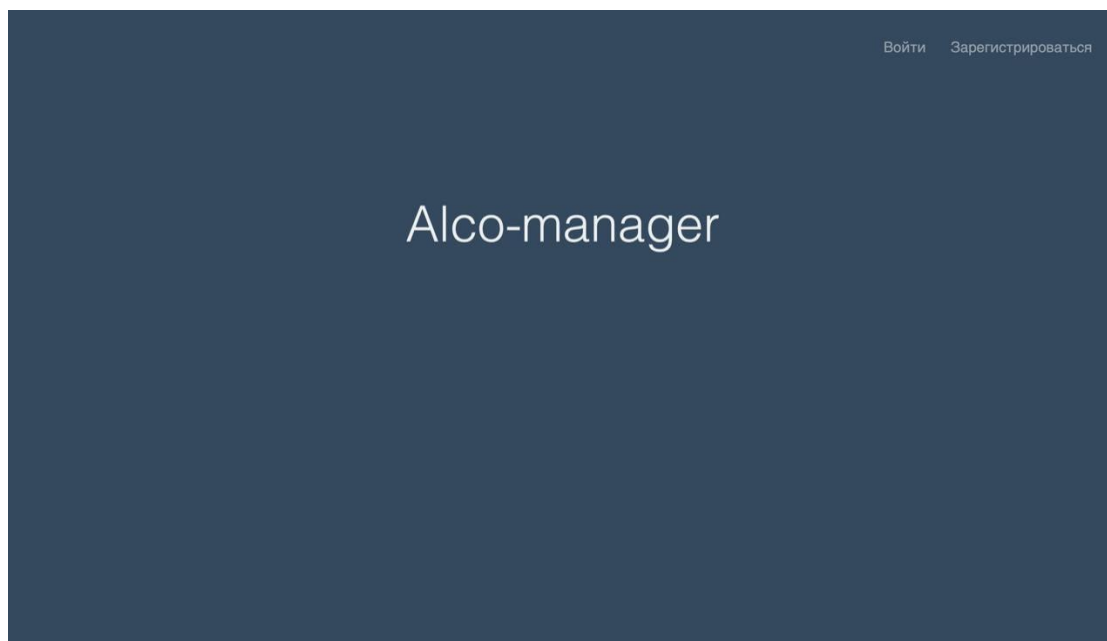


Рисунок 14 - Начальный экран приложения

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.

Рисунок 15 - Экран регистрации

Теперь рассмотрим случай, когда пользователь уже зарегистрирован и хочет войти в свою учетную запись.

После нажатия кнопки «Войти» пользователь увидит экран, на котором ему будет предложено ввести свои учетные данные.

После нажатия кнопки «Войти», если пользователь ввел нужные данные, он увидит главный экран приложения для пользователя, вошедшего в учетную запись.

На рисунке 16 представлен экран авторизации.

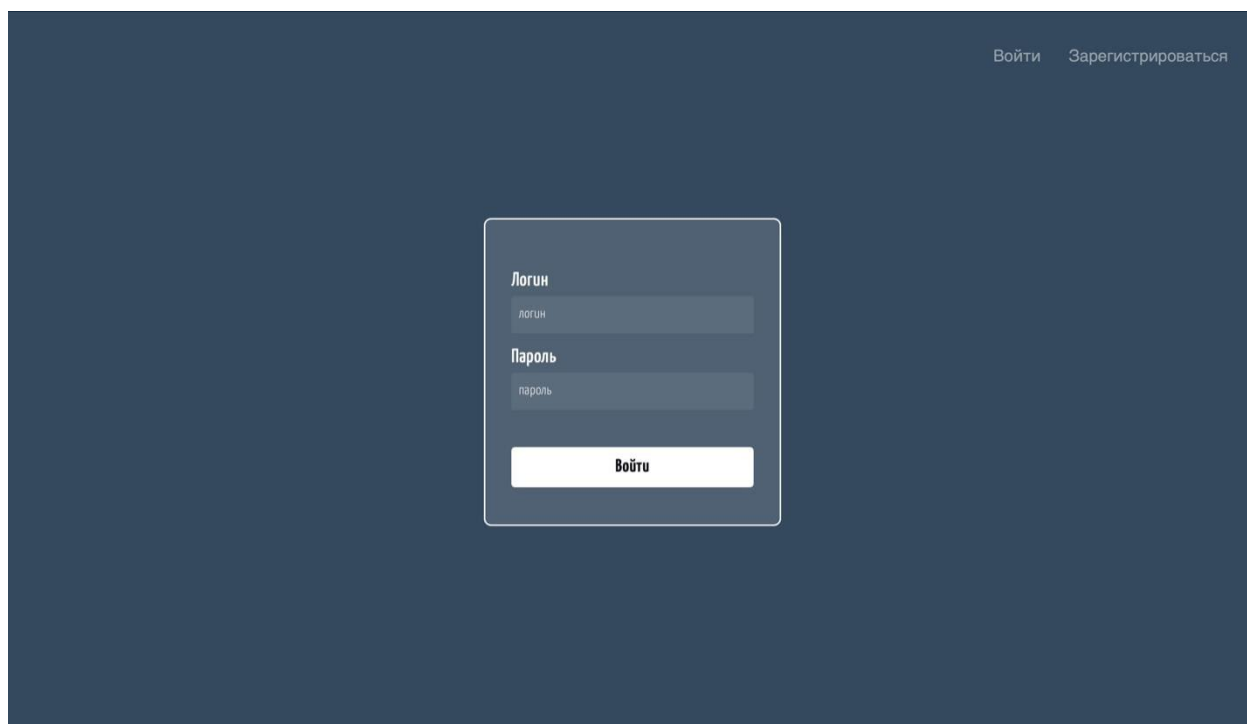


Рисунок 16 - Экран входа в учетную запись

На главном экране находятся кнопки для перехода к экрану с созданными им ботами, экрану поиска ботов других пользователей, экрану просмотра и редактирования информации учетной записи.

На рисунке 17 представлен главный экран приложения для обычного пользователя.

На рисунке 18 представлен главный экран приложения для администратора.

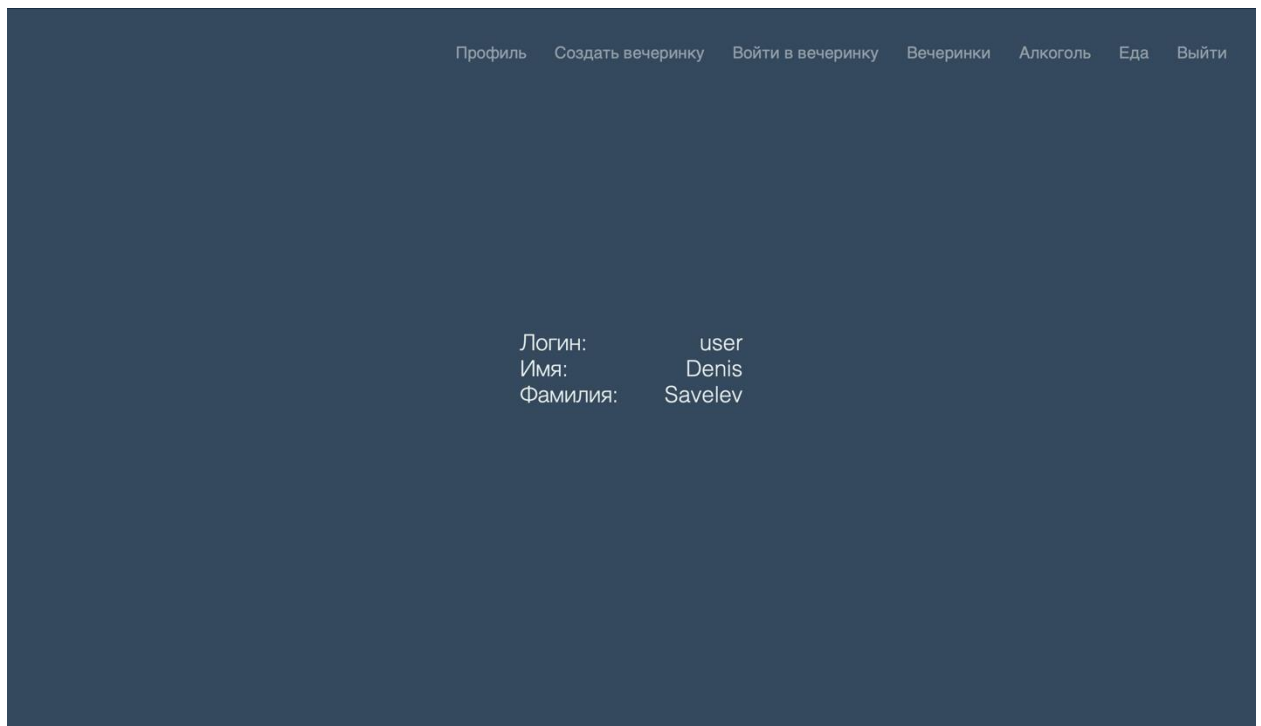


Рисунок 17 - Главный экран приложения для обычного пользователя

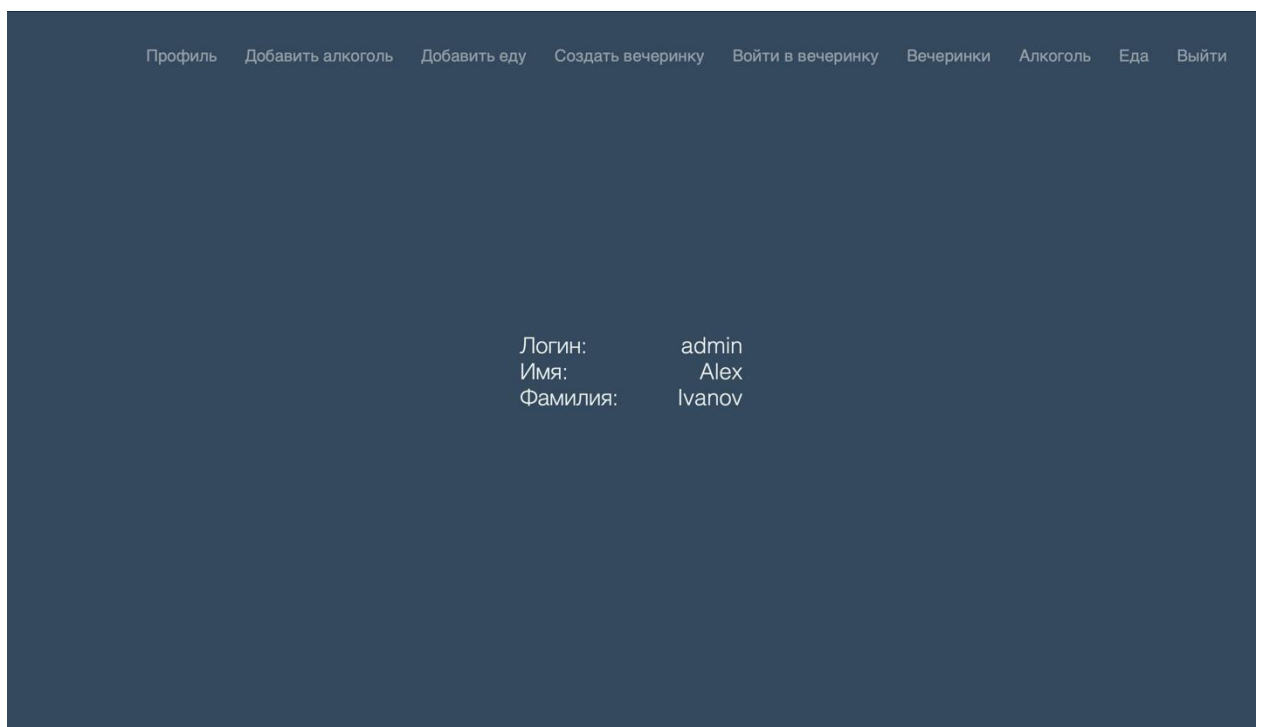


Рисунок 18 - Главный экран приложения для администратора

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

На рисунке 19 представлена страница добавления алкоголя.

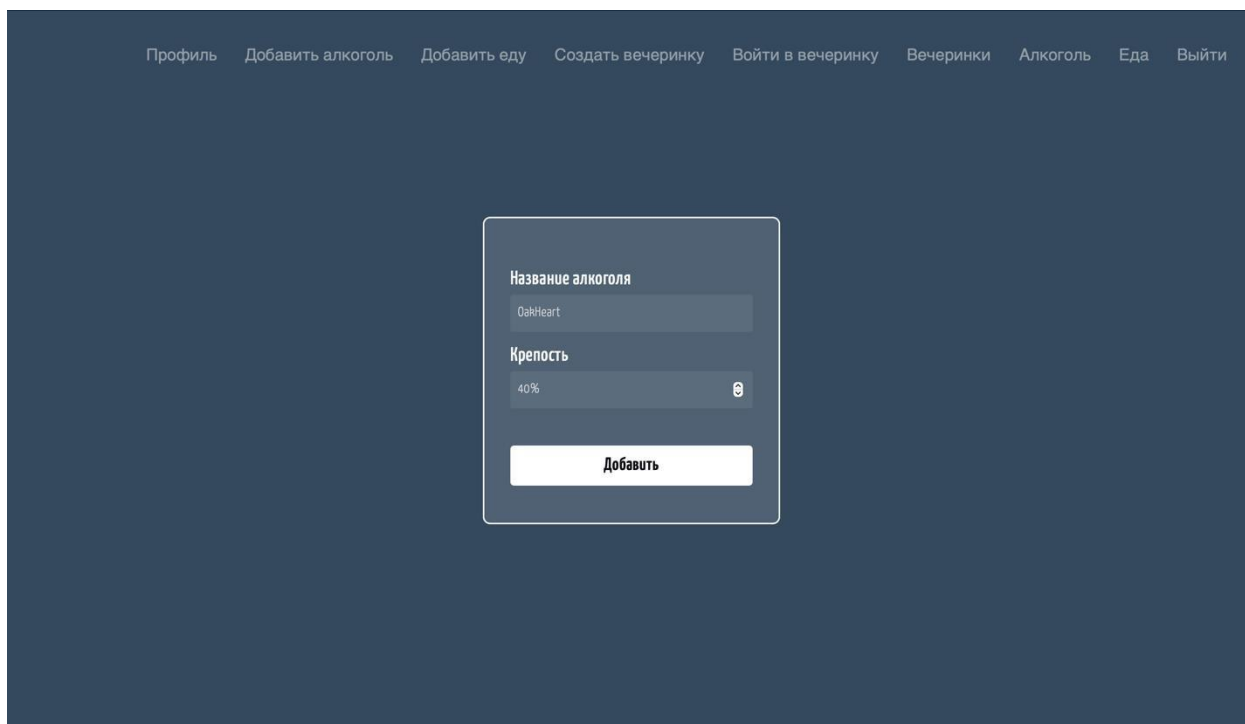


Рисунок 19 - Экран «Добавить алкоголь»

На данной странице можно добавлять алкоголь в БД, после успешного добавления администратор останется на этой странице

На рисунке 20 представлена страница добавления еды.

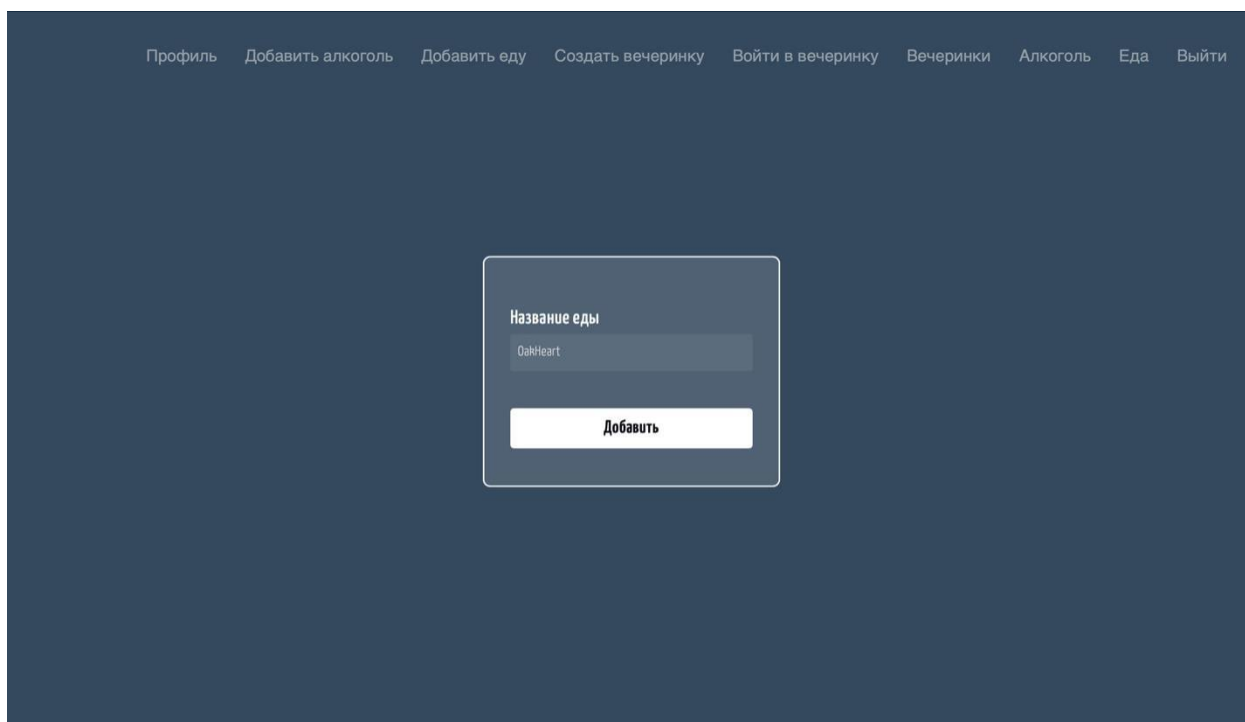


Рисунок 20 - Экран «Добавить еду»



На рисунке 21 представлена страница создания вечеринки.

На данной странице можно добавлять еду в БД, после успешного добавления администратор останется на этой странице.

Далее страницы идентичны для обеих ролей пользователей.

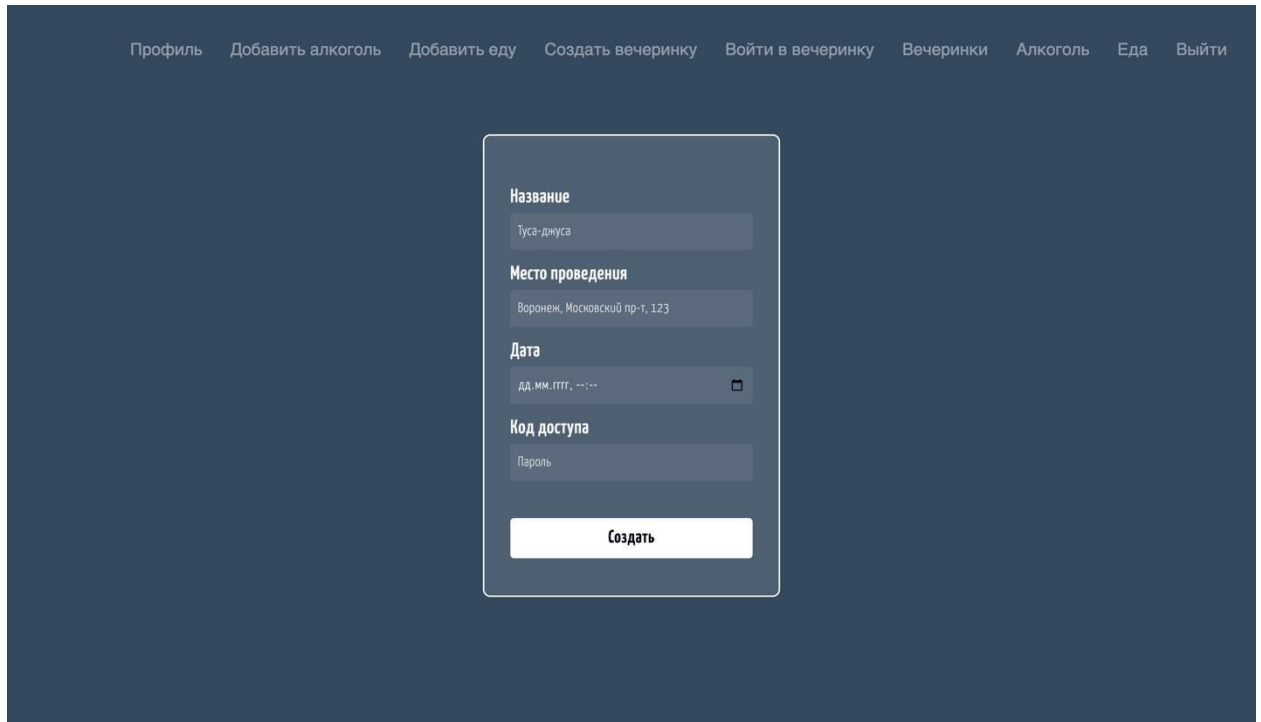


Рисунок 21 - Экран «Создать вечеринку»

На данной странице можно создавать вечеринки, указав все нужные данные, такие как Название мероприятия, Место проведения, Дата и код доступа, по которому другие пользователи смогут присоединиться к вечеринке

На рисунке 22 представлена страница входа в уже созданную вечеринку.

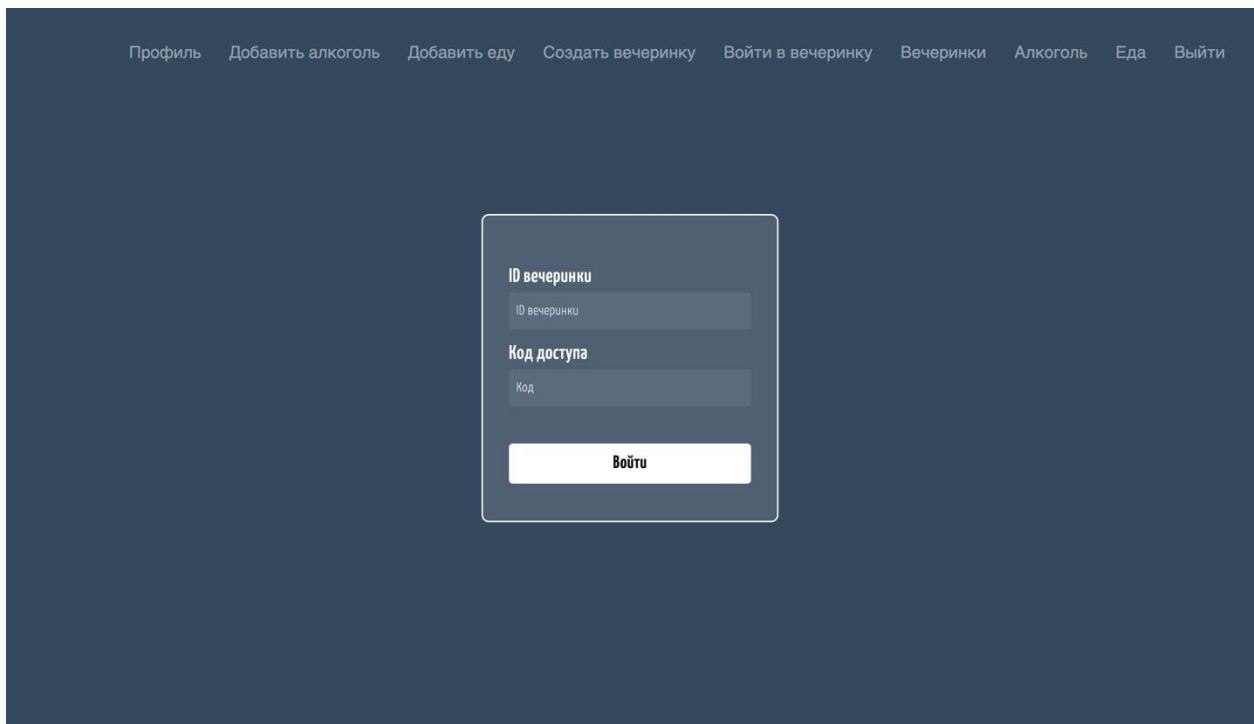


Рисунок 22 - Экран «Войти в вечеринку»

После ввода данных пользователь присоединяется к вечеринке, все его предпочтения добавляются в предпочтения вечеринки. Пользователю после успешного присоединения показывается страница вечеринки, к которой он присоединился.

На рисунке 23 представлена страница текущих вечеринок.

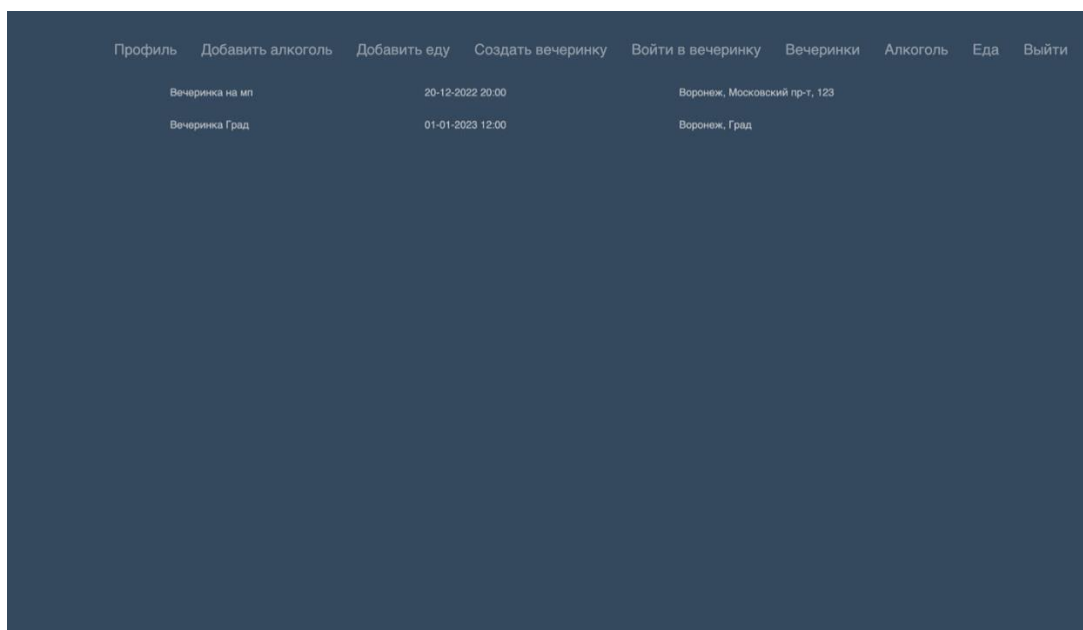


Рисунок 23 - Экран «Вечеринки»

На данной странице отображаются все вечеринки, которые доступны данному пользователю, также можно перейти на какую-либо вечеринку нажав на ее название.

На рисунке 24 представлена страница конкретной вечеринки.

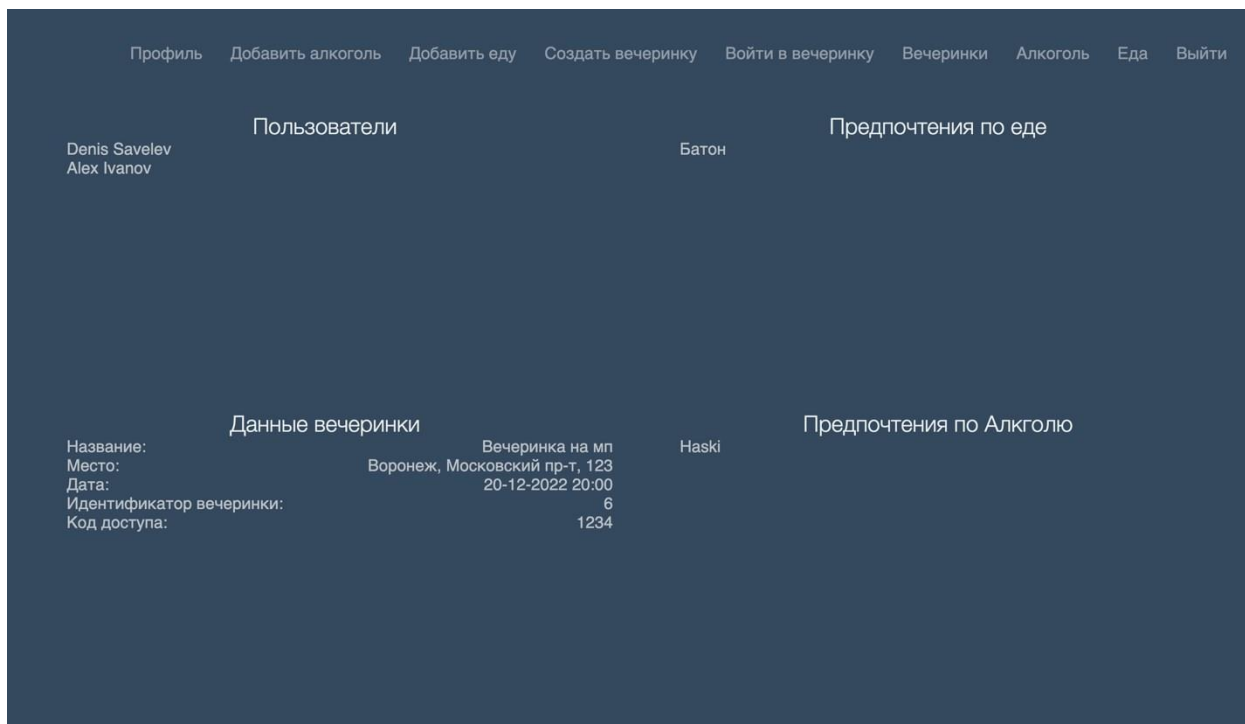


Рисунок 24 - Экран конкретной вечеринки

На данной странице отображаются предпочтения всех пользователей, которые есть в данной вечеринке, по еде и алкоголю, также сам список пользователей и данные о вечеринке

На рисунке 25 представлена страница текущего алкоголя.

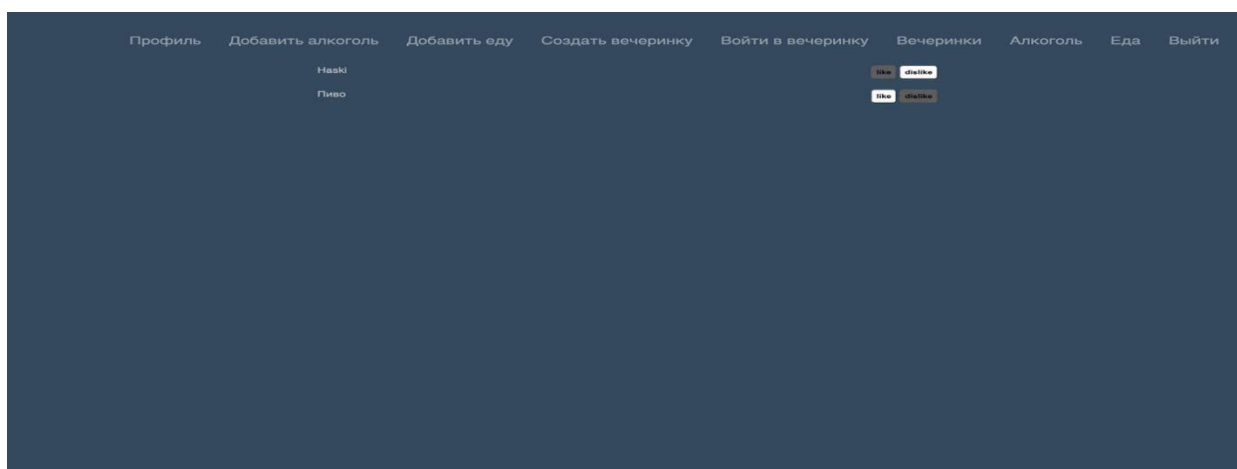


Рисунок 25 - Экран «Алкоголь»

На данной странице можно отметить понравившийся алкоголь, далее он будет отображаться в предпочтениях всех вечеринок, к которым вы причастны

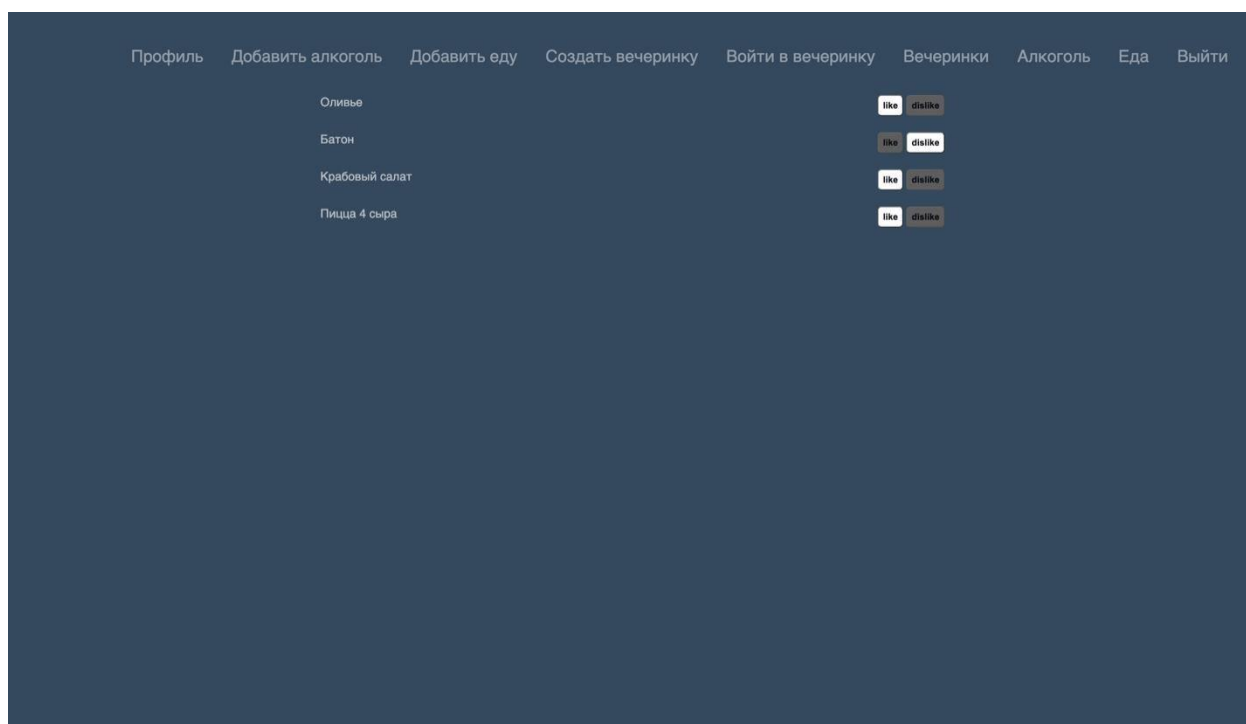


Рисунок 26 - Экран «Еда»

На 26 странице можно отметить понравившуюся еду, далее она будет отображаться в предпочтениях всех вечеринок, к которым вы причастны

Кнопка «Выйти» позволяет выйти из учетной записи и вернуться на начальный экран приложения.

## Заключение

В ходе выполнения данного курсового проекта, командой было разработано веб-приложение для учета и отслеживания вечеринок. Был выполнен анализ предметной области и аналогов разрабатываемого приложения.

Для разработки приложения были разработаны макеты интерфейса, выбрана платформа приложения.

Для контроля версий был создан репозиторий GitHub.

При разработке приложения было реализовано следующее:

- Пользовательский интерфейс
- Бизнес-логика приложения
- Слой доступа к базе данных
- Разработана база данных
- Модуль авторизации

Разработанное приложение удовлетворяет поставленным требованиям.

Все поставленные задачи были выполнены.

В качестве дальнейшего развития проекта рассматриваются следующие усовершенствования:

- Реализация чата между пользователями одной вечеринки;
- Возможность авторизации через различные ресурсы по типу Google;
- Добавление аналитики предпочтений всех пользователей;
- Создание мобильной версии приложения;
- Усовершенствование различных сущностей, для более подробного описания.

### **Список использованных источников**

1. Документация Spring Data <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.query-methods>
2. Документация Spring Boot <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#documentation.first-step>
3. Документация js <https://developer.mozilla.org/en-US/docs/Web/JavaScript>