ФГБОУ ВПО Уральский государственный горный университет

Инженерно-экономический факультет

Кафедра информ@тики

Курсовой проект

По дисциплине «Технологии программирования» На тему «Разработка веб-приложения для кальянной с системой интерактивного бронирования столов и индивидуальной сборки кальяна»

Выполнил:

Студент гр. АУБП-22-2

Адуллин Р.А.

Проверила:

ст. преп. каф. Информатики

Волкова Е.А.

Екатеринбург, 2025г.

Содержание

[1. Постановка задачи 3](#_Toc199329917)

[1.1. Характеристика объекта и проблемной области 3](#_Toc199329918)

[1.2. Постановка целей и задач автоматизации/информатизации 4](#_Toc199329919)

[1.3. Краткое техническое задание 5](#_Toc199329920)

[2. Проектирование системы 7](#_Toc199329921)

[2.1. Моделирование системы 8](#_Toc199329922)

[2.2. Проектирование БД 11](#_Toc199329923)

[2.3. Проектирование интерфейса 13](#_Toc199329924)

[3. Разработка системы 16](#_Toc199329925)

[3.1. Выбор средств реализации 17](#_Toc199329926)

[3.2. Структура проекта 18](#_Toc199329927)

[3.3. Реализация 23](#_Toc199329928)

1. Постановка задачи
   1. Характеристика объекта и проблемной области

Объектом информатизации является кальянная, предоставляющая услуги аренды столов, приготовления кальяна с индивидуальным выбором составляющих, а также развлечений (PlayStation и др.).

В настоящее время бронирование столов и оформление заказов на кальяны происходит вручную: по телефону или в мессенджерах, что приводит к:

ошибкам в учете заказов,

накладкам во времени бронирования,

отсутствию персонализированного подхода к клиенту,

невозможности быстро получить статистику по заказам.

Для устранения этих проблем предлагается автоматизировать процессы бронирования и заказа кальянов с помощью веб-приложения..

* 1. Постановка целей и задач автоматизации/информатизации

Основная цель автоматизации – создание интеллектуальной системы управления умным Цель проекта — разработка веб-приложения для автоматизации процессов бронирования столов и индивидуальной сборки кальяна в кальянной.

Для достижения цели необходимо решить следующие задачи:

* Обеспечить регистрацию и авторизацию пользователей с разграничением прав (администратор/клиент);
* Разработать модуль бронирования столов по дате и времени с учетом доступности;
* Создать визуальный конструктор кальяна с выбором табаков и добавок;
* Реализовать административную панель для просмотра и управления бронями и заказами;
* Обеспечить сохранность данных с помощью СУБД и реляционной модели.
  1. Краткое техническое задание

Наименование системы: Веб-приложение для автоматизации бронирования кальянной и индивидуальной сборки кальяна.

Назначение системы: Автоматизация процессов бронирования столов, оформления заказов на кальяны с индивидуальной настройкой и управление расписанием бронирований через веб-интерфейс.

Функциональные требования (F — Functionality):

1. Система должна обеспечивать регистрацию и аутентификацию пользователей с разграничением ролей (администратор, клиент).
2. Клиент должен иметь возможность просматривать доступные столы на выбранную дату и время.
3. Клиент должен иметь возможность выбрать стол, указать время начала и окончания бронирования (интервалы по 1 часу, максимум 3 часа) и оформить бронь.
4. Клиент должен иметь возможность создать индивидуальный заказ кальяна, выбрав до 3 вкусов табака и любые доступные добавки.
5. Система должна отображать только доступные столы на указанный временной интервал.
6. Администратор должен иметь возможность просматривать, редактировать и удалять записи о бронированиях и заказах кальянов.
7. Администратор должен иметь доступ к списку пользователей с возможностью управления их данными.

Требования к удобству использования (U — Usability):

1. Веб-интерфейс должен быть реализован на русском языке.
2. Основные действия (бронирование, создание кальяна) должны быть доступны не более чем за 3 клика от главной страницы.
3. Интерфейс должен быть адаптирован для экранов шириной от 1024px и выше.
4. Цветовая схема интерфейса: фон — #1c1c1c, текст — #ffffff, акцентные цвета — #d8955b, #8ec6c5.
5. Надёжность (R — Reliability):
6. Система должна обеспечивать корректную работу при одновременной работе не менее 10 пользователей.
7. Все операции, связанные с бронированием и заказом кальянов, должны сопровождаться валидацией и подтверждением.
8. При сбое соединения с базой данных пользователь должен получать понятное сообщение об ошибке.

Производительность (P — Performance):

1. Время отклика при переходе между шагами бронирования и оформления заказа не должно превышать 2 секунд при нормальной нагрузке.
2. Система должна поддерживать хранение информации о не менее чем 5000 бронированиях и заказах без снижения производительности.

Поддержка (S — Supportability):

1. Система должна быть реализована с использованием Java 21, Spring Boot 3.x, MySQL 8.x, что обеспечивает кроссплатформенность и простоту развертывания.
2. Проект должен быть предоставлен в виде Maven-проекта с инструкцией по сборке и запуску.
3. Миграции базы данных должны быть оформлены с использованием Flyway.
4. Проектирование системы
   1. Моделирование системы

Проектируемая система включает двух основных участников: пользователя и администратора. Для представления логики взаимодействия использованы диаграмма вариантов использования (Use Case) и контекстная диаграмма, отражающая связи между внешними участниками и подсистемами.

Диаграмма вариантов использования (Use Case Diagram)

На диаграмме (рис. 1) представлены следующие варианты использования:

* Регистрация и вход в систему;
* Просмотр и бронирование доступных столов;
* Создание индивидуального кальяна;
* Просмотр списка броней и заказов;
* Просмотр пользователей (для администратора);
* Просмотр столов и заказов (для администратора)



Рисунок 1 UML-диаграмма вариантов использования системы

Контекстная диаграмма (рис. 2) демонстрирует основные компоненты системы и взаимодействие между ними.

Контекстная диаграмма отражает взаимодействие основных участников системы — пользователя и администратора — с веб-интерфейсом, системой бронирования кальянной и логикой доступа к данным. Пользователь и администратор работают через браузер, взаимодействуя с представлением (View). Веб-интерфейс обменивается данными с бизнес-логикой, которая, в свою очередь, обращается к базе данных через слой доступа к данным.

Такой подход обеспечивает:

* Четкое разделение уровней ответственности;
* Масштабируемость и простоту поддержки;
* Возможность безопасной работы с данными.

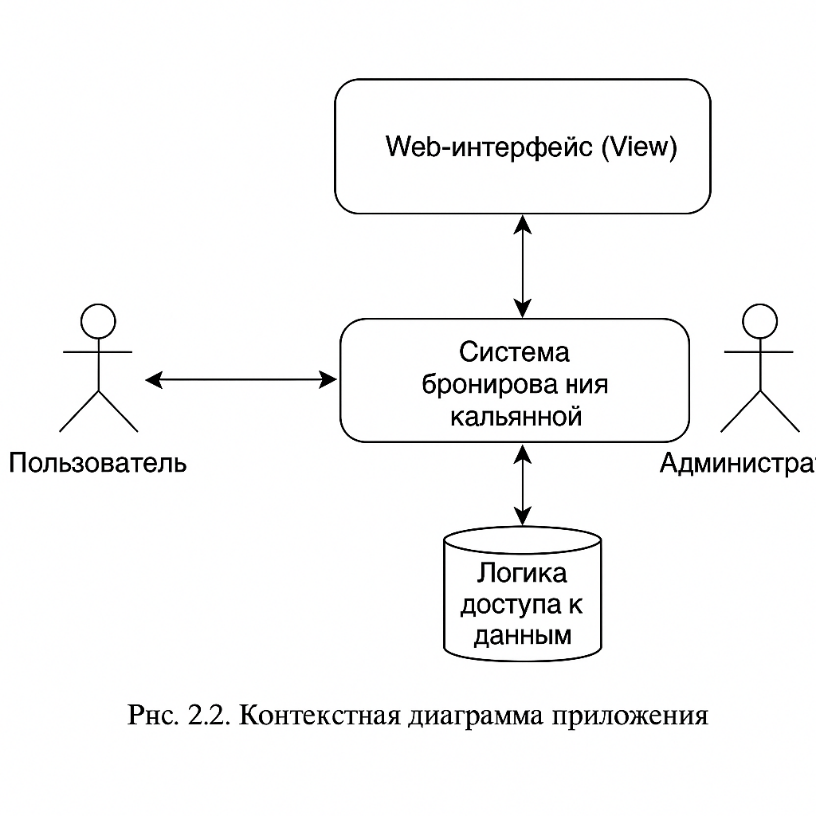


Рисунок 2 Контекстная диаграмма

В качестве архитектурной основы системы выбран паттерн MVC (Model-View-Controller) в сочетании с трехуровневой архитектурой, что позволяет достичь высокой модульности, расширяемости и удобства сопровождения проекта.

1.Выбранные архитектурные решения:

* MVC (Model-View-Controller)
* Model — бизнес-логика, работа с базой данных (JPA-Entity, Repository);
* View — интерфейс пользователя (JSP/Thymeleaf-шаблоны);
* Controller — обработка входящих запросов, маршрутизация и связывание View с Model (Spring MVC).

2.Трехуровневая архитектура

* Уровень представления (Frontend): JSP/Thymeleaf, HTML/CSS, JavaScript;
* Уровень бизнес-логики (Backend): Java 21 + Spring Boot 3, сервисные слои (Service, ServiceImpl);
* Уровень доступа к данным: Spring Data JPA с Hibernate, MySQL, Flyway для миграций.

3.Паттерны проектирования:

* Dependency Injection (внедрение зависимостей) — обеспечивается Spring Framework;
* Repository — используется для абстракции доступа к данным;
* Builder — для гибкого создания объектов кальянов в будущем.

4.Безопасность и сессии:

* Простая авторизация реализуется вручную с проверкой ролей;
* Хранение сессий пользователя через стандартный HTTP Session;
  1. Проектирование БД

В качестве основы для проектирования базы данных использована реляционная модель, реализованная в MySQL. На ER-диаграмме отражены основные сущности системы: пользователи, столы, бронирования, заказы кальянов, вкусы табака, добавки, а также связи между ними.

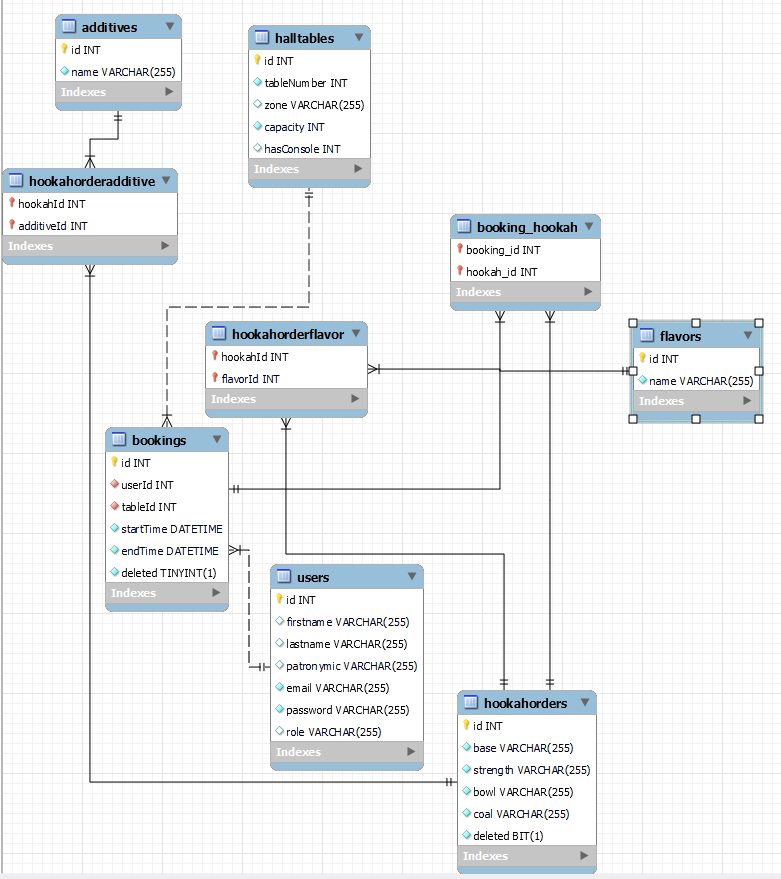


Рисунок 3 ER-диаграмма

Основные таблицы:

* users – пользователи системы;
* hall\_tables – столы кальянной (номер, зона, вместимость, наличие PlayStation);
* bookings – бронирования столов (дата, начальное и конечное время, ссылка на пользователя и стол);
* hookah\_orders – заказы кальянов (связаны с бронированием);
* flavors – вкусы табака;
* additives – добавки (например, лёд, мята);
* hookah\_order\_flavor – связь «многие ко многим» между кальянами и вкусами;
* hookah\_order\_additive – связь «многие ко многим» между кальянами и добавками.

Описание всех основных таблиц и полей в системе:

Пользователи (users)

Хранятся сведения о клиентах и администраторах: уникальный идентификатор (user\_id), электронная почта, хэшированный пароль, имя и отчество, а также роль (клиент или администратор). Роли используются для разграничения прав доступа.

Столы (hall\_tables)

Каждый стол имеет уникальный номер, зону размещения (VIP или стандарт), вместимость (количество человек) и флаг наличия PlayStation. Эта информация используется при бронировании.

Бронирования (bookings)

Содержит записи о бронированиях столов пользователями. Указывается дата, время начала и окончания, идентификатор пользователя (user\_id) и идентификатор стола (hall\_table\_id). В системе реализована проверка на пересечение по времени.

Кальяны (hookah\_orders)

Каждый заказ кальяна привязан к конкретному бронированию. Указывается идентификатор брони, список выбранных вкусов и добавок (через вспомогательные таблицы).

Вкусы (flavors)

Содержит перечень доступных вкусов табака. Каждый вкус имеет уникальный идентификатор и название.

Добавки (additives)

Список дополнительных компонентов к кальяну, таких как лёд, мята, сиропы. Структура аналогична таблице вкусов.

Связи вкусов и добавок с кальяном

Таблицы hookah\_order\_flavor и hookah\_order\_additive реализуют связи многие ко многим между заказами кальянов и выбранными компонентами.

* 1. Проектирование интерфейса

В данном разделе представлены скетчи основных экранов (представлений) приложения для системы умного дома. Интерфейс разрабатывался с учётом удобства пользователя, интуитивной навигации и быстрого доступа к основным функциям.

1. Экран авторизации

Пользователь вводит логин и пароль для входа в систему. На экране предусмотрены ввод логина, пароля, кнопка "Войти", а также, если пользователь ещё не зарегистрирован, то есть ссылка "Зарегистрироваться"

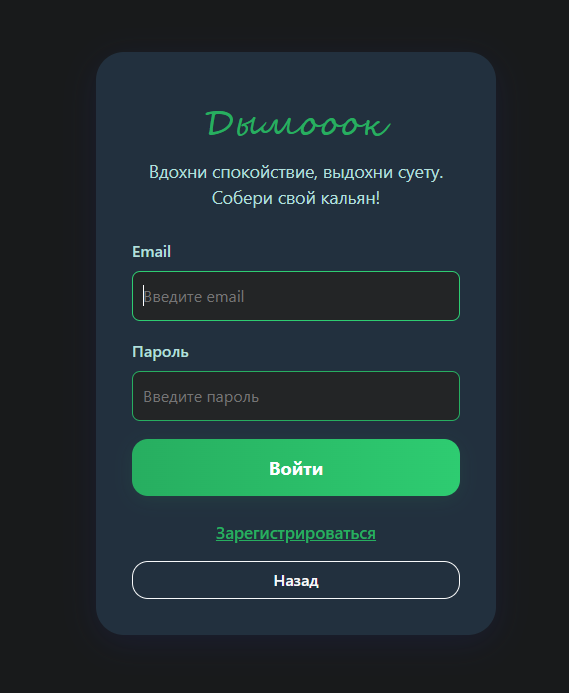


Рисунок 4 Экран аторизации

1. Главная страница пользователя

После входа пользователь видит приветствие по имени и отчеству, а также основные действия: «Забронировать стол», «Мои бронирования», «Выйти». Интерфейс минималистичный, но стильный, с акцентами в фирменных цветах кальянной.

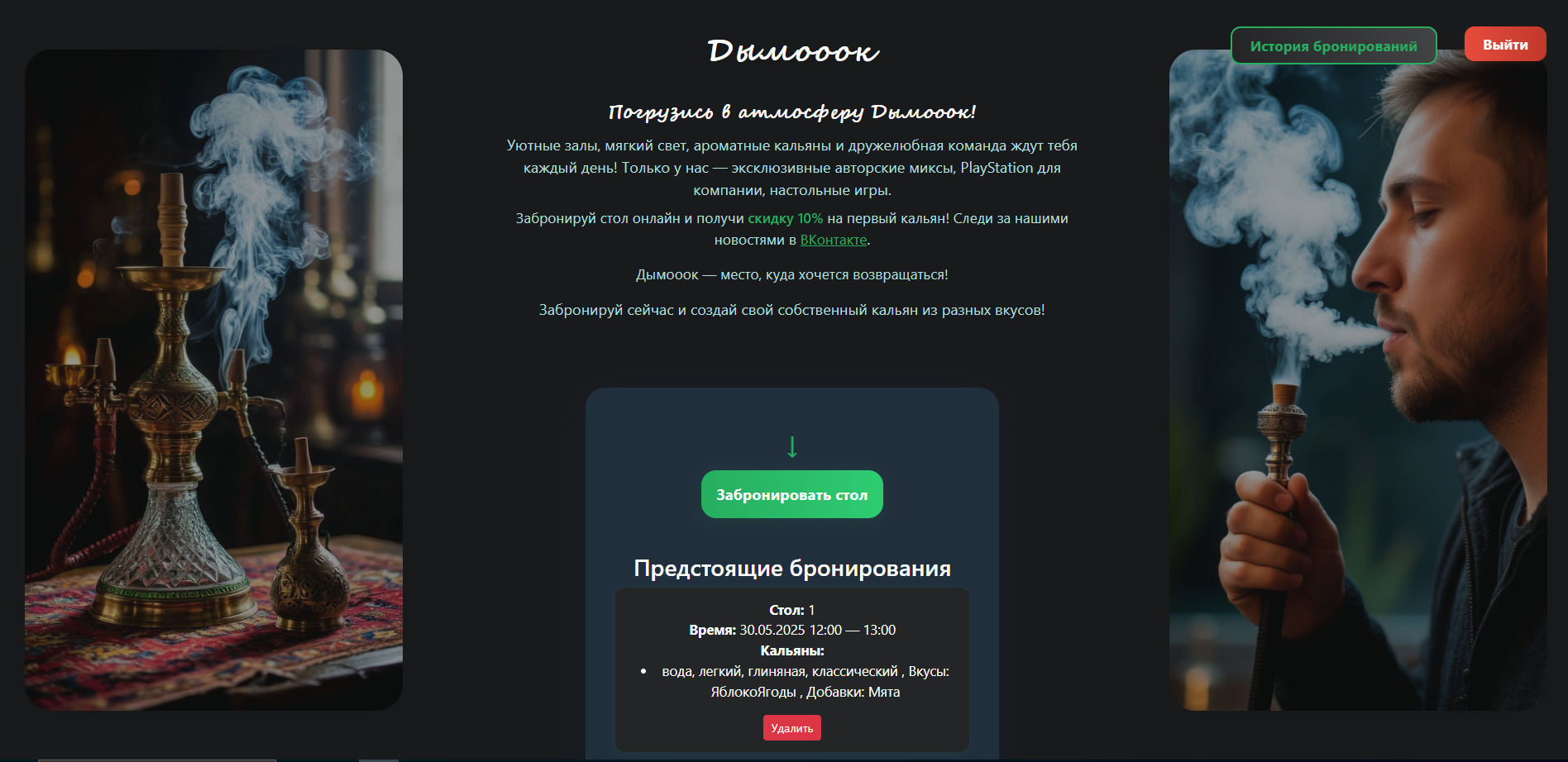


Рисунок 5 Главная страница пользователя

1. Переход к бронированию

Пользователь выбирает дату, затем начальное и конечное время (интервалы по 1 часу, максимум 3 часа). Только после этого отображаются доступные столы.



Рисунок 6 Страница выбора даты бронирования

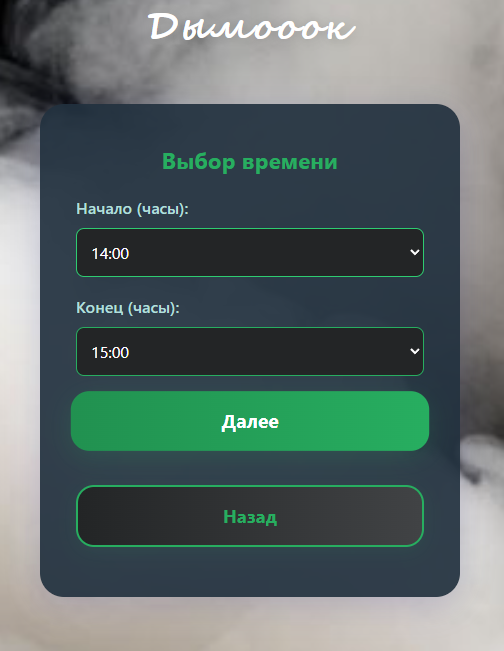


Рисунок 7 Страница выбора времени бронирования

Пользователь видит только свободные столы, отображаемые с параметрами: номер, зона (VIP или стандарт), вместимость, наличие PlayStation. После выбора осуществляется переход к созданию своего кальяна.

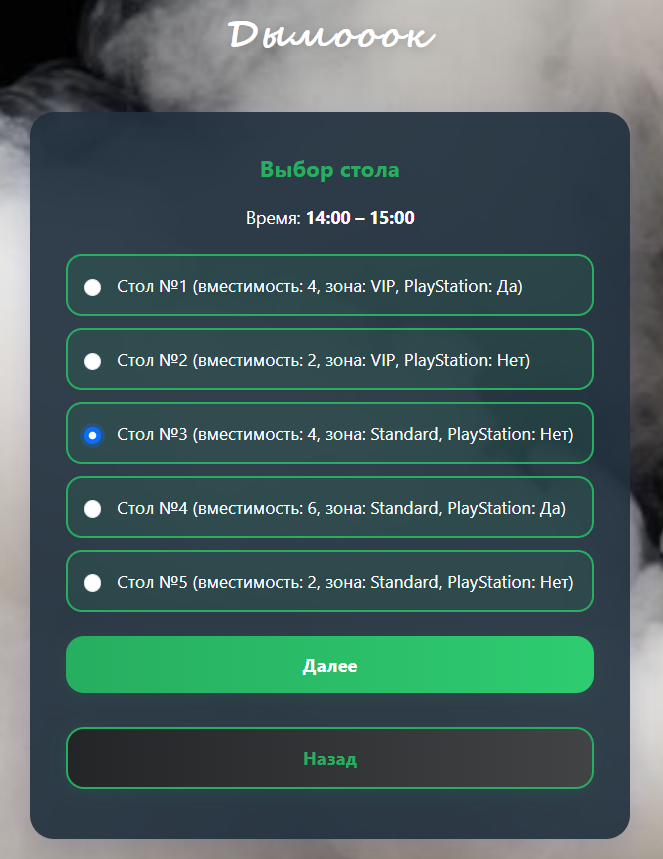


Рисунок 8 Страница выбора стола

Конструктор кальяна

Пользователь может выбрать до 4 вкусов табака и добавить дополнительные ингредиенты (лёд, мята и др.). Интерфейс конструктора визуален и удобен для пользователя

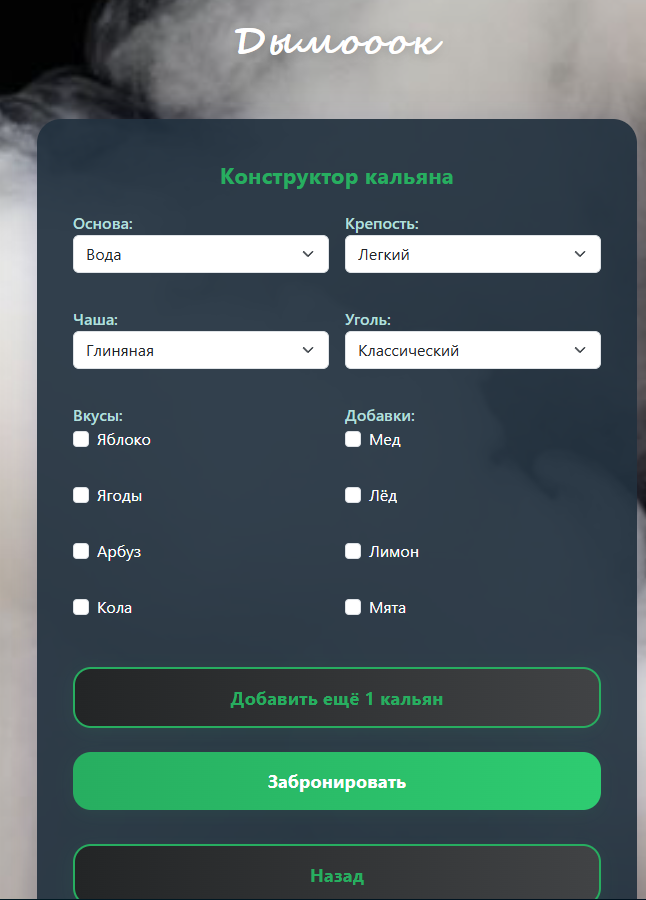


Рисунок 9 Страница создания своего кальяна

1. Панель администратора

Администратор видит общее приветствие, таблицы с пользователями, бронированиями и столами. Также доступна функция редактирования или удаления бронирований. Панель выполнена в том же стиле, что и пользовательский интерфейс, но с расширенным функционалом.

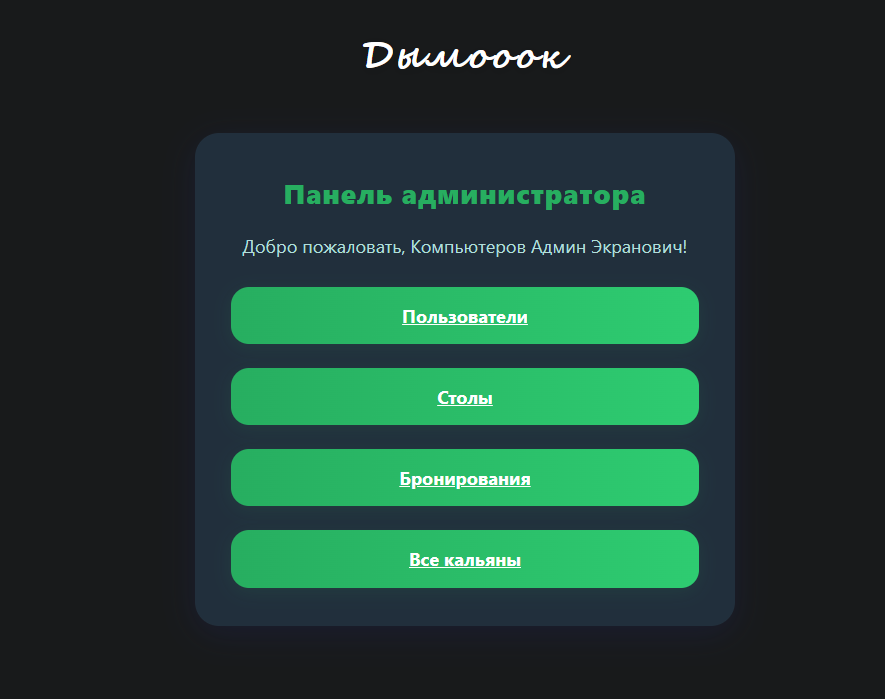


Рисунок 10 Главная страница администратора

1. Разработка системы
   1. Выбор средств реализации

Для разработки веб-приложения «Дымооок» были выбраны следующие технологии:

Язык программирования: Java

Обоснование: Java — один из самых популярных языков для создания enterprise-приложений. Она обладает высокой производительностью, кроссплатформенностью и широким экосистемой библиотек и фреймворков.

Фреймворк: Spring Boot

Обоснование: Spring Boot обеспечивает быстрое создание автономных, готовых к производству приложений. Он предоставляет мощные инструменты для работы с БД, безопасностью, REST API, а также поддерживает внедрение зависимостей и автоматическую конфигурацию.

Шаблонизатор: Thymeleaf

Обоснование: Thymeleaf позволяет генерировать HTML-страницы на стороне сервера с возможностью динамического изменения контента. Он легко интегрируется с Spring и поддерживает прототипирование без необходимости запуска сервера.

База данных: MySQL

Обоснование: MySQL — одна из самых популярных реляционных СУБД. Она обладает высокой производительностью, надежностью и простотой масштабирования, что делает её отличным выбором для долгосрочных проектов. Для хранения пользователей, бронирований, заказов кальянов и других данных используется MySQL.

ORM: Hibernate / JPA

Обоснование: Hibernate реализует стандарт JPA и позволяет работать с объектно-реляционным отображением, упрощая взаимодействие между Java-объектами и таблицами базы данных.

Фронтенд: HTML5 + CSS3 + Bootstrap

Обоснование: Bootstrap предоставляет готовые компоненты и адаптивный дизайн, что ускоряет верстку интерфейса и делает его удобным для пользователей на всех устройствах.

Аутентификация: Spring Security

Обоснование: Spring Security предоставляет мощные механизмы защиты приложения, включая аутентификацию, авторизацию, защиту от CSRF и другие функции безопасности.

Среда разработки: IntelliJ IDEA

Обоснование: Эти IDE предоставляют полную поддержку Spring Boot, встроенный терминал, отладчик, интеграцию с Git и другими инструментами, что повышает производительность разработки.

* 1. Структура проекта

Проект построен по стандартной структуре Spring Boot-приложения. В каталоге src/main/java/com/example размещён основной исходный код, который разделён на отдельные пакеты. Ниже предствлена структура моего проекта.

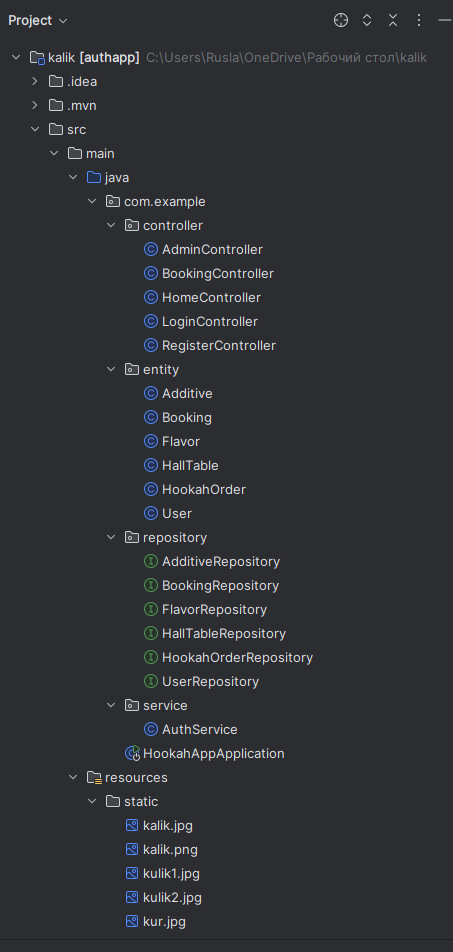


Рисунок 11 Структура проекта1

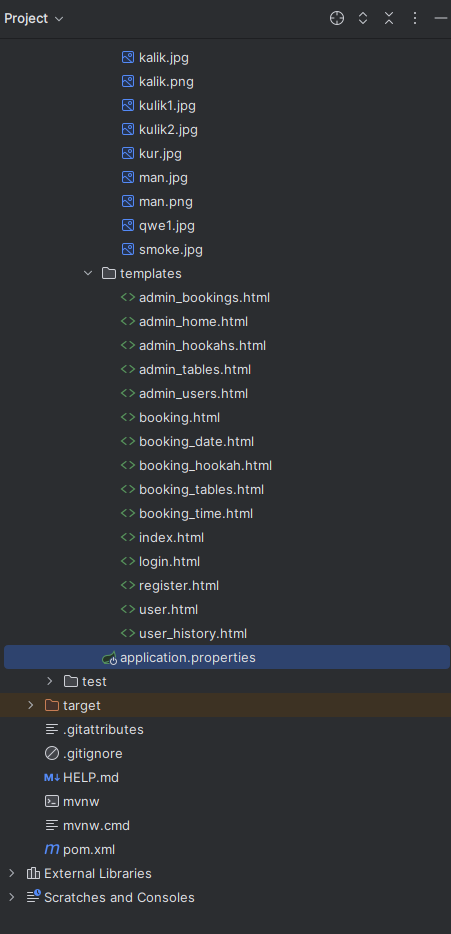


Рисунок 12 Структура проекта2

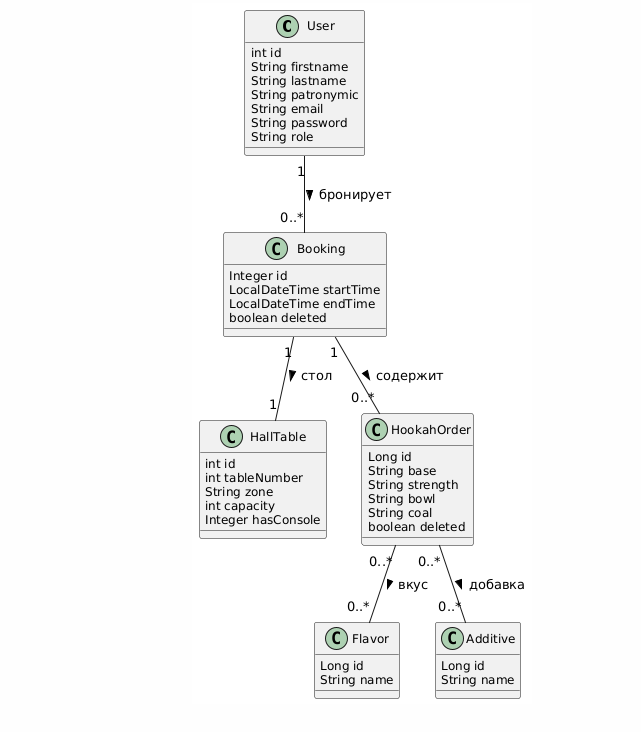


Рисунок 13 UML-диаграмма классов сущностей

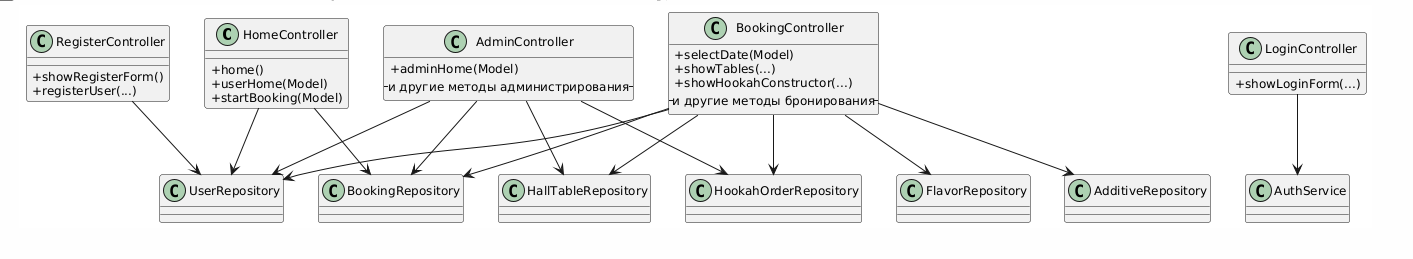


Рисунок 14 UML-диаграмма основных контроллеров проекта

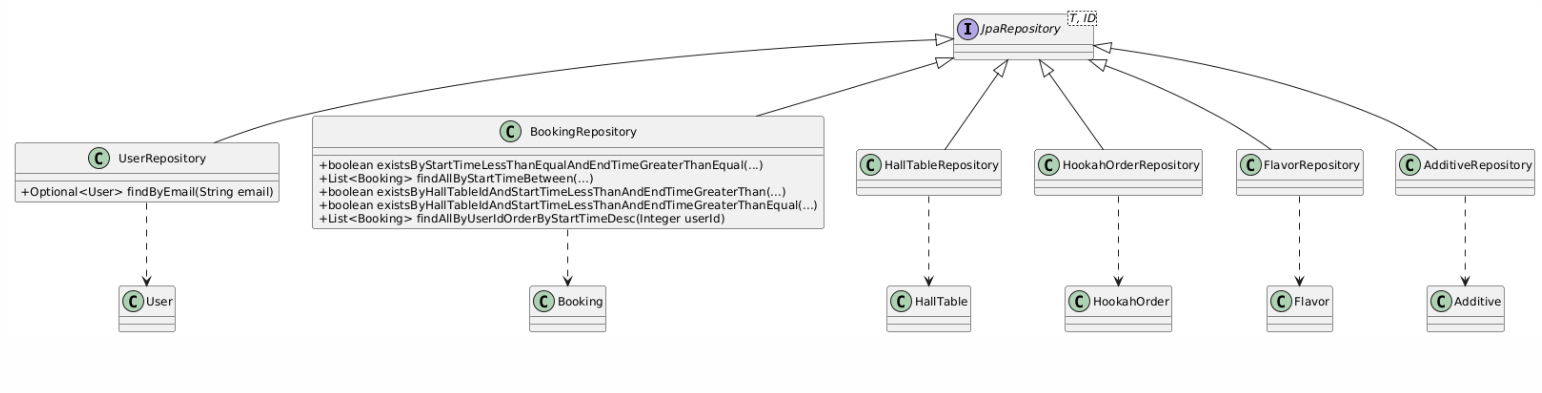


Рисунок 15 UML-диаграмма репозиториев проекта

Краткое описание структуры решения

1. Слои приложения

Контроллеры (Controllers):

Отвечают за обработку HTTP-запросов, маршрутизацию и передачу данных между представлением (HTML-страницами) и бизнес-логикой.

Сервисы (Services):

Содержат бизнес-логику приложения (например, аутентификация, обработка бронирований).

Репозитории (Repositories):

Отвечают за доступ к данным в базе (CRUD-операции для сущностей через Spring Data JPA).

Сущности (Entities):

Классы, отражающие структуру таблиц в базе данных (User, Booking, HallTable, HookahOrder, Flavor, Additive).

2. Основные сущности и их связи

User — пользователь системы (гость или администратор).

Booking — бронирование стола и кальяна пользователем.

HallTable — стол в зале, который можно забронировать.

HookahOrder — заказ кальяна (с выбором вкусов и добавок).

Flavor — вкус для кальяна.

Additive — добавка для кальяна.

Связи:

Один пользователь может иметь много бронирований.

Каждое бронирование связано с одним столом и одним или несколькими кальянами.

Каждый кальян может содержать несколько вкусов и добавок.

3. Взаимодействие компонентов

Пользователь регистрируется и входит в систему.

Через контроллеры пользователь выбирает дату, время, стол, формирует заказ кальяна.

Все данные сохраняются через репозитории в базе данных.

Администратор может просматривать все бронирования и управлять меню кальянов.

* 1. Реализация

В данном разделе приведены примеры реалазиции ключевых функции моего проекта, описание структуры базы данных, а также примеры интерфейса приложения.

1. BookingController.java – реализация этапов бронирования стола

@Controller

@RequestMapping("/booking")

public class BookingController {

private final UserRepository userRepository;

private final HallTableRepository hallTableRepository;

private final BookingRepository bookingRepository;

public BookingController(UserRepository userRepository,

HallTableRepository hallTableRepository,

BookingRepository bookingRepository) {

this.userRepository = userRepository;

this.hallTableRepository = hallTableRepository;

this.bookingRepository = bookingRepository;

}

/\*\*

\* Шаг 1: Выбор даты бронирования (с сегодняшнего дня)

\*/

@GetMapping("/date")

public String selectDate(Model model) {

LocalDate today = LocalDate.now();

model.addAttribute("today", today);

return "booking\_date";

}

/\*\*

\* Шаг 2: Выбор времени начала бронирования

\*/

@PostMapping("/time")

public String selectTime(@RequestParam String selectedDate, Model model) {

LocalDate date = LocalDate.parse(selectedDate);

LocalDateTime dayStart = date.atTime(12, 0);

LocalDateTime dayEnd = date.atTime(0, 0).plusDays(1);

List<LocalTime> timeSlots = new ArrayList<>();

for (LocalTime t = dayStart.toLocalTime(); !t.isAfter(dayEnd.toLocalTime()); t = t.plusHours(1)) {

timeSlots.add(t);

}

model.addAttribute("selectedDate", selectedDate);

model.addAttribute("timeSlots", timeSlots);

return "booking\_time";

}

/\*\*

\* Шаг 3: Выбор доступного стола

\*/

@PostMapping("/tables")

public String showTables(@RequestParam String selectedDate,

@RequestParam int startHour,

@RequestParam int endHour,

Model model) {

Authentication auth = SecurityContextHolder.getContext().getAuthentication();

String email = auth.getName();

var user = userRepository.findByEmail(email).orElse(null);

if (user == null) return "redirect:/login";

LocalDate date = LocalDate.parse(selectedDate);

LocalDateTime start = date.atTime(startHour, 0);

LocalDateTime end = date.atTime(endHour, 0);

List<HallTable> allTables = hallTableRepository.findAll();

List<HallTableData> availableTables = new ArrayList<>();

for (HallTable table : allTables) {

boolean isFree = true;

List<Booking> bookings = bookingRepository.findByTableId(table.getId());

for (Booking b : bookings) {

if (!b.isDeleted() && b.getStartTime().isBefore(end) && b.getEndTime().isAfter(start)) {

isFree = false;

break;

}

}

availableTables.add(new HallTableData(table, isFree));

}

model.addAttribute("availableTables", availableTables);

model.addAttribute("selectedDate", selectedDate);

model.addAttribute("startHour", startHour);

model.addAttribute("endHour", endHour);

return "booking\_tables";

}

}

Этот контроллер управляет процессом пошагового бронирования стола:

Пользователь выбирает дату.

Затем указывает время начала и окончания визита (максимум 3 часа).

Система проверяет, какие столы свободны в это время и предлагает выбрать один из них..

1. HookahOrderController.java – оформление заказа кальяна

@PostMapping("/hookah")

public String showHookahForm(@RequestParam String selectedDate,

@RequestParam int startHour,

@RequestParam int endHour,

@RequestParam Integer tableId,

Model model) {

model.addAttribute("selectedDate", selectedDate);

model.addAttribute("startHour", startHour);

model.addAttribute("endHour", endHour);

model.addAttribute("tableId", tableId);

return "booking\_hookah";

}

1. AdminController.java – административная панель

@GetMapping("/admin")

public String adminPanel(Model model) {

model.addAttribute("users", userRepository.findAll());

model.addAttribute("tables", hallTableRepository.findAll());

model.addAttribute("bookings", bookingRepository.findAllByOrderByStartTimeDesc());

model.addAttribute("flavors", flavorRepository.findAll());

model.addAttribute("additives", additiveRepository.findAll());

return "admin\_panel";

}

Администратор может просматривать список пользователей, столов, бронирований, вкусов и добавок. Это позволяет управлять системой без необходимости ручного изменения БД.

Репозиторий предоставляет методы для поиска и проверки существования пользователей по логину и email.

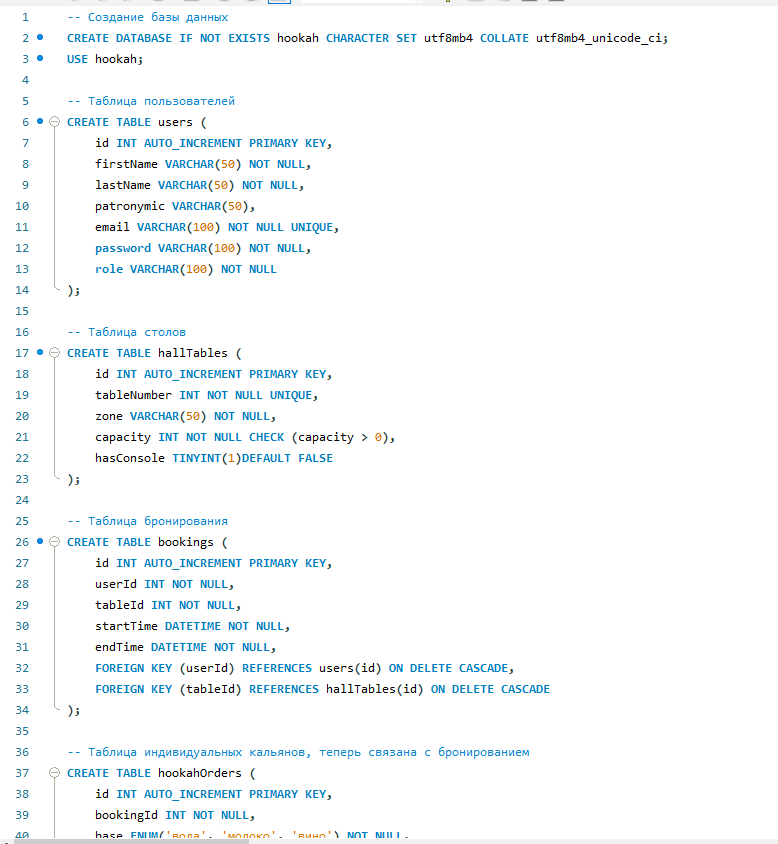


Рисунок 16 Листинг создания БД1

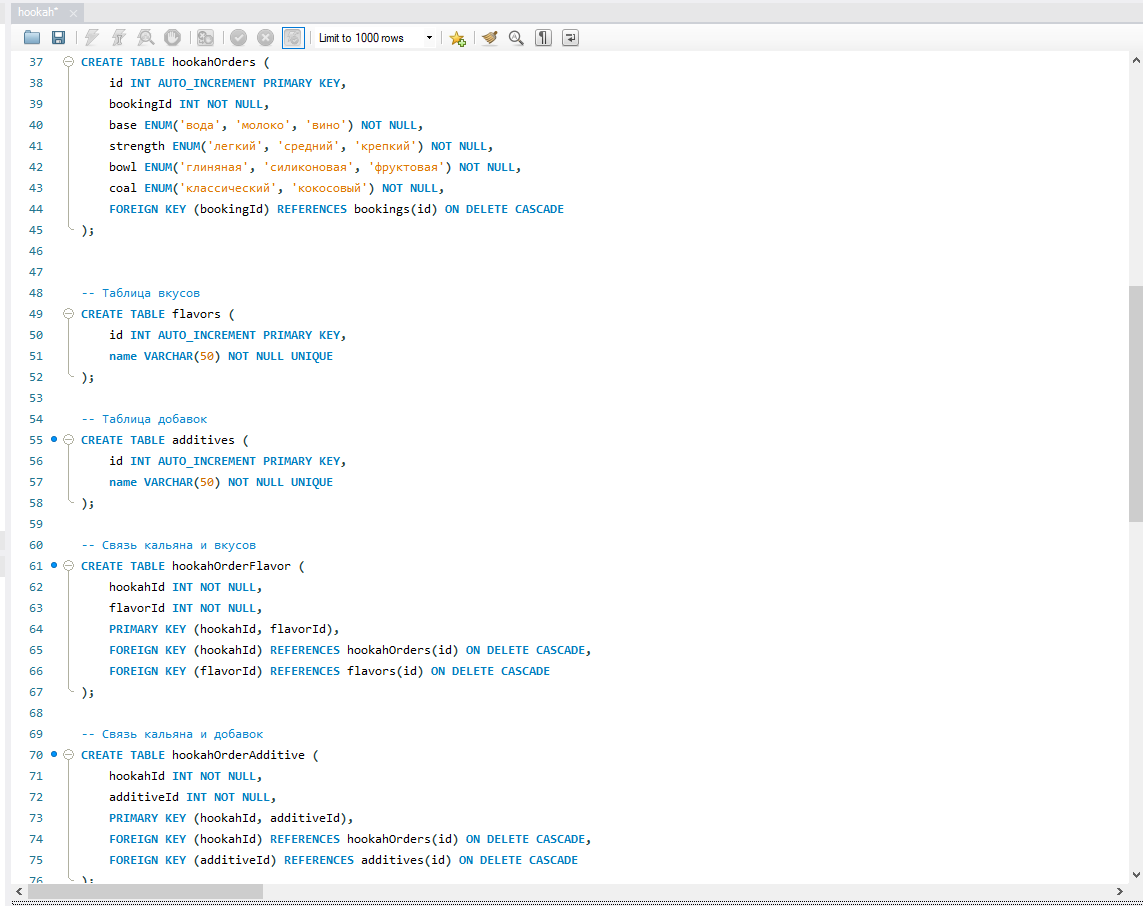


Рисунок 17 Листинг создания БД2

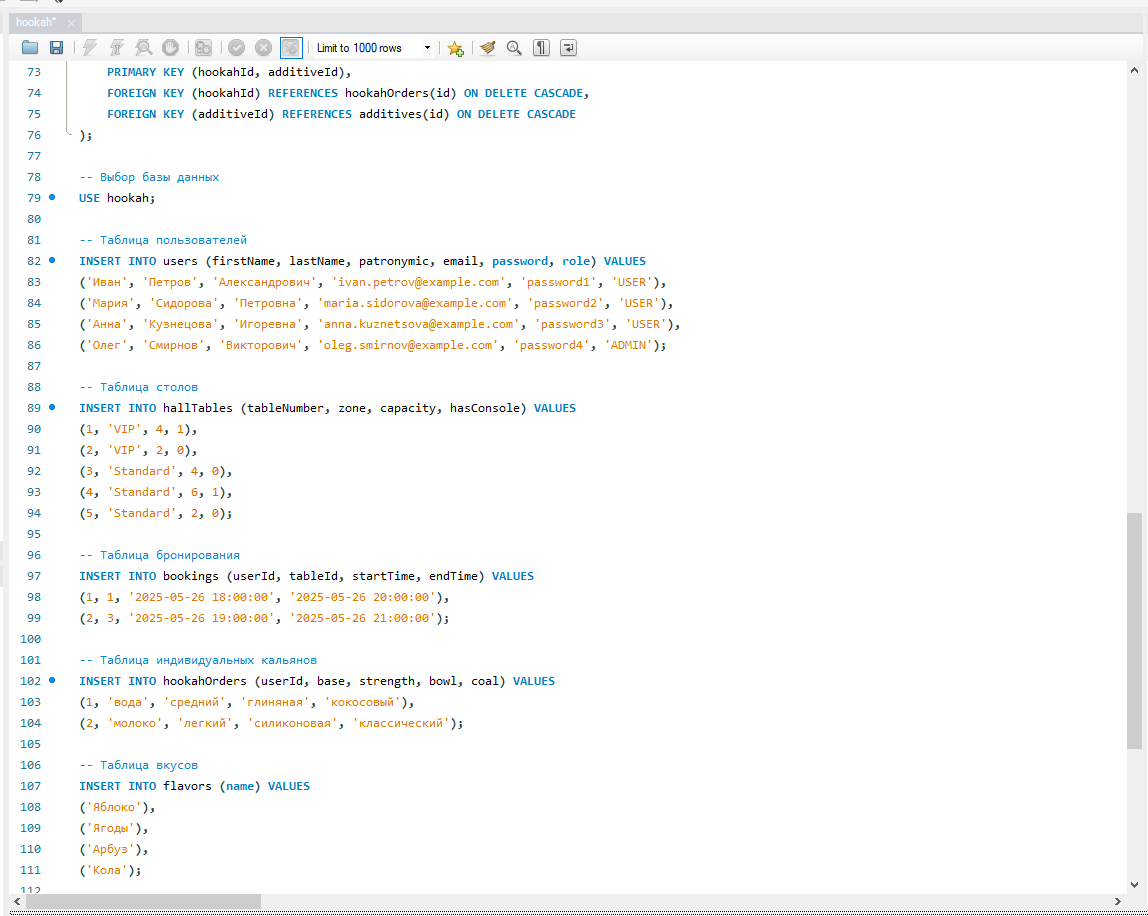


Рисунок 19 Листинг создания БД3

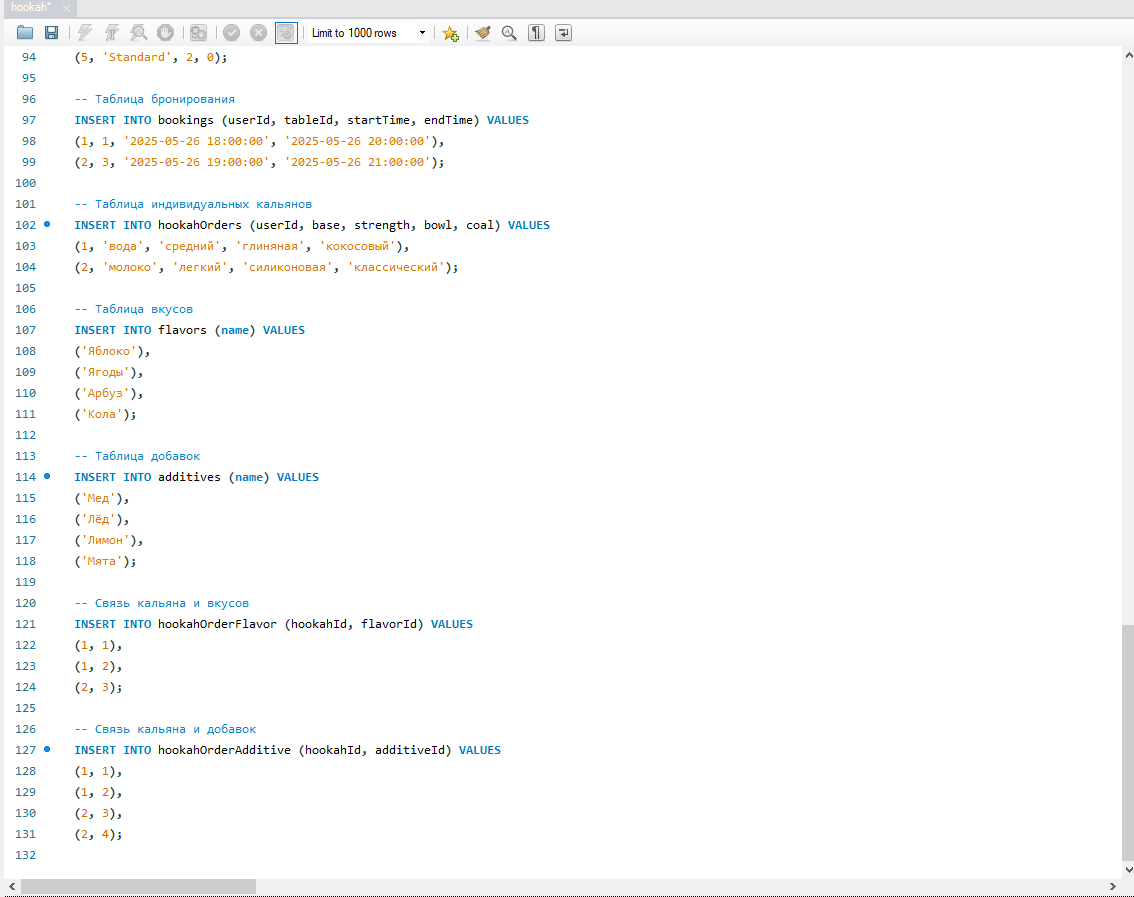


Рисунок 21 Листинг создания БД 4

Описание структуры создания БД

1. Подход к созданию БД

База данных проектируется на основе предметной области и классов-сущностей приложения. Каждая основная сущность (пользователь, бронирование, стол, заказ кальяна, вкус, добавка) соответствует отдельной таблице в базе данных. Связи между сущностями реализуются с помощью внешних ключей и вспомогательных таблиц для отношений многие-ко-многим.

2. Основные таблицы

users

Хранит данные пользователей: id, имя, фамилия, отчество, email, пароль, роль.

bookings

Хранит информацию о бронированиях: id, id пользователя, id стола, время начала и окончания, признак удаления.

halltables

Описывает столы в зале: id, номер стола, зона, вместимость, наличие приставки.

hookah\_orders

Заказы кальянов: id, база, крепость, чаша, уголь, признак удаления.

flavors

Вкусы для кальяна: id, название.

additives

Добавки для кальяна: id, название.

3. Связующие таблицы для отношений многие-ко-многим

booking\_hookah

Связывает бронирование и заказ кальяна (один заказ может входить в несколько бронирований и наоборот).

hookah\_order\_flavor

Связывает заказ кальяна и вкусы (один заказ — несколько вкусов).

hookah\_order\_additive

Связывает заказ кальяна и добавки (один заказ — несколько добавок).

4. Связи между таблицами

users (1) — (M) bookings

Один пользователь может иметь много бронирований.

bookings (M) — (1) halltables

Каждое бронирование связано с одним столом.

bookings (M) — (M) hookah\_orders

Через таблицу booking\_hookah.

hookah\_orders (M) — (M) flavors

Через таблицу hookah\_order\_flavor.

hookah\_orders (M) — (M) additives

Через таблицу hookah\_order\_additive.

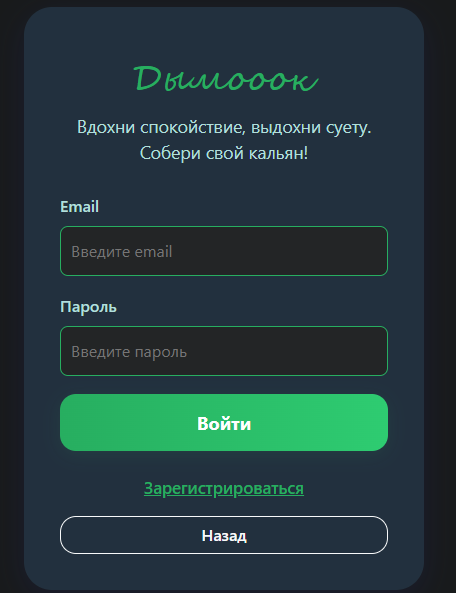


Рисунок 22 Скриншот авторизации экрана

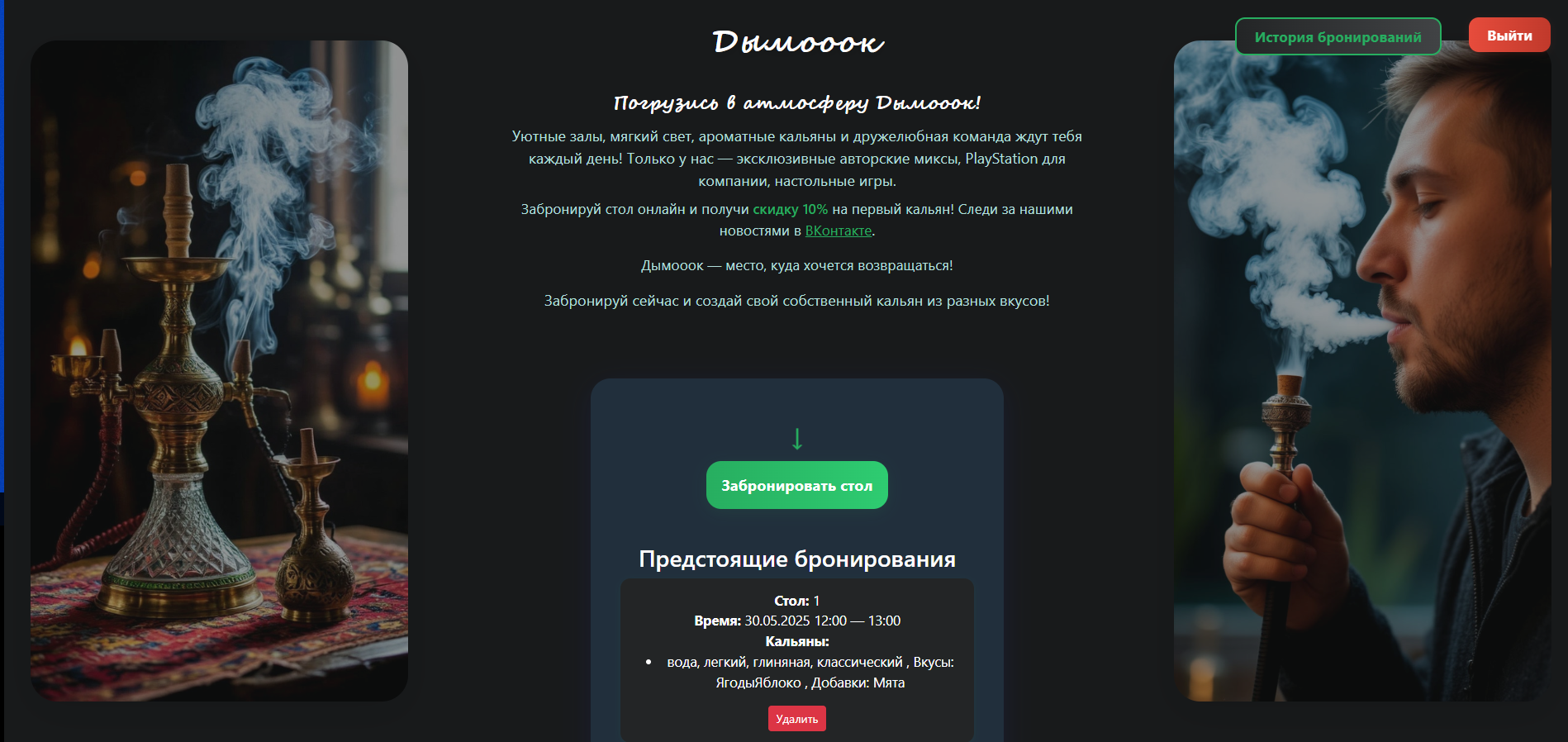


Рисунок 23 Скриншот главной страницы пользователя

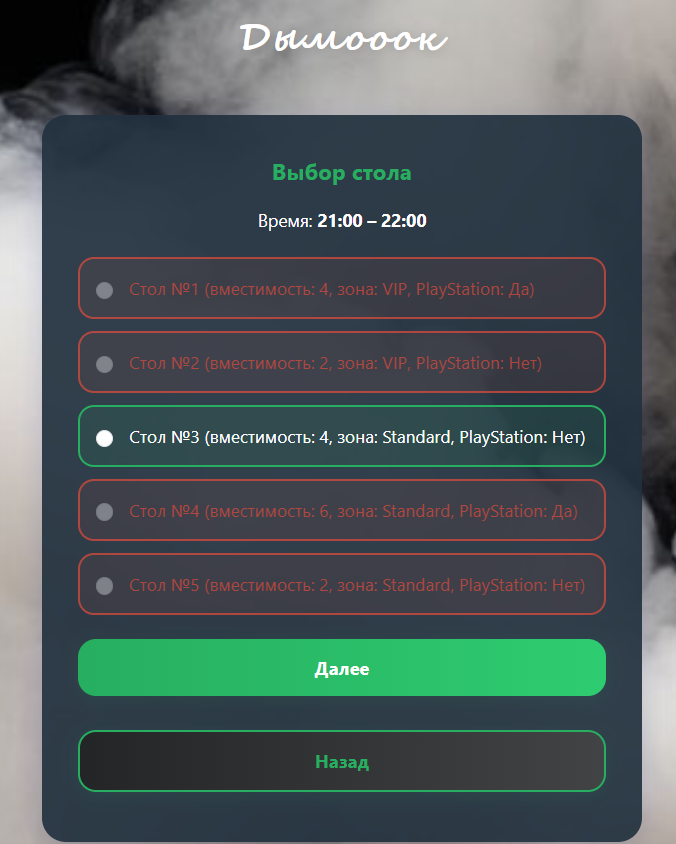


Рисунок 24 Скриншот выбора стола

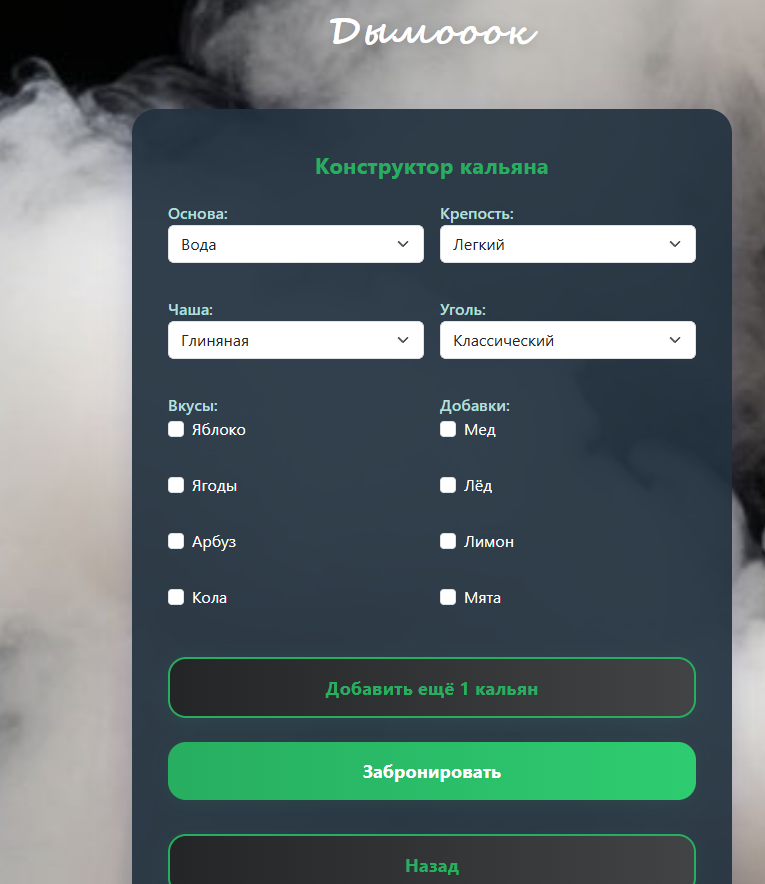


Рисунок 25 Скриншот конструктора кальяна

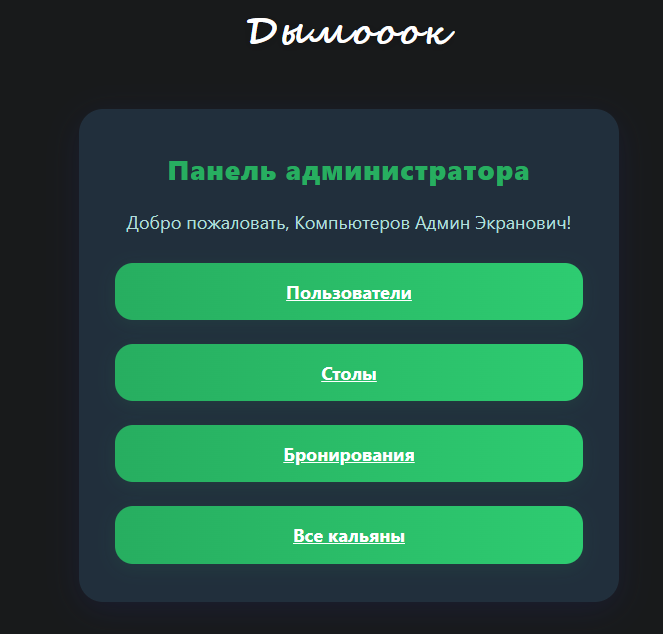


Рисунок 26 Скриншот главной страницы администратора

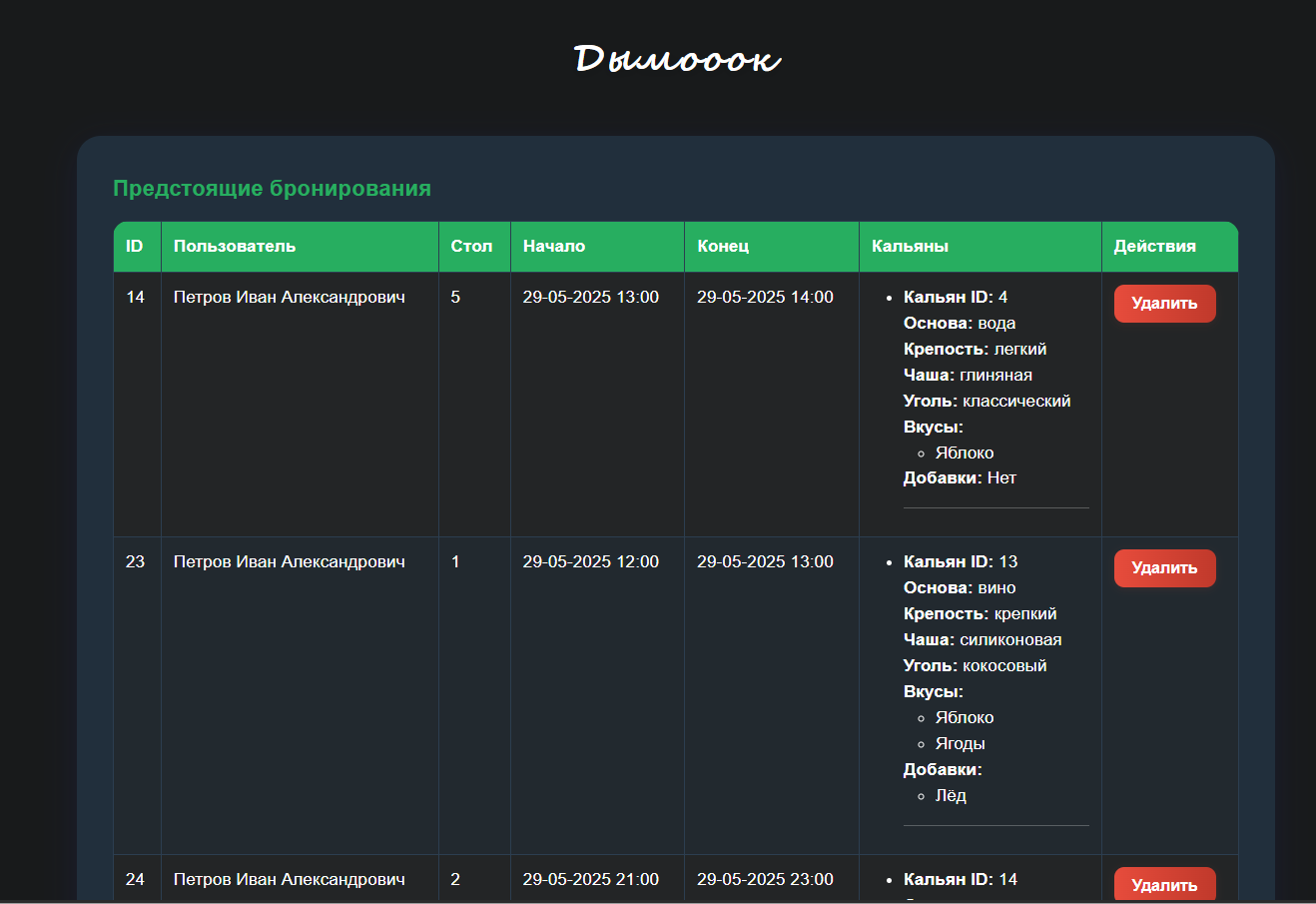


Рисунок 27 Скриншот бронирований у администратора