МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА NODEJS В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «Система бронирования услуг»

Разработчик:

студент группы 3ИСПР

Мишкин М.И.

(подпись)

Оценка защиты курсового проекта

Руководитель проекта:

преподаватель

Степанов С.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

« »

Дата защиты

« » 2025г.

Ногинск, 2025 г

СОДЕРЖАНИЕ КУРСОВОГО ПРОЕКТА

[ВВЕДЕНИЕ 3](#_Toc200319069)

[Цель курсовой работы: 3](#_Toc200319070)

[Задачи курсовой работы: 3](#_Toc200319071)

[1. Описание предметной области 4](#_Toc200319072)

[1.1 Краткая характеристика веб-приложения 4](#_Toc200319073)

[1.2 Описание программных инструментов и средств разработки 5](#_Toc200319074)

[1.2.1 Язык программирования «JavaScript» 5](#_Toc200319075)

[1.2.2 Фреймворк «Node.js» 5](#_Toc200319076)

[1.2.3 Серверная часть 6](#_Toc200319077)

[1.2.4 Среда разработки 7](#_Toc200319078)

[2. Практическая часть 9](#_Toc200319079)

[2.1 Проектирование веб-приложения 9](#_Toc200319080)

[2.2 Подготовка к работе 10](#_Toc200319081)

[2.3 Разработка веб-приложения 12](#_Toc200319082)

[2.4 Размещение веб-приложения на хостинге 26](#_Toc200319083)

[Заключение 28](#_Toc200319084)

[Список использованной литературы 29](#_Toc200319085)

[Приложение 30](#_Toc200319086)

# ВВЕДЕНИЕ

В последние годы информационные технологии стремительно проникают во все сферы жизни общества. Особенно актуальным становится использование веб-приложений в обслуживающем бизнесе — парикмахерских, салонах красоты, медицинских учреждениях, сервисных центрах и других предприятиях сферы услуг.

Данная курсовая работа посвящена разработке системы онлайн-бронирования услуг на базе фреймворка Node.js — одного из самых популярных и перспективных инструментов для серверной разработки. Проект реализуется в виде веб-приложения, ориентированного на сферу парикмахерских услуг, и предоставляет возможности как клиентам (для записи), так и администраторам и мастерам (для управления расписанием и контролем над процессами). Актуальность выбранной темы заключается в том, что традиционные способы записи по телефону или через администратора не всегда эффективны и могут привести к ошибкам, путанице в расписании и потере клиентов.

### Цель курсовой работы:

Разработка веб-приложения для онлайн-бронирования услуг в сфере парикмахерских с использованием фреймворка **Node.js**, обеспечивающего удобство использования, безопасность, масштабируемость и автоматизацию бизнес-процессов.

### Задачи курсовой работы:

1. Провести анализ предметной области и выявить ключевые функции системы бронирования;
2. Изучить программные технологии, применимые для разработки веб-приложений (Node.js, Express.js, MySQL, Bootstrap и др.);
3. Разработать архитектуру приложения, включающую клиентскую и серверную части;
4. Реализовать модули регистрации, аутентификации, бронирования, управления расписанием и панель администратора;
5. Обеспечить защиту пользовательских данных и контроль доступа на основе ролей;
6. Организовать хранение данных с использованием СУБД и ORM-инструментов;
7. Выполнить размещение готового веб-приложения на хостинге и провести его тестирование;
8. Подготовить техническую документацию и описание проекта.

# Описание предметной области

## 1.1 Краткая характеристика веб-приложения

Веб-приложение — это программный продукт, доступный пользователю через интернет-браузер, функционирующий по клиент-серверной архитектуре. Клиентская часть отображается в браузере пользователя и отвечает за взаимодействие с интерфейсом, а серверная часть обрабатывает данные, управляет логикой приложения и осуществляет работу с базой данных. Веб-приложения не требуют установки, что делает их доступными с любого устройства — компьютера, смартфона или планшета, имеющего доступ к сети интернет.

В отличие от настольных приложений, веб-приложения обладают рядом преимуществ:

1. Кроссплатформенность — одинаково работают на Windows, macOS, Linux, Android и iOS;
2. Обновления происходят на стороне сервера, не требуя участия пользователя;
3. Удобная масштабируемость и возможность использования облачных хранилищ;
4. Простота доступа и распространения среди пользователей.

Система бронирования услуг — это специализированный вид веб-приложения, предназначенный для автоматизации процесса записи клиентов на оказание различных услуг. Такие системы позволяют заменить ручную запись по телефону, вести электронный график специалистов, рассылать уведомления и предоставлять пользователю возможность выбора оптимального времени. Системы бронирования широко применяются в парикмахерских, салонах красоты, стоматологиях, медицинских центрах, спортивных залах, автомастерских и других сферах, где необходимо чёткое расписание и эффективная загрузка специалистов.

## 1.2 Описание программных инструментов и средств разработки

### 1.2.1 Язык программирования «JavaScript»

JavaScript — это мультипарадигменный язык программирования, изначально созданный для интерактивных элементов веб-страниц. Благодаря появлению Node.js, JavaScript стал полноценным инструментом для серверной разработки. Его популярность обусловлена следующими преимуществами:

Универсальность — один язык для фронтенда и бэкенда;

Поддержка большого количества библиотек (npm);

Простота входа для начинающих и высокая гибкость для опытных разработчиков; Асинхронная обработка событий — важная для систем с большим числом пользователей.

В рамках курсового проекта JavaScript используется во всех слоях архитектуры — от шаблонов интерфейса до обработки бизнес-логики на сервере и взаимодействия с базой данных.

### 1.2.2 Фреймворк «Node.js»

Node.js — это среда выполнения JavaScript на серверной стороне. Она построена на движке V8 от Google Chrome и позволяет обрабатывать тысячи запросов одновременно благодаря неблокирующему вводу-выводу. Node.js особенно хорош для:

Сетевых приложений и REST API;

Приложений реального времени;

Сервисов с большим количеством пользователей и взаимодействий.

В нашем проекте Node.js используется совместно с Express.js — минималистичным веб-фреймворком, который упрощает создание маршрутов, работу с HTTP-запросами, сессиями, куками и middleware.

Дополнительно используются:

Sequelize — ORM для MySQL;

Passport.js — библиотека для аутентификации;

dotenv — работа с переменными окружения;

### 1.2.3 Серверная часть

Серверная часть выполняет следующие задачи:

1. Обработка HTTP-запросов (GET, POST, PUT, DELETE);
2. Аутентификация и авторизация пользователей;
3. Работа с базой данных (CRUD-операции);
4. Защита маршрутов и валидация данных;
5. Хранение и загрузка статики (изображения, стили).
6. Архитектура сервера построена по принципу разделения на модули:
7. Маршруты (routes) — определяют URL-пути и их обработку;
8. Контроллеры (controllers) — логика обработки запросов;
9. Модели (models) — описывают структуру данных;
10. Middleware — промежуточные функции (например, защита маршрутов).

Для повышения отказоустойчивости и производительности также реализована логика обработки ошибок и подключение логгера.

### 1.2.4 Среда разработки

Для реализации проекта будет использоваться современная и удобная среда разработки, включающая в себя набор инструментов, обеспечивающих эффективную работу над веб-приложением. В качестве основного текстового редактора планируется использовать Visual Studio Code — мощную платформу с широким набором функций, удобным интерфейсом и поддержкой расширений. Благодаря встроенному терминалу, возможностям отладки кода и автодополнению, данный редактор обеспечит комфортную работу с языком JavaScript и средой Node.js.

Для управления версиями проекта будет применяться система контроля версий Git, которая позволит отслеживать изменения, работать с ветками и надёжно сохранять прогресс разработки. Репозиторий будет размещён на платформе GitHub, что обеспечит доступ к проекту из любой точки и упростит совместную работу и резервное копирование.

Установка и управление зависимостями будет осуществляться с помощью npm (Node Package Manager) — стандартного менеджера пакетов в экосистеме Node.js. Через него будут подключаться необходимые библиотеки, такие как Express, Sequelize, Passport.js, bcryptjs и другие.

Для тестирования API и проверки корректности маршрутов и взаимодействия клиент-сервер будет использоваться Postman — популярный инструмент для симуляции HTTP-запросов.

Работа с базой данных будет вестись через MySQL Workbench — графическую оболочку, предназначенную для управления базами данных MySQL. Она позволит визуализировать структуру данных, выполнять SQL-запросы и отслеживать изменения.

Для размещения веб-приложения на сервере планируется использовать хостинг Beget, предоставляющий доступ по протоколу SSH и инструменты для настройки серверного окружения. Установка зависимостей, настройка и запуск приложения будет происходить через командную строку с использованием PM2 — утилиты, обеспечивающей стабильную работу приложения и его автоматический перезапуск при сбоях.

Также в проекте будет применяться файл .env, содержащий переменные окружения: параметры подключения к базе данных, секретные ключи и другие конфиденциальные данные. Это обеспечит безопасность и гибкость при развёртывании проекта в различных средах (локально и на сервере).

Таким образом, выбранная среда разработки будет обладать всем необходимым функционалом для быстрой, надёжной и масштабируемой реализации веб-приложения, обеспечивая гибкость на всех этапах проектирования, разработки и внедрения.

# Практическая часть

## 2.1 Проектирование веб-приложения

Проектирование веб-приложения для бронирования парикмахерских услуг началось с анализа целевой аудитории и постановки ключевых задач, которые должно решать приложение. Основная идея заключалась в создании удобной и автоматизированной платформы, позволяющей пользователям записываться на услуги мастеров в режиме онлайн, а также обеспечивающей полноценное управление расписанием и бронированиями для администраторов и сотрудников. С самого начала предполагалось, что в системе будет три уровня пользователей: обычные пользователи, менеджеры (мастера или сотрудники), а также администраторы с расширенными правами.

В основе приложения лежит классическая клиент-серверная архитектура, где фронтенд взаимодействует с сервером через маршруты, а сервер, в свою очередь, обрабатывает данные и обращается к базе данных. Бэкенд написан на Node.js с использованием фреймворка Express, что позволило эффективно организовать маршруты, middleware и контроллеры. Визуальная часть сайта реализована на EJS-шаблонизации с применением SCSS и Bootstrap для стилизации. Это дало возможность создать адаптивный и современный интерфейс, ориентированный на интуитивное взаимодействие пользователя с системой.

На этапе проектирования были определены основные модули: система аутентификации, бронирование, управление слотами времени, система ролей и панели для разных типов пользователей. Пользователь после регистрации и входа получает доступ к личному кабинету, где он может просматривать историю своих записей, выбирать и бронировать свободные временные слоты, а также отменять бронь до наступления времени оказания услуги. Особое внимание было уделено логике работы слотов: приложение отслеживает, свободен ли нужный интервал времени.

Менеджерская часть включает панель с отображением всех записей пользователей, возможностью подтверждать или отклонять заявки. Неподтверждённые записи автоматически аннулируются через час, а освободившиеся слоты снова становятся доступными для других клиентов. Также предусмотрена функция закрытия слотов на определённые даты, что позволяет мастерам отмечать отпуск или нерабочее время. Админ-панель предоставляет доступ к управлению категориями, подкатегориями и услугами. Администратор может добавлять, редактировать перечень предоставляемых услуг и управлять пользователями.

Дополнительно была реализована система фильтрации и поиска, которая позволяет пользователям находить нужную услугу по дате, категории или ключевому слову. Это значительно облегчает навигацию по сайту при большом количестве доступных записей. Также встроена система комментариев и оценок, благодаря которой пользователи могут оставлять отзывы после посещения. Это мотивирует участников рынка повышать качество услуг и укрепляет доверие со стороны новых клиентов.

В ходе разработки большое внимание уделялось безопасности. Все пароли пользователей шифруются перед сохранением в базе данных, а доступ к чувствительным разделам (например, админ-панели) ограничивается в зависимости от роли пользователя. Маршруты защищены с помощью middleware, а вход в систему осуществляется через защищённую форму с валидацией.

Таким образом, архитектура и логика веб-приложения были изначально спроектированы с учётом реальных сценариев использования и требований к функционалу. Проект был построен модульно и масштабируемо, что в будущем позволит легко расширять его, добавлять новые функции или адаптировать под другие виды бронирования (например, студии, гостиницы или сервисы аренды).

## 2.2 Подготовка к работе

Подготовка к разработке веб-приложения включает в себя установку всех необходимых инструментов и фреймворков, которые обеспечат стабильную работу приложения и облегчат процесс разработки. Важным первым шагом является настройка серверного окружения. В нашем проекте для серверной части используется Node.js, который предоставляет среду для выполнения JavaScript-кода на сервере. На основе Node.js выбран фреймворк Express, который позволяет быстро создавать маршруты, обрабатывать запросы и управлять сессиями пользователей.

Для работы с базой данных был выбран Sequelize — ORM (Object-Relational Mapping) инструмент, который упрощает взаимодействие с реляционными базами данных. Sequelize позволяет легко создавать модели для различных сущностей, таких как пользователи, бронирования, категории и слоты времени. Это значительно ускоряет разработку и упрощает взаимодействие с базой данных.

Система аутентификации и авторизации в приложении реализована с помощью библиотеки Passport.js, которая поддерживает различные стратегии входа, включая работу с сессиями и JWT-токенами. Это позволяет безопасно хранить данные о пользователях и их сеансах.

Для фронтенда используется EJS (Embedded JavaScript) для динамической генерации HTML-страниц. Это даёт возможность удобно вставлять данные, полученные с сервера, в шаблоны и отображать их на стороне клиента. Внешний вид интерфейса и адаптивность обеспечиваются через Bootstrap и **SCSS**. Bootstrap помогает быстро создавать привлекательный и функциональный интерфейс, а SCSS позволяет работать с расширенными возможностями CSS для более гибкой стилизации.

В процессе подготовки важно также правильно настроить структуру проекта. Это включает создание соответствующих папок для моделей, маршрутов, шаблонов и стилей, чтобы проект оставался организованным и масштабируемым. С учетом всех установленных фреймворков и инструментов, подготовка к работе завершена, и проект готов к дальнейшему развитию, начиная с реализации базовых функций и интерфейса.

Таким образом, этап подготовки к работе является ключевым для успешной разработки веб-приложения. Это не только настройка всех инструментов и библиотек, но и создание структуры проекта, которая будет удобна для дальнейшего расширения и поддержки.

## 2.3 Разработка веб-приложения

Разработка веб-приложения для бронирования парикмахерских услуг — это сложный и многогранный процесс, который включает в себя несколько ключевых этапов. Каждый из них играет важную роль в создании качественного и эффективного продукта, который будет удобен пользователю и надежен в эксплуатации. Мы стремились обеспечить не только функциональность, но и безопасность данных, удобство использования и возможность масштабирования системы в будущем. Важно было продумать все аспекты разработки: от проектирования архитектуры и взаимодействия с базой данных до создания интуитивно понятного интерфейса для пользователей.

В этом проекте был использован стек технологий Node.js, Express и MySQL. Эта связка предоставила мощные инструменты для создания высокопроизводительного и гибкого веб-приложения. Приложение предназначено для двух типов пользователей: клиентов, которые хотят забронировать услуги парикмахера, и парикмахеров, которые управляют своим расписанием и предлагают свои услуги. Использование этих технологий обеспечило стабильную работу системы при высокой нагрузке и позволило легко адаптировать приложение под расширение функционала.

2.3.1 Архитектура приложения и базовые сущности

На начальном этапе создания веб-приложения особенно важно не только продумать его функциональность, но и тщательно спроектировать внутреннюю организацию файлов и каталогов. Чёткая и логичная структура проекта облегчает его разработку, поддержку, тестирование, а также обучение новых участников команды. При построении архитектуры проекта "barber-booking" мы руководствовались принципами модульности, читаемости и расширяемости.

Изначально был создан корневой каталог проекта "barber-booking/", в котором были размещены все ключевые компоненты. Внутри проекта мы выделили следующие основные директории и файлы (Рис. 1):

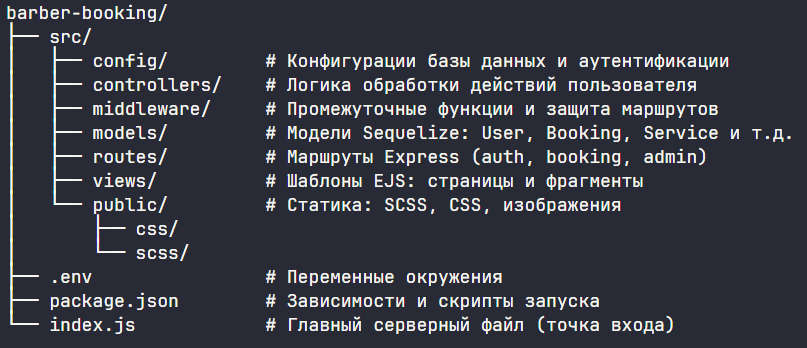


Рисунок 1. Структура проекта.

* src**/** — основной каталог, содержащий исходный код приложения. Все рабочие файлы проекта сосредоточены именно здесь.
* config**/** — директория для хранения конфигурационных файлов, в том числе настроек подключения к базе данных (database.js) и параметров аутентификации через Passport (passport.js).
* controllers**/** — содержит модули с логикой обработки пользовательских действий. Например, контроллер регистрации, входа, бронирования и отзывов.
* middleware**/** — предназначена для размещения промежуточных обработчиков запросов, например, функций авторизации или проверки ролей пользователей.
* models**/** — включает модели базы данных, созданные с помощью ORM Sequelize. Здесь описываются все сущности: пользователи, услуги, бронирования, отзывы и т. д.
* routes**/** — файлы маршрутов, которые принимают HTTP-запросы и направляют их к соответствующим контроллерам.
* views**/** — шаблоны HTML-страниц, написанные с использованием EJS (Embedded JavaScript Templates). Подкаталоги pages/ и partials/ позволяют удобно структурировать представления.
* public**/** — содержит статические ресурсы сайта: стили CSS, препроцессоры SCSS, изображения, иконки и сторонние библиотеки. Подразделение ресурсов помогает разделить визуальное оформление от логики приложения.

Центральным элементом архитектуры является файл src/index.js (Рис. 2), выполняющий роль точки входа в приложение. Именно здесь осуществляется начальная конфигурация, подключение модулей и запуск сервера. В этом файле происходит настройка Express, сессий, подключение базы данных и маршрутов. Пример кода:

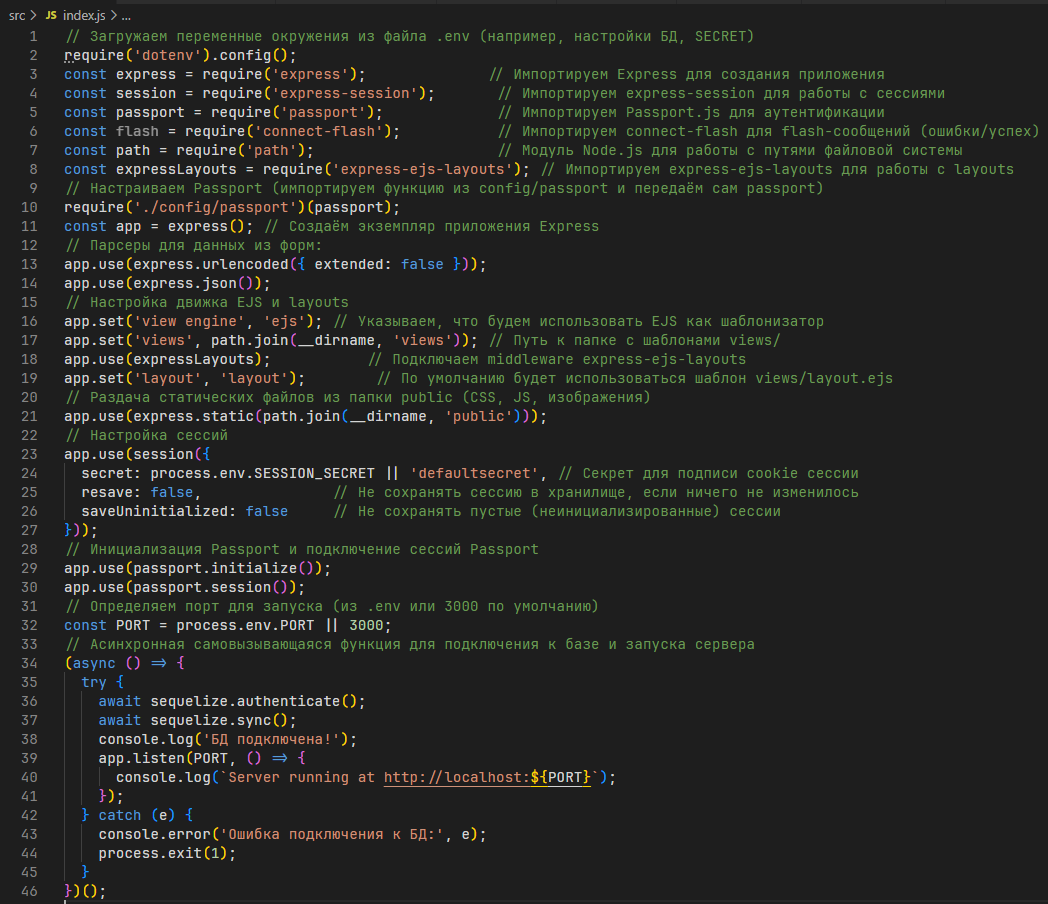


Рисунок 2. Настройка проекта.

В этом коде мы подключаем Express и настраиваем его: задаём парсинг форм, подключаем шаблонизатор EJS, указываем папку для статических файлов. Далее устанавливаем сессии с помощью express-session и активируем Passport для аутентификации пользователей.

Таким образом, структура проекта была спроектирована с прицелом на расширяемость, удобство поддержки и соблюдение принципов чистой архитектуры. Такая организация каталогов и файлов позволяет разработчику легко ориентироваться в коде, быстрее вносить изменения и эффективно масштабировать систему в будущем.

2.3.2 Создание и настройка базы данных

После завершения проектирования структуры проекта следующим важным этапом стала реализация базы данных. База данных играет ключевую роль в любом динамическом веб-приложении: именно она обеспечивает хранение, обработку и целостность информации о пользователях, услугах, бронированиях и отзывах. Мы подошли к этому этапу с особым вниманием к структуре и связям между сущностями.

В качестве системы управления базами данных (СУБД) была выбрана MySQL — широко распространённая реляционная СУБД, подходящая как для разработки, так и для эксплуатации в продакшен-среде. Для работы с MySQL на стороне Node.js мы использовали ORM Sequelize, которая предоставляет удобный и безопасный способ взаимодействия с базой через объектно-ориентированные модели.

Подключение к базе данных осуществляется через файл config/database.js (Рис. 3).

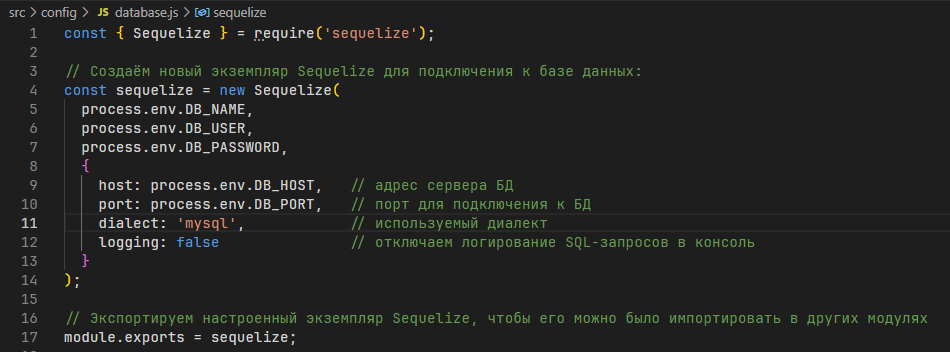


Рисунок 3. Подключение к БД с параметрами из .env.

В данном коде создаётся экземпляр Sequelize с параметрами, получаемыми из переменных окружения. Это повышает безопасность и позволяет гибко управлять настройками без изменения исходного кода. Метод authenticate() проверяет корректность соединения.

Далее, после успешного подключения, мы приступили к созданию моделей. Каждая модель представляет собой таблицу в базе данных, а поля модели соответствуют столбцам таблицы. Например, для хранения информации о клиентах, администраторах и мастерах была реализована модель User, содержащая имя, электронную почту, пароль и роль пользователя.

Кроме User, были созданы и другие модели:

* Service — представляет услугу, которую можно забронировать. Содержит название, описание, стоимость и категорию.
* Booking — хранит информацию о бронировании: дату, время, выбранного мастера и клиента.
* TimeSlot — определяет возможные временные интервалы для записи.
* Review — содержит отзывы клиентов на оказанные услуги.
* Category**/**Subcategory — группируют услуги по типам (например, "стрижка", "окрашивание").

Между моделями устанавливались связи типа "один ко многим" и "многие ко многим". Например, один пользователь может иметь множество бронирований, а одно бронирование относится к конкретной услуге и временному слоту. Sequelize позволяет описать эти связи декларативно, на уровне моделей.

Для синхронизации моделей с базой данных использовался метод sequelize.sync({ alter: true }). Он автоматически создаёт таблицы и применяет изменения в их структуре, если они были внесены в модель.

Дополнительно для заполнения базы данных начальными значениями был реализован файл seed.js, в котором описан скрипт генерации категорий, подкатегорий, услуг и тайм-слотов. Скрипт удаляет существующие таблицы, пересоздаёт их, и далее наполняет тестовыми данными (Рис. 4). В частности, создаются категории услуг (например, "Стрижки"), каждая из которых включает несколько подкатегорий и набор типовых услуг с описанием, ценой и продолжительностью.

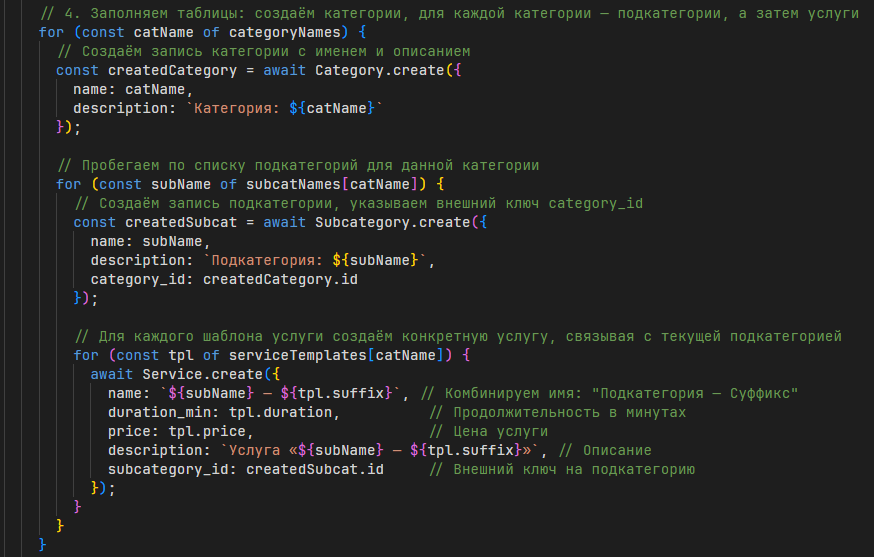


Рисунок 4. Работа seeder’a.

Кроме того, отдельно реализована генерация временных слотов (TimeSlot) — диапазон дат от месяца назад до месяца вперёд, с шагом 30 минут, только по будним дням с 10:00 до 18:00. Это позволяет имитировать реальное рабочее расписание салона.

Таким образом, с помощью ORM и специально созданного генератора (сидера), мы обеспечили как корректную архитектуру базы данных, так и её первичное наполнение тестовыми данными. Это облегчило начальную отладку, демонстрацию возможностей платформы и последующее масштабирование.

2.3.3 Руководство программиста: маршруты, логика, страницы

После настройки базы данных и моделей наступает следующий ключевой этап разработки — реализация логики взаимодействия между пользователем, сервером и базой данных. Этот этап включает в себя создание HTTP-маршрутов, подключение соответствующих контроллеров, обработку запросов и отображение результатов через шаблоны EJS. Каждый маршрут является связующим звеном между действиями пользователя и функциональностью веб-приложения.

Маршруты проекта структурированы по принципу ответственности: пользовательские маршруты обрабатываются в routes/index.js, маршруты авторизации — в routes/auth.js, а административные и менеджерские — в routes/admin.js и routes/manager.js соответственно. Такая организация позволяет обеспечить модульность и масштабируемость проекта.

Рассмотрим поэтапно реализацию ключевых маршрутов, начиная с главной страницы сайта.

Главная страница:

На главной странице отображаются все категории и связанные с ними подкатегории, в которых, в свою очередь, содержатся услуги (Рис. 5).

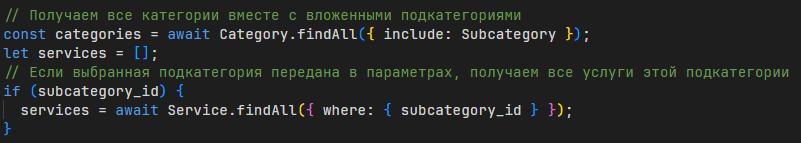


Рисунок 5. Обработка категорий, подкатегорий и услуг.

В данном коде осуществляется асинхронный запрос к базе данных с использованием метода findAll(). Метод получает все категории и по связям (через include) — соответствующие подкатегории. Полученные данные передаются в шаблон home.ejs, где визуально отображаются как список доступных направлений услуг.

Регистрация и вход пользователя:

Регистрация и аутентификация пользователя организована при помощи библиотеки passport. В маршрутах /register и /login реализованы как GET-запросы для отображения форм, так и POST-запросы для обработки введённых данных (Рис. 6).

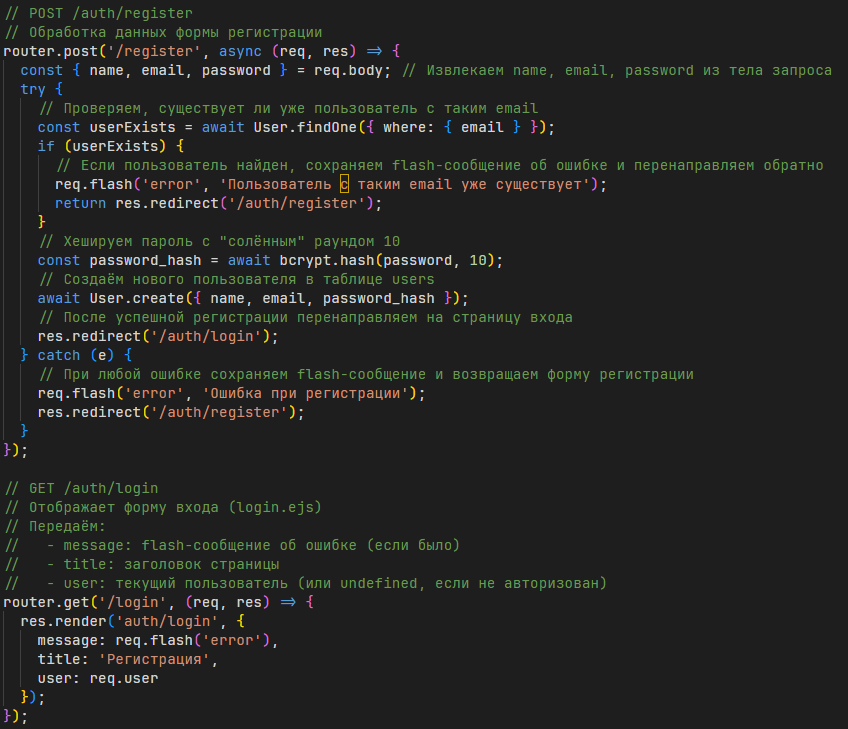


Рисунок 6. Роуты для отображения формы и для обработки введенных данных.

Контроллер регистрации выполняет проверку, существует ли пользователь с таким e-mail, и в случае его отсутствия хеширует пароль с использованием bcrypt, после чего создаёт нового пользователя. Авторизация производится с применением сессий express-session, в рамках которых в объект req.user сохраняется информация о текущем пользователе.

Личный кабинет клиента (/dashboard):

После успешной авторизации пользователь перенаправляется в личный кабинет, где отображаются все его активные и прошедшие бронирования (Рис. 7)

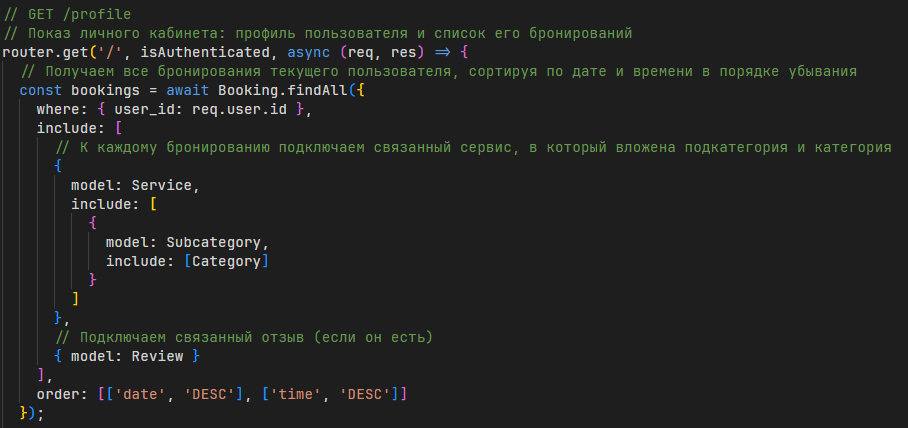


Рисунок 7. Роут профиля пользователя.

Middleware isAuthenticated проверяет, авторизован ли пользователь. Далее выполняется запрос всех бронирований пользователя с подгрузкой связанных моделей — Service и TimeSlot. В шаблоне dashboard.ejs эти данные отображаются в виде таблицы с возможностью отмены визита.

Метод findByPk() находит услугу по ID, полученному из параметра запроса. Связанные подкатегория и категория загружаются через include, а также получаются только те слоты, которые ещё доступны. Это обеспечивает фильтрацию по актуальности.

Бронирование слота времени:

После выбора времени пользователь может забронировать слот (Рис. 8).

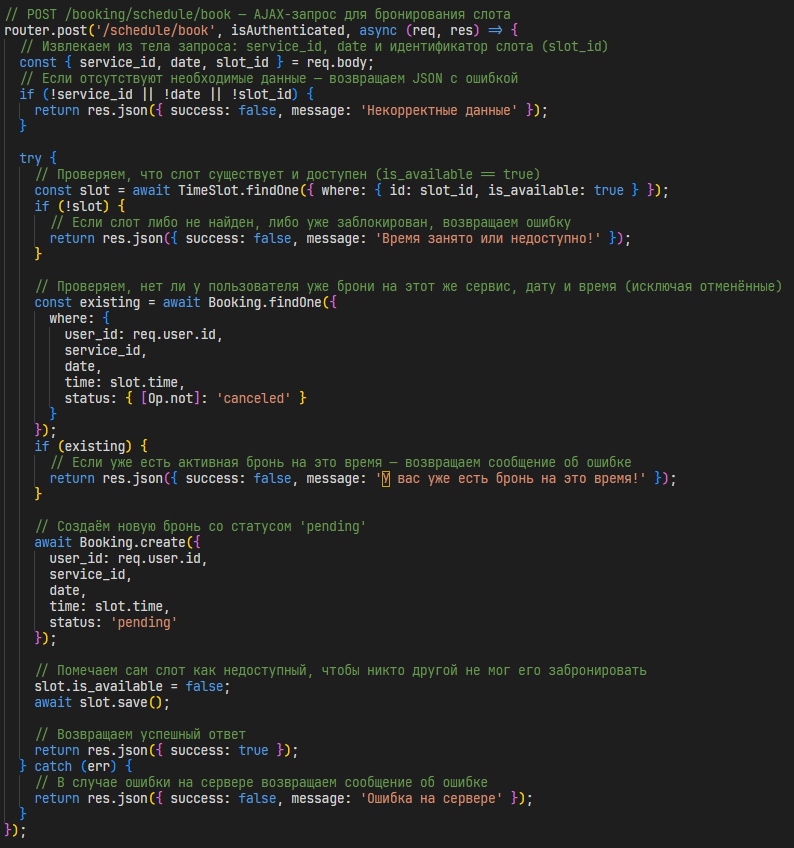


Рисунок 8. Обработка бронирования слота с условиями.

Здесь происходит три ключевых действия: создание записи в таблице Booking, изменение состояния слота на недоступный и перенаправление в личный кабинет. Этот механизм предотвращает повторное бронирование одного и того же времени.

Отзывы пользователей:

Функциональность отзывов добавляет социальной составляющей проекту.

Каждый отзыв содержит идентификаторы пользователя и услуги, а также текст и рейтинг. Далее он отображается в карточке услуги (Рис. 9).

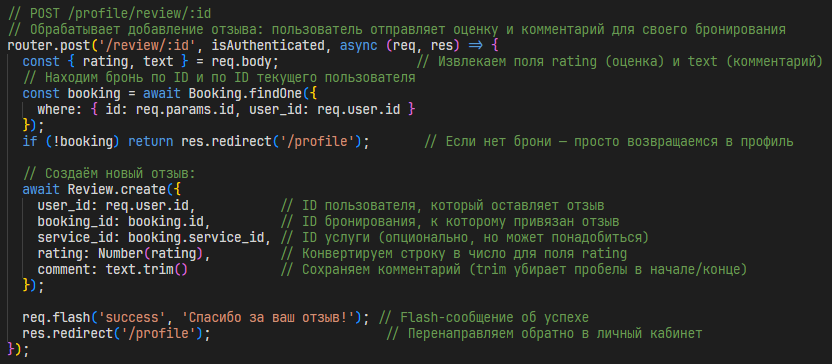


Рисунок 9. Реализация функции добавления отзыва и комментария.

Админ-панель:

Пользователи с ролью admin получают доступ к панели администратора (Рис. 10). Здесь они могут управлять категориями, услугами, подкатегориями и временными слотами.

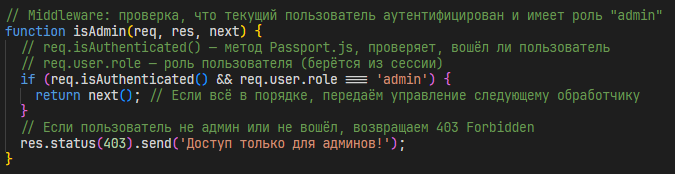


Рисунок 10. Защита от несанкционированного доступа.

Middleware isAdmin ограничивает доступ к этому маршруту, обеспечивая защиту административного интерфейса от обычных пользователей. В шаблоне реализован интерфейс управления с CRUD-операциями над сущностями платформы.

Панель менеджера:

Менеджер имеет доступ к более узкому интерфейсу, сосредоточенному на операционном управлении бронированиями (Рис. 11).

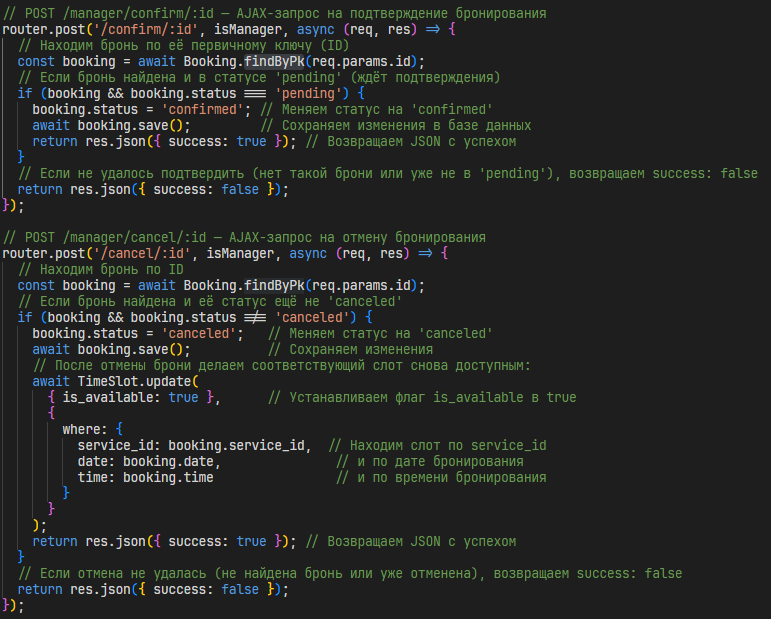


Рисунок 11. Маршрут для подтверждения/отмены бронирования.

В шаблоне менеджера предусмотрено табличное отображение записей с возможностью подтверждения, отмены или добавления комментария. Реализация таких интерфейсов способствует эффективной работе с клиентами.

Шаблоны EJS:

Визуальная составляющая веб-приложения реализована при помощи шаблонов EJS — Embedded JavaScript. Это серверный шаблонизатор, позволяющий внедрять динамические данные в HTML-страницы. Каждый шаблон был создан в директории views/pages/, что обеспечивает логическую изоляцию и структурированность компонентов пользовательского интерфейса. Разработка шаблонов велась параллельно с реализацией маршрутов, что позволило быстро проверять корректность отображения данных и взаимодействия элементов.

Все шаблоны наследуют общий макет layout.ejs, в котором определены основные элементы: навигационное меню, хедер, футер, подключение стилей и глобальных скриптов. Благодаря такому подходу обеспечивается единообразие интерфейса и повторное использование элементов без дублирования кода. Макет также содержит условную отрисовку блоков в зависимости от статуса пользователя (гость, клиент, менеджер или администратор), что позволяет адаптировать интерфейс под каждую роль.

Ниже представлено описание ключевых шаблонов, реализованных в рамках проекта:

* schedule.ejs — стартовая страница, на которой пользователь впервые знакомится с платформой. Здесь отображаются карточки с категориями парикмахерских услуг, каждая из которых включает иконку, название, краткое описание и список подкатегорий. Подкатегории кликабельны и ведут к соответствующим страницам с конкретными услугами. Если категорий много, предусмотрен механизм пагинации или динамической подгрузки (Рис. 12).

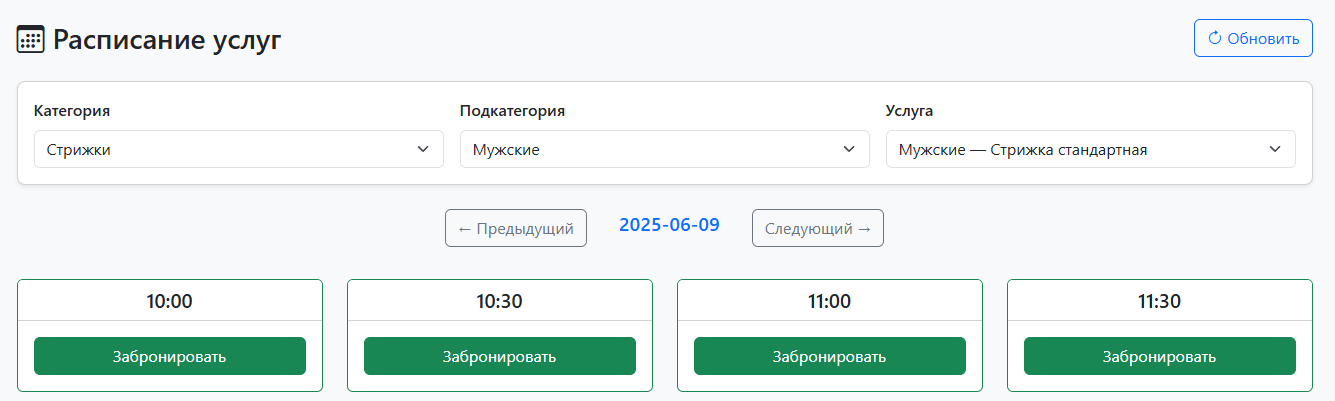


Рисунок 12. Главная страница с слотами для брони.

* login.ejs и register.ejs — шаблоны для аутентификации и регистрации. Страницы содержат формы с полями для ввода email, пароля, подтверждения пароля и имени. Также реализована система сообщений об ошибках (например, «email уже используется» или «пароли не совпадают») и всплывающие уведомления об успехе. При помощи классов Bootstrap и SCSS обеспечена адаптивная верстка и современный внешний вид форм.
* dashboard.ejs — это личный кабинет клиента. В центральной части экрана размещена таблица со всеми бронированиями, выполненными данным пользователем. Каждая строка включает информацию о названии услуги, дате, времени, статусе и доступных действиях (например, отменить запись или оставить отзыв). Также присутствует блок приветствия, где отображается имя пользователя и краткая статистика: количество записей, оставленных отзывов и т.п. Дополнительно присутствует кнопка выхода из аккаунта.
* service.ejs — шаблон страницы конкретной услуги. Здесь детально описаны услуги: название, цена, длительность, описание, категория и подкатегория.
* admin.ejs — шаблон административной панели. Он включает панели управления категориями, услугами и пользователями. Категории и услуги отображаются в виде таблиц с возможностью редактирования и удаления, а также кнопками для создания новых записей. Для каждого пользователя в таблице отображается имя, email, роль и возможность изменить её или заблокировать пользователя. Панель снабжена всплывающими окнами (modal), в которых можно ввести новые значения без перезагрузки страницы (Рис. 13).

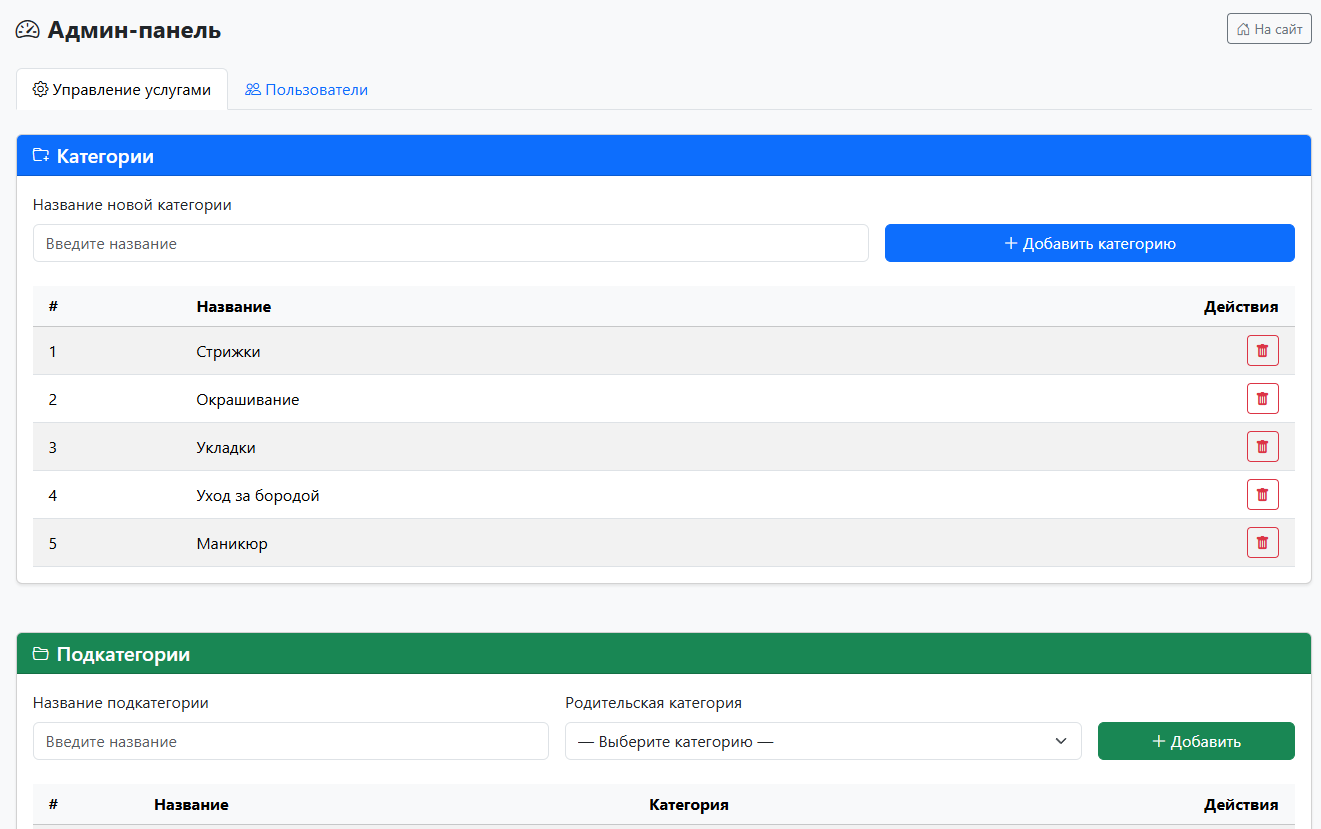


Рисунок 13. Админ-панель.

* manager.ejs — интерфейс для менеджеров, фокусирующийся на бронированиях и отзывах. На странице представлен фильтр по дате, статусу и услуге. Каждое бронирование оформлено в виде строки с колонками: пользователь, услуга, дата, время, статус и доступные действия (например, подтвердить или отклонить запись, добавить внутренний комментарий). Также реализована визуальная индикация новых записей и отметка уже обработанных (Рис. 14).

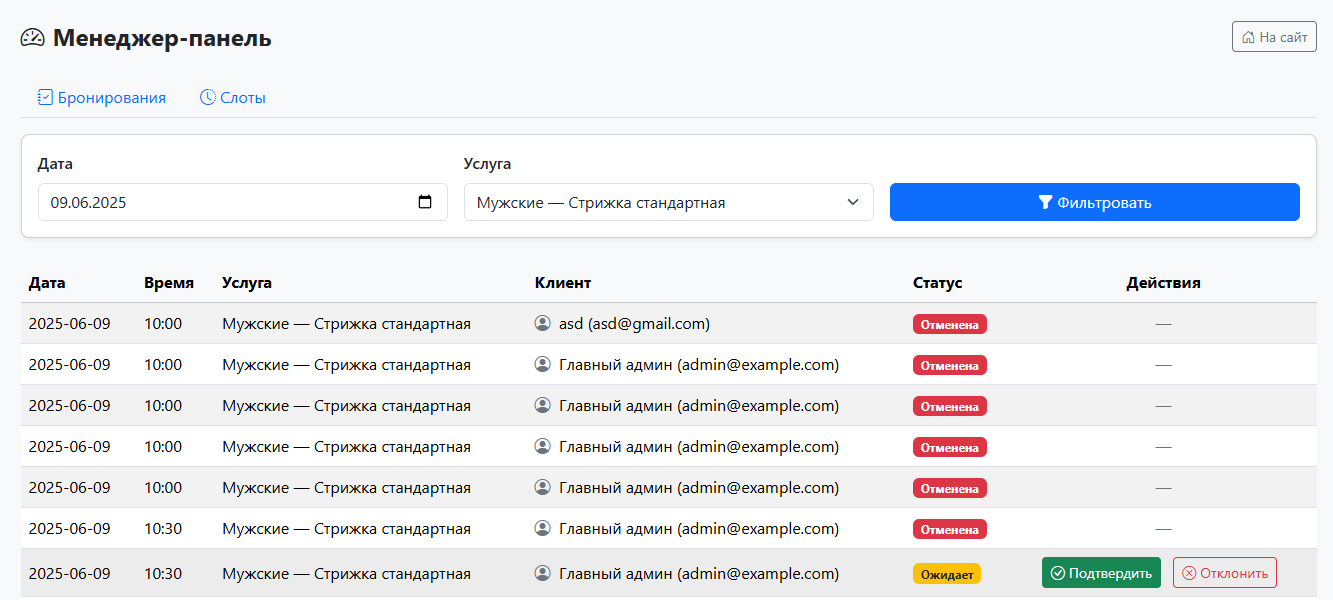


Рисунок 14. Менеджер-панель.

Таким образом, каждая страница в проекте обладает чёткой структурой, понятным интерфейсом и адаптирована под конкретный сценарий использования. Использование EJS в связке с SCSS позволило добиться визуальной целостности, а логическое разделение шаблонов сделало возможной быструю доработку и расширение интерфейса в будущем.

Ручное тестирование функциональности:

На финальном этапе разработки было проведено ручное тестирование маршрутов. Были протестированы следующие сценарии:

* Успешная и неуспешная регистрация (ввод пустых и дублирующих e-mail).
* Вход пользователя и корректность отображения данных в dashboard.
* Бронирование услуги и автоматическое скрытие занятого слота.
* Отображение отзывов, добавление новых оценок.
* Работа панели администратора: удаление, редактирование, добавление сущностей.
* Работа панели менеджера: проверка, изменение и фильтрация бронирований.
* Попытка несанкционированного доступа к /admin и /manager и блокировка через middleware.

В ходе тестирования фиксировались замечания по удобству использования и точности отображения данных. После итеративных правок приложение достигло стабильного состояния, способного обслуживать целевой поток пользователей и удовлетворять требованиям бизнес-логики проекта.

## 2.4 Размещение веб-приложения на хостинге

Для размещения веб-приложения на хостинге был выбран сервис Beget, который предоставляет удобные инструменты для работы с сервером через SSH и настройку веб-сервера. Этот хостинг был выбран благодаря своей надежности и простоте использования, а также наличию всех необходимых функций для запуска веб-приложений.

Процесс развертывания сайта на Beget включал несколько ключевых шагов. Сначала было выполнено подключение к серверу через SSH, что позволило получить доступ к командной строке и установить необходимые компоненты для работы приложения, такие как Node.js, npm и другие зависимости, указанные в package.json. Далее были загружены файлы приложения на сервер, используя SCP или FTP. После этого настроена система PM2 для управления процессом Node.js, что обеспечило его автоматический перезапуск при сбоях и облегчило мониторинг.

Основной задачей при настройке веб-сервера Nginx было обеспечить правильную маршрутизацию запросов. Nginx выполняет роль обратного прокси-сервера, который принимает HTTP-запросы и перенаправляет их на приложение Node.js. В конфигурационном файле Nginx были прописаны параметры, которые обеспечивают правильную работу статических файлов (например, CSS, изображения и JavaScript) и гарантируют их быструю доставку пользователям. Также был настроен проксинг запросов на Node.js-сервер, работающий на определённом порту.

Кроме того, на сервере были настроены переменные окружения в .env файле для безопасного хранения конфиденциальных данных, таких как параметры подключения к базе данных, ключи API и прочее. Также были предприняты меры по обеспечению безопасности, такие как настройка HTTPS с использованием SSL-сертификата для шифрования данных между сервером и клиентом.

Таким образом, благодаря использованию Beget, SSH-подключению и настройке Nginx, веб-приложение было успешно размещено на сервере с высокой доступностью и безопасностью.

# Заключение

Курсовой проект по разработке системы бронирования услуг на базе Node.js позволил применить на практике полученные знания в области веб-разработки, системного проектирования и баз данных. Были решены следующие задачи:

* Изучена предметная область автоматизации записи на услуги;
* Выбраны современные технологии (JavaScript, Node.js, Express, MySQL);
* Разработана архитектура и реализован серверный функционал;
* Создано удобное и адаптивное пользовательское представление;
* Обеспечена защита данных и безопасность системы;
* Выполнено размещение на хостинге.

Полученное приложение является масштабируемым и может быть дополнено интеграцией с платёжными сервисами, Telegram-ботами, push-уведомлениями и другими функциями. Разработка подобного рода систем — важный шаг на пути становления как начинающего, так и профессионального разработчика.

# Список использованной литературы

1. Грибков, А. А. MySQL. Основы работы с базами данных / А. А. Грибков. — Москва : БХВ-Петербург, 2021. — 352 с. [Печатный источник].
2. Бэнкс, А. Node.js. Профессиональная разработка серверных приложений / А. Бэнкс, Э. Порселл. — Москва : Диалектика, 2021. — 448 с. [Печатный источник].
3. Флэнаган, Д. JavaScript. Подробное руководство / Д. Флэнаган. — Санкт-Петербург : Питер, 2022. — 1088 с. [Печатный источник].
4. Sequelize documentation [Электронный ресурс]. — URL: https://sequelize.org/docs/ (дата обращения: 16.05.2025).
5. MySQL documentation [Электронный ресурс]. — URL: <https://dev.mysql.com/doc/> (дата обращения: 16.05.2025).
6. Visual Studio Code documentation [Электронный ресурс]. — URL: <https://code.visualstudio.com/docs/> (дата обращения: 10.05.2025).
7. Bootstrap Documentation [Электронный ресурс].– URL: https://getbootstrap.com/docs/5.3/getting-started/introduction (Дата обращения: 17.05.2025).

# приложение

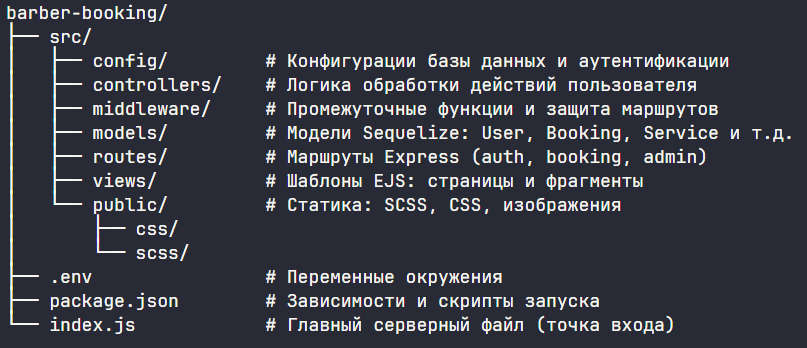


Рисунок 1. Структура проекта.

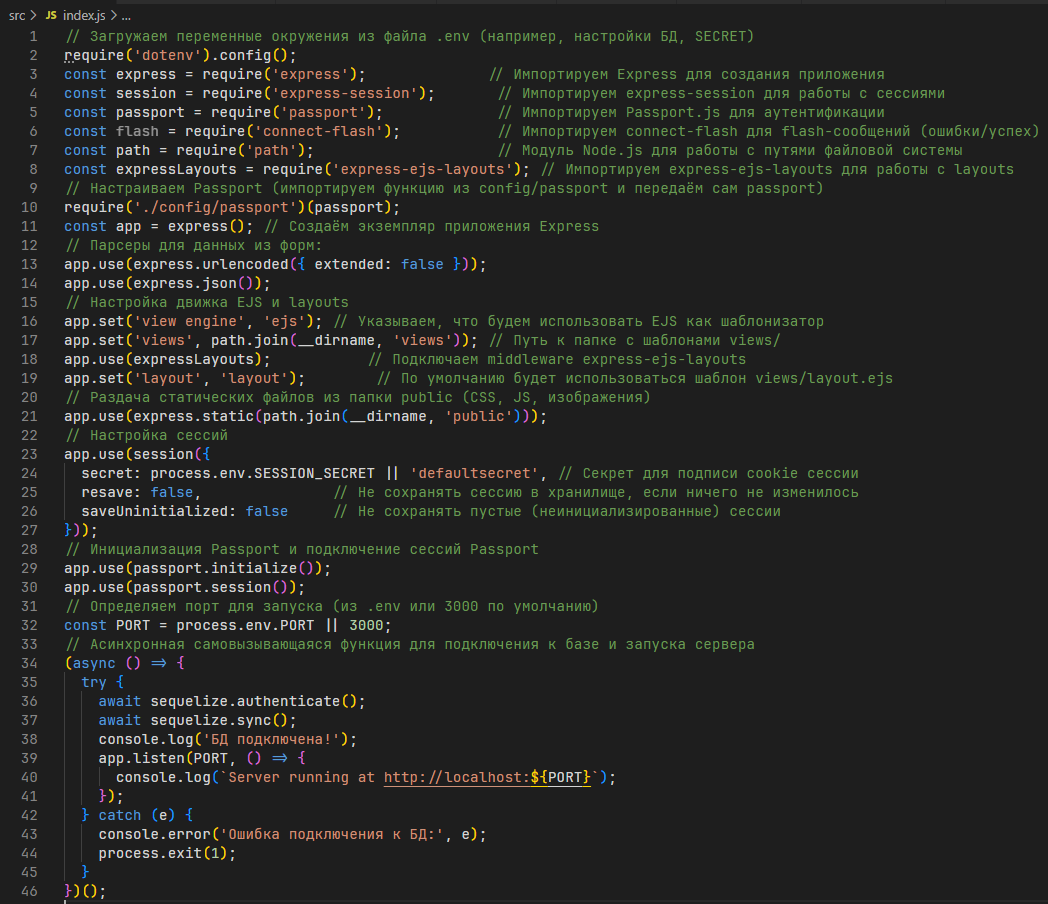


Рисунок 2. Настройка проекта.

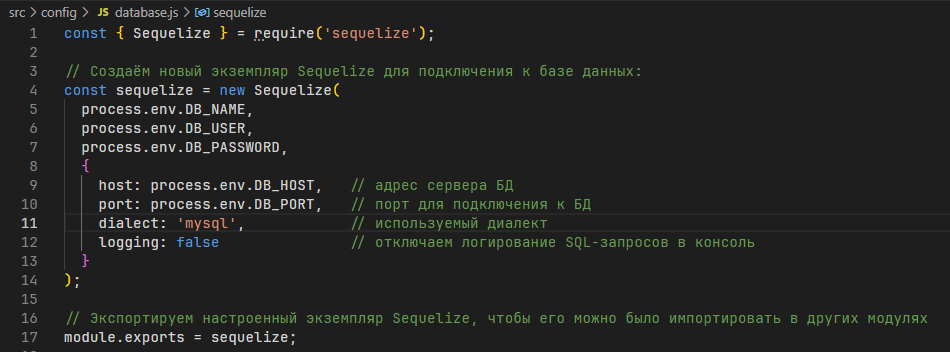


Рисунок 3. Подключение к БД с параметрами из .env.

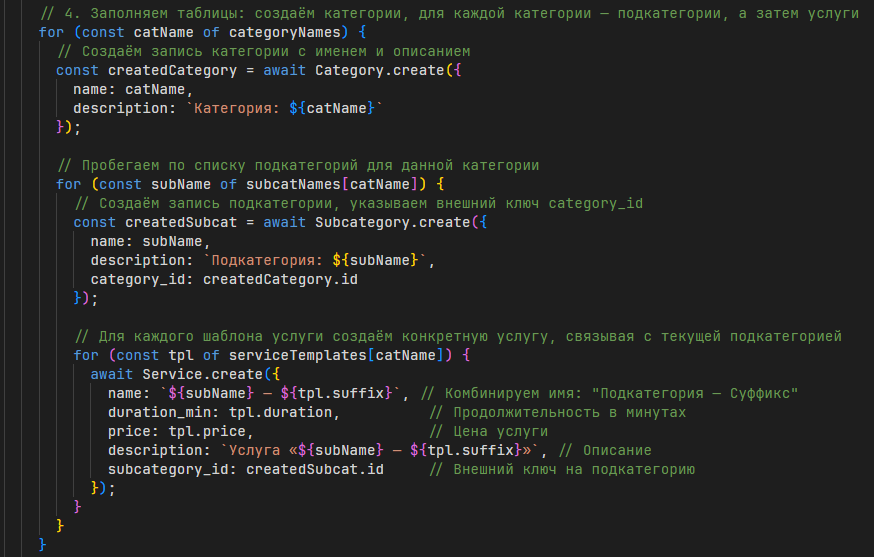


Рисунок 4. Работа seeder’a.

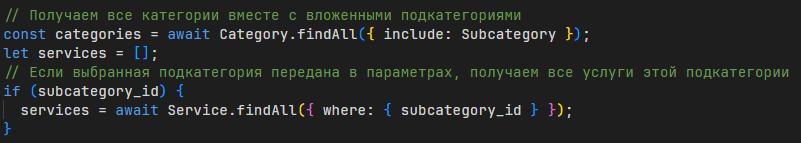


Рисунок 5. Обработка категорий, подкатегорий и услуг.

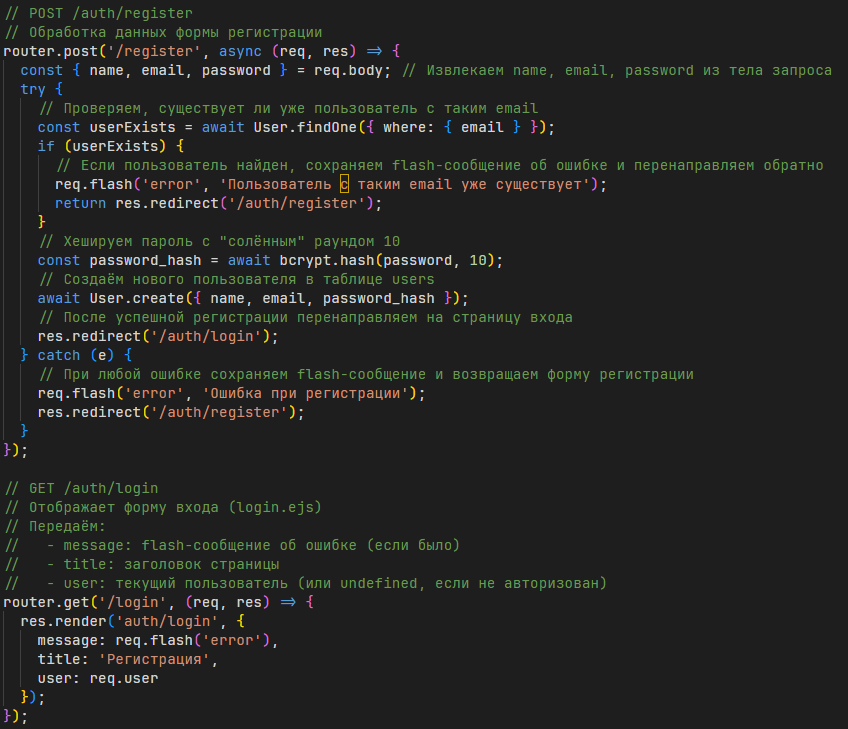


Рисунок 6. Роуты для отображения формы и для обработки введенных данных.

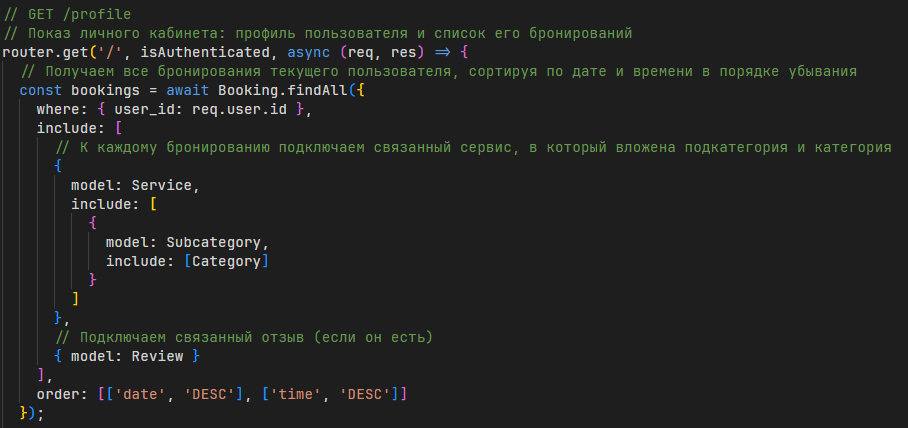


Рисунок 7. Роут профиля пользователя.

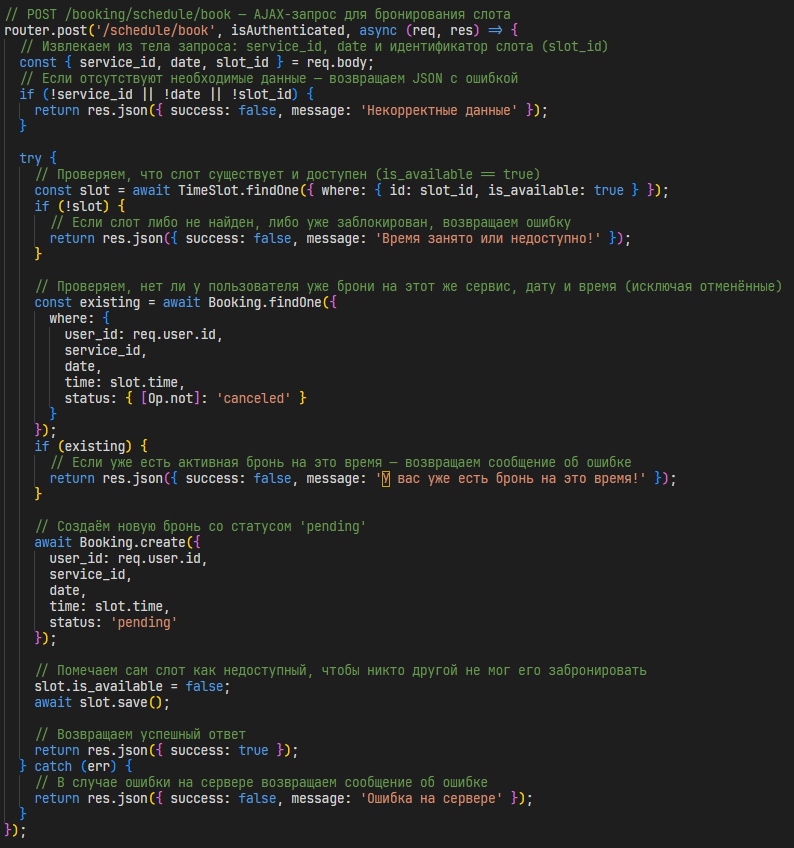


Рисунок 8. Обработка бронирования слота с условиями.

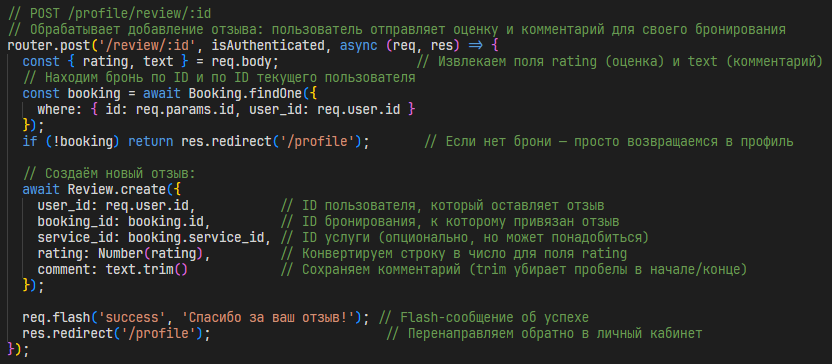


Рисунок 9. Реализация функции добавления отзыва и комментария.

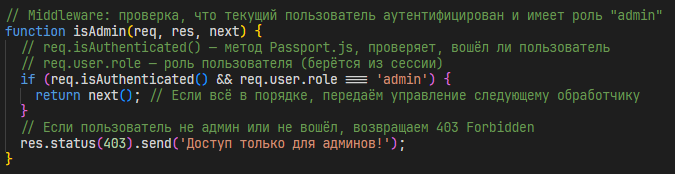


Рисунок 10. Защита от несанкционированного доступа.

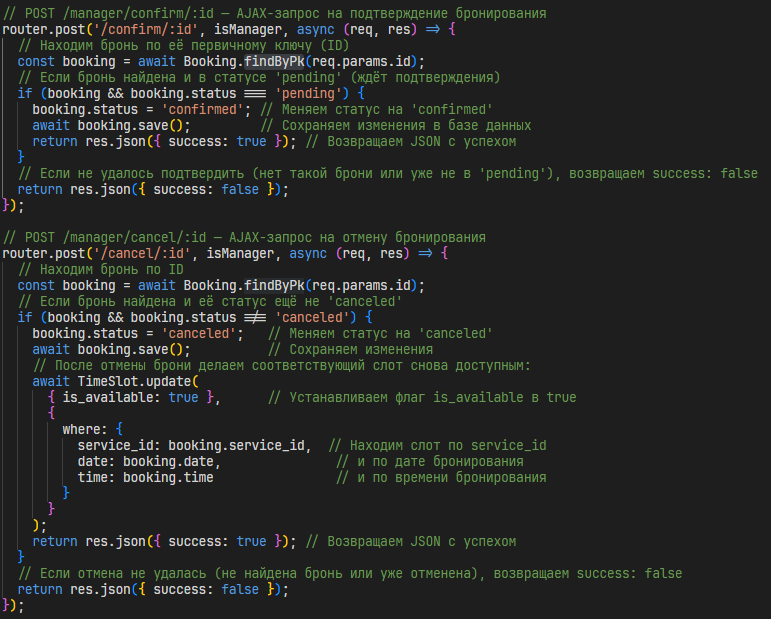


Рисунок 11. Маршрут для подтверждения/отмены бронирования.

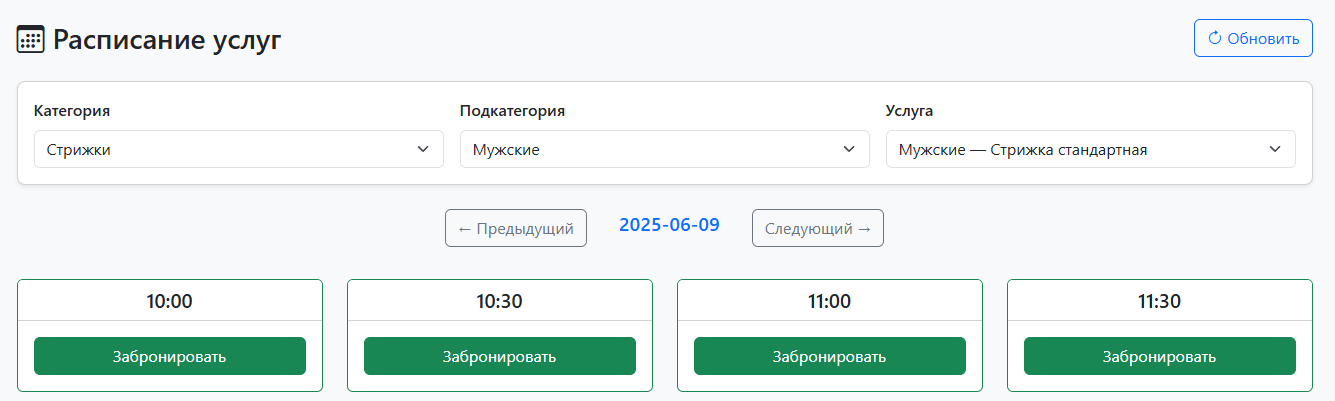


Рисунок 12. Главная страница с слотами для брони.

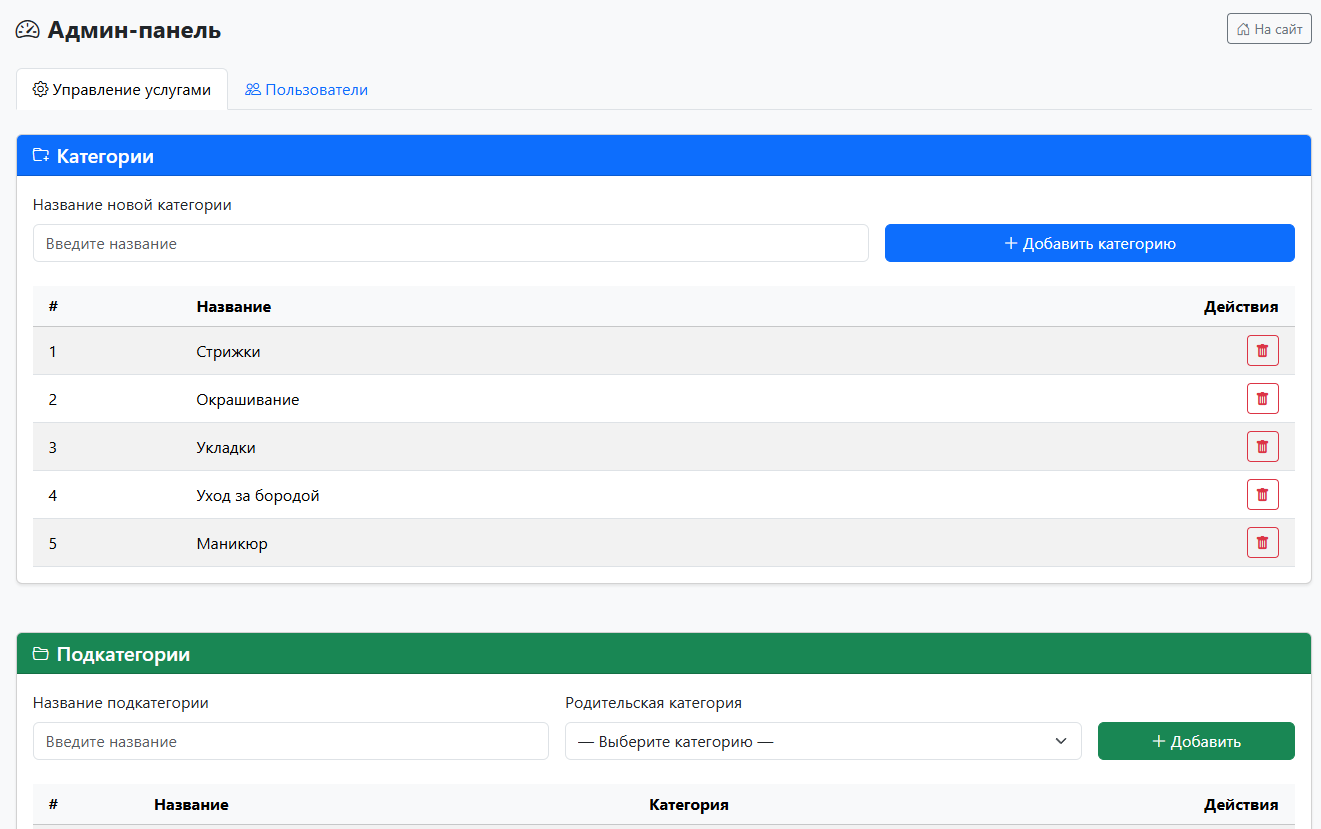


Рисунок 13. Админ-панель.

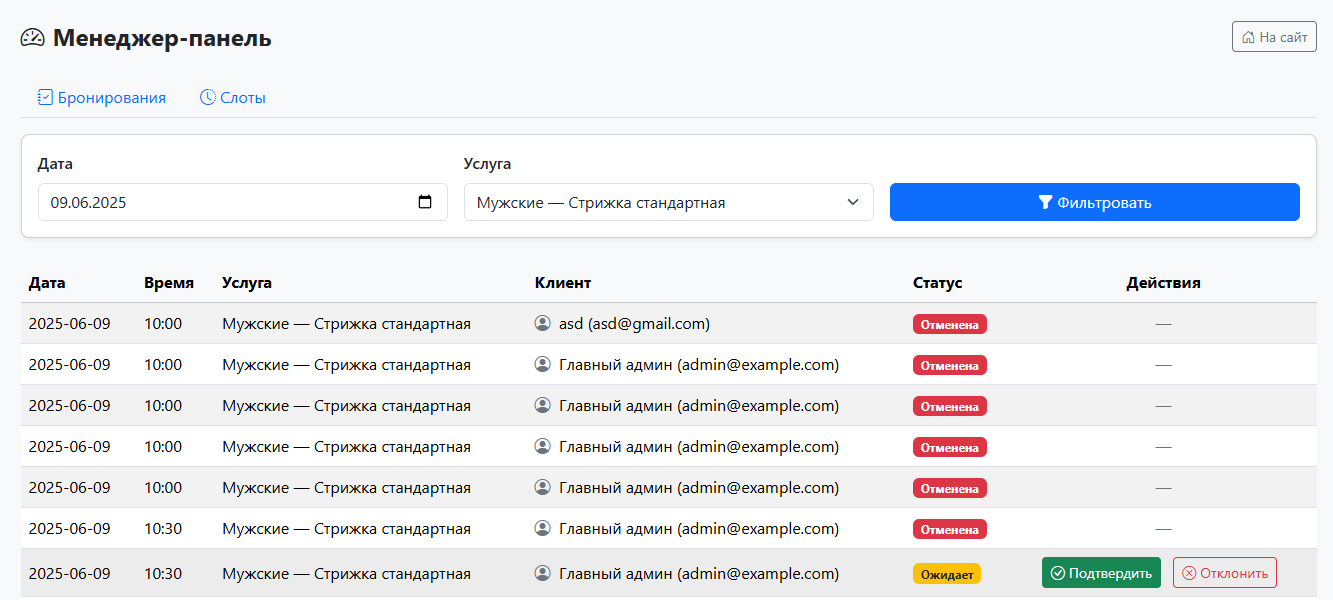


Рисунок 14. Менеджер-панель.