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](#_Toc197501843)

[1. Описание предметной области 4](#_Toc197501844)

[1.1 Краткая характеристика веб-приложения 4](#_Toc197501845)

[1.2 Описание программных инструментов и средств разработки 5](#_Toc197501846)

[1.2.1 Язык программирования «JavaScript» 5](#_Toc197501847)

[1.2.2 Фреймворк «Node.js» 5](#_Toc197501848)

[1.2.3 Серверная часть 5](#_Toc197501849)

[1.2.4 Среда разработки 5](#_Toc197501850)

[2. Практическая часть 6](#_Toc197501851)

[2.1 Проектирование веб-приложения 6](#_Toc197501852)

[2.2 Подготовка к работе 6](#_Toc197501853)

[2.3 Разработка веб-приложения 6](#_Toc197501854)

[2.4 Размещение веб-приложения на хостинге 6](#_Toc197501855)

[Заключение 7](#_Toc197501856)

[Список использованной литературы 8](#_Toc197501857)

# 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. Они позволяют автоматизировать процессы, минимизировать человеческий фактор и повысить удобство взаимодействия с клиентами.

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. Современная система бронирования способна автоматизировать эти процессы и существенно повысить качество сервиса.

Разработка такого веб-приложения — это не только практическая задача по программированию, но и решение реальной проблемы автоматизации бизнес-процессов в сфере оказания услуг.

# Описание предметной области

## 1.1 Краткая характеристика веб-приложения

Веб-приложение — это программный продукт, работающий через браузер и обеспечивающий интерактивное взаимодействие с пользователем. В отличие от настольных приложений, веб-приложения не требуют установки и доступны с любого устройства, подключённого к интернету. Это делает их особенно удобными для конечного пользователя и привлекательными для бизнеса.

Система бронирования услуг — это тип веб-приложения, основной задачей которого является автоматизация записи клиентов на определённые услуги. Такие системы позволяют пользователю выбрать услугу, специалиста, дату и время, а также дают бизнесу возможность управлять расписанием, принимать и подтверждать заявки, вести клиентскую базу.

Разрабатываемое приложение ориентировано на сферу парикмахерских услуг. Оно позволяет клиентам регистрироваться в системе, просматривать доступные услуги, выбирать мастера и бронировать подходящее время. В свою очередь, мастера могут управлять своим расписанием, а администратор — контролировать все процессы.

## 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 использовалась современная и удобная среда разработки, включающая в себя набор инструментов, обеспечивающих эффективную работу над веб-приложением.

Основным текстовым редактором стал Visual Studio Code, обладающий широкой функциональностью, удобным интерфейсом и поддержкой множества расширений. Благодаря встроенному терминалу, отладке кода и автодополнению, этот редактор стал идеальной платформой для написания кода на JavaScript и работе с Node.js.

Для управления версиями проекта применялась система Git, позволяющая отслеживать изменения в коде, работать в ветках и обеспечивать безопасное сохранение прогресса разработки. Репозиторий размещался на GitHub, что упростило резервное копирование и доступ к проекту с разных устройств.

В процессе настройки зависимостей проекта использовался npm (Node Package Manager) — стандартный менеджер пакетов для Node.js. С его помощью подключались библиотеки, такие как Express, Sequelize, Passport.js, bcryptjs и другие.

Для тестирования API-запросов применялась программа Postman, которая позволила проверить корректность обработки маршрутов, аутентификации, получения и отправки данных между клиентом и сервером.

Работа с базой данных осуществлялась через MySQL Workbench — графическую оболочку для управления базами данных MySQL. С её помощью создавались таблицы, проверялись SQL-запросы и визуализировалась структура данных.

Для размещения веб-приложения на сервере использовался хостинг Beget, предоставляющий доступ по SSH и инструменты для настройки серверного окружения. Установка зависимостей, настройка окружения и запуск приложения происходили через командную строку с использованием утилиты PM2, обеспечивающей автоматический перезапуск приложения при сбоях.

Также в проекте применялись файлы .env, содержащие переменные окружения — такие как параметры подключения к базе данных, секретные ключи и другие конфиденциальные данные. Это обеспечивало безопасность и гибкость при развёртывании проекта в разных средах.

Таким образом, выбранная среда разработки сочетала в себе удобство, производительность и надёжность, что обеспечило успешную реализацию проекта с возможностью дальнейшего масштабирования и поддержки.

# 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

Проектирование архитектуры приложения является одним из самых важных этапов разработки. На этом этапе мы детально проработали сущности, которые будут служить основой для всех операций с данными и взаимодействия с пользователями. Основные сущности приложения включают Пользователей, Услуги, Записи, Отзывы и Парикмахеров (Рис. 1). Каждая из этих сущностей была тщательно спроектирована для обеспечения надежности, удобства и гибкости системы.

Пользователи являются основными участниками системы. Каждый пользователь, будь то клиент или администратор, имеет уникальный аккаунт, который включает контактные данные, настройки профиля и историю бронирований. Для обеспечения безопасности и удобства пользователей была реализована система регистрации и аутентификации. Мы использовали стандартные механизмы безопасности, такие как хеширование паролей с использованием bcrypt и валидацию данных на серверной стороне, чтобы гарантировать защиту информации. В зависимости от роли пользователя (клиент или администратор) предоставляются разные уровни доступа и функционала.

Услуги — это перечень всех доступных для бронирования действий, которые могут быть выполнены парикмахерами. Каждая услуга имеет подробное описание, цену и продолжительность. Услуги классифицируются по различным критериям, таким как тип услуги (стрижка, укладка, окрашивание) или сложность, что позволяет пользователям легко выбирать нужную услугу. Важно отметить, что каждый парикмахер может предложить свой набор услуг, который можно будет отфильтровать по категориям на веб-странице.

Записи — это процессы бронирования услуг пользователями. Каждая запись привязана к определенному пользователю, парикмахеру и услуге, а также имеет дату и время выполнения. Записи могут быть в разных состояниях (например, "забронировано", "выполнено", "отменено"), что помогает отслеживать их текущий статус. Эта сущность является ключевой для управления расписанием парикмахеров и позволяет пользователю забронировать услугу на удобное время.

Отзывы — это мнения пользователей о качестве оказанных услуг. Каждый отзыв связан с конкретной записью и услугой, что дает возможность другим пользователям ознакомиться с опытом других людей. Отзывы могут включать оценку, текстовое описание, а также фотографии или видеоматериалы, что делает их более информативными и полезными.

Для хранения данных была выбрана реляционная база данных MySQL, что обеспечило надежность и удобство работы с данными. База данных была спроектирована с использованием ORM Sequelize, что позволило избежать прямых SQL-запросов и упростило взаимодействие с базой данных.

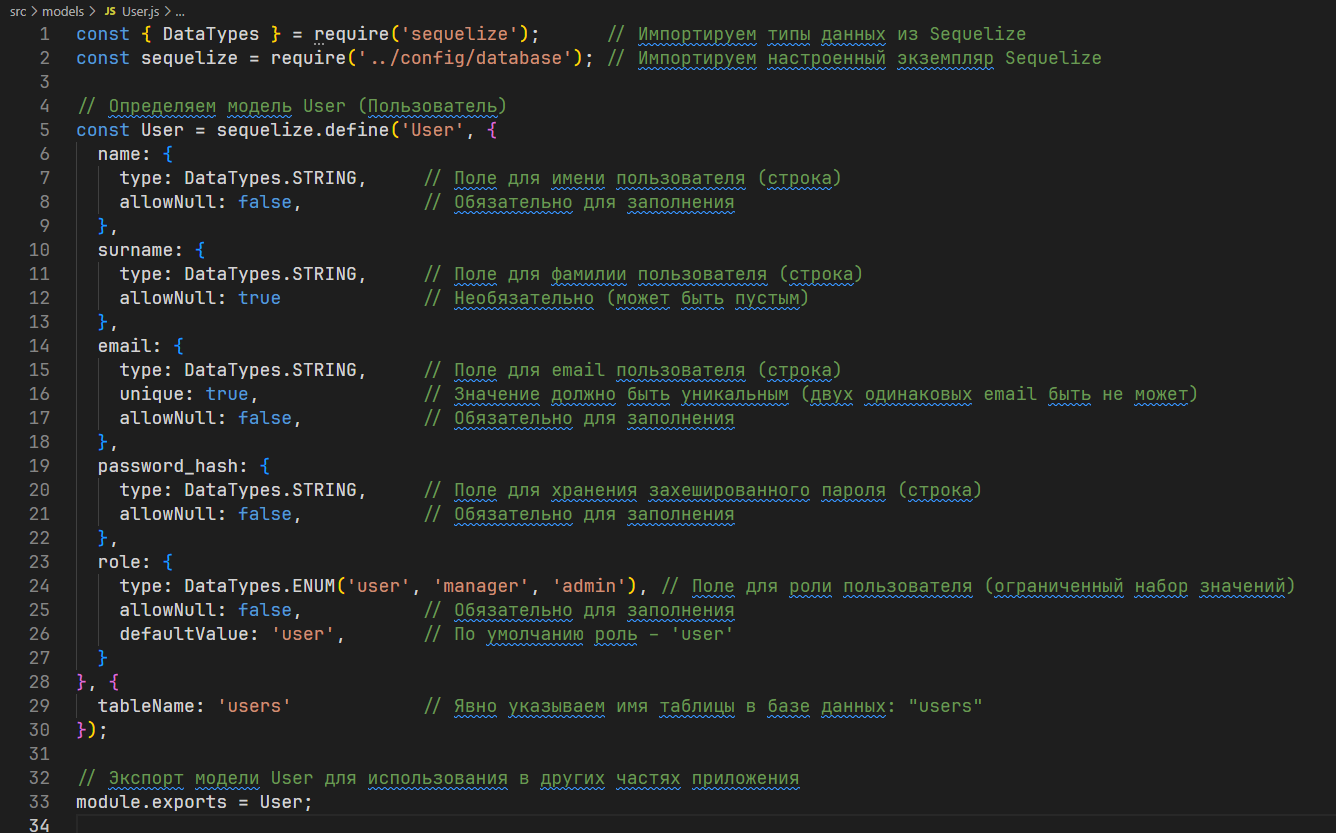


Рисунок 1. Модель user’a.

2.3.2 Организация маршрутов и логики обработки запросов

Одним из ключевых аспектов работы веб-приложения является эффективная маршрутизация запросов, которая должна обеспечивать быструю и корректную обработку действий пользователей. В нашем приложении маршруты были разделены на публичные и защищённые. Публичные маршруты доступны всем пользователям, включая неавторизованных, и включают страницы регистрации, авторизации, а также страницы с информацией о доступных услугах и парикмахерах (Рис. 2). Эти маршруты позволяют пользователю ознакомиться с системой и начать взаимодействовать с приложением.

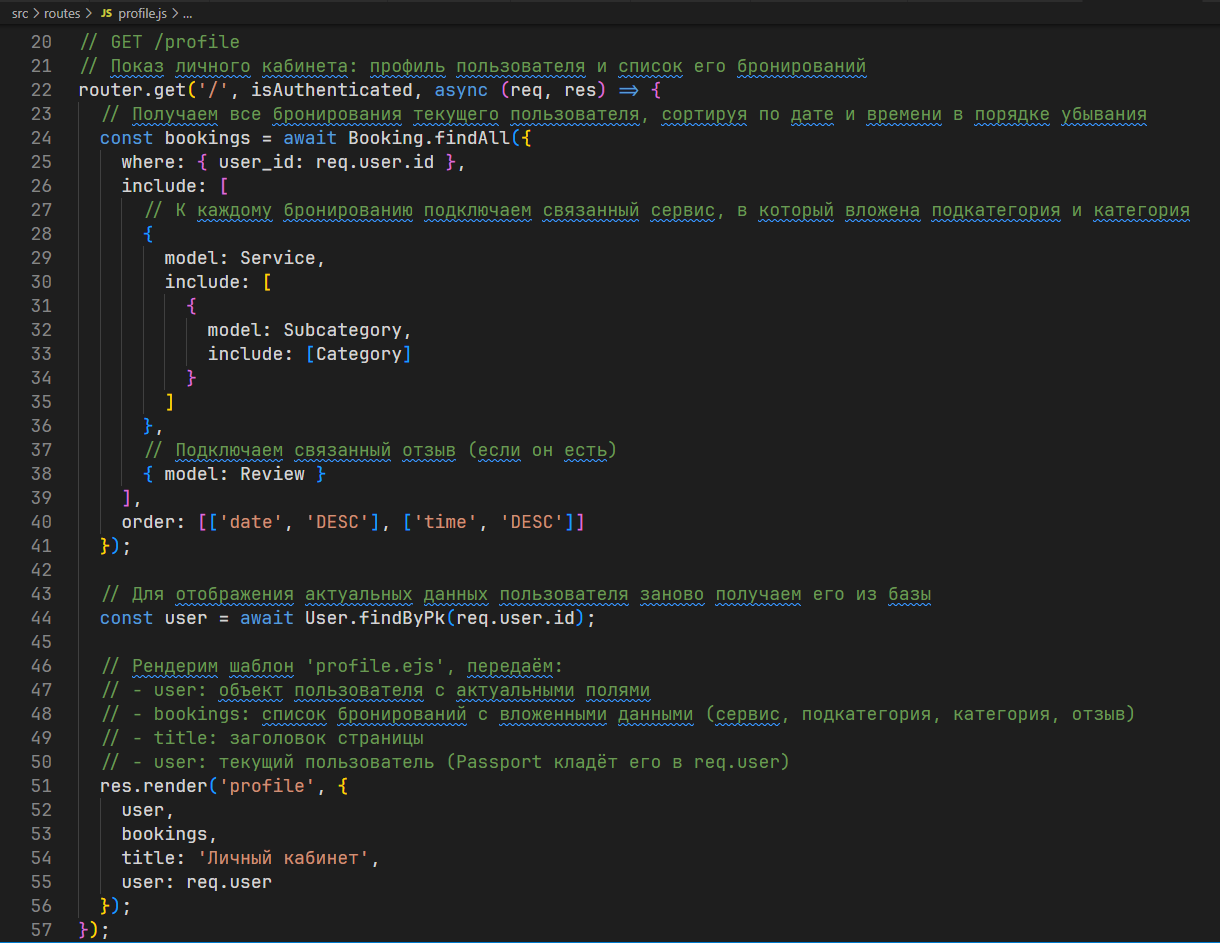


Рисунок 2. Публичный маршрут.

Защищённые маршруты доступны только авторизованным пользователям. Это могут быть страницы для создания записей, редактирования профиля или управления бронированиями. Контроллеры для таких маршрутов обрабатывают логику, связанную с пользователем, в том числе проверку прав доступа и взаимодействие с базой данных. Например, пользователю, не авторизованному в системе, будет отказано в доступе к странице профиля или созданию записи, тем более входу в админ-панель. (Рис. 3)



Рисунок 3. Защищенный маршрут.

Каждый маршрут организован с использованием Express.js, что позволяет создавать чистую и легко поддерживаемую систему маршрутов. Контроллеры обеспечивают выполнение бизнес-логики для обработки данных, отправленных пользователем, и взаимодействуют с моделями данных для сохранения или получения информации.

2.3.3 Интерфейс пользователя и дизайн

Для создания удобного и интуитивно понятного интерфейса для всех категорий пользователей был использован фреймворк Bootstrap. Это позволило быстро реализовать адаптивный дизайн, который корректно отображается как на мобильных устройствах, так и на большом экране. Все элементы интерфейса были разработаны с учетом потребностей конечных пользователей, чтобы сделать их взаимодействие с приложением максимально удобным.

Основные страницы включают главную страницу, страницу профиля пользователя и страницу бронирования. На главной странице пользователи могут увидеть все доступные услуги, отсортированные по типу, цене или рейтингу. Страница профиля позволяет пользователям редактировать свои личные данные, просматривать историю своих записей и оставлять отзывы о полученных услугах (Рис. 4). Страница бронирования позволяет выбрать услугу, выбрать парикмахера, указать удобное время для записи и завершить процесс бронирования.

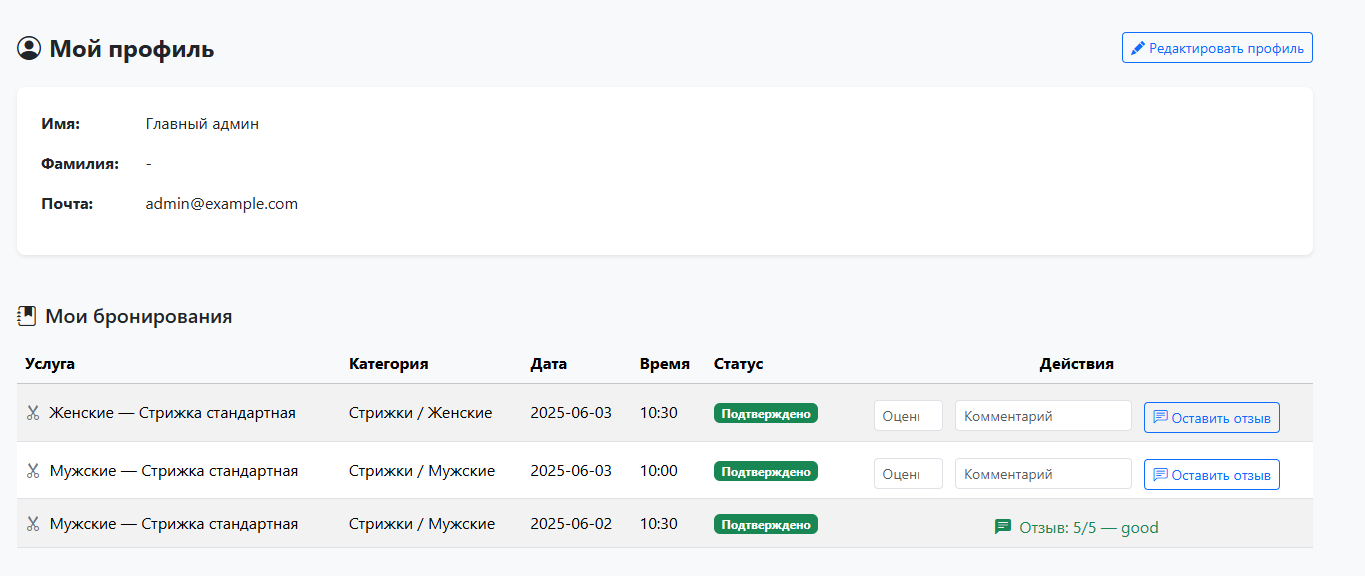


Рисунок 4. Профиль пользователя.

Дизайн страницы бронирования был продуман таким образом, чтобы минимизировать количество шагов для пользователя при бронировании. Формы с валидацией на стороне клиента и сервера помогают предотвратить ошибки ввода данных, а доступные парикмахеры и услуги отображаются только в нужном контексте (Рис. 5).

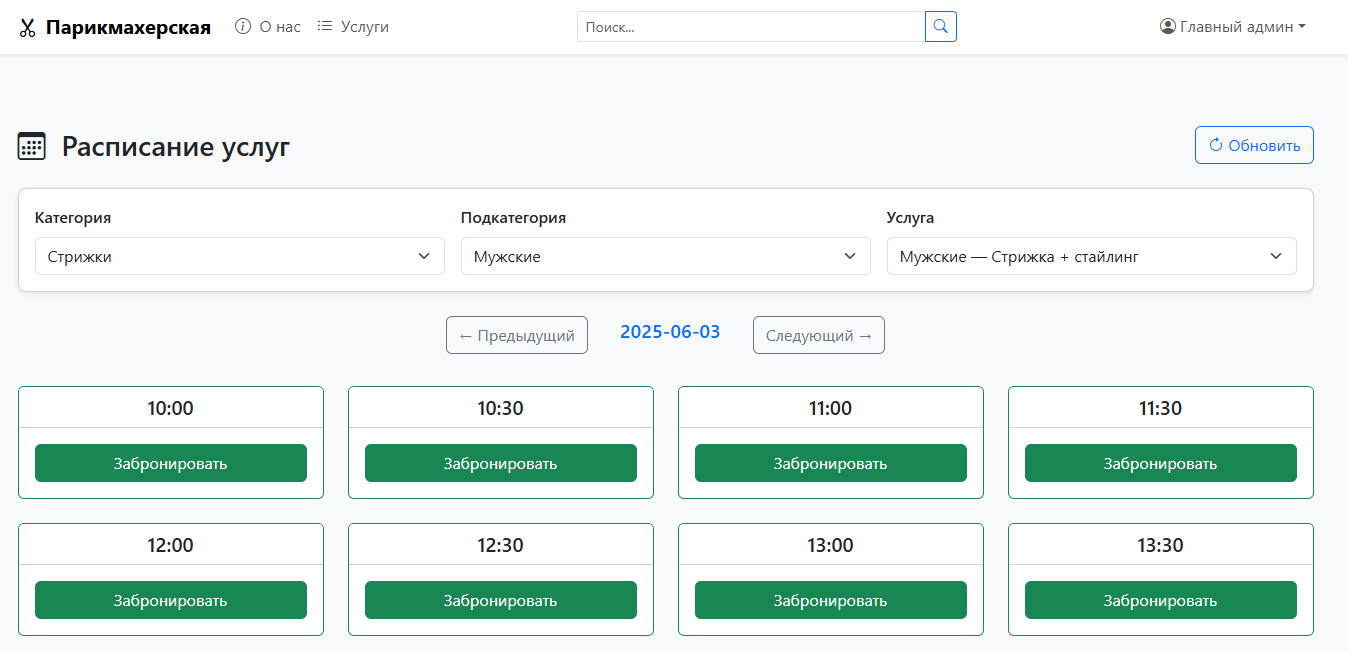


Рисунок 5. Страница бронирования.

2.3.4 Реализация ключевых модулей

Для полноценного функционирования веб-приложения были реализованы несколько ключевых модулей, каждый из которых решает важную задачу в системе. Модуль регистрации и аутентификации позволяет пользователю создать аккаунт с помощью электронной почты и пароля, а также предоставляет систему безопасности с хешированием паролей и проверкой уникальности.

Модуль записи на услугу включает в себя функциональность выбора услуги, парикмахера, времени и даты. После бронирования пользователь получает уведомление о подтверждении своей записи, а сама запись сохраняется в базе данных. Пользователи могут фильтровать доступные услуги по типу, цене или времени.

Модуль профиля позволяет пользователю редактировать личные данные, просматривать историю своих записей и оставлять отзывы. Также пользователи могут управлять настройками аккаунта, такими как пароль или предпочтительные уведомления.

Административная панель дает администратору возможность управлять пользователями, просматривать их профили, редактировать информацию, а также блокировать аккаунты при необходимости (Рис. 6). Панель также включает функциональность управления услугами, расписаниями парикмахеров и статистикой по бронированиям.

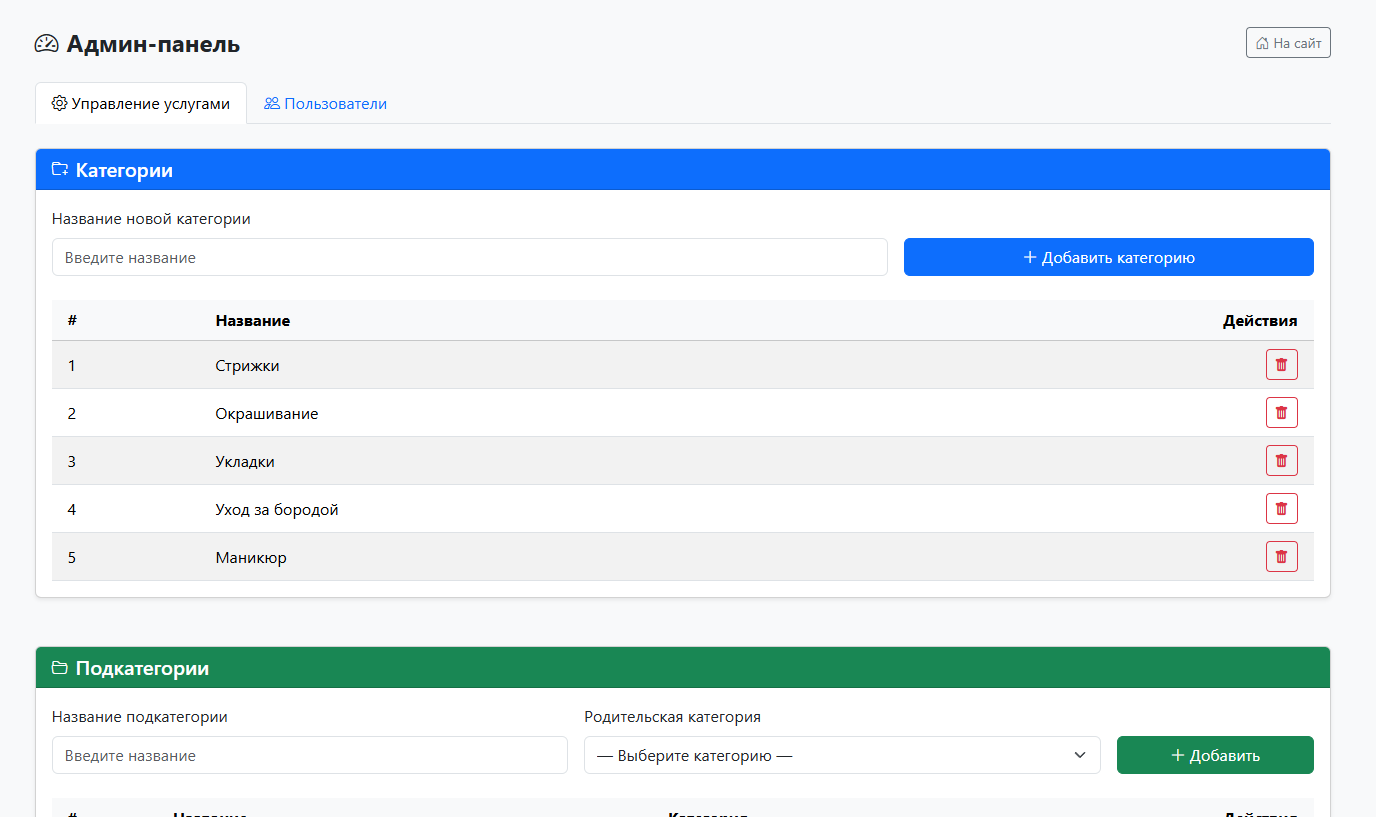


Рисунок 6. Админ-панель.

2.3.5 Безопасность и технические меры

Для обеспечения безопасности веб-приложения были предприняты несколько технических мер. Все пароли пользователей хранятся в базе данных в зашифрованном виде с использованием bcrypt, что гарантирует их защиту даже в случае утечки данных (Рис. 7). Также для предотвращения атак CSRF используется токен CSRF, который добавляется ко всем запросам и защищает от подделки запросов.

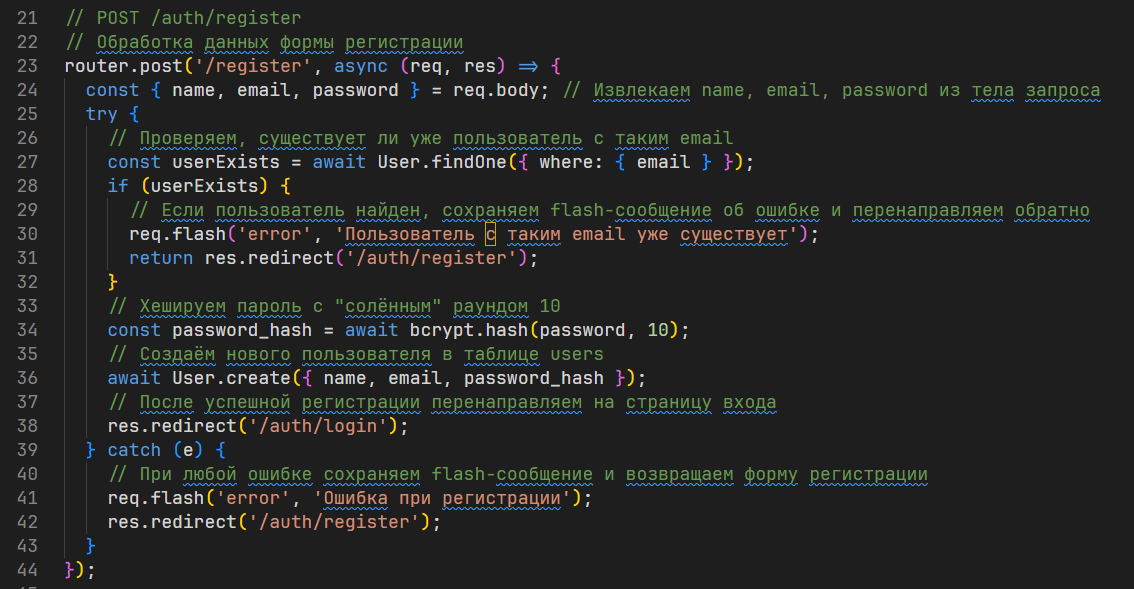


Рисунок 7. Обработка данных формы регистрации с использованием bcrypt.

Кроме того, система включает в себя валидацию данных как на клиентской, так и на серверной стороне. Это гарантирует, что данные, вводимые пользователем, всегда корректны и безопасны для хранения. Для защиты информации, доступной только авторизованным пользователям, используются механизмы контроля доступа, такие как middleware, которые ограничивают доступ к определённым функциям приложения в зависимости от роли пользователя.

Проект веб-приложения для бронирования парикмахерских услуг представляет собой пример успешного использования современных технологий для создания функциональной, удобной и безопасной системы. Система, реализованная в этом проекте, позволяет эффективно управлять бронированиями, а также обеспечивает высокий уровень безопасности и удобство для пользователей. В дальнейшем проект может быть расширен дополнительными функциями, такими как интеграция с платёжными системами, улучшение интерфейса, добавление новых возможностей для управления данными и анализа активности пользователей.

## 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. Флэнаган Д.* JavaScript. Подробное руководство*. — СПб.: Питер, 2022.*

*2. Бэнкс А., Порселл Э.* Node.js в действии*. — М.: Диалектика, 2021.*

*3. Браун Э.* Express.js: быстрый старт*. — СПб.: БХВ-Петербург, 2020.*

*4.* Документация Node.js — <https://nodejs.org/en/docs/>

*5.* Документация Express.js — <https://expressjs.com/>

*6.* Sequelize ORM — <https://sequelize.org/>

7. Mozilla Developer Network (MDN) — <https://developer.mozilla.org/>

8. Bootstrap 5 Documentation — <https://getbootstrap.com/>

9. Postman API Platform — <https://www.postman.com/>