МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ИНФОРМАЦИОННАЯ СИСТЕМА ОНЛАЙН-БРОНИРОВАНИЯ АВТОБУСНЫХ БИЛЕТОВ С РЕАЛИЗАЦИЕЙ ПАНЕЛИ АДМИНИСТРАТИВНОГО УПРАВЛЕНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Симагиной Анастасии Владимировны

Научный руководитель
доцент, к. т. н. _____ М. В. Хамутова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

ВВЕДЕНИЕ

Целью выпускной квалификационной работы является разработка информационной системы онлайн-бронирования автобусных билетов, оснащённой комплексной панелью административного управления.

Для достижения указанной цели были поставлены следующие задачи:

— Провести анализ аналогов систем бронирования и сформулировать функциональные требования.
— Спроектировать архитектуру веб-приложения, разработать структуру базы данных, создать UML-диаграммы логики работы системы.
— Реализовать пользовательский модуль с регистрацией и авторизацией, профилем, поиском и бронированием билетов.
— Реализовать административную панель с управлением пользователями и с редактированием и созданием билетов.

Разработка системы онлайн-бронирования билетов на автобус представляет собой важный шаг к улучшению качества обслуживания пассажиров и оптимизации процессов управления транспортом. Такая система позволяет пользователям легко и быстро находить доступные маршруты, выбирать удобные места и бронировать билеты, не выходя из дома. Это экономит время и снижает вероятность ошибок при ручном вводе данных. Кроме того, система предоставляет пассажирам актуальную информацию о расписании, изменениях в маршрутах и наличии мест.

Особое внимание в работе уделено административному компоненту системы, который обеспечивает управление её ключевыми аспектами. Реализованная административная панель предоставляет администратору удобный инструмент для полного контроля за работой сервиса. Её функционал включает гибкие настройки билетов: создание и редактирование маршрутов, управление расписанием и контроль занятости мест. Кроме того, административная панель обеспечивает возможности для модерации пользователей, позволяя просматривать их активность, а также при необходимости блокировать аккаунты для поддержания безопасности и порядка в системе.

В рамках данной работы рассмотрен процесс разработки системы онлайн-бронирования билетов на автобус и панели административного управления, включая анализ требований, проектирование архитектуры, выбор технологий и реализацию ключевых функций.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и двух приложений. Общий объем работы – 55 страниц, из них 40 страниц — основное содержание, включая 17 рисунков, список использованных источников информации — 23 наименования.

1 Анализ предметной области разработки системы онлайн-бронирования и панели администратора

1.1 Актуальность создания системы онлайн-бронирования билетов и панели администратора

Актуальность разработки системы онлайн-бронирования автобусных билетов обусловлена растущим спросом на удобные цифровые сервисы. Система позволяет пользователям круглосуточно искать и бронировать билеты без географических и временных ограничений, экономит время, предоставляет актуальные данные о расписании, ценах и наличии мест в реальном времени, что снижает ошибки и повышает доверие.

Рост интернет-торговли и мобильных устройств переносит спрос на услуги бронирования в цифровую среду. В условиях повышения мобильности населения возрастает потребность в унифицированных платформах для планирования поездок. С операционной точки зрения онлайн-системы устойчивы и адаптивны к изменениям рынка, позволяя быстро корректировать расписания, тарифы и маршруты.

1.2 Постановка задачи и цели проекта

Целью выпускной квалификационной работы является разработка информационной системы онлайн-бронирования автобусных билетов, оснащённой комплексной панелью административного управления.

Для достижения указанной цели были поставлены следующие задачи:

- провести анализ предметной области и исследовать существующие аналогичные системы онлайн-бронирования, выявив их преимущества и недостатки;
- выполнить постановку задачи и сформулировать функциональные требования к разрабатываемой системе;
- спроектировать архитектуру веб-приложения, включая клиентскую часть и серверную логику;
- разработать структуру базы данных для хранения информации о пользователях, маршрутах, рейсах, бронированиях и местах;
- создать UML-диаграммы для описания логики работы системы;
- реализовать модуль регистрации и авторизации пользователей с разграничением прав доступа;

— разработать функционал поиска и фильтрации рейсов по направлениям, датам и времени;
— создать интерфейс для просмотра доступных мест и выполнения бронирования билетов;
— реализовать личный кабинет пользователя для просмотра истории бронирований и управления своим профилем;
— разработать интерфейс административной панели для управления системой;
— реализовать функционал управления пользователями: просмотр списка, блокировка и разблокировка аккаунтов;
— создать модуль управления билетами и маршрутами: добавление, редактирование и удаление рейсов.

1.3 Обзор существующих систем бронирования билетов

Существуют различные системы бронирования, например туту.ру, busfor и avtovokzaly.ru. Они предлагают широкий выбор маршрутов, сравнение цен, выбор мест и разные способы оплаты.

Разработанная система обладает ролевой моделью с JWT-аутентификацией, удобной админ-панелью и высоким уровнем безопасности. Архитектура на React, Flask и PostgreSQL масштабируема. Главное преимущество перед аналогами — отсутствие комиссии, полный контроль над данными пассажиров и гибкая настройка под нужды перевозчика. Система подходит для небольших транспортных компаний, желающих иметь собственный канал продаж без посредников. Недостатки: отсутствие онлайн-оплаты, уведомлений, мобильного приложения и системы лояльности. Тем не менее, как локальное решение она выигрывает в простоте и прозрачности.

1.4 Анализ основных алгоритмов разработки системы

Для создания эффективной системы онлайн-бронирования автобусных билетов требуется применение ряда ключевых алгоритмов. Пользовательский интерфейс базируется на алгоритмах поиска оптимальных маршрутов, фильтрации по времени, цене и датам, а также сортировке результатов. Надёжность системы обеспечивают алгоритмы проверки доступности мест в реальном времени с применением транзакций и алгоритмы управления квотой мест. Для администрирования необходимы алгоритмы создания, редактирования и

удаления записей о билетах и маршрутах с валидацией данных, а также алгоритмы просмотра пользователей, их поиска и принудительной блокировки или разблокировки аккаунтов. Безопасность доступа поддерживается алгоритмами аутентификации и авторизации на основе JWT-токенов с разделением на access и refresh токены. Для защиты информации применяются параметризованные SQL-запросы, экранирование пользовательского ввода и ограничение частоты запросов.

1.5 Обзор основных инструментальных средств для системы онлайн-бронирования и обоснование их выбора

Для разработки системы бронирования выбраны JavaScript и Python. JavaScript используется для создания интерактивных интерфейсов на стороне клиента, что ускоряет отклик и снижает нагрузку на сервер. Он поддерживается всеми браузерами, позволяет выполнять асинхронную загрузку данных, интегрируется с HTML и CSS, а также имеет большое сообщество и множество фреймворков (React, Angular, Vue.js). Python выбран благодаря быстрой обработке данных, удобству тестирования, мощным фреймворкам и широкой поддержке сообщества.

Из фреймворков Python выбран Flask — компактный, гибкий и простой в использовании. Он минималистичен, не навязывает структуру приложения, легко интегрируется с дополнительными библиотеками и подходит для быстрой разработки.

Для клиентской части выбрана библиотека React.js. Её главные особенности — компоненты и состояния. React использует виртуальный DOM, что позволяет обновлять только изменившиеся части интерфейса и повышает производительность. Благодаря обширному сообществу доступно множество готовых компонентов и инструментов для разработки, отладки и тестирования. React также легко интегрируется с разными технологиями и платформами.

1.6 Обоснования выбора и описание метода разработки системы онлайн-бронирования билетов

При разработке системы онлайн-бронирования автобусных билетов важно учитывать удобство пользователей и масштабируемость. Существует несколько методов разработки. Waterfall — линейный подход с чёткой структурой,

подходящий для небольших проектов с неизменными требованиями. RAD фокусируется на быстром прототипировании и снижении рисков.

Для данной системы выбран метод Agile — итеративный подход, обеспечивающий гибкость и быстрое реагирование на изменения. Agile позволяет адаптироваться к требованиям пользователей, получать обратную связь и улучшать продукт на реальных данных, а также способствует тесному взаимодействию в команде. Основные этапы включают сбор требований, планирование спринтов, разработку, тестирование и обратную связь. Преимущества Agile для этой системы: постоянная обратная связь для создания удобного интерфейса, быстрое выявление и исправление ошибок, а также адаптивность к изменениям рынка и требований.

1.7 Описание архитектуры системы онлайн-бронирования

Система построена на архитектуре «клиент-сервер», обеспечивающей разделение ответственности, масштабируемость и удобство сопровождения.

Клиентская часть — это одностраничное приложение на React с TypeScript. Пользователи ищут маршруты, выбирают места и оформляют бронирование. Клиент взаимодействует с сервером через HTTP-запросы в формате JSON без перезагрузки страницы. Для управления состоянием используются React-контексты, для маршрутизации — TanStack Router.

Серверная часть реализована на Flask и предоставляет RESTful API, который обрабатывает запросы, взаимодействует с БД и возвращает результаты. Логика включает поиск маршрутов, проверку мест, управление бронированиями, аутентификацию и авторизацию. Для производительности используются многопоточность и пул соединений с БД. Данные передаются в JSON через методы GET, POST, PUT, DELETE.

База данных — PostgreSQL с поддержкой транзакций и индексов. Хранит маршруты, автобусы, пользователей, бронирования и места. Внешние ключи обеспечивают целостность, индексы ускоряют поиск.

Безопасность: JWT-аутентификация с access и refresh токенами, ролевая модель (пользователь/администратор), хеширование паролей, HTTPS, защита от SQL-инъекций через параметризованные запросы, экранирование ввода и ограничение частоты запросов.

Для визуализации структуры разработаны ER-диаграмма и UML-диаграммы.

2 Описание разработки системы

2.1 Разработка frontend-части системы

Структура React-приложения построена по модульному принципу. В папке app — глобальные стили и логика инициализации. contexts содержит React-контексты (например, AuthProvider). entities — ядро предметной области (пользователь, билет, маршрут) и валидация. features — процессы поиска, бронирования, администрирования. pages — компоненты страниц с отдельными URL. Маршрутизация хранится в routes.

Переиспользуемые UI-компоненты (кнопки, поля ввода) собраны в shared, сложные блоки (шапка, карточка билета) — в widgets. Точка входа — main.tsx.

В проекте реализованы главная страница, страница поиска, авторизация и регистрация пользователей, личный профиль, процесс оформления билета, административная панель с возможностью редактирования билетов и просмотра списка пользователей, а также модальные окна для различных взаимодействий.

Среди реализованных компонентов: карточки билетов, поле поиска, пагинация, фильтры, схема автобуса с местами, уведомления, индикаторы загрузки, защищённый маршрут, валидация форм, шапка со статусом авторизации и всплывающие подсказки.

2.2 Разработка backend-части системы

В серверной части системы онлайн-бронирования были реализованы следующий функционал:

- авторизация, регистрация пользователя;
- поиск билетов;
- оформление билета;
- администрирование билетов и пользователей;
- получение списка заказов.

При регистрации пользователь добавляется в таблицу users. Поиск билетов выполняется по городам и дате через POST-запрос. При оформлении билета открывается транзакция: проверяется наличие мест, создаётся заказ, резервируются места. При ошибке изменения откатываются. Администратор может получить список билетов с пагинацией и создать новый билет. Заказы пользователя возвращаются из соединения таблиц, отсортированными по дате

создания.

2.3 Проектирование базы данных

В ходе работы созданы модели базы данных: пользователи, билеты и заказы.

Модель пользователей хранит идентификатор, email (уникальный), логин, хэшированный пароль, персональные данные (ФИО, дату рождения, пол, паспорт), а также время создания и обновления записи.

Модель билетов содержит города, дату и время отправления и прибытия, стоимость, время в пути и количество доступных мест. Для ускорения поиска созданы индексы по городам, датам и составной индекс по направлению и дате.

Модель заказов связывает пользователя и билет через внешние ключи с каскадным удалением. Включает номер места, данные пассажира, стоимость и время создания заказа.

Связи: один пользователь может иметь много заказов, один билет может быть забронирован многими пользователями. Внешние ключи обеспечивают ссылочную целостность.

Типы данных: SERIAL (ключи), VARCHAR, INTEGER, TIMESTAMP, DATE, TIME. Ограничения: NOT NULL и UNIQUE на email.

2.4 Взаимодействие frontend и backend

Взаимодействие бекенда (Flask API) и фронтенда (React) происходит через HTTP-запросы по схеме клиент-сервер.

При аутентификации фронтенд отправляет учётные данные с предварительной валидацией, бекенд проверяет их в БД и генерирует JWT-токены. Фронтенд сохраняет токены в глобальном состоянии React и автоматически добавляет их в заголовки защищённых запросов через интерцепторы. Реализована автоматическая проверка истечения токена с попыткой обновления.

При управлении билетами используется двухуровневая валидация: на фронтенде для мгновенного отклика и на бекенде для безопасности. Операции создания, обновления и удаления выполняются с транзакционной обработкой. Проверяются корректность дат, цены и доступность мест. После операции интерфейс обновляется в реальном времени.

Управление пользователями включает просмотр списка с фильтрацией и пагинацией, блокировку и разблокировку пользователей с подтверждением. Администратор видит детальную информацию о заказах каждого пользователя. В основе системы лежит ролевая модель: обычные пользователи и администраторы.

Настроена CORS-политика, разрешающая запросы только с указанных источников. Реализована комплексная валидация: мгновенная на фронтенде и глубокая на бекенде. Защита от SQL-инъекций — через параметризованные запросы, от XSS-атак — экранированием пользовательского ввода.

2.5 Обеспечение безопасности данных в веб-приложении

Для безопасности приложения реализовано несколько механизмов.

Аутентификация основана на JWT-токенах: при успешном входе сервер генерирует токен с идентификатором пользователя и сроком 24 часа. Токен сохраняется в localStorage и отправляется в заголовке Authorization при каждом запросе к защищённым эндпоинтам.

Декоратор token_required проверяет валидность токена и наличие пользователя в базе данных.

Компонент ProtectedRoute на фронтенде проверяет наличие токена, обновляет состояние авторизации, для административных маршрутов выполняет дополнительную проверку прав и перенаправляет неавторизованных пользователей на страницу входа.

Пароли хешируются алгоритмом SHA-256 и никогда не хранятся в открытом виде. При аутентификации сравниваются хеши.

На клиенте реализована строгая валидация пароля: минимум 6 символов, наличие строчной и заглавной буквы, а также цифры.

Смена пароля при обновлении профиля возможна только после подтверждения текущего пароля. Сервер хеширует введённый текущий пароль и сравнивает с сохранённым. Это предотвращает несанкционированную смену пароля.

Для защиты от SQL-инъекций используются параметризованные запросы через библиотеку psycopg2. Значения передаются отдельно от SQL-команды через заполнители.

Настроена CORS-политика с ограничением допустимых источников и обработкой OPTIONS-запросов.

Реализован автоматический выход при обнаружении невалидного токена: токен удаляется из localStorage, сбрасываются данные пользователя и флаг администратора, после чего происходит перенаправление на страницу авторизации.

2.6 Оптимизация приложения

Для ускорения поиска билетов созданы индексы: по городам отправления и назначения, по датам (составной) и для сортировки по цене. Это исключает полное сканирование таблицы и ускоряет выполнение запросов.

При определении внешних ключей используется каскадное удаление (ON DELETE CASCADE), что автоматически удаляет связанные записи в заказах и занятых местах, сокращая объём кода и количество запросов.

Оптимизация запросов выполнена через JOIN, что позволяет получать данные из нескольких таблиц за одно обращение к БД, снижая время ответа и нагрузку на сеть.

Для работы с большими объёмами данных применяется пагинация, которая ограничивает количество записей за один запрос и сокращает объём передаваемых данных.

На сервере внедрено кэширование редко обновляемых данных (списки маршрутов, расписания), что снижает нагрузку на СУБД и ускоряет ответ в несколько раз.

На клиенте используется проверка состояния isLoading для блокировки повторной отправки форм, условный рендеринг компонентов и ленивая загрузка маршрутов. Кнопка отправки становится недоступной на время запроса, а её текст меняется на индикатор выполнения, что защищает от двойной отправки данных.

Для предотвращения утечек соединений с БД соединения корректно закрываются в блоке finally. Настроен пул соединений, поддерживающий ограниченное количество активных подключений.

Статические ресурсы сжимаются и кэшируются браузером, что ускоряет начальную загрузку приложения.

ЗАКЛЮЧЕНИЕ

Цель выпускной квалификационной работы — разработка информационной системы онлайн-бронирования автобусных билетов, оснащённой комплексной панелью административного управления, была достигнута.

Для достижения указанной цели были выполнены следующие задачи:

- проведен анализ предметной области и исследованы существующие аналогичные системы онлайн-бронирования, выявлены их преимущества и недостатки;
- выполнена постановка задачи и сформулированы функциональные требования к разрабатываемой системе;
- спроектирована архитектура веб-приложения, включая клиентскую часть и серверную логику;
- разработана структура базы данных для хранения информации о пользователях, маршрутах, рейсах, бронированиях и местах;
- созданы UML-диаграммы для описания логики работы системы;
- реализован модуль регистрации и авторизации пользователей с разграничением прав доступа;
- разработан функционал поиска и фильтрации рейсов по направлениям, датам и времени;
- создан интерфейс для просмотра доступных мест и выполнения бронирования билетов;
- реализован личный кабинет пользователя для просмотра истории бронирований и управления своим профилем;
- разработан интерфейс административной панели для управления системой;
- реализован функционал управления пользователями: просмотр списка, блокировка и разблокировка аккаунтов;
- создан модуль управления билетами и маршрутами: добавление, редактирование и удаление рейсов.