МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ РАБОТЫ АВТОСЕРВИСА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Бондаря Фёдора Алексеевича

Научный руководитель
профессор, к. ф.-м. н., д. экон. н. _____ Л. В. Кальянов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2025

ВВЕДЕНИЕ

Современные автосервисы представляют собой сложные системы, включающие управление записями на обслуживание, складом запчастей, расписанием работников и взаимодействием с клиентами. Ручное управление этими процессами приводит к ошибкам, задержкам и снижению удовлетворённости клиентов. Автоматизация с использованием веб-приложений позволяет оптимизировать процессы, минимизировать ошибки и повысить прозрачность, что делает разработку таких систем актуальной задачей.

Актуальность темы обусловлена необходимостью повышения эффективности работы автосервисов в условиях роста числа транспортных средств и клиентских ожиданий. Веб-приложения обеспечивают централизованный доступ к данным, автоматизацию рутинных операций и прозрачный расчёт стоимости услуг, что способствует улучшению качества обслуживания и конкурентоспособности бизнеса. Разработка приложения требует анализа предметной области, проектирования архитектуры и реализации, что представляет как научный, так и практический интерес.

Цель работы — разработка веб-приложения для автоматизации работы автосервиса, обеспечивающего управление записями, учётом запчастей, расписанием работников, взаимодействием с клиентами и расчётом предварительной стоимости обслуживания. Для достижения цели поставлены задачи:

- Провести анализ предметной области и сформулировать требования к приложению.
- Определить технологии и инструменты для разработки.
- Спроектировать архитектуру приложения, включая базу данных и интерфейсы.
- Реализовать функциональность приложения.
- Оценить эффективность приложения с точки зрения функциональности и удобства использования.

Материалы исследования включают научную и техническую литературу по веб-разработке, REST API, реляционным базам данных и UX-дизайну, а также анализ процессов автосервиса. Использованы технологии React, Node.js, Express, PostgreSQL, Sequelize, JWT, bcrypt, инструменты Webpack, Nodemon и Postman.

Структура работы состоит из введения, трёх основных глав, заклю-

чения, списка литературы и приложений. Первая глава посвящена анализу требований и выбору технологий. Вторая глава описывает проектирование архитектуры, базы данных, API и интерфейса. Третья глава содержит реализацию серверной и клиентской частей, а также демонстрацию работы приложения. В заключении представлены выводы и перспективы развития. Общий объём работы — 67 страниц, включая 12 рисунков, 20 источников и 2 приложения.

Краткое содержание работы

В первой главе проведён анализ предметной области и сформулированы требования к веб-приложению для автоматизации процессов автосервиса. Объект исследования — процессы управления автосервисом, включая создание записей, учёт запчастей, планирование работы персонала и взаимодействие с клиентами. Предмет исследования — веб-приложение, автоматизирующее указанные процессы с акцентом на прозрачность и удобство. Анализ выявил ключевые проблемы ручного управления: неэффективное планирование записей, отсутствие прозрачного учёта ресурсов, сложность координации расписания работников и недостаточная информация о стоимости услуг. На основе анализа сформулированы функциональные требования, включающие:

— Управление записями: создание, редактирование, отмена и просмотр записей с проверкой доступности временных слотов и предотвращением конфликтов расписания.

— Расчёт предварительной стоимости: предоставление клиентам прозрачной информации о стоимости услуг, запчастей и коэффициента загруженности сервиса.

— Учёт запчастей: регистрация закупок, отслеживание остатков, списание и возврат запчастей с проверкой совместимости с автомобилями.

— Управление расписанием работников: создание и редактирование графиков с учётом квалификации и доступности сотрудников.

— Администрирование пользователей: управление ролями (клиенты, администраторы, работники) с безопасной аутентификацией и разграничением доступа.

Обосновано использование веб-приложения как оптимального решения благодаря кроссплатформенности, доступности через браузер и лёгкости обновления. В сравнении с десктопными приложениями, ограниченными одной платформой, и мобильными, требующими отдельной разработки для iOS и Android, веб-приложение обеспечивает баланс между стоимостью, функциональностью и удобством. Для реализации выбраны технологии: React для клиентской части за счёт компонентного подхода и виртуального DOM, Node.js с фреймворком Express для серверной части благодаря асинхронной обработке, PostgreSQL с ORM Sequelize для базы данных за поддержку сложных реляционных структур и транзакций, а также JWT и bcrypt для аутен-

тификации и безопасности. Инструменты разработки включают Webpack для сборки клиентской части, Nodemon для автоматической перезагрузки сервера и Postman для тестирования API. Выбор технологий обоснован их производительностью, масштабируемостью, совместимостью и поддержкой сообщества.

Во второй главе описано проектирование приложения. Разработана клиент-серверная архитектура, где клиентская часть, реализованная на React, обеспечивает динамический интерфейс в виде одностраничного приложения (SPA), а серверная часть на Node.js и Express обрабатывает запросы, бизнес-логику и взаимодействие с базой данных PostgreSQL. Коммуникация осуществляется через REST API, использующий HTTP-запросы с JSON-форматом данных. Приложение разделено на модули для обеспечения модульности и упрощения поддержки:

- Клиентская часть: компоненты интерфейса (AdminPanel.js, AppointmentForm.js, CreateCar.js), маршруты через react-router-dom, управление состоянием с помощью Context API.
- Серверная часть: маршруты (adminRouter.js, appointmentRouter.js), контроллеры (adminController.js, appointmentController.js), middleware для аутентификации (authMiddleware.js) и проверки ролей (checkRoleMiddleware.js).
- База данных: модели (models.js) для сущностей Users, Appointments, Purchases, Workers и транзакции через Sequelize.

Спроектирована реляционная база данных на PostgreSQL, включающая таблицы Users (пользователи), Appointments (записи), Purchases (запчасти), Workers (работники), Services (услуги), WorkerSchedules (расписание), Cars (автомобили), CarBrands и CarModels (марки и модели). Таблицы связаны через внешние ключи, обеспечивая целостность данных: связи один-ко-многим (Users ? Appointments, Workers ? WorkerSchedules) и многие-ко-многим (Appointments ? Purchases через AppointmentPurchases). ER-диаграмма, созданная в dbdiagram.io, иллюстрирует структуру и связи. Транзакции Sequelize применяются для операций, затрагивающих несколько таблиц, таких как создание записи (проверка запчастей, списание, создание записи) и отмена (возврат запчастей, обновление статуса), предотвращая несогласованность данных.

REST API спроектировано по принципам ресурсориентированности, стан-

дартизации HTTP-методов и без сохранения состояния. Ключевые эндпоинты включают:

- Для клиентов: POST /api/user/registration, POST /api/appointment, GET /api/car (регистрация, создание записи, список автомобилей).
- Для администраторов: GET /api/admin/users, POST /api/admin/appointments, DELETE /api/admin/appointments/:id (управление пользователями, записями).

Аутентификация реализована через JWT, с middleware для проверки токенов и ролей. Обработка ошибок осуществляется через класс ApiError и ErrorHandlingMiddleware.js, возвращая стандартные HTTP-коды (400, 403, 500). Интерфейс разделён на административную панель (управление данными) и клиентский интерфейс (бронирование, управление автомобилями), разработанные с учётом UX-принципов: минимализм, прозрачность, интуитивная навигация. Админ-панель включает таблицы и формы для CRUD-операций, клиентский интерфейс — формы для записи и расчёта стоимости.

В третьей главе представлена реализация приложения. Серверная часть включает маршруты для обработки запросов, контроллеры для бизнес-логики и middleware для аутентификации и обработки ошибок. Реализованы функции:

- Создание записи (appointmentController.js): проверка доступности работников, услуг и запчастей, расчёт стоимости с учётом загруженности, создание записи в транзакции.
- Управление запчастями (adminController.js): регистрация, списание, возврат при отмене записи.
- Управление расписанием: назначение работников с учётом их графика и квалификации.

Клиентская часть построена на React с использованием react-bootstrap для стилизации и react-router-dom для навигации. Реализованы компоненты для аутентификации, регистрации, добавления автомобилей, создания записей и админ-панели. Управление состоянием осуществляется через Context API, взаимодействие с сервером — через HTTP-запросы к REST API. Обработка ошибок и логирование реализованы на обоих уровнях.

Демонстрация работы приложения включает описание ключевых интерфейсов:

— Начальная страница: приветственное сообщение и навигация.
— Страница аутентификации: формы для входа и регистрации.
— Форма добавления автомобиля: выбор марки, модели, ввод года и VIN.
— Форма записи на обслуживание: выбор автомобиля, услуг, даты, времени, отображение предварительной стоимости.
— Административная панель: таблицы записей, пользователей, запчастей с функциями создания, редактирования и удаления.

Функция расчёта предварительной стоимости интегрирует данные об услугах, запчастях и коэффициенте загруженности, обеспечивая прозрачность для клиентов. Функциональное и пользовательское тестирование подтвердило соответствие приложения требованиям, удобство интерфейса и надёжность операций, таких как создание записей и управление запасами.

Заключение

В результате выполненной работы разработано веб-приложение для автоматизации процессов автосервиса, обеспечивающее управление записями на обслуживание, учётом автомобилей и запчастей, планированием работы персонала и взаимодействием с клиентами. Разработка велась в соответствии с поставленной целью и задачами, что позволило создать функциональное решение, отвечающее потребностям современных автосервисов.

В рамках аналитической части проведён анализ предметной области, выявлены ключевые проблемы ручного управления процессами автосервиса, такие как неэффективное планирование записей, отсутствие прозрачного учёта ресурсов и сложность расчёта стоимости услуг. На основе анализа сформулированы требования к приложению, включающие управление записями, учёт запчастей, планирование расписания работников, администрирование пользователей и предоставление клиентам предварительной стоимости обслуживания. Эти требования легли в основу проектирования и реализации системы.

На этапе проектирования разработана архитектура приложения, включающая клиентскую и серверную части, а также структуру базы данных для хранения информации о пользователях, автомобилях, записях, услугах и запчастях. Серверная часть реализована с использованием набора программных модулей, обеспечивающих обработку запросов, аутентификацию и управление данными. Клиентская часть включает интерактивные интерфейсы, такие как формы аутентификации, добавления автомобилей, создания записей и административную панель, что обеспечивает удобство использования для клиентов и администраторов.

Особое внимание уделено функциональности расчёта предварительной стоимости обслуживания, которая позволяет клиентам получать прозрачную информацию о затратах до подтверждения записи. Эта функция реализована через интеграцию данных об услугах, запчастях и коэффициенте загруженности автосервиса, что повышает доверие пользователей и упрощает процесс бронирования. Демонстрация работы программы с помощью скриншотов ключевых интерфейсов подтвердила интуитивность и функциональность разработанного приложения. В ходе работы достигнуты все поставленные задачи: сформулированы требования, выбраны подходящие технологии, спро-

ектирована архитектура, реализованы обе части приложения, а также проведена демонстрация его работы. Разработанное приложение успешно решает проблему автоматизации процессов автосервиса, предоставляя функциональный и удобный инструмент для управления.

Основные источники информации:

1 Fielding R. T., Taylor R. N. Architectural Styles and the Design of Network-based Software Architectures. – Irvine: University of California, 2000. – 180 p. (дата обращения: 08.02.2025)

2 Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. 3rd ed. – San Francisco: No Starch Press, 2018. – 472 p. (дата обращения: 08.02.2025)

3 Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. 2nd ed. – Sebastopol: O'Reilly Media, 2020. – 310 p. (дата обращения: 10.02.2025)

4 React Bootstrap [Электронный ресурс] URL: https://react-bootstrap.netlify.app/ (дата обращения: 10.02.2025)

5 Single Page Application: как работает сайт-приложение [Электронный ресурс] URL: https://thecode.media/spa/ (дата обращения: 10.02.2025)

6 Node.js — Run JavaScript Everywhere [Электронный ресурс] URL: https://nodejs (дата обращения: 10.02.2025)

7 Express - Node.js web application framework [Электронный ресурс] URL: https://expressjs.com/ (дата обращения: 12.02.2025)

8 Руководство по PostgreSQL [Электронный ресурс] URL: https://metanit.com/sq (дата обращения: 15.02.2025)

9 Sequelize [Электронный ресурс] URL: https://sequelize.org/ (дата обращения: 19.02.2025)

10 JSON Web Token Introduction - jwt.io [Электронный ресурс] URL: https://jwt.io/ (дата обращения: 02.03.2025)

11 Postman: The World's Leading API Platform [Электронный ресурс] URL: https://www.postman.com/ (дата обращения: 12.03.2025)

12 Маршрутизация в React Router: как она работает и почему ее выбирают разработчики [Электронный ресурс] URL: https://ru.hexlet.io/blog/posts/react-router-v6 (дата обращения: 12.03.2025)

13 Middleware (Промежуточный слой) [Электронный ресурс] URL: https://fastapi.

(дата обращения: 15.03.2025)
14 Связи между таблицами базы данных [Электронный ресурс] URL: https://selec
for-databases/ (дата обращения: 15.03.2025)
15 Идеальный инструмент для работы с СУБД без SQL для Node.js [Электронный ресурс] URL: https://habr.com/ru/articles/567912/ (дата обращения: 21.03.2025)
16 REST API: что такое REST API (RESTful API)? [Электронный ресурс] URL: https://www.astera.com/ru/type/blog/rest-api-definition/ (дата обращения: 21.03.2025)
17 BCrypt [Электронный ресурс] URL: https://appmaster.io/ru/glossary/bcrypt-9 (дата обращения: 04.04.2025)
18 Основные принципы UX и UI дизайна [Электронный ресурс] URL: https://www
ux-dizayna-kotorye-ponravyatsya-vashim-polzovatelyam (дата обращения: 08.04.2025)
19 Bootstrap. Документация на русском языке [Электронный ресурс] URL: https://bootstrap-4.ru/ (дата обращения: 16.02.2025)
20 MobX tutorial [Электронный ресурс] URL: https://mobx.js.org/getting-started (дата обращения: 16.02.2025)