

Содержание

Введение.....	3
1 Техничко-экономическое обоснование.....	6
2 Техническое задание	8
3 Технический проект	11
3.1 Документация функциональной части	11
3.1.1 Описание постановки задачи	11
3.1.2 Описание функций	12
3.2 Документация обеспечивающей части.....	12
3.2.1 Информационное обеспечение	12
3.2.2 Техническое обеспечение.....	13
3.2.3 Программное обеспечение	13
4 Разработка веб-приложения	14
4.1. Создание шаблонов страниц веб-приложения	14
5 Тестирование веб-приложения	18
5.1 Ревьюирование программного кода.....	18
5.2 Тестирование кода веб-приложения	20
5.3 Публикация веб-приложения в сети	20
6 Документация к веб-приложению	22
6.1 Требования к аппаратным ресурсам	22
6.2 Руководство администратора	22
6.2.1 Назначение программного средства	23
6.2.2 Характеристики программного средства.....	23
6.2.3 Обращение к программе.....	23
6.2.4 Входные и выходные данные	24
6.2.5 Сообщения программисту.....	24
6.3.1 Руководство оператора	25
6.3.1.1 Назначение программного средства	25
6.3.1.2 Условия выполнения программы.....	25
6.3.1.3 Выполнение веб-приложения	25
Заключение	27
Список использованных источников	28
Приложение А	29
Приложение Б	30
Приложение В.....	31
Приложение Г	32
Приложение Д.....	34

					ОКЭИ 09.02.07. 1024. 18 ПЗ					
Изм.	Лист	№ докум.	Подпись	Дата						
Разраб.		Харламов И.С			Пояснительная записка			Лит.	Лист	Листов
Провер.		Юнаковская								2
Реценз.								Отделение информационных технологий, группа 4вб3		
Н. Контр.										
Утверд.										

Введение

Данная работа посвящена разработке CRM-системы для ООО «МИНИН СЕРВИС». Компания ООО «МИНИН СЕРВИС» занимается предоставлением услуг по ремонту и техническому обслуживанию оргтехники и бытовой техники. В своей деятельности компания сталкивается с необходимостью оптимизации и автоматизации внутренних рабочих процессов, а также улучшения взаимодействия с клиентами.

В рамках текущей деятельности ООО «МИНИН СЕРВИС» существует проблема эффективного управления заказами, гарантийными ремонтами от партнерских компаний, а также взаимодействия с клиентами через телефонные звонки. Эти задачи требуют значительных временных и человеческих ресурсов, что затрудняет поддержание высокого уровня обслуживания и оперативности выполнения работ. Автоматизация данных процессов с помощью CRM-системы позволит существенно улучшить качество обслуживания, сократить время на выполнение задач и минимизировать ошибки, связанные с человеческим фактором.

Задачи, решаемые CRM-системой, включают автоматизацию:

- обработки заказов, поступающих от клиентов и партнерских организаций;
- управления гарантийными ремонтами, обеспечивая быстрый и качественный сервис;
- прослушивания и анализа телефонных звонков для улучшения взаимодействия с клиентами;
- упрощение задач внутри компании для повышения эффективности работы сотрудников.
- просмотра и редактирования заказов, обеспечивая актуальность данных и оперативность обработки.

Одной из ключевых проблем, с которыми сталкивается ООО «МИНИН СЕРВИС», является большое количество заказов и разнообразие выполняемых работ. Каждый заказ включает в себя множество этапов — от первоначальной заявки клиента до окончательного выполнения работы и получения обратной связи. В условиях высокой загруженности персонала актуальность и точность данных имеют первостепенное значение для поддержания эффективного процесса обслуживания.

Основные сложности, возникающие при управлении заказами и взаимодействии с клиентами, включают:

- большое количество поступающих заказов и связанных с ними задач. В условиях высокой загруженности персонала критически важно своевременно обновлять информацию по каждому заказу, включая его статус, сроки выполнения и контактные данные клиентов;
- частое изменение информации о заказах. В процессе выполнения работ могут изменяться сроки, требования и другие параметры заказов, что требует оперативного внесения изменений в систему;

– необходимость прослушивания и анализа звонков с клиентами. Это важный аспект работы, так как позволяет выявлять проблемные моменты и улучшать качество обслуживания на основе полученной информации.

Проект разработки CRM-системы включает несколько этапов:

- анализ требований;
- проектирование;
- разработка;
- тестирование и внедрение.

На каждом этапе проводится тщательная проверка и корректировка, что позволяет обеспечить соответствие системы всем требованиям и ожиданиям компании.

Разработка CRM-системы для ООО «МИНИН СЕРВИС» является актуальной задачей, направленной на повышение эффективности работы компании и улучшение качества обслуживания клиентов. Внедрение системы позволит значительно сократить время на обработку заказов, улучшить взаимодействие с клиентами и обеспечить высокое качество предоставляемых услуг.

Для решения данной проблемы необходимо разработать CRM-систему, которая позволит автоматизировать процессы управления заказами и взаимодействия с клиентами, обеспечивая оперативное обновление информации и контроль её достоверности. Система должна быть гибкой, удобной в использовании, безопасной, и обеспечивать быстрый доступ к данным о заказах и клиентах.

В настоящее время администрирование осуществляется посредством ручной обработки данных и использования различных программных инструментов. Последовательность действий в общем виде такова:

- получение заказа от клиента или партнера;
- внесение данных о заказе в систему 1С;
- обработка заказа, включающая подтверждение, выполнение работы и обновление статуса;
- взаимодействие с клиентом через телефонные звонки для уточнения деталей и получения обратной связи;
- завершение заказа и внесение соответствующих данных в систему.

Такой подход к управлению заказами имеет ряд недостатков:

- необходимы глубокие знания различных программных инструментов и понимание структуры данных, что может быть сложно для неподготовленных пользователей;
- ручное внесение данных увеличивает риск ошибок, которые могут привести к задержкам или неверной обработке заказов;
- без надлежащей защиты данных существует риск несанкционированного доступа, что может привести к утечке информации или нежелательным изменениям;
- дополнительные временные затраты, связанные с тестированием каждого изменения, обнаружением и исправлением ошибок.

В связи с наличием данных недостатков необходимо создать программный продукт, который обеспечит более эффективное управление заказами и взаимодействие с клиентами. Основные этапы работы с CRM-системой будут включать:

- авторизация. Сотрудник должен войти в систему, используя предоставленные логин и пароль;
- добавление заказов. Менеджер может добавить новый заказ, заполнив необходимые поля и указав все детали;
- редактирование заказов. Менеджер может изменять существующие заказы, обновляя информацию о статусе, сроках и других параметрах;
- сохранение изменений. После внесения всех необходимых изменений пользователь должен сохранить их, чтобы они вступили в силу.

Цель данной работы — разработать CRM-систему, автоматизирующую и модернизирующую управление заказами и взаимодействие с клиентами в ООО «МИНИН СЕРВИС».

Для достижения данной цели были поставлены следующие задачи:

- произвести анализ предметной области;
- обосновать необходимость создания или модификации CRM-системы;
- разработать техническое задание;
- разработать проектную документацию;
- разработать CRM-систему согласно техническому заданию;
- разработать базу данных для CRM-системы;
- объединить новую систему с имеющимися системами;
- произвести оптимизацию и ревьюирование программного кода;
- произвести публикацию CRM-системы в сети;
- произвести отладку и тестирование CRM-системы;
- разработать документацию для пользователей: руководство оператора и руководство системного администратора;
- произвести расчет экономической эффективности разработанной CRM-системы.

1 Технико-экономическое обоснование

Разработка CRM-системы для ООО «МИНИН СЕРВИС» является актуальной задачей, поскольку текущий процесс администрирования рабочих задач и взаимодействия с клиентами осуществляется вручную и требует значительных временных затрат, что увеличивает вероятность ошибок. Внедрение CRM-системы с админ-панелью позволит упростить и автоматизировать эти процессы, делая их более удобными и эффективными. Кроме того, разработка такой системы соответствует глобальным трендам цифровизации и автоматизации бизнес-процессов, что повышает конкурентоспособность компании на рынке.

ООО «МИНИН СЕРВИС» занимается предоставлением услуг по ремонту и техническому обслуживанию оргтехники и бытовой техники. В структуру компании входят различные подразделения, занимающиеся выполнением заказов, гарантийными ремонтами и взаимодействием с клиентами через телефонные звонки и электронную почту. Основные функции компании включают:

- прием и обработку заказов на ремонт и обслуживание техники. Это включает в себя регистрацию заказов, назначение мастеров и контроль за выполнением работ;
- управление гарантийными ремонтами, поступающими от компаний-партнеров. Это позволяет обеспечить своевременное и качественное выполнение гарантийных обязательств;
- учет выполненных работ и потребленных ресурсов, что необходимо для корректного выставления счетов клиентам и контроля за расходами компании;
- взаимодействие с клиентами через телефонные звонки и электронную почту для уточнения деталей заказов, получения обратной связи и решения возникающих проблем;
- прием и обработка обращений и жалоб клиентов, обеспечивая оперативное решение возникающих вопросов и улучшение качества обслуживания;
- предоставление технической поддержки клиентам, включая консультации и помощь в решении проблем, связанных с ремонтом и обслуживанием техники.

Основные технико-экономические показатели деятельности ООО «МИНИН СЕРВИС» за 2023 год:

- выручка: общий объем продаж товаров и услуг компании составил 1 485 000 рублей, что отражает эффективность работы компании на рынке;
- прибыль: чистая прибыль компании составила 899 000 рублей, что свидетельствует о высокой рентабельности деятельности;
- количество направлений деятельности: компания специализируется на двух основных направлениях — ремонте и техническом обслуживании оргтехники и бытовой техники;
- себестоимость продаж: общие затраты на производство и реализацию продукции составили 648 000 рублей;
- валовая прибыль: составила 837 000 рублей, что является разницей между выручкой и себестоимостью продаж;

					ОКЭИ 09.02.07. 1024. 18 ПЗ	Лист
	Лист	№ докум	Подпись	Дата		6

– общая сумма поступлений от текущих операций: все поступления от продажи товаров, выполнения работ и оказания услуг составили 827 000 рублей.

Создание CRM-системы позволит оптимизировать рабочие процессы, улучшить качество обслуживания клиентов и повысить общую эффективность деятельности компании. Автоматизация задач по управлению заказами, взаимодействию с клиентами и обработке обращений сократит временные затраты и минимизирует риск ошибок, что приведет к улучшению финансовых показателей и укреплению позиций компании на рынке.

Деятельность ООО «МИНИН СЕРВИС» охватывает широкий спектр услуг, направленных на обеспечение качественного ремонта и обслуживания оргтехники и бытовой техники для различных категорий клиентов. Основные направления деятельности включают предоставление услуг по ремонту и техническому обслуживанию для частных клиентов, компаний и организаций. Компания активно работает над улучшением качества обслуживания и внедрением современных технологий для повышения эффективности работы и удовлетворения потребностей клиентов.

Основными пользователями CRM-системы являются клиенты, мастера, менеджеры — сотрудники компании. Клиенты, включающие как физические, так и юридические лица, могут просматривать информацию о заказах, отслеживать статус выполнения работ, оставлять отзывы и получать поддержку. Менеджеры, сотрудники компании, отвечают за управление заказами, взаимодействие с клиентами и поддержание актуальности данных в системе. Они могут добавлять, удалять и редактировать информацию о заказах и клиентах.

Введение CRM-системы позволит ООО «МИНИН СЕРВИС» значительно повысить эффективность управления заказами и улучшить качество обслуживания клиентов. Автоматизация рабочих процессов, интеграция с корпоративным сайтом и возможность гибкого управления филиалами обеспечат высокую производительность и конкурентоспособность компании на рынке.

2 Техническое задание

Техническое задание является основным документом, создаваемым на стадии разработки CRM-системы для ООО «МИНИН СЕРВИС». Этот документ отражает требования, предъявляемые пользователями-заказчиками к разрабатываемой системе, и представляет собой описание совокупности характеристик, которым должна удовлетворять создаваемая система. Также техническое задание включает макеты интерфейсов, которые будут реализованы в CRM-системе. Техническое задание отвечает на вопрос, каким должно быть создаваемое приложение и что должно дать его внедрение для улучшения работы предприятия.

В техническом задании приводится полное наименование и условное обозначение CRM-системы, документы, на основании которых ведется разработка, оформленные в приложении. Указываются основные требования к графическому материалу, используемому в системе, сроки начала и окончания работ по созданию CRM-системы. Обосновывается очередность проектирования системы.

Основные цели создания CRM-системы включают:

- автоматизация процессов управления заказами и взаимодействия с клиентами;
- снижение временных затрат на администрирование и выполнение стандартных задач;
- улучшение качества обслуживания клиентов;
- обеспечение безопасности и целостности данных.

Перечень функций, необходимых для достижения поставленных целей, включает:

- авторизация пользователей в системе;
- добавление, редактирование и удаление данных о заказах;
- управление информацией о клиентах;
- взаимодействие с клиентами через систему (отправка уведомлений, получение обратной связи);
- учет выполненных работ и потребленных ресурсов;
- генерация отчетов и аналитики по заказам и клиентам;
- интеграция с корпоративным сайтом и другими внутренними системами компании.

Для каждой функции создаются и описываются основные макеты интерфейсов CRM-системы, которые определяют требования к дизайну и удобству использования. Макеты должны быть представлены в виде иллюстраций, созданных в графических редакторах, и включают:

- главную страницу системы;
- страницу авторизации;
- интерфейс для управления заказами;
- интерфейс для управления данными о клиентах;
- страницу отчетности и аналитики.

Критерии эффективности функционирования CRM-системы после публикации в сети включают:

- скорость обработки и отображения данных;
- удобство и интуитивность интерфейса для конечных пользователей;
- надежность и устойчивость системы к ошибкам и сбоям;
- безопасность данных и защита от несанкционированного доступа.

Требования к визуальной части:

– адаптивность: система должна корректно отображаться и функционировать на всех устройствах, включая компьютеры, планшеты и смартфоны.

– удобство использования: интерфейс должен быть интуитивно понятным, позволяющим пользователям без специальных знаний быстро освоить основные функции.

– интуитивно понятный интерфейс: все элементы управления должны быть легко распознаваемыми и доступными.

– минималистичный дизайн: интерфейс должен быть чистым и не перегруженным деталями.

– основной цвет интерфейса: белый, для создания чистого и профессионального вида.

Выделение интерактивных элементов: все кликабельные элементы, такие как кнопки и ссылки, должны быть визуально выделены.

Цвет кнопок: синий, что позволяет четко выделять их на фоне основного белого цвета.

Функции системы представлены в виде функциональной модели, построенной с использованием case-средств проектирования.

Эта модель описывает взаимодействие пользователей с системой и последовательность выполнения задач.

Стадии и этапы разработки CRM-системы включают:

1 предпроектное исследование:

- анализ предметной области;
- сбор и систематизация требований пользователей.

2 проектирование:

- разработка технического задания;
- создание архитектуры системы;
- разработка макетов интерфейсов.

3 разработка:

- программирование основных модулей системы;
- интеграция с внешними системами и базой данных;
- тестирование и отладка.

4 внедрение:

- установка и настройка системы на серверах компании;
- обучение пользователей и администраторов.

5 поддержка и сопровождение:

- регулярное обновление системы;
- устранение выявленных ошибок;
- добавление новых функций по мере необходимости.

Выполнение всех этапов разработки и внедрения CRM-системы обеспечит достижение поставленных целей, улучшит эффективность управления заказами и взаимодействие с клиентами, а также повысит общую конкурентоспособность ООО «МИНИН СЕРВИС».

					ОКЭИ 09.02.07. 1024. 18 ПЗ	Лист
						10
	Лист	№ докум	Подпись	Дата		

3 Технический проект

3.1 Документация функциональной части

3.1.1 Описание постановки задачи

Назначение задачи заключается в создании CRM-системы для ООО «МИНИН СЕРВИС», автоматизирующей рабочие процессы внутри компании и обеспечивающей эффективное взаимодействие с клиентами. Основные функции включают управление заказами, гарантийными ремонтами, прослушивание звонков и изменения заказов клиентов. Веб-приложение должно поддерживать полный цикл работы с заказами, начиная с их поступления и заканчивая выполнением и закрытием заказов.

Цепочка действий от момента поступления информации до получения выходных данных:

- ввод данных о заказах, ремонтных работах и взаимодействиях с клиентами через формы интерфейса;
- обработка введенной информации с последующим обновлением базы данных;
- генерация отчетов и аналитики на основе обработанных данных.

Перечень информации, поступающей в веб-приложение:

- данные о клиентах (ФИО, контактная информация, история заказов);
- данные о заказах (описание проблемы, дата поступления, статус выполнения);
- данные о гарантийных ремонтных работах (партнеры, детали ремонта, сроки);
- данные о звонках (дата и время, содержание разговора).

Перечень и описание выходных сообщений и экранных форм веб-приложения:

- отчет о статусе заказов с фильтрацией по различным параметрам;
- аналитические отчеты по выполненным работам и их эффективности;
- экранные формы с детальной информацией по каждому заказу и клиенту;
- уведомления о статусе заказа для клиентов и сотрудников.

Ввод информации осуществляется через интуитивно понятные формы, созданные для различных типов данных (заказы, клиенты, ремонты).

Вывод информации осуществляется через отчетные и аналитические формы, с возможностью экспорта в различные форматы (PDF, Excel).

CRM-система использует стандартные алгоритмы для расчетов показателей эффективности работы, таких как время выполнения заказов, процент выполненных заказов в срок и т.д.

Последовательность шагов выполнения расчётов:

- получение исходных данных из базы данных;
- выполнение расчетов на серверной стороне;
- отображение результатов пользователю.

Веб-приложение состоит из нескольких модулей, таких как модуль управления заказами, модуль взаимодействия с клиентами и модуль аналитики. Эти модули взаимодействуют между собой и с базой данных для обеспечения полной функциональности системы.

3.1.2 Описание функций

Перечень функций веб-приложения:

- управление заказами;
- управление клиентами;
- управление гарантийными ремонтами;
- управление взаимодействиями с клиентами;
- аналитика и отчеты.

Для каждого режима работы системы указано:

- назначение режима: управление заказами, клиентами, ремонтами, взаимодействиями и аналитикой;
- результат работы режима: выходная информация в виде отчетов и обновленных данных в системе;
- входная информация: данные о клиентах, заказах, звонках и ремонтах, поступающие из различных источников;
- процесс выполнения режима: система обрабатывает введенную информацию, обновляет базу данных и генерирует отчеты.

В приложении приводится схема, отображающая работу каждого модуля веб-приложения, включая их взаимодействие с базой данных и друг с другом. Схема также включает функциональную модель системы.

3.2 Документация обеспечивающей части

3.2.1 Информационное обеспечение

Входные документы: заявки на ремонт, контактная информация клиентов, данные о звонках и гарантийных ремонтах.

Структура базы данных: включает таблицы клиентов, заказов, ремонтов, звонков и отчетов. Каждая таблица имеет свои столбцы, описывающие соответствующие данные.

ER-диаграмма: в приложении приводятся концептуальная, логическая и физическая модели данных.

Выходные данные:

- отчеты по заказам и ремонтам;
- аналитические данные для управленческих решений;
- формы экранных документов, описывающие текущие статусы заказов и клиентов.

3.2.2 Техническое обеспечение

Выбор хостинга и технических средств обоснован следующими параметрами:

- производительность;
- надежность;
- масштабируемость.

Основные характеристики:

- выделенный сервер с 8 ядрами и 32 Гб оперативной памяти;
- SSD-накопитель для быстрого доступа к данным;
- обеспечение резервного копирования данных.

3.2.3 Программное обеспечение

Инструменты и средства разработки включают:

- проектирование дизайна: Figma для создания макетов и прототипов.
- язык программирования: JavaScript (React.js для фронтенда) и Node.js для бэкенда.
- язык разметки: HTML5 и CSS3 для оформления интерфейсов.
- библиотеки: Redux для управления состоянием приложения, Axios для работы с API.
- среда разработки: Visual Studio Code.
- СУБД: PostgreSQL для хранения данных.
- веб-сервер: Nginx для управления HTTP-запросами.

Тестирование: Jest и Cypress для автоматизированного тестирования функциональности веб-приложения.

4 Разработка веб-приложения

4.1. Создание шаблонов страниц веб-приложения

Во время реализации разработки веб-приложения, важно учитывать все пункты, которые были описаны ранее. Важный акцент при разработке делается на удобстве и безопасности работы веб-приложения.

Страница для поиска заказа реализована на фронтенд части фрейворка React JS, который по мимо отображения данных на странице у пользователя делает запросы на сервер, созданный на базе NodeJS и языке ExpressJS. Серверная часть (бекенд) принимает запрос API и отправляет его на внешний сервер для получения данных, благодаря реализации системы микро серверной архитектуре. Данная система позволяет отлавливать потенциальные ошибки во время обращений между серверами, и «разгружать» систему, что позволяет обрабатывать большее количество запросов сразу.

Страница для авторизации пользователя отправляет хешированные данные на сервер, который сверяет данные с базы данных, в случае успешной проверки дает разрешение на вход пользователя (рисунок 1).

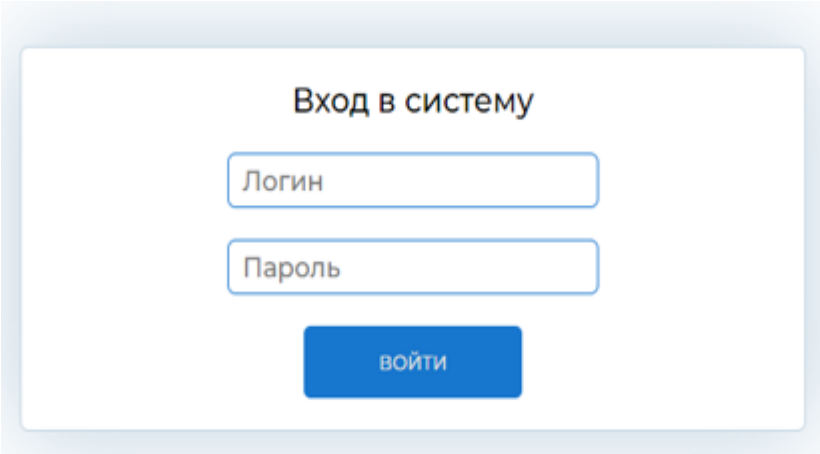


Рисунок 1 – Страница авторизации пользователя.

Перед отправкой данных на сервер важно убедиться, что все обязательные поля формы заполнены данными в корректном формате. Это называется валидацией на стороне клиента и помогает убедиться, что данные, введенные в каждый элемент формы, соответствуют требованиям.

Валидация на стороне клиента — это первичная проверка введенных данных, которая существенно улучшает удобство взаимодействия с интерфейсом; обнаружение некорректных данных на стороне клиента позволяет пользователю немедленно их исправить. Если же проверка происходит только на сервере, процесс заполнения может быть более трудоемким, так как требует повторения одних и тех же действий отправки данных на сервер для получения обратного ответа с сообщением о том, что нужно исправить.

Во всем проекте созданная проверка на валидность полей, система не даст продолжить работу с формами, если одно из необходимых полей не заполнена, и выдаст ошибку подсветив пол.

Валидация – это процесс проверки данных на соответствие различным критериям. При разработке любого приложения в большинстве случаев разработчику приходится иметь дело с обработкой данных, которые ввел пользователь в соответствующие поля на форме. По разным причинам пользователь может вводить некорректные данные.

Чем раньше интерфейс сообщает об ошибке, тем лучше — пользователю проще вернуться и исправить ошибку.

Самый быстрый способ сообщить об ошибке — мгновенная валидация. Но она возможна только в тех случаях, когда в процессе ввода понятно, что значение некорректное. Обычно такие ошибки связаны с неправильной раскладкой клавиатуры (кириллица вместо латиницы) или вводом букв в цифровое поле (ИНН, КПП и др.)

Проверка происходит после того, как пользователь нажал кнопку отправки данных: все поля с ошибками на форме подсвечиваются, страница прокручивается к первому полю с ошибкой, фокус перемещается в это поле, курсор встает в конец строки, рядом с полем появляется тултип с подсказкой.

Красный текст ошибки появляется сразу, как только произошла валидация и ошибочное поле подсветилось.

Текст ошибки пропадает по потере фокуса и больше не появляется, если поле заново получает фокус. Это правило одинаково работает для всех типов валидаций: и по потере фокуса, и при отправке формы. Код реализации формы авторизации на React JS представлен на рисунке 4.

Пример реализации системы валидации представлен на рисунке 3.

Авторизация

32

Пароль

ВОЙТИ

Рисунок 3 – Пример валидации полей формы.

```

import React, { useEffect, useState } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { loginThunk } from '../redux/authSlice';
import { useNavigate } from 'react-router-dom';
import './login.css';

const Log = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [errorMessage, setErrorMessage] = useState('');
  const [invalidFields, setInvalidFields] = useState([]);

  const authState = useSelector((state) => state.auth);
  const dispatch = useDispatch();
  const nav = useNavigate();

  useEffect(() => {
    setErrorMessage(authState.error || '');
  }, [authState.error]);

  const handleLogin = () => {
    if (!username.trim() || !password.trim()) {
      setInvalidFields([username.trim() ? '' : 'username', password.trim() ? '' : 'password']);
      return;
    }

    dispatch(loginThunk({ username: username, password: password }))
      .then(() => {
        nav('/');
      });
  };

  const handleInputChange = (field, value) => {
    if (invalidFields.includes(field)) {
      setInvalidFields(invalidFields.filter(item => item !== field));
    }

    if (field === 'username') {
      setUsername(value);
    } else if (field === 'password') {
      setPassword(value);
    }
  };

  return (
    <div className="container-login">
      <h1>Авторизация</h1>
      <input
        className={invalidFields.includes('username') ? 'invalid' : ''}
        value={username}
        onChange={(e) => handleInputChange('username', e.target.value)}
        type="text"
        placeholder="Логин"
      />
      <input
        className={invalidFields.includes('password') ? 'invalid' : ''}
        value={password}
        onChange={(e) => handleInputChange('password', e.target.value)}
        type="password"
        placeholder="Пароль"
      />
      {errorMessage && <p className="error-message">{errorMessage}</p>}
      <button onClick={handleLogin}>Войти</button>
    </div>
  );
};

export default Log;

```

Рисунок 4 – Код авторизации

Когда пользователь занимается работой с заказами клиентов, это является одним из ключевых аспектов его работы. Отслеживание статуса выполнения ремонта позволяет ему оперативно реагировать на изменения и информировать клиентов о текущем положении дел. Проверка наличия необходимых запчастей также играет важную роль, поскольку это позволяет избежать задержек в процессе ремонта и обеспечить его бесперебойность.

С целью облегчения выполнения этих задач была разработана специальная страница, предназначенная для просмотра информации о заказах. Эта страница предоставляет пользователю удобный интерфейс для быстрого доступа к данным заказов. Пользователь может легко найти нужный заказ, введя его номер в соответствующее поле поиска.

После ввода номера заказа на странице отображаются все имеющиеся данные по данному заказу.

Это включает в себя информацию о клиенте, описание проблемы, список необходимых запчастей, статус выполнения работ и другие детали заказа.

Пользователь может ознакомиться с этой информацией и при необходимости произвести изменения в данных. Пример страницы поиска заказа представлен на рисунке 5.

5 Тестирование веб-приложения

5.1 Ревьюирование программного кода

При ревьюировании программы был проанализирован следующий фрагмент программного кода, представленный на рисунке 7.

```
import React from 'react';
import axios from 'axios';

class UserList extends React.Component {
  state = {
    users: [],
  };

  componentDidMount() {
    axios.get('/api/users')
      .then(response => {
        this.setState({ users: response.data });
      });
  }

  render() {
    return (
      <div>
        <h1>User List</h1>
        <ul>
          {this.state.users.map(user => (
            <li key={user.id}>{user.name}</li>
          ))}
        </ul>
      </div>
    );
  }
}

export default UserList;
```

Рисунок 7 - Фрагмент кода до ревьюирования

Сначала проведена проверка структуры кода React JS - оценивалось правильное использование компонентов, жизненных циклов и состояний React. Далее производилась проверка безопасности запросов к бэкенду на Express JS - в данном случае, запросы должны быть защищены от атак типа CSRF или XSS. После этого проводился анализ SQL-запросов к базе данных PostgreSQL - оценивалась безопасность и эффективность запросов. Далее проверялось соответствие стандартам именования, комментариев, а также наличие обработки возможных ошибок.

					ОКЭИ 09.02.07. 1024. 18 ПЗ	Лист
						18
	Лист	№ докум	Подпись	Дата		

В результате ревьюирования программный код был приведен к следующему виду, представленный на рисунке 8.

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

const UserList = () => {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    axios.get('/api/users')
      .then(response => {
        setUsers(response.data);
      })
      .catch(error => {
        console.error('Error fetching users:', error);
      });
  }, []);

  return (
    <div>
      <h1>User List</h1>
      <ul>
        {users.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>
    </div>
  );
};

export default UserList;
```

Рисунок 8 - Фрагмент кода после ревьюирования

В результате ревьюирования были внесены следующие изменения:

- использование функционального компонента вместо классового для упрощения кода;
- использование хука `useState` и `useEffect` для управления состоянием и выполнения побочных эффектов;
- добавлена обработка ошибок при запросе данных с сервера для повышения надежности приложения.

Таким образом, процесс ревьюирования помог улучшить структуру и безопасность кода, а также сделал его более современным и поддерживаемым

					ОКЭИ 09.02.07. 1024. 18 ПЗ	Лист
						19
	Лист	№ докум	Подпись	Дата		

5.2 Тестирование кода веб-приложения

Когда подходим к тестированию кода веб-приложения, мы принимаем во внимание все аспекты, начиная от его функциональности до безопасности и производительности. В нашем подходе к тестированию мы комбинируем несколько методов и используем различные инструменты для того, чтобы обеспечить высокое качество и надежность приложения.

Ручное тестирование играет ключевую роль в нашем процессе. Этот метод позволяет нашим тестировщикам активно взаимодействовать с приложением, воспроизводя реальные сценарии использования и проверяя его работу на практике. Мы проводим тесты на различных этапах разработки, начиная с проверки базовой функциональности и заканчивая тщательным тестированием каждого нового функционала перед его выпуском. В ходе ручного тестирования мы проверяем не только основные функции, но и реакцию приложения на нестандартные ситуации, такие как некорректный ввод данных или неправильное использование функций.

Помимо ручного тестирования, мы также применяем метод модульного тестирования. Этот метод позволяет нам проверить отдельные компоненты приложения на корректность их работы. Мы разрабатываем тестовые сценарии для каждого модуля, проверяя их функциональность в изоляции от других частей приложения. Это помогает нам выявить и устранить потенциальные проблемы на ранних стадиях разработки, что уменьшает вероятность ошибок и повышает общее качество кода.

Кроме того, мы активно используем инструменты для автоматизации тестирования, такие как Jest и Selenium. Эти инструменты позволяют нам автоматизировать выполнение тестовых сценариев, что упрощает процесс тестирования и сокращает время, затраченное на проверку функциональности приложения. Мы также используем инструменты для анализа кода и выявления потенциальных проблем, такие как ESLint и SonarQube.

5.3 Публикация веб-приложения в сети

Для успешного развертывания веб-приложения «МИНИН СЕРВИС» в локальной сети компании, используя контейнеры Docker, необходимо выполнить ряд важных шагов с учетом доступа, защиты и обновлений:

- подготовка сервера и настройка Docker: начните с установки и настройки Docker на сервере, где будет размещено веб-приложение. Убедитесь, что сервер доступен только ограниченному числу сотрудников с соответствующими правами доступа;
- создание Docker-образа приложения: разработаем Dockerfile для веб-приложения, включая все зависимости и конфигурации. Гарантируем безопасность путем установки необходимых обновлений и патчей на уровне ОС и контейнеров;

					ОКЭИ 09.02.07. 1024. 18 ПЗ	Лист
						20
Лист	№ докум	Подпись	Дата			

– запуск контейнера с приложением: запуск контейнера с веб-приложением на сервере. Убедимся, что доступ к серверу защищен с помощью пароля или других методов аутентификации;

– проверка работоспособности: после запуска контейнера тщательно проверьте работу приложения, включая функциональность, производительность и безопасность. Протестируйте доступ к приложению из различных сетей и устройств;

– обеспечение безопасности и доступа: реализуем механизмы аутентификации и авторизации, чтобы обеспечить доступ к приложению только авторизованным пользователям. Регулярно необходимо анализировать журналы доступа для выявления и предотвращения возможных угроз безопасности;

– обновление и обслуживание: регулярно обновляем веб-приложение и контейнеры Docker, включая патчи безопасности и новые версии приложения. Планируем регулярные процедуры обслуживания для поддержания стабильной и безопасной работы приложения;

– масштабирование и управление ресурсами: используем возможности масштабирования Docker для обработки увеличенной нагрузки. Разработаем стратегии управления ресурсами для эффективного использования вычислительных мощностей и оптимизации производительности приложения.

Развертывание веб-приложения с использованием контейнеров Docker обеспечивает высокий уровень безопасности, доступности и гибкости. С соблюдением всех указанных шагов ваше приложение будет готово к продуктивной и надежной работе в локальной сети компании.

6 Документация к веб-приложению

6.1 Требования к аппаратным ресурсам

При развертывании серверной части веб-приложения, требуется учитывать минимальные аппаратные требования, обеспечивающие стабильную и эффективную работу программного обеспечения. Организации, занимающиеся разработкой и внедрением веб-приложений, должны быть внимательны к этим требованиям, чтобы обеспечить оптимальную производительность и надежность приложения.

С учетом разнообразия технологий и платформ, используемых для разработки веб-приложений, включая React JS, Express JS и PostgreSQL, определены следующие минимальные требования к аппаратным ресурсам:

- процессор: рекомендуется использовать процессор Intel Core i3 или аналогичный с поддержкой архитектуры x86_64 и частотой не менее 2.4 ГГц. Это обеспечит достаточную вычислительную мощность для обработки запросов и выполнения бизнес-логики приложения;
- объем оперативной памяти (ОЗУ): Минимальный объем оперативной памяти должен составлять 4 Гбайт или выше. Это позволит эффективно управлять памятью при обработке больших объемов данных и одновременном обслуживании множества пользователей;
- жесткий диск: для установки серверного ПО и хранения данных рекомендуется выделить не менее 10 Гбайт на жестком диске. Это обеспечит достаточное пространство для хранения программных файлов и базы данных;
- монитор и видеокарта: для развертывания серверной части веб-приложения требуется монитор с поддержкой разрешения не менее 1024x768 пикселей и видеокарта, обеспечивающая SVGA качество изображения. Это обеспечит комфортное взаимодействие с интерфейсом приложения;
- устройства ввода: для управления серверной частью веб-приложения рекомендуется использовать стандартные устройства ввода, такие как клавиатура и мышь. Они обеспечат удобство работы с приложением и взаимодействие с его интерфейсом.

Внимательное соблюдение этих требований к аппаратным ресурсам позволит обеспечить стабильную и эффективную работу серверной части веб-приложения на практике.

6.2 Руководство администратора

Одним из ключевых аспектов технического сопровождения является регулярное резервное копирование данных. Создание и регулярное обновление резервных.

					ОКЭИ 09.02.07. 1024. 18 ПЗ	Лист
	Лист	№ докум	Подпись	Дата		22

В данном разделе представлено руководство администратора для CRM-системы компании «БытService». Это руководство содержит необходимую информацию для управления и поддержки CRM-системы.

6.2.1 Назначение программного средства

CRM-система компании «БытService» разработана для управления клиентскими отношениями, автоматизации бизнес-процессов и повышения эффективности работы компании. Она предназначена для использования сотрудниками компании для учета клиентов, обработки запросов, управления заказами и предоставления качественного обслуживания клиентов.

6.2.2 Характеристики программного средства

Этот раздел содержит основные характеристики CRM-системы «БытService», необходимые для успешного запуска и функционирования.

CRM-система «БытService» разработана с использованием современных технологий веб-разработки:

- Фронтенд: веб-интерфейс CRM-системы разработан с использованием фреймворка React JS, обеспечивающего динамичный и удобный пользовательский интерфейс;
- Бэкенд: серверная часть реализована с использованием фреймворка Express JS для обработки запросов и взаимодействия с базой данных;
- База данных: для хранения данных используется PostgreSQL, обеспечивающая надежное хранение информации о клиентах, заказах и других бизнес-данных.

6.2.3 Обращение к программе

Этот подраздел предоставляет инструкции по запуску CRM-системы «БытService».

Локальный запуск: для локального запуска CRM-системы «БытService» необходимо перейти в директорию с программным кодом и выполнить команду запуска основного файла с помощью интерпретатора языка программирования. После успешного запуска система будет доступна по адресу, указанному в выводе команды.

Удаленный доступ: для удаленного доступа к CRM-системе «БытService» необходимо [указать способы доступа к приложению, например, через веб-браузер по URL или через специализированный клиентский софт]. Пользователи могут получить доступ к системе, введя адрес и учетные данные, если они требуются.

Эти инструкции обеспечивают администраторам простой и понятный способ запуска CRM-системы «БытService», как локально, так и удаленно, и помогают обеспечить эффективное использование системы в рабочей среде. Пример запуска проекта в локальной сети представлен на рисунке 9.

```

PS C:\Users\User\Desktop\CRM-Service> npm run dev

> crm-service@0.0.0 dev
> vite --host --port 24

VITE v5.0.11 ready in 1371 ms

→ Local:   https://localhost:24/

```

Рисунок 9— запуск проекта в локальной сети

6.2.4 Входные и выходные данные

В данном подразделе описываются входные и выходные данные CRM-системы компании «БытService».

Входными данными являются информация о технических устройствах, а также справочные данные, необходимые для работы системы.

К ним относятся:

- список ВТ (в том числе информация о каждом устройстве);
- документы о перемещении ВТ между подразделениями или сотрудниками;
- заявки на проверку технических устройств;
- справочник кабинетов, содержащий информацию о местоположении устройств;
- справочник типов ВТ с описанием и характеристиками каждого типа устройств;
- справочник состояний технических устройств.

Выходными данными являются результаты обработки входных данных и формирование отчетов.

К ним относятся:

- реестр приоритетных устройств, который содержит информацию о наиболее важных устройствах для компании;
- отчет о состоянии технических устройств, включающий в себя информацию о текущем состоянии устройств, проведенных проверках, ремонтах и т. д.

6.2.5 Сообщения программисту

В данном разделе описываются сообщения CRM-системы «БытService», предназначенные для программиста.

В системе «БытService» специальных сообщений для программиста не предусмотрено. Однако, в случае возникновения системных ошибок или иных проблем, могут появляться стандартные системные сообщения, которые могут содержать информацию о нарушениях целостности данных в базе данных или о неожиданных ситуациях при выполнении операций.

Для облегчения поддержки и развития системы в исходном коде предусмотрено множество комментариев, адресованных программисту. Эти комментарии содержат объяснения структуры и логики программы, что помогает

разработчику лучше понять код, а также упрощает внесение изменений и добавление нового функционала.

6.3 Руководство пользователя

6.3.1 Руководство оператора

6.3.1.1 Назначение программного средства

Разработчик: веб-приложение «БытService» предоставляет разработчику доступ к средствам управления и настройки приложения. Разработчик использует его для добавления новых функций, исправления ошибок, оптимизации производительности и обеспечения безопасности приложения.

Клиент: для клиента, CRM-Система представляет собой средство подачи заявок на обслуживание, отслеживания их статуса и взаимодействия с сервисным персоналом. Клиенты могут управлять своими запросами и получать информацию о выполненных работах через это приложение.

Руководитель: руководитель использует CRM-Систему для мониторинга работы сервисного отдела, просмотра отчетов о выполненных работах, контроля за сроками и качеством обслуживания клиентов.

6.3.1.2 Условия выполнения программы

Для запуска CRM-Системы требуется только устройство с доступом в интернет и веб-браузером. Это может быть персональный компьютер, ноутбук, смартфон или планшет.

Аппаратные требования ограничиваются наличием монитора с поддержкой SVGA видеокарты, клавиатуры и мыши. Это минимальные требования, обеспечивающие базовое взаимодействие с приложением.

Эти условия делают CRM-Систему доступным и удобным в использовании для всех операторов, независимо от их технического уровня и доступности аппаратного обеспечения.

6.3.1.3 Выполнение веб-приложения

В данном разделе представлено описание работы веб-приложения для конкретной роли пользователя в CRM-системе «БытService».

Если формат корректен, приложение позволяет отправить данные на сервер и (обычно) сохранить в базу данных. После ввода адреса на экране появится главная страница, которая также служит окном входа, представленная на рисунке 10.

Рисунок 10 – страница авторизации

В начале процесса проверки авторизации и валидации полей входа пользователь вводит свое имя пользователя и пароль. Эти данные проверяются на соответствие в базе данных, где хранится информация о пользователях. Первым этапом является проверка наличия введенного имени пользователя в системе. Если пользователь с таким именем существует в базе данных, происходит переход к следующему шагу.

На следующем этапе происходит проверка корректности введенного пароля. Обычно пароли хранятся в базе данных в зашифрованном виде, чаще всего в виде хэш-значений. Таким образом, введенный пользователем пароль преобразуется в хэш и сравнивается с сохраненным хэшем в базе данных для соответствующего пользователя. Если хэши совпадают, это означает, что введенный пароль верен.

Если пользователь вводит неверные учетные данные, система выведет соответствующее сообщение об ошибке, сообщение при неудачной авторизации представлена на рисунке 11.

Рисунок 11 – ошибка авторизации

Если введенный пароль совпадает с сохраненным в базе данных и учетная запись активна, то пользователь успешно проходит аутентификацию и считается аутентифицированным. Это позволяет ему получить доступ к защищенным ресурсам или функциональности системы.

Заключение

Веб-приложение «БытService» было создано с целью предоставления клиентам удобного и эффективного инструмента для решения проблем с их техникой. В ходе производственной практики были выполнены работы по анализу предметной области, созданию технического задания, разработке проектной документации, включая макеты и эскизы веб-приложения, а также реализации шаблонов и динамических элементов приложения.

Это веб-приложение представляет собой интегрированную платформу, позволяющую пользователям легко оформлять заказы на обслуживание и ремонт различных устройств, от компьютеров до мобильных телефонов. Одним из ключевых достоинств приложения является его интуитивно понятный интерфейс и возможность быстрой обработки заказов.

В дальнейшем планируется реализация дополнительных функциональностей, таких как система онлайн-консультаций, возможность отслеживания статуса заказа в реальном времени, а также интеграция с платежными системами для удобства оплаты услуг.

Однако успех веб-приложения не зависит только от его функциональности, но и от качества его технического сопровождения. Регулярное обновление, мониторинг производительности и безопасности, а также оперативная поддержка пользователей играют важную роль в обеспечении непрерывной и стабильной работы приложения.

«БытService» представляет собой высококачественное веб-приложение, способное удовлетворить потребности пользователей и достичь успеха на рынке. Его создание и развитие являются результатом тщательного анализа и проектирования, а также усилий целой команды профессионалов, стремящихся обеспечить лучший опыт для своих клиентов.

Список использованных источников

1 ГОСТ 7.32-2001 «Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления» - устанавливает правила оформления отчетов о научно-исследовательской работе, включая структуру работы, формат страниц, оформление заголовков, списков литературы и прочее.

2 ГОСТ 7.0.5-2008 «Библиографическая ссылка. Общие требования и правила составления» - определяет правила составления библиографических ссылок на использованные источники информации.

3 ГОСТ 2.105-95 «Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам» - содержит общие требования к текстовой документации, включая размеры и форматы бумаги, оформление текста, заголовков, таблиц, чертежей и др.

4 ГОСТ 7.1-2011 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления» - устанавливает правила библиографического описания документов.

5 ГОСТ 19.104-78 «Единая система программной документации. Общие требования к программным документам» - содержит требования к оформлению программной документации, если ваша работа включает программные коды, схемы алгоритмов и прочее.

6 ГОСТ 7.9-95 «Система стандартов автоматизированных систем управления. Библиографическая запись. Библиографическое описание» - определяет правила библиографического описания документов для автоматизированных систем управления.

7 Шарпи Ф. А. и др. (2022). «ASP.NET Core 6 для профессионалов». БХВ-Петербург.

8 Кузнецов А. С. (2023). «Разработка веб-приложений с использованием React: от начинающего до профессионала». Питер.

9 Документация Microsoft. «Документация по ASP.NET Web API».

10 Документация React. «React - библиотека JavaScript для создания пользовательских интерфейсов».

11 Троцкий И. П. (2021). «Микросервисы на платформе ASP.NET Core: проектирование, тестирование и развертывание». Питер.

12 Холмс Д. (2020). «Построение микросервисов с использованием ASP.NET Core: разработка, тестирование и развертывание кросс-платформенных служб в облаке». Питер.

13 Эспозито Д. (2021). «Программирование в ASP.NET Core (Справочник разработчика)». Microsoft Press.

14 Эллиот Дж., & Джексон Н. (2023). «Программирование микросервисов: создание надежного и масштабируемого программного обеспечения». Питер.

15 Elliott, J., & Jackson, N. (2020). «Programming Microservices: Build Robust and Scalable Software.» O'Reilly Medi

Приложение А (обязательное)

UML-диаграммы

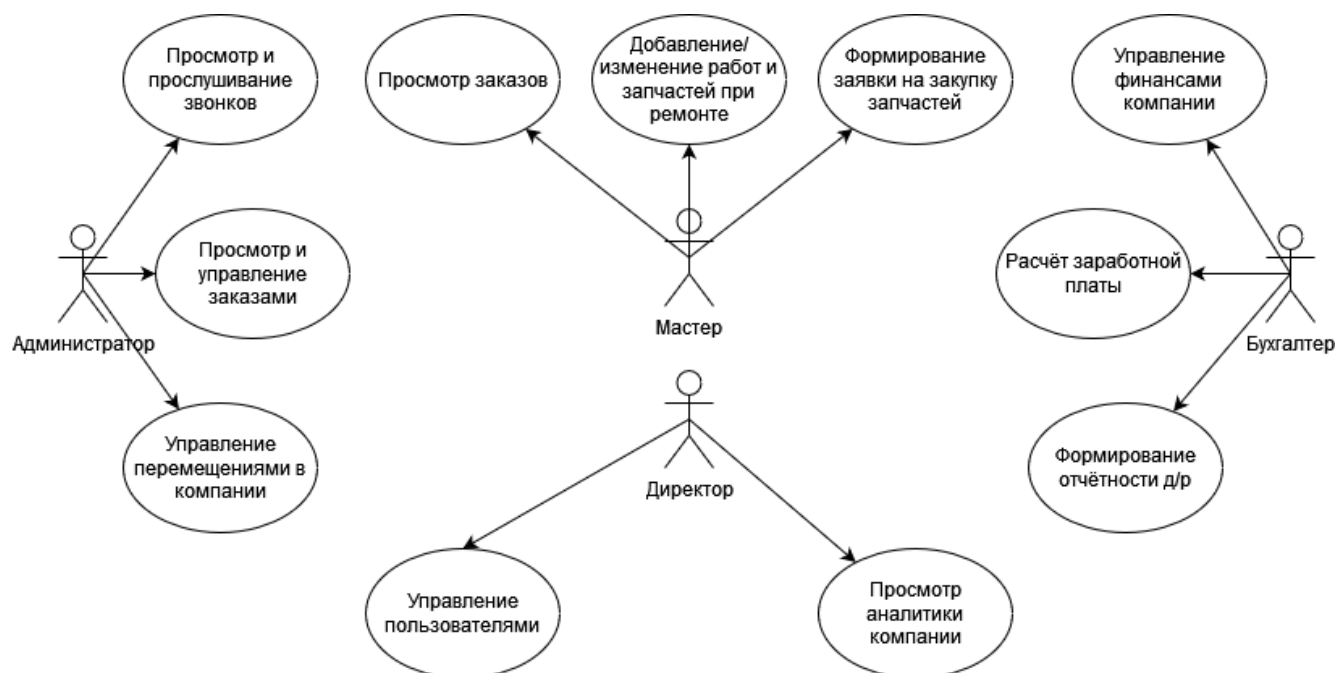


Рисунок А.1 - Диаграмма вариантов использования

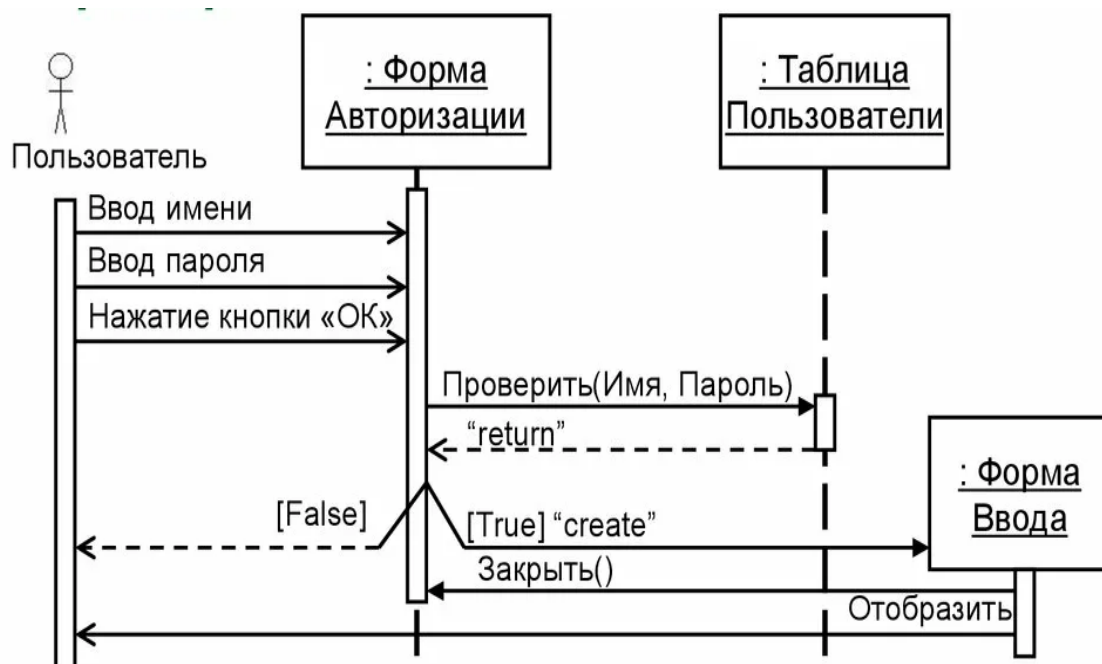


Рисунок А.2 – Диаграмма последовательности авторизации

Приложение Б
(обязательное)

Информационная модель



Рисунок Б1 – Диаграмма нотации DFD

Приложение В
(обязательное)
ER-диаграмма

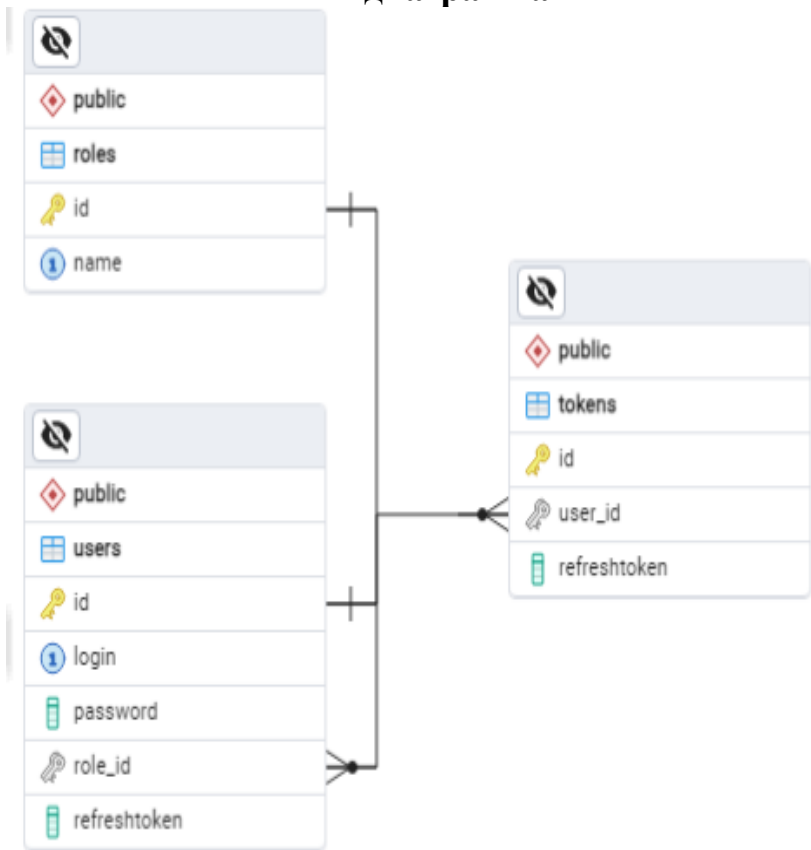


Рисунок В1 - ER-диаграмма базы данных

Приложение Г (обязательное)

SQL-скрипты

```
//создаем таблицы
await sql`create table if not exists Roles(
  role varchar(100) unique primary key
)`
await sql`create table if not exists Users(
  id SERIAL PRIMARY KEY NOT NULL,
  name varchar(100) NOT NULL,
  role varchar(100),
  password varchar(100),
  FOREIGN KEY (role) REFERENCES Roles(role)
)`

//запустить в первый раз и больше не запускать
//чтобы добавить роли в таблицу ролей

// await sql`insert into Roles(role) values('USER')`
// await sql`insert into Roles(role) values('ADMIN')`
// Контроллер получения списка пользователей
export const getUsers = async (req, res) => {
  try {
    // Выполнение SQL-запроса для получения списка пользователей и их
ролей
    const users = await sql`SELECT name, role FROM public.users ORDER BY id
ASC`;

    // Отправка списка пользователей в ответ на запрос
    res.json(users);
  } catch (error) {
    console.error('Ошибка при получении списка пользователей:', error);
    // Отправка сообщения об ошибке в случае неудачи
    res.status(500).json({ error: 'Ошибка сервера' });
  }
};

export const auth = async (req, res) =>{
  const {username, password} = req.body
  const user = await sql`select * from Users where name = ${username}`
  if(!user[0]){
    return res.status(400).json({message: `Пользователь ${username} не
найден` })
  }
```

```

    }
    const validPassword = bcrypt.compareSync(password, user[0].password)
    if(!validPassword){
        return res.status(400).json({ message: `Неверный пароль` })
    }
    const token = generateAccessToken(user[0].id, user[0].role)
    return res.json({
        token: token,
        user: user[0]
    })
}
// Контроллер регистрации
export const register = async (req, res) => {
    const { username, password, roleName } = req.body;
    console.log('Полученные данные:', { username, password, roleName });
    try {
        console.log('Имя пользователя:', username);
        console.log('Пароль:', password);
        console.log('Роль:', roleName);

        const candidate = await sql`SELECT * FROM Users WHERE name =
${username} LIMIT 1`;

        if (candidate) {
            return res.status(400).send("Пользователь уже существует");
        }

        const hashPassword = bcrypt.hashSync(password, 7);

        const userRole = await sql`SELECT role FROM Roles WHERE role =
${roleName} LIMIT 1`;
        if (!userRole || userRole.length === 0) {
            return res.status(400).send("Роль не найдена");
        }
        // Передаем только значение роли, а не объект целиком
        await sql`INSERT INTO Users(name, role, password) VALUES (${username},
${userRole[0].role}, ${hashPassword})`;

        return res.json({ message: "Пользователь успешно зарегистрирован" });
    } catch (error) {
        console.error('Ошибка при регистрации:', error);
        return res.status(500).json({ error: 'Ошибка сервера' });
    }
};

```


Приложение Д (обязательное)

Код реализации веб-приложения

```
import React, { useState, useEffect } from 'react';
import Header from '../Header';
import Messenger from './messenger/Messenger';

function Maxvi() {
  const [isLoading, setIsLoading] = useState(false);
  const [isItsLoading, setIsItsLoading] = useState(false);
  const [records, setRecords] = useState("");
  const [searchValue, setSearchValue] = useState("");
  const [selectedDefect, setSelectedDefect] = useState("");
  const [defectsList, setDefectsList] = useState([]);
  const [selectedPart, setSelectedPart] = useState("");
  const [partsList, setPartsList] = useState([]);
  const [worksList, setWorksList] = useState([]);
  const [manualPartName, setManualPartName] = useState("");
  const [manualPartPrice, setManualPartPrice] = useState("");
  const [selectedWork, setSelectedWork] = useState("");
  const [aktData, setAktData] = useState({
    device_defect_parts: [],
    device_defect_causes: [],
    device_location_after: "",
    act_issuing_reason: ""
  });
  const [partsData, setPartsData] = useState(null);
  const [selectedDefectPart, setSelectedDefectPart] = useState("");
  const [selectedDefectCause, setSelectedDefectCause] = useState("");
  const [selectedLocationAfter, setSelectedLocationAfter] = useState("");
  const [selectedIssuing, setSelectedIssuing] = useState("");
  const [showManualDefectCause, setShowManualDefectCause] = useState(false);
  const [showManualDefectParts, setShowManualDefectParts] = useState(true);
  const [combinedPartsData, setCombinedPartsData] = useState(null);
  const [manualDefectCause, setManualDefectCause] = useState("");
  const [manualDefectCauseParts, setManualDefectCauseParts] = useState("");
  const [manualDefectPartsReason, setManualDefectPartsReason] = useState("");
  const [actIssuingReasonDescription, setActIssuingReasonDescription] = useState(false);
  const [deviceErrors, setDeviceErrors] = useState({
    device_id: "",
    device_full_model: "",
    device_type: "",
    device_brand: "",
    device_model: "",
    device_sn: "",
    device_imei: "",
    device_appearance: "",
    device_equipment: ""
  });
```

```

    });
    const [isSaveEnabled, setIsSaveEnabled] = useState(false);
    const [allFieldsSelected, setAllFieldsSelected] = useState(false);
    const [showModal, setShowModal] = useState(false);
    const [showModalParts, setShowModalParts] = useState(false);

    useEffect(() => {
        if (selectedIssuing && selectedDefectPart && selectedDefectCause &&
selectedLocationAfter) {
            setAllFieldsSelected(true);
        } else {
            setAllFieldsSelected(false);
        }
    }, [selectedIssuing, selectedDefectPart, selectedDefectCause, selectedLocationAfter]);

    const handleManualDefectCauseChange = (e) => {
        const value = e.target.value;
        setManualDefectCause(value);
    };

    const handleSelectedIssuingChange = (e) => {
        const value = e.target.value;
        setManualDefectCauseParts(value);
    };

    const handleManualDefectPartsChange = (e) => {
        const value = e.target.value;
        setManualDefectPartsReason(value);
    };

    const handleSelectedDefectPartChange = (e) => {
        const value = e.target.value;
        setSelectedDefectPart(value);
    };

    const handleSelectedDefectCauseChange = (e) => {
        const value = e.target.value;
        setSelectedDefectCause(value);
    };

    const handleSelectedLocationAfterChange = (e) => {
        const value = e.target.value;
        setSelectedLocationAfter(value);
    };

```