|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Государственное бюджетное профессиональное образовательное учреждение Новосибирской области  «Новосибирский химико-технологический колледж им.  Д.И. Менделеева»  (ГБПОУ НСО «Новосибирский химико-технологический колледж им.  Д.И. Менделеева»)   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | КУРСОВАЯ РАБОТА | | | | | | | | | | (форма – *курсовая работа*) | | | | | | | | | |  | | | | | | | | | |  | | | | | | | | | |  | | | | | | | | | | на тему: | | Разработка веб-приложения "Система управления | | | | | | | | задачами". | | | | | | | |  | |  | | | | | | | |  | | *наименование темы курсовой работы* | | | | | | | | Специальность: | | *Информационные системы и программирование* | | | | | | | |  | | *шифр и наименование специальности* | | | | | | | | Выполнил(а) | | | | | | | | | | студент(ка) | группы | *09.07.31* |  | *4* | курса | *Очной* | | формы обучения | |  |  |  |  |  |  | *очной, заочной* | |  | |  | | |  | | |  | *Ромасько-Николенко Роман Евгеньевич* | | |  | | | *подпись* | | |  | *фамилия, имя, отчество* | | | Руководитель курсовой работы | | |  | | |  | *Воробьев Алексей Анатольевич* | | | *подпись* | | |  | *фамилия, инициалы* | | | «Допущен(а) к защите» | | |  | | |  |  | | | *Председатель ПЦК* | | |  | | |  | *Чубрикова Ольга Викторовна* | | |  | | | *подпись* | | |  | *фамилия, инициалы* | | | Зав. отделением | | |  | | |  | *Бондаренко Ольга Анатольевна* | | |  | | | *подпись* | | |  | *фамилия, инициалы* | | |  | | | | | | | | | | Новосибирск 2024 | | | | | | | | | |

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 4

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc184688291)

[1.1 Актуальность работы 5](#_Toc184688292)

[1.2 Требования к приложению 5](#_Toc184688292)

[1.3 Стек технологий 7](#_Toc184688293)

[1.4 Выводы по первому разделу 8](#_Toc184688294)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 10](#_Toc184688295)

[2.1 Проектирование архитектуры веб-приложения 10](#_Toc184688296)

[2.2 Проектирование макета интерфейса системы 10](#_Toc184688297)

[2.3 Выводы по второму разделу 13](#_Toc184688298)

[ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ 14](#_Toc184688299)

[3.1 Разработка базы данных 14](#_Toc184688300)

[3.2 Разработка бэкенд-сервисов веб-приложения. 14](#_Toc184688301)

[3.3 Разработка фронтенд-сервиса веб-приложения. 16](#_Toc184688302)

[3.4 Описание кода 17](#_Toc184688303)

[3.5 Выводы по третьему разделу 21](#_Toc184688305)

[ЗАКЛЮЧЕНИЕ 22](#_Toc184688306)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc184688307)

ВЕДЕНИЕ

В современном мире эффективное управление задачами является важным аспектом личной и профессиональной жизни. Постоянный поток информации и множество обязательств могут создать ощущение хаоса, что затрудняет выполнение целей и задач. В условиях такого перегруза важность правильно организованного подхода к задачам становится неоспоримой.

Разработка системы управления задачами направлена на упрощение процесса планирования и выполнения повседневных задач. Целью является создание удобного инструмента, который позволит пользователям эффективно структурировать свои задачи, управлять ими и поддерживать порядок в повседневной жизни. Система предлагает простой интерфейс для добавления, редактирования и удаления задач, что позволяет быстро реагировать на изменения и приоритеты.

Кроме того, приложение обеспечивает возможность пользователям работать с задачами на разных уровнях, начиная от личных дел до задач, связанных с командной работой. Внимание уделено безопасности хранения данных, что гарантирует сохранность информации в процессе использования платформы.

Таким образом, система управления задачами призвана помочь пользователям не только организовывать свои обязательства, но и повысить эффективность работы, минимизируя риск забывания важных задач и улучшая организацию рабочего процесса.

# ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. Актуальность работы

Разработка веб-приложений в последние годы стала важным аспектом в сфере информационных технологий. Особое место занимают системы управления задачами, которые являются важным инструментом как для индивидуального, так и для командного выполнения проектов. Учитывая рост числа удалённых и гибридных команд, наличие удобных и функциональных систем управления задачами становится особенно актуальным.

Система управления задачами позволяет оптимизировать рабочий процесс, улучшить взаимодействие между участниками команды и повысить производительность. В рамках данной курсовой работы разрабатывается веб-приложение, которое будет включать функции регистрации пользователей, создания, обновления и удаления задач, а также административного управления.

* 1. Требования к приложению

Приложение "Система управления задачами" должно удовлетворять следующим требованиям, чтобы эффективно выполнять свою задачу и быть удобным в использовании как для конечных пользователей, так и для администраторов системы.

1. Функциональные требования:

*Регистрация и аутентификация пользователей:* Система должна обеспечивать возможность регистрации новых пользователей, а также входа в систему через авторизацию с использованием имени пользователя и пароля. Для безопасности данных будет использоваться хеширование паролей с помощью библиотеки bcrypt.js, а для аутентификации — JWT-токены.

*Управление задачами:* Пользователи должны иметь возможность создавать новые задачи, редактировать их. Для этого будет реализована соответствующая форма и API.

*Разделение пользователей по ролям:* Для реализации контроля доступа необходимо разделить пользователей на несколько ролей, таких как "администратор" и "пользователь". Администратор должен иметь расширенные права, такие как возможность удаления задач пользователей и их самих, и просмотра всех задач в системе.

*Отображение задач:* Для каждого пользователя будет предусмотрена страница с перечнем всех его задач.

*Административный интерфейс:* Администратор должен иметь возможность просматривать все зарегистрированные задачи и управлять пользователями, а также их задачами.

1. Нефункциональные требования:

*Производительность:* Приложение должно работать быстро и без задержек, даже при большом количестве пользователей и задач. Важно обеспечить оптимизацию работы с базой данных.

*Масштабируемость:* Веб-приложение должно быть масштабируемым, то есть, при необходимости, должно легко поддерживать увеличение количества пользователей и данных без значительного ухудшения производительности.

*Безопасность:* Все данные, передаваемые между клиентом и сервером, должны быть зашифрованы, для чего будет использоваться HTTPS. Также предусмотрены механизмы защиты от SQL-инъекций, XSS-атак и других уязвимостей.

*Удобство интерфейса:* Интерфейс приложения должен быть простым и интуитивно понятным. Все основные действия, такие как создание задачи или ее изменение, должны быть доступны в несколько кликов.

* 1. Стек технологий

Выбор технологий для разработки веб-приложения "Система управления задачами" обусловлен необходимостью обеспечения высокого уровня безопасности, производительности и масштабируемости. В ходе разработки был выбран следующий стек технологий:

**- Node.js** — это JavaScript-исполнительная среда, которая позволяет запускать код JavaScript вне браузера. Node.js является основой серверной части веб-приложения. Он использует событийно-ориентированную архитектуру, что позволяет обрабатывать множество запросов одновременно, делая приложение высокоэффективным и масштабируемым. Node.js идеально подходит для создания быстрых и масштабируемых веб-сервисов.

**- Express.js** — это минималистичный веб-фреймворк для Node.js, который упрощает создание API, управление маршрутами, а также обработку запросов и ответов. Express позволяет сосредоточиться на логике приложения, минимизируя необходимость в написании boilerplate-кода.

**- Sequelize** — это ORM (Object-Relational Mapping) для Node.js, которая упрощает работу с реляционными базами данных. Sequelize позволяет использовать объектно-ориентированные подходы для взаимодействия с базой данных, что значительно ускоряет разработку. Он поддерживает различные базы данных, включая PostgreSQL, MySQL и SQLite, но для данного проекта будет использована PostgreSQL, благодаря её стабильности и производительности.

**- PostgreSQL** — это реляционная база данных с открытым исходным кодом. PostgreSQL известна своей надежностью, поддержкой сложных типов данных и транзакционной целостностью. Для данного проекта PostgreSQL будет использоваться для хранения данных о пользователях, задачах и ролях.

**- HTML, CSS, JavaScript (Frontend)** — фронтенд-приложение будет построено с использованием стандартных технологий веб-разработки: HTML для разметки, CSS для стилизации и JavaScript для взаимодействия с сервером. Для улучшения пользовательского опыта на клиентской стороне будет использован чистый JavaScript и AJAX-запросы через fetch API.

**- JWT (JSON Web Token)** — это стандарт, который используется для безопасной передачи информации между клиентом и сервером в виде токенов. В проекте JWT будет использоваться для аутентификации пользователей. Токены позволят пользователям оставаться авторизованными в течение сессии без необходимости каждый раз вводить логин и пароль.

**- Bcrypt.js** — это библиотека для хеширования паролей, которая будет использоваться для обеспечения безопасности данных пользователей. Каждый пароль, введенный пользователем при регистрации, будет хешироваться перед сохранением в базе данных, что защитит его от утечек данных.

Этот стек технологий был выбран с учетом современных требований к разработке безопасных, высокопроизводительных и масштабируемых веб-приложений. Он обеспечивает необходимые инструменты для эффективной работы и защиты данных пользователей, а также позволяет создавать интуитивно понятный и удобный интерфейс.

* 1. Выводы по первому разделу

В первом разделе курсовой работы был проведен анализ предметной области и определены основные требования, которым должно соответствовать разрабатываемое веб-приложение. Система управления задачами является важным инструментом для эффективной организации работы как индивидуальных пользователей, так и целых команд. В условиях растущего спроса на удаленную работу и гибридные формы сотрудничества, эффективное управление задачами становится необходимостью.

Также был выбран стек технологий, который обеспечит надежную, безопасную и масштабируемую платформу для реализации системы управления задачами. Node.js с Express.js предоставляет мощные возможности для серверной разработки, в то время как Sequelize и PostgreSQL позволят эффективно работать с данными. Для аутентификации пользователей и защиты данных выбран JWT, что гарантирует безопасность при работе с приложением. В свою очередь, использование Bcrypt.js обеспечит защиту паролей пользователей и повысит общую безопасность системы.

Таким образом, в первом разделе был обоснован выбор технологий и сформулированы ключевые требования, которые легли в основу дальнейшей разработки приложения. Эти требования направлены на создание удобного, безопасного и функционального инструмента для управления задачами, который будет востребован в условиях современного рабочего процесса.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

* 1. Проектирование архитектуры веб-приложения

Архитектура системы управления задачами будет построена по принципу клиент-сервер. Серверная часть будет реализована с использованием Node.js и Express.js, которые обеспечат обработку запросов и взаимодействие с базой данных через ORM Sequelize.

Веб-приложение будет разделено на две основные части:

**-** Frontend — интерфейс пользователя, который позволит регистрировать и аутентифицировать пользователей, а также просматривать, создавать и обновлять задачи.

- Backend — серверная логика, включая API для работы с задачами и пользователями, а также системы аутентификации и авторизации.

* 1. Проектирование макета интерфейса системы

Интерфейс системы будет простым и интуитивно понятным. Он будет включать страницы для регистрации и входа пользователя, страницу для работы с задачами, а также панель админа.

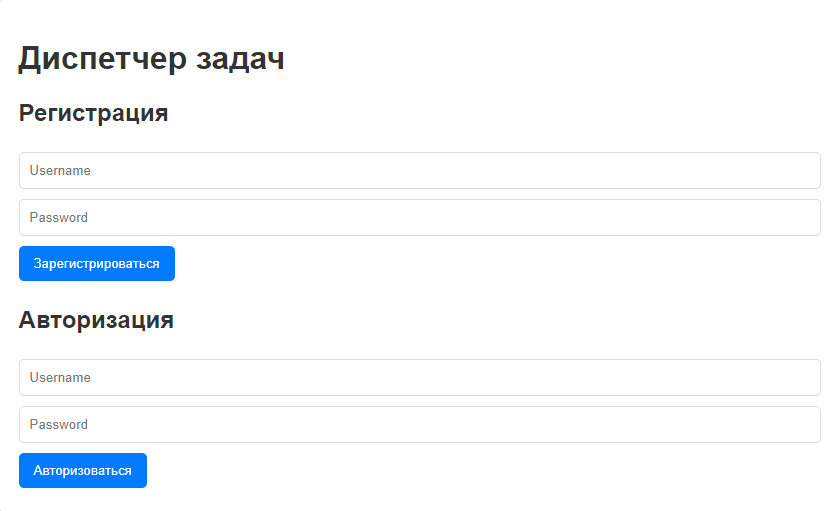


Рисунок 1 – «Интерфейс главной страницы веб-приложения»

Для каждого пользователя будет доступен личный кабинет с возможностью создания, редактирования и удаления задач.

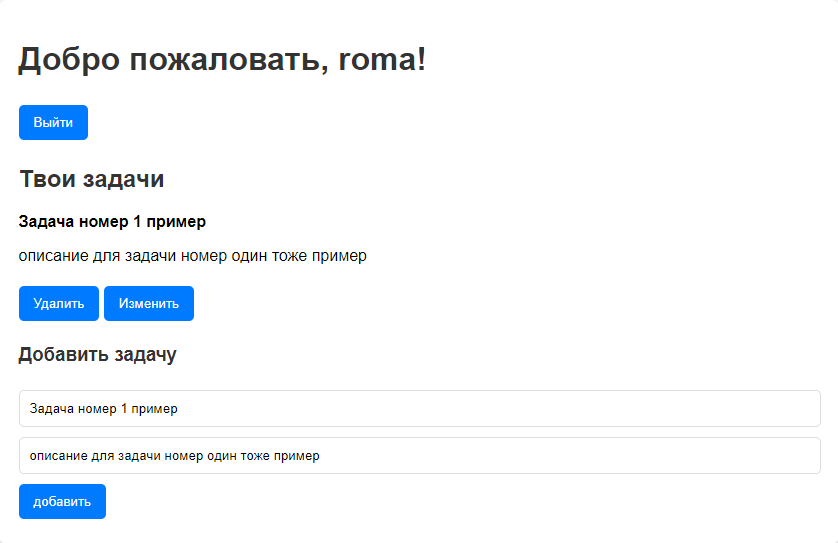


Рисунок 2 – «Интерфейс страницы пользователя»

Этот скриншот демонстрирует интерфейс пользователя после авторизации, который был спроектирован с акцентом на минимализм и простоту использования. Такой подход был выбран для обеспечения удобства взаимодействия с приложением и уменьшения перегрузки визуальной информацией.

К примеру, кнопки "Удалить" и "Изменить" позволяют быстро взаимодействовать с задачами, в то время как форма для добавления новых задач четко выделяется и интуитивно понятна. Кнопка выхода из системы ("Выйти") размещена в верхней части, чтобы быстро завершить сессию.

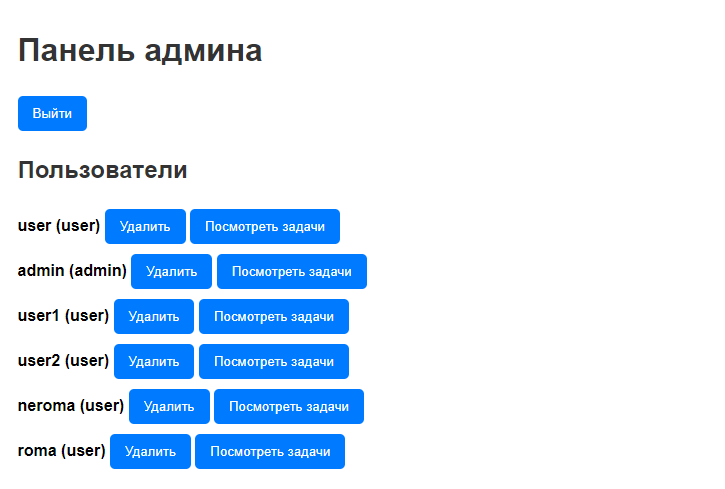


Рисунок 3 – «Интерфейс панели администратора»

**Панель администратора** имеет простой и понятный интерфейс с набором функций, таких как просмотр и удаление задач пользователей, и их самих. Все ключевые операции расположены так, чтобы они были доступны с нескольких кликов. Это позволяет администраторам быстро выполнять свои обязанности, не тратя время на поиск нужных опций.

* 1. Выводы по второму разделу

В результате проектирования системы было уделено особое внимание созданию интуитивно понятного и удобного интерфейса как для конечных пользователей, так и для администраторов. Выбор минималистичного дизайна помогает сократить визуальное беспорядочное пространство и ускоряет процесс взаимодействия с приложением, что особенно важно для таких систем, где пользователи и администраторы должны выполнять задачи быстро и эффективно.

Проектирование архитектуры веб-приложения и интерфейса было направлено на создание гибкой и масштабируемой системы, которая может быть легко адаптирована под различные требования. Макет интерфейса, ориентированный на простоту и функциональность, создает четкую визуальную иерархию, что позволяет избежать путаницы у пользователей и ускоряет выполнение их действий.

# ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

* 1. Разработка базы данных

Для функционирования системы была разработана база данных, которая служит основой для хранения всех данных о пользователях, задачах и их статусах. Система использует реляционную модель базы данных с использованием SQL, что позволяет упростить взаимодействие с данными.

База данных содержит несколько ключевых таблиц, таких как Users и Tasks. Каждая таблица была спроектирована с учетом требований к быстродействию и масштабируемости, что позволяет эффективно хранить и извлекать данные. Важно отметить, что связаны они через уникальные идентификаторы, что гарантирует целостность данных. Например, каждая задача в таблице Tasks ассоциирована с конкретным пользователем из таблицы Users.

Кроме того, для обеспечения безопасности и контроля доступа была внедрена система аутентификации и авторизации. Пароли пользователей хранятся в зашифрованном виде с использованием алгоритма bcrypt, что минимизирует риски утечек личных данных. Применение JWT (JSON Web Tokens) в процессе авторизации также играет важную роль, гарантируя защиту данных и надежность сессий пользователей.

* 1. Разработка сервисов веб-приложения.

Разработка бэкенд-сервисов веб-приложения является основой для функционирования всей системы. В данном проекте бэкенд-сервис отвечает за обработку запросов от клиента, выполнение бизнес-логики, взаимодействие с базой данных и возвращение нужной информации пользователям. В качестве технологий для бэкенда выбраны Node.js и Express, которые предоставляют гибкие инструменты для построения RESTful API.

Основным компонентом бэкенда является **REST API**, который состоит из нескольких маршрутов, отвечающих за различные операции с задачами и пользователями. Например, для пользователей реализованы маршруты для регистрации и авторизации, а для задач — маршруты для создания, получения, обновления и удаления задач.

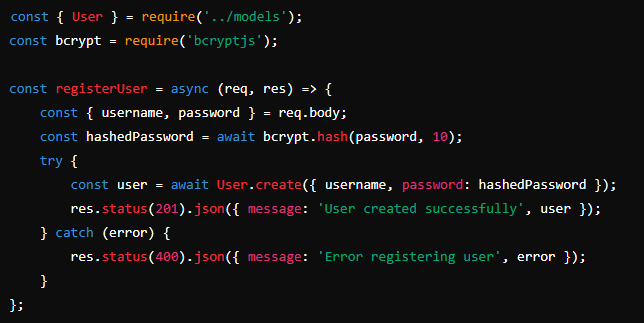


Рисунок 4 – «Пример кода для создания нового пользователя»

В этом примере создается новый пользователь в базе данных с хэшированием пароля для повышения безопасности. В случае успешной регистрации возвращается сообщение с информацией о созданном пользователе.

Особое внимание уделяется **обработке ошибок**, чтобы обеспечить устойчивость приложения к возможным сбоям и корректно информировать пользователей о возможных проблемах.

Взаимодействие с базой данных осуществляется через **Sequelize**, который является ORM для работы с реляционными базами данных в Node.js. Sequelize позволяет удобно создавать и управлять моделями, такими как User, Task, которые представляют таблицы в базе данных.

* 1. Разработка фронтенд-сервиса веб-приложения.

Фронтенд-сервис веб-приложения ориентирован на удобство и простоту взаимодействия с пользователем. Для реализации интерфейса используется стандартный набор технологий: **HTML, CSS и JavaScript**. Важной частью интерфейса является форма авторизации и регистрации пользователей, а также представление задач с возможностью их добавления, редактирования и удаления.

Одним из приоритетных направлений в разработке фронтенда является **минимализм в дизайне**. Это позволяет пользователю легко и быстро ориентироваться в приложении, сосредоточив внимание на функциональных элементах. Например, в интерфейсе отображается список задач с возможностью их модификации и удаления, а также форма для добавления новых задач.

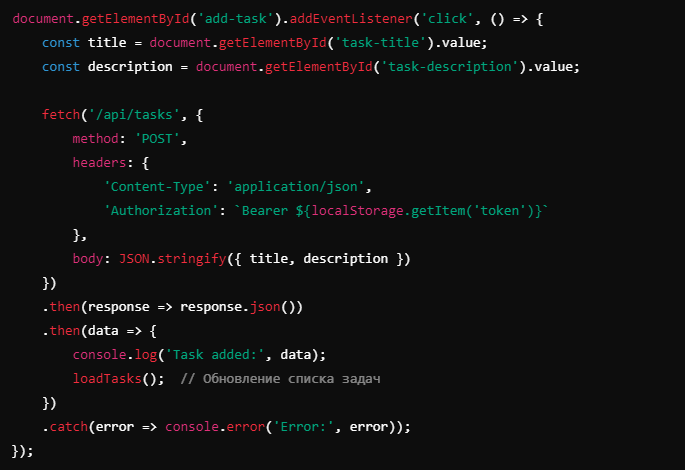


Рисунок 5 – «Пример кода для добавления новой задачи на фронтенде».

Этот код отправляет запрос на создание новой задачи на сервер, передавая данные, введенные пользователем в форму. Задача добавляется в базу данных, и после этого происходит обновление списка задач, отображаемого на странице. Важно, что для взаимодействия с API используется **JWT токен**, который передается в заголовке запроса для аутентификации пользователя.

Вся работа с интерфейсом организована с использованием **AJAX запросов**, чтобы не перезагружать страницу при добавлении или удалении задач, что делает взаимодействие с приложением более плавным и быстрым.

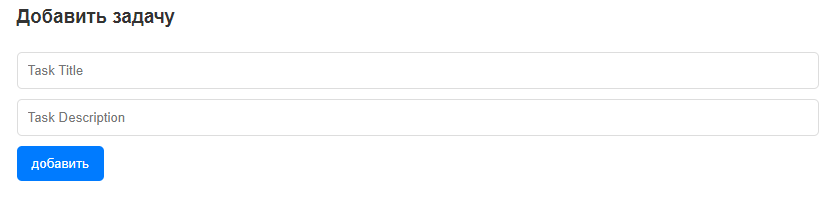


Рисунок 6 – «Форма добавления задачи».

Разработка фронтенд-сервиса веб-приложения в рамках данного проекта обеспечила создание удобного и функционального интерфейса, который позволяет пользователям эффективно взаимодействовать с системой и выполнять необходимые действия.

* 1. Описание кода

Важные участки кода можно выделить следующим образом:

***Авторизация пользователя (backend):*** Этот процесс включает проверку учетных данных пользователя и создание токена, который затем используется для аутентификации в последующих запросах.

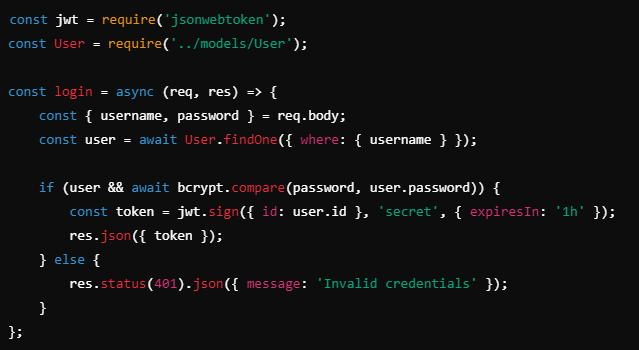


Рисунок 7 – «Пример кода для авторизации».

В данном примере, если введенные пользователем данные правильны, сервер генерирует JWT токен, который используется для аутентификации в дальнейшем. Это решение эффективно защищает систему от несанкционированного доступа.

***Удаление задачи (frontend):*** Пользователь может удалить задачу из своего списка. Для этого отправляется запрос на сервер с указанием ID задачи.



Рисунок 8 – «Пример кода для удаления задачи».

Этот код инициирует запрос на удаление задачи, и после успешного удаления обновляется список задач на интерфейсе.

***Обработка ошибок (backend):***  Важно обрабатывать возможные ошибки, такие как неверные данные, проблемы с подключением к базе данных и другие.



Рисунок 9 – «Пример обработки ошибки при создании пользователя».

Здесь, в случае ошибки при создании пользователя, сервер возвращает статус 400 и информацию об ошибке.

***Получение списка задач для пользователя (backend):*** Важным элементом является возможность пользователя видеть свои задачи.

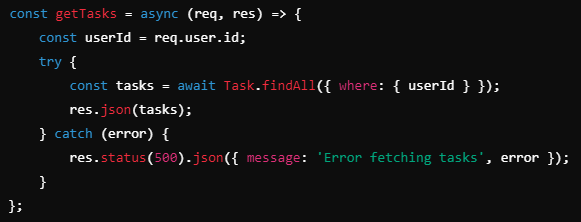


Рисунок 10 – «Пример кода для получения списка задач пользователя».

Этот код выполняет запрос к базе данных для получения всех задач пользователя, идентификатор которого извлекается из информации о токене (с помощью middleware для аутентификации). В случае ошибки, сервер возвращает сообщение с кодом 500.

***Добавление новой задачи (frontend):*** Важной частью приложения является создание новой задачи, которая позволяет пользователю эффективно управлять своими задачами и отслеживать их выполнение.

Рисунок 11 – «Пример кода для добавления задачи на фронтенде».

Этот код отправляет данные формы на сервер для создания новой задачи. После успешного добавления задачи в базу данных, данные обновляются на интерфейсе, и список задач становится актуальным.

Каждый из приведенных примеров демонстрирует ключевые операции, такие как взаимодействие с сервером, обработка данных пользователя и управление задачами. Эти части кода обеспечивают основные функции системы и показывают, как бэкенд и фронтенд взаимодействуют друг с другом для обеспечения функциональности веб-приложения.

## Выводы по третьему разделу

В результате выполнения третьего раздела курсовой работы был успешно реализован функционал как на стороне бэкенда, так и на стороне фронтенда. Веб-приложение эффективно выполняет свои основные функции: регистрацию и авторизацию пользователей, создание, редактирование и удаление задач.

Особое внимание было уделено безопасности данных пользователей через использование хэширования паролей и JWT для аутентификации. Также внедрены механизмы для минимизации времени отклика системы, такие как асинхронные запросы и обработка ошибок.

Пользовательский интерфейс был спроектирован таким образом, чтобы обеспечить удобство использования и максимальную простоту взаимодействия с системой. В будущем возможно расширение функционала, включая добавление новых возможностей для пользователей, улучшение производительности и оптимизацию безопасности.

Таким образом, разработка бэкенда и фронтенда была успешно завершена с учетом требований к функциональности и безопасности, а также с применением современных технологий.

# ЗАКЛЮЧЕНИЕ

В ходе разработки веб-приложения "Система управления задачами" была успешно реализована функциональная и удобная система для пользователей, позволяющая создавать, изменять и удалять задачи. Приложение включает в себя все необходимые компоненты для эффективного управления задачами и обеспечения доступа через безопасную авторизацию. Были использованы современные технологии, такие как Node.js, Express, Sequelize для работы с базой данных.

Процесс разработки был разделен на несколько этапов: от проектирования архитектуры и интерфейса системы до реализации серверной и клиентской части. Важным моментом является внимание к безопасности, использованию JWT-токенов для авторизации, а также продуманный дизайн интерфейса, что значительно улучшает пользовательский опыт.

В целом приложение функционирует корректно, обеспечивая стабильную работу для пользователей. Реализованные функции позволяют эффективно отслеживать и управлять задачами, что соответствует современным требованиям к таким системам.

В результате работы была достигнута основная цель проекта — создание удобного и функционального инструмента для управления задачами с поддержкой безопасной авторизации и интуитивно понятного интерфейса.

# СПИСОК ЛИТЕРАТУРЫ

1. **Keller, M., & Pivonka, A. (2018).** Web Development with MongoDB and Node.js. O'Reilly Media.
2. **"Express Documentation" (2021).** Express.js Official Documentation. <https://expressjs.com>
3. **"Sequelize Documentation" (2021).** Sequelize ORM Documentation. <https://sequelize.org>
4. **Flanagan, D.** JavaScript: The Definitive Guide. – O'Reilly Media, 2020.
5. **"React Documentation" (2021).** React Official Documentation. <https://reactjs.org> электронный.
6. **Flanagan, D. (2011).** JavaScript: The Definitive Guide. O'Reilly Media
7. **Swagger Official Documentation.** Swagger OpenAPI Specification. – URL: https://swagger.io/docs/ (датаобращения: 04.12.2024). – Текст: электронный.
8. **W3C.**Web Architecture: The Web Platform. – URL: <https://www.w3.org/>