Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

|  |  |  |  |
| --- | --- | --- | --- |
| Допускаю к защите | | | |
| Руководитель |  |  | А.В. Жуков |
|  | подпись |  | И.О. Фамилия |

|  |
| --- |
| Распределенное хранилище |
| наименование темы |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА |
| к курсовому проекту по дисциплине |

|  |
| --- |
| Web-программирование |

|  |
| --- |
| 1.019.00.00 |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | АСУб-21-1 |  |  |  | Рытик Е.Д. |
|  | шифр группы |  | подпись |  | Фамилия И.О. |
| Нормоконтроль | |  |  |  | Жуков А.В. |
|  | |  | подпись |  | Фамилия И.О |
| Курсовой проект защищен с оценкой | |  |  | | |

Иркутск 2024 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ЗАДАНИЕ

НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |  |  |
| --- | --- | --- | --- |
| По курсу | Web-программирование | | |
| Студенту | Рытик Елене Дмитриевне | | |
|  | (фамилия, инициалы) | | |
|  |  | | |
| Тема проекта | Разработка веб-приложения для облачного хранения данных | | |
| Исходные данные | | Разработка веб-приложения для эффективного хранения | |
| и управления файлами в распределенной среде | | | |
|  | | | |
| Рекомендуемая литература | | | React. Быстрый старт, 2-е изд, / Стефанов С. |
| 2024 г. – 304с., Node.js в действии / Кантелон Майк, Янг Алекс, Мек Бредли, | | | | |
| 2018 г. – 432с., Грокаем алгоритмы. Иллюстрированное пособие для | | | | |
| программистов и любопытствующих / Бхаргава А., 2024 г. – 288с. | | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Графическая часть на |  | листах. | | | | | |
|  |  |  | | | | | |
| Дата выдачи задания | «2» февраля 2024 г. | | | | | |
|  |  | | | | | |
| Задание получил студент | | |  | |  | Рытик Е.Д. |
|  | | | подпись | |  | Фамилия И.О |
|  | | | | | | |
| Дата представления проекта руководителю | | | | «20» июня 2024 г. | | |
|  | | | |  | | |
| Руководитель курсового проекта | | |  | |  | Жуков А.В. |
|  | | | подпись | |  | Фамилия И.О. |

**Содержание**

[Введение 4](#_Toc169594777)

[1 Постановка задачи 5](#_Toc169594778)

[3 Стек технологий для разработки 6](#_Toc169594779)

[4 Разработка архитектуры и проектирование системы 8](#_Toc169594780)

[5 Реализация веб-приложения 11](#_Toc169594781)

[6 Тестирование 13](#_Toc169594782)

[Заключение 22](#_Toc169594783)

[Список использованных источников 23](#_Toc169594784)

# Введение

Сегодняшний мир цифровых технологий требует от систем хранения данных не только высокой надежности и доступности, но и возможности масштабироваться для обработки и хранения больших объемов информации. Одним из подходов к решению этих задач является использование распределенных хранилищ данных, которые позволяют разделить данные на части и распределить их по нескольким серверам или узлам сети. Это обеспечивает не только повышенную надежность за счет отказоустойчивости, но и позволяет эффективно использовать ресурсы, оптимизируя процесс хранения и обработки данных.

В рамках данного проекта мы сосредоточимся на разработке и реализации распределенного хранилища, которое специально предназначено для обработки и хранения файлов. Основная идея заключается в том, чтобы при загрузке файла на сервер, он автоматически разделялся на несколько частей, которые затем распределялись по различным серверам. Это позволяет не только улучшить общую производительность системы за счет параллельной обработки данных.

Целью нашего исследования является разработка веб-приложения для распределенного хранилища, которая будет учитывать особенности работы с большими файлами и требования к надежности и скорости обработки данных.

# Постановка задачи

Объектом исследования является распределенное хранилище данных, предназначенное для обработки и хранения больших файлов путем их разделения на части и распределения по различным серверам.

Целью исследования: разработка и реализация веб-приложения для распределенного хранилища данных, обеспечивающего эффективное разделение файлов на части, их распределение по серверам и обеспечение доступности данных.

1. Определение требований к функциональности приложения

Приложение должно обладать следующими функциональными возможностями:

* Авторизация и управление пользователями
* Регистрация новых пользователей
* Вход в систему с использованием почты и пароля
* Загрузка файлов
* Поддержка загрузки файлов различного типа (тексты, изображения, видео, документы)
* Разделение загружаемого файла на части и их распределение по серверам
* Доступ к файлам
* Предоставление доступа к файлам после успешной загрузки всех частей
* Возможность скачивания файлов в их исходном формате
* Управление файлами
* Просмотр списка загруженных файлов пользователя
* Удаление файлов из хранилища
* Уведомления
* Отправка уведомлений о завершении загрузки файлов
* Отправка уведомлений о возникнувших ошибках

# Стек технологий для разработки

При разработке веб-приложения для распределенного хранилища данных, выбор технологического стека играет ключевую роль в определении его производительности, масштабируемости, безопасности и удобства использования. В этом проекте были выбраны следующие технологии: React для frontend части, Node.js с Express для backend, а также ряд дополнительных инструментов и библиотек для обеспечения функциональности и безопасности. Ниже приведено подробное обоснование выбора каждой из этих технологий.

React был выбран в качестве основы для фронтенда из-за его компонентной архитектуры, которая позволяет создавать динамические и интерактивные пользовательские интерфейсы. Компонентный подход упрощает разработку и поддержку кода, а также обеспечивает высокую производительность благодаря использованию виртуального DOM. React поддерживается крупнейшим сообществом разработчиков и обладает богатой экосистемой, что обеспечивает доступ к множеству готовых решений и инструментов.

Node.js и Express были выбраны для бэкенда из-за их производительности и гибкости. Node.js позволяет использовать JavaScript на стороне сервера, что обеспечивает единообразие разработки и упрощает интеграцию с фронтендом. Express предоставляет минимальный и гибкий фреймворк для создания веб-приложений, поддерживающий маршрутизацию, middleware и другие функции, необходимые для обработки HTTP-запросов.

Дополнительные технологии:

* **Axios:** Используется для выполнения HTTP-запросов к серверу из браузера, обеспечивая удобный и гибкий способ взаимодействия с API.
* **React-Redux и @reduxjs/toolkit:** Для управления состоянием приложения, что обеспечивает централизованное хранение данных и упрощает разработку сложных интерфейсов.
* **Multer:** Обеспечивает обработку multipart/form-data запросов, необходимых для загрузки файлов.
* **Express-Validator:** Позволяет валидировать входящие запросы, улучшая безопасность и качество данных.
* **Sequelize и PostgreSQL:** Sequelize — это ORM для работы с PostgreSQL, что обеспечивает удобное взаимодействие с базой данных и ее масштабируемость.
* **BcryptJS:** Используется для хеширования паролей, повышая безопасность пользовательских данных.
* **CORS:** Необходим для контроля доступа к API из различных источников, обеспечивая безопасность.
* **DotEnv:** Упрощает управление переменными окружения, повышая безопасность и удобство конфигурации.
* **JSON Web Tokens (JWT):** Используются для аутентификации и авторизации пользователей, обеспечивая безопасный механизм идентификации.
  1. Обоснование выбора
* **Производительность и масштабируемость:** React и Node.js обеспечивают высокую производительность и возможность масштабирования приложения, что критично для работы с большим количеством пользователей и данными.
* **Безопасность:** BcryptJS и JWT обеспечивают защиту данных и аутентификацию, а CORS контролирует доступ к API, минимизируя риски безопасности.
* **Удобство разработки:** Использование единого языка программирования (JavaScript) на всем протяжении разработки упрощает процесс и повышает продуктивность команды. Инструменты, такие как @reduxjs/toolkit и Sequelize, снижают сложность кода и ускоряют разработку.
* **Гибкость и поддержка:** Выбранные технологии имеют большое сообщество и множество ресурсов, что облегчает решение возникающих проблем и поиск решений.

# Разработка архитектуры и проектирование системы

* 1. Архитектурный подход

Архитектура системы распределенного хранилища данных должна быть спроектирована таким образом, чтобы обеспечить высокую доступность, надежность, масштабируемость и безопасность данных. В качестве основы для архитектуры выбран микросервисный подход, который позволяет разбить систему на отдельные модули, каждый из которых выполняет свою функцию и может развиваться независимо от других.

* 1. Ключевые компоненты системы

**Фронтенд:** Реализован на React и отвечает за пользовательский интерфейс, предоставляя функциональность для загрузки, поиска и управления файлами.

**Бэкенд:** Развернуто на Node.js с использованием Express. Обрабатывает логику бизнес-процессов, включая аутентификацию пользователей, управление файлами и взаимодействие с базой данных.

**База данных:** PostgreSQL используется для хранения метаданных о файлах, включая информацию о владельцах и других связанных данных.

**Сервисы аутентификации и авторизации:** Обеспечивают безопасность системы, используя JWT для аутентификации пользователей.

* 1. Детализация логики работы
* Вход и регистрация
  + Пользователь вносит данные для входа или регистрации через интерфейс на страницах «Регистрация» и «Вход»
  + Клиентская часть отправляет запрос на сервер
  + Сервер проверяет правильность введенных данных
  + Сервер проверяет наличие пользователя в базе данных, генерирует jsonwebtoken, который служит для аутентификации
  + Сервер отправляет данные о пользователе, token на клиентскую часть
  + Клиент записывает полученный token в локальное хранилище и перенаправляет пользователя на главную страницу приложения «Home» и отправляет запрос на получение списка файлов пользователя на сервер
  + Сервер отправляет данные о файлах, полученных из базы данных, на клиент
  + Клиент отображает полученный список файлов пользователю
* Загрузка файлов
  + Пользователь выбирает файл для загрузки на сервер через интерфейс главной страницы
  + Клиент отправляет запрос на сервер
  + Сервер проверяет доступность дополнительных серверов, на которых будут храниться части файлов.
  + Сервер разбивает файл на несколько частей (количество частей зависит от дополнительных серверов) и отправляет их на сервера.
  + Сервер добавляет информацию о файле в базу данных
  + Сервер отправляет сообщение об успешной загрузке частей файла на клиент
  + При возникновении ошибок, сервер отправляет сообщение об ошибке клиенту
  + Клиентская часть отображает полученное сообщение пользователю
* Скачивание файла
  + Пользователь выбирает файл для скачивания через интерфейс главной страницы
  + Клиент отправляет запрос на скачивание файла на сервер
  + Сервер проверяет доступность дополнительных серверов
  + Сервер отправляет запросы на скачивание частей файла на сервера.
  + Сервер собирает файл в единое целое
  + Сервер отправляет сообщение об успешном скачивании частей файла и целый файл
  + При возникновении ошибок, сервер отправляет сообщение об ошибке клиенту
  + Клиент формирует ссылку на скачивание.
* Удаление файла
  + Пользователь выбирает файл для удаления через интерфейс главной страницы
  + Клиент отправляет запрос на удаление файла на сервер
  + Сервер проверяет доступность дополнительных серверов
  + Сервер отправляет запросы на удаление частей файла на сервера
  + Сервер удаляет информацию о файле и базы данных
  + Сервер отправляет сообщение об успешном удалении файла
  + При возникновении ошибок, сервер отправляет сообщение об ошибке клиенту
  + Клиентская часть отображает полученное сообщение пользователю
* Выход из системы
  + Пользователь выходит из системы через интерфейс главной страницы
  + Клиент удаляет данные о пользователе и локального хранилища и перенаправляет его на страницу входа
  1. Проектирование интерфейсов
* Страница регистрации
  + Поля для ввода email и пароля
  + Ссылка на страницу входа
* Страница входа
  + Поля для ввода email и пароля
  + Ссылка на страницу регистрации
* Главная страница
  + Навигационное меню с пунктами: Загрузка файлов, Выход из системы
  + Список файлов с кнопками для скачивания и удаления

# Реализация веб-приложения

* 1. Подготовка окружения и установка зависимостей

Перед началом разработки необходимо настроить среду разработки, включающую в себя установку Node.js, npm, создание нового проекта React для frontend, создание нового проекта Node.js для backend и создание нескольких проектов Node.js для дополнительных серверов. Также следует настроить базу данных PostgreSQL.

* 1. Создание проекта для frontend

Выполним команду в консоли для создания нового проекта React:

npm create vite@latest

Далее необходимо установить дополнительные библиотеки, для этого перейдем в созданную папку проекта и выполним команду:

npm install axios react-redux @reduxjs-toolkit

* 1. Создание проекта для backend

Создадим папку для проекта, перейдем в нее и выполним команду для создания нового проекта Node.js:

npm init

Далее установим библиотеки:

npm install –D nodemon

npm install express express-validator cors multer dotenv pg sequelize jsonwebtoken bcryptjs

* 1. Описание классов и методов

В frontend части используются следующие классы и методы:

Таблица 1 – Описание классов frontend

|  |  |  |
| --- | --- | --- |
| № | Название | Назначение |
| 1 | App | Отображает текущую страницу |
| 2 | Login | Отображает страницу входа |
| 3 | Registration | Отображает страницу регистрации |
| 4 | Home | Отображает главную страницу |
| 5 | Navbar | Отображает навигационное меню на главной странице |
| 6 | FileList | Отображает список файлов |
| 7 | File | Отображает файл |
| 8 | Notification | Отображает уведомления |
| 9 | NotFound | Отображает страницу при переходе на несуществующий адрес |

Таблица 2 – Описание методов file.js

|  |  |  |
| --- | --- | --- |
| № | Название | Назначение |
| 1 | getFiles | Отправляет запрос на сервер для получения списка файлов |
| 2 | uploadFile | Отправляет запрос на загрузку файла на сервер |
| 3 | downloadFile | Отправляет запрос на сервер на скачивание файла |
| 4 | deleteFile | Отправляет запрос на сервер на удаление файла |

Таблица 3 – Описание методов user.js

|  |  |  |
| --- | --- | --- |
| № | Название | Назначение |
| 1 | registration | Отправляет запрос на сервер для регистрации нового пользователя |
| 2 | login | Отправляет запрос на сервер для авторизации существующего пользователя |
| 3 | auth | Отправляет запрос на сервер для получения token |

В backend части:

Таблица 5 – Описание методов auth.controller.js

|  |  |  |
| --- | --- | --- |
| № | Название | Назначение |
| 1 | registration | Добавляет данные о пользователе в базу данных |
| 2 | login | Сверяет данные с базой данных |
| 3 | auth | Генерирует token |

Таблица 6 – Описание методов file.controller.js

|  |  |  |
| --- | --- | --- |
| № | Название | Назначение |
| 1 | upload | Загружает файл и добавляет данные о файле в базу данных |
| 2 | download | Скачивает части с дополнительных серверов и объединяет их в один |
| 3 | delete | Удаляет части файла с дополнительных серверов и удаляет данные о файле из базы данных |
| 4 | getFiles | Получает данные о файлах из базы данных |

С кодом приложения можно ознакомиться по ссылке <https://github.com/lks-bogd/storage>

# Тестирование

Таблица 7 – Тестирование

|  |  |  |
| --- | --- | --- |
| № | Тестовый случай | Результат |
| 1 | Регистрация нового пользователя | Успешная регистрация и перенаправление на главную страницу |
| 2 | Вход существующего пользователя | Перенаправление на главную страницу |
| 3 | Ввод некорректных данных | Сообщение об ошибке |
| 4 | Загрузка файла | Успешная загрузка файла на сервер, соответствующее сообщение, отображение обновленного списка файлов |
| 5 | Скачивание файла | Успешное скачивание файла |
| 6 | Удаление файла | Успешное удаление файла, соответствующее сообщение, отображение обновленного списка файлов |
| 7 | Загрузка, скачивание, удаление файла при сбое одного из дополнительных серверов | Сообщение об ошибке |
| 8 | Выход из системы | Перенаправление на страницу входа |

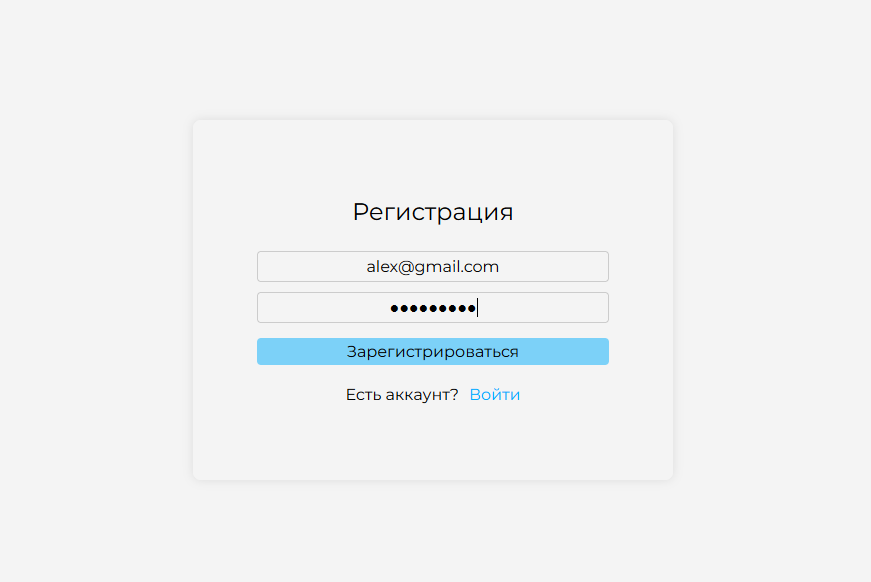


Рисунок - Корректная регистрация пользователя

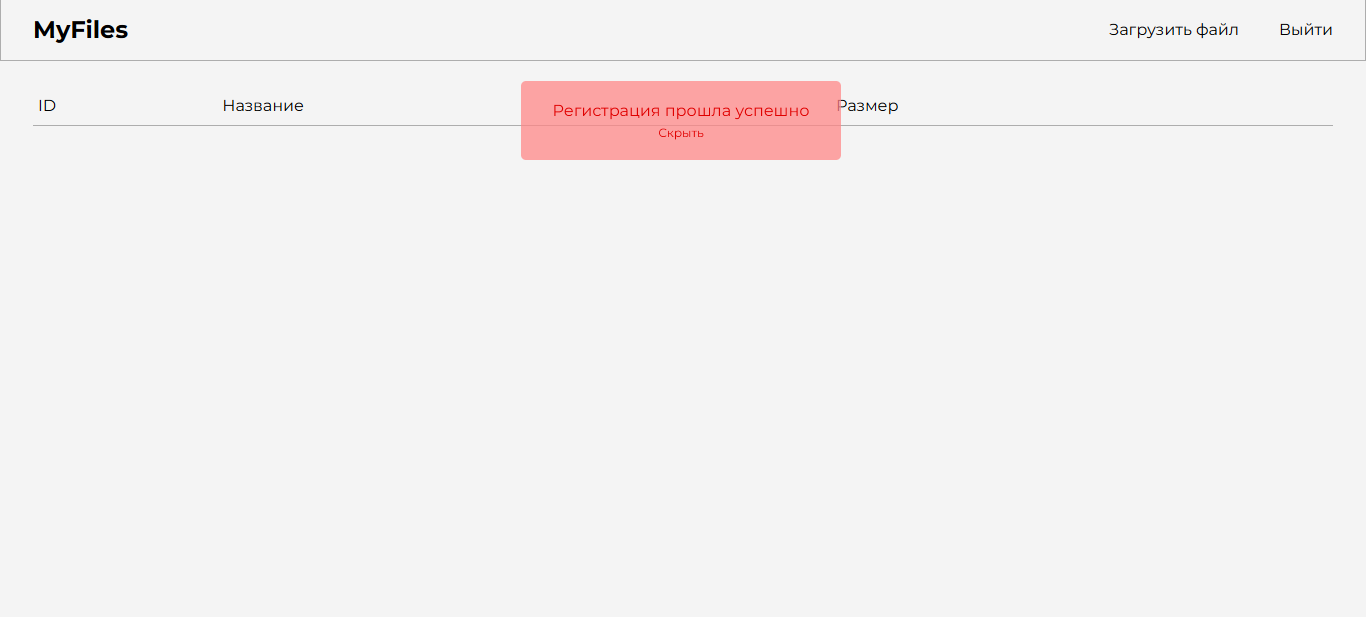


Рисунок - Корректная регистрация пользователя (продолжение)

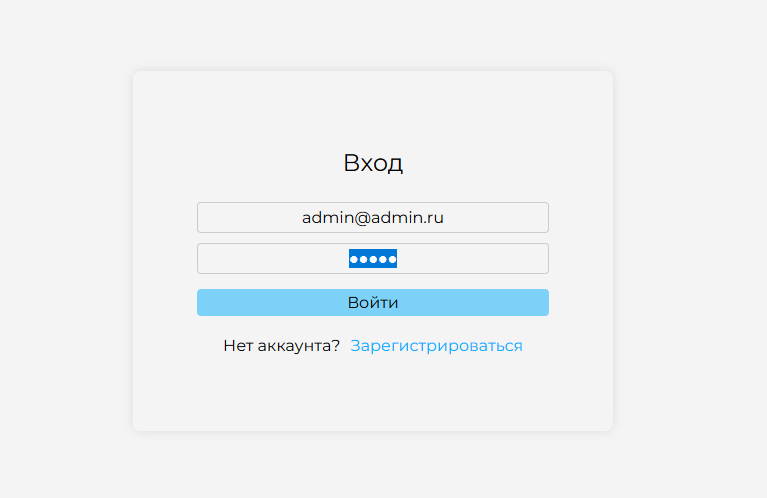


Рисунок - Корректный вход

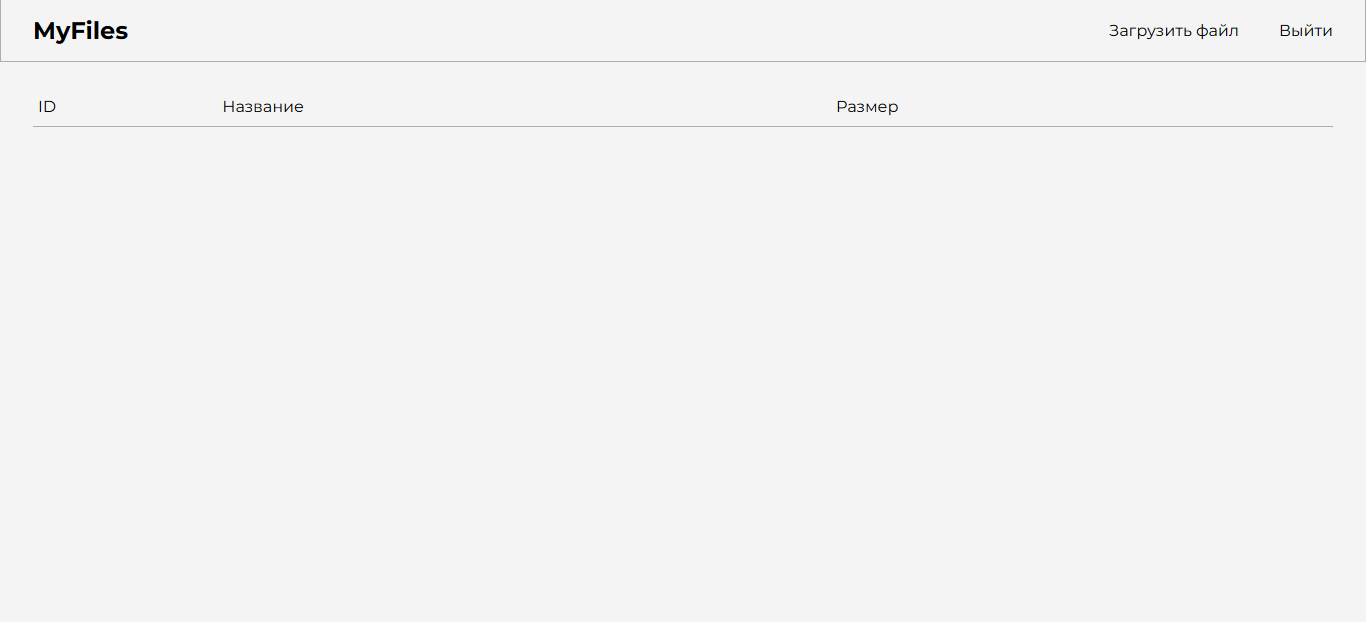


Рисунок - Корректный вход (продолжение)

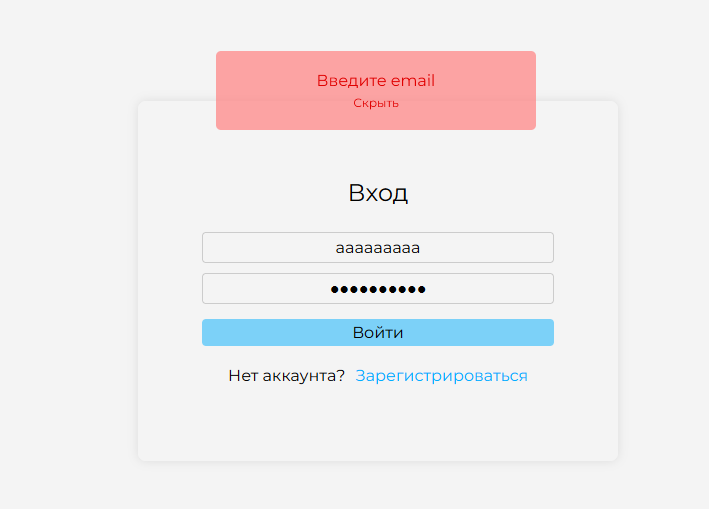


Рисунок - Ввод некорректного email

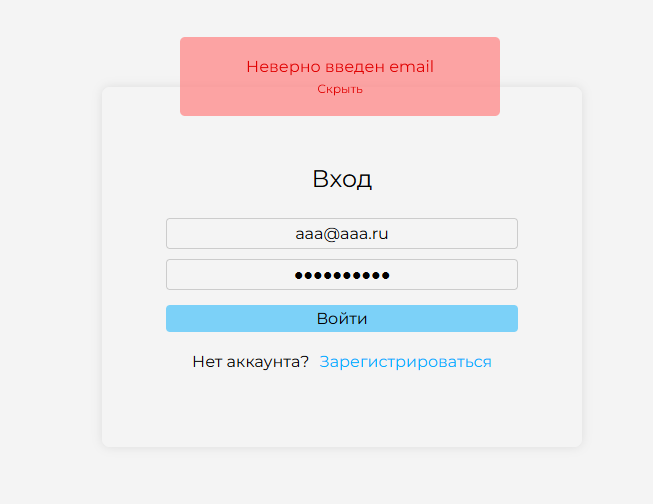


Рисунок - Ввод несуществующего email

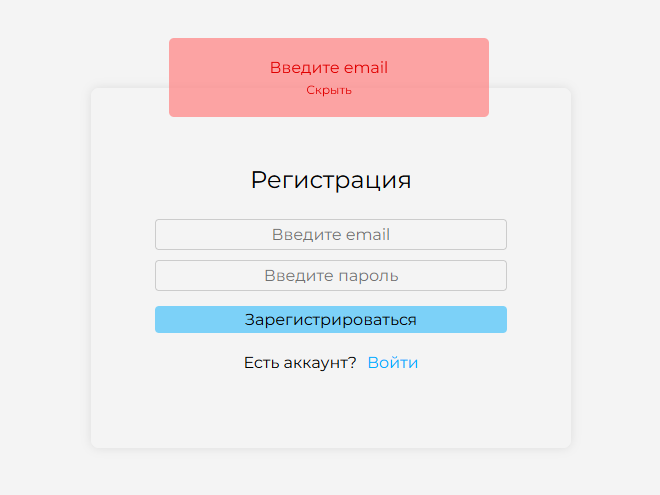


Рисунок - Пустой ввод

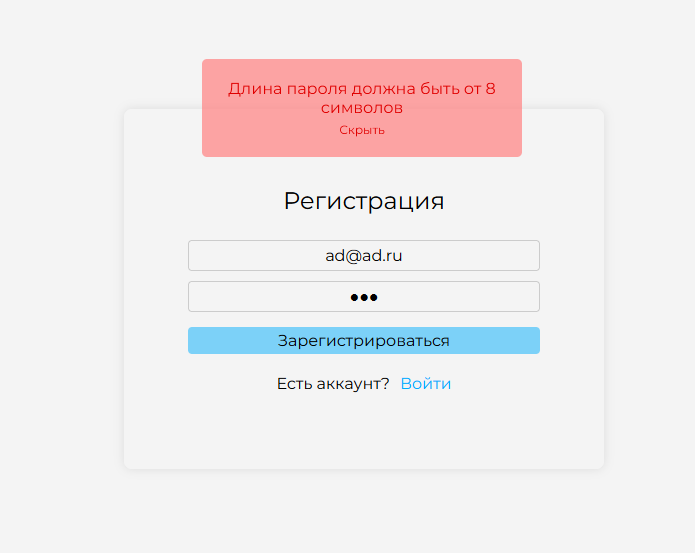


Рисунок - Ввод некорректного пароля

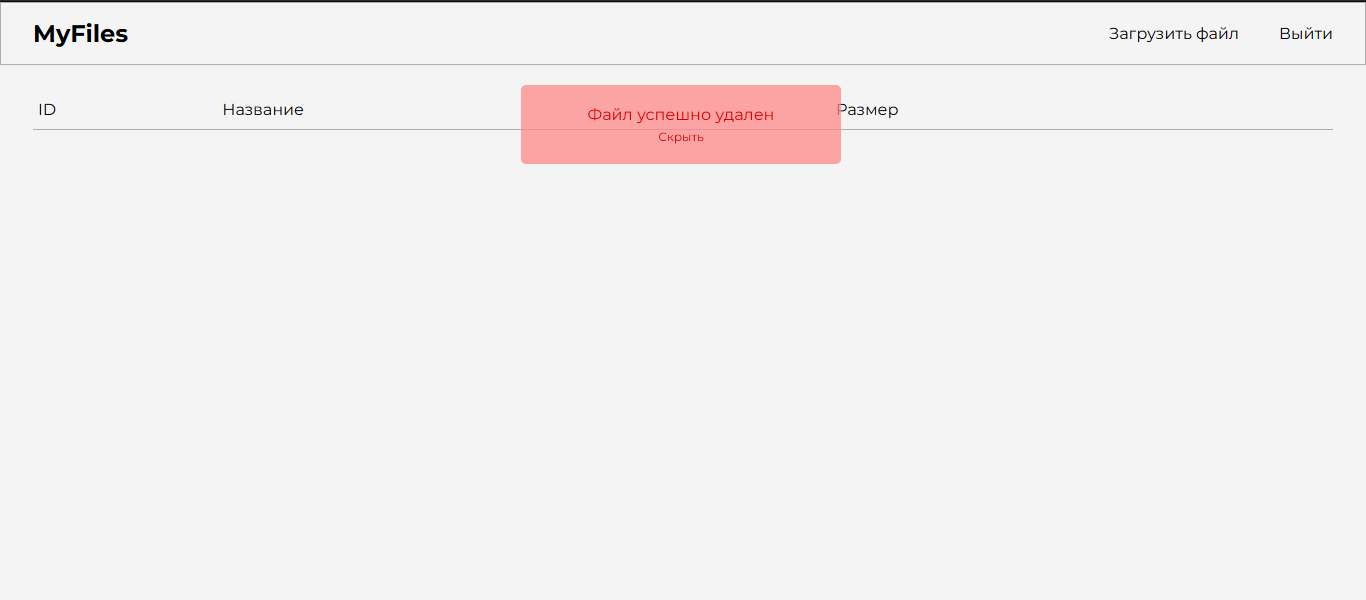


Рисунок 9 - Удаление файла

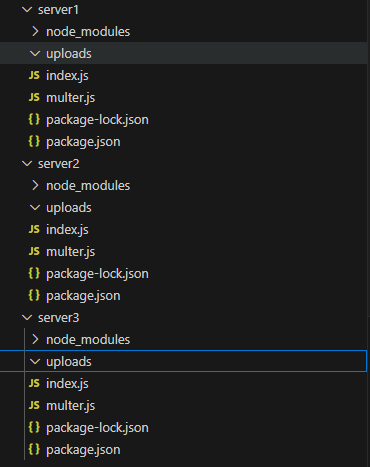


Рисунок 10- Удаление файла (продолжение)

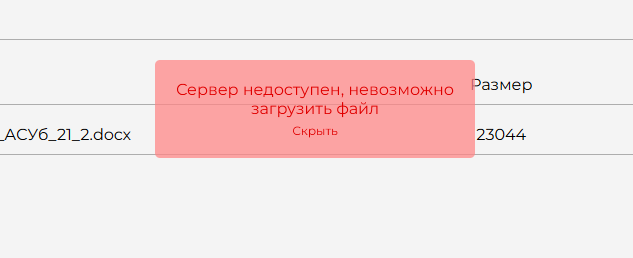


Рисунок 11- Загрузка файла при сбое сервера (продолжение)

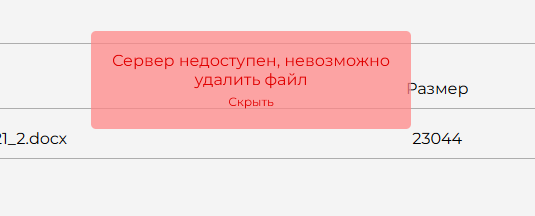


Рисунок 12 - Удаление файла при сбое сервера

# Заключение

Разработка веб-приложения для распределенного хранилища данных представляет собой комплексную задачу, требующую глубоких знаний в области веб-разработки, архитектуры систем и управления данными. Выбор современных технологий, таких как React для фронтенда и Node.js с Express для бэкенда, в сочетании с PostgreSQL для хранения данных, обеспечивает создание мощной и масштабируемой платформы, способной обрабатывать большие объемы информации с высокой степенью надежности и безопасности. Применение микросервисной архитектуры и облачных технологий позволяет обеспечить гибкость и масштабируемость системы, а также упрощает процесс разработки и поддержки.

В ходе работы над проектом были разработаны эффективные механизмы для загрузки, хранения и управления файлами, а также реализованы системы аутентификации и авторизации для защиты пользовательских данных. Использование таких инструментов, как Multer для обработки файлов, Sequelize для работы с базой данных и JWT для аутентификации, подчеркивает внимание к деталям и качеству реализации.

Реализация веб-приложения включала в себя не только техническую сторону, но и важные аспекты, такие как тестирование, документация и поддержка, что является ключом к успешному запуску и дальнейшему развитию проекта. Благодаря тщательному планированию и исполнению, система способна обеспечить высокий уровень доступности и отказоустойчивости, что критически важно для хранилищ данных, где потеря информации недопустима.

В целом, проект демонстрирует возможность создания сложных и функциональных веб-приложений с использованием современных технологий и лучших практик разработки. Он служит примером того, как можно объединить различные технологии для достижения целей проекта, обеспечивая при этом удобство использования и безопасность для пользователей. Это веб-приложение не только решает задачу хранения и управления файлами, но и представляет собой платформу для дальнейших инноваций и расширений, открывая возможности для интеграции с другими сервисами и сервисами.

Разработка такого рода систем требует тесного взаимодействия между разработчиками, аналитиками и пользователями для обеспечения соответствия потребностям рынка и удовлетворения ожиданий пользователей. Веб-приложение для распределенного хранилища данных является ярким примером того, как можно комбинировать технологии для создания решений, отвечающих вызовам современного цифрового мира, обеспечивая при этом высокую производительность, безопасность и удобство для пользователей.

# Список использованных источников

1. MERN – FullStack разработка облачного хранилища [Электронный ресурс] // youtube.com // URL: <https://www.youtube.com/playlist?list=PL6DxKON1uLOGd4E6kG6d5K-tsTFj-Deln> (дата обращения 07.06.2024)
2. React – the library for web and native user interface [Электронный ресурс] // react.dev // URL: <https://react.dev/learn> (дата обращения 01.06.2024)
3. Introduction to Node.js [Электронный ресурс] // nodejs.org // URL: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> (дата обращения 26.05.2024)
4. The Express + Node.js Handbook – Learn the Express JavaScript Framework for Beginners [2022 Edition] [Электронный ресурс] // freecodecamp.org // URL: <https://www.freecodecamp.org/news/the-express-handbook/> (дата обращения 30.05.2024)
5. Axios – getting started [Электронный ресурс] // asios-http.com // URL: <https://axios-http.com/docs/intro> (дата обращения 03.06.2024)