МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики и кибернетики

Кафедра программных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
  
 к курсовой работе по дисциплине «Технологии сети интернет»

по теме: «Интернет-магазин для продажи подержанных автомобилей»

Обучающийся М.А. Мананников

Руководитель М.В. Додонов

Самара 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc153888473)

[1 Описание и анализ предметной области 4](#_Toc153888474)

[1.1 Основные понятия и определения 4](#_Toc153888475)

[1.2 Описание систем-аналогов 5](#_Toc153888476)

[1.2.1 Облачное хранилище «Google Диск» 5](#_Toc153888477)

[1.2.2 Облачное хранилище «Яндекс Диск» 6](#_Toc153888478)

[1.2.3 Облачное хранилище «Dropbox» 7](#_Toc153888479)

[1.3 Постановка задачи 10](#_Toc153888480)

[2 Проектирование web-сайта 11](#_Toc153888481)

[2.1 Функциональная модель 11](#_Toc153888482)

[2.1.1 Входные и выходные данные 11](#_Toc153888483)

[2.1.2 Описание внешней информационной среды 11](#_Toc153888484)

[2.1.3 Перечень исключительных ситуаций 12](#_Toc153888485)

[2.1.4 Диаграмма вариантов использования 13](#_Toc153888486)

[2.2 Логическая схема данных 15](#_Toc153888487)

[2.2.1 Перечень таблиц и атрибутов 16](#_Toc153888488)

[2.3 Схемы алгоритмов обработки данных 17](#_Toc153888489)

[2.3.1 Алгоритм 1 17](#_Toc153888490)

[2.3.2 Алгоритм 2 18](#_Toc153888491)

[2.4 Прототип пользовательского интерфейса 19](#_Toc153888492)

[3 Реализация web-сайта 22](#_Toc153888493)

[3.1 Разработка физической модели БД 22](#_Toc153888494)

[3.2 Разработка серверной части 23](#_Toc153888495)

[3.3 Разработка клиентской части 24](#_Toc153888496)

[3.4 Тестирование сайта 26](#_Toc153888497)

[ЗАКЛЮЧЕНИЕ 30](#_Toc153888498)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc153888499)

[ПРИЛОЖЕНИЕ А Листинг программы 33](#_Toc153888500)

ВВЕДЕНИЕ

В современном мире интернет играет ключевую роль в упрощении и ускорении многих аспектов нашей жизни, включая поиск и покупку товаров и услуг. Одной из быстрорастущих областей электронной коммерции является продажа автомобилей, включая подержанные. Интернет-магазин для продажи подержанных автомобилей становится удобным и востребованным решением как для покупателей, так и для продавцов.

Интернет-магазин – это веб-сайт или приложение, позволяющее пользователям покупать и продавать товары и услуги через интернет. Он представляет собой виртуальную платформу, где покупатели могут просматривать доступные продукты, сравнивать их характеристики и цены, выбирать подходящие варианты и совершать покупки, не выходя из дома. Основой работы интернет-магазина является его каталог товаров, который обычно включает фотографии, описания, спецификации и отзывы пользователей. Все процессы, от выбора товара до оформления заказа и оплаты, происходят в онлайн-среде, что делает шопинг быстрым и удобным. Интернет-магазин также предоставляет функционал для управления заказами, доставки и обработки возвратов. Для продавцов это эффективный инструмент для расширения рынка сбыта и автоматизации взаимодействия с покупателями, а для покупателей – возможность быстро находить и приобретать необходимые товары, получая доступ к широкому ассортименту и выгодным условиям. [1].

1. Описание и анализ предметной области

Предметная область ­– это область знаний, которая изучает, описывает и анализирует определенный сегмент реальности или явление, фокусируясь на определенных объектах, процессах или проблемах. Она определяет предмет и структуру исследования, определяет основные понятия и термины, а также ограничения и рамки изучаемой области [2].

* 1. Основные понятия и определения

Облачное хранилище – это сервис, предоставляемый через интернет, который позволяет пользователям сохранять, управлять и обрабатывать свои данные на удаленных серверах, доступ к которым осуществляется через сеть. Оно обеспечивает масштабируемое хранение данных, синхронизацию, резервное копирование и возможность совместной работы над файлами, а также обеспечивает высокий уровень безопасности и доступность данных [3].

Масштабируемое хранение данных – это способ организации хранения и управления большим объемом данных, который позволяет легко и эффективно масштабировать систему с ростом объема данных [4].

База данных – это структурированная коллекция данных, организованных и хранимых в компьютерной системе. Она представляет собой специально организованный набор данных, которые логически связаны и обрабатываются с использованием базы данных и соответствующего программного обеспечения. База данных предоставляет средства для хранения, организации, управления и доступа к данным, а также поддерживает выполнение запросов и обработку данных. Базы данных используются для хранения различных типов данных, таких как тексты, числа, изображения, видео и другие, и обеспечивают эффективный и структурированный способ работы с информацией [5].

Хэширование – это процесс преобразования произвольного входного значения фиксированной длины в уникальную строку фиксированной длины, называемую хэшем. Хэш-функция, используемая для этого преобразования, принимает произвольные данные и вычисляет хэш-значение, которое представляет собой результат хэширования [6].

Объектный анализ:

* пользователь – объект, представляющий пользователя системы. Может иметь свойства, такие как имя, электронная почта и пароль;
* файл – объект, представляющий загруженный в облако файл. Может содержать информацию о названии, типе, размере и дате загрузки;
* папка – объект, представляющий каталог или папку в облачном хранилище. Может содержать информацию о названии, дате создания и дате последнего изменения.
  1. Описание систем-аналогов

Перед созданием облачного хранилища необходимо провести сравнительный анализ с аналогичными системами. Этот анализ позволит выявить преимущества и недостатки конкурирующих решений, а также определить, какие функциональные возможности можно дополнить или усовершенствовать по сравнению с уже существующими.

* + 1. Облачное хранилище «Google Диск»

Google Диск (Google Drive) – это облачное хранилище и сервис синхронизации файлов, предоставляемый компанией Google. Он позволяет пользователям сохранять, совместно использовать и обмениваться файлами и папками через интернет.

Пользователи могут хранить и загружать файлы с устройства или создавать их собственноручно при помощи инструментов Google (документы, таблицы и презентации). Пользователь может открыть доступ отдельному файлу или к папке. Доступ передаётся при помощи ссылки или уведомления по электронной почте, также можно настроить параметры доступа: читатель, комментатор или редактор [7].

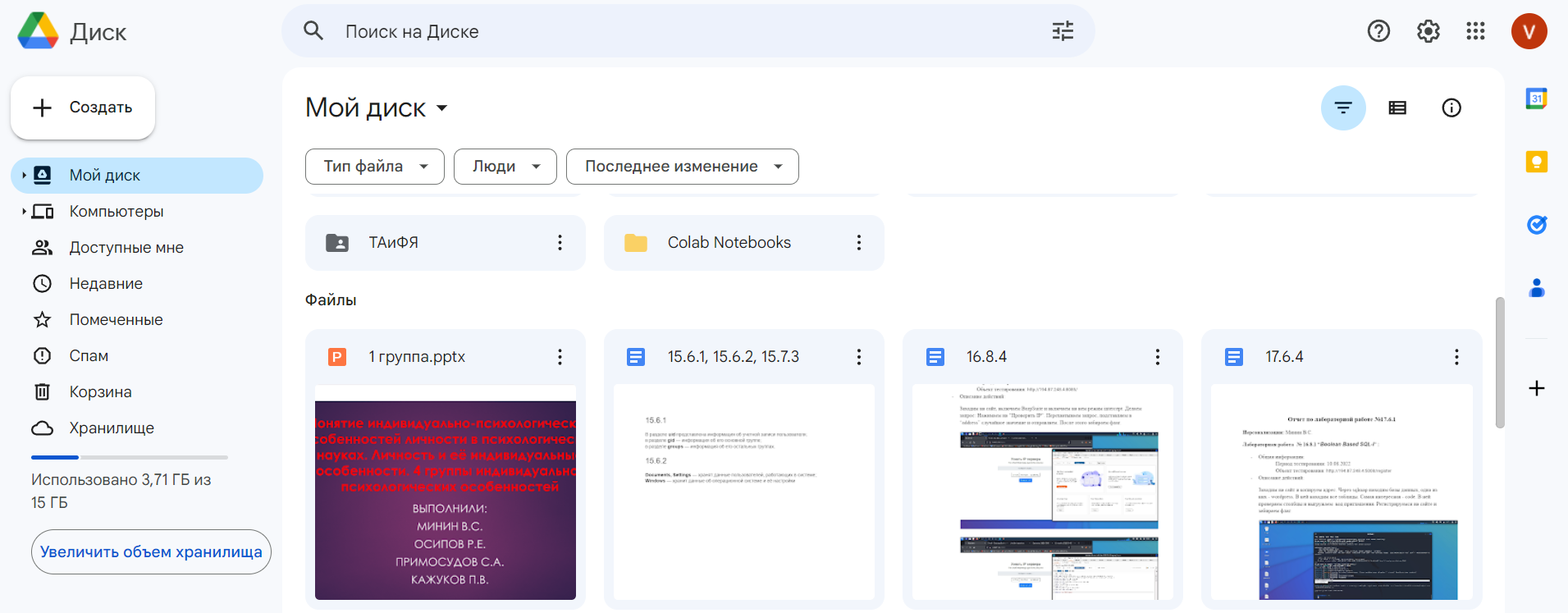
К преимуществам системы можно отнести:

* доступ к диску с любого устройства, подключённого к интернету;
* интуитивно понятный простой интерфейс;
* гибкое управление доступом к файлам;
* бесплатное предоставление 15 ГБ памяти;
* удобное взаимодействие с другими сервисами Google благодаря единому аккаунту.

К недостаткам можно отнести:

* работа с сервисом целиком зависит от доступа к интернету пользователя;
* платное масштабирование хранилища.

На рисунке 1 продемонстрирована вкладка Мой диск системы Google Диск.

  
Рисунок 1 – Вкладка Мой диск

* + 1. Облачное хранилище «Яндекс Диск»

Яндекс Диск – это облачное хранилище и сервис для хранения, совместного использования и обмена файлами, предоставляемый компанией Яндекс. Яндекс Диск позволяет пользователям загружать свои файлы в облако и получать к ним доступ через интернет с любого устройства [8].

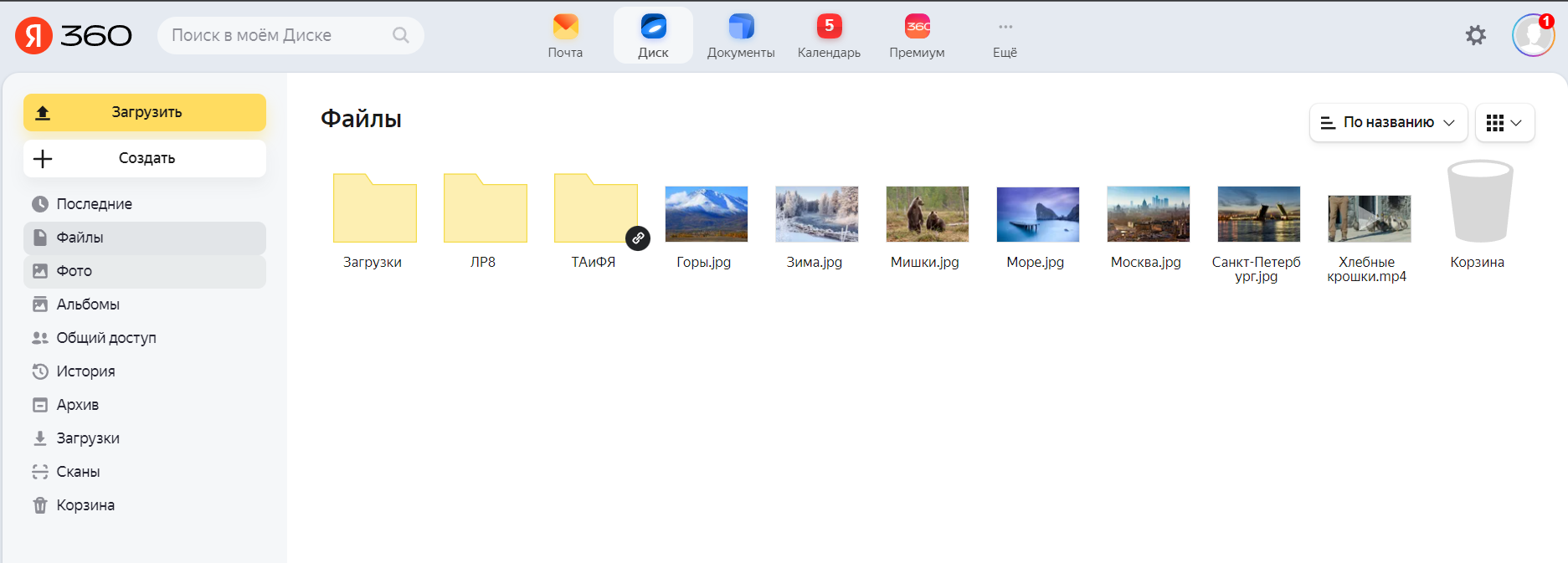
Преимущества данной системы:

* бесплатное предоставление 10 ГБ памяти;
* система имеет интуитивно понятный и простой интерфейс, что делает её легкой в использовании для всех категорий пользователей;
* хранилище позволяет пользователям делиться файлами и папками с другими пользователями, а также работать над одними и теми же файлами совместно, что улучшает коллаборацию;
* интеграция с другими сервисами Яндекса, такими как Яндекс Почта, Яндекс Календарь и Яндекс Документы, обеспечивает удобство использования и обмена данными между различными сервисами.

К недостаткам данной системы можно отнести:

* ограничения на бесплатный объем хранилища;
* зависимость от интернет-соединения;
* ограниченная географическая доступность.

На рисунке 2 показано хранилище Яндекс Диска.

  
Рисунок 2 – Хранилище Яндекс Диска

* + 1. Облачное хранилище «Dropbox»

Dropbox – это облачное хранилище и сервис синхронизации файлов, который позволяет пользователям хранить, совместно использовать и обмениваться файлами и папками через интернет. Он был запущен в 2007 году и стал одним из самых популярных и широко используемых облачных хранилищ.

Преимущества системы:

* система имеет интуитивно понятный интерфейс, что делает её легкой в использовании даже для непрофессиональных пользователей;
* обеспечивает возможность совместной работы над файлами и папками [9].

К недостаткам системы относятся:

* ограниченный бесплатный объем хранилища;
* ограничения на размер загружаемых файлов;
* зависимость от интернет-соединения;
* ограниченные функциональности для бесплатного аккаунта;
* приватность данных.

На рисунке 3 показан интерфейс Dropbox.

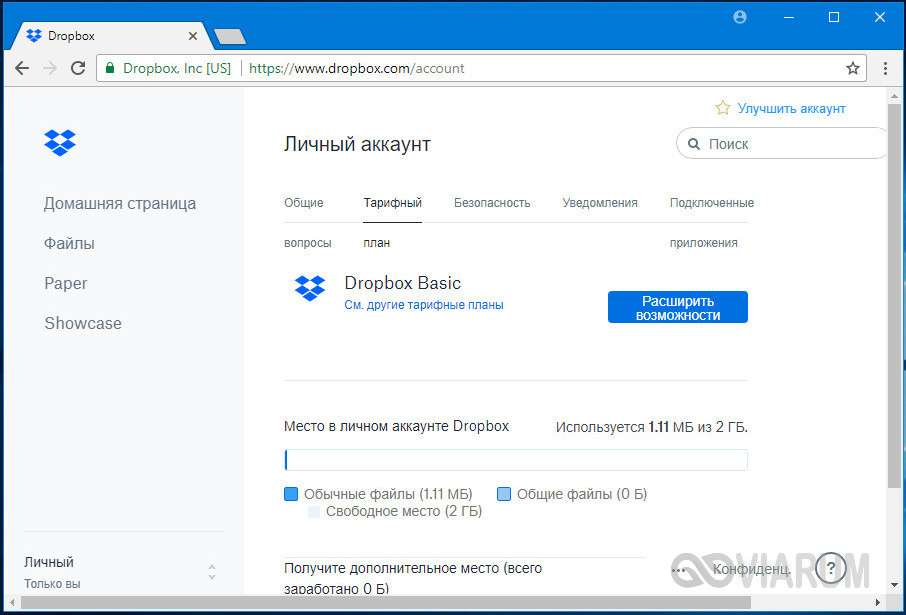
  
Рисунок 3 – Интерфейс системы

Таблица 1 – Сравнительные характеристики систем-аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Название системы  Название показателя | Google Диск | Яндекс Диск | Dropbox |
| Защита данных | + | + | + |
| Система управления доступом | + | + | + |
| Поддержка файлов разных форматов | + | + | + |
| Объём бесплатной версии | + | + | – |
| Удобный интерфейс | + | + | + |
| Интеграция с другими сервисами | + | + | – |
| Зависимость от интернет-соединения | – | – | – |
| Географическая доступность | + | – | + |
| Реклама | + | – | + |
| Возможности для бесплатного аккаунта | + | + | – |

* 1. Постановка задачи

Во время курсового проектирования необходимо разработать облачное хранилище, которое позволяет пользователям создавать профиль, создавать и удалять папки, загружать, скачивать и удалять файлы.

Требования к информационному, техническому и программному обеспечению:

* в базе данных должна храниться информация о пользователях и их данных;
* клиентская часть должна предоставлять удобный интерфейс;
* клиентская часть сайта должна быть разработана на JavaScript;
* серверная часть сайта должна быть разработана на Express.js;
* для хранения данных должна использоваться реляционная база данных;
* в качестве СУБД должна использоваться MySQL;
* для стилизации пользовательского интерфейса можно использовать CSS.

1. Проектирование web-сайта

На этапе проектирования системы нужно решить ряд задач:

* + определить требования к системе;
  + создать структуру и архитектуру системы;
  + разработать пользовательский интерфейс;
  + выбрать технологию разработки;
  + выбрать платформу разработки.
  1. Функциональная модель

Функциональная модель – это структурированное представление функций в смоделированной системе или предметной области [10].

* + 1. Входные и выходные данные

Входные данные:

* электронная почта;
* пароль;
* файлы, которые пользователь хочет хранить в облачном хранилище.

Выходные данные:

* файлы, которые пользователь хочет получить из облачного хранилища;
* сообщения и уведомления для пользователя о состоянии хранения данных или выполнении операций в облачном хранилище;
* данные об ошибках.
  + 1. Описание внешней информационной среды

Внешняя информационная среда включает в себя следующие компоненты:

* интернет-подключение: облачное хранилище требует высокоскоростного и стабильного подключения к Интернету для обеспечения быстрой передачи данных между пользовательскими устройствами и облачным сервером;
* устройства доступа: различные устройства, такие как компьютеры, ноутбуки, смартфоны и планшеты, на которых пользователи могут получить доступ к облачному хранилищу с помощью различных операционных систем;
* браузеры: пользователи могут обращаться к облачному хранилищу через веб-браузеры;
* протоколы и стандарты: облачное хранилище может использовать различные протоколы для передачи данных, такие как HTTP, FTP и протоколы, обеспечивающие безопасное шифрование и аутентификацию;
* безопасность и конфиденциальность: облачное хранилище должно быть защищено от несанкционированного доступа, взломов и утечек данных.
  + 1. Перечень исключительных ситуаций

Исключительная ситуация – это ситуация, при которой система не может выполнить возложенных на нее функций или которая может привести к денормализации работы системы.

В таблице 2 приведен перечень исключительных ситуаций для разрабатываемой системы и описаны реакции системы на их возникновение.

Таблица 2 – Перечень исключительных ситуаций

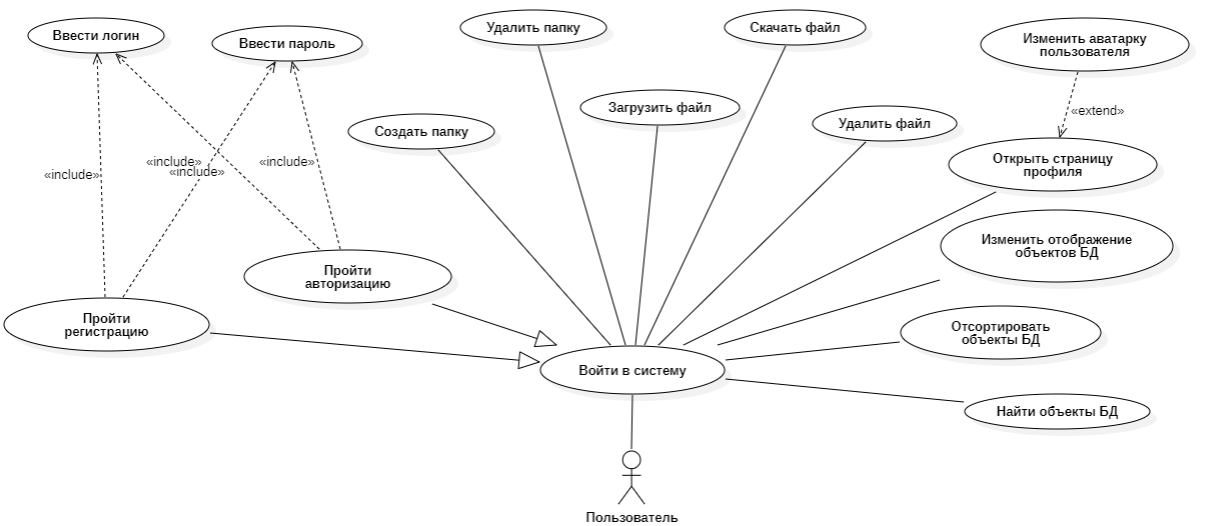
|  |  |  |
| --- | --- | --- |
| Название  подсистемы | Название исключительной ситуации | Реакция системы |
| 1 Регистрация и авторизация | 1.1 Некорректный логин | Выдача сообщения «Некорректный email» |
| 1.2 Некорректный пароль | Выдача сообщения «Некорректный пароль» |
| 2 Взаимодействие с файлом | 3.3 Невозможно загрузить файл с из-за отсутствия памяти | Выдача сообщения «Недостаточно памяти» |
| 3.4 Невозможно загрузить файл из-за ошибки на сервере | Выдача сообщения «Ошибка сервера» |

* + 1. Диаграмма вариантов использования

Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель сложной системы, которая является исходной для построения всех остальных диаграмм. На ней изображаются отношения между актерами и вариантами использования [11].

На рисунке 4 приведена диаграмма вариантов использования (пользователя). На диаграмме представлен пользователь, который может взаимодействовать с системой следующим образом:

* «войти в систему» – пользователь может войти в систему. Для этого ему необходимо создать свой профиль в процессе регистрации и успешно пройти авторизацию;
* «создать папку» – пользователь может создать новую папку в облачном хранилище;
* «удалить папку» – пользователь может удалить пустую папку;
* «загрузить файл» – пользователь может загрузить файлы в облачное хранилище;
* «скачать файл» – пользователь может скачать файлы;
* «удалить файл» – пользователь может удалить файлы;
* «открыть страницу профиля» – пользователь может открыть страницу своего профиля, где ему будет доступна возможность изменения аватарки;
* «изменить отображение объектов БД» – пользователь может изменить стиль отображения объектов БД. Для этого ему нужно будет нажать кнопку с соответствующим стилем;
* «отсортировать объекты БД» – пользователь может отсортировать папки и файлы выбрав один из трёх параметров сортировки;
* «найти объекты БД» – пользователь может найти папку или файл, указав в поисковике подстроку, входящую в название объекта.

  
Рисунок 4 – Диаграмма вариантов использования

* 1. Логическая схема данных

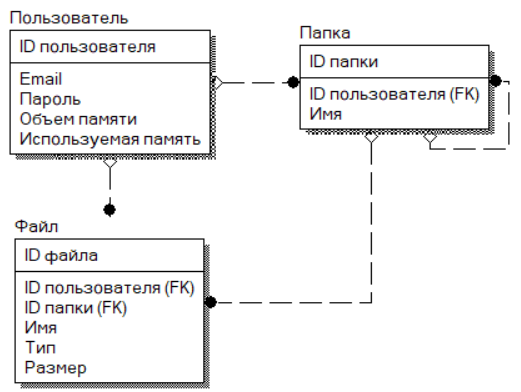
Логическая информационная модель – модель данных, в которой учитывается способ логического хранения данных в памяти ЭВМ. При построении модели базы данных (БД) используются следующие понятия.

Сущность – объект предметной области, который можно отличить от других понятий по некоторым признакам. Сущность состоит из множества своих экземпляров. Каждая сущность обладает свойствами – атрибутами.

Атрибут – определенное свойство сущности. Именно набор атрибутов, в общем случае уникальный для каждой сущности, позволяет выделить ее среди других объектов и назвать уникальным именем.

Атрибут или набор атрибутов, используемый для идентификации экземпляра сущности, называется ключом сущности. В случае если для идентификации экземпляра используется один атрибут, ключ называется простым; в противном случае ключ составной. Каждый экземпляр сущности однозначно определяется ключом [12].

Логическая модель БД разрабатываемой системы приведена на рисунке ХХХ.

  
Рисунок 5 – Логическая модель данных

* + 1. Перечень таблиц и атрибутов

Описание объектов рассматриваемой предметной области, которые хранятся в базе данных, приведено в таблицах 3-5.

Таблица 3 – Сущность «Пользователь»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID пользователя | Целый | Уникальный идентификатор пользователя |
| Пароль | Символьный [10] | Пароль пользователя, преобразованный в захешированную строку |
| Email | Символьный [50] | Электронная почта, указанная пользователем при регистрации |
| Объем памяти | Целый | Память, предоставляемая пользователю |
| Используемая память | Целый | Объем памяти, который пользователь использует на текущий момент |

Таблица 4 – Сущность «Файл»

|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID файла | Целый | Уникальный идентификатор пользователя |
| Имя | Символьный [30] | Имя файла |
| Тип | Символьный [10] | Расширение файла |
| Размер | Целый | Размер файла |
| ID пользователя | Целый | Внешний ключ из таблицы «Пользователь» |
| ID папки | Целый | Внешний ключ из таблицы «Папка» |

Таблица 5 – Сущность «Папка»

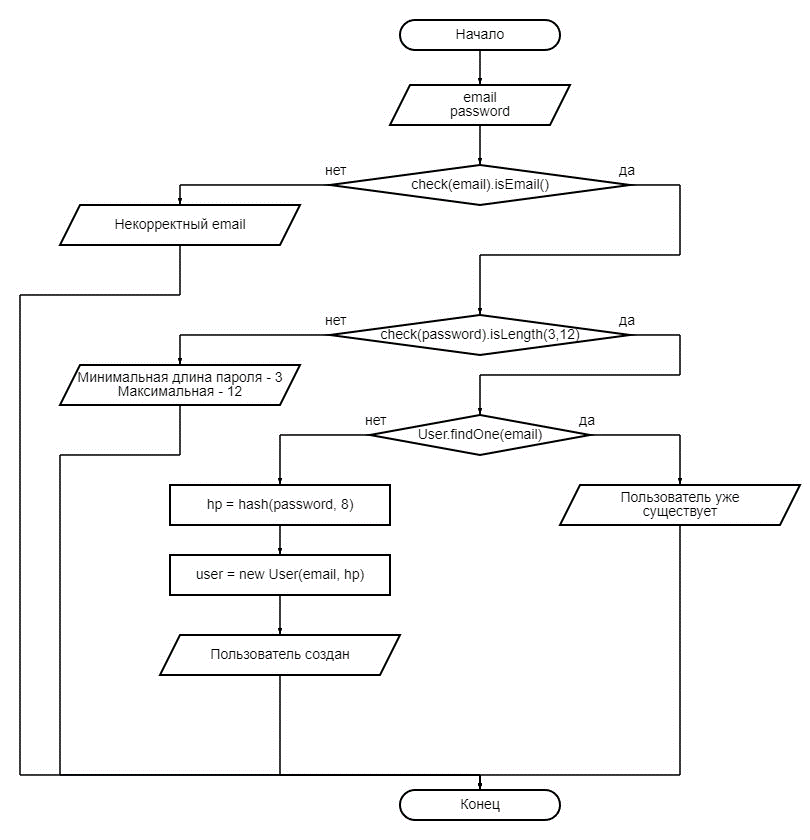
|  |  |  |
| --- | --- | --- |
| Идентификатор | Тип данных | Описание |
| ID папки | Целый | Уникальный идентификатор папки |
| Имя | Символьный [30] | Имя папки |
| ID пользователя | Целый | Внешний ключ из таблицы «Пользователь» |

* 1. Схемы алгоритмов обработки данных

Алгоритмом обработки данных называют метод решения задачи, который возможно реализовать в выбранной среде программирования. Тщательная разработка алгоритма является весьма эффективной частью процесса решения задачи в любой области применения [13].

* + 1. Алгоритм 1

На рисунке 7 приведена схема алгоритма регистрации пользователя

  
Рисунок 6 – Схема алгоритма регистрации пользователя

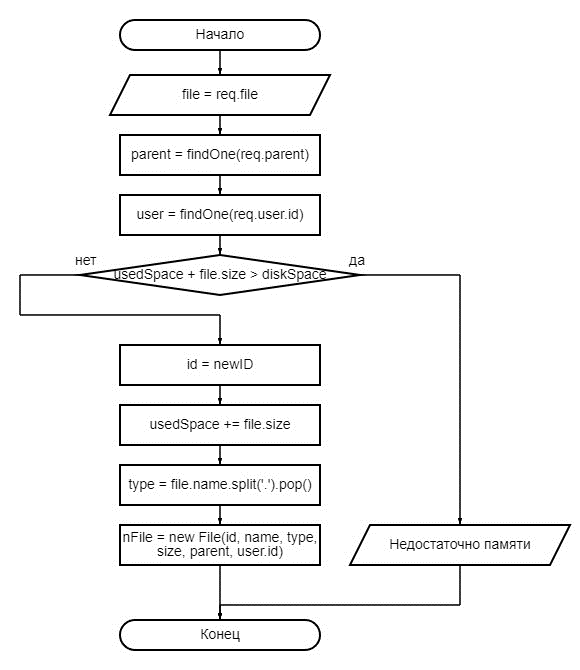
Описание алгоритма.

Пользователь вводит логин и пароль. Система осуществляет проверку корректности введенных данных. В случае ошибки система выводит сообщение об ошибке.

Если введенные данные корректны, то они отправляются на сервер, где осуществляется проверка уникальности электронной почты. Если введенная почта находится в БД, то выводится сообщение, что пользователь уже существует. Когда введенная почта уникальна, то система хеширует пароль, создаёт аккаунт и выводит сообщение, что пользователь создан.

* + 1. Алгоритм 2

На рисунке 8 приведена схема алгоритма загрузки файла в хранилище.

  
Рисунок 7 – Схема алгоритма загрузки файла в хранилище

Описание алгоритма.

На сервер поступает запрос загрузки файла. Система определяет директорию, в которую собираются загрузить файл и профиль пользователя.

Проверяется размер загружаемого файла. Если занятая память и размер файла в сумме превышают объем памяти, которая выделялась профилю, то выводится сообщение об отсутствии места. В противном случае создаётся ID нового файла, определяется его тип и размер. После этого создаётся новый объект в БД.

* 1. Прототип пользовательского интерфейса

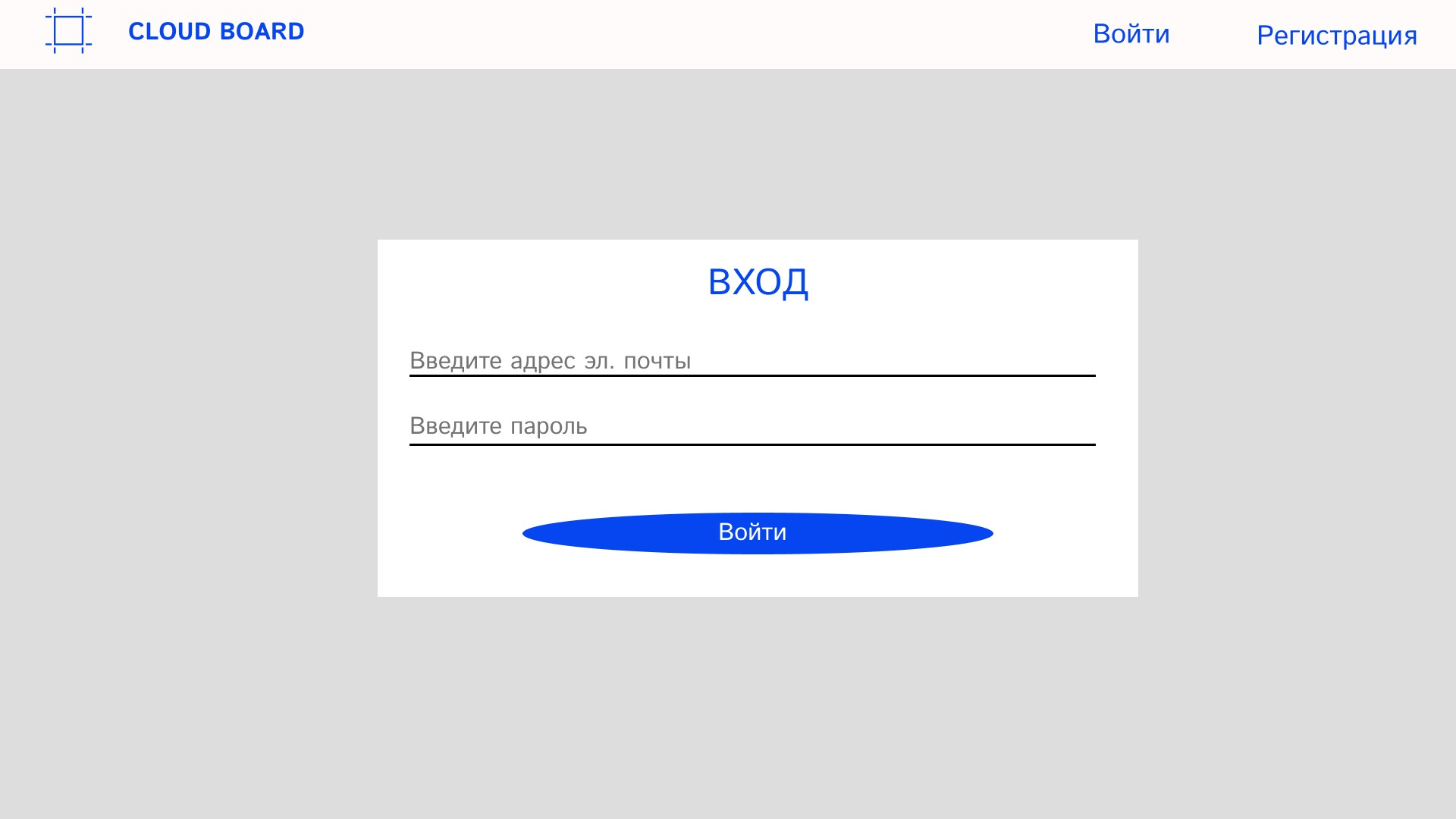
Интерфейс – это набор инструментов, который позволяет пользователю взаимодействовать с программой.

Прототипирование помогает выявить потенциальные проблемы или недостатки интерфейса, что дает возможность внести корректировки на ранних стадиях разработки [14].

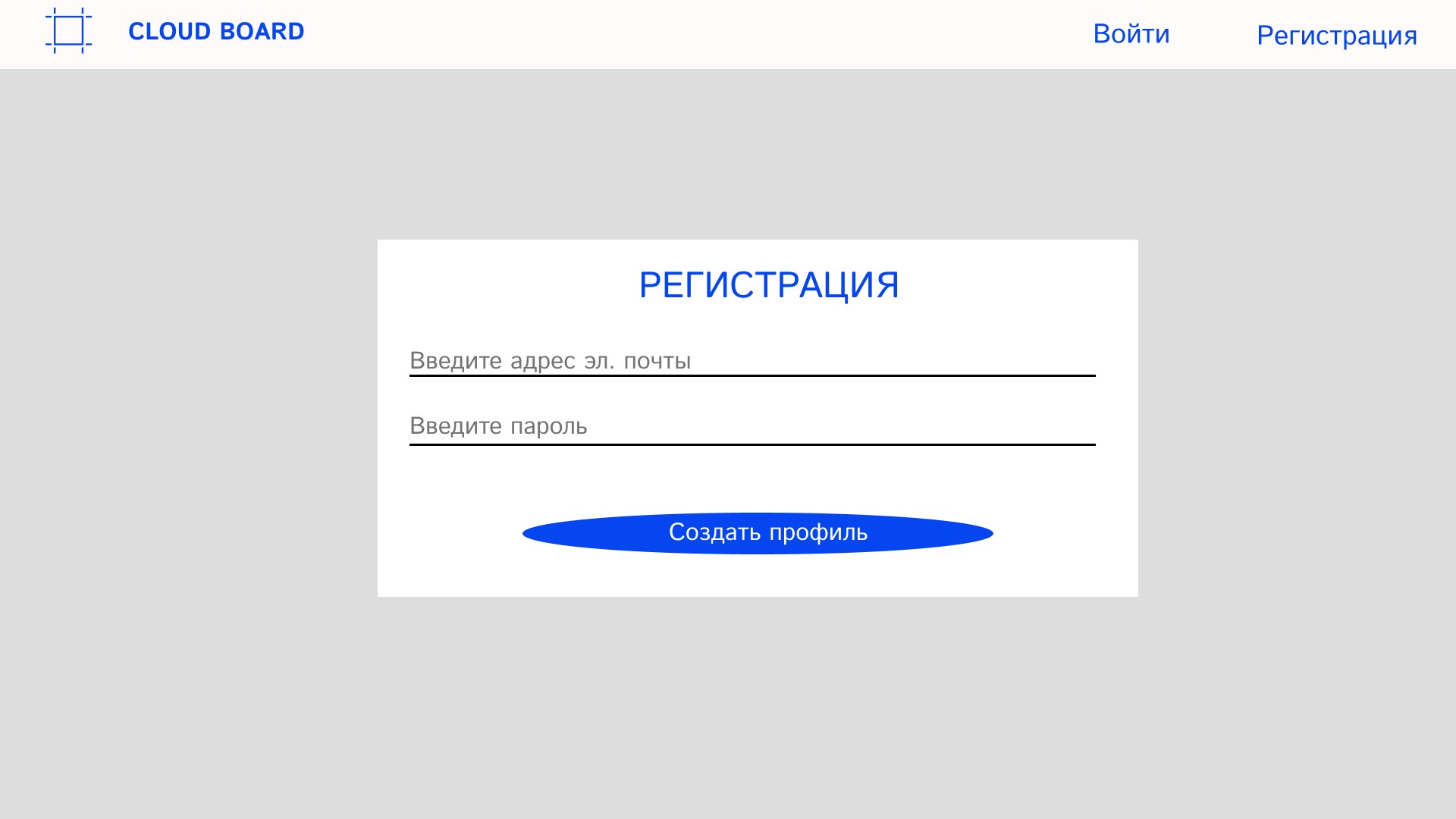
Основными требованиями к интерфейсу являются:

* функциональность – набор возможностей, операций и действий, предоставляемых пользователю или другим системам через данный интерфейс;
* понятность – интуитивно понятный интерфейс.

На рисунке 6 приведен прототип формы авторизации. При запуске приложения будет открываться форма авторизации. Здесь пользователь сможет ввести адрес электронной почты и пароль для входа в систему.

  
Рисунок 8 – Прототип экранной формы авторизации

При отсутствии профиля пользователь сможет перейти на форму регистрации, где ему потребуется ввести адрес электронной почты и придумать пароль. На рисунке 7 приведён прототип формы регистрации.

  
Рисунок 9 – Прототип экранной формы регистрации

При успешном входе в систему откроется экранная форма главной страницы (рисунок 8). На странице будут представлены:

* доска со списком папок и файлов пользователя, хранящихся в БД;
* кнопка «Создать новую папку» – при нажатии на кнопку система должна будет открыть форму создания папки, на которой пользователь должен будет указать её имя;
* кнопка «Загрузить файл» – при нажатии на кнопку система должна будет открыть диалоговое окно для загрузки файла;
* кнопки, меняющие отображение объектов БД на странице;
* меню сортировки папок и файлов: по имени, по дате создания, по типу;
* кнопка «Скачать файл» - при нажатии на кнопку файл будет загружен на ПК;
* кнопка «Удалить» – при нажатии на кнопку файл или папка будут удалены;
* значок профиля – при нажатии система откроет страницу профиля;
* кнопка «Назад», позволяющая пользователю выходить из папок.

  
Рисунок 10 – Прототип экранной формы главной страницы

1. Реализация web-сайта
   1. Разработка физической модели БД

Физическое проектирование является последним этапом проектирования базы данных, при выполнении которого принимается решение о способах реализации разрабатываемой базы данных. Во время логического проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области).

Физическая модель базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей [15].

В качестве СУБД была выбрана MySQL. Среди главных ее достоинств, необходимых для решения поставленной задачи, можно отметить:

* + - * оптимизацию связей с присоединением многих данных за один проход;
      * гибкую поддержку форматов чисел, строк переменной длины и меток времени;
      * быструю поддержку транзакций через механизм InnoDB [16].

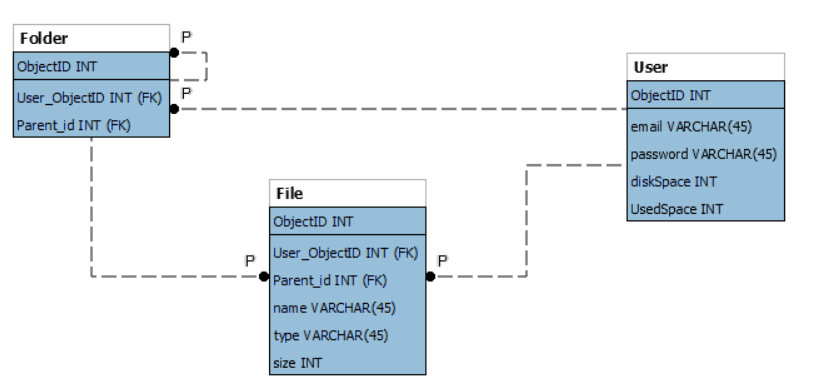
Кроме указанных достоинств, в MySql также есть поддержка хранимых процедур и функций.

Хранимые процедуры – это набор инструкций, которые компилируются и сохраняются в базе данных. Они могут быть вызваны из других программ или запросов к базе данных. Хранимые функции – это подобные процедурам, но они всегда возвращают значение, которое может быть использовано в других запросах или выражениях [17].

В облачном хранилище используются хранимые процедуры для реализации различных операций, таких как:

* создание нового пользователя: хранимая процедура принимает на вход логин, пароль и другую необходимую информацию о пользователе, а затем создает новую запись в таблице пользователей;
* удаление файла: хранимая процедура принимает на вход имя файла и удаляет соответствующую запись из базы данных;
* создание папки: хранимая процедура принимает на вход имя папки, а затем создает новую запись в таблице папок;
* удаление папки: хранимая процедура принимает на вход имя папки и удаляет соответствующую запись из базы данных.

На рисунке 11 представлена физическая модель данных системы.

  
Рисунок 11 – Физическая модель базы данных

* 1. Разработка серверной части

Разработка серверной части является важнейшим аспектом веб-разработки и разработки программного обеспечения. Это включает в себя создание и поддержку серверных компонентов приложения, которые отвечают за обработку данных и управление ими, а также за обработку скрытой логики [18].

Серверная часть облачного хранилища разработана на языке JavaScript (JS). Использование языка JavaScript на сервере и на клиенте упрощает разработку.

Рассмотрим реализацию серверной части.

Регистрация пользователя:

* сервер получает запрос, хеширует пароль и вызывает хранимую процедуру в БД.

Создание директории:

* пользователь отправляет запрос с данными о директории: имя, родительская папка, ID пользователя;
* сервер получает запрос и вызывает хранимую процедуру.

Загрузка файла:

* пользователь отправляет файл, указывая директорию, в которую нужно загрузить файл;
* сервер сохраняет файл на сервере и создает запись о файле в таблице "Files", указывая его имя, размер, тип, путь к файлу и связь с директорией.

Получение списка файлов и папок:

* пользователь запрашивает список файлов и папок в конкретной директории, отправляя запрос;
* сервер извлекает информацию о файлах и папках из таблиц "Files" и "Folders", используя запросы SQL JOIN, и отправляет обратно клиенту.

Удаление файла или папки:

* пользователь отправляет запрос, указывая ID файла или папки, которую нужно удалить;
* сервер вызывает хранимую процедуру и удаляет файл с сервера.

Поиск файлов и папок:

* пользователь отправляет запрос, указывая ключевые слова или фильтры для поиска;
* сервер выполняет поиск по базе данных, используя запросы SQL с оператором LIKE или другими методами, и возвращает результаты клиенту.
  1. Разработка клиентской части

Разработка клиентской части веб-приложения является важным этапом создания любого веб-проекта. Клиентская часть, также известная как front-end, отвечает за то, как приложение будет выглядеть и взаимодействовать с пользователями [19].

Клиентская часть облачного хранилища написана на языке JavaScript (JS). Рассмотрим её реализацию.

Регистрация пользователя:

* пользователь заполняет форму регистрации, вводит email и пароль;
* клиент отправляет POST-запрос на сервер с учетными данными пользователя.

Аутентификация пользователя:

* пользователь вводит свои учетные данные (email и пароль) в форму авторизации;
* клиент отправляет POST-запрос на сервер с учетными данными.

Отображение файлов и папок:

* после входа в систему пользователь видит основное окно облачного хранилища, где отображаются файлы и папки в его корневой директории;
* клиент отправляет GET-запрос на сервер для получения списка файлов и папок.

Загрузка файла:

* пользователь выбирает файл на своем устройстве и нажимает кнопку загрузки;
* клиент отправляет POST-запрос на сервер с загружаемым файлом и его метаданными.

Создание папки:

* пользователь в интерфейсе облачного хранилища выбирает опцию создания новой папки;
* клиент отправляет POST-запрос на сервер с данными о новой папке (название, родительская папка).

Удаление файла или папки:

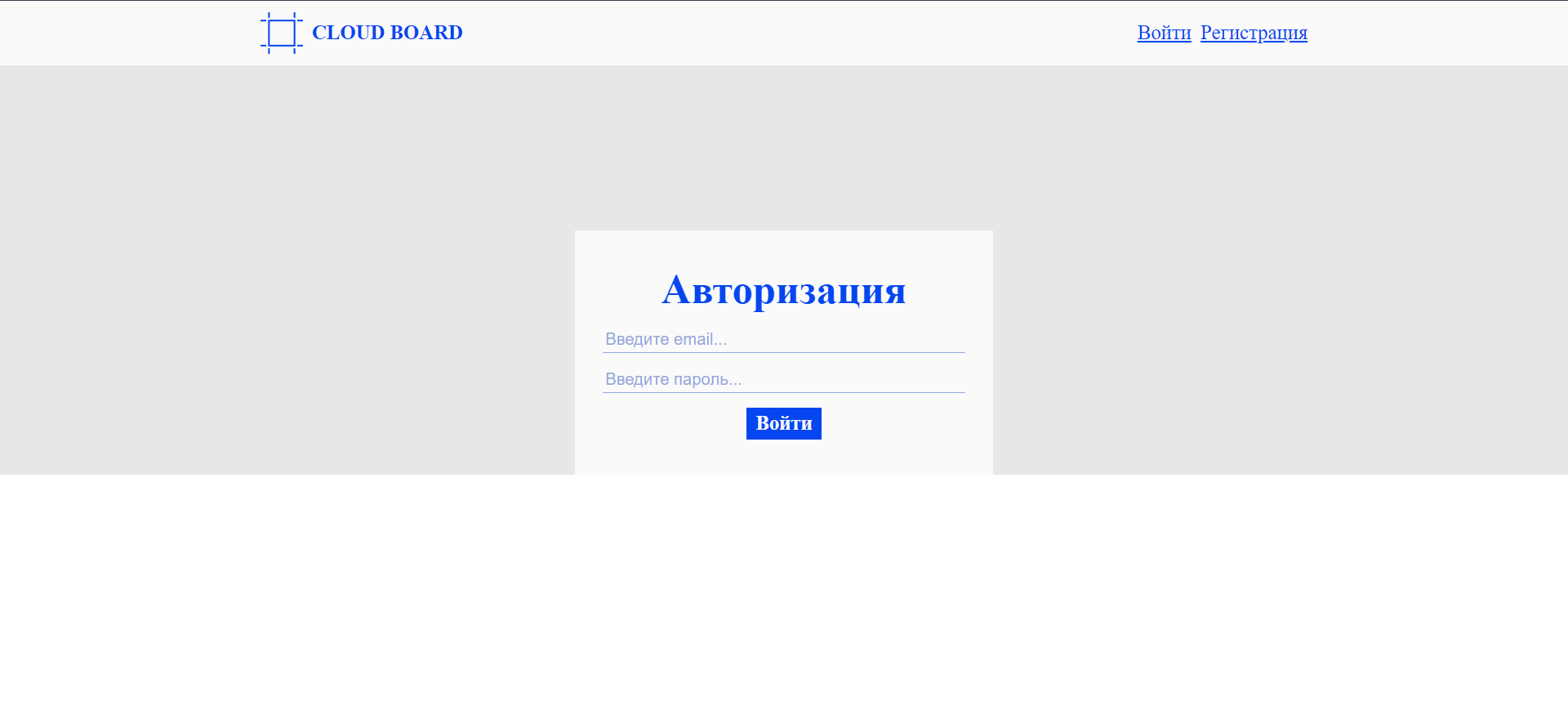
* пользователь выбирает файл или папку для удаления и нажимает кнопку "Удалить";
* клиент отправляет DELETE-запрос на сервер с ID удаляемого файла или папки.

Поиск файлов и папок:

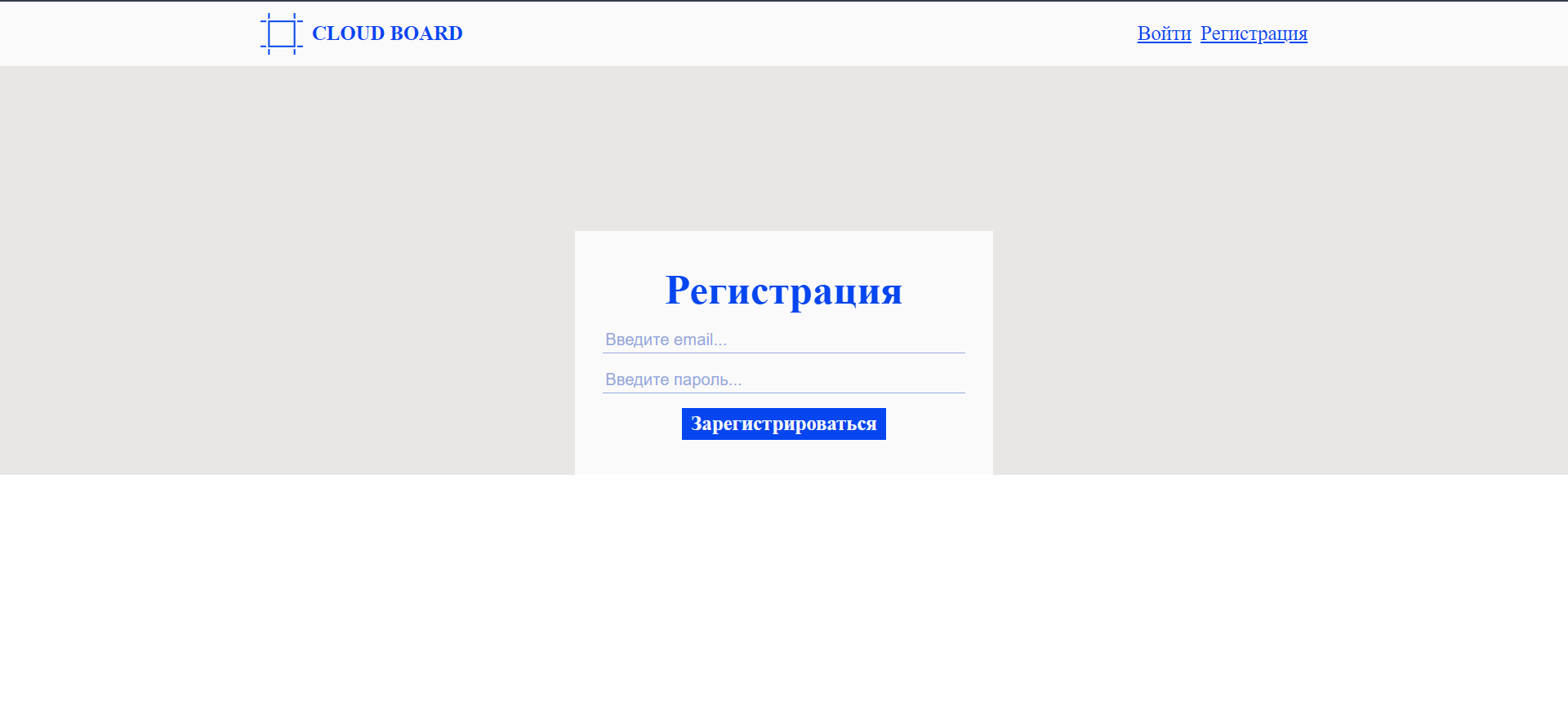
* пользователь вводит ключевые слова в поле поиска;
* клиент отправляет GET-запрос на сервер с данными для поиска.
  1. Тестирование сайта

Веб-тестирование – это процесс проверки и оценки качества веб-приложений или сайтов, чтобы обеспечить их надежность, производительность, безопасность и совместимость с различными браузерами, устройствами и операционными системами [20].

На рисунке 12 приведена страница авторизации, где пользователю предоставляется возможность войти в свой аккаунт.

  
Рисунок 12 – Страница авторизации

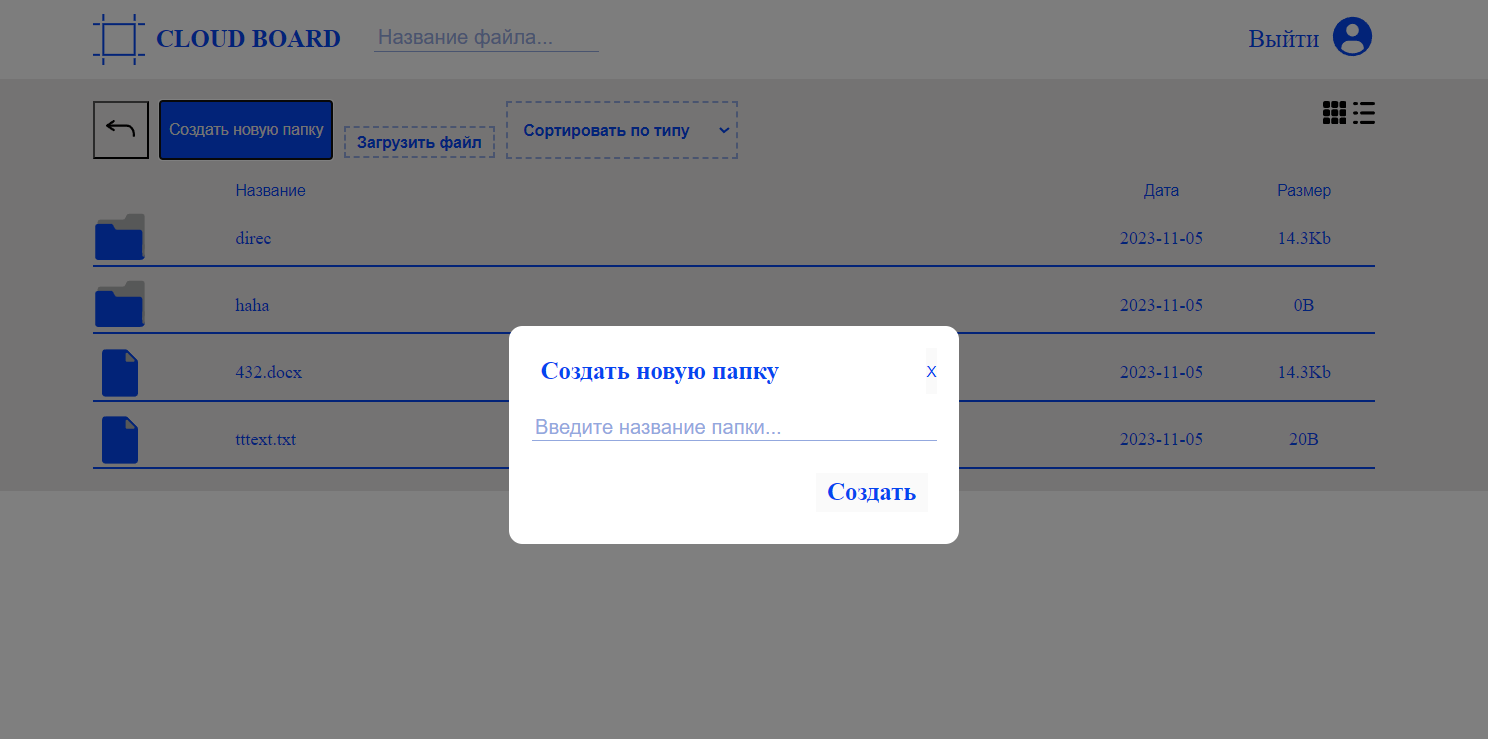
Если у пользователя нет аккаунта, он может перейти на страницу регистрации, где представлена возможность создать профиль. Страница регистрации приведена на рисунке 13

  
Рисунок 13 – Страница регистрации

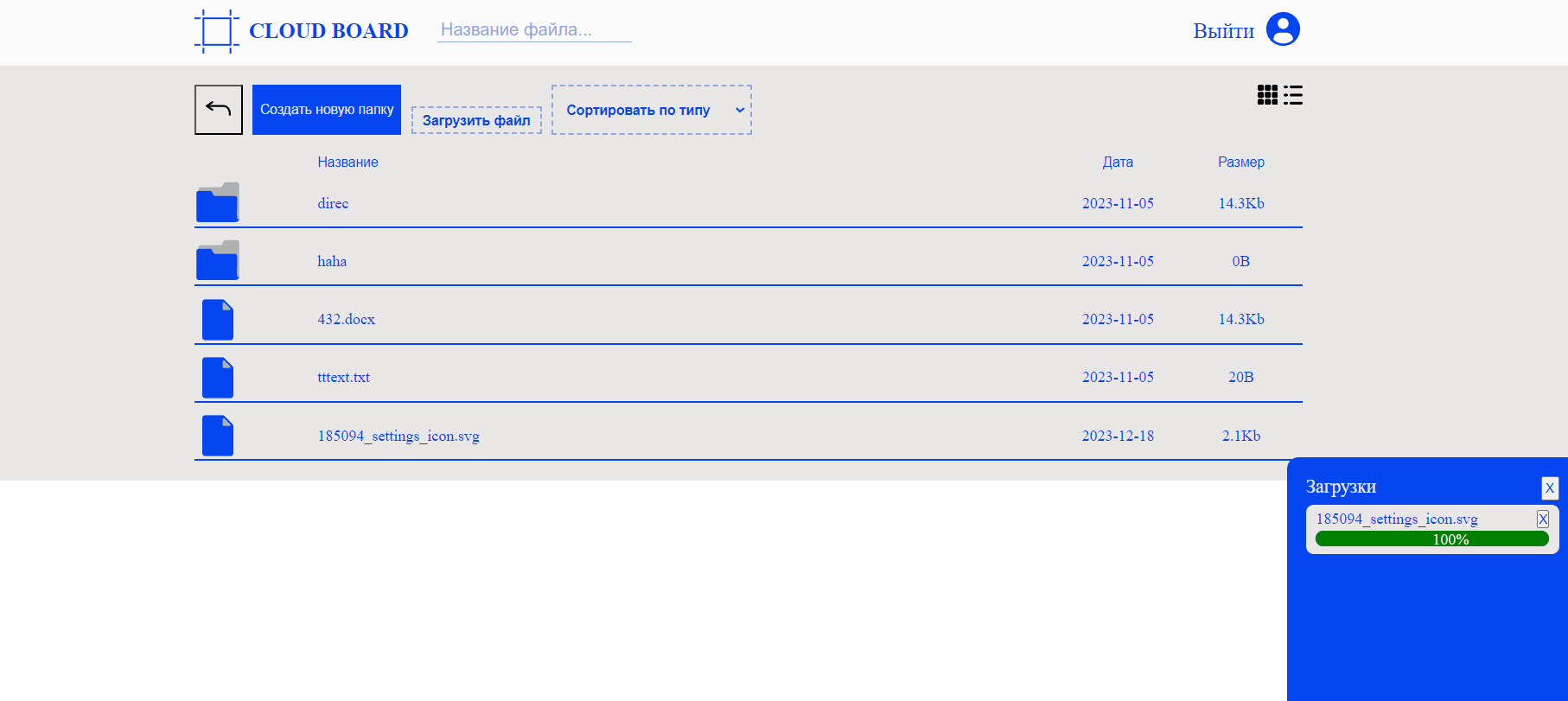
После успешного входа в систему открывается главная страница облачного хранилища, которая приведена на рисунке 14. На этой странице отображается список файлов и папок, а пользователю становятся доступными все основные функции.

  
Рисунок 14 – Главная страница

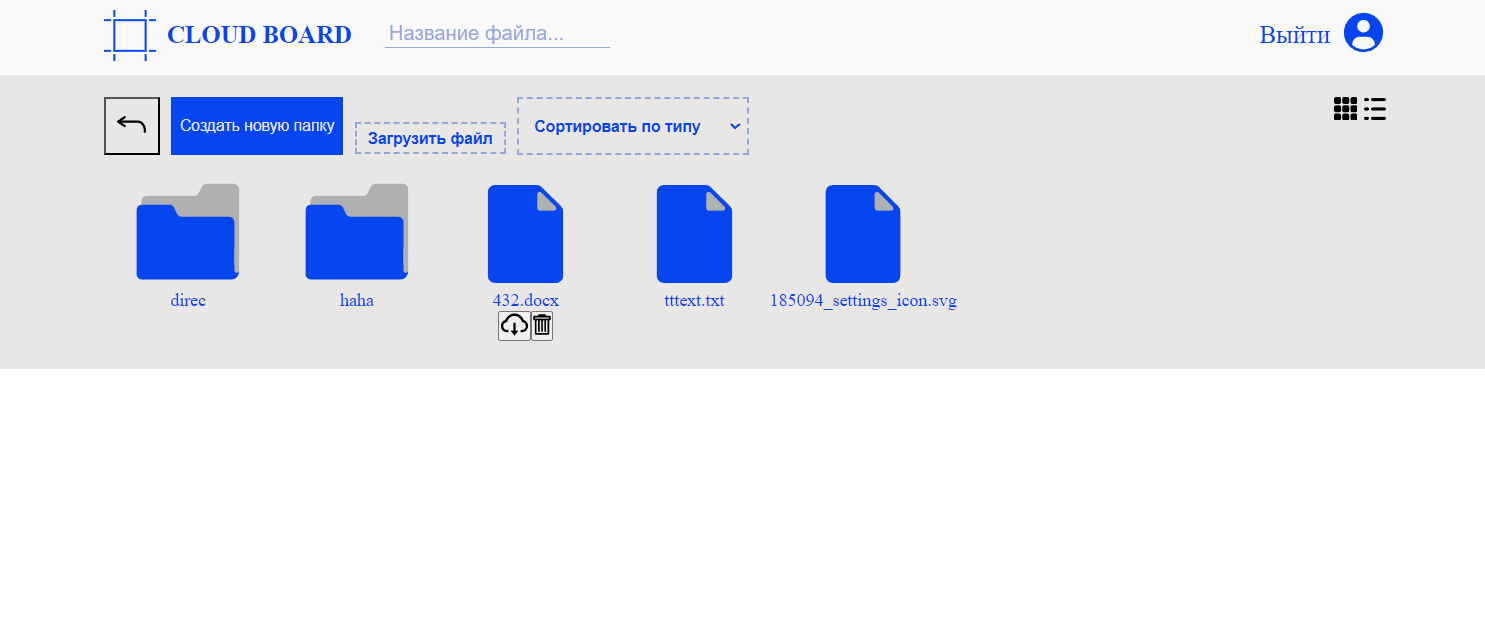
При нажатии на кнопку «Создать новую папку» появляется форма, которая позволяет пользователю ввести имя новой папки. На рисунке 15 показана работа функции «Создать новую папку».

  
Рисунок 15 – Работа функции создания папки

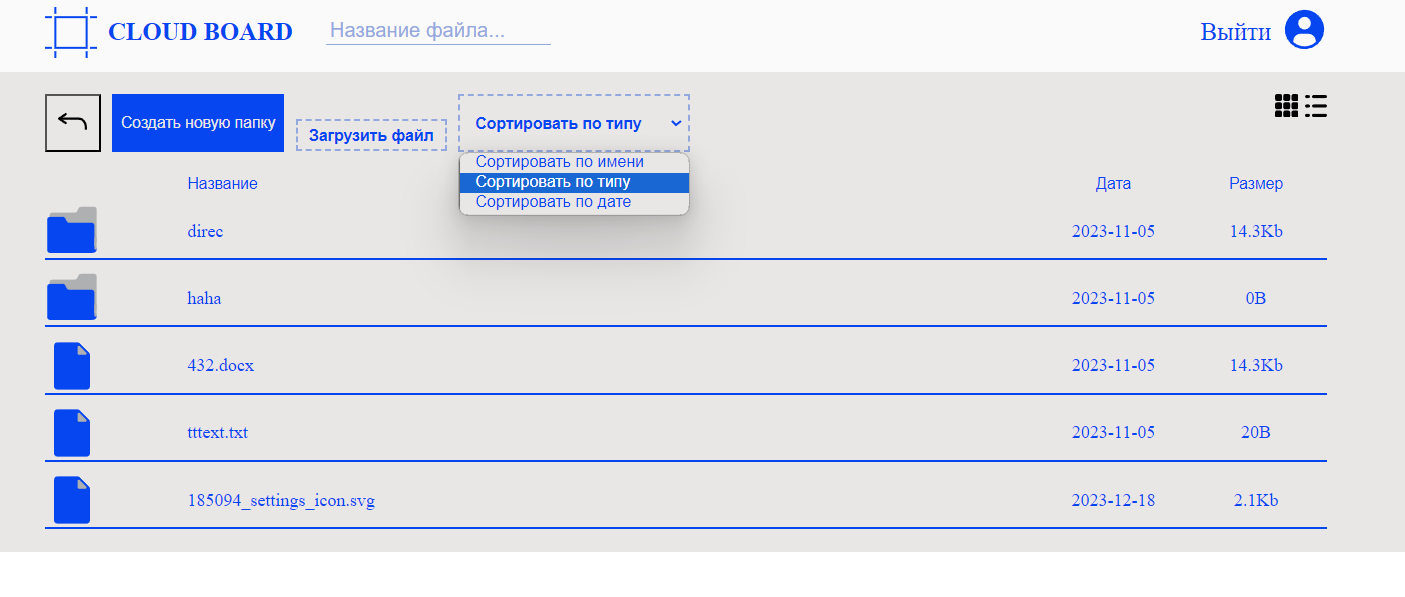
При нажатии на кнопку «Загрузить файл» открывается диалоговое окно. После выбора файла открывается окно, на котором отображается прогресс загрузки. На рисунке 16 приведена работа функции «Загрузить файл».

  
Рисунок 16 – Работа функции загрузки файла

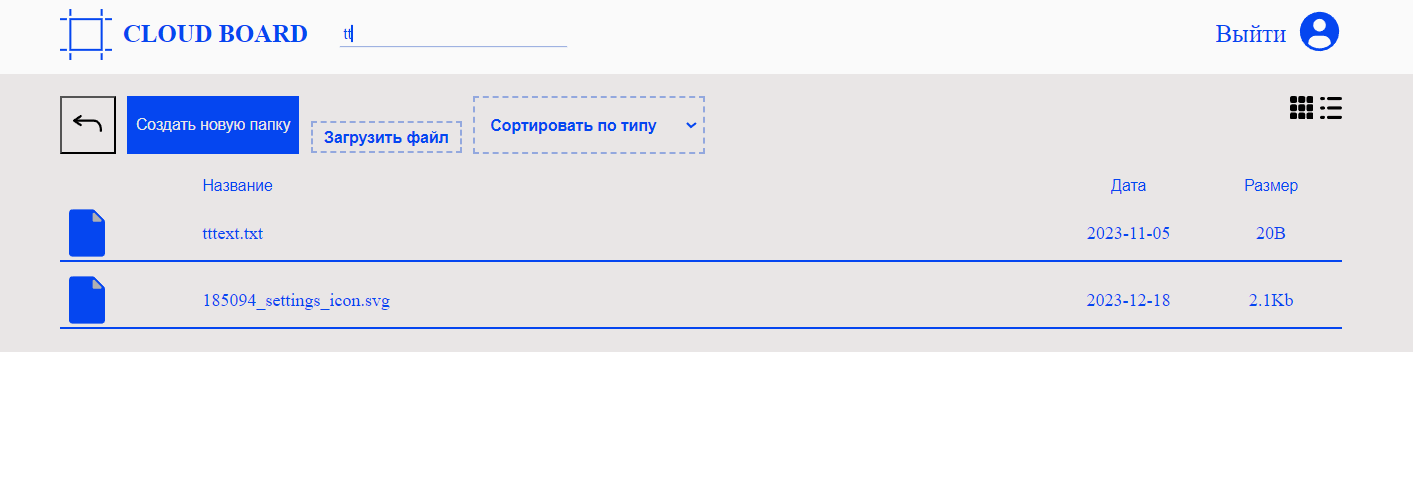
Также пользователь может менять отображение файлов и папок в хранилище. Для этого он должен выбрать один из двух стилей: «Плитка», «Список». Кнопки для выбора отображения находятся на правой стороне интерфейса. На рисунке 17 показано отображение файлов и папок в стиле «Плитка».

  
Рисунок 17 – Отображение файлов и папок

Также пользователь может сортировать папки и файлы по 3 параметрам: по имени, по типу, по дате. На рисунке 18 приведена работа функции сортировки.

  
Рисунок 18 – Работа функции сортировки

Также пользователю доступна функция поиска. Для этого необходимо ввести подстроку в поле поиска. На рисунке 19 приведена работа функции поиска.

  
Рисунок 19 – Работа функции поиска

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта было разработано облачное хранилище.

В первом разделе приведены основные понятия предметной области, описаны характеристики систем-аналогов, приведен их сравнительный анализ. На основе проведенного анализа сформулирована постановка задачи.

В разделе «Проектирование web-сайта» была построена логическая модель данных, описаны входные и выходные данные, перечислены исключительные ситуации, построена диаграмма вариантов использования и разработаны прототипы интерфейса пользователя.

В разделе «Реализация web-сайта» была описана разработка физической модели БД, реализация клиентской и серверной частей, и проведено тестирование системы.

Разработанная система может использоваться для удобного и надежного хранения данных. Она позволяет загружать, хранить и получать доступ к файлам и документам из любого места, где есть доступ к интернету.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Что такое облачное хранилище [Электронный ресурс]. URL: https://aws.amazon.com/ru/what-is/cloud-storage/ (дата обращения: 10.12.2023).

Предметная область: понятие и основные принципы [Электронный ресурс]. URL: https://uchet-jkh.ru/i/predmetnaya-oblast-ponyatie-i-osnovnye-principy/ (дата обращения: 10.12.2023).

Облачное хранилище [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/oblachnoe-hranilishe/ (дата обращения: 10.12.2023).

Масштабируемые хранилища данных [Электронный ресурс]. URL: https://habr.com/ru/articles/126580/ (дата обращения: 12.12.2023).

Как работаю базы данных [Электронный ресурс]. URL: https://practicum.yandex.ru/blog/chto-takoe-bazy-dannyh/ (дата обращения: 22.12.2023).

Хеширование: что такое хеш [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/heshirovanie/ (дата обращения: 12.12.2023).

Google Диск – обзор сервиса [Электронный ресурс]. URL: https://startpack.ru/application/google-drive#. (дата обращения: 12.12.2023).

Преимущества и недостатки Яндекс Диска [Электронный ресурс]. URL: https://islamnews.ru/preimushhestva-i-nedostatki-jandeks-diska (даты обращения: 12.12.2023).

Сервис Dropbox [Электронный ресурс]. URL: https://www.internet-technologies.ru/articles/newbie/servis-dropbox-stoprocentnaya-garantiya-bezopasnosti-dannyh.html (дата обращения: 14.12.2023).

Функциональная модель [Электронный ресурс]. URL: https://wiki5.ru/wiki/Function\_model (дата обращения: 14.12.2023).

Диаграмма вариантов использования [Электронный ресурс]. URL: https://itonboard.ru/analysis/629diagramma\_variantov\_ispolzovanija\_use\_case\_diagram/ (дата обращения: 14.12.2023).

Логическая схема данных [Электронный ресурс]. URL: https://docs.arenadata.io/adtm/Admin/Logical.html (дата обращения: 14.12.2023).

Построение и анализ алгоритмов обработки данных [Электронный ресурс]. URL: https://elar.urfu.ru/bitstream/10995/34782/1/978-5-7996-1489-8 (дата обращения: 14.12.2023).

Зачем нужно прототипирование интерфейсов [Электронный ресурс]. URL: https://inostudio.com/blog/articles-design/zachem-nuzhno-prototipirovanie-interfeysov/# (дата обращения: 14.12.2023).

Физическая модель базы данных [Электронный ресурс]. URL: https://neerc.ifmo.ru/wiki/index.php?title (дата обращения: 14.12.2023).

MySQL: основные достоинства и недостатки [Электронный ресурс]. URL: https://timeweb.cloud/blog/mysql-preimushchestva-i-nedostatki (дата обращения 14.12.2023).

Хранимые процедуры и функции [Электронный ресурс]. URL: https://wiki.dieg.info/xranimye\_procedury\_funkcii\_i\_triggery (дата обращения: 16.12.2023).

Разработка серверной части [Электронный ресурс]. URL: https://dzen.ru/a/ZPJt3skCSEafwlkq (дата обращения: 16.12.2023).

Разработка клиентской части [Электронный ресурс]. URL: https://science-engineering.ru/ru/article/view?id=1278 (дата обращения: 16.12.2023).

Что такое web-тестирование и как его проводить [Электронный ресурс]. URL: https://sky.pro/media/chto-takoe-veb-testirovanie-i-kak-ego-provodit/ (дата обращения: 16.12.2023).

ПРИЛОЖЕНИЕ А  
Листинг программы

import {applyMiddleware, combineReducers, createStore} from "redux";

import {composeWithDevTools} from 'redux-devtools-extension';

import thunk from "redux-thunk";

import userReducer from "./userReducer";

import fileReducer from "./fileReducer";

import uploadReducer from "./uploadReducer";

import appReducer from "./appReducer";

const rootReducer = combineReducers({

user: userReducer,

files: fileReducer,

upload: uploadReducer,

app: appReducer

})

export const store = createStore(rootReducer, composeWithDevTools(applyMiddleware(thunk)))

const SHOW\_LOADER = 'SHOW\_LOADER'

const HIDE\_LOADER = 'HIDE\_LOADER'

const defaultState = {

loader: false

}

export default function userReducer(state = defaultState, action) {

switch (action.type) {

case SHOW\_LOADER: return {...state, loader: true}

case HIDE\_LOADER: return {...state, loader: false}

default:

return state

}

}

export const showLoader = () => ({type: SHOW\_LOADER})

export const hideLoader = () => ({type: HIDE\_LOADER})

function cors(req, res, next) {

res.header("Access-Control-Allow-Origin", "\*");

res.header("Access-Control-Allow-Methods", "GET, PUT, PATCH, POST, DELETE");

res.header("Access-Control-Allow-Headers", "Content-Type, Authorization");

next();

}

module.exports = cors

const jwt = require('jsonwebtoken')

const config = require('config')

module.exports = (req, res, next) => {

if (req.method === 'OPTIONS') {

return next()

}

try {

const token = req.headers.authorization.split(' ')[1]

if (!token){

return res.status(401).json({message: 'Auth error'})

}

const decoded = jwt.verify(token, config.get('secretKey'))

req.user = decoded

next()

} catch (e) {

return res.status(401).json({message: 'Auth error'})

}

}

const Router = require("express");

const User = require("../models/User")

const bcrypt = require("bcryptjs")

const config = require("config")

const jwt = require("jsonwebtoken")

const{check, validationResult} = require("express-validator")

const router = new Router()

const authMiddleware = require('../middleware/auth.middleware')

const fileService = require('../services/fileService')

const File = require('../models/File')

router.post('/registration',

[

check('email', "Uncorrect email").isEmail(),

check('password', 'Password must be longer than 3 and shorter than 12').isLength({min:3, max:12})

],

async (req,res) => {

try{

const errors = validationResult(req)

if (!errors.isEmpty()){

return res.status(400).json({message:"Uncorrect requests", errors})

}

const{email, password} = req.body

const candidate = await User.findOne({email})

if(candidate){

return res.status(400).json({message: `User with email ${email} already exist`})

}

const hashPassword = await bcrypt.hash(password, 8)

const user = new User({email, password: hashPassword})

await user.save()

await fileService.createDir(new File({user: user.id, name: ''}))

return res.json({message: "User was created"})

} catch(e){

console.log(e)

res.send({message: "Server error"})

}

})

router.post('/login',

async (req,res) => {

try{

const {email, password} = req.body

const user = await User.findOne({email})

if(!user){

return res.status(404).json({message: "User not found"})

}

const isPassValid = bcrypt.compareSync(password, user.password)

if(!isPassValid){

return res.status(400).json({message: "Invalid password"})