|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***«Разработка базы данных веб-приложения для обмена файлами в сети Интернет»***

Студент \_\_\_\_\_\_ИУ7-65Б\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**Д.Р.Жигалкин **\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_**В.И.Солодовников**\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*Москва, 2021 г.*

**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**



УТВЕРЖДАЮ

Заведующий кафедрой \_\_ИУ7\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_И. В. Рудаков\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Базы данных\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Веб-приложение для обмена файлами в сети Интернет\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Тема курсового проекта)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Жигалкин Д.Р., гр. ИУ7-65Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

1. ***Техническое задание***

Разработать веб-приложение для обмена файлами с разграничением прав доступа, ведением истории файлообмена и действий пользователей. Выделить следующих акторов: гость, пользователь, модератор, администратор. Реализовать следующий функционал гостя: регистрация, просмотр списка загруженных файлов на сервер, скачивание файла с сервера. Реализовать следующий функционал пользователя: авторизация пользователя, загрузка файла на сервер, скачивание файла с сервера, написание комментария к файлу, изменение рейтинга пользователя, изменение рейтинга файла, просмотр списка загруженных файлов на сервер. Функционал модератора должен включать в себя функционал пользователя, а также возможность удаления любого комментария к файлу, удаление любого файла. Функционал администратора должен включать в себя функционал модератора, а также возможность удаления любого аккаунта(за исключением аккаунта администратора).

1. ***Оформление курсового проекта:***

2.1. Расчетно-пояснительная записка на 25-30 листах формата А4. Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

2.2. Перечень графического (иллюстративного) материала (плакаты, схемы, чертежи и т.п.).

На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |  |  |  |  |  |
| **Руководитель курсового проекта** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_В.И.Солодовников \_\_\_ | | | | |
|  |  |  |  |  |  |
|  | (Подпись, дата) |  | (И.О. Фамилия) | | |
| **Студент** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Д.Р.Жигалкин \_\_\_ | | | | |
|  |  |  |  |  | |
|  | (Подпись, дата) | а) | (И.О. Фамилия) | | |

Оглавление

[Введение 5](#_Toc74056911)

[1. Аналитическая часть 6](#_Toc74056912)

[1.1 Формализация задачи 6](#_Toc74056913)

[1.2 Акторы системы 6](#_Toc74056914)

[1.2.1 Гость 6](#_Toc74056915)

[1.2.2 Пользователь 7](#_Toc74056916)

[1.2.3 Модератор 7](#_Toc74056917)

[1.2.4 Администратор 7](#_Toc74056918)

[1.3 Сценарии использования 7](#_Toc74056919)

[1.4 Обзор существующих решений 11](#_Toc74056920)

[1.4.1 Dropbox 11](#_Toc74056921)

[1.4.2 Google Диск 12](#_Toc74056922)

[1.4.3 Microsoft OneDrive 12](#_Toc74056923)

[1.5 Анализ моделей данных 14](#_Toc74056924)

[1.5.1 Реляционная модель 15](#_Toc74056925)

[1.5.2 Иные подходы 15](#_Toc74056926)

[1.6 Выбор СУБД 16](#_Toc74056927)

[1.6.1 MSSQL 16](#_Toc74056928)

[1.6.2 MySQL 16](#_Toc74056929)

[1.6.3 PostgreSQL 17](#_Toc74056930)

[1.6.4 Oracle 18](#_Toc74056931)

[1.7 Требования к безопасности базы данных 18](#_Toc74056932)

[1.8 Выводы из аналитического раздела 18](#_Toc74056933)

[2. Конструкторская часть 19](#_Toc74056934)

[2.1 Проектирование базы данных 19](#_Toc74056935)

[2.1.1 Ролевая модель 21](#_Toc74056936)

[2.1.2 Хранимые процедуры 22](#_Toc74056937)

[2.1.3 Триггеры 26](#_Toc74056938)

[2.1.4 Обеспечение безопасности базы данных 27](#_Toc74056939)

[2.2 Нормализация базы данных 28](#_Toc74056940)

[2.2.1 Первая нормальная форма 28](#_Toc74056941)

[2.2.2 Вторая нормальная форма 28](#_Toc74056942)

[2.2.3 Третья нормальная форма 29](#_Toc74056943)

[2.3 Проектирование архитектуры 29](#_Toc74056944)

[2.4 Выводы из конструкторского раздела 31](#_Toc74056945)

[3. Технологическая часть 32](#_Toc74056946)

[3.1 Выбор и обоснование языка программирования и среды разработки 32](#_Toc74056947)

[3.2 Диаграмма базы данных 32](#_Toc74056948)

[3.3 UML-диаграммы компонентов 34](#_Toc74056949)

[3.4 Интерфейс веб-приложения 38](#_Toc74056950)

[3.5 Выводы из технологического раздела 42](#_Toc74056951)

[Заключение 43](#_Toc74056952)

[Список использованной литературы 44](#_Toc74056953)

[А Презентация 45](#_Toc74056954)

# Введение

Жизнь в современном мире сложно представить без обмена информацией между людьми. Для того, чтобы поделиться файлами используются специальные веб-приложения, будь то облачный сервис, торрент-трекер или файловый хостинг. Из всего этого разнообразия конечному пользователю необходимо выбрать определенный, удобный именно ему сервис.

Целью данной работы является реализация простого в использовании и в то же время функционального веб-приложения для обмена файлами в сети Интернет.

Для достижения поставленной цели необходимо решить следующие задачи:

1) формализовать задание, выделив соответствующих акторов и прецеденты;

2) провести анализ существующих решений, выделить преимущества и недостатки;

3) провести анализ СУБД и выбрать наиболее подходящую;

4) спроектировать базу данных;

5) спроектировать архитектуру приложения;

6) разработать приложение.

# Аналитическая часть

В данном разделе разделе рассматриваются возможные акторы системы и их сценарии использования приложения, существующие решения и производится выбор СУБД.

## 1.1 Формализация задачи

В ходе выполнения данной курсовой работы должно быть спроектировано и реализовано клиент-серверное приложение с поддержкой следующего функционала:

1) авторизация и регистрация пользователя;

2) просмотр списка файлов, поиск по имени файла, загрузка/скачивание файла;

3) комментирование файлов, оценивание файлов и пользователей;

4) удаление комментариев к файлу, удаление файлов;

5) удаление аккаунта пользователя и просмотр истории действий пользователей.

## 1.2 Акторы системы

В системе существуют следующие акторы:

1) гость;

2) пользователь;

3) модератор;

4) администратор.

Рассмотрим каждого актора по отдельности.

## 1.2.1 Гость

Гость — это неавторизованный пользователь, обладающий минимальным набором возможностей взаимодействия с системой. Он может авторизоваться, зарегистрироваться, просмотреть список загруженных файлов, использовать поиск файла по имени и скачать файл.

## 1.2.2 Пользователь

В функционал авторизированного пользователя входит загрузка файла на сервер, скачивание файла с сервера, поиск файла по названию, написание комментария к файлу, изменения рейтинга файлов и пользователей и просмотр списка загруженных файлов. Также присутствует возможность выйти из аккаунта.

## 1.2.3 Модератор

Модератор является пользователем с повышенным уровнем полномочий — он обладает всеми правами пользователя, а также может удалять комментарии и файлы других пользователей.

## 1.2.4 Администратор

Администратор — это пользователь, который обладает всеми правами модератора, а также может удалять аккаунты пользователей, за исключением аккаунтов администраторов, просматривать историю файлообмена.

## 1.3 Сценарии использования

Рассмотрим диаграммы использования. В них описываются действующие лица и прецеденты — конкретные действия, которые выполняет актор при работе с системой.

На рисунках 1.1 — 1.4 представлены диаграммы для каждого актора.

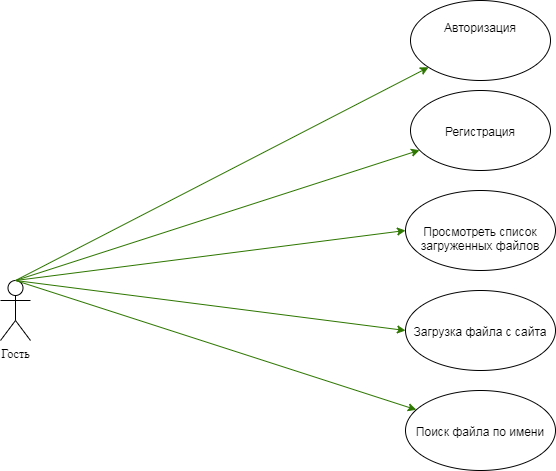


Рисунок 1.1 — Диаграмма сценариев использования приложения гостем.



Рисунок 1.2 — Диаграмма сценариев использования приложения пользователем.

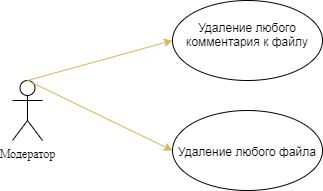


Рисунок 1.3 — Диаграмма сценариев использования приложения модератором.

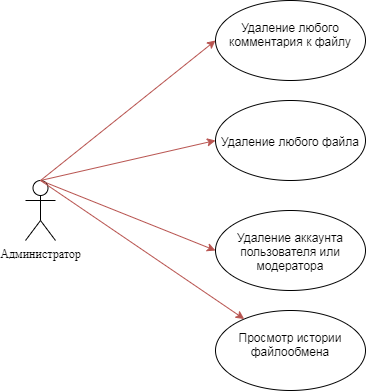


Рисунок 1.4 — Диаграмма сценариев использования приложения администратором.

## 1.4 Обзор существующих решений

В настоящее время существует большое число различных файловых хранилищ, которые решают вопрос передачи файлов разным людям. Рассмотрим некоторые из них.

## 1.4.1 Dropbox

Dropbox — файловый хостинг, включающий персональное облачное хранилище, синхронизацию файлов и программу-клиент.

Dropbox позволяет пользователям создать специальную папку на своих компьютерах, которая синхронизируется таким образом, что она имеет одинаковое содержимое независимо от того, какое устройство используется для просмотра. Файлы, размещённые в этой папке, также доступны через веб-сайт Dropbox и мобильные приложения. Dropbox работает по модели Freemium[1], в которой пользователи имеют возможность создать бесплатный аккаунт с заданным количеством свободного пространства, в то время как для увеличения объёма аккаунта необходима платная подписка.

На рисунке 1.5 показан интерфейс Dropbox.

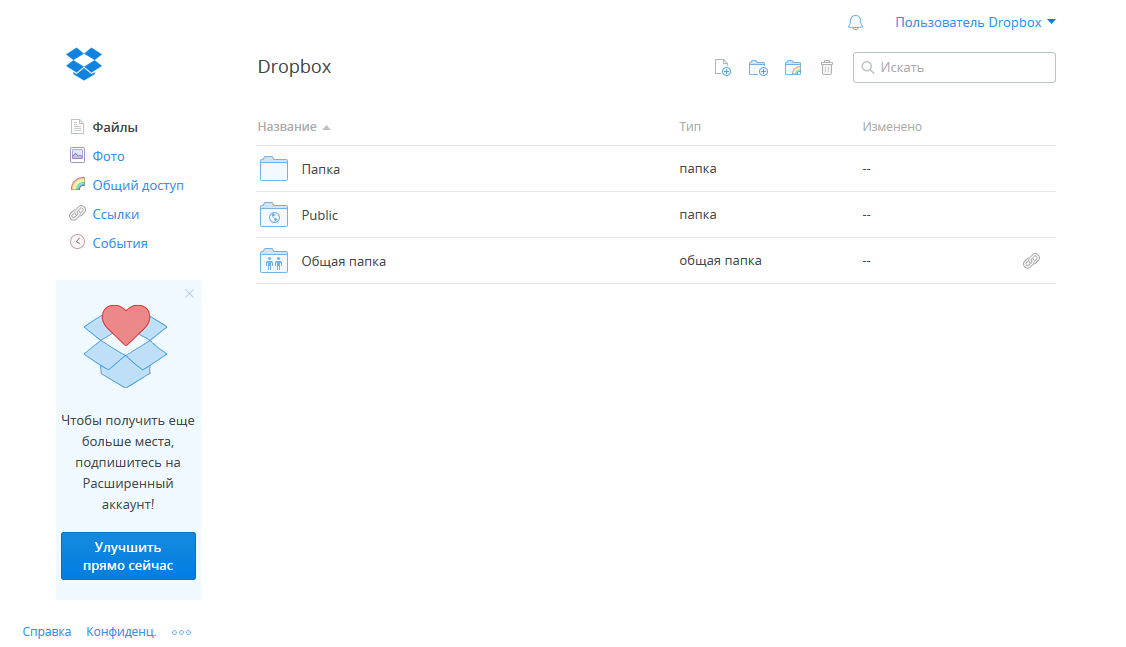


Рисунок 1.5 — Интерфейс Dropbox.

## 1.4.2 Google Диск

Google Диск — это сервис хранения, редактирования и синхронизации файлов, разработанный компанией Google.

Его функции включают [хранение файлов в Интернете](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D0%BE%D0%B5_%D1%85%D1%80%D0%B0%D0%BD%D0%B8%D0%BB%D0%B8%D1%89%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [общий доступ](https://ru.wikipedia.org/wiki/File_sharing) к ним и совместное редактирование. В состав Google Диска входят [Google Документы, Таблицы и Презентации](https://ru.wikipedia.org/wiki/Google_Docs) — набор офисных приложений для совместной работы над текстовыми документами, электронными таблицами, презентациями, чертежами, веб-формами и другими файлами. Общедоступные документы на Диске индексируются [поисковыми системами](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0). Для использования неограниченного пространства для хранения необходима премиальная подписка.

На рисунке 1.6 показан интерфейс Google Диск.

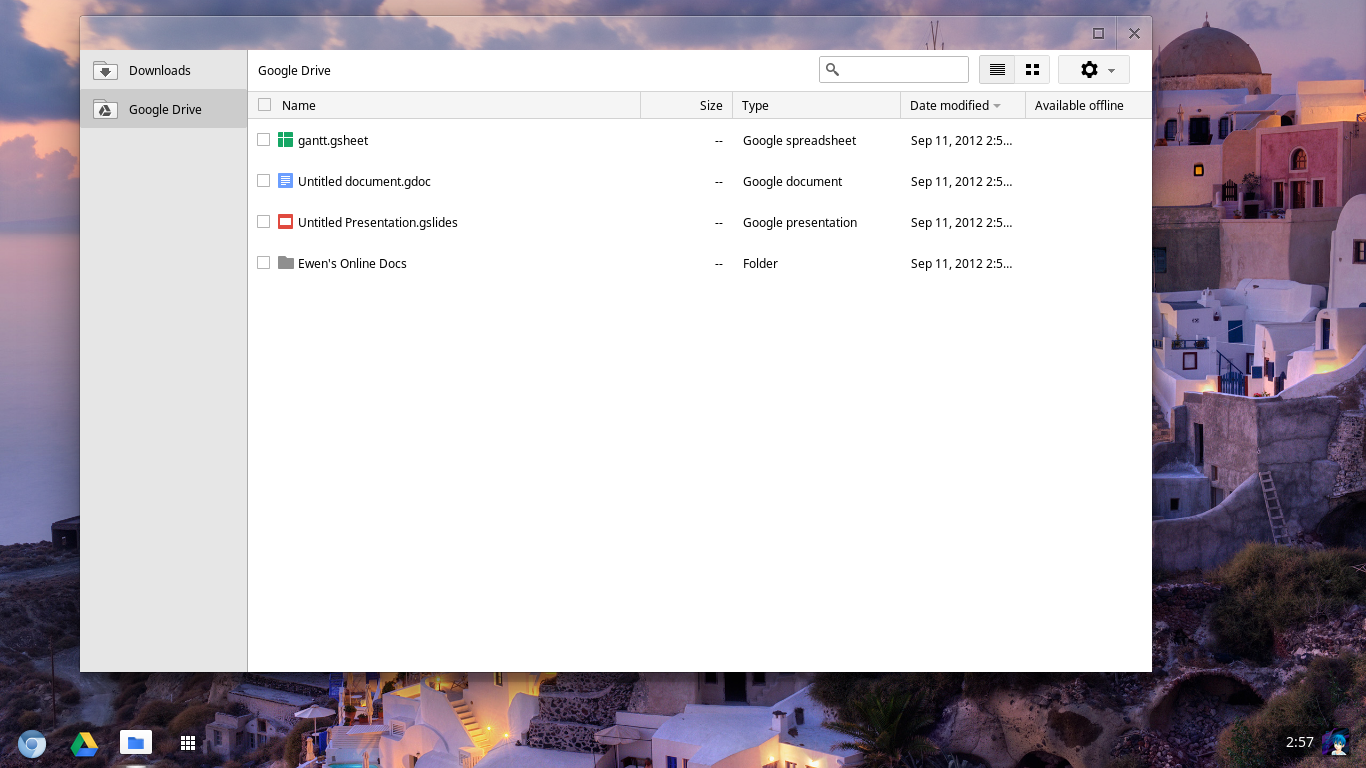


Рисунок 1.6 — Интерфейс Google Диск.

## 1.4.3 Microsoft OneDrive

OneDrive — [облачное хранилище](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F), созданное компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) в августе [2007 года](https://ru.wikipedia.org/wiki/2007_%D0%B3%D0%BE%D0%B4)[2]. Сервис OneDrive позволяет хранить до 5 ГБ информации бесплатно. Для изображений предусмотрен предварительный просмотр в виде эскизов, а также возможность их просмотра в виде слайдов. Есть недокументированный доступ по протоколу [WebDAV](https://ru.wikipedia.org/wiki/WebDAV)[3]. Для получения повышенной безопасности и увеличенного объема дискового пространства необходим доступ к премиум-возможностям.

На рисунке 1.7 показан интерфейс Microsoft OneDrive.

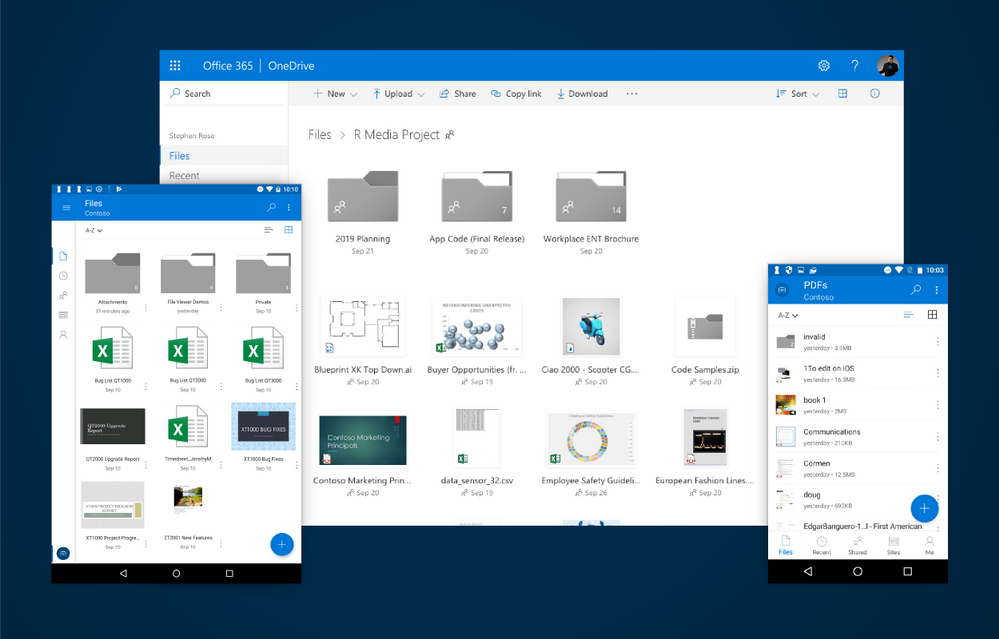


Рисунок 1.7 — Интерфейс Microsoft OneDrive.

Были выделены следующие критерии для сравнения функциональных возможностей существующих решений:

1) комментирование файлов;

2) выставление рейтинга пользователям;

3) выставление рейтинга файлам;

4) возможность увеличения файлового пространства;

5) удобный и простой интерфейс.

В таблице 1.1 представлено сравнение приложений по вышеперечисленным параметрам.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dropbox | Google Диск | Microsoft OneDrive |
| Комментирование файлов | + | - | + |
| Выставление рейтинга пользователям | - | - | - |
| Выставление рейтинга файлам | - | - | - |
| Возможность увеличения файлового пространства | + | + | + |
| Удобный и простой интерфейс | + | + | - |

Таблица 1.1 — Сравнение возможностей приложений.

Как видно из таблицы выше, тема создания простого в управлении, с точки зрения пользователя, и функционального приложения для обмена файлами весьма актуальна.

## 1.5 Анализ моделей данных

Существует много различных типов моделей баз данных, которые представляют средства структурирования данных. Рассмотрим некоторые из них.

## 1.5.1 Реляционная модель

Представленная в 70-х, реляционная модель предлагает структурированное хранение данных, что позволило создать стандарт языка структурированных запросов (sql). Отношения дают возможность группировки данных как связанных наборов, представленных в виде таблиц, содержащих упорядоченную информацию и соотносящих значения и атрибуты. Для расширения структуры таблицы в реляционной модели необходимо создавать таблицы расширений, связанные отношением «один-к-одному» с главной таблицей. Такое расширение в случае иерархии таблиц является выделением подтипов. В случае множественного подчинения речь идёт об агрегации.

## 1.5.2 Иные подходы

NoSQL-способ структуризации данных заключается в избавлении от ограничений при хранении и использовании информации. Базы данных NoSQL, используя неструктуризированный подход, предлагают много эффективных способов обработки данных в отдельных случаях.

В таблице 1.2 представлено сравнение двух этих подходов.

|  |  |  |
| --- | --- | --- |
|  | SQL | NoSQL |
| Структура и тип хранящихся данных | Однозначно определенная структура хранения данных | Ограничений нет |
| Запросы | Получение данных при помощи языка SQL | Каждая NoSQL база данных реализует свой способ работы с данными |
| Масштабируемость | Не подходит для постоянно меняющихся структур данных | Подходит для постоянно меняющихся структур данных |
| Поддержка | Развитая СУБД, наличие большого сообщества вокруг неё, множество примеров | Только недавно стала популярной |
| Объединение таблиц | Применение join | Отсутствие join |

Таблица 1.2 — Сравнение моделей БД.

## 1.6 Выбор СУБД

Одними из самых популярных реляционных систем управления базами данных сейчас являются MSSQL, MySQL, PostgreSQL, Oracle.

## 1.6.1 MSSQL

Microsoft SQL Server – разработанная корпорацией майкрософт, система управления реляционными базами данных.

Преимущества

— простота: MSSQL легко устанавливается и используется;

— возможность регулировки и отслеживания уровня производительности, чтобы уменьшить загрузку;

— возможность интеграции с другими продуктами Microsoft;

— визуализация на мобильных устройствах.

Недостатки

— высокая стоимость продукта для юридических лиц;

— возможны проблемы в работе служб интеграции импорта файлов;

— высокая ресурсоемкость SQL Server.

## 1.6.2 MySQL

MySQL – это самая популярная из всех крупных серверных баз данных. Разобраться в ней просто, имеется большое количество информации.

Преимущества

— простота: MySQL легко устанавливается и используется;

— много функций: MySQL поддерживает большую часть функционала SQL;

— безопасность: в MySQL встроено много функций безопасности;

— мощность и масштабируемость: MySQL может работать с действительно большими объёмами данных, и неплохо походит для масштабируемых приложений;

— скорость: пренебрежение некоторыми стандартами позволяет MySQL работать производительнее.

Недостатки

— ограничения функциональности, так как не полностью реализованы SQL-стандарты;

— надёжность: некоторые операции реализованы менее надёжно, чем в других реляционных системах управления базами данных;

## 1.6.3 PostgreSQL

PostgreSQL – это реляционная система управления базами данных, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость.

Преимущества

— полная SQL-совместимость;

— cообщество: PostgreSQL поддерживается опытным сообществом 24/7;

— поддержка сторонними организациями;

— расширяемость: PostgreSQL можно программно расширить за счёт хранимых процедур.

Недостатки

— производительность: в простых операциях чтения PostgreSQL может уступать своим аналогам.

## 1.6.4 Oracle

Oracle – это объектно-реляционная система управления базами данных.

Преимущества

— поддержка огромных баз данных и большого числа пользователей;

— быстрая обработка транзакций;

— большой и постоянно развивающийся функционал.

Недостатки

— высокая стоимость;

— требуются значительные вычислительные ресурсы.

MySQL выигрывает по многим параметрам и бесплатно распространяется. Поэтому было принято решение использовать именно эту реляционную систему управления базами данных.

## 1.7 Требования к безопасности базы данных

Безопасность базы данных — одного из основных требований, предъявляемых к СУБД. Для выполнения данного требования необходимо реализовать ролевую модель на уровне базы данных, в которой будут описаны пользователи и разрешенные им действия. Созданную ролевую модель необходимо испытать, подключаясь к БД с различными учетными именами, проверяя работу разрешений и выполняя аудит. Также необходимо использовать систему защиты от внешних подключений к БД.

## 1.8 Выводы из аналитического раздела

В данном разделе была проанализирована поставленная задача, выделены акторы и прецеденты, рассмотрены существующие аналоги приложений. Также были рассмотрены разные типы систем управления базами данных и сформулированы требования к безопасности. В качестве используемой в данной работе была выбрана реляционная система управления базами данных MySQL.

# 2. Конструкторская часть

В данном разделе будут спроектированы база данных и приложение.

## 2.1 Проектирование базы данных

Необходимо выделить сущности предметной области и построить ER-диаграмму. Диаграмма представлена на рисунках 2.1(а) — 2.1(б). Таблица 2.1 содержит описание сущностей базы данных.

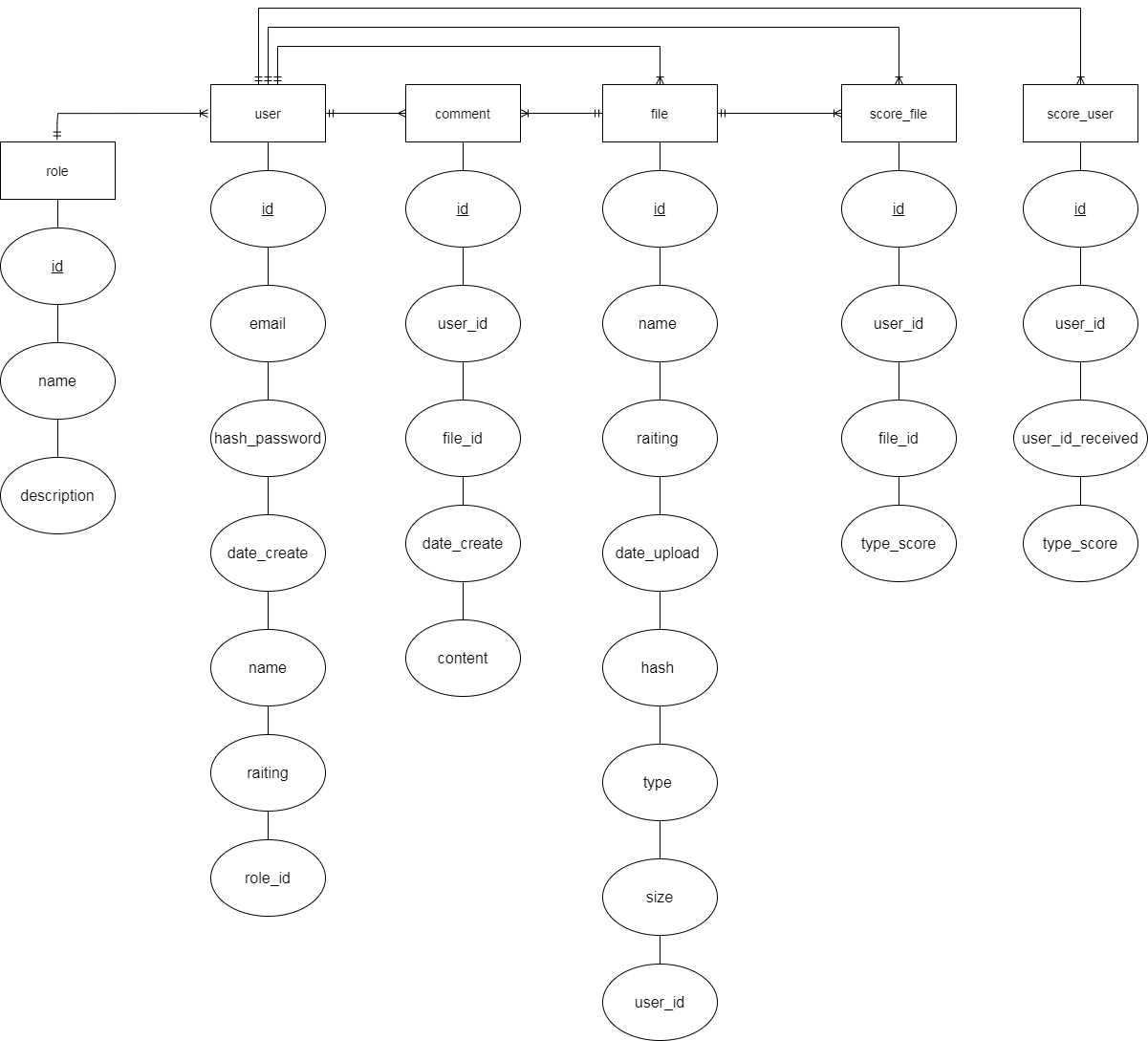


Рисунок 2.1(а) — ER-диаграмма.

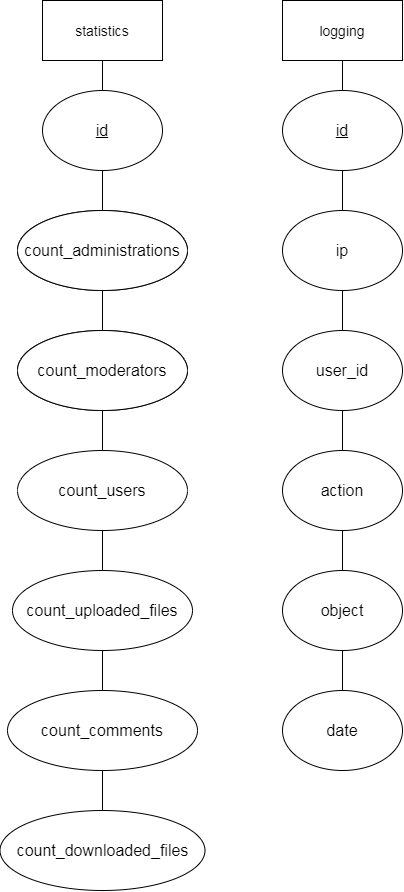


Рисунок 2.1(б) — ER-диаграмма.

|  |  |
| --- | --- |
| Название таблицы | Описание |
| user | Пользователи |
| file | Файлы |
| role | Роли пользователей |
| comment | Комментарии к файлам |
| score\_file | Оценки пользователей к файлам |
| score\_user | Оценки пользователей другим пользователям |
| logging | История действий пользователей |
| statistics | Статистика сайта |

Таблица 2.1 — Описание сущностей базы данных.

## 2.1.1 Ролевая модель

При проектировании базы данных следует учитывать, что на её уровне должна быть реализована ролевая модель — пользователи и разрешенные им действия над базой данных. Это является одной из мер безопасности, находящейся в компетенции СУБД. Всего необходимо выделить 4 пользователя, в соответствии с use-case диаграммой:

1) гость;

2) пользователь;

3) модератор;

4) администратор.

При реализации данных ролей нужно учесть особенности предметной области, такие как комментирование файлов, загрузку файлов на сервер и тому подобное.

На рисунке 2.2 можно увидеть выделенных пользователей и их права.

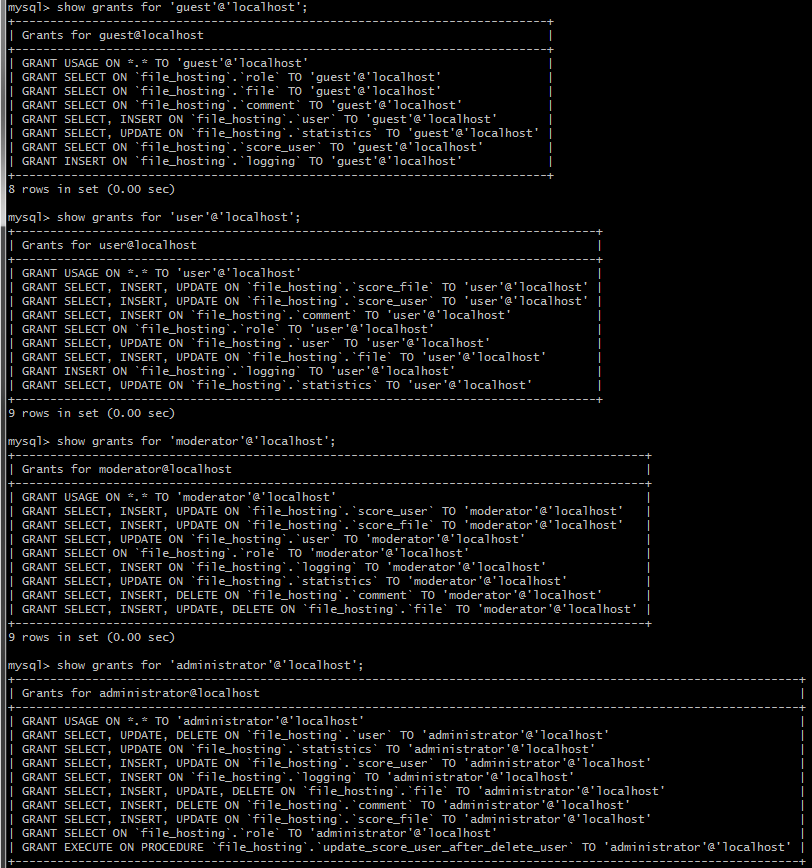


Рисунок 2.2 — Ролевая модель на уровне базы данных.

## 2.1.2 Хранимые процедуры

Прежде чем приступать к реализации триггеров, необходимо разработать хранимые процедуры, которые будут в них использоваться. Необходимы хранимые процедуры, имеющие следующий функционал:

1) пересчёт и обновление статистики пользователей — update\_count\_users;

2) пересчёт и обновление рейтинга пользователей — update\_score\_files;

3) пересчёт и обновление рейтинга файлов — update\_score\_users.

В листинге 2.1 представлен код хранимой процедуры update\_count\_users.

1. CREATE PROCEDURE `update\_count\_users`()
2. BEGIN
3. declare number\_users, number\_moderators, number\_administrators, score INT;
4. SELECT COUNT(\*) INTO number\_users FROM user WHERE role\_id = 1;
5. SELECT COUNT(\*) INTO number\_moderators FROM user WHERE role\_id = 2;
6. SELECT COUNT(\*) INTO number\_administrators FROM user WHERE role\_id = 3;
8. UPDATE statistics SET count\_users = number\_users where statistics.id = 1;
9. UPDATE statistics SET count\_moderators = number\_moderators where statistics.id = 1;
10. UPDATE statistics SET count\_administrators = number\_administrators where statistics.id = 1;
11. END

Листинг 2.1 — Хранимая процедура update\_count\_users.

В листинге 2.2 представлен код хранимой процедуры update\_score\_files.

1. CREATE PROCEDURE `update\_score\_files `(in id int)
2. begin
3. declare received\_id, score INT;
4. declare my\_count INT;
5. DECLARE cur1 CURSOR FOR SELECT file\_id, type\_score FROM score\_file WHERE user\_id = id;
6. SELECT COUNT(\*) INTO my\_count FROM score\_file WHERE user\_id = id;
7. OPEN cur1;
8. while my\_count > 0 do
9. set my\_count = my\_count - 1;
10. FETCH cur1 INTO received\_id, score;
11. if score > 0 then
12. UPDATE file SET raiting = raiting - 1 where file.id = received\_id;
13. else
14. UPDATE file SET raiting = raiting + 1 where file.id = received\_id;
15. end if;
16. end while;
17. CLOSE cur1;
18. end

Листинг 2.2 — Хранимая процедура update\_score\_files.

В листинге 2.3 представлен код хранимой процедуры update\_score\_users.

1. CREATE PROCEDURE `update\_score\_users `(in id int)
2. begin
3. declare received\_id, score INT;
4. declare my\_count INT;
5. DECLARE cur1 CURSOR FOR SELECT user\_id\_received, type\_score FROM score\_user WHERE user\_id = id;
6. SELECT COUNT(\*) INTO my\_count FROM score\_user WHERE user\_id = id;
7. OPEN cur1;
8. while my\_count > 0 do
9. set my\_count = my\_count - 1;
10. FETCH cur1 INTO received\_id, score;
11. if score > 0 then
12. UPDATE user SET raiting = raiting - 1 where user.id = received\_id;
13. else
14. UPDATE user SET raiting = raiting + 1 where user.id = received\_id;
15. end if;
16. end while;
18. CLOSE cur1;
19. end

Листинг 2.3 — Хранимая процедура update\_score\_users.

В реализации хранимых процедур update\_score\_files и update\_score\_users были использованы курсоры — поименованная область памяти, содержащая результирующий набор select запроса, позволяет обрабатывать полученные данные построчно.

В силу ограничений, накладываемых системой управления базами данных, пересчет рейтинга других пользователей перед удалением пользователя нельзя реализовать в триггере, поэтому соответствующая хранимая процедура будет вызвана в форме запроса к базе данных из приложения.

## 2.1.3 Триггеры

Для обеспечения необходимой функциональности базы данных, целесообразно добавить несколько триггеров для решения поставленных задач:

1) триггер, который будет пересчитывать рейтинг файлов с помощью хранимой процедуры, в случае удаления пользователя;

2) триггер, который будет пересчитывать рейтинг других пользователей с помощью хранимой процедуры, в случае удаления пользователя;

3) триггер, который будет обновлять таблицу статистики, когда будет происходить добавление или обновление пользователя.

Были реализованы триггеры следующих типов:

1) after insert;

2) after update;

3) before delete.

Триггеры after insert и after update на таблице user используются для обновления статистики. В листинге 2.4(а) — 2.4(б) представлен код данных триггеров.

1. CREATE TRIGGER `file\_hosting`.`user\_after\_insert` AFTER INSERT ON `user` FOR EACH ROW
2. BEGIN
3. call update\_count\_users();
4. END

Листинг 2.4(а) — Триггер after insert на таблице user.

1. CREATE TRIGGER `file\_hosting`.`user\_after\_update` AFTER UPDATE ON `user` FOR EACH ROW
2. BEGIN
3. call update\_count\_users();
4. END

Листинг 2.4(б) — Триггер after update на таблице user.

Последний триггер before delete на таблице user используется для вызова хранимой процедуры — пересчет рейтинга файлов, которые оценил пользователь. В листинге 2.5 представлен код данного триггера.

1. CREATE TRIGGER `file\_hosting`.`user\_before\_delete` BEFORE DELETE ON `user` FOR EACH ROW
2. BEGIN
3. call update\_score\_file\_before\_delete\_user(OLD.id);
4. END

Листинг 2.5 — Триггер before delete на таблице user.

## 2.1.4 Обеспечение безопасности базы данных

Была реализована ролевая модель — базовый уровень безопасности, но также необходимо решить вопрос с внешними подключениями к базе данных, поскольку открытая для внешних подключений база представляет собой мишень для злоумышленников. На начальных этапах разработки БД данный уровень не был реализован, в результате чего база данных была взломана, все таблицы удалены и оставлено следующее сообщение: "All your data is a backed up. You must pay 0.15 BTC to 1Jpw3G5tpns3g9CyvbjQU7TaJK2Vr7hNba 48 hours for recover it. After 48 hours expiration we will leaked and exposed all your data." В результате проделанной работы, были запрещены все внешние подключения.

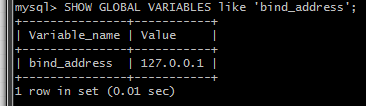


Рисунок 2.3 — Значение переменной, отвечающей за внешние подключения.

## 2.2 Нормализация базы данных

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

## 2.2.1 Первая нормальная форма

Чтобы база данных находилась в 1 нормальной форме, необходимо чтобы ее таблицы соблюдали следующие реляционные принципы:

— в таблице не должно быть дублирующих строк;

— в каждой ячейке таблицы хранится атомарное значение (одно не составное значение);

— в столбце хранятся данные одного типа;

— отсутствуют массивы и списки в любом виде.

## 2.2.2 Вторая нормальная форма

Чтобы база данных находилась во второй нормальной форме (2NF), необходимо чтобы ее таблицы удовлетворяли следующим требованиям:

— таблица должна находиться в первой нормальной форме;

—- таблица должна иметь ключ;

— все неключевые столбцы таблицы должны зависеть от полного ключа (в случае если он составной).

## 2.2.3 Третья нормальная форма

Требование третьей нормальной форме (3NF) заключается в том, чтобы таблицы были во второй нормальной форме и в таблицах отсутствовала транзитивная зависимость.

Транзитивная зависимость — это когда неключевые столбцы зависят от значений других неключевых столбцов.

Чтобы нормализовать базу данных до третьей нормальной формы, необходимо сделать так, чтобы в таблицах отсутствовали неключевые столбцы, которые зависят от других неключевых столбцов.

Спроектированная база данных удовлетворяет требованиям третьей нормальной формы.

## 2.3 Проектирование архитектуры

Необходимо спроектировать компоненты приложения. На рисунке 2.2 представлена диаграмма компонентов.

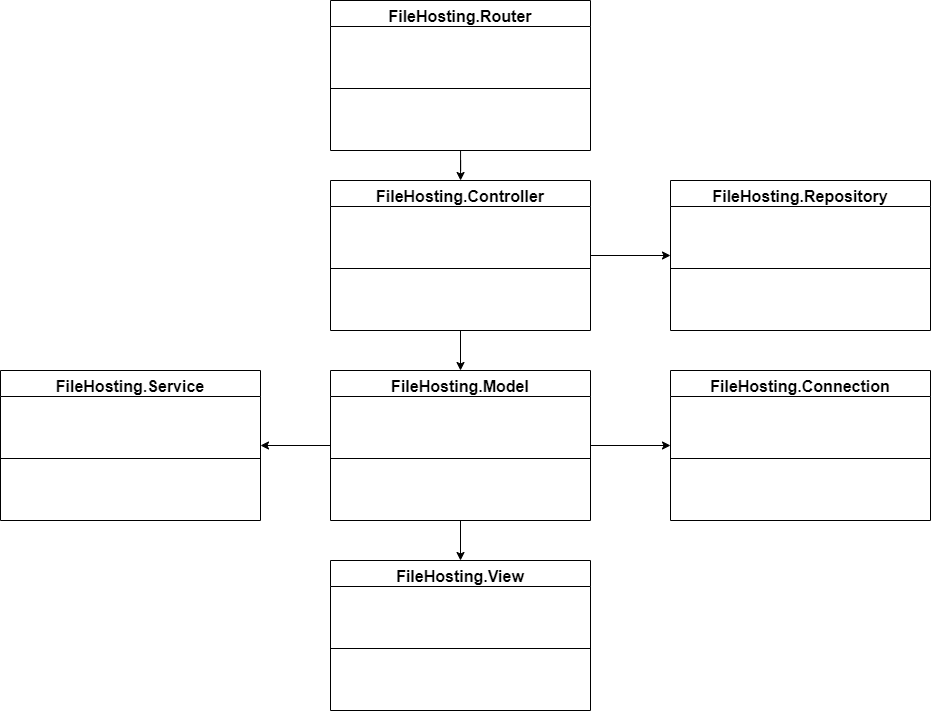


Рисунок 2.4 — Диаграмма компонентов приложения.

1. FileHosting.Router — точка входа в проект, инициализирует контроллеры и вызывает соответствующие методы.

2. FileHosting.Controller — компонент, являющийся промежуточным слоем между бизнес-логикой и представлением данных.

3. FileHosting.Repository — компонент, в котором описывается реализация репозиториев. Паттерн репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным слоем между классами, непосредственно взаимодействующими с данными, и основными классами программы.

Преимущества использования данного паттерна:

— разделение логики (обращаемся только к тем данным, которые нужны);

— абстрагирование от способа хранения данных;

— легкость тестирования (можно передавать заглушку репозитория при тестировании бизнес-логики).

4. FileHosting.Model — компонент, содержащий всю основную бизнес-логику приложения.

Функциональность данного компонента:

— регистрация, авторизация, удаление пользователей;

— скачивание, загрузка, поиск, просмотр, удаление файлов;

— добавление/изменение оценок пользователям, файлам;

— добавление/удаление комментариев к файлам.

5. FileHosting.Service — компонент содержащий дополнительные сервисы приложения.

Функциональность данного компонента:

— ведение истории файлообмена;

— ведение статистики.

6. FileHosting.Connection — компонент, отвечающий за конфигурирование и подключение базы данных к приложению.

7. FileHosting.View — компонент, отвечающий за визуальное представление информации.

## 2.4 Выводы из конструкторского раздела

В данном разделе были спроектированы база данных и архитектура приложения.

# 3. Технологическая часть

В соответствии с поставленной задачей, необходимо выбрать средства реализации, создать компоненты, описать порядок работы.

## 3.1 Выбор и обоснование языка программирования и среды разработки

В качестве языка программирования был выбран PHP[4] поскольку данный язык программирования динамично развивается и является объектно-ориентированным, что позволит в полной мере использовать знания, полученные на курсах объектно-ориентированное программирование и проектирование программного обеспечения.

В качестве среды разработки была выбрана Microsoft Visual Studio Code[5] по следующим причинам:

* она бесплатна в пользовании студентами;
* большое количество плагинов;
* она имеет множество удобств, которые облегчают процесс написания и отладки кода.

Для работы с базой данных был выбран RedBean ORM(object-relational mapping)[6], поскольку он позволяет абстрагироваться от конкретной системы управления базами данных и работать на более высоком уровне.

## 3.2 Диаграмма базы данных

На рисунках 3.1(а) — 3.1(б) представлена диаграмма базы данных.

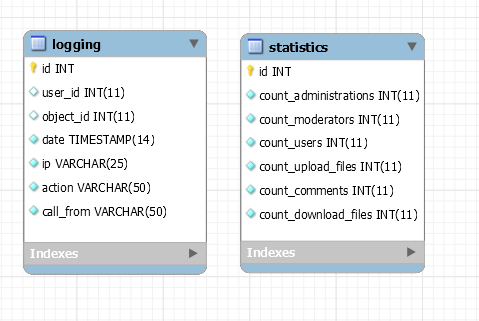


Рисунок 3.1(а) — Диаграмма базы данных.

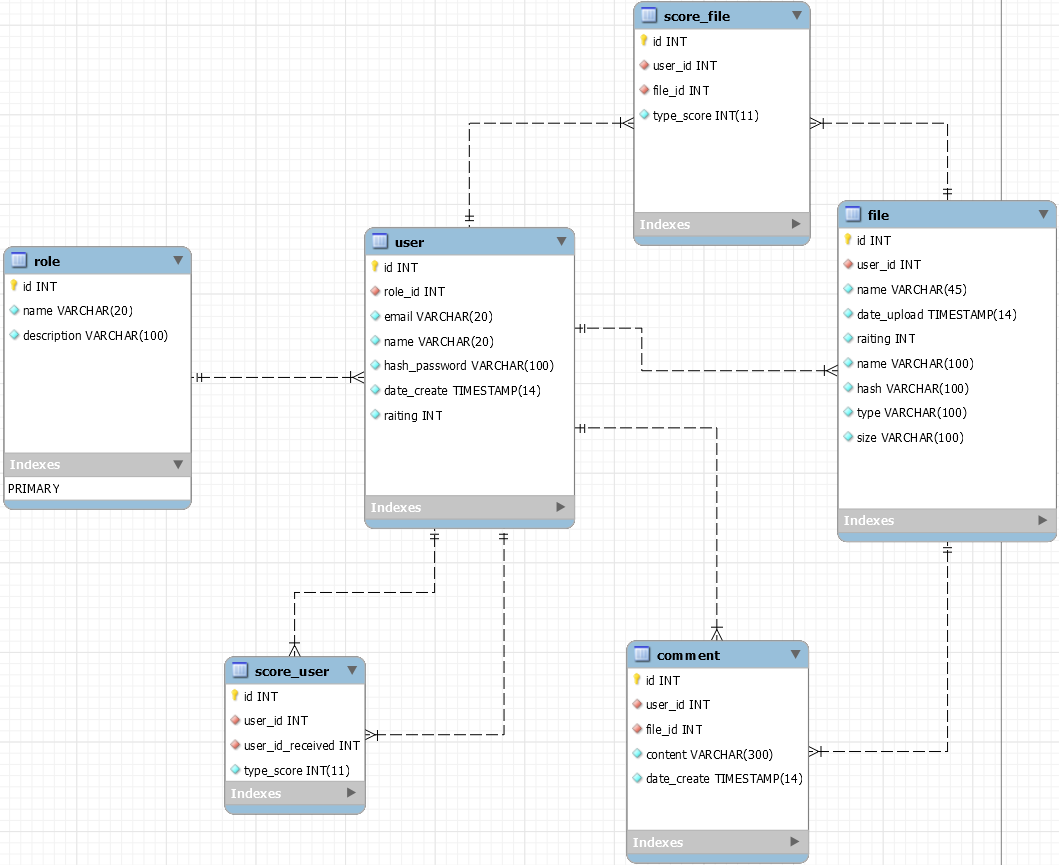


Рисунок 3.1(б) — Диаграмма базы данных.

## 3.3 UML-диаграммы компонентов

На рисунке 3.2 представлена UML-диаграмма класса, который содержит один метод, вызывающий соответствующий контроллер, компонента FileHosting.Router.

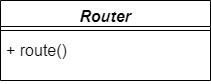


Рисунок 3.2 — UML-диаграмма класса компонента FileHosting.Router.

На рисунке 3.3 представлена UML-диаграмма классов компонента FileHosting.Controller.

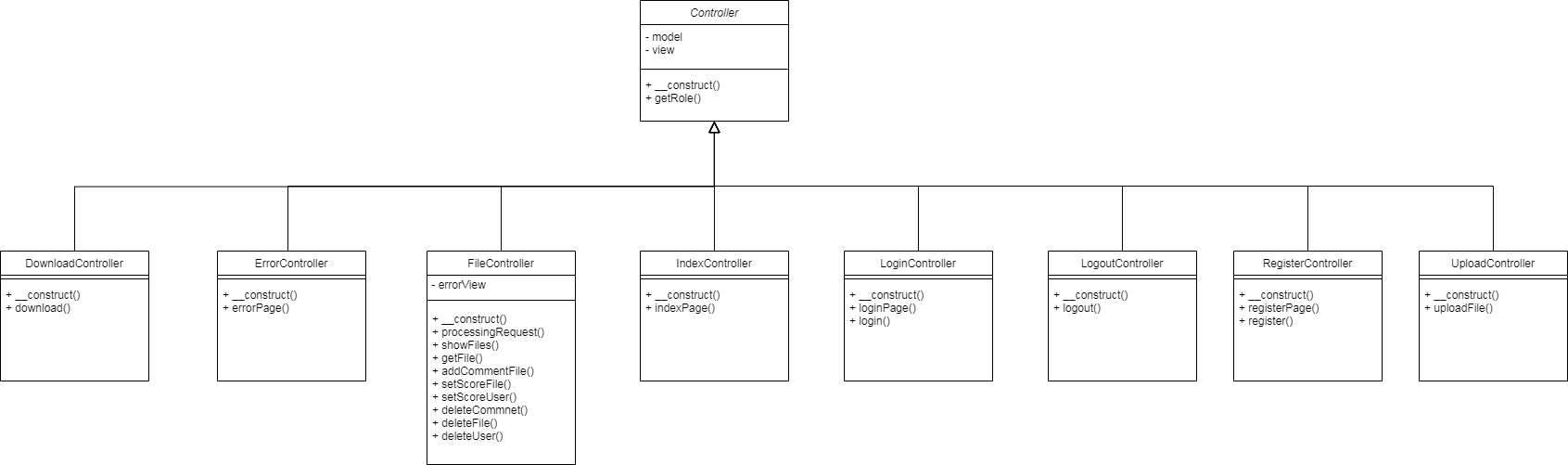


Рисунок 3.3 — UML-диаграмма класса компонента FileHosting.Controller.

На рисунках 3.4(а) — 3.4(б) представлена UML-диаграмма классов компонента FileHosting.Repository.

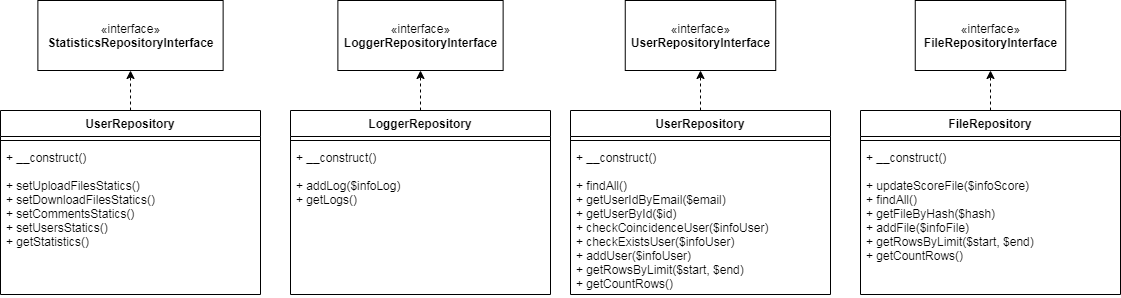


Рисунок 3.4(а) — UML-диаграмма классов компонента FileHosting.Repository.

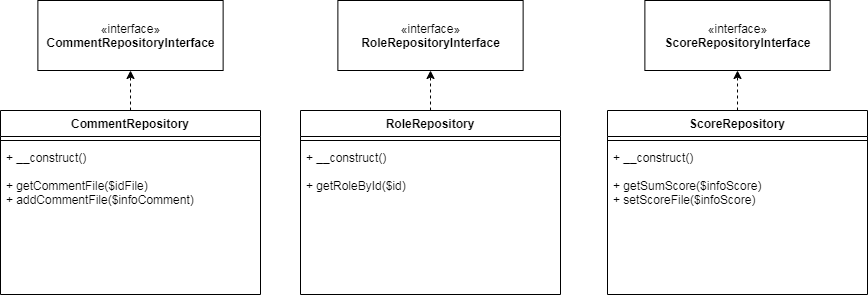


Рисунок 3.4(б) — UML-диаграмма классов компонента FileHosting.Repository.

На рисунке 3.5 представлена UML-диаграмма классов компонента FileHosting.Model.

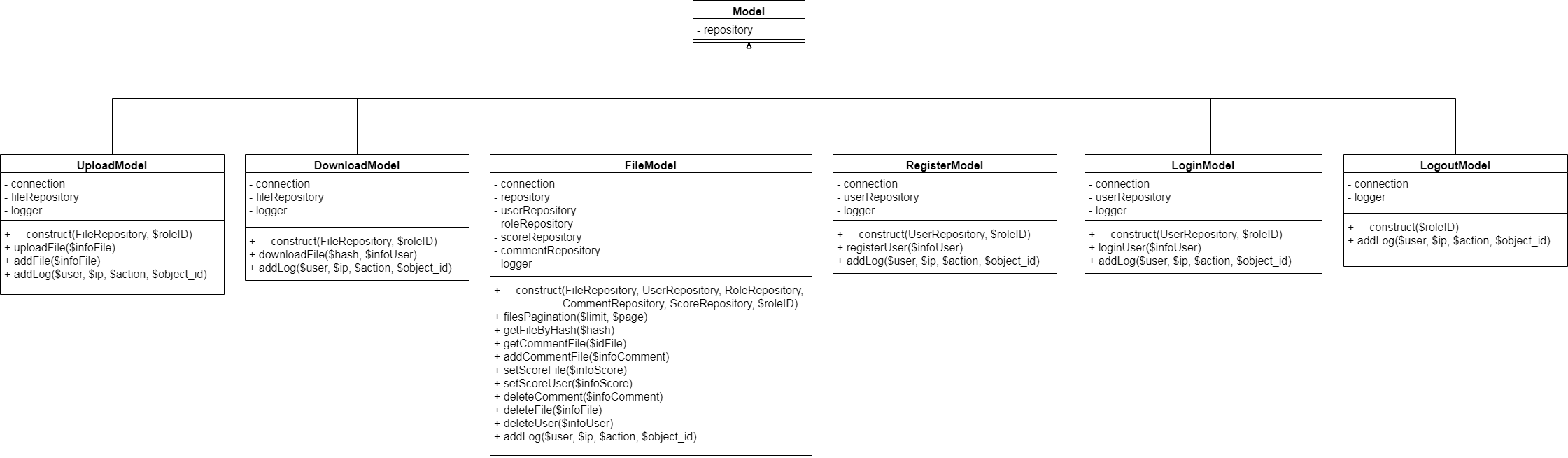


Рисунок 3.5 — UML-диаграмма классов компонента FileHosting.Model.

На рисунке 3.6 представлена UML-диаграмма классов компонента FileHosting.Service.

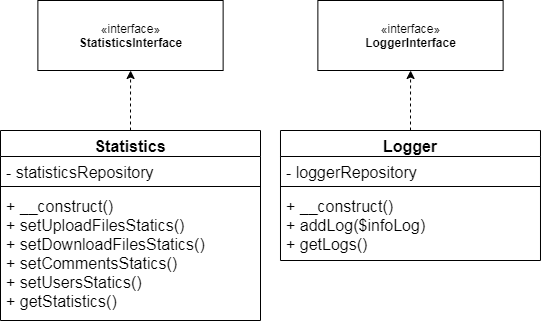


Рисунок 3.6 — UML-диаграмма классов компонента FileHosting.Service.

На рисунке 3.7 представлена UML-диаграмма классов компонента FileHosting.Connection.

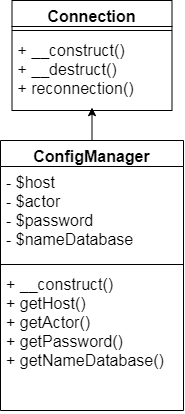


Рисунок 3.7 — UML-диаграмма классов компонента FileHosting.Connection.

На рисунке 3.8 представлена UML-диаграмма классов компонента FileHosting.View.

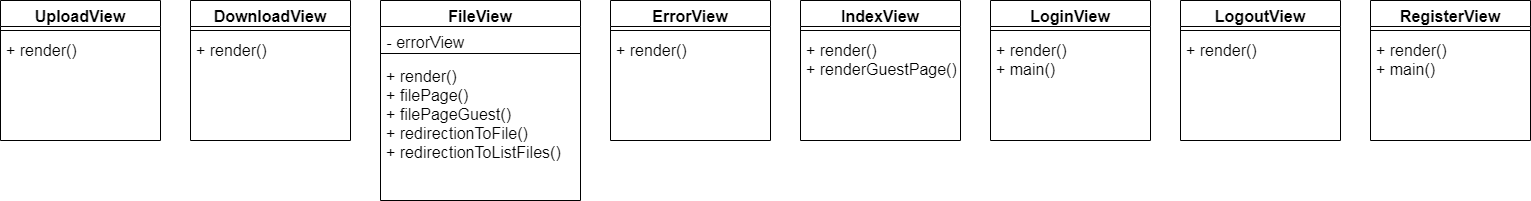
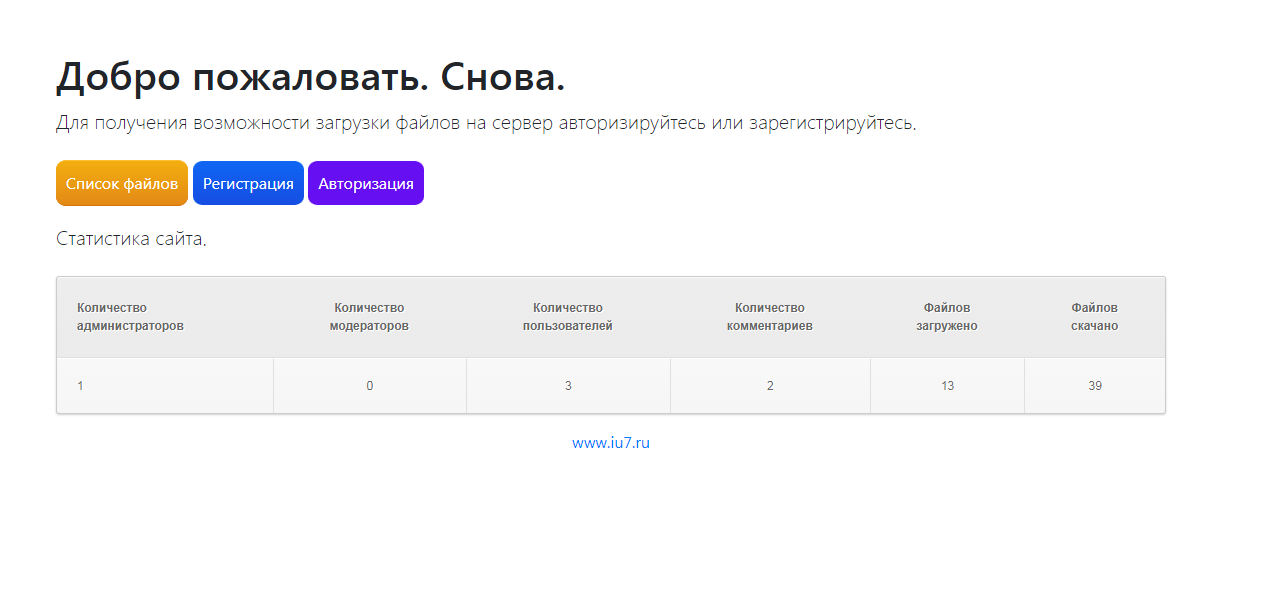


Рисунок 3.8 — UML-диаграмма классов компонента FileHosting.View.

## 3.4 Интерфейс веб-приложения

На рисунках ниже показан интерфейс различных страниц веб-приложения.

Рисунок 3.9 — Интерфейс главной страницы для неавторизованного пользователя.

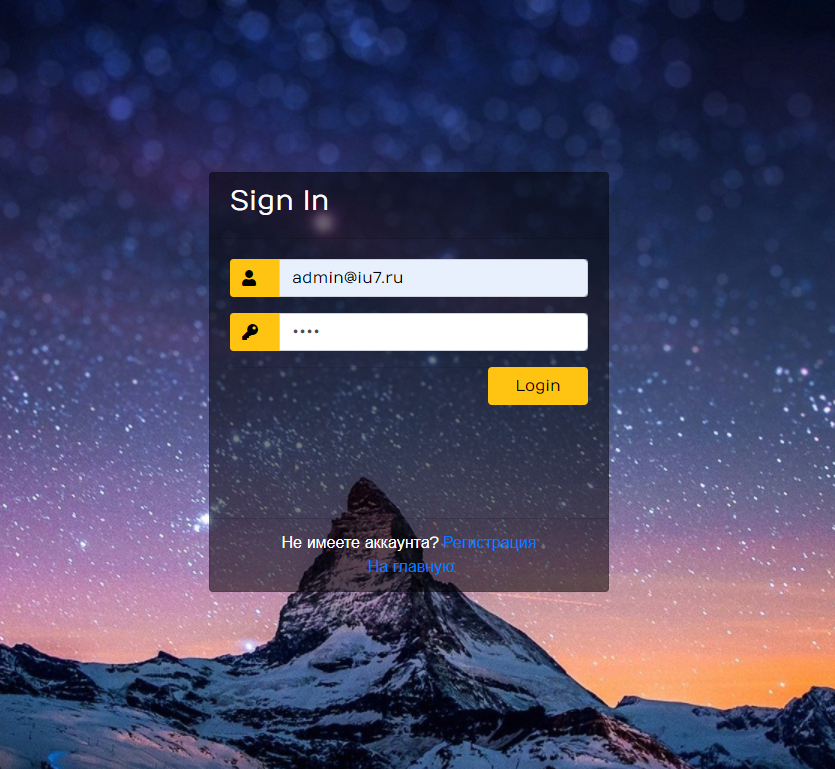


Рисунок 3.10 — Интерфейс страницы авторизации.

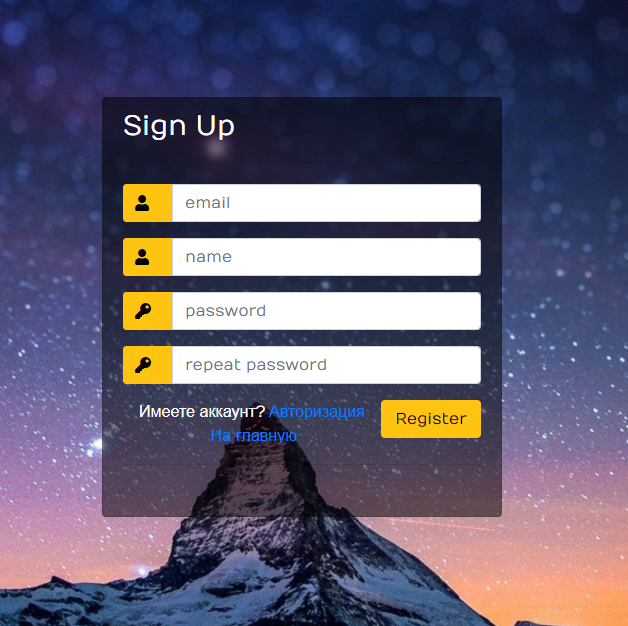
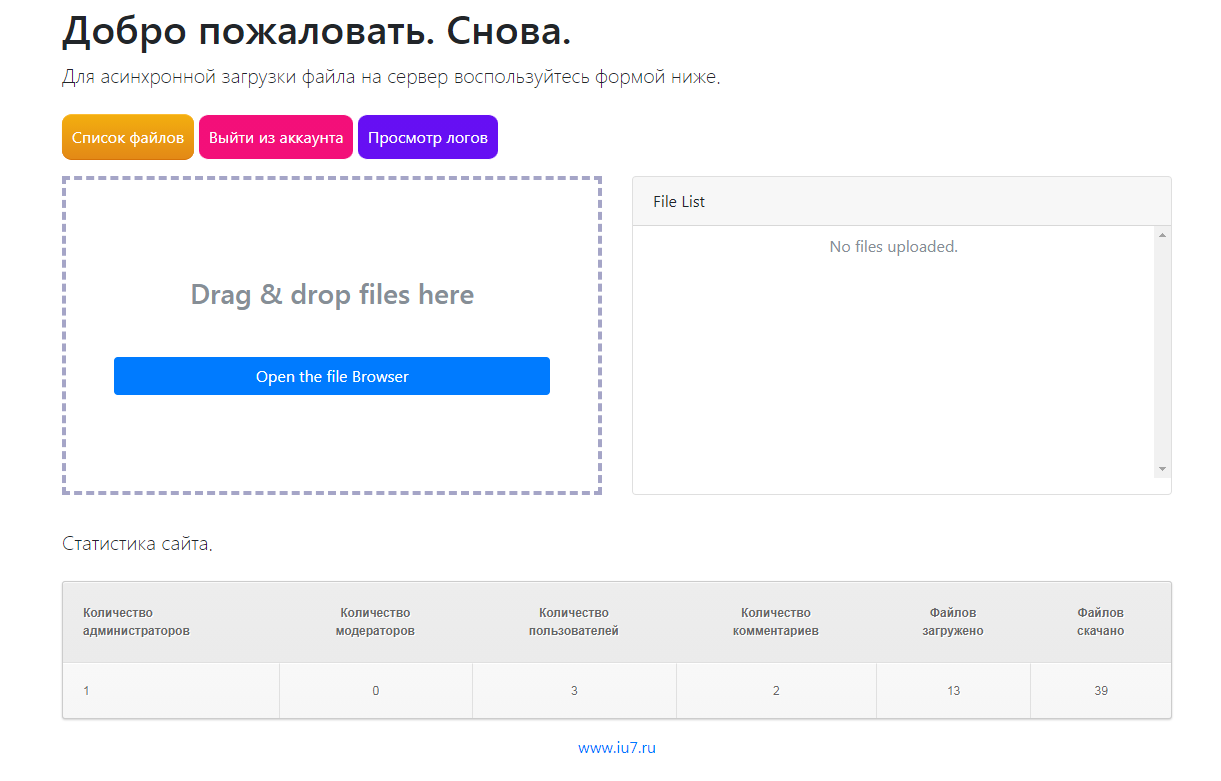


Рисунок 3.11 — Интерфейс страницы регистрации.

Рисунок 3.12 — Интерфейс страницы авторизированного администратора.

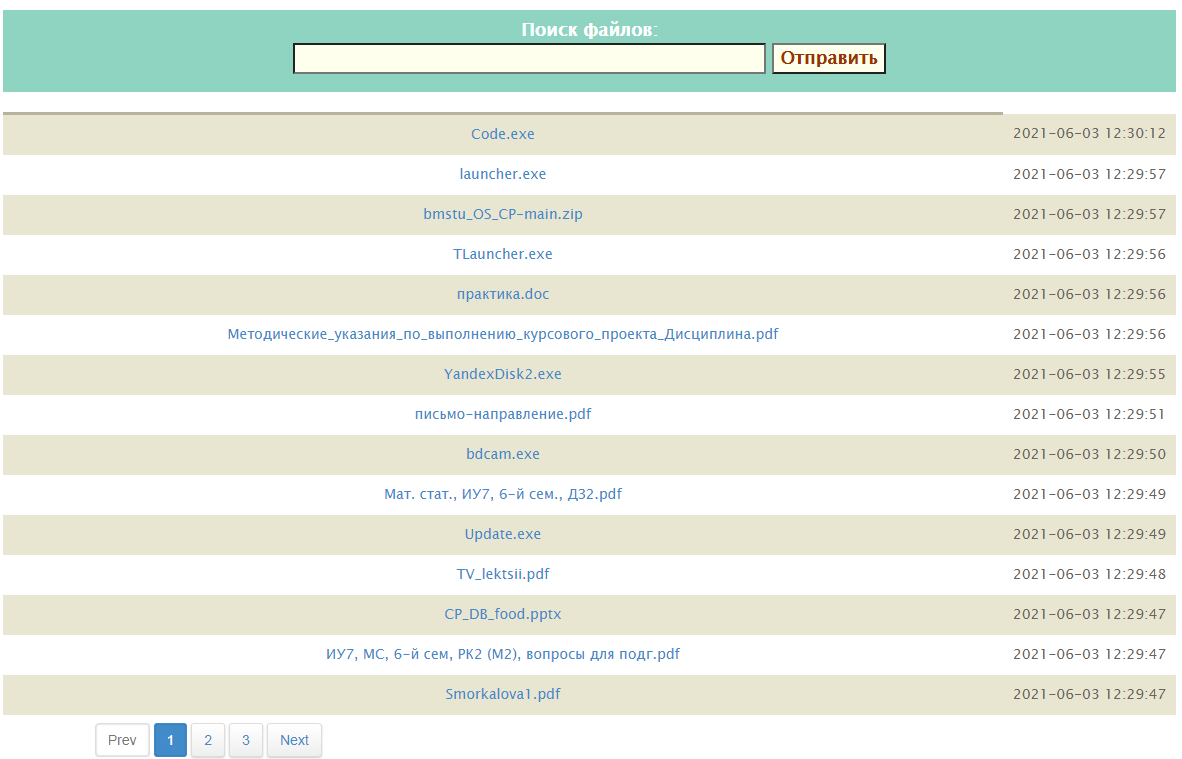
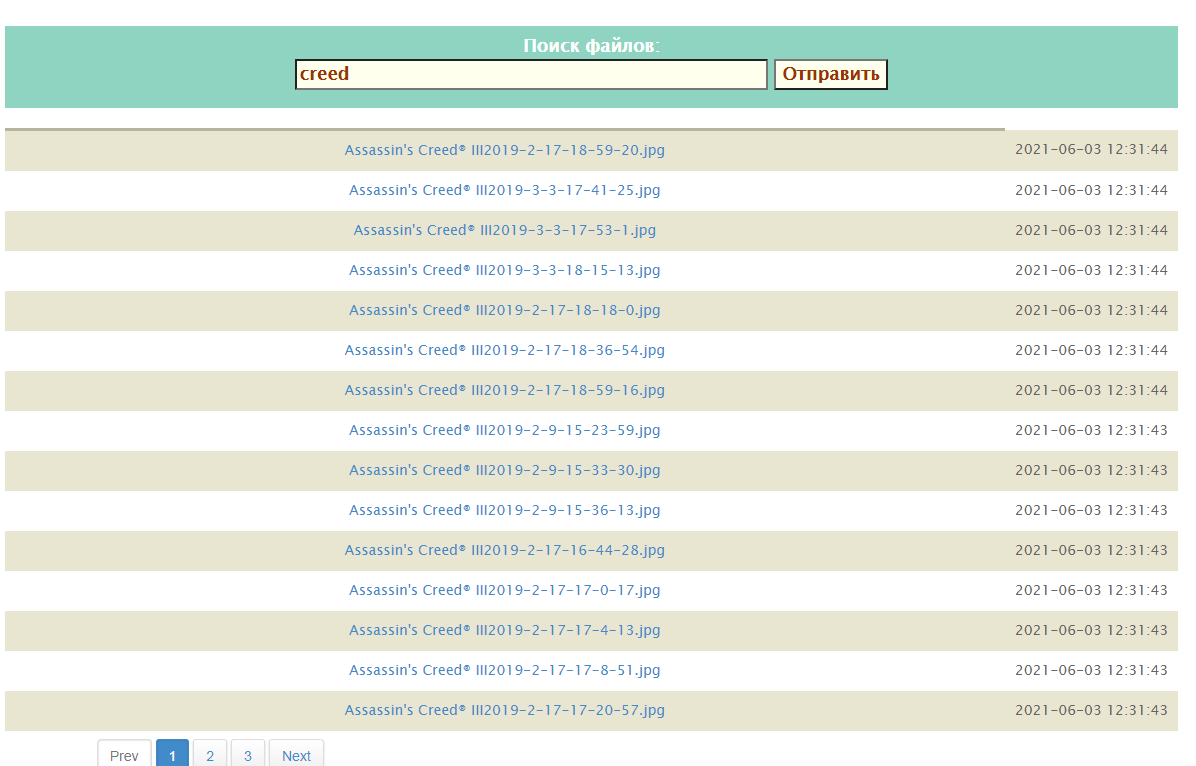


Рисунок 3.13 — Интерфейс страницы со списком файлов.

Произведем поиск файлов, в названии которых присутствует аббревиатура "creed".

Рисунок 3.14 — Результат поиска среди файлов.

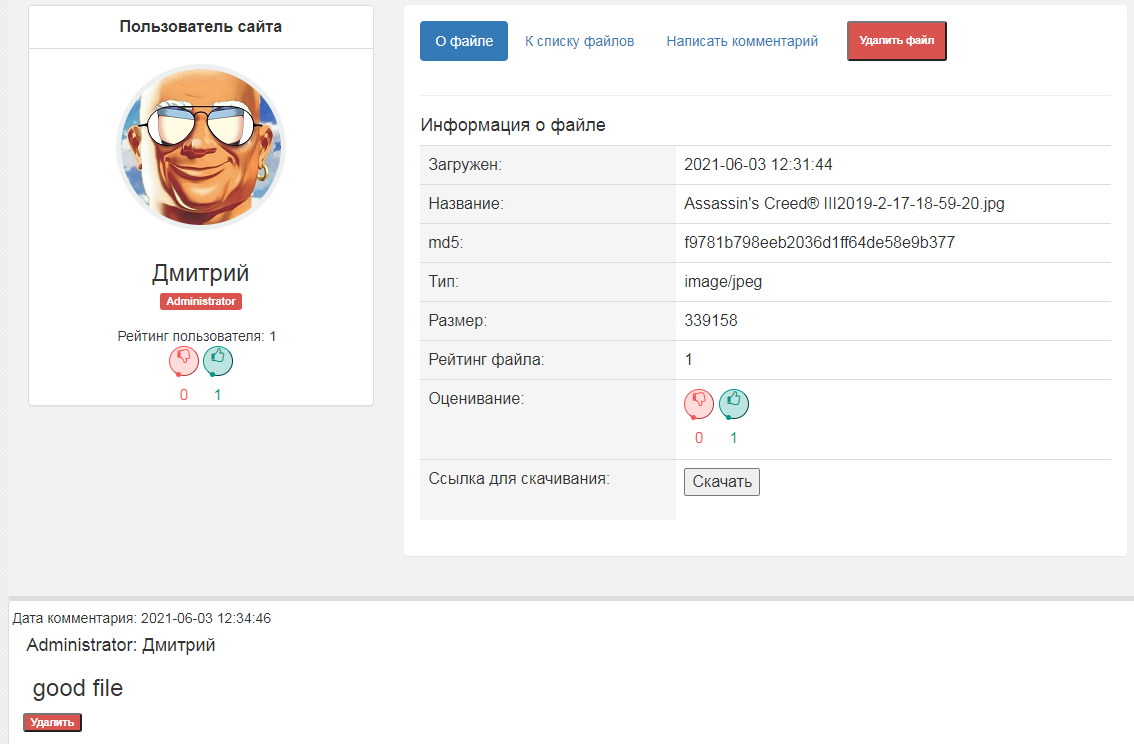


Рисунок 3.15 — Страница файла.



Рисунок 3.16 — Страница просмотра истории действий пользователей.

## 3.5 Выводы из технологического раздела

В данном разделе было реализовано спроектированное веб-приложение и представлены результаты его работы.

# Заключение

В результате проделанной работы были выполнены следующие задачи:

* формализовано задание, путем выделения соответствующих акторов и прецедентов;
* проведен анализ существующих решений, выделены преимущества и недостатки;
* проведен анализ СУБД и выбрана наиболее подходящая;
* спроектирована база данных;
* спроектирована архитектура приложения;
* разработано приложение.

Была достигнута цель проекта — реализация веб-приложения для обмена файлами в сети Интернет с разделением прав пользователей, комментированием и оцениванием файлов, а также оцениванием пользователей и ведением истории действий всех пользователей.

# Список использованной литературы

1. Freemium [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Freemium (дата обращения 31.05.2021).
2. Microsoft OneDrive [Электронный ресурс] Режим доступа: https://onedrive.live.com (дата обращения 02.06.2021).
3. WebDAV [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/WebDAV (дата обращения 02.06.2021)
4. PHP [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/PHP (дата обращения 03.06.2021)
5. Microsoft Visual Studio Code [Электронный ресурс] Режим доступа: https://code.visualstudio.com (дата обращения 05.06.2021)
6. RedBean ORM [Электронный ресурс] Режим доступа: https://redbeanphp.com (дата обращения 05.06.2021)

## А Презентация