



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  
(Группа)

Д.Р.Жигалкин  
(Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта

В.И.Солодовников  
(Подпись, дата) (И.О.Фамилия)

Москва, 2021 г.

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).

### 2. 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) (И.О. Фамилия)

## Оглавление

Введение .....	5
1. Аналитическая часть .....	6
1.1 Формализация задачи .....	6
1.2 Акторы системы .....	6
1.2.1 Гость .....	6
1.2.2 Пользователь .....	7
1.2.3 Модератор .....	7
1.2.4 Администратор .....	7
1.3 Сценарии использования .....	7
1.4 Обзор существующих решений .....	11
1.4.1 Dropbox .....	11
1.4.2 Google Диск .....	12
1.4.3 Microsoft OneDrive .....	12
1.5 Анализ моделей данных .....	14
1.5.1 Реляционная модель .....	15
1.5.2 Иные подходы .....	15
1.6 Выбор СУБД .....	16
1.6.1 MSSQL .....	16
1.6.2 MySQL .....	16
1.6.3 PostgreSQL .....	17
1.6.4 Oracle .....	18
1.7 Требования к безопасности базы данных .....	18
1.8 Выводы из аналитического раздела .....	18
2. Конструкторская часть .....	19
2.1 Проектирование базы данных .....	19
2.1.1 Ролевая модель .....	21
2.1.2 Хранимые процедуры .....	22
2.1.3 Триггеры .....	26
2.1.4 Обеспечение безопасности базы данных .....	27
2.2 Нормализация базы данных .....	28
2.2.1 Первая нормальная форма .....	28
2.2.2 Вторая нормальная форма .....	28

2.2.3 Третья нормальная форма .....	29
2.3 Проектирование архитектуры.....	29
2.4 Выводы из конструкторского раздела.....	31
3. Технологическая часть.....	32
3.1 Выбор и обоснование языка программирования и среды разработки .....	32
3.2 Диаграмма базы данных.....	32
3.3 UML-диаграммы компонентов .....	34
3.4 Интерфейс веб-приложения.....	38
3.5 Выводы из технологического раздела.....	42
Заключение .....	43
Список использованной литературы.....	44
А Презентация .....	45

## 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.

# **1. Аналитическая часть**

В данном 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 актер при работе с системой.

На рисунках 1.1 — 1.4 представлены диаграммы для каждого актора.

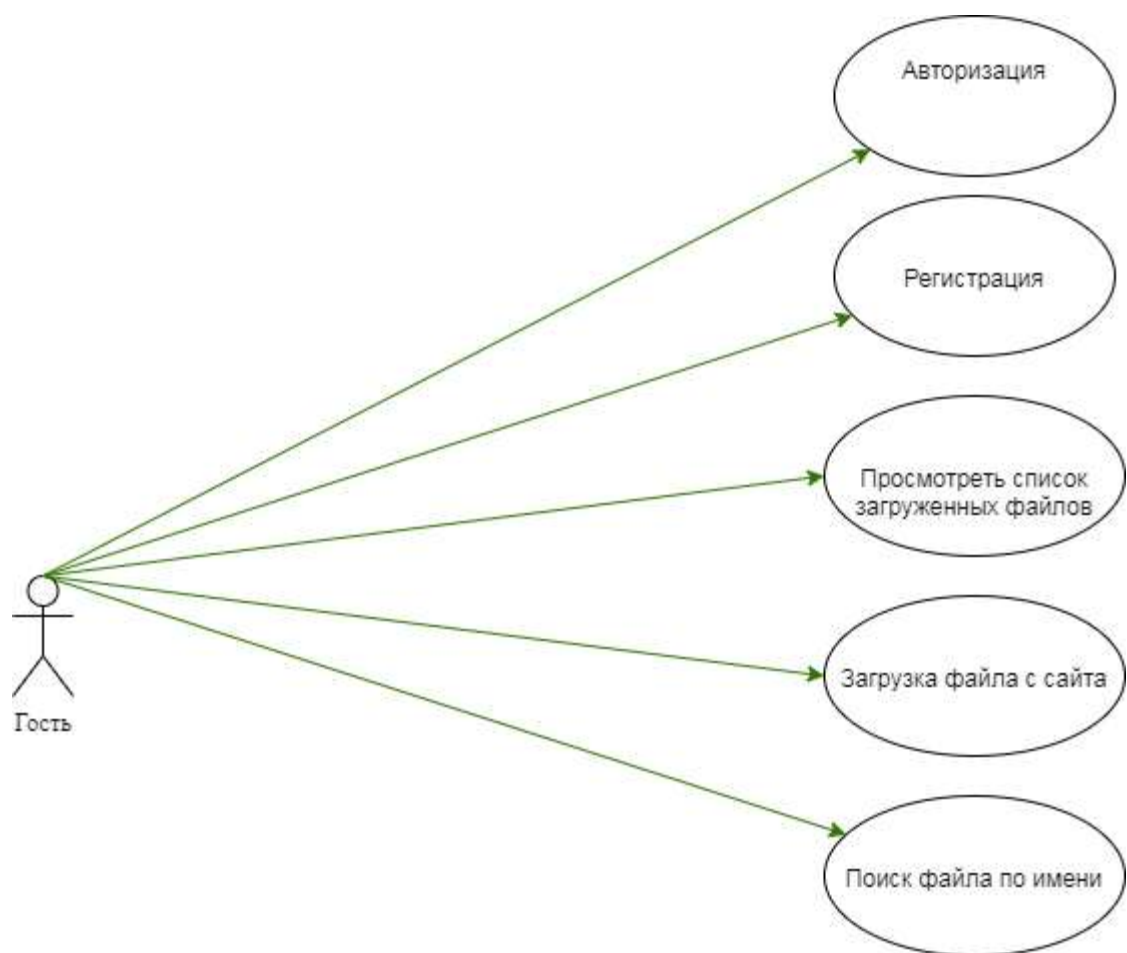


Рисунок 1.1 — Диаграмма сценариев использования приложения гостем.



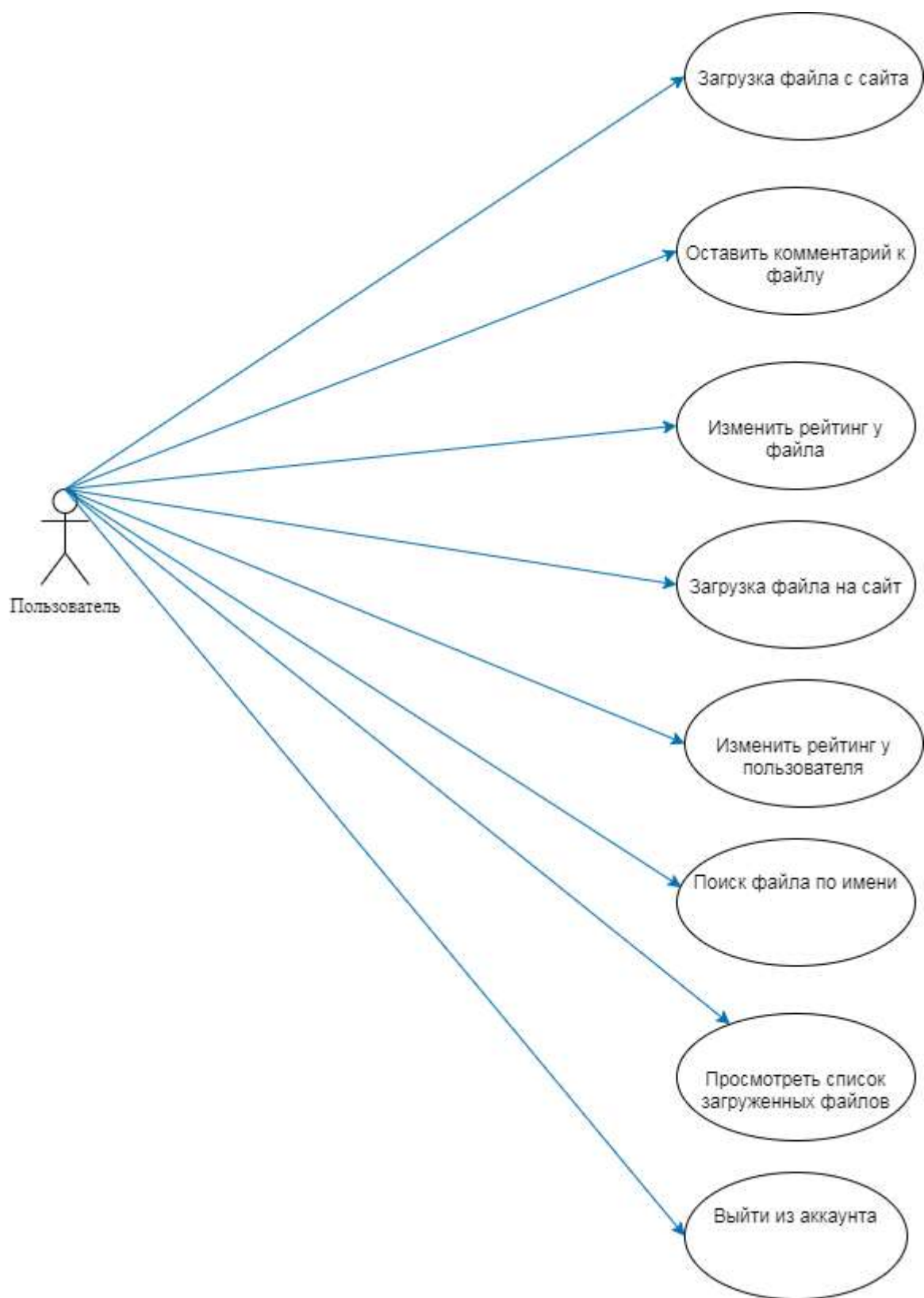


Рисунок 1.2 — Диаграмма сценариев использования приложения пользователем.



Рисунок 1.3 — Диаграмма сценариев использования приложения модератором.

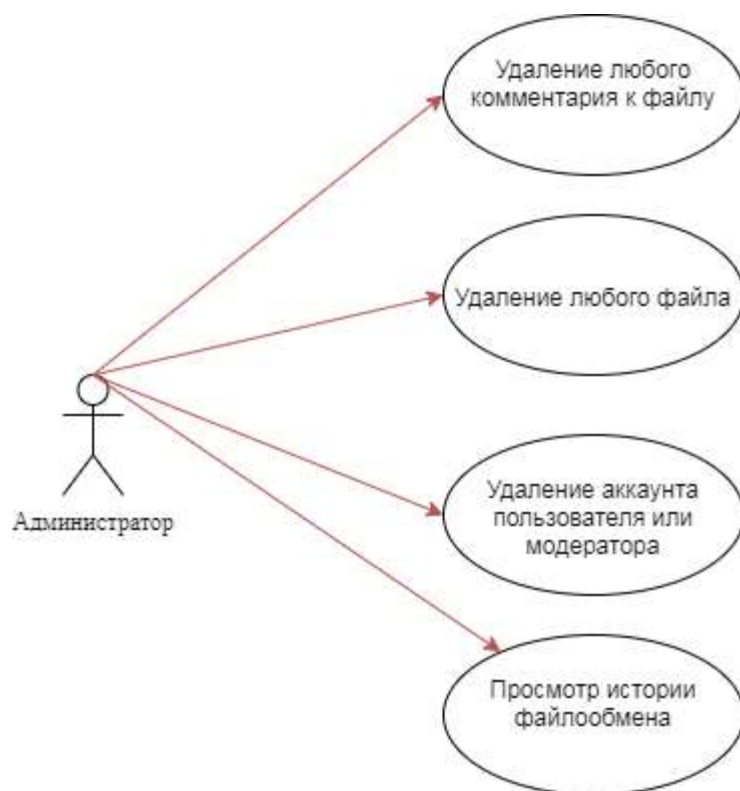


Рисунок 1.4 — Диаграмма сценариев использования приложения администратором.

## 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.

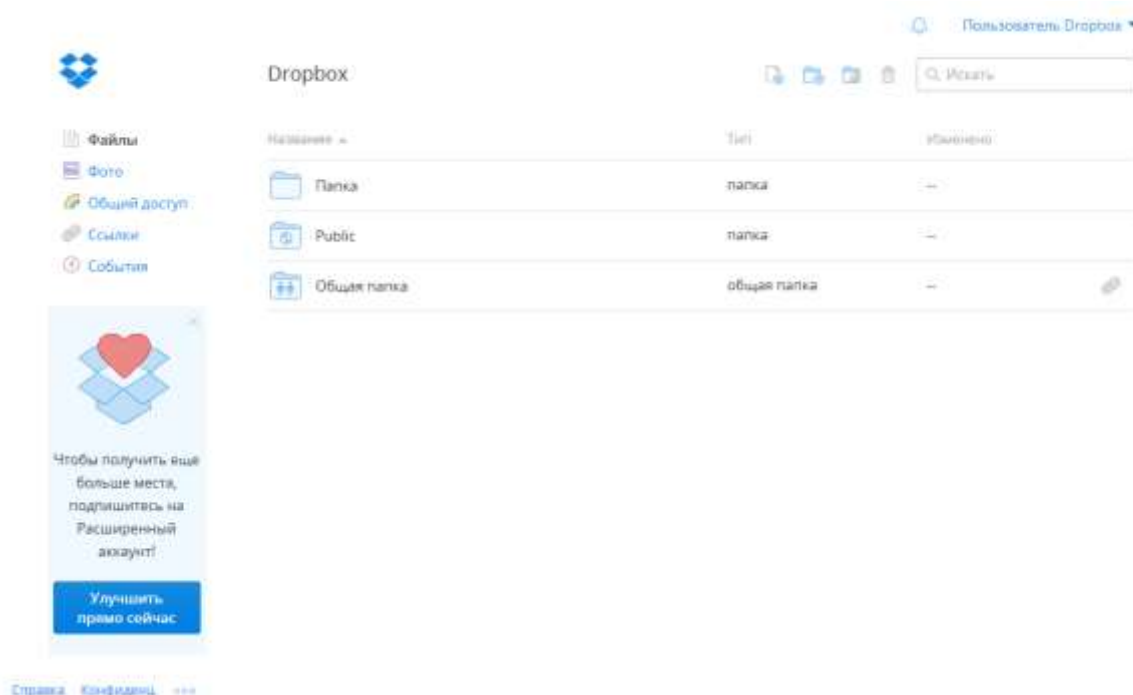


Рисунок 1.5 — Интерфейс Dropbox.

### 1.4.2 Google Диск

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

Его функции включают хранение файлов в Интернете, общий доступ к ним и совместное редактирование. В состав Google Диска входят Google Документы, Таблицы и Презентации — 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. Для использования неограниченного пространства для хранения необходима премиальная подписка.

На рисунке 1.6 показан интерфейс Google Диск.

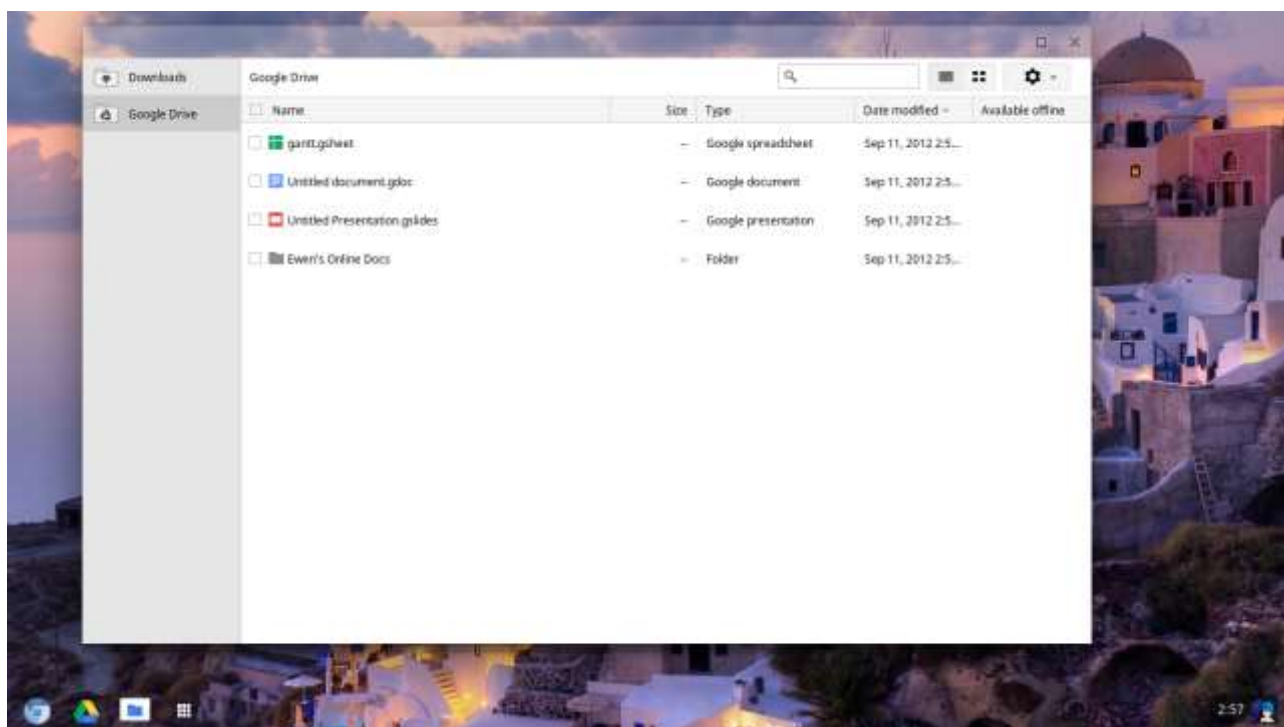


Рисунок 1.6 — Интерфейс Google Диск.

### 1.4.3 Microsoft OneDrive

OneDrive — облачное хранилище, созданное компанией Microsoft в августе 2007 года[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[3]. Для получения повышенной безопасности и увеличенного объема дискового пространства необходим доступ к премиум-возможностям.

На рисунке 1.7 показан интерфейс Microsoft OneDrive.

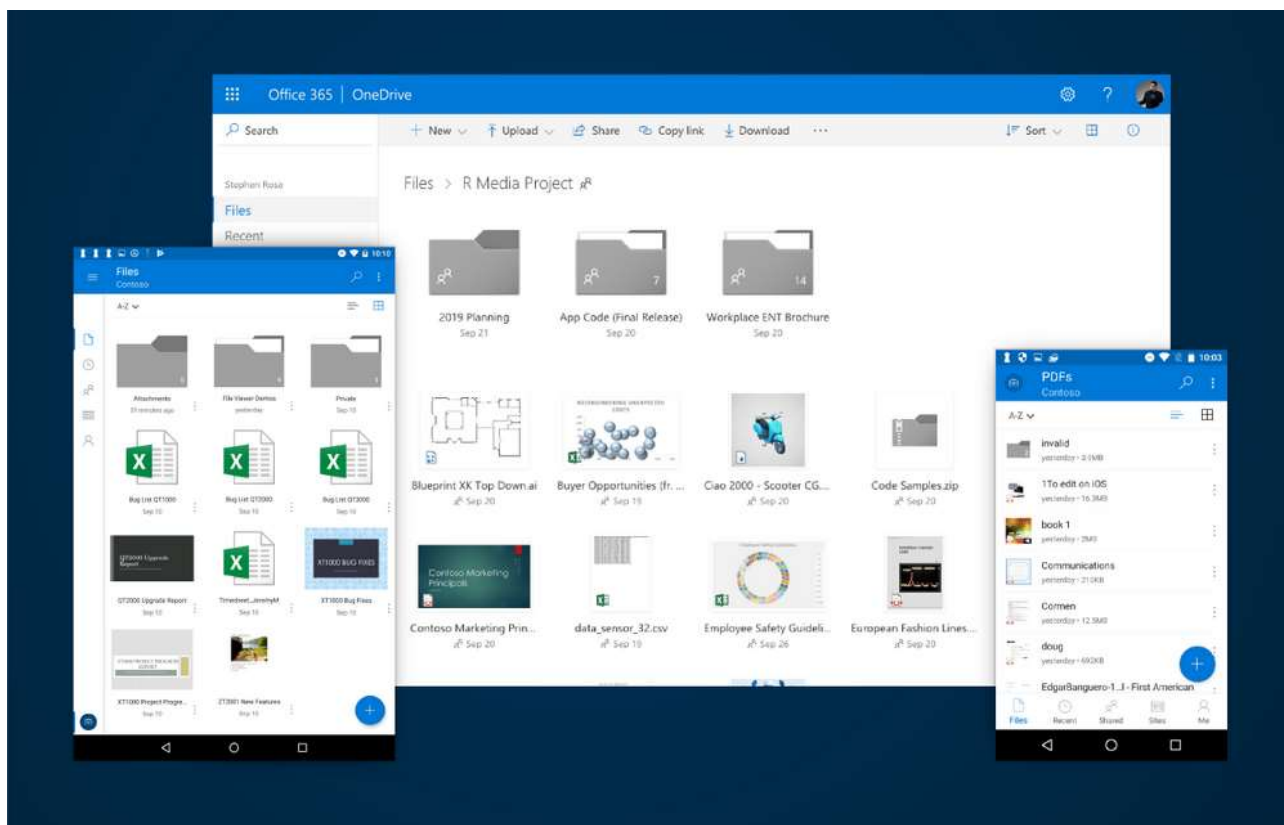


Рисунок 1.7 — Интерфейс Microsoft OneDrive.

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 неструктурированный 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	Подходит для постоянно меняющихся структур

	структур данных	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 подходит 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;
- сообщество: 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(a) — 2.1(б). Таблица 2.1 содержит описание сущностей базы данных.

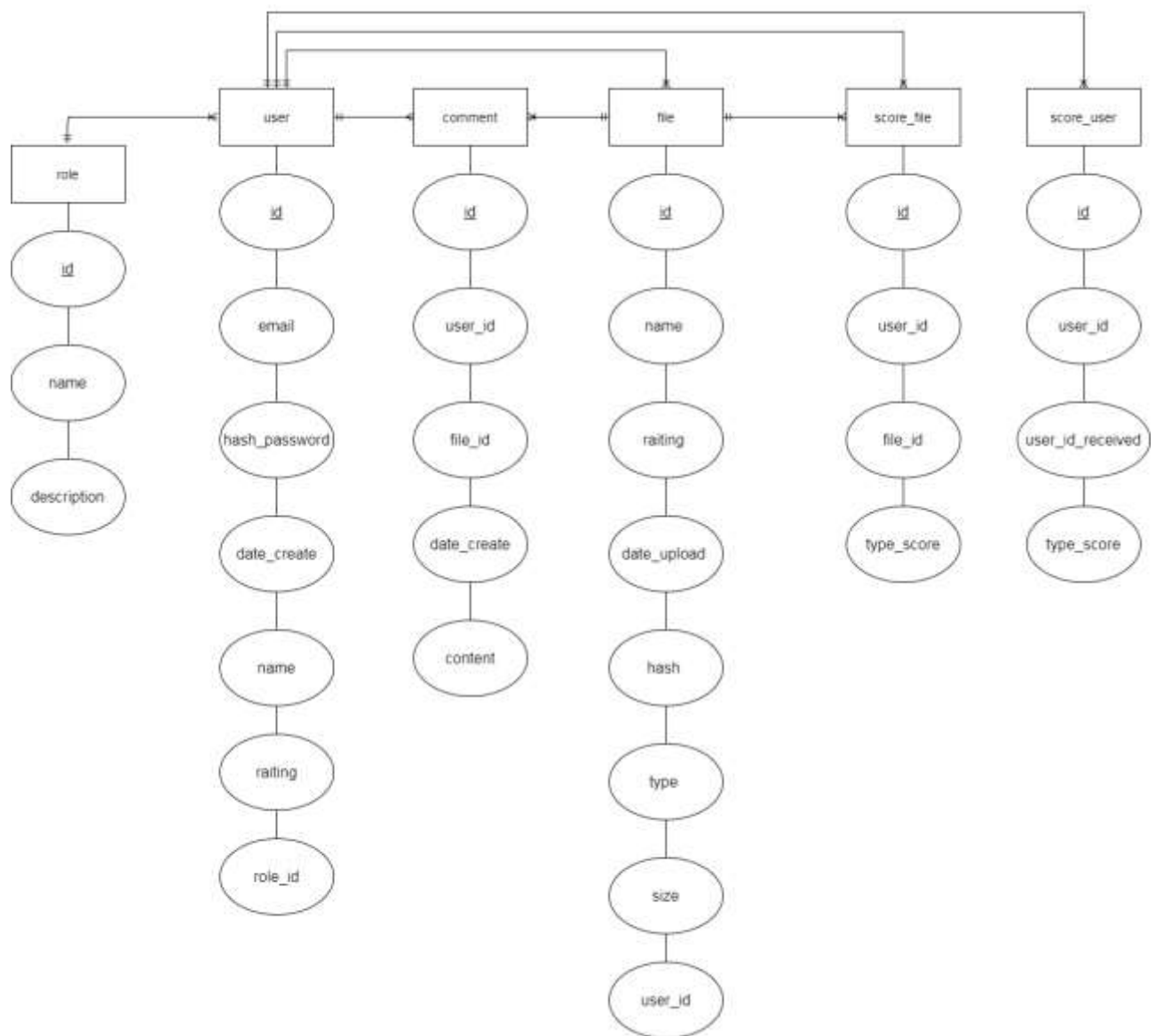


Рисунок 2.1(a) — ER-диаграмма.

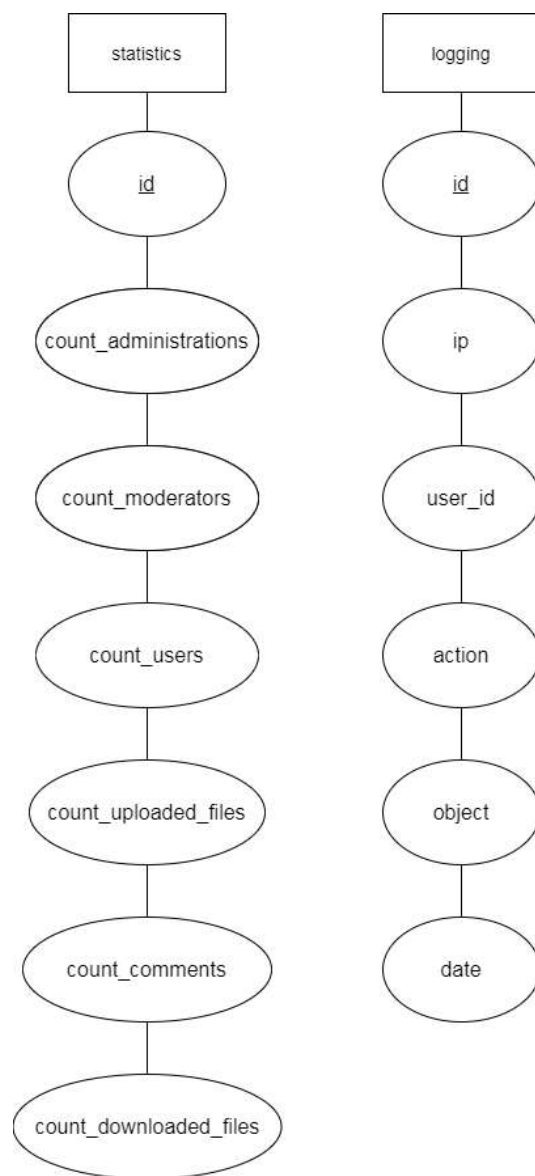


Рисунок 2.1(б) — ER-диаграмма.

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.

```

mysql> show grants for 'guest'@'localhost';
+-----+
| Grants for guest@localhost |
+-----+
| GRANT USAGE ON *.* TO 'guest'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'role' TO 'guest'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'file' TO 'guest'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'comment' TO 'guest'@'localhost' |
| GRANT SELECT, INSERT ON 'file_hosting'.'user' TO 'guest'@'localhost' |
| GRANT SELECT, UPDATE ON 'file_hosting'.'statistics' TO 'guest'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'score_user' TO 'guest'@'localhost' |
| GRANT INSERT ON 'file_hosting'.'logging' TO 'guest'@'localhost' |
+-----+
8 rows in set (0.00 sec)

mysql> show grants for 'user'@'localhost';
+-----+
| Grants for user@localhost |
+-----+
| GRANT USAGE ON *.* TO 'user'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'score_file' TO 'user'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'score_user' TO 'user'@'localhost' |
| GRANT SELECT, INSERT ON 'file_hosting'.'comment' TO 'user'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'role' TO 'user'@'localhost' |
| GRANT SELECT, UPDATE ON 'file_hosting'.'user' TO 'user'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'file' TO 'user'@'localhost' |
| GRANT INSERT ON 'file_hosting'.'logging' TO 'user'@'localhost' |
| GRANT SELECT, UPDATE ON 'file_hosting'.'statistics' TO 'user'@'localhost' |
+-----+
9 rows in set (0.00 sec)

mysql> show grants for 'moderator'@'localhost';
+-----+
| Grants for moderator@localhost |
+-----+
| GRANT USAGE ON *.* TO 'moderator'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'score_user' TO 'moderator'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'score_file' TO 'moderator'@'localhost' |
| GRANT SELECT, UPDATE ON 'file_hosting'.'user' TO 'moderator'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'role' TO 'moderator'@'localhost' |
| GRANT SELECT, INSERT ON 'file_hosting'.'logging' TO 'moderator'@'localhost' |
| GRANT SELECT, UPDATE ON 'file_hosting'.'statistics' TO 'moderator'@'localhost' |
| GRANT SELECT, INSERT, DELETE ON 'file_hosting'.'comment' TO 'moderator'@'localhost' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON 'file_hosting'.'file' TO 'moderator'@'localhost' |
+-----+
9 rows in set (0.00 sec)

mysql> show grants for 'administrator'@'localhost';
+-----+
| Grants for administrator@localhost |
+-----+
| GRANT USAGE ON *.* TO 'administrator'@'localhost' |
| GRANT SELECT, UPDATE, DELETE ON 'file_hosting'.'user' TO 'administrator'@'localhost' |
| GRANT SELECT, UPDATE ON 'file_hosting'.'statistics' TO 'administrator'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'score_user' TO 'administrator'@'localhost' |
| GRANT SELECT, INSERT ON 'file_hosting'.'logging' TO 'administrator'@'localhost' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON 'file_hosting'.'file' TO 'administrator'@'localhost' |
| GRANT SELECT, INSERT, DELETE ON 'file_hosting'.'comment' TO 'administrator'@'localhost' |
| GRANT SELECT, INSERT, UPDATE ON 'file_hosting'.'score_file' TO 'administrator'@'localhost' |
| GRANT SELECT ON 'file_hosting'.'role' TO 'administrator'@'localhost' |
| GRANT EXECUTE ON PROCEDURE 'file_hosting'.'update_score_user_after_delete_user' TO 'administrator'@'localhost' |
+-----+

```

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:
5:   SELECT COUNT(*) INTO number_users FROM user WHERE role_id = 1;
6:
7:   SELECT COUNT(*) INTO number_moderators FROM user WHERE role_id
   = 2;
8:
9:   SELECT COUNT(*) INTO number_administrators FROM user WHERE
   role_id = 3;
10:
11:  UPDATE statistics SET count_users = number_users where statistics.id = 1;
12:
13:  UPDATE statistics SET count_moderators = number_moderators where
   statistics.id = 1;
14:
15:  UPDATE statistics SET count_administrators = number_administrators where
   statistics.id = 1;
16: 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:
6:   DECLARE cur1 CURSOR FOR SELECT file_id, type_score FROM
   score_file WHERE user_id = id;
7:
8:   SELECT COUNT(*) INTO my_count FROM score_file WHERE user_id = id;
9:   OPEN cur1;
10:  while my_count > 0 do
11:    set my_count = my_count - 1;
12:    FETCH cur1 INTO received_id, score;
13:    if score > 0 then
14:      UPDATE file SET rating = rating - 1 where file.id = received_id;
15:    else
16:      UPDATE file SET rating = rating + 1 where file.id = received_id;
17:    end if;
18:  end while;
19:  CLOSE cur1;
20: 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:
6:   DECLARE cur1 CURSOR FOR SELECT user_id_received, type_score
   FROM score_user WHERE user_id = id;
7:
8:   SELECT COUNT(*) INTO my_count FROM score_user WHERE user_id =
   id;
9:   OPEN cur1;
10:  while my_count > 0 do
11:    set my_count = my_count - 1;
12:    FETCH cur1 INTO received_id, score;
13:    if score > 0 then
14:      UPDATE user SET rating = rating - 1 where user.id = received_id;
15:    else
16:      UPDATE user SET rating = rating + 1 where user.id = received_id;
17:    end if;
18:  end while;
19:
20:  CLOSE cur1;
21: 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.

```

22: CREATE TRIGGER `file_hosting`.`user_after_insert` AFTER INSERT ON
    `user` FOR EACH ROW
23:
24: BEGIN
25:     call update_count_users();
26: END

```

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

```

1: CREATE TRIGGER `file_hosting`.`user_after_update` AFTER UPDATE ON
    `user` FOR EACH ROW
2:
3: BEGIN
4:     call update_count_users();
5: 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.

```

6: CREATE TRIGGER `file_hosting`.`user_before_delete` BEFORE DELETE
    ON `user` FOR EACH ROW
7: BEGIN
8:     call update_score_file_before_delete_user(OLD.id);
9: 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.

```
mysql> SHOW GLOBAL VARIABLES like 'bind_address';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| bind_address  | 127.0.0.1 |
+-----+-----+
1 row in set (0.01 sec)
```

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.

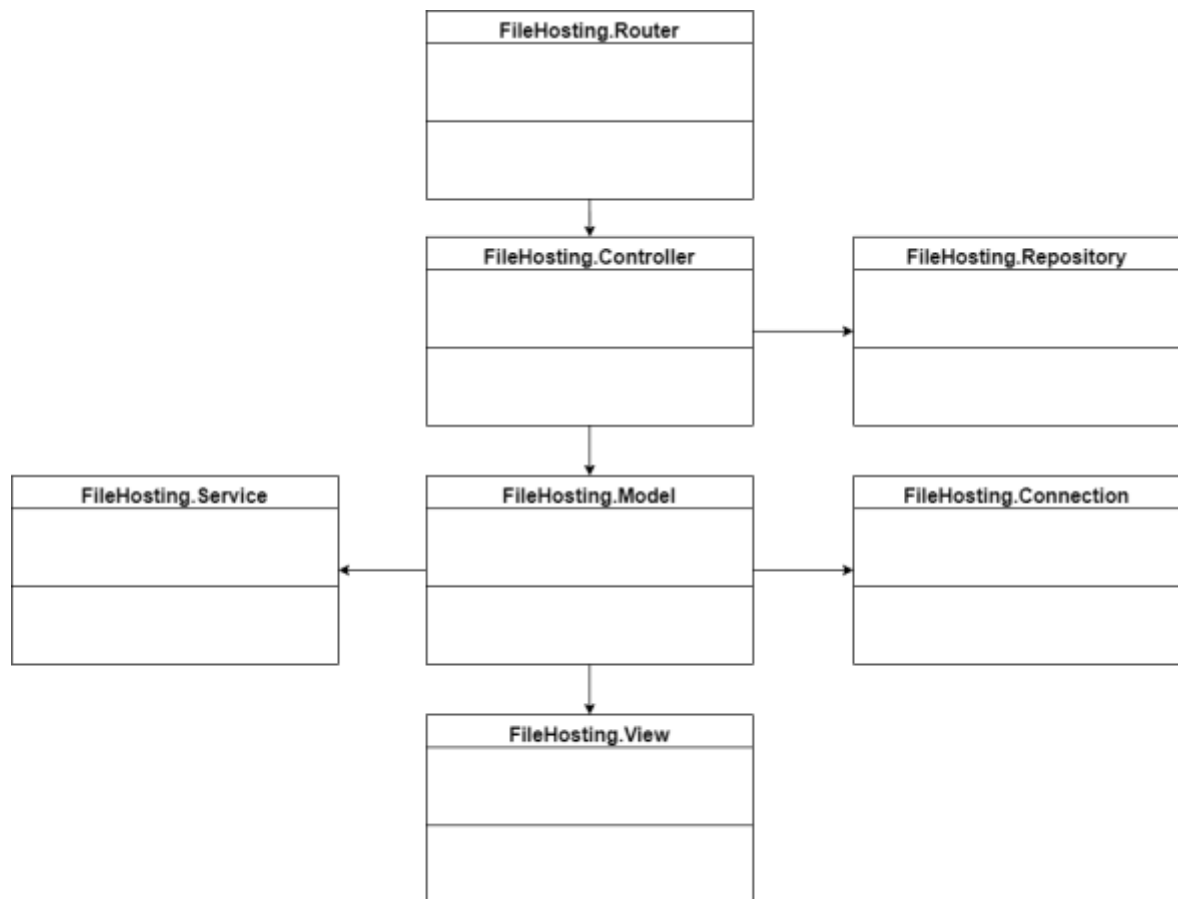


Рисунок 2.4 — Диаграмма компонентов приложения.

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 репозитория. 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 использовании 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.



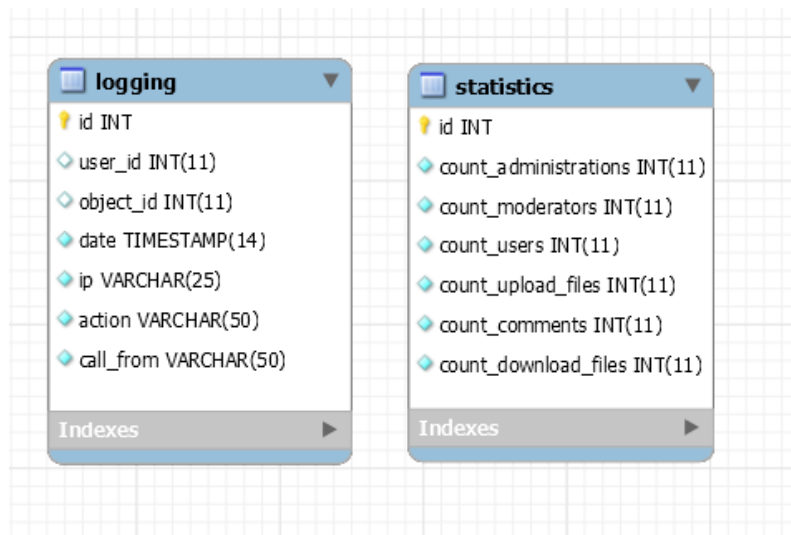


Рисунок 3.1(а) — Диаграмма базы данных.

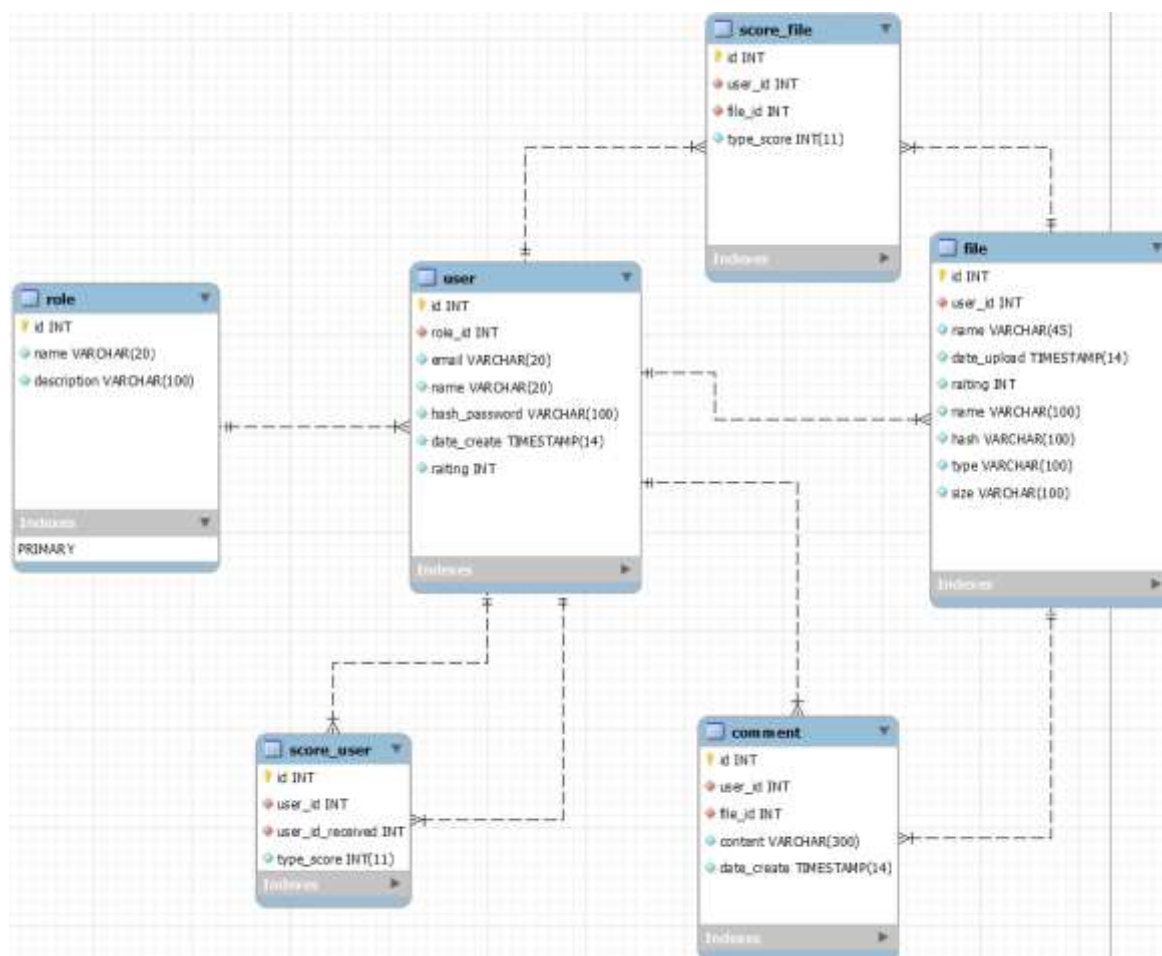


Рисунок 3.1(б) — Диаграмма базы данных.

### 3.3 UML-диаграммы компонентов

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

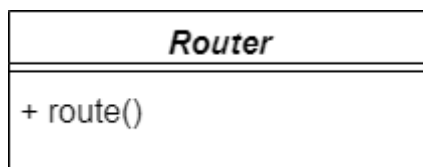


Рисунок 3.2 — UML-диаграмма класса компонента FileHosting.Router.

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

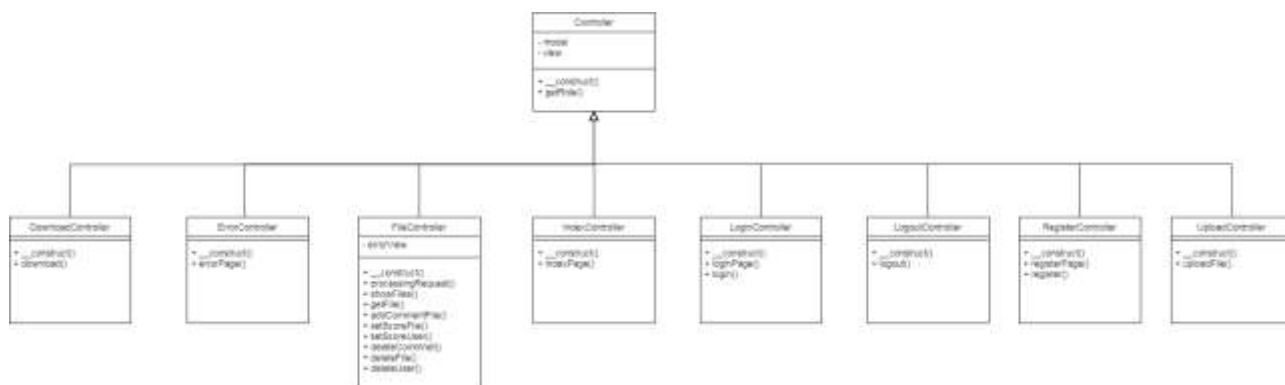


Рисунок 3.3 — UML-диаграмма класса компонента FileHosting.Controller.

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

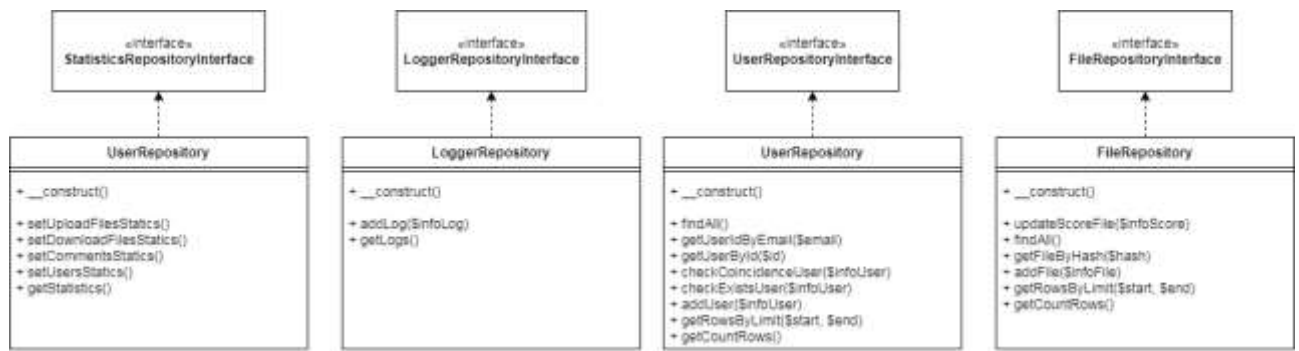


Рисунок 3.4(а) — UML-диаграмма классов компонента FileHosting.Repository.

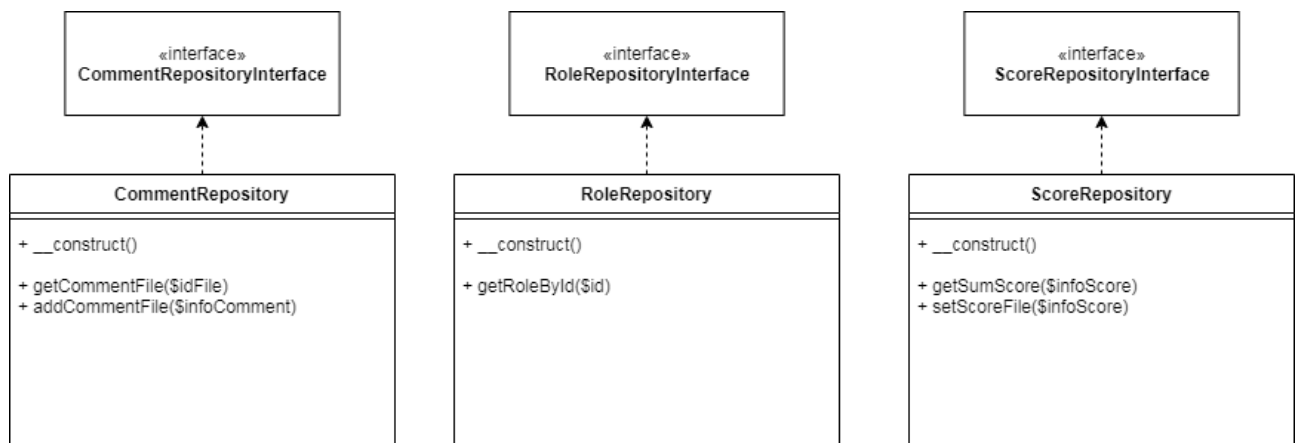


Рисунок 3.4(б) — UML-диаграмма классов компонента FileHosting.Repository.

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

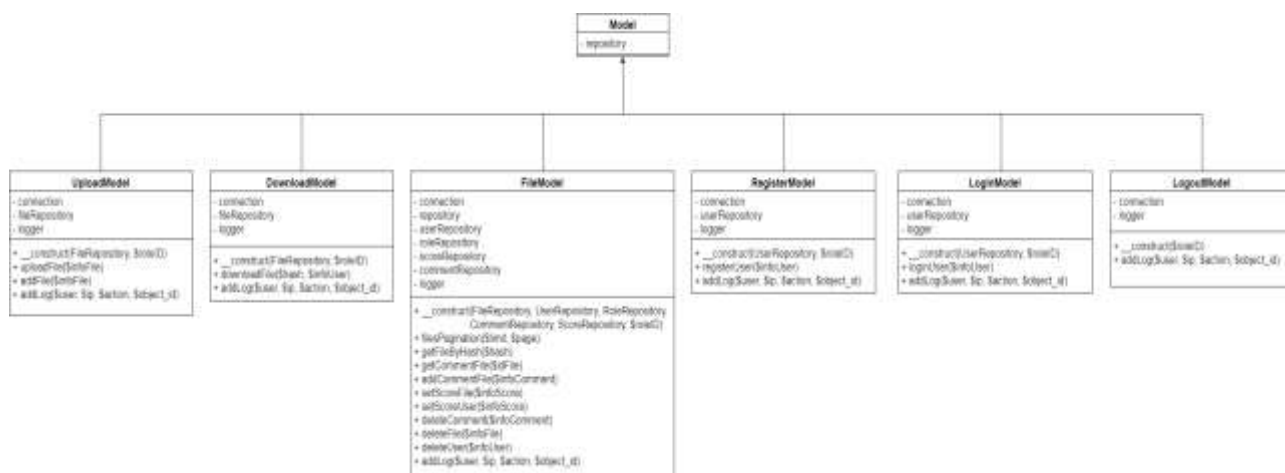


Рисунок 3.5 — UML-диаграмма классов компонента FileHosting.Model.

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

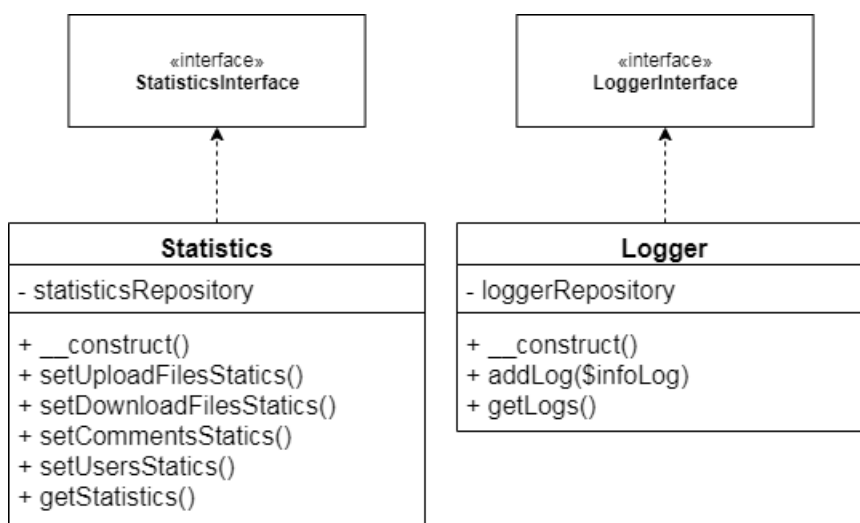


Рисунок 3.6 — UML-диаграмма классов компонента FileHosting.Service.

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

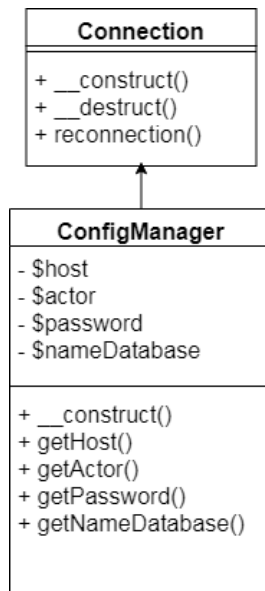


Рисунок 3.7 — UML-диаграмма классов компонента FileHosting.Connection.

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.

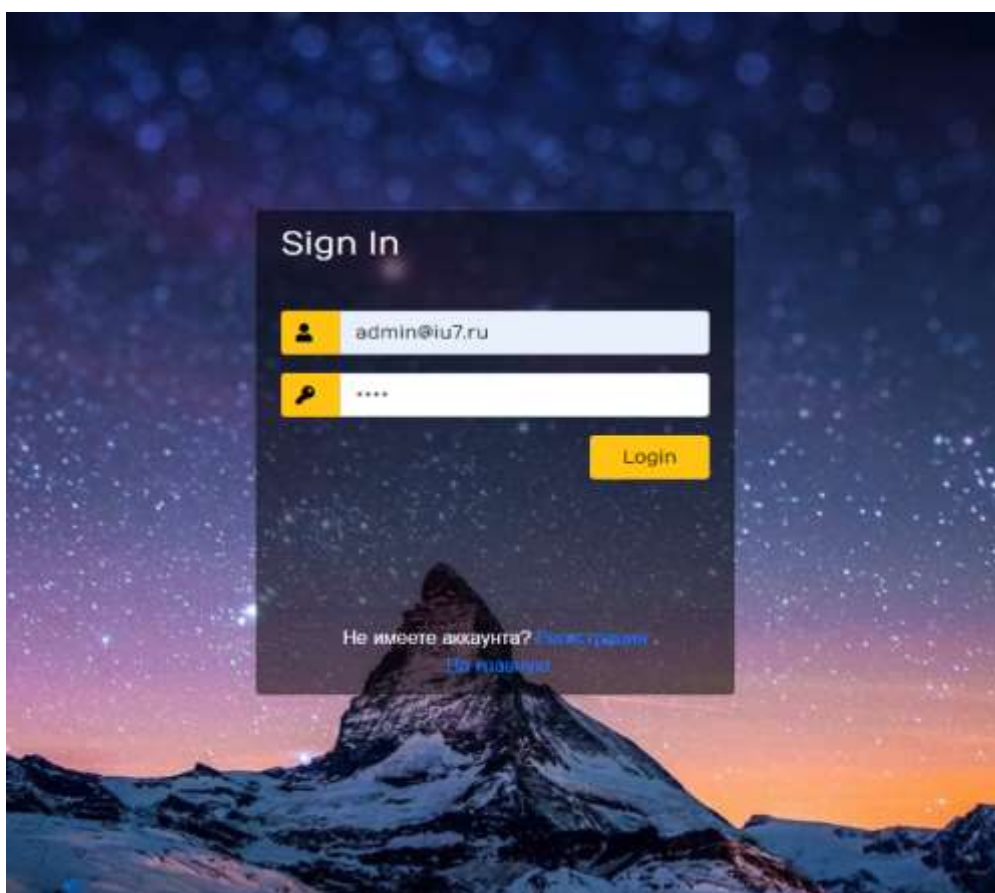


Рисунок 3.10 — Интерфейс страницы авторизации.

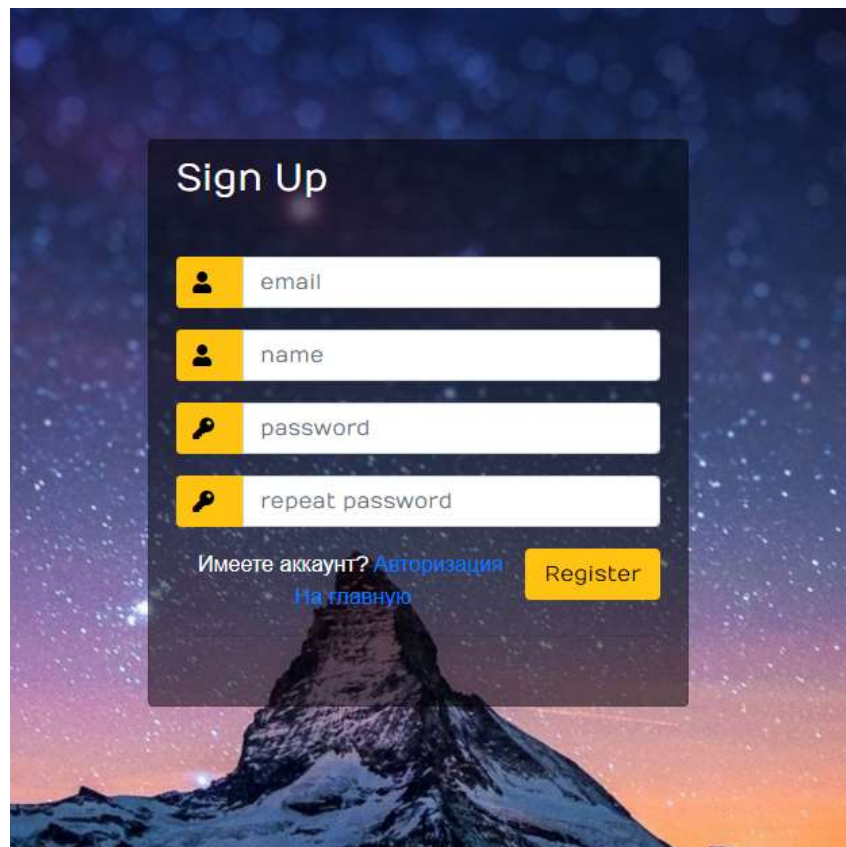


Рисунок 3.11 — Интерфейс страницы регистрации.

## Добро пожаловать. Снова.

Для асинхронной загрузки файла на сервер воспользуйтесь формой ниже.

[Список файлов](#)
[Выйти из аккаунта](#)
[Просмотр логов](#)

Drag & drop files here

Open the File Browser

File List

No files uploaded.

Статистика сайта:

Количество администраторов	Количество модераторов	Количество пользователей	Количество комментариев	Файлов загружено	Файлов скачено
1	0	3	2	13	30

www.lu7.ru

Рисунок 3.12 — Интерфейс страницы авторизованного администратора.

Поиск файлов	
<input type="text"/>	<input type="button" value="Отправить"/>
Code.exe	2021-06-03 12:30:12
launcher.exe	2021-06-03 12:29:57
bmstu_OS_CP-main.zip	2021-06-03 12:29:57
TLauncher.exe	2021-06-03 12:29:56
практика.doc	2021-06-03 12:29:56
Методические_указания_по_выполнению_курсового_проекта_Дисциплина.pdf	2021-06-03 12:29:56
YandexDisk2.exe	2021-06-03 12:29:55
письмо-направление.pdf	2021-06-03 12:29:51
bdcam.exe	2021-06-03 12:29:50
Мат. стат., ИУ7, 6-й сем., Д32.pdf	2021-06-03 12:29:49
Update.exe	2021-06-03 12:29:49
TV_lects11.pdf	2021-06-03 12:29:48
CP_DB_food.pptx	2021-06-03 12:29:47
ИУ7, МС, 6-й сем, РК2 (М2), вопросы для подг.pdf	2021-06-03 12:29:47
Smorkalova1.pdf	2021-06-03 12:29:47
Prev	1 2 3 Next

Рисунок 3.13 — Интерфейс страницы со списком файлов.


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

Поиск файлов	
<input type="text" value="creed"/>	<input type="button" value="Отправить"/>
Assassin's Creed® III2019-2-17-18-59-20.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-3-3-17-41-25.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-3-3-17-53-1.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-3-3-18-15-13.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-2-17-18-18-0.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-2-17-18-36-54.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-2-17-18-59-16.jpg	2021-06-03 12:31:44
Assassin's Creed® III2019-2-9-15-23-59.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-9-15-33-30.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-9-15-36-13.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-17-16-44-28.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-17-17-0-17.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-17-17-4-13.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-17-17-8-51.jpg	2021-06-03 12:31:43
Assassin's Creed® III2019-2-17-17-20-57.jpg	2021-06-03 12:31:43
Prev	1 2 3 Next

Рисунок 3.14 — Результат поиска среди файлов.



Пользователь сайта



Дмитрий

Администратор

Рейтинг пользователя: 1

0

1

О файле

К списку файлов

Написать комментарий

Удалить файл

Информация о файле

Загружен:	2021-06-03 12:31:44
Название:	Assassin's Creed® III2019-2-17-18-59-20.jpg
md5:	f9781b798eeb2036d1ff64de58e9b377
Тип:	image/jpeg
Размер:	339158
Рейтинг файла:	1
Оценивание:	<div><div>0</div><div>1</div></div>
Ссылка для скачивания:	<div>Скачать</div>

Дата комментария: 2021-06-03 12:34:46

Administrator: Дмитрий

good file

Удалить

Рисунок 3.15 — Страница файла.

**Список последних 1000 записей в таблице логов**  
(Сверху более свежие логи)

date: 2021-06-03 12:34:47   log_id: 714   user_id: 28   action: add comment   object_id: 89   ip: 109.252.45.40   call_from: FileModel Class
date: 2021-06-03 12:34:29   log_id: 713   user_id: 28   action: increase rating file   object_id: 12288   ip: 109.252.45.40   call_from: FileModel Class
date: 2021-06-03 12:31:44   log_id: 712   user_id: 28   action: upload file   object_id: 12291   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:44   log_id: 711   user_id: 28   action: upload file   object_id: 12290   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:44   log_id: 710   user_id: 28   action: upload file   object_id: 12289   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:44   log_id: 709   user_id: 28   action: upload file   object_id: 12288   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:44   log_id: 708   user_id: 28   action: upload file   object_id: 12287   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:44   log_id: 707   user_id: 28   action: upload file   object_id: 12286   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:44   log_id: 706   user_id: 28   action: upload file   object_id: 12285   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 705   user_id: 28   action: upload file   object_id: 12284   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 704   user_id: 28   action: upload file   object_id: 12283   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 703   user_id: 28   action: upload file   object_id: 12282   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 702   user_id: 28   action: upload file   object_id: 12281   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 701   user_id: 28   action: upload file   object_id: 12280   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 700   user_id: 28   action: upload file   object_id: 12279   ip: 109.252.45.40   call_from: UploadModel Class
date: 2021-06-03 12:31:43   log_id: 699   user_id: 28   action: upload file   object_id: 12278   ip: 109.252.45.40   call_from: UploadModel Class

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**