**Содержание**

[Введение 3](#_Toc134844974)

[1. Анализ требований к программному средству. 4](#_Toc134844975)

[1.1. Аналитический обзор аналогов 4](#_Toc134844976)

[1.1.1 Аналог «Pinterest». 4](#_Toc134844977)

[1.1.2 Аналог «Tumblr». 5](#_Toc134844978)

[1.1.3 Аналог «Imgbb». 5](#_Toc134844979)

[1.2. Разработка функциональных требований, определение вариантов использования. 6](#_Toc134844980)

[1.3. Вывод по разделу. 6](#_Toc134844981)

[2. Разработка архитектуры проекта. 8](#_Toc134844982)

[2.1 Диаграммы UML, взаимосвязь всех компонентов. 8](#_Toc134844983)

[2.2. Описание информационных объектов и ограничений целостности. 9](#_Toc134844984)

[2.2.1 Таблица users. 9](#_Toc134844985)

[2.2.2 Таблица comments. 10](#_Toc134844986)

[2.2.3 Таблица posts. 10](#_Toc134844987)

[2.2.4 Таблица favorites. 10](#_Toc134844988)

[2.2.5 Таблица comment\_likes. 11](#_Toc134844989)

[2.2.6 Таблица tags. 11](#_Toc134844990)

[2.2.7 Таблица post\_tags. 11](#_Toc134844991)

[2.2.8 Таблица subscriptions. 11](#_Toc134844992)

[2.3. Вывод по разделу. 12](#_Toc134844993)

[3. Разработка модели базы данных. 13](#_Toc134844994)

[3.1. Создание необходимых объектов. 13](#_Toc134844995)

[3.2. Описание используемой технологии. 15](#_Toc134844996)

[3.3. Вывод по разделу. 15](#_Toc134844997)

[4. Установка, настройка и использование PostgreSQL. 16](#_Toc134844998)

[4.1. Создание таблиц. 16](#_Toc134844999)

[4.2. Создание ролей для разграничения доступа. 16](#_Toc134845000)

[4.4. Описание процедур экспорта и импорта. 16](#_Toc134845001)

[4.5. Тестирование производительности базы данных. 17](#_Toc134845002)

[4.6. Вывод. 18](#_Toc134845003)

[5. Тестирование, проверка работоспособности и анализ полученных результатов. 20](#_Toc134845004)

[5.1. Тестирование клиентской части. 20](#_Toc134845005)

[5.2. Тестирование области работы модератора. 25](#_Toc134845006)

[5.3. Вывод. 27](#_Toc134845007)

[Заключение 28](#_Toc134845008)

[Список использованных литературных источников 29](#_Toc134845009)

## **Введение**

Неотъемлемой частью жизни большинства людей в современном информационном обществе являются различного вида социальные сети. Одним из видов таких приложений являются фотохостинги, позволяющие публиковать и хранить любые изображения. Такие приложения дают возможность обмениваться новыми идеями и работами в форме графической информации. Стремительный рост количества интернет-пользователей, социальные сети, развитие всех сфер деятельности человека позволили такого рода приложениям стать весьма востребованными.

Социальные сети подобного рода направленны на миллионную аудиторию. Они позволяют увлекательно провести досуг, ознакомить пользователей всего мира со своими идеями и работами удобным и доступным способом.

В процессе работы над проектом были поставлены следующие задачи:

* Реализовать роли пользователя и модератора.
* Регистрация и авторизация пользователей.
* Обеспечение добавления и удаления постов со стороны пользователя.
* Удаление нежелательного контента модератором.
* Добавление и удаление комментариев пользователями.
* Обеспечение добавление постов в избранное со стороны пользователя.

Основная цель курсового проекта: создание базы данных для локального фотохостинга.

## **Анализ требований к программному средству.**

* 1. **Аналитический обзор аналогов.**

В современном мире фотохостинги являются достаточно популярными веб-сервисами. Их основная задача – это хранение фотоизображений, которые загружают пользователи. В дальнейшем эти изображения могут быть использованы другими. При разработке базы данных для своего приложения я обратила внимания на самые успешные примеры популярнейших фотохостингов. Аналоги рассмотрены ниже.

* + 1. **Аналог «Pinterest».**

В качестве одного из аналогов был выбран сайт «https://www.pinterest.com», представленный на рисунке 1.1. Этот сайт предназначен для публикации изображений, видео и статей, сопровождающихся различной графической информацией. На данном сайте есть возможность публикации графической информации, комментирования, сохранения на свой аккаунт и чата с другими пользователями. Данный фотохостинг имеет наибольшую популярность из всех подобных сайтов.

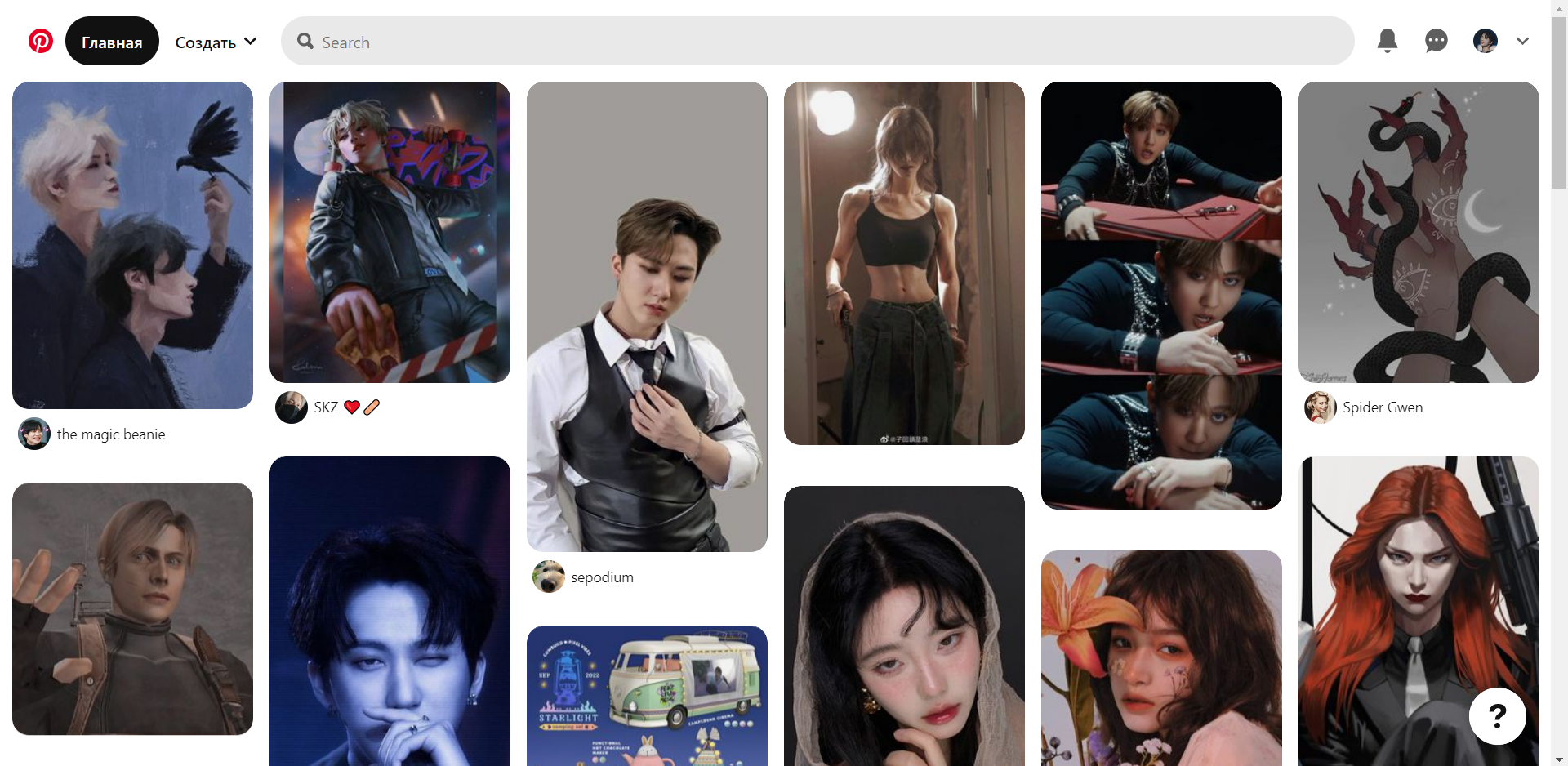
****

Рисунок 1.1 – Сайт «Pinterest»

Он пользуется популярностью также своей возможностью создавать «доски», то есть объединять и добавлять изображения в группы. Пример таких групп можно увидеть на рисунке 1.2.

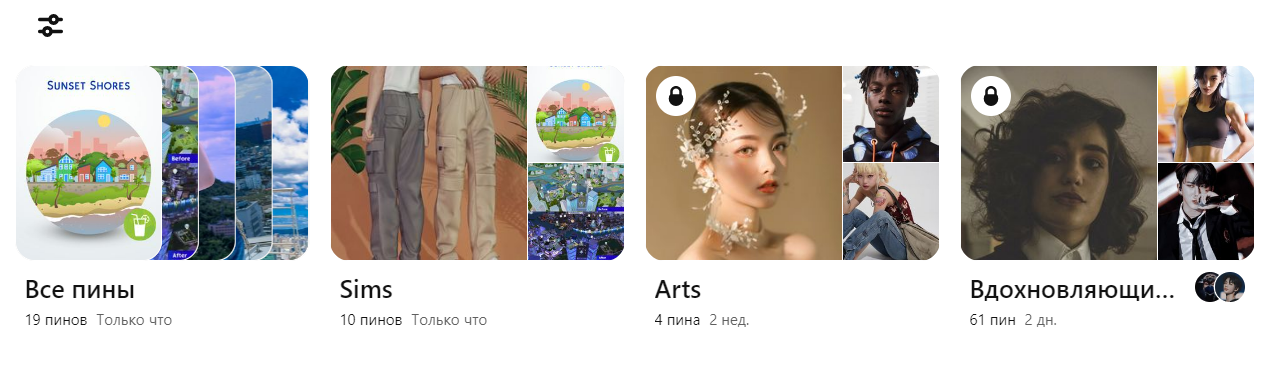


Рисунок 1.2 – «Доски» на сайте «Pinterest»

Pinterest хранит изображения в своей базе данных, а точнее, ссылки на изображения. Когда пользователь загружает изображение на платформу, оно сохраняется на серверах Pinterest.

Из минусов хотелось бы отметить довольно большое количество функционала, что начинающим пользователям затрудняет использование сайта.

* + 1. **Аналог «Tumblr».**

Еще одним аналогом выбран сайт «https://www.tumblr.com», который показан на рисунке 1.3. Платформа «Tumblr» — это служба микроблогов, включающая в себя множество картинок, статей, видео и gif-изображений по разным тематикам и позволяющая пользователям публиковать посты в свой блог. Пользователь может подписываться на блоги других пользователей, после чего их записи будут появляться на его ленте новостей.

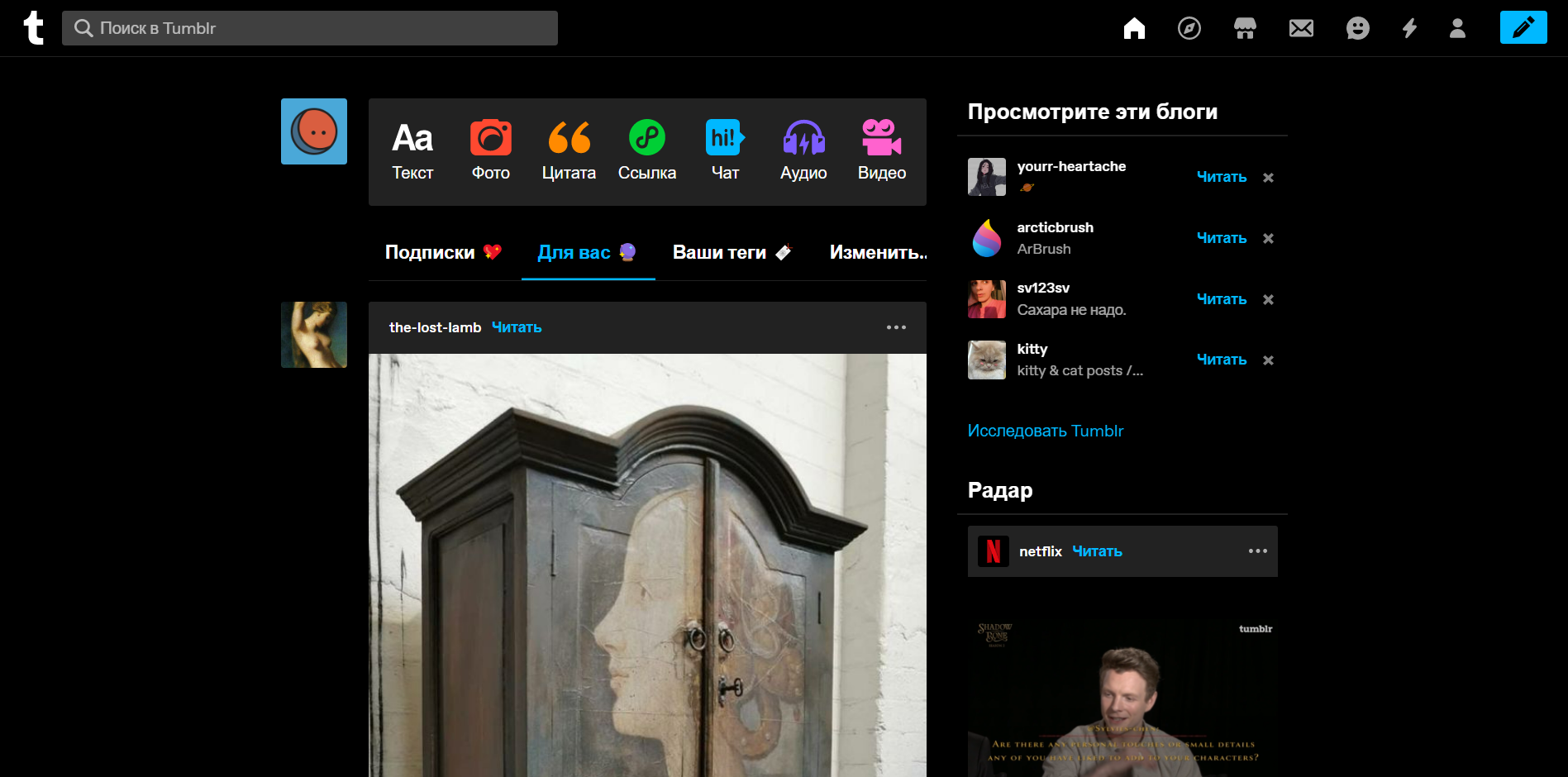
****

Рисунок 1.3 – Сайт «Tumblr»

В Tumblr изображения хранятся на удаленных серверах, а не в базе данных. Когда пользователь загружает изображение на данный фотохостинг, оно сохраняется на сервере Tumblr и присваивается URL-адрес. Затем URL-адрес используется для отображения изображения на странице пользователей.

* + 1. **Аналог «Imgbb».**

Заключительным в списке обозреваемых аналогов является сайт «https://imgur.com/», продемонстрирован на рисунке 1.4. Функционал схож с «Pinterest», так же имеется статистика просмотра, оценка публикаций. Имеется возможность отправки ссылки на публикацию на электронную почту. Хостинг позволяет загружать фотографии через компьютер или со смартфона. Сервис позволяет устанавливать различные параметры конфиденциальности.

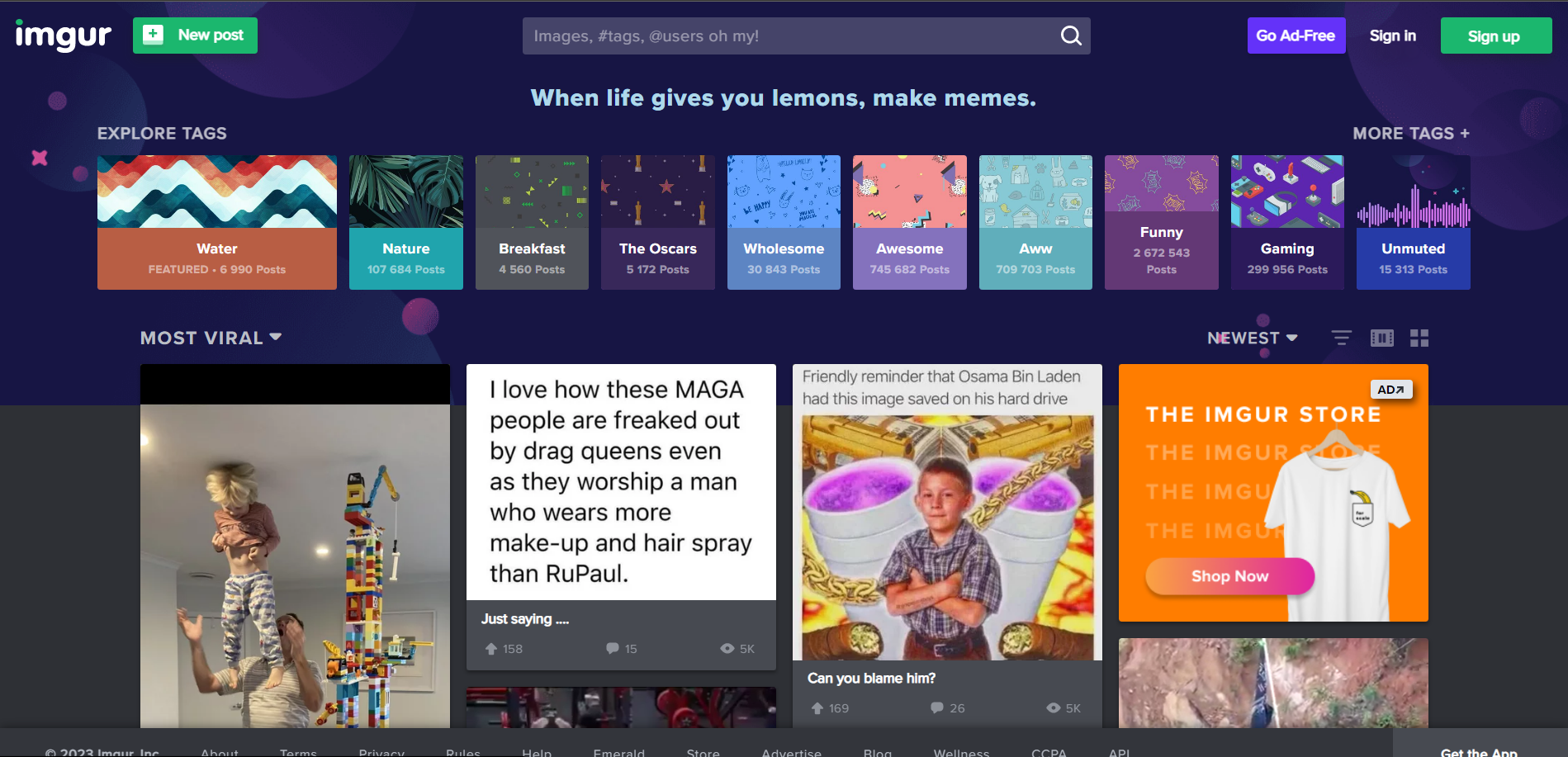
****

Рисунок 1.3 – сайт «Imgur»

Imgbb использует серверное хранение изображений вместо хранения в базе данных. При загрузке изображения на сервер Imgbb создает копию файла изображения на своих серверах и присваивает ему уникальный идентификатор, который затем используется для отображения изображения на сайте.

Из недостатков можно отметить то факт, что на сайт можно загружать не более 50 файлов в час.

* 1. **Разработка функциональных требований, определение вариантов использования.**

Разобрав приведенные выше сервисы для хранения фотоизображений, можно сформировать общий функционал необходимый для работы приложения.

Прежде всего, база данных должна корректно работать с файлами изображений. Обязательно пользователь должен быть способен сохранять в избранное понравившейся ему пост и скачивать изображение на свой компьютер. Пользователь должен иметь возможность оставлять комментарии и оценивать их – таким образом авторы, смогут узнать мнение об их творчестве и качестве материала.  
 Вариантами использования базы данных являются:

* Регистрация и авторизация пользователей;
* Добавление и удаление постов со стороны пользователей;
* Удаление нежелательного контента со стороны модератора;
* Добавление и удаление комментариев;
* Добавление постов в избранное пользователями;
  1. **Вывод по разделу.**

В данном разделе были рассмотрены аналогичные решения, проанализированы их недостатки и преимущества. Был разработаны функциональные требования. Все вышеперечисленные пункты понадобятся для достижения поставленной цели и для создания в дальнейшем качественного продукта.

1. **Разработка архитектуры проекта.**

**2.1 Диаграммы UML, взаимосвязь всех компонентов.**

Диаграмма UML базы данных можно увидеть на рисунке 2.1.

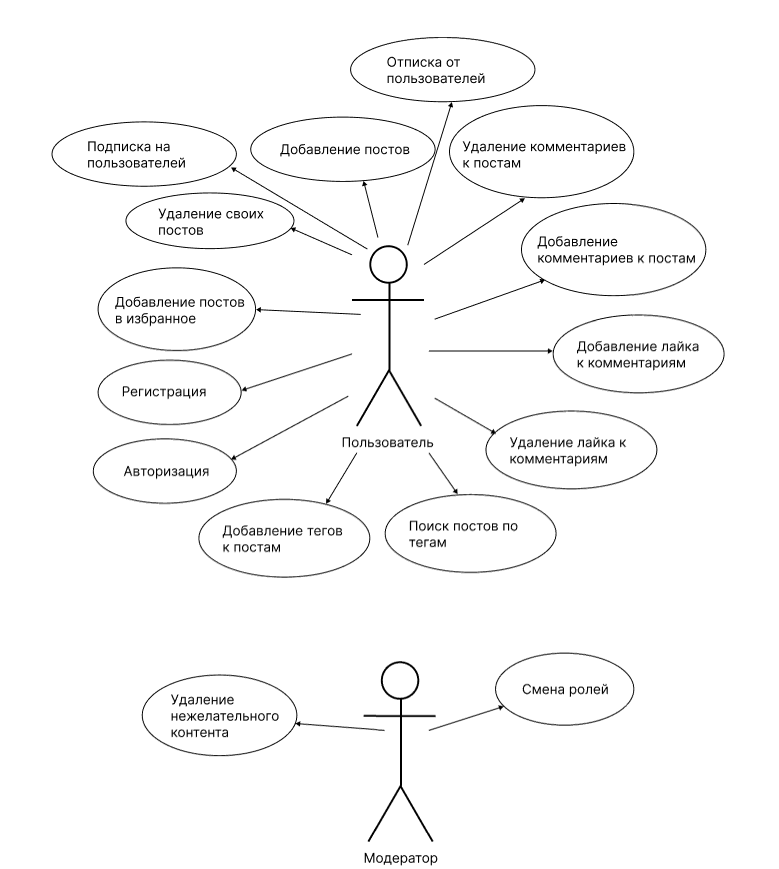


Рисунок 2.1 – Диаграмма использования

Схема базы данных ограничения целостности, связи и поля представлены на рисунке 2.2.

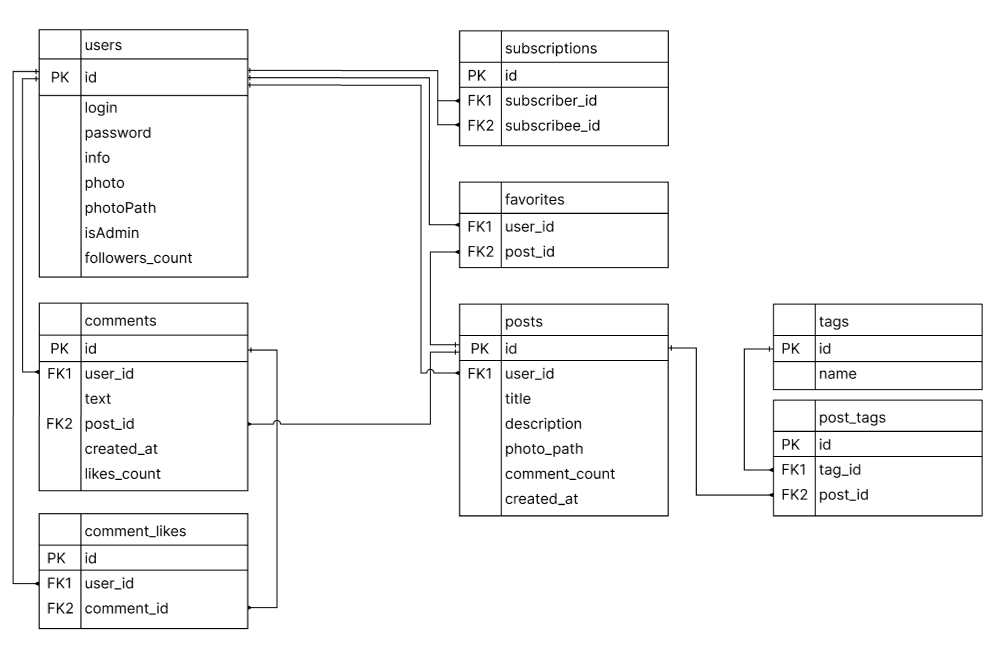
****

Рисунок 2.2 – Схема БД проекта

В проектируемой базе данных располагается 5 таблиц: 4 из которых основные для хранения сущностей.

* 1. **Описание информационных объектов и ограничений целостности.**

Далее будет описана структура каждой таблицы. В таблице users хранятся данные для идентификации пользователей, зарегистрированных в приложении, и пользователей, имеющих права модератора, posts – данные о загруженных изображениях, comments – список комментариев к опубликованным изображениям, favorites для хранения избранных постов пользователя, comment\_likes для лайков к комментариям, tags для хранения тегов к постам, post\_tags для связи между постами и тегами и subscriptions для подписок пользователей на других пользователей.

## **2.2.1 Таблица users.**

Таблица состоит из следующих столбцов:

* id. Хранит уникальный идентификатор пользователя;
* login. Хранит логин пользователя;
* password. Хранит хешированный пароль;
* info. Хранит информацию о пользователе;
* photoPath. Хранит информацию о пути к иконке пользователя;
* photo. Хранит двоичные данные в виде последовательности байт иконки пользователя;
* isAdmin. Хранит логическое указание о том, является ли пользователь администратором;
* followers\_count. Хранит количество подписанных пользователей.
* Данная таблица содержит ограничение первичного ключа для столбца id.

## **2.2.2 Таблица comments.**

В состав таблицы Comments входят следующие столбцы:

* id. Хранит уникальный идентификатор комментария;
* user\_id. Хранит уникальный идентификатор пользователя;
* text. Хранит сообщение комментария;
* post\_id. Хранит уникальный идентификатор поста, к которому оставлен комментарий;
* created\_at. Хранит дату создания комментария;
* likes\_count. Хранит количество лайков к комментарию.

Таблица содержит ограничение первичного ключа для столбца id. Также для столбца user\_id существует ограничения по внешнему ключу к таблице users и для столбца post\_id к таблице posts.

## **2.2.3 Таблица posts.**

Таблица posts состоит из следующих столбцов:

* id. Хранит уникальный идентификатор поста;
* user\_id. Хранит уникальный идентификатор пользователя, который создал пост;
* title. Хранит название поста;
* description. Хранит краткое описание поста;
* photo\_path. Хранит путь к графическому изображению;
* comment\_count. Хранит количество комментариев к посту;
* created\_at. Хранит дату созданияпоста.

Таблица содержит ограничение первичного ключа для столбца id и для столбца user\_id существует ограничения по внешнему ключу к таблице users.

## **2.2.4 Таблица favorites.**

В состав таблицы favorites входят следующие столбцы:

* user\_id. Хранит уникальный идентификатор пользователя, добавившего пост в избранное;
* post\_id. Хранит уникальный идентификатор поста, который добавили в избранное.

Для столбца user\_id существует ограничения по внешнему ключу к таблице users и для столбца post\_id существует ограничения по внешнему ключу к таблице posts.

## **2.2.5 Таблица comment\_likes.**

В состав таблицы comment\_likes входят следующие столбцы:

* id. Хранит уникальный идентификатор лайка;
* user\_id. Хранит уникальный идентификатор пользователя, поставившего лайк к комментарию;
* comment\_id. Хранит уникальный идентификатор комментария к которому поставили лайк.

Для столбца id таблица содержит ограничение первичного ключа. Таблица для столбцов user\_id и comment\_id содержит ограничение внешнего ключа для таблиц Users и Comments соответственно.

## **2.2.6 Таблица tags.**

В состав таблицы tags входят следующие столбцы:

* id. Хранит уникальный идентификатор тега;
* name. Хранит название тега.

Для столбца id таблица содержит ограничение первичного ключа.

## **2.2.7 Таблица post\_tags.**

В состав таблицы tags входят следующие столбцы:

* id. Хранит уникальный идентификатор каждой отдельной связи между постом и тегом;
* post\_id. Хранит уникальный идентификатор поста, к которому относится тег;
* tag\_id. Хранит уникальный идентификатор тега, которой добавляется к посту.

Для столбца id таблица содержит ограничение первичного ключа. Таблица для столбцов user\_id и tag\_id содержит ограничение внешнего ключа для таблиц users и tags соответственно.

## **2.2.8 Таблица subscriptions.**

В состав таблицы tags входят следующие столбцы:

* id. Хранит уникальный идентификатор подписки;
* subscriber\_id. Хранит уникальный идентификатор пользователя, который оформляет подписку на другого пользователя;
* subscribee\_id. Хранит уникальный идентификатор пользователя, на которого оформляют подписку.

Для столбца id таблица содержит ограничение первичного ключа.

* 1. **Вывод по разделу.**

В данном разделе была разработана архитектура проекта, составлена обобщенная структура управления приложением и диаграмма UML. Были описаны все таблицы базы данных и ограничения целостности к строкам таблицы.

1. **Разработка модели базы данных.**

## **3.1. Создание необходимых объектов.**

**3.1.1 Таблицы**

Таблица — это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк.

База данных данного курсового проекта содержит 8 таблиц, которые описаны в главе 2.

**3.1.2 Процедуры**

Процедуры — это именованные блоки кода на языке PL/pgSQL, которые хранятся в базе данных и могут быть вызваны при необходимости. Они используются для группировки и переиспользования кода, который должен выполняться несколько раз, а также для уменьшения нагрузки на сеть при обращении к базе данных. Процедуры могут содержать в себе SQL-запросы, циклы, условные операторы, переменные и многое другое.

Процедуры, разработанные в рамках курсового проекта:

* add\_new\_user. Процедура регистрации нового пользователя;
* authenticate\_user. Процедура авторизации пользователя;
* make\_user\_moderator. Процедура смены роли;
* subscribe\_to\_user. Процедура подписки одного пользователя на другого;
* unsubscribe\_from\_user. Процедура отписки одного пользователя от другого;
* add\_post. Процедура добавление нового поста;
* delete\_post. Процедура удаления поста;
* add\_tags\_to\_post. Процедура добавления тегов к посту;
* add\_comment. Процедура добавления комментария к посту;
* delete\_comment. Процедура удаление комментария к посту;
* add\_likeCom. Процедура добавления лайка к комментарию;
* remove\_likeCom. Процедура удаления лайка с комментария;
* add\_to\_favorites. Процедура добавления поста в избранное;
* remove\_from\_favorites. Процедура удаления поста из избранного;
* export\_post\_to\_json. Процедура экспорта постов в json файл;
* fill\_post. Процедура заполнения таблицы posts 100000 строк.

**3.1.3 Функции**

Функции являются объектами базы данных, которые позволяют выполнять определенные действия на стороне сервера базы данных. Функции могут использоваться для выполнения различных задач, таких как обработка данных, преобразование данных, агрегация данных и т. д.

Функции, разработанные в рамках курсового проекта:

* find\_posts\_by\_tags. Функция поиска поста по тегам;
* check\_user\_fields. Функция проверки заполнения данных при регистрации;
* update\_followers\_count\_func. Функция обновления количества подписчиков у пользователя;
* export\_post\_to\_json. Функция экспорта поста в json файл.
* update\_comments\_count\_func. Функция обновления количества комментариев у поста;
* update\_comment\_likes\_counter. Функция обновления количество лайков у комментария.ч

**3.1.4 Триггеры**

Триггеры в PostgreSQL — это функции, которые автоматически вызываются при определенных событиях, происходящих в базе данных, таких как вставка, обновление или удаление строк в таблицах. Триггеры могут выполнять различные действия, например, проверять данные перед вставкой или изменением, обновлять связанные данные или записывать изменения в другие таблицы.

Триггеры, разработанные в рамках курсового проекта:

* user\_insert\_trigger. Триггер на добавление строк в таблицу users;
* update\_followers\_count\_trigger. Триггер на обновление количества подписчиков у пользователя;
* update\_comments\_count\_trigger. Триггер на обновление количества комментариев;
* update\_comment\_likes\_trigger. Триггер на обновления количества лайков.

**3.1.5 Индексы**

Индексы являются объектами базы данных, которые ускоряют выполнение запросов на поиск, сортировку или группировку данных в таблицах. Индексы представляют собой специальные структуры данных, которые позволяют быстро находить записи в таблице на основе значений столбцов, входящих в индекс.

Индексы, разработанные в рамках курсового проекта:

* tags\_name\_idx. Индекс для поиска постов по тегам;
* post\_user\_idx. Индекс для поиска постов по пользователю.

## **3.2. Описание используемой технологии.**

Для создания базы данных для локального фотохостинга была выбрана технология мультимедийные типы в БД.

Мультимедийные типы в базах данных (БД) позволяют хранить и обрабатывать различные типы мультимедийных данных, таких как изображения, аудио- и видеофайлы.

Существует несколько типов мультимедийных данных, которые могут быть хранены в БД:

1. Изображения — это статические графические файлы, такие как JPEG, PNG, GIF и другие. Они могут быть хранены в БД как бинарные данные.

2. Аудиофайлы — это файлы, содержащие звуковые данные, такие как MP3, WAV и другие. Они могут быть хранены в БД как бинарные данные или в виде ссылок на файлы на сервере.

3. Видеофайлы — это файлы, содержащие видеоданные, такие как MP4, AVI, MKV и другие. Они могут быть хранены в БД как бинарные данные или в виде ссылок на файлы на сервере.

Для работы с мультимедийными типами в БД используются специальные технологии. Например, для работы с изображениями используется технология BLOB (Binary Large Object), которая позволяет хранить бинарные данные в БД. Для работы с аудио- и видеофайлами используются технологии, такие как URL-адреса и ссылки на файлы на сервере.

* 1. **Вывод по разделу.**

В данном разделе было описаны созданные необходимые объекты для работы с разрабатываемой базой данных фотохостинга и описана используемая технология: мультимедийные типы в БД.

1. **Установка, настройка и использование PostgreSQL.**

PostgreSQL (сокращенно Postgres) — это мощная, бесплатная и открытая реляционная система управления базами данных (СУБД), которая является одной из самых популярных СУБД в мире. PostgreSQL предоставляет множество возможностей, включая поддержку многих языков программирования, индексы, триггеры, хранимые процедуры, функции и многое другое. Он может использоваться как в небольших приложениях, так и в больших корпоративных системах. PostgreSQL также известен своей надежностью, масштабируемостью и производительность.

## **4.1. Создание таблиц.**

При разработке приложения для курсового проекта была использована СУБД PostgreSQL.

Для реализации базы данных была создана база данных, которая хранит в себе набор схем, объектов и связанных структур как отдельная база данных**.**

В базе данных представлены 8 таблиц: users, subscriptions, posts, tags, post\_tags, favorites, comments и comment\_like.

## **4.2. Создание ролей для разграничения доступа.**

Для разграничения доступа создаются две роли: модератора и пользователя. В возможности модератора входит:

* Удаление постов всех пользователей;
* Удаление комментариев всех пользователей;
* Назначение роли модератора другим пользователям.

В возможности пользователя входит:

* Добавление и удаление своих постов;
* Добавление и удаление своих комментариев;
* Добавление и удаление лайков на комментарии;
* Подписка и отписка от других пользователей;
* Добавление тегов к постам;
* Поиск постов по тегам.

За роль пользователя отвечает булевый столбец isAdmin в таблице users.

## **4.4. Описание процедур экспорта и импорта.**

JSON — это формат, который хранит структурированную информацию и в основном используется для передачи данных между сервером и клиентом.

Файл JSON представляет собой более простую и лёгкую альтернативу расширению с аналогичными функциями XML (Extensive Markup Language).

Часто возникает необходимость импортировать и экспортировать JSON-файлы, в данной курсовой работе используются функция row\_to\_json() для экспорта. Процедура экспорта используются для таблицы Posts, так как эта таблица является важнейшей в базе данных. Пример создания процедуры можно посмотреть в листинге 4.1.

CREATE OR REPLACE PROCEDURE export\_post\_to\_json()

LANGUAGE plpgsql

AS $$

BEGIN

COPY (SELECT row\_to\_json(t)

FROM(

select id, user\_id, title, description, photo\_path, created\_at from posts)t) TO 'D:\course\_project\posts.json';

END;

$$;

call export\_post\_to\_json();

Листинг 4.1 – Пример создания процедуры экспорта в json файл

## **4.5. Тестирование производительности базы данных.**

Производительность БД является решающим фактором эффективности управленческих и коммерческих приложений. Если поиск или запись данных выполняется медленно – способность к нормальной работе приложения падает. Существует единственный путь выяснить причину плохой производительности – выполнить количественные измерения и определить, что является причиной проблемы производительности.

Для того чтобы правильно организовать процесс тестирования БД, тестировщики должны обладать хорошими знаниями SQL и DML и иметь ясное представление о внутренней структуре БД. Это самый лучший и надежный способ тестирования БД особенно для приложений с низким и средним уровнем сложности. Данный метод не только дает уверенность, что тестирование выполнено качественно, но также повышает мастерство написания SQL-запросов.

В MS SQL Server оптимизация запросом в основном заключается в построение индексов над таблицами, и изменением плана запроса.

Для тестирования производительности в таблицу Posts были добавлены 100 000 записей. Код процедуры добавления 100 000 записей представлен в листинге 4.2.

CREATE OR REPLACE PROCEDURE export\_post\_to\_json()

LANGUAGE plpgsql

as $$

declare

us\_id integer;

begin

for i in 1..100000 loop

select "id" into us\_id from users where is\_admin = false order by random() limit 1;

insert into posts(user\_id,title, description,photo\_path) values(us\_id, 'title' || i::text, 'description' || i::text, 'path');

end loop;

end;

$$;

Листинг 4.2 – Добавление 100 000 строк в таблицу posts

Ниже на рисунке 4.1 представлен результат select-запроса к таблице posts до добавления индекса. Время выполнения составляет 131 мс.

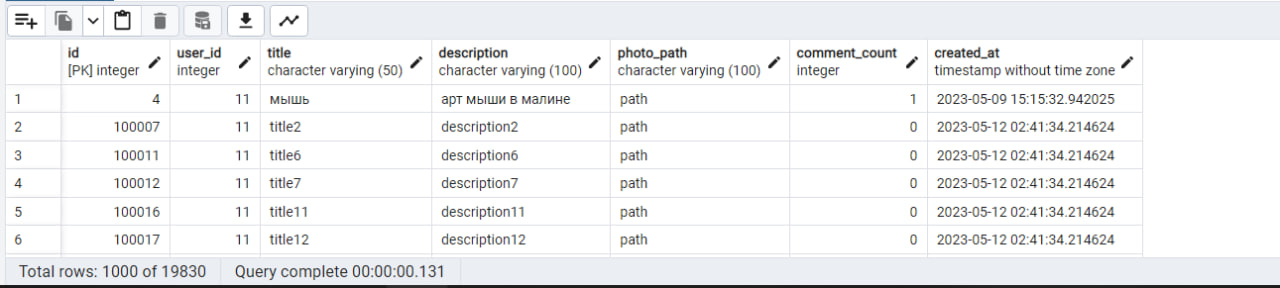


Рисунок 4.1 – Результат Select-запроса к таблице posts без индекса

На рисунке 4.2 представлен результат select-запроса к таблице posts после добавления индекса. Время выполнения уже составляет 53 мс.

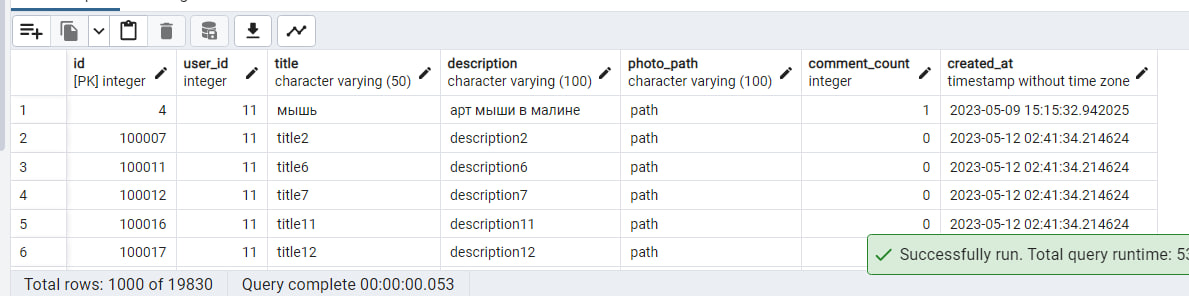


Рисунок 4.2 – Результат Select-запроса к таблице posts с индексом

## **4.6. Вывод.**

В этом разделе была описана выбранная СУБД, рассказано о разработанных процедурах, триггерах, функциях и индексах. Кроме того, продемонстрирован листинг экспорта данных из таблицы posts в json файл. Также была проверена работоспособность разработанной БД посредством добавления в одну из таблиц 100000 строк.

1. **Тестирование, проверка работоспособности и анализ полученных результатов.**

## **5.1. Тестирование клиентской части.**

Для начала тестирования клиентской части необходимо создать пользователя, используя регистрацию в разработанном приложении. Также будет создано несколько других пользователей и модератор для проверки работы некоторых функций. Регистрация пользователя представлена на рисунке 5.1.

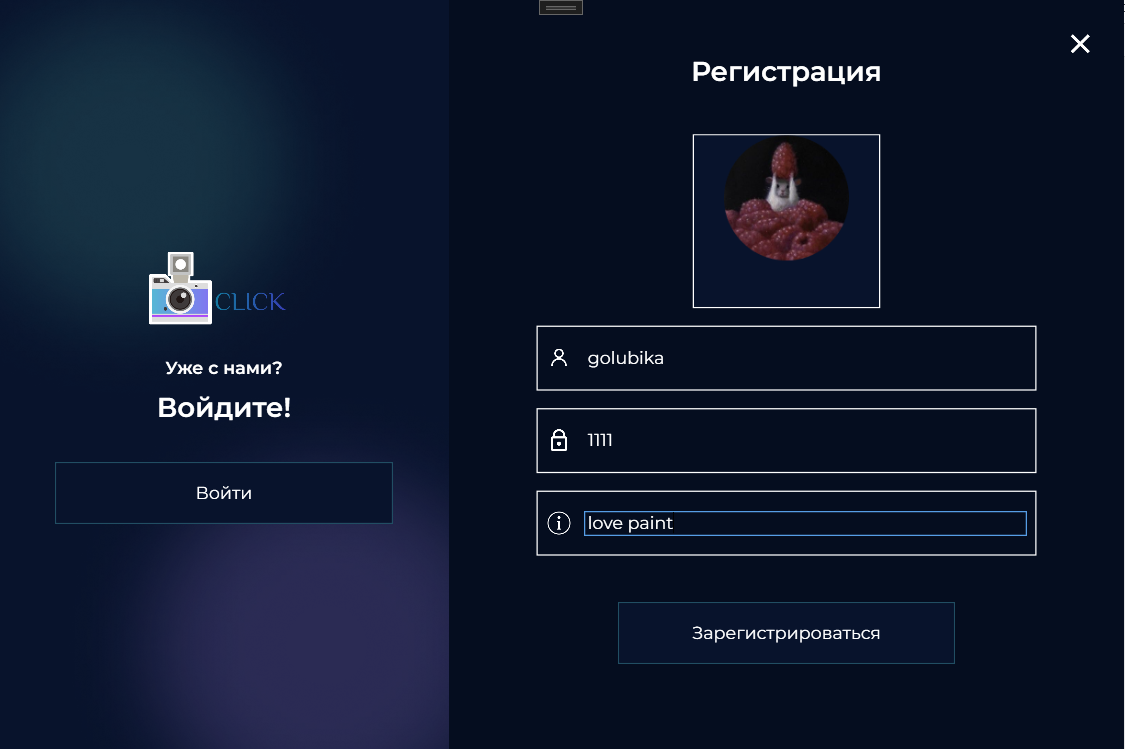


Рисунок 5.1 – Регистрация пользователя

После нажатия на кнопку зарегистрироваться данные о пользователе добавляются в разработанную в данном курсовом проекте БД. Рисунок, подтверждающий добавление нового пользователя в таблицу users продемонстрирован под пунктом 5.2.

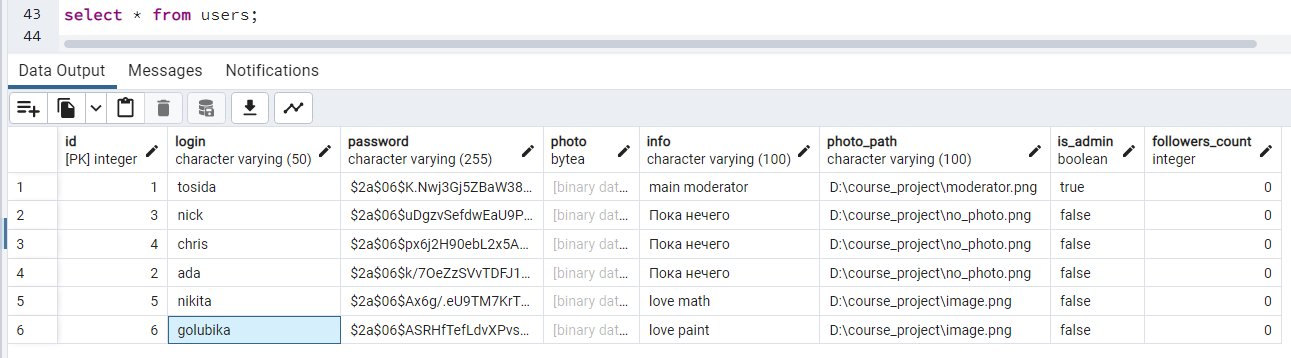


Рисунок 5.2 – Добавление нового пользователя в БД

На данном этапе происходит реализация технологии мультимедийные типы в БД. С помощью разработанного приложения пользователь выбирает себе иконку, которая хранится в базе данных как ссылка и как последовательность байт.

Также пользователь имеет возможность авторизации. Для этого необходимо ввести данные о логине и пароле. Авторизация происходит только в том случае, если такой пользователь уже существует в БД. Авторизация продемонстрирована на рисунке 5.3.

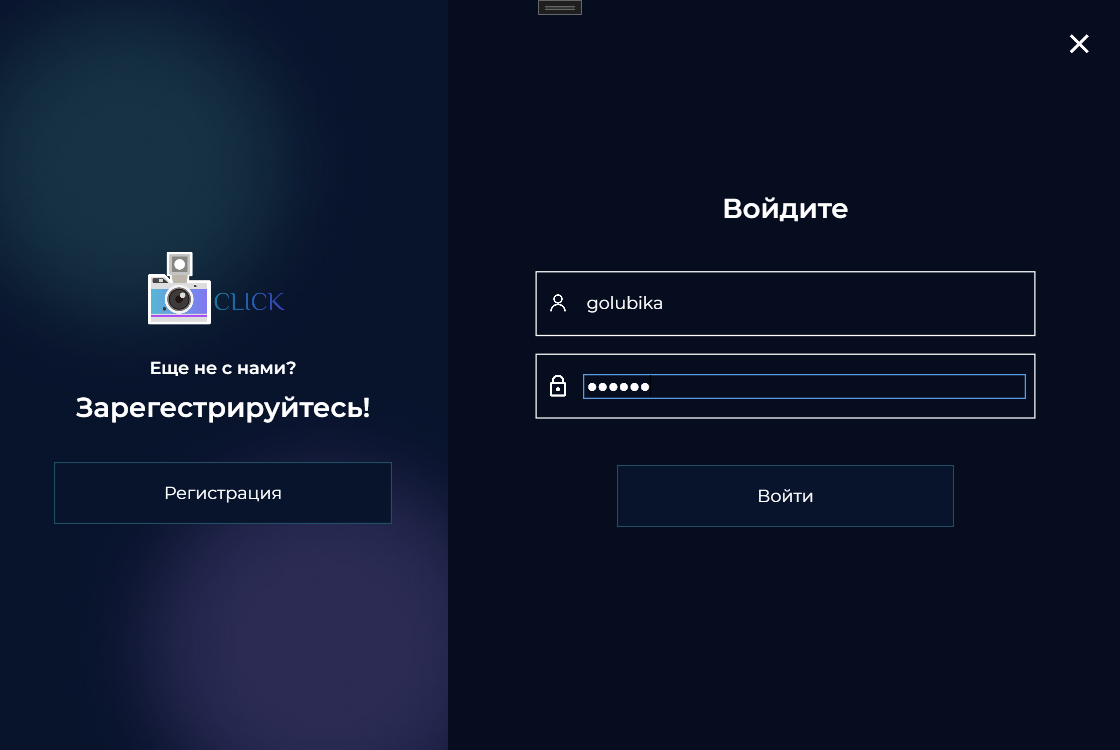


Рисунок 5.3 – Авторизация пользователя

Если авторизация прошла успешно, то есть такой пользователь существует и все данные указаны верно, то клиент входит в свой аккаунт и попадает в основную часть приложения. Показано на рисунке 5.4.



Рисунок 5.4 – Успешная авторизация

Теперь протестируем создание поста. Для этого необходимо использовать процедуру add\_post. Для демонстрации был создан блок кода, автоматически добавляющий посты от имени разных существующих пользователей. Продемонстрировано на рисунке 5.5.

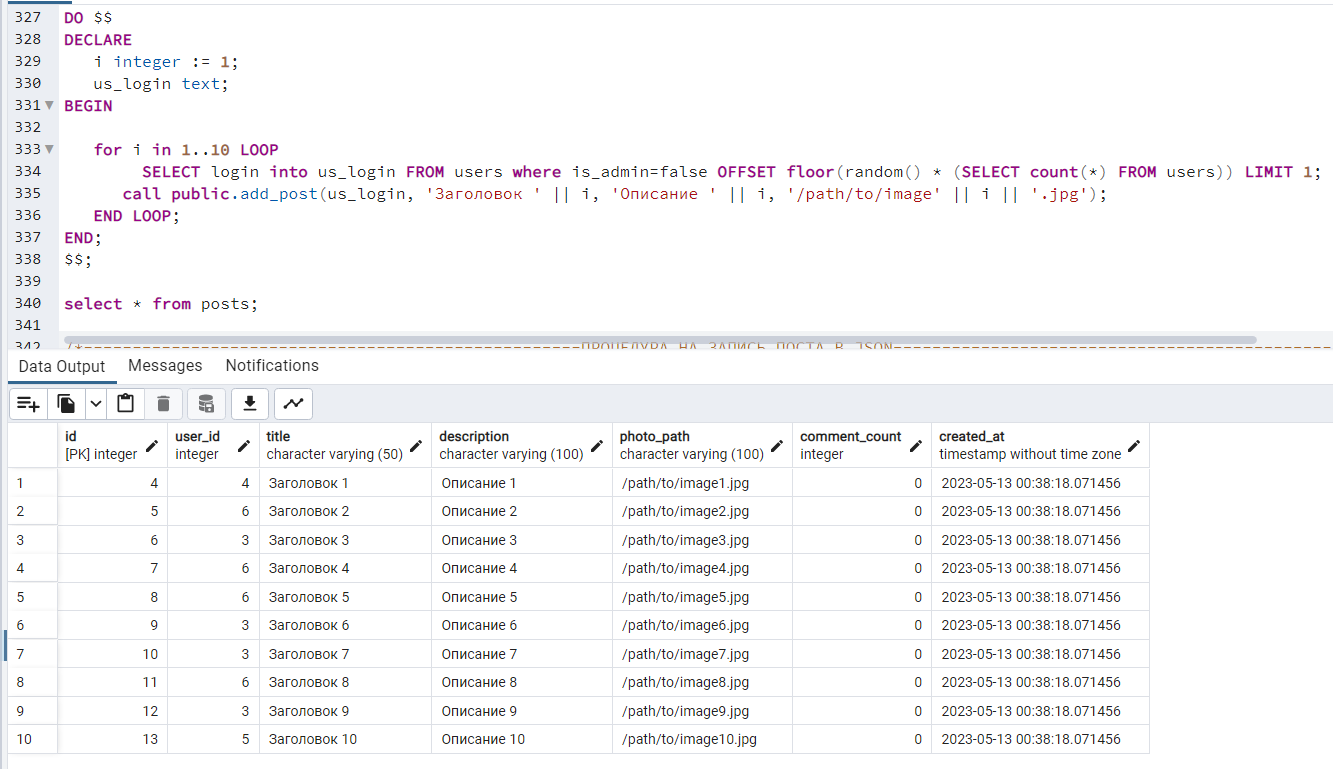


Рисунок 5.5 – Выполнение процедуры добавления поста

К постам пользователи могут добавлять теги, по которым можно осуществлять поиск постов. Добавление тегов осуществляется с помощью процедуры, выполнение которой показано на рисунке 5.6.

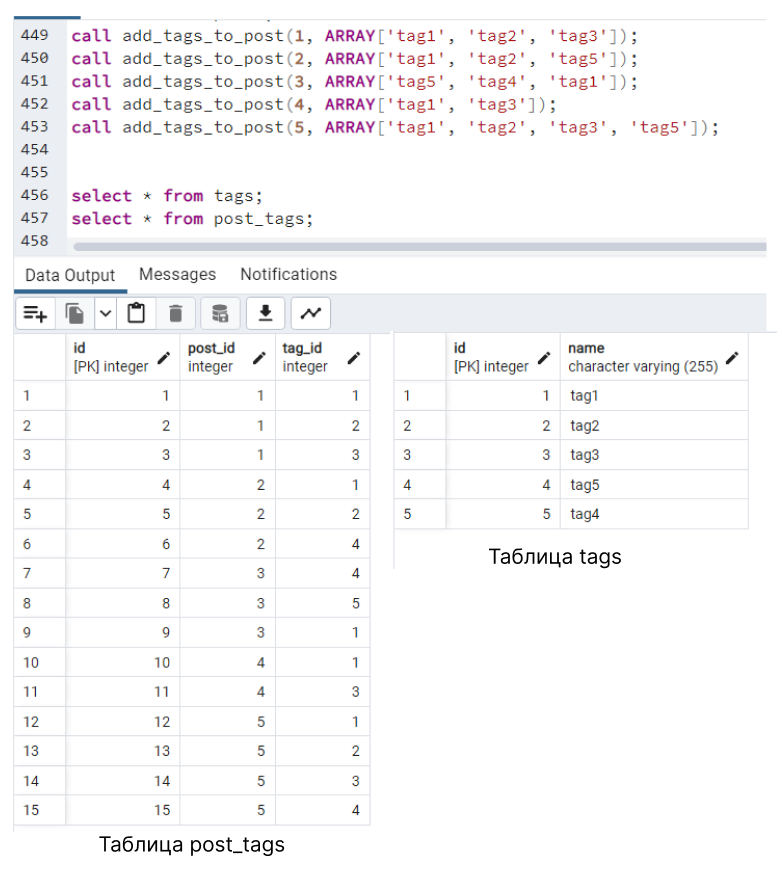


Рисунок 5.6 – Выполнение процедуры добавления тегов

Как говорилось ранее, пользователи могут осуществлять поиск по тегам, для чего применяется функция find\_posts\_by\_tags. Функция выводит все посты включающие указанные теги. Продемонстрировано на рисунке 5.7.



Рисунок 5.7 – Выполнение функции поиска постов по тегам

Пользователям добавлена возможность удаления тегов к своим постам. Если тег удалили все пользователи, то он удаляется и из таблицы tags.

Для поста пользователи могут оставлять комментарии, для этого используется процедура add\_comment. Был также разработан блок кода, вызывающий данную процедуру несколько раз с произвольными значениями. Можно увидеть на рисунке 5.8.

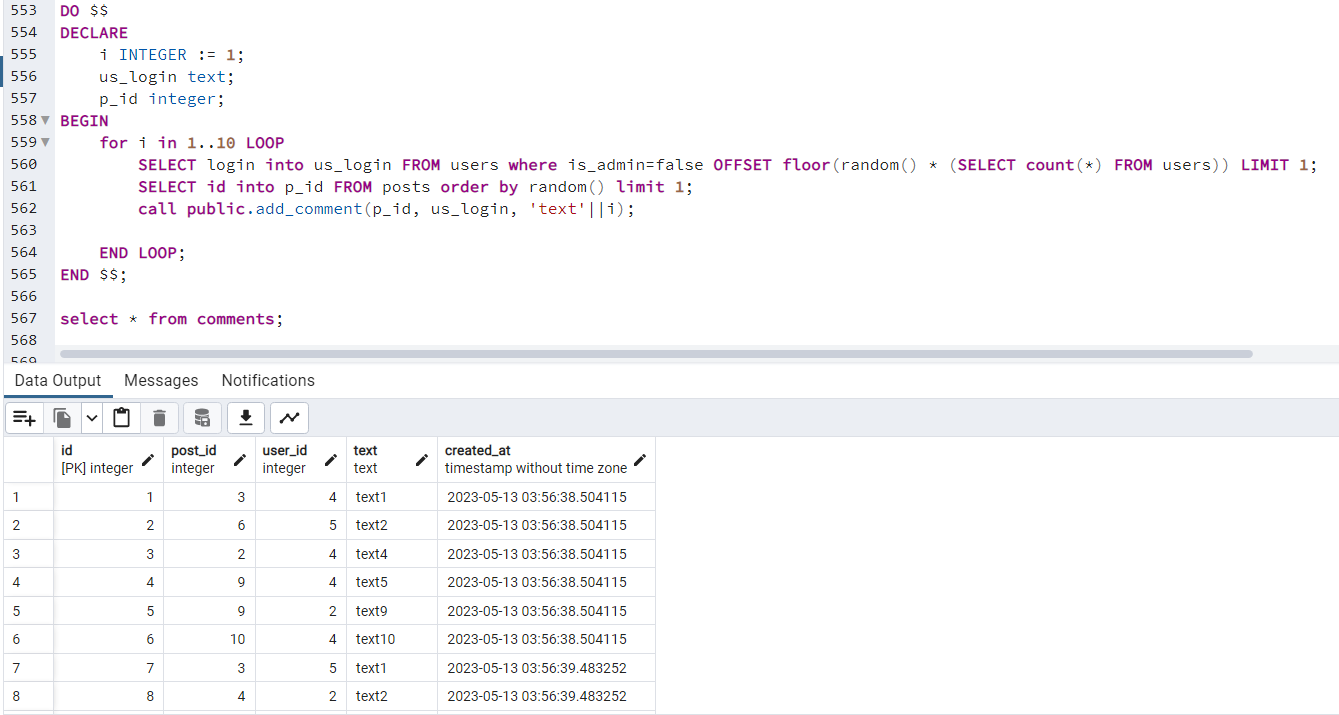


Рисунок 5.8 – Выполнение процедуры добавления комментария

В таблице posts есть вычисляемый столбец comment\_count, который рассчитывается на основе добавленных комментариев к посту. Это было осуществлено за счет триггера, который срабатывает при добавлении или удалении данных в таблице posts. Каждый пользователь имеет возможность удалять свои посты. При этом происходит удаление всех связанных комментариев и лайков.

К комментариям пользователь может ставить лайки. Это выполняется с помощью процедуры add\_likeCom. Продемонстрировано на рисунке 5.9.

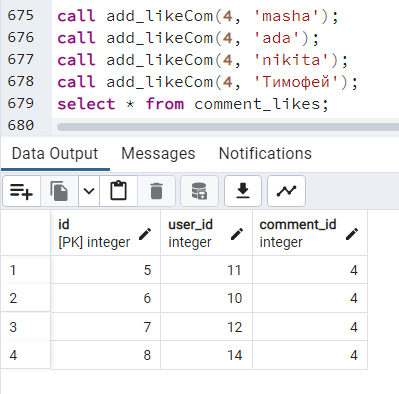


Рисунок 5.9 – Выполнение процедуры добавления лайков на комментарии

В таблице comments также есть вычисляемый столбец, отвечающий за количество лайков на комментарии. Это также осуществляется за счет разработанного триггера. Пользователь может удалить лайк, но только свой.

Клиент может добавить любой пост в избранное. Выполнение функции можно увидеть на рисунке 5.10.

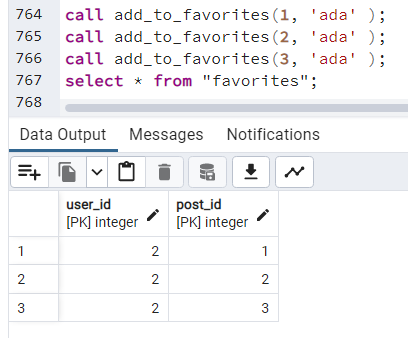


Рисунок 5.10 – Выполнение функции добавления в избранное

Из избранного также можно удалять.

## **5.2. Тестирование области работы модератора.**

Для тестирования возможностей модератора, необходимо чтобы в таблице users был пользователем со значением столбца isAdmin равным true.

Такой пользователь есть, показано на рисунке 5.11.

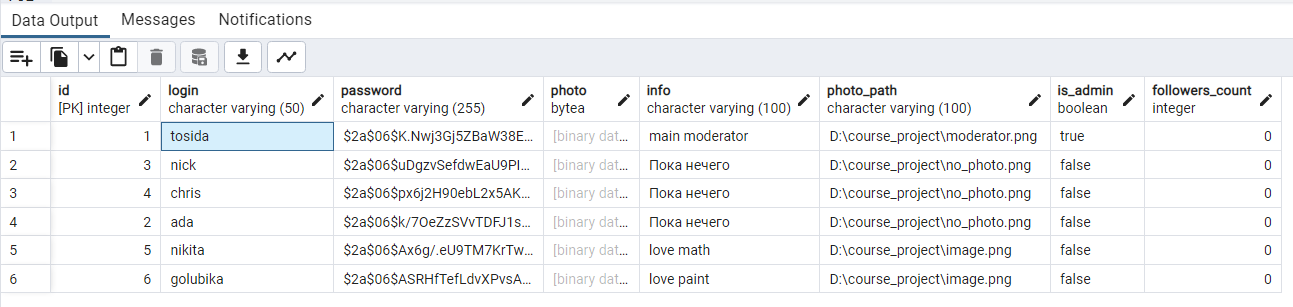


Рисунок 5.11 – Список всех зарегистрированных пользователей

Основная возможность модератора – это удаление нежелательного контента. Нежелательным контентом могут выступать посты и комментарии.

Если пользователи могут удалять только свои комментарии и посты, то модератор может удалять комментарии и посты всех пользователей. Удаление поста модератором продемонстрировано на рисунках 5.12 и 5.13.

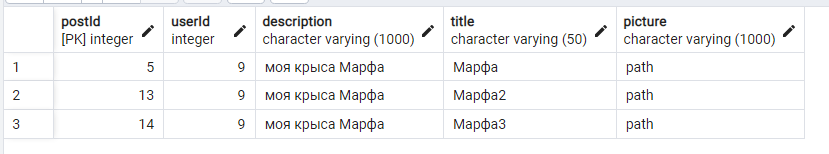


Рисунок 5.12 – Список всех постов в базе данных до удаления

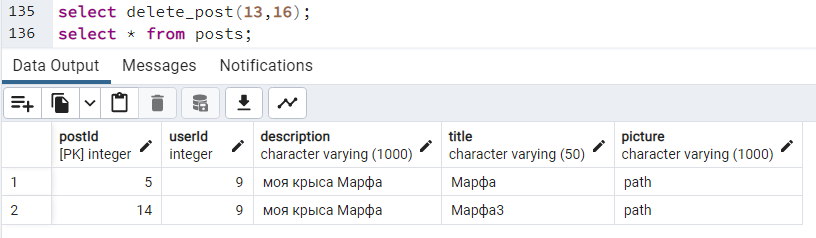


Рисунок 5.13 – Список всех постов в базе данных после удаления

Удаление комментария модератором продемонстрировано на рисунках 5.14 и 5.15.

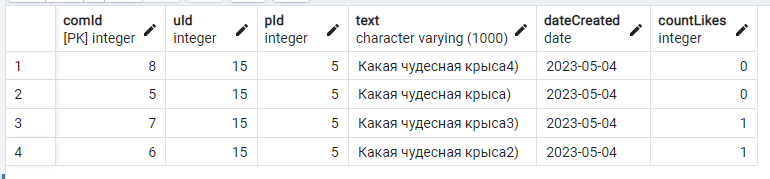


Рисунок 5.14 – Список всех комментариев в базе данных до удаления

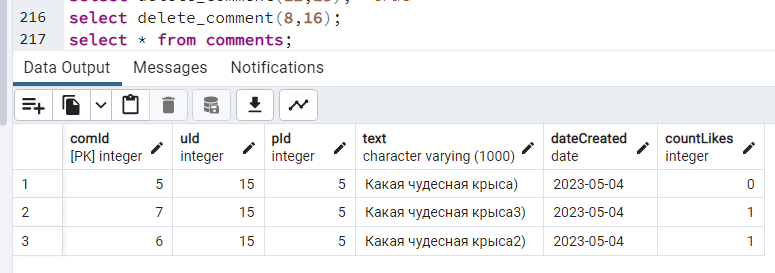


Рисунок 5.15 – Список всех комментариев в базе данных после удаления

Также у модератора есть возможность назначения обычного пользователя модератором. Выполнение это функции показано на рисунке 5.16 - 5.18.



Рисунок 5.16 – Список всех пользователей в базе данных до назначения нового модератора

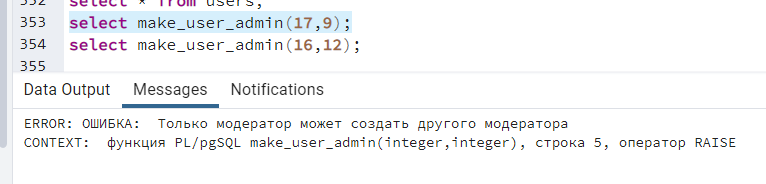


Рисунок 5.17 – Выполнение функции назначения не модератором

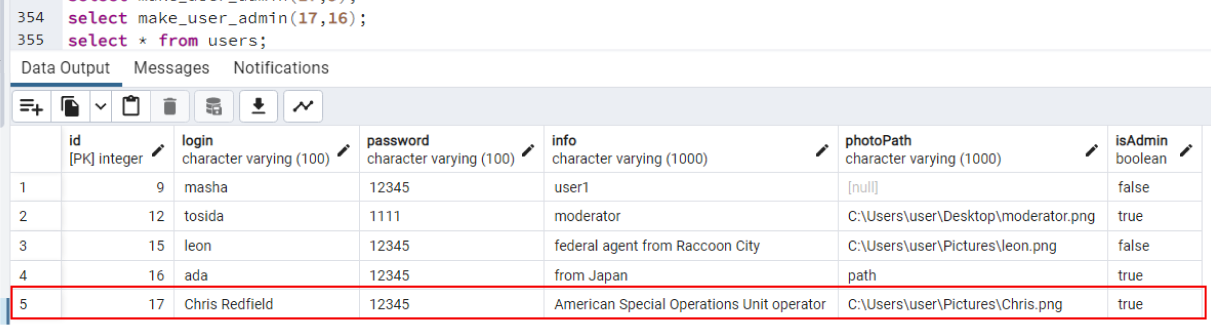


Рисунок 5.18 – Список всех пользователей в базе данных после назначения нового модератора

## **5.3. Вывод.**

В данном разделе были протестированы возможности клиентов и модераторов. По итогам тестирования можно сказать, что все работает корректно.

## **Заключение**

В ходе данного курсового проекта: была разработана модель и объекты базы данных для фотохостинга; были разработаны, описаны и применены на практике процедуры экспорта и импорта данных; было проведено тестирование производительности базы данных, в ходе которого были разработаны индексы, ускорившие получение данных; разработаны различные процедуры для взаимодействия с базой данных.

Функционально были выполнены следующие задачи:

* + - авторизация и регистрация пользователей;
    - добавление и удаление постов;
    - добавление и удаление комментариев;
    - добавление и удаление лайков к комментариям;
    - подписка и отписка на пользователей;
    - добавление и удаление постов из избранного;
    - добавление тегов к постам;
    - поиск постов по тегам.

## **Список использованных литературных источников**

1. Pinterest [Электронный ресурс] / Режим доступа: URL https://www.pinterest.com/ - Дата доступа: 22.03.2023

2. Tumblr [Электронный ресурс] / Режим доступа: URL https://www.tumblr.com/ - Дата доступа: 22.03.2023

3. Imgbb [Электронный ресурс] / Режим доступа: URL https://ru.imgbb.com/ - Дата доступа: 22.03.2023

4. Постгрес: проектирование и реализация БД / ред. С. Л. Шумский. - СПб.: БХВ-Петербург, 2009. - 287 с.

5. Постгрес: cправочник / Дж. Вайт, Р. Мак-Кинни, П. Гершенфельд и др. - М.: ООО "И.Д. Вильямс", 2003. - 688 с.

6. SQL и реляционные базы данных: учебное пособие / М. П. Баранов, С. М. Бабенко, Ю. В. Михайленко. – М.: Издательский центр "Академия", 2019. - 200 с.