**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ  
МЕДИА-СЕРВИСА «WEPICSYNC»**

***ОТЧЁТ ПО ДИПЛОМНОЙ РАБОТЕ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнили:

Студент группы 1912с

\_\_\_\_\_\_\_\_\_\_\_ / Матохнюк А.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Проверил:

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_\_ / Уляхин В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Оглавление

[Введение 3](#_Toc181154449)

[Информация о проекте 3](#_Toc181154450)

[Назначение документа 3](#_Toc181154451)

[Глоссарий 3](#_Toc181154452)

[Общее описание 4](#_Toc181154453)

[Функционал сервера 5](#_Toc181154454)

[Ролевая модель 5](#_Toc181154455)

[Хранимые данные 6](#_Toc181154456)

[Требования к REST API 7](#_Toc181154457)

[Общие требования 7](#_Toc181154458)

[Авторизация 8](#_Toc181154459)

[Работа с альбомами 9](#_Toc181154460)

[Работа с альбомными доступами 10](#_Toc181154461)

[Работа с картинками 11](#_Toc181154462)

[Работа с тегами 13](#_Toc181154463)

[Требования к мобильному приложению 15](#_Toc181154464)

[Проектирование и разработка базы данных 16](#_Toc181154465)

[Концептуальная модель данных 16](#_Toc181154466)

[Логическая модель данных 16](#_Toc181154467)

[Физическая модель данных 17](#_Toc181154468)

# Введение

## Информация о проекте

|  |  |
| --- | --- |
| Название проекта | Медиа-сервис «Wepicsync» |
| Цель проекта | Спроектировать и разработать медиа-сервис «Wepicsync» |
| Задачи проекта | 1. Разработать базу данных сервиса для MySQL. 2. Спроектировать структуру API сервиса. 3. Разработать API сервер сервиса на фреймворке Fastify. 4. Разработать сайт сервиса на фреймворке Next.js. 5. Разработать приложение синхронизации картинок на MAUI. 6. Разработать автоматизированные тесты и провести тестирование. |

## Назначение документа

Настоящий документ рассчитан на использование в разработке в качестве ключевого документа, определяющего требование по реализации задачи по разработке медиа-сервиса «Wepicsync».

## Глоссарий

|  |  |
| --- | --- |
| **Понятие** | **Определение понятия** |
| API | API (интерфейс программирования приложений) – это набор определенных правил и соглашений, который используется для взаимодействия между различными программными компонентами. API определяет, как различные части программного обеспечения могут взаимодействовать друг с другом, обычно путем обмена данными и выполнения определенных операций. |
| Postman | Postman – это инструмент для тестирования и разработки API. Он предоставляет удобный интерфейс для отправки HTTP запросов к API, а также для анализа ответов. |
| Draw.io | Это бесплатный онлайн инструмент для создания диаграмм и схем. Он предоставляет широкий набор инструментов и элементов, которые позволяют пользователям создавать различные типы диаграмм, такие как организационные диаграммы, блок-схемы, ER-диаграммы, сетевые диаграммы, диаграммы потока процессов и многое другое. |
| CRUD | CRUD – акроним, обозначающий четыре базовые функции, используемые при работе с данными: создание, чтение, модификация, удаление. |

# Предметная область

Разрабатываемый сервис предназначен для удобного хранения, управления и получения пользователями своих картинок и видео на удаленном сервере. Основная цель сервиса – обеспечить надёжное хранение медиафайлов путём повышения числа копий. Пользователи смогут загружать на сервер фотографии и снятые видео с телефона, скаченные коллекции медиафайлов с ПК через удобный и интуитивно понятный интерфейс, организовывать их в альбомы, добавлять описания и теги для упрощения поиска. Сервис также предоставит возможность делиться фотографиями с другими пользователями или выставлять на публику посредством приглашения через ссылки.

Сервис ориентирован:

1. Для лица, который хочет хранить медиафайлы на своём сервере, не платя подписки и не делясь со сторонними компаниями своими личными фотографиями и видео.
2. Для команды/организации, которая хочет поделится большим количеством медиафайлов для публики или конкретным лицам в удобном виде.

# Требования к функциональности

В системе есть следующие роли:

1. Гость — авторизация, регистрация, получение публичных настроек
2. Пользователь — манипуляции над альбомами
3. Администратор — манипуляции над учётными записями

Гости могут авторизироваться и регистрироваться если разрешено настройками. Авторизация производится по токеном, генерируемым по авторизации и отзываемых при выходе. Авторизация на другом устройстве не должна вызывать «вылет» из аккаунта на всех устройствах.

Администратор может создать учётные записи и задавать квоты для каждого.

Пользователь может создать альбом с указанием какие устройства синхронизируют на сервер этот альбом.

Пользователь может загрузить в альбом картинки и просмотреть их. Есть максимальная квота на размер всех загруженных картинок (регулирующаяся профилем).

Пользователь может для себя создать теги и назначать несколько тегов на каждую свою картинку.

Пользователь может создать код приглашения на свой альбом. Он может сделать альбом публичным или установить лимит вступлений и дату просрочки или минут жизни.

Пользователь может добавить доступ к альбому через электронную почту другого пользователя.

Пользователь может получить доступ к чужому альбому путём перехода по коду приглашения (ссылке) от автора альбома.

Пользователь-владелец может отозвать выданные доступы и коды приглашения. Пользователь, который имеет доступ, может отказаться от доступа.

Администратор может менять следующие настройки: разрешённые типы файлов (MIME), список размеров превью, лимит предупреждений до бана, квота на хранилище для каждого пользователя и лимит занятого место на диске (в процентах) при котором будет отключена загрузка на сервер.

Гость может получить следующую информацию о сервере: разрешённые типы файлов (MIME), список размеров превью, лимит предупреждений до бана, квота на хранилище для каждого пользователя, статус разрешения загрузки (который автоматически меняется при достижении лимита занятого места на диске) и список типов жалоб.

При настроенном планировщике сервер каждые 10 минут проверяет и кеширует размер диска и размер свободного места.

## Ролевая модель

Исходя из основных требований функциональности системы можно построить таблицу, разделяя роли и функциональность (см. таблицу 1):

Таблица 1. Ролевая модель

| **Функционал** | | **Админ** | **Юзер** | **Гость** |
| --- | --- | --- | --- | --- |
| Пользователи | Регистрация | да (если включено) | | |
| Авторизация | да | | |
| Выход | да | | нет |
| Просмотр | да | да (частич.) |
| Изменение | да | да (себя) |
| Альбомы | CRUD | да (свои) | | нет |
| Просмотр публичного | да | | |
| Приглашения | Вступление в альбом | да (на чужие) | | нет |
| Создание | да (на свои альбомы) | |
| Удаление |
| Картинки | CRUD | да (свои) | | нет |
| Теги | CRUD | да (свои) | | нет |
| Добавление к картинке | да (к своим картинкам) | |
| Удаление с картинки |
| Доступы на альбомы | Создание | да (на свои альбомы) | | нет |
| Удаление | да (на свои альбомы, на чужие) | |
| Настройки | Получение публичных | да | | |
| Получение приватных | да | нет | |
| Изменение |
| Синхронизации меж серверами | CRUD | да | нет | |

## Диаграмма прецедентов

Поведение системы (т.е. функциональность, которую она обеспечивает) описывают с помощью функциональной модели **— *диаграмма прецедентов (вариантов использования, use case diagram)* —** которая отображает системные прецеденты (use cases, случаи использования), системное окружение (действующих лиц, актеров, actors) и связи между ними.

Между прецедентами также возможны взаимоотношения, которые описываются отношениями двух типов: включения (include) и расширения (extend).

В системе выделено 3 лица как роли пользователей и 8 групп прецедентов как общие объекты случаев использования и была разработана диаграмма прецедентов (см. рисунок 1).

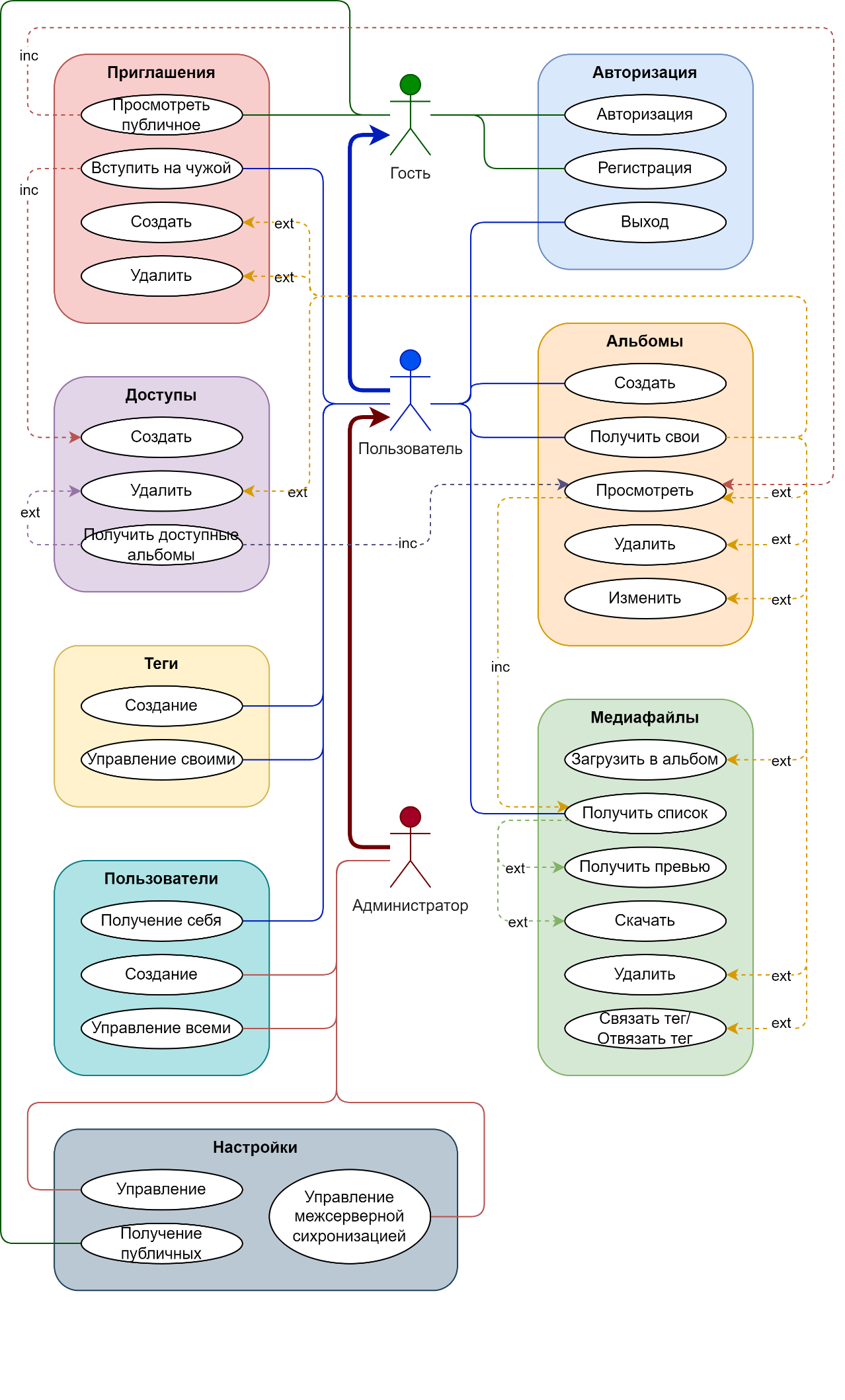


Рисунок 1. Диаграмма прецедентов

## Диаграммы деятельностей

Диаграмма деятельности — один из способов моделирования потока событий. Состоит из обозначений состояний (начала: кружка; и концов: кружок в круге), деятельностей (скруглённый прямоугольник), переходов (стрелки), секций (столбцы, делящиеся на кто отвечает за деятельность), условиями (ромбы) и линий синхронизаций (две горизонтальные линии, между которыми находятся действия).

# Технологический стэк

**Серверная часть**

Язык программирования: TS

Фреймворк для разработки: Fastify

Система управление базами данными: Postgres

**Сайт**

Язык программирования, разметки и стилей: TS, HTML и SCSS

Фреймворк для разработки: Next.js

**Мобильное приложение**

Язык программирования и разметки: C# и XAML

Фреймворк для разработки: .NET MAUI

# Хранимые данные

1. **Токены:** значение и связь с пользователем
2. **Пользователи:** никнейм, роль, логин, хеш пароля и состояние бана
3. **Альбомы:** название, связанный пользователь и путь на клиентском устройстве
4. **Доступы к альбомам:** пользователь и альбом
5. **Приглашения доступа:** связанный альбом, ссылка, лимит вступлений и дата просрочки
6. **Картинки:** связанныйальбом, название, размер файла, ширина, высота, дата, хеш и связанные теги
7. **Теги:** название, связанный пользователь

# Требования к REST API

## Общие требования

Должно быть открытое АPI — правила CORS должны разрешать доступ с других доменов.

Идентификацию пользователя для не гостевых функции должно быть организовано посредством Bearer Token.

При попытке доступа к защищенным авторизацией функциям системы во всех запросах необходимо возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 401 Unauthorized |

При попытке доступа авторизованным пользователем к функциям недоступным для своей роли во всех запросах необходимо возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 403 Forbidden |

При попытке получить несуществующий ресурс возвращать ответ следующего вида:

|  |
| --- |
| **Response** |
| **Status:** 404 Not Found |

В случае ошибок связанных с валидацией данных в теле запроса необходимо возвращать следующий ответ:

|  |
| --- |
| **Response** |
| **Status:** 422 Unprocessable Content  **Content-Type:** application/json  **Body:** {  "errors": [  <ключ>: [  <ошибка>,  <ошибка>,  ...  ],  <ключ>: [...],  ...  ]  } |

Вместо поля <ключ> должно быть название ключа запроса, в котором произошла ошибка валидации, а вместо <ошибка> — описание допущенной ошибки в этом ключе. Ошибок в одном ключе может быть множество, как и в одном запросе — множество ключей с ошибками.

## Авторизация

**Регистрация**

Запрос позволяет пользователю зарегистрироваться и вернуть его токен.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/register  **Method:** POST  **Content-Type:** application/json  **Body:** {  "name": **{nickname}**,  "login": **{login}**,  "password": **{password}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "token": **<токен>**  "user": <**данные\_пользователя**>  }  **Возможные статус-коды:** 201, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | api | string | да |  | URL до API | https://api.pics.com |
| 2 | nickname | string | да | мин: 2, макс: 255 | Имя юзера | Testello |
| 3 | login | string | да | мин: 2, макс: 64, уник,  regex: ^[a-zA-Z0-9\_-]+$ | Логин | test |
| 4 | password | string | да | мин: 8, макс: 255 | Пароль | Test123! |

**Авторизация**

Запрос позволяет авторизировать пользователя и получить токен доступа.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/login  **Method:** POST  **Content-Type:** application/json  **Body:** {  "login": **{login}**  "password": **{password}**  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "token":<**токен**>  "user": {  <**"role":** **"admin"**>  "name": "Testello",  "login": "test"  }  }  **Возможные статус-коды:** 200, 401, 422 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | login | string | да | Логин | mylogin |
| 2 | password | string | да | Пароль | Pass123! |

**Выход**

Запрос предназначен для удаления текущего токена пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/logout  **Method:** POST  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 200, 401 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | token | string | да | API токен | 27|fxR7b3H3o5nPYAdtWhbk61do |

## Работа с альбомами

**Создание альбома**

Запрос для создания нового личного альбома пользователя. Возвращает объект альбома с его ID в БД и создаёт папку с выведенным названием в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "**name**": **{name}**,  "path": **{path}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "id": <**id\_альбома**>,  "name": **{name}**,  "path": **{path}**  }  **Возможные статус-коды:** 201, 401, 409, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | name | string | да | макс: 255, уник(для юзера) | Название | Камера |
| 2 | path | string | нет | макс: 255, уник(для юзера) | Путь на клиенте для синхронизации | DCIM/Camera |

**Получение альбома**

Запрос для получения конкретного альбома.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:**  Свой  {  "album": {  "id": 1,  "name": "Камера",  "createdAt": "2024-12-16T10:13:23",  "grantAccesses": [  {  "id": 215,  "name": "Чел"  }, { ... }, { ... }  ],  "picturesCount": 31,  "picturesInfo": {  "sign": "2\_JDJ5JDEyJG11RGc24aWRX",  "ids": [84, 30, 69, 59]  }  }  }  Чужой доступный  {  "album": {  "id": 1,  "name": "Скаченное",  "accessLevel": <read|write|delete>  "owner": {  "id": 215,  "name": "Чел"  },  "picturesCount": 29,  "picturesInfo": {  "sign": "2\_JDJ5JDEyJHoMVpWaEJjVzNSV2",  "ids": [49, 41, 30, 87]  }  }  }  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | albumId | integer | да | существовать | Код альбома | 87 |

**Удаление альбома**

Запрос для удаления альбома. Удаляет все связанные данные в БД и все картинки в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | **Status:** 204 Deleted  **Возможные статус-коды:** 204, 401, 403, 404 |

**Получение альбома**

Запрос для получения списка альбомов. Возвращает список всех альбомов пользователя и доступных из БД.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:**  {  "own": [  {  "id": 354,  "name": "Камера",  "path": "DCIM/Camera",  "createdAt": "2024-12-16T10:13:23",  "grantAccessesCount": 3,  "picturesCount": 31,  "picturesInfo": { ... }  }, { ... }, { ... }  ],  "accessible": [  {  "id": 5372,  "name": "Скаченное",  "accessLevel": <read|write|delete>  "owner": {  "id": 215,  "name": "Чел"  },  "picturesCount": 31,  "picturesInfo": { ... }  }, { ... }, { ... }  ]  }  **Возможные статус-коды:** 200, 401 |

## Работа с альбомными доступами

**Создание кода приглашения**

Запрос для создания кода приглашения для доступа к альбому. Если лимиты не указаны, то приглашение бессрочное и вступить может любое количество человек.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/invite  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "expiresAt": **{expiresAt},**  "timeLimit": **{timeLimit}**,  "joinLimit": **{joinLimit}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "invitation": {  "code": "928Lm8e4",  "expiresAt": "2025-12-30 20:50:00",  "joinLimit": 1  }  }  **Возможные статус-коды:** 201, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | expiresAt | datetime | нет | > текущего | Дата просрочки | 2024-12-30 |
| 2 | timeLimit | integer | нет | мин: 1 | Лимит времени (минуты) | 60 |
| 3 | joinLimit | integer | нет | мин: 1 | Лимит вступлений | 1 |

**Получение альбома в приглашении**

Запрос для получения информации об альбоме по коду приглашения. Показывает 30 последних картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**/album  **Method:** GET | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "album": {  "id": 3,  "name": "Огурец",  },  "pictures": [  {...}, {...}, {...}  ]  }  **Возможные статус-коды:** 200, 404 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | code | string | да | Код приглашения | Bc7ELcIf |

**Вступление по приглашении**

Запрос для получения доступа к альбому чужого пользователя по коду приглашению.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**/join  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 404, 409 |

**Удаление кода приглашения**

Запрос для удаления кода приглашения.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**  **Method:** DELETE | **Status:** 204 No Content  **Возможные статус-коды:** 204, 403, 404 |

**Отзыв доступа для пользователя**

Запрос для удаления доступа к своему альбому другого пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{album}**/accesses/**{user}**  **Method:** DELETE | **Status:** 204 No Content  **Возможные статус-коды:** 204, 403, 404 |

**Отзыв доступа на себе**

Запрос для удаления доступа к чужому альбому от себя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{album}**/accesses  **Method:** DELETE | **Status:** 204 No Content  **Возможные статус-коды:** 204, 403, 404 |

## Работа с картинками

**Загрузка картинок в альбом**

Запрос для загрузки на ФС сервера картинок и информации об каждой в БД

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/pictures  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** multipart/form-data  **Body:** {  "pictures[]": **{pictures}**,  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "successful": [  {  "id": 1,  "name": "test.jpg",  "date": "2024-10-29 19:56:10",  "size": 1696738,  "width": 1360,  "height": 2128,  “hash”: "df59b7d94c03b59a”  }, { ... }  ],  "errored": [  {  "name": "image.png",  "message": “already exist"  }, { ... }  ]  }  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** |
| 1 | pictures | array | да | мин: 1 | Массив картинок |
| 1 | pictures.\* | file | нет | mimes: jpeg,png,gif | Картинка |

**Получение списка картинок**

Запрос для получения списка всех картинок в указанном альбоме. Так же выводится временная (на сутки) сигнатура доступа для получения превью/оригинала/загрузки.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/pictures  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "sign": <**сигнатура**>  "pictures": [  {  "id": 1,  "name": "test.jpg",  "date": "2024-10-29 19:56:10",  "size": 1696738,  "width": 1360,  "height": 2128,  “hash”: "df59b7d94c03b59a”  },  { ... }  ],  }  **Возможные статус-коды:** 200, 401, 403, 404 |

**Отображение превью картинки**

Запрос для получения файла-превью указанного размера (разрешённый размер из публичной информации о сервере). Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок. Если указать запрещённый размер, то будет перенаправление на приближённый размер.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /thumb/**{orient}{size}**  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/jpeg  **Возможные статус-коды:** 200, 301, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | pictureId | integer | да | Код картинки | 529 |
|  | orient | "w"|"h"|"q" |  | Ось указания размера (по ширине, высоте или обрезка в квадрат) |  |
| 2 | size | integer | да | Размер по оси (пиксели) | 720 |
| 3 | sign | string | да | Сигнатура доступа | 2\_5EyJHc3aBoY1Ju |

**Отображение картинки (оригинал)**

Запрос для отображения файла. Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /original  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/<"jpeg"|"png"|...>  **Возможные статус-коды:** 200, 403, 404 |

**Скачивание картинки**

Запрос для скачивания файла. Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /download  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/<"jpeg"|"png"|...>  **Content-Disposition:**  attachment; filename=<имя\_файла>  **Возможные статус-коды:** 200, 403, 404 |

**Удаление картинки**

Запрос для удаления картинки. Удаляет связанные данные в БД и картинку в ФС.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | Успех  **Status:** 204 Deleted  **Возможные статус-коды:** 204, 401, 403, 404 |

## Работа с тегами

**Создание личного тега**

Запрос для создания личного тега пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "value": **{value}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "id": <**id\_тега**>,  "name": <**название**>  }  **Возможные статус-коды:** 201, 401, 409 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | value | string | да | макс: 255, уник(для юзера) | Название | Семья |

**Получение списка личных тегов**

Запрос на получение списка личных тегов.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** [  {  "id": <**id\_тега**>,  "name": <**название**>  },  { ... },  ]  **Возможные статус-коды:** 204, 401 |

**Добавление тега к картинке**

Запрос для добавления связи тега с картинкой.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /pictures/**{pictureId}**  /tags/**{tagId}**  **Method:** POST  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | tagId | string | да | существовать | Код тега | 5 |

**Удаление тега с картинки**

Запрос для удаления связи тега с картинкой.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /pictures/**{pictureId}**  /tags/**{tagId}**  **Method:** DELETE  **Authorization:** Bearer **{token}** | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

## Работа с пользователями

**Получение себя**

Запрос на данных о себе.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/me  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 1,  "name": "Testello",  "login": "test",  <"role": "admin">  }  }  **Возможные статус-коды:** 200, 401 |

**Редактирование себя**

Запрос на изменение данных о себе.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/me  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "name": **{nickname}**,  "login": **{login}**,  "password": **{password}**  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 1,  "name": "Testello",  "login": "test",  <"role": "admin">  }  }  **Возможные статус-коды:** 200, 401, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | nickname | string | нет | мин: 2, макс: 255 | Имя юзера | Testello2 |
| 2 | login | string | нет | мин: 2, макс: 64, уник,  regex: ^[a-zA-Z0-9\_-]+$ | Логин | test2 |
| 3 | password | string | нет | мин: 8, макс: 255 | Пароль | NewPass! |

**Просмотр списка пользователей**

Запрос на просмотр списка пользователей.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "users": [  {  "id": 2,  "name": "Чел",  "isBanned": false,  "login": "test1",  <"role": "admin">  "complaintsAboutCount": 21,  "complaintsAboutAcceptedCount": 0,  "complaintsFromCount": 18,  "complaintsFromAcceptedCount": 9,  "albumsCount": 1,  "albumsViaAccessCount": 2,  "picturesCount": 31  }, { ... }, { ... }  ]  }  **Возможные статус-коды:** 200, 401, 403 |

**Просмотр пользователя**

Запрос просмотра пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/**{userId}**  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 2,  "name": "Чел",  "isBanned": false,  "login": "test1",  <"role": "admin">  "complaintsAbout": [  {  "id": 1,  "status": null,  "description": "Плохие картинки"  }, { ... }, { ... }  ],  "complaintsFrom": [  {  "id": 1,  "status": null,  "description": "Мне не нравится"  }, { ... }, { ... }  ],  "albumsCount": 1,  "albumsViaAccessCount": 2,  "picturesCount": 31,  "warnings": [  {  "id": 1,  "comment": "Больше так не делай"  }, { ... }, { ... }  ],  }  }  **Возможные статус-коды:** 200, 401, 403, 404 |

**Редактирование пользователя**

Запрос редактирование пользователя.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/users/**{userId}**  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "name": **{nickname}**,  "login": **{login}**,  "password": **{password},**  "role\_id": **{roleId}**,  "is\_banned": **{**isBanned**},**  } | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "user": {  "id": 2,  "name": "Чел",  "isBanned": false,  "login": "test1",  <"role": "admin">  }  }  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | roleId | integer | нет | существовать | ИД роль | 2 |
| 2 | isBanned | boolean | нет |  | Статус блокировки | true |

## Работа с настройками

**Просмотр публичных настроек**

Запрос публичных настроек.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/  **Method:** GET | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "settings": {  "allowed\_upload\_mimes": [  "jpeg", "jpg", "png", "gif"  ],  "allowed\_preview\_sizes": [  144, 240, 360, 480, 720, 1080  ],  "warning\_limit\_for\_ban": 3,  "free\_storage\_limit": 5368709120,  "complaint\_types": [  {  "id": 5,  "name": "Нарушение авторских прав"  },  {  "id": 6,  "name": "Шок-контент"  }  ],  "is\_upload\_disabled": true  }  }  **Возможные статус-коды:** 200 |

# Проектирование и разработка базы данных

## Концептуальная модель данных

Концептуальная модель — это формализованное описание предметной области, не связанное с какими-либо компьютерными средствами. Сущности (таблицы) обозначаются прямоугольниками; Атрибуты (поля таблиц) — овалы, которые могут быть объедены в скруглённые прямоугольники; Связи с обозначением мощности — стрелками (в направлении, где мощность равна многим — двойная стрелка, а со стороны, где она равна единице — одинарная).

Анализируя предметную область, были выделены следующие сущности с их атрибутами (см. рисунок 2):

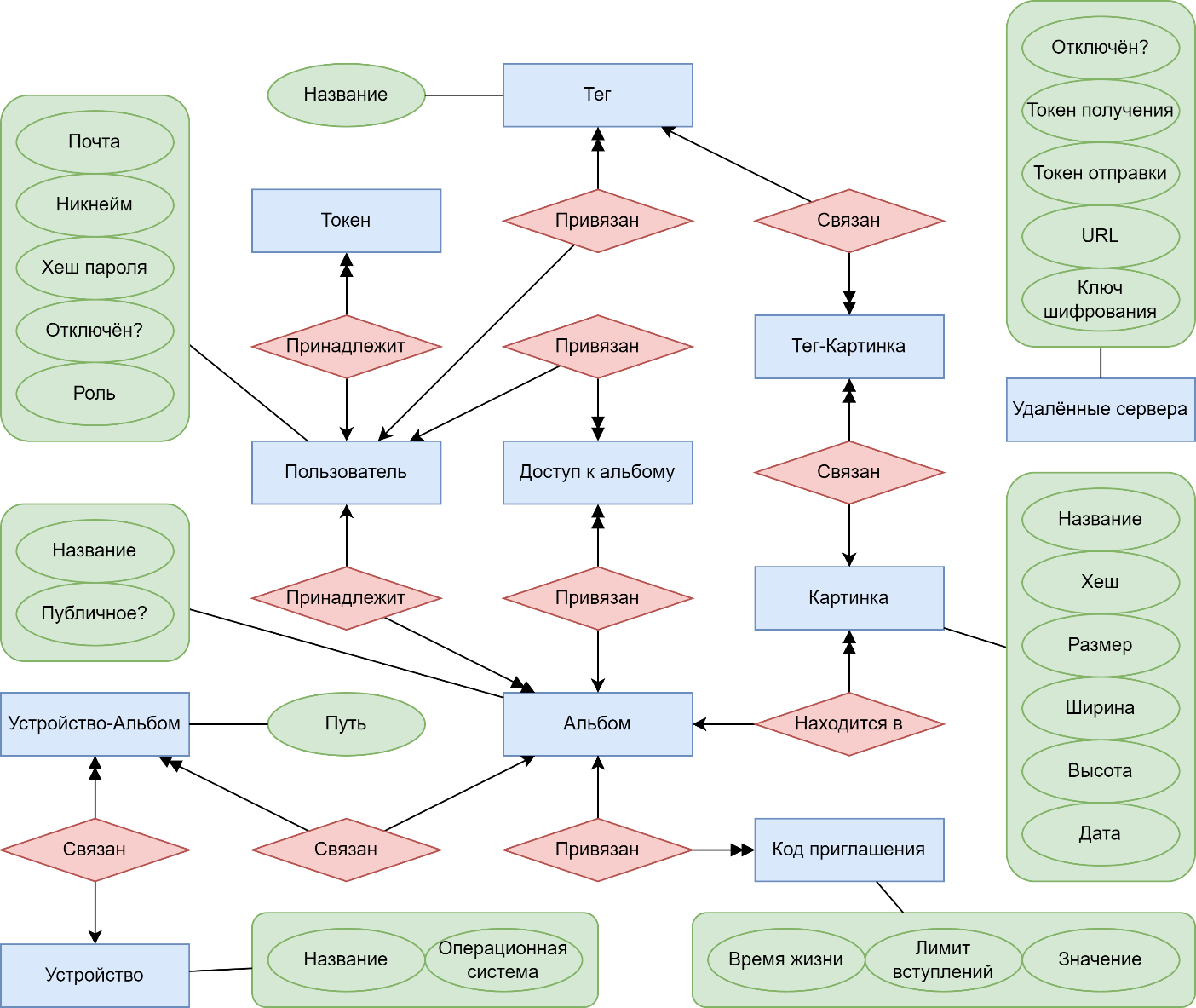


Рисунок 2. Концептуальная модель данных

## Логическая модель данных

Логическая модель базы данных — схема базы данных, выраженная в понятиях модели данных. Этим отличается от концептуальной модели, описывающей семантику предметной области без указания технологии (конкретных методов реализации), и от физической модели, которая описывает конкретные физические механизмы, применяемые для хранения данных в накопителях.

Используя концептуальную модель данных и выполнив нормализацию данных до 3 нормальной формы, была разработана логическая модель данных (см. рисунок 3):

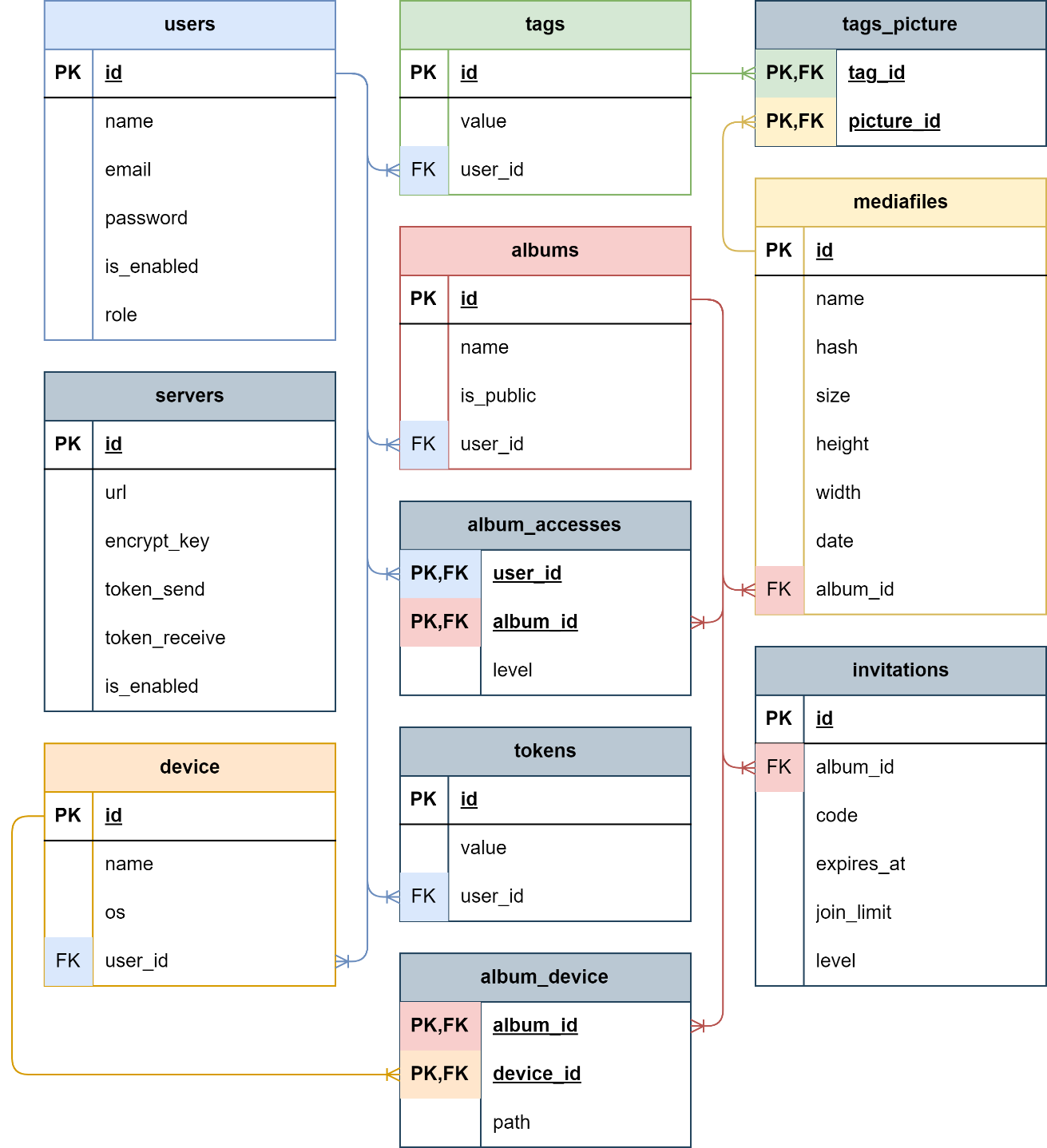


Рисунок 3. Логическая модель данных

## Физическая модель данных

Физическая модель данных — это модель данных, описанная с помощью средств конкретной системы управления базами данных (СУБД). Физическая модель данных строится на базе логической путем добавления особенностей конкретной СУБД. К таким особенностям могут относиться поддерживаемые СУБД типы данных, соглашения о присвоении имен таблицам, атрибутам и т.д. Физическая модель данных фактически является готовым заданием на создание БД, имея которое можно реализовать БД в выбранной СУБД.

Выбранной СУБД является MySQL. Используя логическую модель, была разработана физическая модель данных (см. рисунок 4):

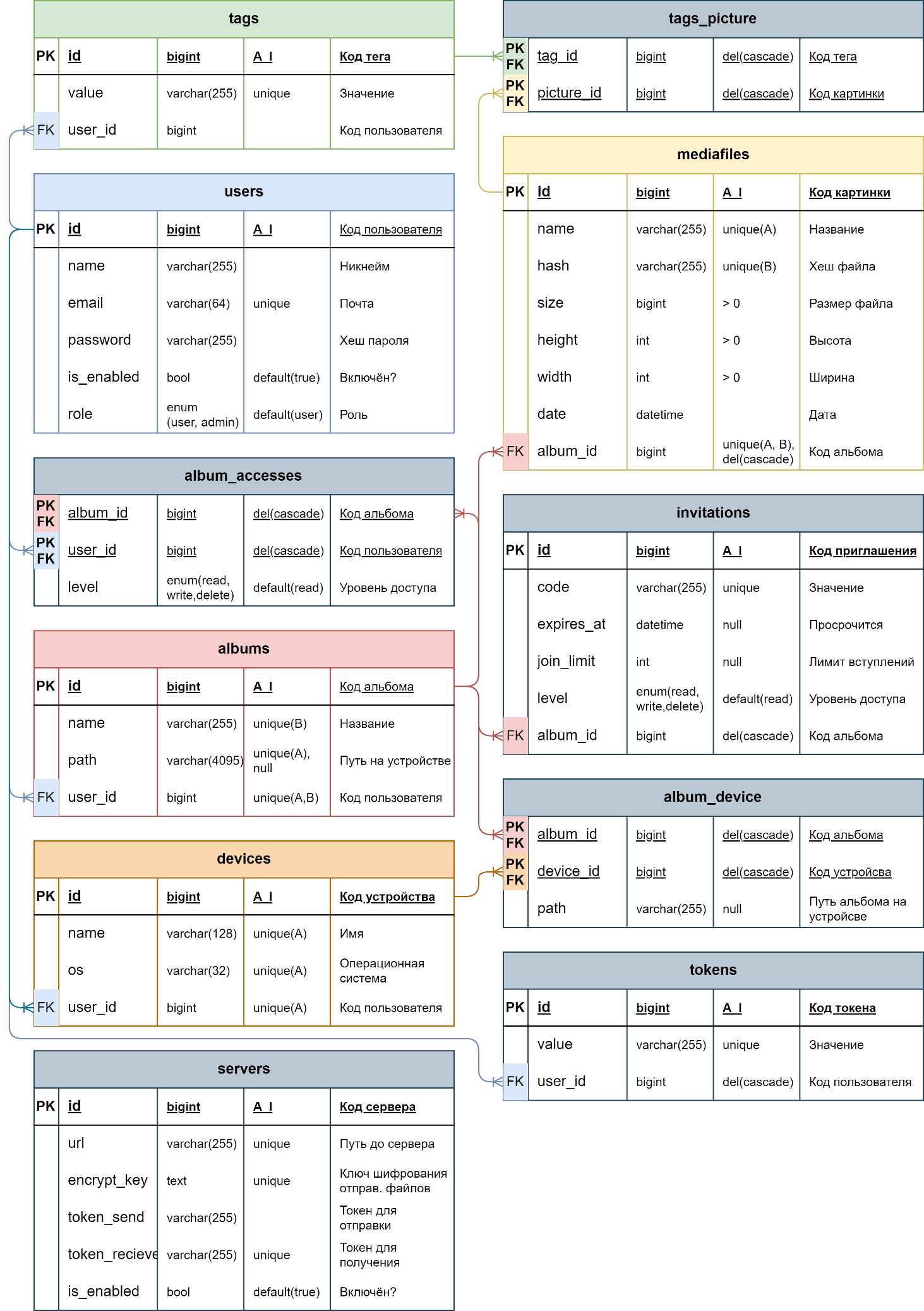


Рисунок 4. Физическая модель данных