**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ  
СЕРВИСА СИНХРОНИЗАЦИИ КАРТИНОК «PICSYNC»**

***ОТЧЁТ ПО КУРСОВОЙ РАБОТУ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнили:

Студенты группы 1912с

\_\_\_\_\_\_\_\_\_\_\_ / Заикин В.А

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

\_\_\_\_\_\_\_\_\_\_\_ / 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 | Сервис синхронизации фотографий «PicSync» |
| Цель проекта | Спроектировать и разработать сервис синхронизации картинок «PicSync» |
| Задачи проекта | 1. Разработать базу данных сервиса на MySQL. 2. Спроектировать структуру API сервиса. 3. Разработать сервер API сервиса на фреймворке Laravel. 4. Разработать мобильное приложение на MAUI 5. Разработать автоматизированные тесты и провести тестирование. |

## Назначение документа

Настоящий документ рассчитан на использование в проектной команде в качестве ключевого документа, определяющего требование по реализации задачи по разработке сервиса синхронизации картинок «PicSync».

## 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. |
| Laravel | Laravel – это бесплатный, открытый фреймворк для создания веб-приложений на языке программирования PHP. Он предоставляет структуру и базовые инструменты для быстрого создания веб-приложений, используя принципы модели-представления-контроллер (MVC) и другие современные подходы к разработке. |

# Общее описание

Разрабатываемый сервис предназначен для удобного хранения, управления и получения пользователями своих фотографий и картинок с удаленного сервера. Основная цель сервиса – освободить память на мобильных устройствах пользователей, предлагая им безопасное и надежное облачное хранилище для их медиафайлов. Пользователи смогут загружать фотографии и картинки с телефона на сервер 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 посредством приглашения через ссылки. Важной особенностью сервиса будет возможность мгновенного получения фотографий и картинок с сервера на телефон по запросу пользователя, что позволит им всегда иметь доступ к своим медиафайлам, не зависимо от места нахождения.

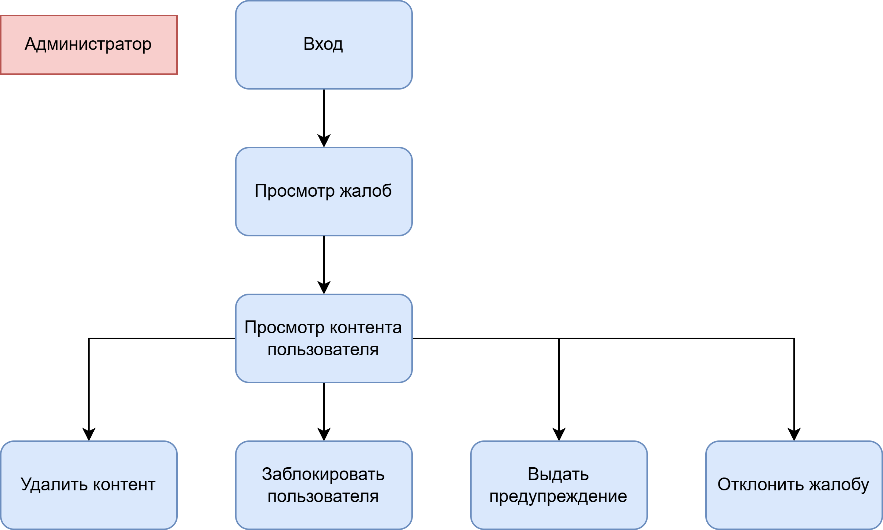


Рисунок 1. Функционал администратора

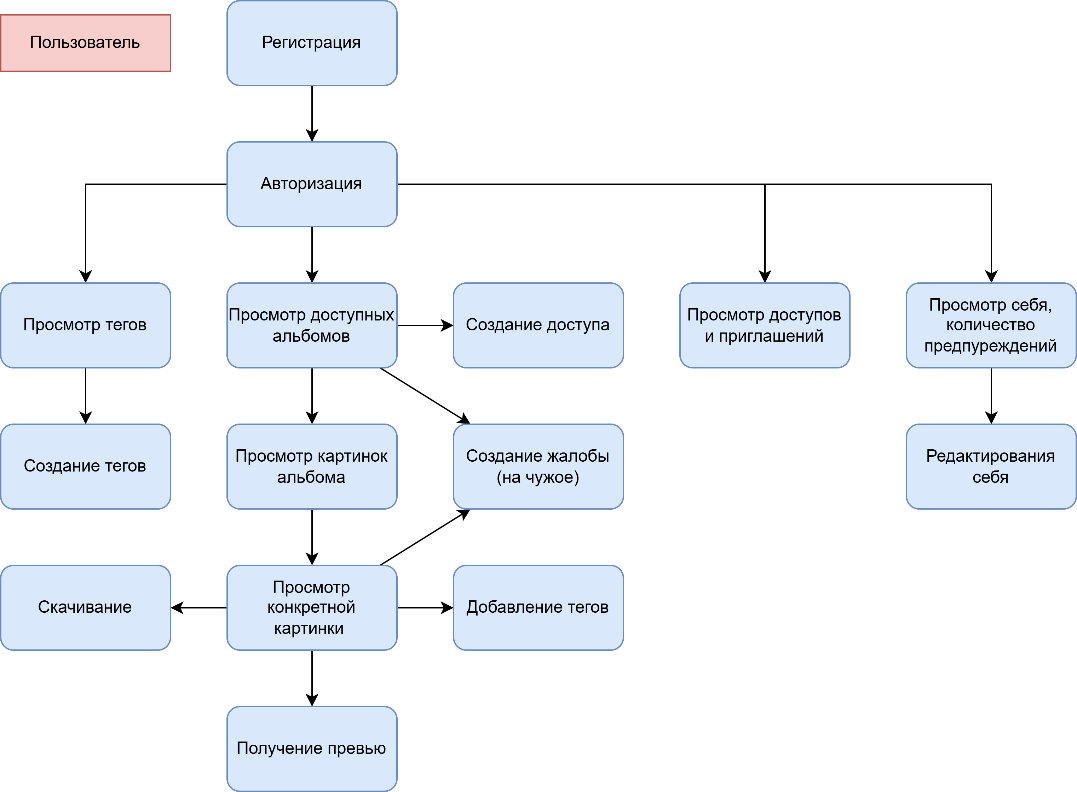


Рисунок . Функционал пользователя

# Функционал сервера

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

Пользователь может создать альбом с указанием по какому пути он хранится на устройстве. В альбом может загрузить картинки.

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

Пользователь может получить доступ к чужому альбому путём перехода по ссылке на доступ от автора альбома или если автор альбома самолично введёт логин пользователя.

Пользователь может пожаловаться на альбом или картинку доступного чужого альбома. После некоторого количество жалоб (указанных в настройках) пользователь автоматически становится заблокированным (статус “автобан”).

Администратор может посмотреть все жалобы и всех автоматически заблокированных пользователей. С жалобами и автоматическими заблокированными пользователями администратор может: отклонить, удалить материал и выдать предупреждение или заблокировать пользователя.

Администратор может всё с данными пользователя, если на пользователя поступила жалоба.

## Ролевая модель

| **Функционал** | | **Админ** | **Клиент** |
| --- | --- | --- | --- |
| Пользователи | CRUD (Read, Update) | да | да (себя) |
| Авторизация | да |
| Регистрация |
| Выход с одного устройства |
| Выход со всех устройств |
| Альбомы | CRUD | да (с жалобой) | да (свои) |
| Приглашения на доступ | Создание | нет | да (на свои альбомы) |
| Удаление |
| Картинки | CRUD | да (с жалобой) | да (свои) |
| Теги | CRUD | да (с жалобой) | да (свои) |
| Добавление тега к картинке | нет | да (к своим) |
| Удаление тега с картинки |
| Жалобы | Изменение | да | нет |
| Просмотр | да (свои отправленные) |
| Создание |
| Удаление |
| Доступы на альбомы | Создание | да (с жалобой) | да (на свои альбомы; к себе) |
| Удаление |

# Хранимые данные

1. **Токены:** значение и связь с пользователем
2. **Пользователи:** никнейм, логин, хеш пароля, количество жалоб и состояние бана
3. **Роли:** значение
4. **Альбомы:** название, связанный пользователь и путь на клиентском устройстве
5. **Доступы к альбомам:** пользователь и альбом
6. **Приглашения доступа:** связанный альбом, ссылка и время жизни
7. **Картинки:** связанныйальбом, название, размер файла, ширина, высота, дата, хеш и связанные теги
8. **Теги:** название, связанный пользователь
9. **Жалобы:** пояснение, тип, пользователя-отправителя и альбомом/картинка

# 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 | Логин | test |
| 4 | password | string | да | мин: 8, макс: 255, должна быть 1 большая и 1 маленькая буква, 1 цифра и 1 спецсимвол | 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"**|"**user"**>  <**данные\_пользователя**>  }  }  **Возможные статус-коды:** 200, 401, 422 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Описание** | **Пример** |
| 1 | login | string | да | Логин | test |
| 2 | password | string | да | Пароль | Test123! |

**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": **{название}**,  "path": **{путь\_на\_клиенте}**  }  **Возможные статус-коды:** 201, 401, 403, 422 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | name | string | да | макс: 255, уник(для юзера) | Название | DCIM |
| 2 | path | string | да | макс: 255, уник(для юзера) | Путь на клиенте для синхронизации | /emulated/0/DCIM |

**Получение списка альбомов**

Запрос для получения списка альбомов. Возвращает список всех альбомов пользователя в БД.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** [  {  "id": <**id\_альбома**>,  "name": <**название**>,  "path": <**путь\_на\_клиенте**>  },  { ... },  { ... }  ]  **Возможные статус-коды:** 200, 401, 403 |

**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 |

Входные параметры:

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

## Работа с альбомными доступами

**Создание кода приглашения**

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

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**/invite  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "timeLimit": **{timeLimit}**,  "joinLimit": **{joinLimit}**  } | **Status:** 201 Created  **Content-Type:** application/json  **Body:** {  "invitationCode": <**код\_приглашения**>  }  **Возможные статус-коды:** 201, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | timeLimit | integer | нет | мин: 1 | Лимит времени (минуты) | 60 |
| 2 | joinLimit | integer | нет | мин: 1 | Лимит вступлений | 1 |

**Получение альбома в приглашении**

Запрос для получения информации об альбоме по коду приглашения.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/invitation/**{code}**/album  **Method:** GET | **Status:** 200 OK  **Content-Type:** application/json  **Body:** {  "album": {  "id": <**id\_альбома**>,  "name": <**название**>,  "user": {  <**инфо\_об\_пользователе**>  },  "pictures" [  <**содержашиеся\_картинки**>  ]  }  }  **Возможные статус-коды:** 200, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **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 |

## 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 получению 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. Для аутентификация используется сигнатура доступа, которую можно получить из запроса списка картинок.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/**{albumId}**  /pictures/**{pictureId}**  /thumb/**{size}**  ?sign=**{sign}**  **Method:** GET | **Status:** 200 OK  **Content-Type:** image/jpeg  **Возможные статус-коды:** 200, 401, 403, 404 |

Входные параметры:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** | **Пример** |
| 1 | pictureId | integer | да |  | Код картинки | 529 |
| 2 | size | integer | да | мин: 120 | Размер квадрата (пикс) | 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, 401, 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, 401, 404 |

**Удаление картинки**

Запрос для удаления альбома. 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:** 200, 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}** | Успех  **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 жалобами

**Создание жалобы на альбом**

Запрос для создания жалобы на альбом.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /complaint  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "description": **{пояснение}**,  "type\_id": **{id\_типа}**  } | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

**Создание жалобы на картинку**

Запрос для добавления жалобы на картинку.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/albums/{**albumId**}  /pictures/**{pictureId}**  /complaint  **Method:** POST  **Authorization:** Bearer **{token}**  **Content-Type:** application/json  **Body:** {  "description": **{description}**,  "type\_id": **{typeId}**  } | **Status:** 204 No Content  **Возможные статус-коды:** 204, 401, 403, 404, 409 |

Входные параметры:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Название** | **Тип** | **Обяз** | **Условия** | **Описание** |
| 1 | description | string | нет |  | Пояснение |
| 1 | typeId | integer | да | существовать | Тип жалобы |

**Получение списка жалоб**

Запрос для получения списка жалоб.

|  |  |
| --- | --- |
| **Request** | **Response** |
| **URL: {api}**/complaints  **Method:** GET  **Authorization:** Bearer **{token}** | **Status:** 200 OK  **Content-Type:** application/json  **Body:** [  {  "id": <**id\_жалобы**>,  "description": <**пояснение**>,  "from\_user": <**данные\_жалующегося**>,  "about\_user": <**данные\_на\_кого\_жалоба**>,  <"album"|"picture">: {  <**данные\_предмета\_жалобы**>  }  },  { ... },  ]  **Возможные статус-коды:** 204, 401, 403 |

# Требования к мобильному приложению

При входе неавторизированного пользователя выводить страницу с переключением регистрации/авторизации

При входе авторизированного пользователю доступны страницы:

* Главная — недавние фотографии и картинки
* Альбомы — список всех доступных альбомов-папок
* Настройки — настройки профиля, управление доступами

На главной авторизированного пользователя выводить сетку недавних картинки устройства со альбомов, которые установлены как синхронизирующийся, и свои недавние, полученные с сервера. На каждой картинке выводить статус синхронизации с сервером.

На странице альбомов показывать карусель доступных альбомов других пользователей, список альбомов, которые есть на сервере, список синхронизирующийся альбомов и список локальных не синхронизирующийся альбомов. При нажатии на альбом открывать сетку картинок.

На странице настроек отображать информацию о себе с возможностью редактирования, список альбомов, которые открыты пользователям и возможность выйти со всех устройств.

# 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, можно выделить следующие сущности с их атрибутами (см. рисунок 3):

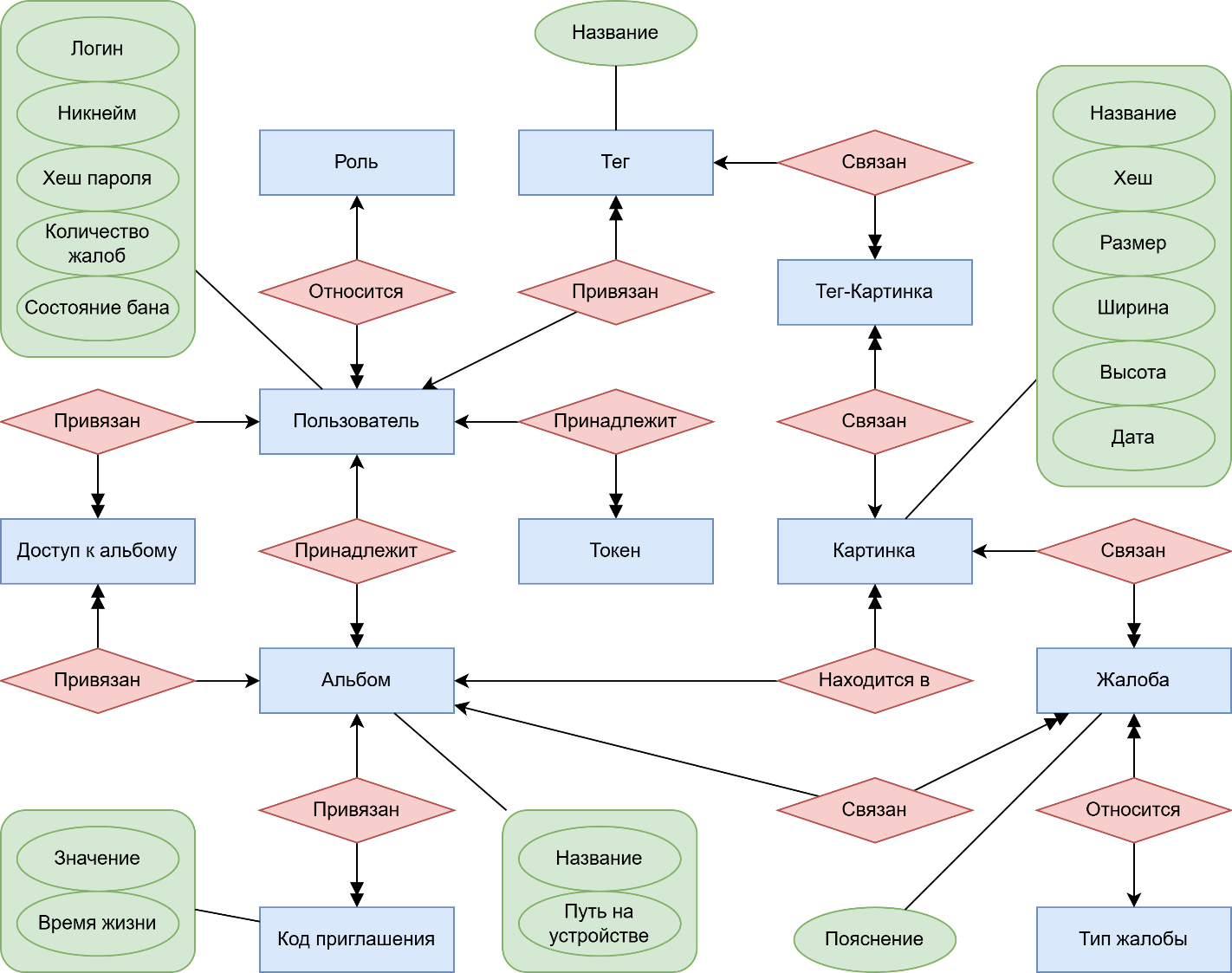


Рисунок . Концептуальная модель данных

## 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 нашу концептуальную модель данных и выполнив нормализацию данных до 3 нормальной формы включительно, построим логическую модель данных (см. рисунок 4):

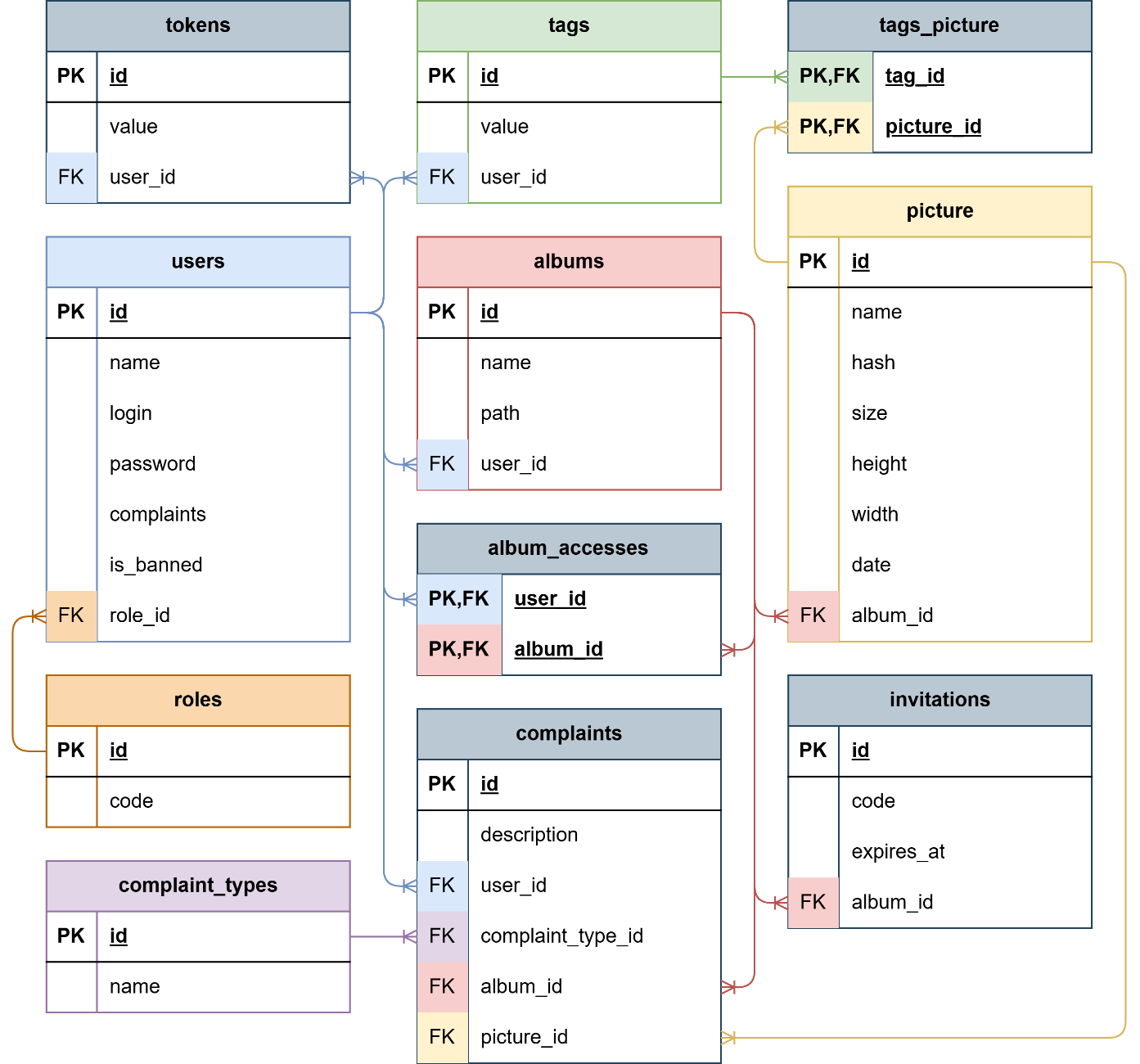


Рисунок . Логическая модель данных

## 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, построим физическую модель данных (см. рисунок 5):

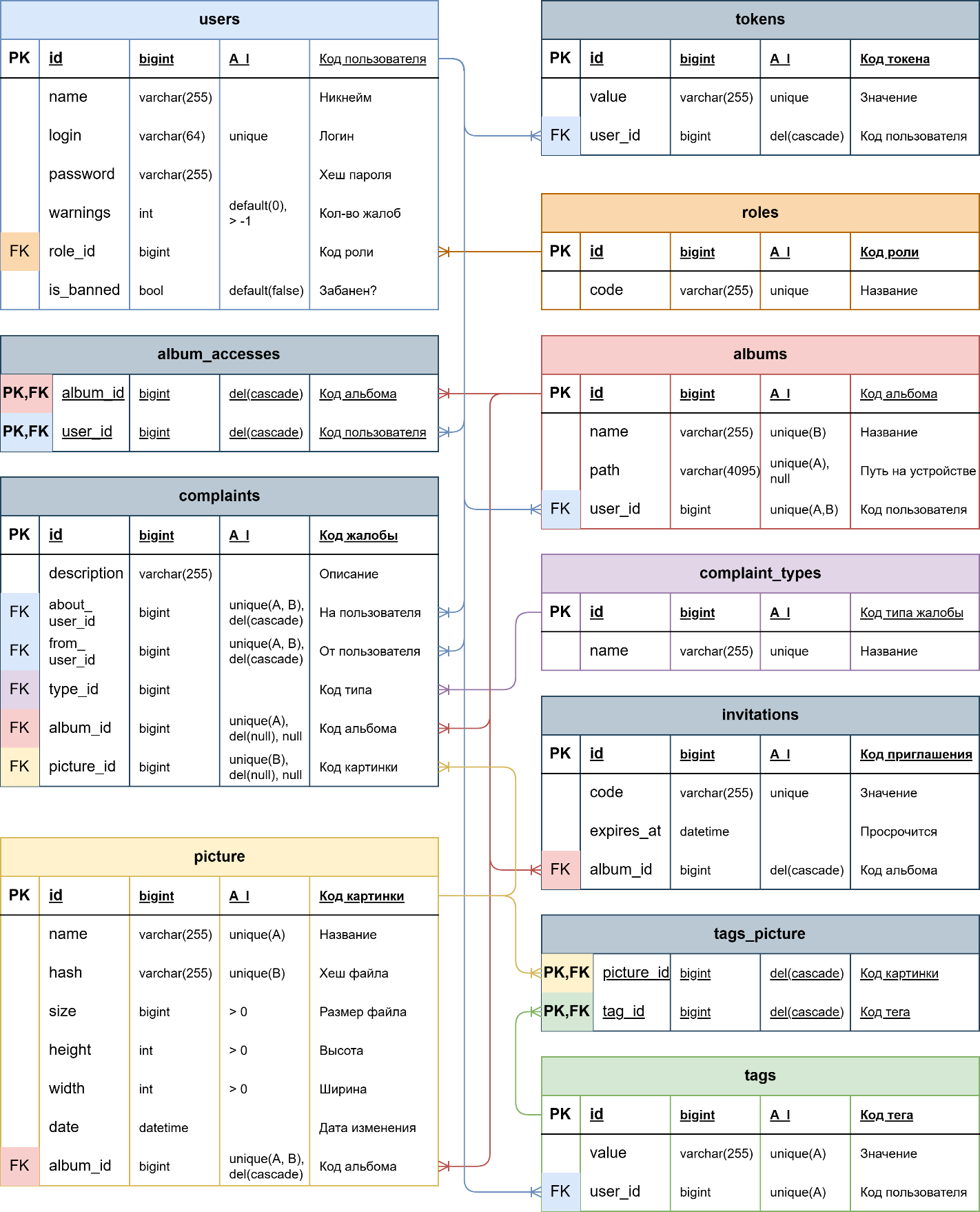


Рисунок . Физическая модель данных