

# API

## 1. registerUser

Описание: регистрация нового пользователя.

Входные данные:

- `username` (string) - логин пользователя.
- `user_email` (string) - email пользователя.
- `password` (string) - пароль пользователя.

Выходные данные:

- `user_id` (int | none) - уникальный идентификатор пользователя.
- `session_id` (int | none) - уникальный идентификатор сессии.
- `message` (string) - сообщение об успешной регистрации или ошибке.

## 2. logIn

Описание: авторизация пользователя и создание новой сессии.

Входные данные:

- `username` (string) - логин пользователя.
- `password` (string) - пароль пользователя.
- `device_id` (int) - идентификатор устройства, с которого выполняется вход.

Выходные данные:

- `session_id` (int | none) - уникальный идентификатор сессии.
- `status` (string) - статус сессии операции.

## 3. logOut

Описание: завершение текущей сессии пользователя на указанном устройстве.

Входные данные:

- `session_id` (int) - идентификатор сессии.

Выходные данные:

- `status` (string) - статус завершения сессии ("успешно завершена" или ошибка).

## 4. updatePassword

Описание: смена пароля пользователя.

Входные данные:

- `user_id` (int) - идентификатор пользователя.
- `old_password` (string) - текущий пароль.
- `new_password` (string) - новый пароль.

Выходные данные:

- `status` (string) - результат обновления ("успешно" или ошибка).

## 5. removeDeviceFromNetwork

Описание: удаление устройства из сети пользователя.

- **Входные данные:**

- `user_id` (int) - идентификатор пользователя.
- `session_id` (int) - идентификатор сессии для удаления.
- **Выходные данные:**
  - `status` (string) - статус выполнения ("устройство удалено" или ошибка).

#### 6. `uploadImages`

Описание: индексирование новой локальной папки с изображениями и загрузка векторных представлений изображений.

Входные данные:

- `session_id` (int) - идентификатор устройства.
- `image_embeddings` (array) - векторные представления изображений.

Выходные данные:

- `status` (string) - статус загрузки ("успешно проиндексировано" или ошибка).

#### 7. `searchImagesByText`

Описание: поиск изображений по текстовому запросу.

Входные данные:

- `session_id` (int) - идентификатор сессии.
- `devices_ids` (list of int | none) - список id устройств, на которых нужно осуществить поиск, если none, то на всех возможных.
- `search_query` (string) - текстовый запрос.

Выходные данные:

- `results` (list of objects) - список объектов, содержащих идентификаторы изображений и соответствующие данные.

#### 8. `searchImagesByFace`

Описание: поиск изображений по выбранному лицу из предложенных вариантов.

Входные данные:

- `session_id` (int) - идентификатор сессии.
- `devices_ids` (list of int | none) - список id устройств, на которых нужно осуществить поиск, если none, то на всех возможных.
- `faces` (list of base64 objects) - список из бинарных объектов - выбранных лиц для поиска.

Выходные данные:

- `results` (list of objects) - список изображений, где присутствует выбранное лицо.

#### 9. `getDevicesStatus`

Описание: получение статуса устройств (в сети или не в сети).

Входные данные:

- `session_id` (int) - идентификатор сессии.

Выходные данные:

Список устройств с соответствующими статусами:

- `device_name` (string) - название устройства.
- `status` (string) - "в сети" или "не в сети".

#### 10. `getImageDetails`

Описание: получение детальной информации о выбранной фотографии и возможность её сохранения на локальное устройство.

Входные данные:

- `session_id` (int) - идентификатор сессии.
- `image_id` (int) - уникальный идентификатор изображения.
- `download` (boolean) - флаг для скачивания изображения (если оно на другом устройстве).

Выходные данные:

- `image_details` (object) - объект с детальной информацией об изображении:
  - `image_path` (string) - путь к изображению на локальном устройстве, если изображение доступно.
  - `metadata` (dictionary) - метаданные изображения, такие как дата создания, размер, и др.
- `status` (string) - статус операции ("детали получены" или "скачивание завершено" либо ошибка).

#### 11. `sendTechQuery`

Описание: создание нового вопроса тех. поддержки.

Входные данные:

- `session_id` (int) - идентификатор пользователя
- `topic` (str) - тема обращения
- `message` (str) - обращение

Выходные данные:

- `status` (str) - статус обращения

## ИСП

### 1. Подготовка и проектирование

#### 1.1. Разработка технического задания

##### 1.1.1. Основания для разработки (цели и задачи проекта)

##### 1.1.1. Проектирование функциональных требований к проекту

#### 1.2. Проектирование интерфейса пользователя

##### 1.2.1. Окна входа и регистрации пользователей

##### 1.2.2. Главное окно приложения

##### 1.2.3. Окно данных о пользователе

##### 1.2.4. Элементы отображения результатов поиска

##### 1.2.5. Окно деталей об изображении

- 1.2.6. Окно локальных отслеживаемых папок
- 1.2.7. Окно выбора лиц для поиска
- 1.2.8. Окно истории поиска
- 1.3. Проектирование серверной архитектуры
  - 1.3.1. Определение структуры баз данных (векторной и реляционной)
  - 1.3.2. Проектирование архитектуры API (описание функции)
- 2. Разработка серверного приложения
  - 2.1. Создание и настройка векторной базы данных
    - 2.1.1. Выбор конкретной технологии
    - 2.1.2. Создание базы данных на основе ранее спроектированной архитектуры
  - 2.2. Создание и настройка реляционной базы данных
    - 2.2.1. Выбор конкретной СУБД
    - 2.2.2. Создание базы данных на основе ранее спроектированной архитектуры
  - 2.3. Реализация API функции:
    - 2.3.1. `registerUser` — регистрация пользователя.
    - 2.3.2. `logIn` — авторизация и создание сессии.
    - 2.3.3. `logOut` — завершение сессии.
    - 2.3.4. `updatePassword` — смена пароля.
    - 2.3.5. `removeDeviceFromNetwork` — удаление устройства из сети.
    - 2.3.6. `uploadImages` — индексирование изображений.
    - 2.3.7. `searchImagesByText` — поиск по текстовому запросу.
    - 2.3.8. `searchImagesByFace` — поиск по лицу.
    - 2.3.9. `getDevicesStatus` — проверка статуса устройств.
    - 2.3.10. `getImageDetails` — получение информации о фотографии.
  - 2.4. Тестирование серверного решения
    - 2.4.1. Тестирование баз данных
    - 2.4.2. Тестирование API функций
- 3. Настройка передачи данных пользователей p2p (peer-to-peer)
  - 3.1. Проектирование архитектуры передачи данных
    - 3.1.1. Определение протокола передачи данных (WebRTC с использованием ICE, STUN и TURN).
    - 3.1.2. Проектирование схемы взаимодействия устройств через центральный сервер для обмена ICE-кандидатами.
    - 3.1.3. Выбор и настройка механизмов шифрования (DTLS/SRTP).
    - 3.1.4. Проектирование обработки ошибок при установке соединения (fallback на TURN сервер).
  - 3.2. Настройка центрального сервера
    - 3.2.1. Поднятие STUN/TURN сервера (с использованием Coturn).
    - 3.2.2. Настройка API для обмена ICE-кандидатами:
      - 3.2.2.1. Эндпоинт для авторизации устройств.

- 3.2.2.2. Эндпоинт для обмена IP-адресами, портами и типами NAT.
- 3.2.3. Конфигурация HTTPS для защиты данных.
- 3.2.4. Мониторинг и логирование соединений между устройствами.
- 3.3. Реализация P2P передачи данных
  - 3.3.1. Реализация Data Channel для передачи файлов между устройствами.
  - 3.3.2. Разработка механизма передачи данных:
    - 3.3.2.1. Разбиение файлов на чанки.
    - 3.3.2.2. Контроль целостности данных (проверка контрольных сумм).
    - 3.3.2.3. Ретрансляция пропущенных данных.
  - 3.3.3. Интеграция с центральным сервером для инициализации соединения.
- 3.4. Тестирование p2p передачи данных между устройствами
- 4. Разработка клиентского приложения
  - 4.1. Реализация клиентской логики
    - 4.1.1. Верстка окон по ранее спроектированному дизайну
    - 4.1.2. Реализация взаимодействия с API сервера
    - 4.1.2. Обработка пользовательских данных и синхронизация локальных папок с данными на удаленном сервере
  - 4.2. Тестирование клиентского приложения
  - 4.2. Тестирование поведения графических элементов окон
  - 4.3. Тестирование взаимодействия клиентского приложения с серверным решением
- 5. Разработка административной панели
  - 5.1. Разработка модуля статистики
  - 5.2. Разработка модуля технической поддержки
  - 5.3. Разработка функционала управления пользователями и устройствами.
- 6. Конечное тестирование и завершение проекта
  - 5.1. Разработка тестовых сценариев, модульное и интеграционное тестирование
  - 5.2. Приемка и сдача готового продукта

## Оценка времени выполнения проекта по методу PERT

Формула для метода PERT:

$$E_i = \frac{P_i + 4M_i + O_i}{6}$$

где  $M$  - наиболее вероятная оценка трудозатрат,  $O$  - минимально возможные затраты на реализацию пакета работ,  $P$  - пессимистическая оценка трудозатрат, все риски реализовались.

Среднеквадратическое отклонение:

$$CKO_i = \frac{P_i - O_i}{6}$$

Количественно оценим состав работ:

- кол-во сущностей: 9 сущностей
- API методов: 11 методов
- кол-во форм графического интерфейса: 11 форм
- STUN/TURN сервер: 1 сущность

## Серверная часть

Элемент	<i>O</i> (часов)	<i>M</i> (часов)	<i>P</i> (часов)	<i>E</i>	СКО
API	3	10	30	12.7	5.5
Схема БД	1	3	20	5.5	3.5
STUN/TURN сервер	10	50	100	51.67	18.3

## Клиентское приложение

Элемент	<i>O</i> (часов)	<i>M</i> (часов)	<i>P</i> (часов)	<i>E</i>	СКО
Верстка форм	2	5	20	7	3.67
Взаимодействие с API	1	3	10	3.84	1.84
Логика р2р передачи	5	15	50	19.17	9.17

## Административная панель

Элемент	<i>O</i> (часов)	<i>M</i> (часов)	<i>P</i> (часов)	<i>E</i>	СКО
Модель статистики	10	40	50	36.67	10
Модуль техподдержки	3	15	50	18.84	8.84
Управление пользователями	1	5	10	5.17	1.84

Рассчитаем среднюю трудоёмкость работ в проекте:

$$E = 12.7 * 11 + 5.5 * 9 + 51.67 + 7 * 11 + 3.84 * 11 + 19.7 + 36.67 + 18.84 + 5.17 = 440.49 \text{ ч.}$$

$$\text{СКО} = \sqrt{5.5^2 * 11 + 3.5^2 * 9 + 18.3^2 + 3.67^2 * 11 + 1.84^2 * 11 + 9.17^2 + 10^2 + 8.84^2 + 1.84^2} = 35.1 \text{ ч.}$$

Теперь, для оценки суммарной трудоемкости проекта, которая не будет превышена с вероятностью 0.95, рассчитаем следующим образом:

$$E_{95\%} = E + 2 * СКО = 440.49 + 2 * 35.1 = 510.69 \text{ чел} * \text{ час}$$

Учитывая, что в месяц сотрудник будет тратить, примерно,  $165 * 0.8 = 132 \text{ чел} * \text{ час} / \text{ мес}$ , то на выполнение данного проекта в месяцах уйдет  $479.85 / 132 = 3.87 \text{ месяца}$ .