Auth API /api/v1 Получаем логин и пароль Валидируем данные POST /signup Создаем пользователя Доступно при наличии валидного access токена и прав в нём Сохраняем историю Возвращаем список ролей Создаем и возвращаем access и refresh токены GET / "id": UUID, Получаем логин и пароль "name": str, "description": str Валидируем данные } POST /login Обновляем refresh токен в Redis Создаем новую роль Сохраняем историю Получаем: Возвращаем access и refresh токены refresh-токен хранится в Redis "name": str, "description": str Получаем access токен POST / Валидируем токен Проверяем что роли с таким name **DELETE /logout** нет Сохраняем историю Возвращаем і созданной роли -Отзываем access и refresh токен, сохраняем в Redis на время их /user Удаляем роль действия DELETE /{role_id} Возвращаем: Получаем refresh токен 202 - удалили 404 - нет такой роли Валидируем токен POST /refresh Обновляем роль Отзываем старые токены, сохраняя их в Redis Получаем минимум одно из полей: Отправляем новые access и refresh "name": str /roles токены "description": str Получаем access токен и новые PATCH /{role_id} данные Проверяем валидность role id Валидируем токен PATCH /update Проверяем валидность нового названия · Валидируем данные (login, password) Обновляем Обновляем данные в БД Проверяем наличие роли у - Получаем **access** токен пользователя Валидируем токен **GET /history** Получаем: Возвращаем список **N** последних действий из истории "user_id": UUID "role_id": UUID GET /

Возвращаем:

200 - роль у пользователя есть — **404** - роли у пользователя нет

Получаем:

"user_id": UUID "role_id": UUID

Сохраняем в БД -

"user_id": UUID "role_id": UUID

Удаляем из БД -

/assign

POST /

DELETE /

Добавляем пользователю роль

В случае успеха возвращаем 200 -

Получаем:

Удаялем роль у пользователя

В случае успеха возвращаем 200 и 404 если такой записи не было