**Документация с описанием и примерами запуска.**

**Запуск и настройка приложения:**

Шаг 1.

Скачать репозиторий по ссылке: <https://github.com/Gipnotyin/Vk_friends>

Шаг 2.

Запустить приложение и создать виртуальное окружение с помощью команды:

python -m venv venv

Шаг 3.

Запускаем виртуальное окружение с помощью команды:

venv\Scripts\activate.bat

Шаг 4.

Устанавливаем все нужные нам зависимости с помощью команды:

python -m pip install -r requirements.txt

Шаг 5.

Создаём миграции: python manage.py makemigrations

После чего: python manage.py migrate

Шаг 6.

Запускаем наш проект: python manage.py runserver

(Да да, кто пишет документацию не Readme, а в ворд документе..)

**Или же можно использовать Dockerfile!!!**

Тогда, сначала переходим в корневую папку проекта и выполняем команду:

docker-compose build

После чего выполняем команду:

docker-compose up -d

И переходим к примеру использования.

**Пример использования сервиса**

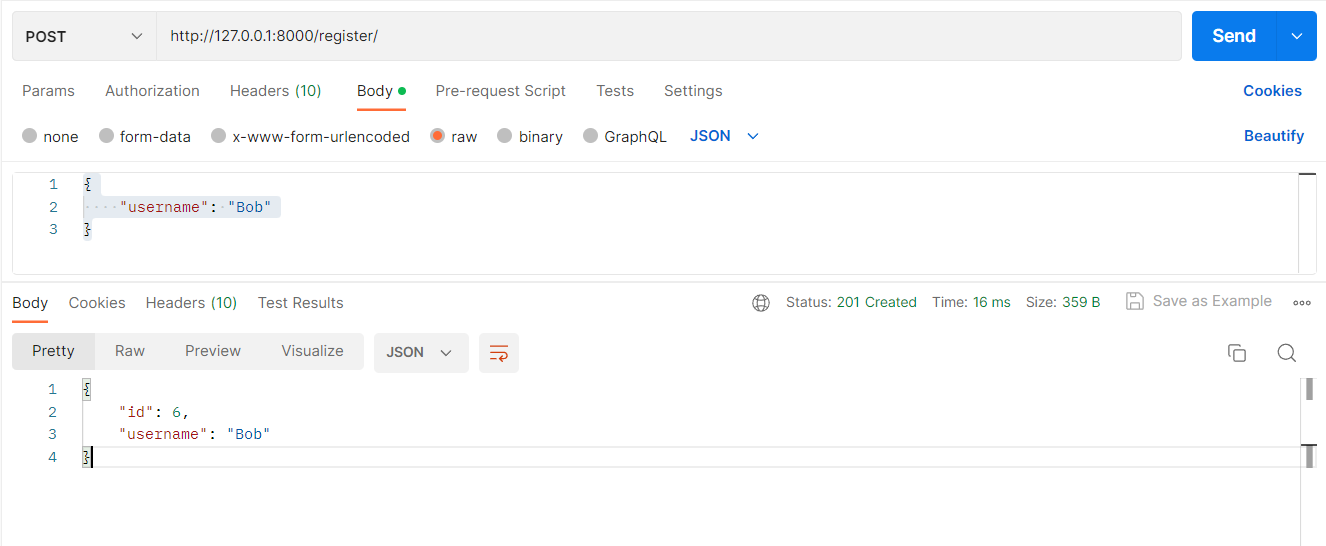
Для начала хочется уточнить, лучше всего проверять сервис в Postman, по крайней мере все примеры реализованы через него: <https://web.postman.co/>

**1. Зарегистрировать нового пользователя.**

Запрос: POST: http://127.0.0.1:8000/register/

{

    "username": "Bob"

}

**2. Отправить одному пользователю заявку в друзья.**

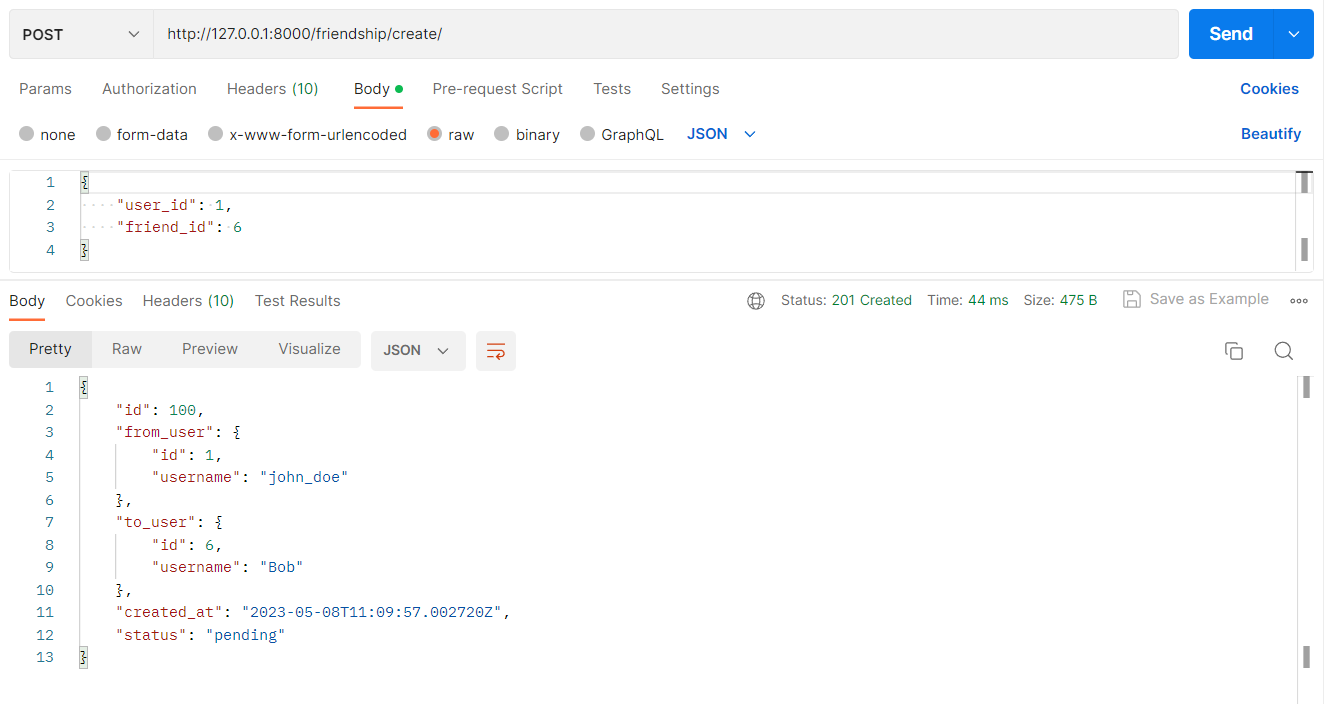
Запрос:

POST: <http://127.0.0.1:8000/friendship/create/>

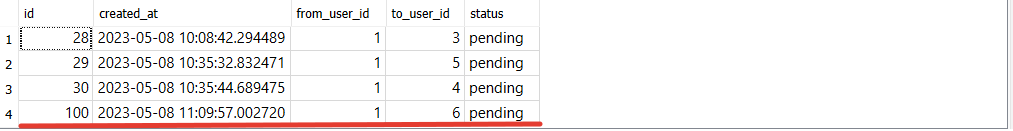
{

    "user\_id": 1,

    "friend\_id": 6

}

Если мы сделаем ещё один запрос, только с обратными id, то автоматически статус станет accepted, а пока в бд:



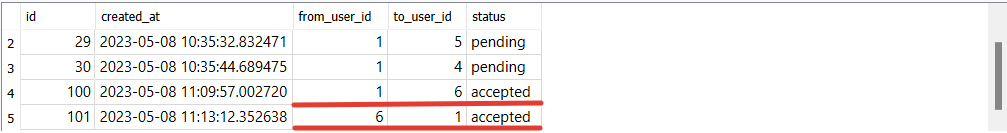
Давайте сделаем обратный запрос: POST: <http://127.0.0.1:8000/friendship/create/>

{

    "user\_id": 6,

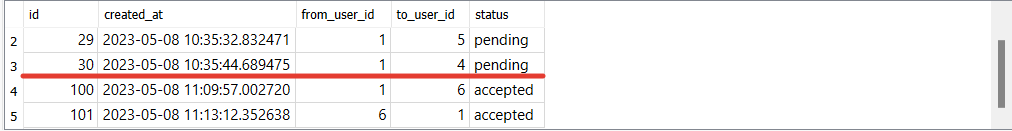
    "friend\_id": 1

}

Теперь посмотрим изменения в бд:

**Отсюда делаем вывод, что 8 пункт реализован!**

**3. Принять\отклонить пользователю заявку в друзья от другого пользователя.**

Если есть только входящий запрос, и пользователь принимает заявку, тогда у входящего запроса меняется статус на ‘accepted’ и у принимающей стороны появляется запись в бд с этим пользователем со статусом ‘accepted’. Если же пользователь отклоняет запрос, тогда у входящей заявки так и останется ‘pending’, а у пользователя статус будет ‘rejected’.

В нашем примере есть исходящая заявка от пользователя 1 пользователю 4. Давайте сделаем запрос:

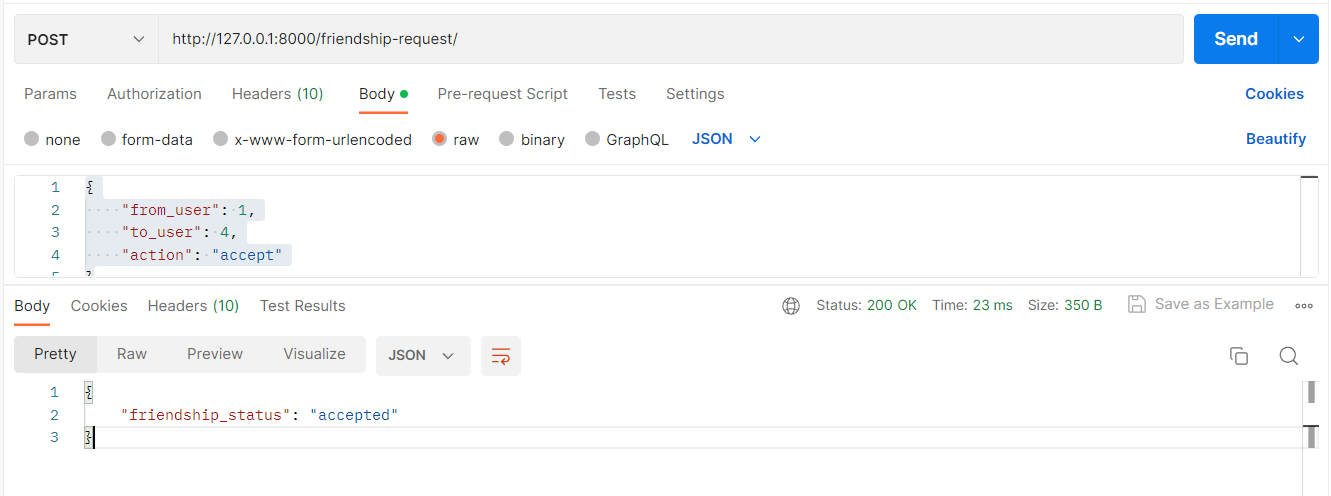
POST: <http://127.0.0.1:8000/friendship-request/>

{

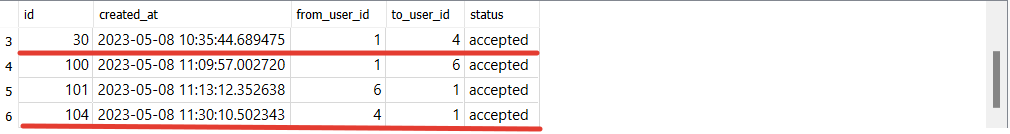
    "from\_user": 1,

    "to\_user": 4,

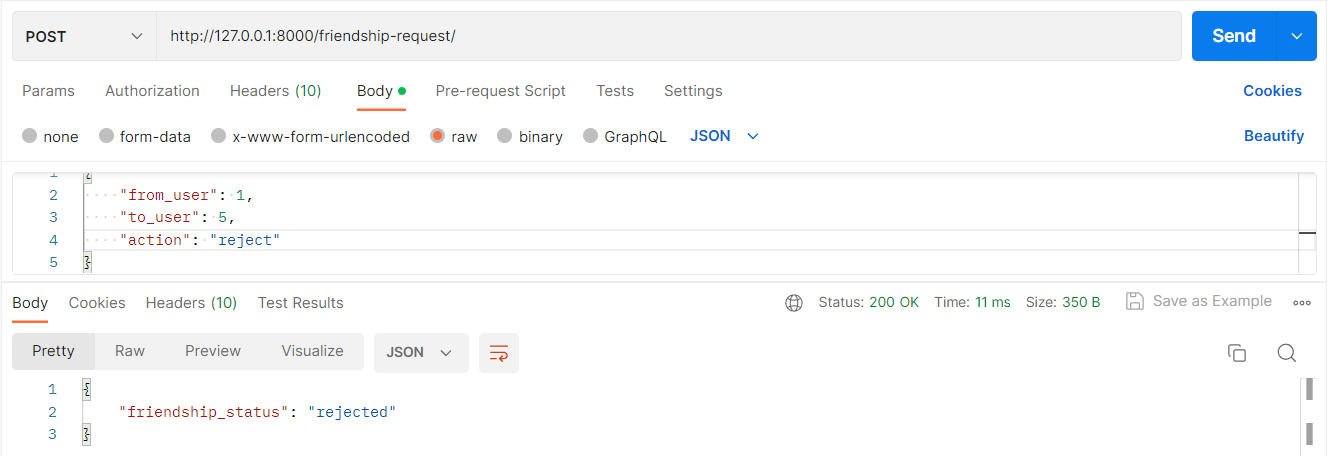
    "action": "accept"

}

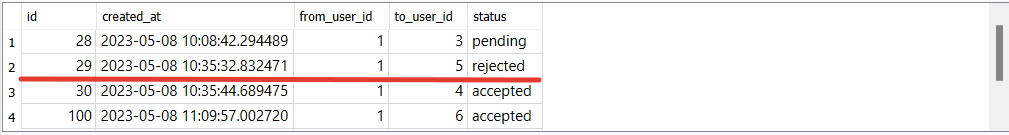
Получаем такой результат, теперь смотрим в бд:



Теперь отклоним запрос:



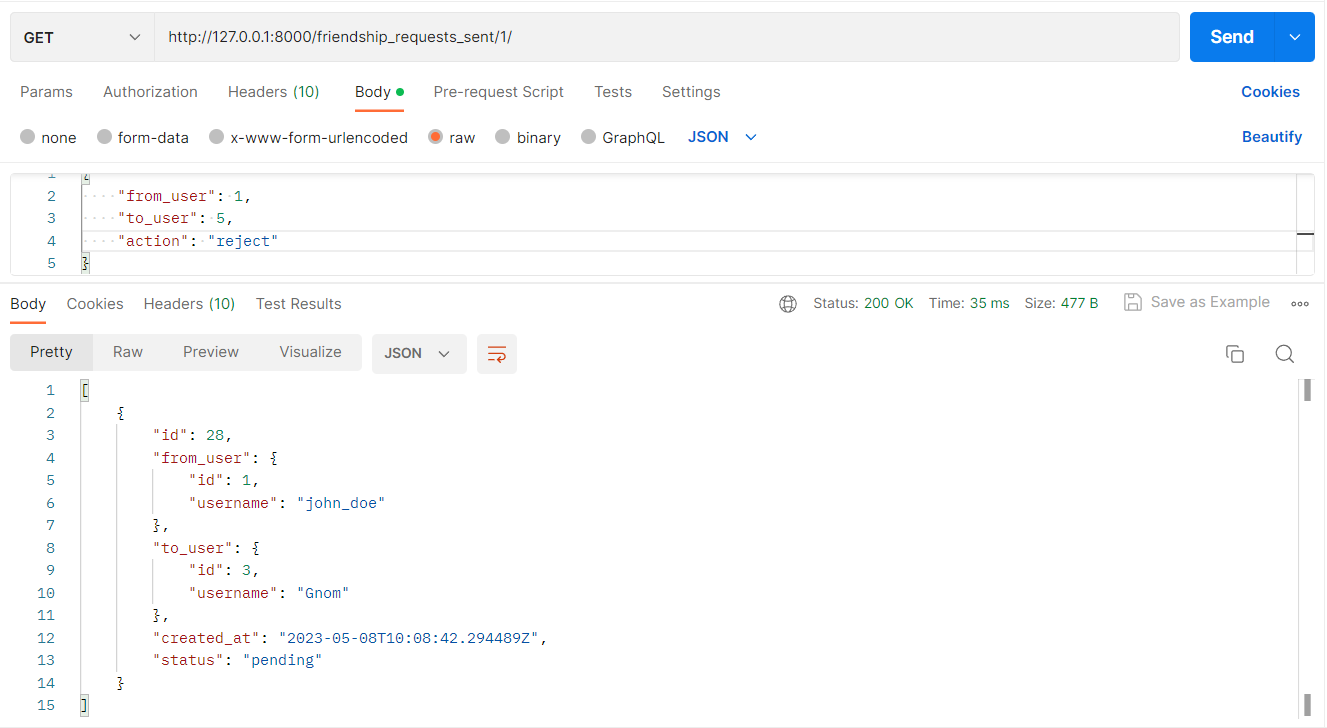
Результат в бд:



**4. Посмотреть пользователю список своих исходящих и входящих заявок в друзья.**

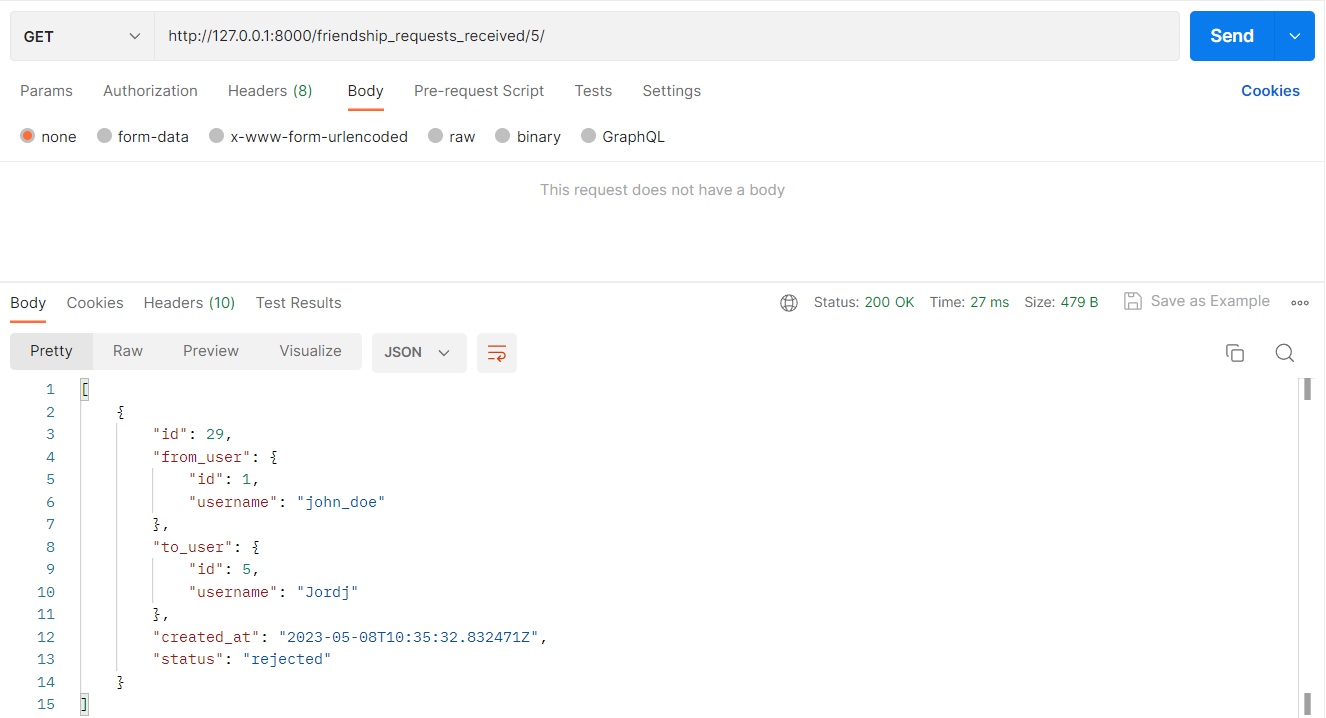
Чтобы посмотреть исходящие заявки, используем такой запрос:

GET: http://127.0.0.1:8000/friendship\_requests\_sent/<int:id>/



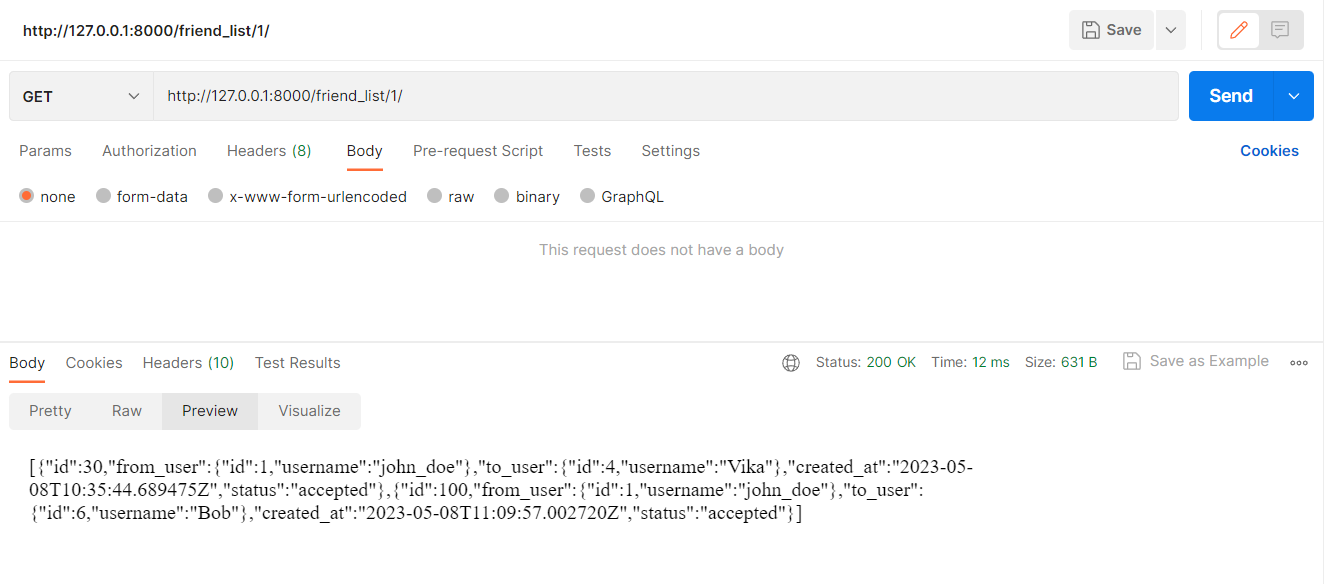
Чтобы посмотреть входящие заявки, используем такой запрос:

GET: <http://127.0.0.1:8000/friendship_requests_received/5/>

Здесь будут отображены все заявки со статусом rejected и pending.

**5. Посмотреть пользователю список своих друзей.**

Делаем такой запрос:

GET: <http://127.0.0.1:8000/friend_list/1/>

**6. Получить пользователю статус дружбы с каким-то другим пользователем (нет ничего\есть исходящая заявка\есть входящая заявка\уже друзья).**

Запрос:

[http://127.0.0.1:8000/friendship\_status/?user=<int>&friend=<int](http://127.0.0.1:8000/friendship_status/?user=%3cint%3e&friend=%3cint)>

Из запроса мы можем получить 4 ответа:

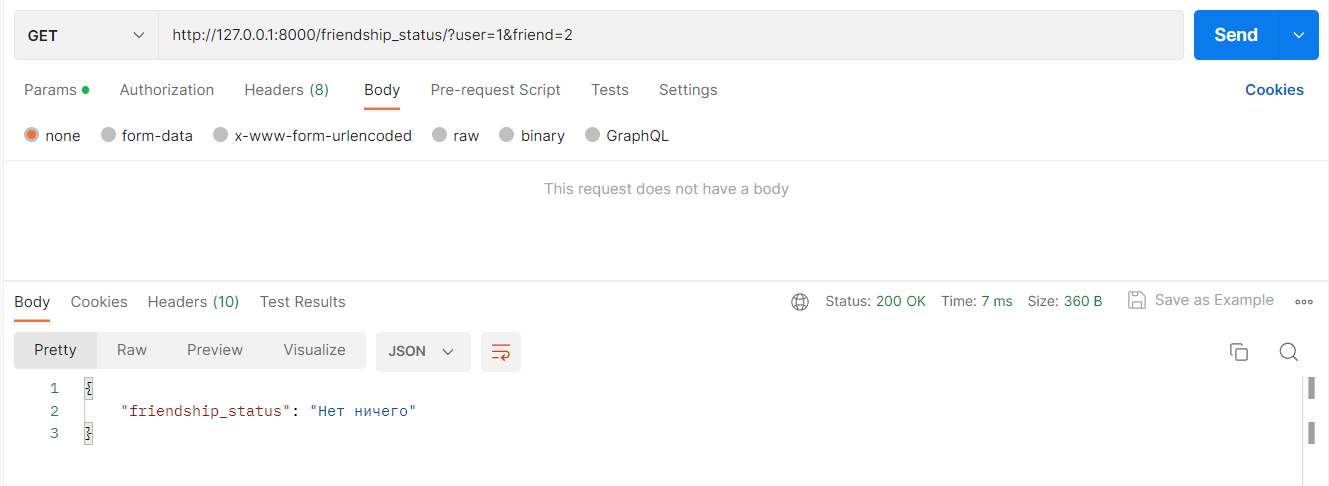
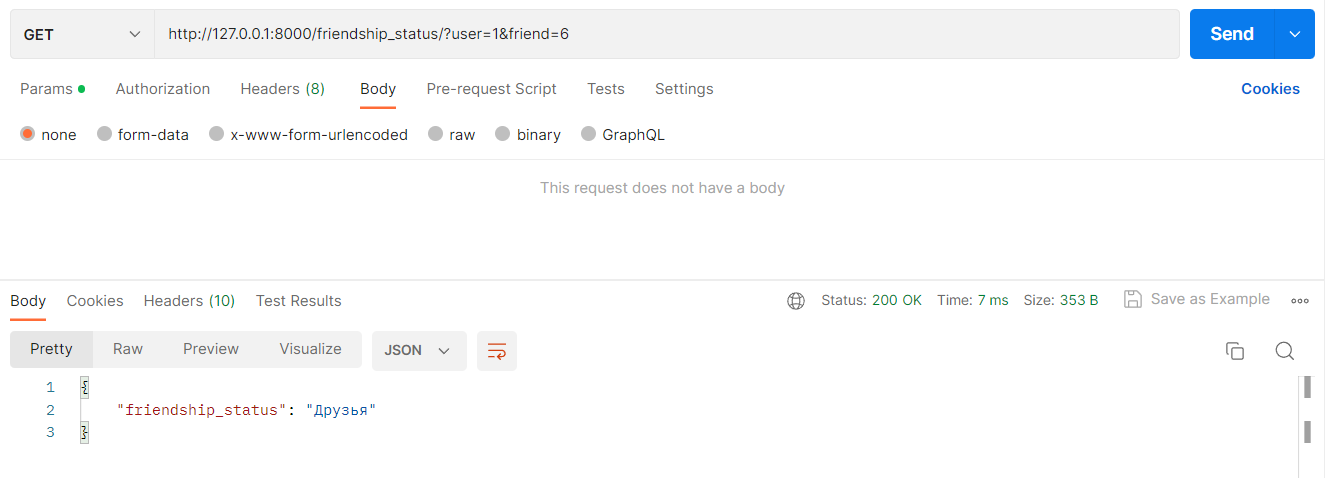
-Нет ничего

-Друзья

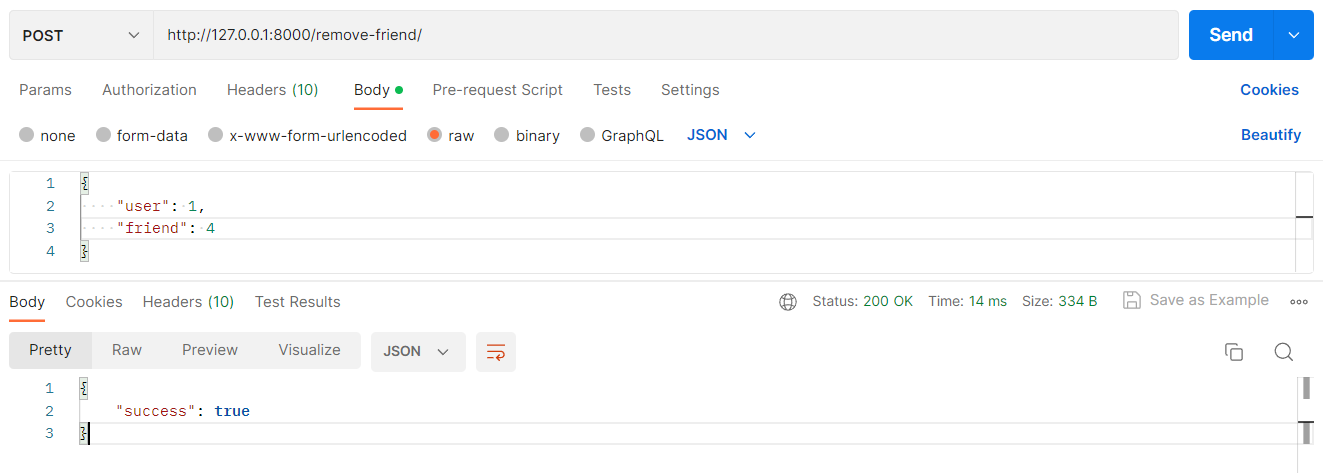
-Исходящая заявка

-Входящая заявка

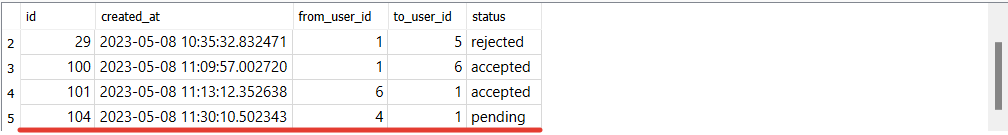
GET: <http://127.0.0.1:8000/friendship_status/?user=1&friend=2>

GET: <http://127.0.0.1:8000/friendship_status/?user=1&friend=6>

**7. Удалить пользователю другого пользователя из своих друзей.**

POST: <http://127.0.0.1:8000/remove-friend/?user=1&friend=4>

Из бд запись 1-4 удаляется со статусом accepted, а у записи 4-1 меняется на pending:



**8 пункт реализован в пункте 2.**