1. **Запуск.**

Для запуска сервиса перейдите в папку **VK\_project/"**, где находится Dockerfile, откройте командную строку и выполните следующие команды для билдинга и запуска образа Docker:

docker build . --tag oparindv\_task

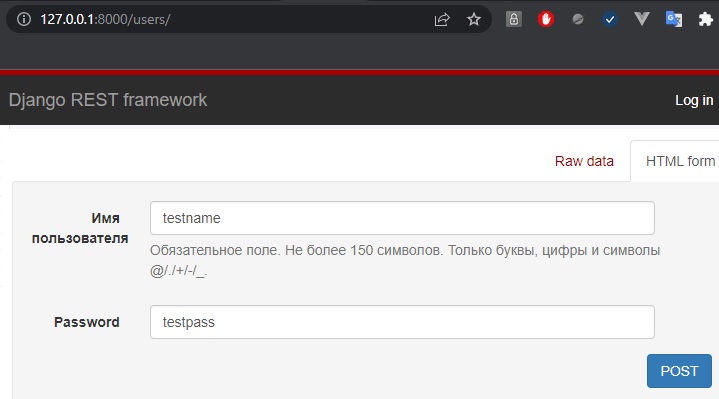
docker run -p 8000:8000 oparindv\_task

1. **Открытие сервиса.**

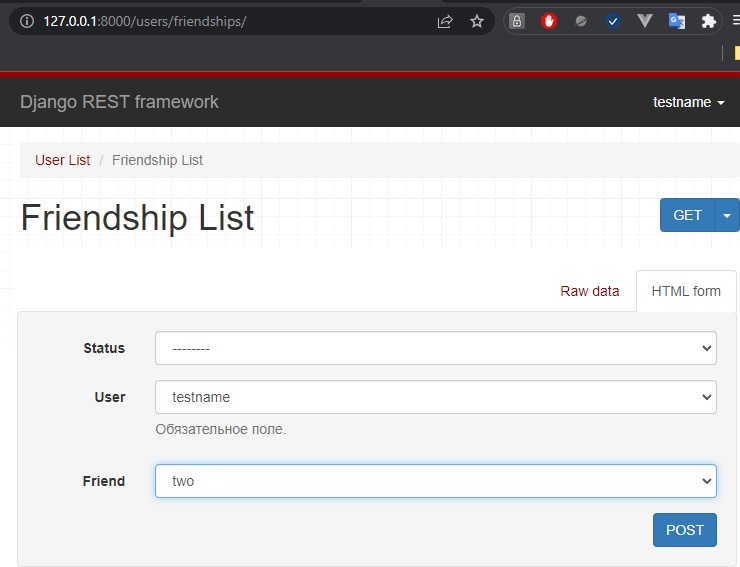
В браузере перейдите по адресу <http://127.0.0.1:8000/users/>

Так же адресу <http://127.0.0.1:8000/api/schema/swagger-ui/> можно увидеть список ендпоинтов с описанием методов согласно OpenAPI 3.0.

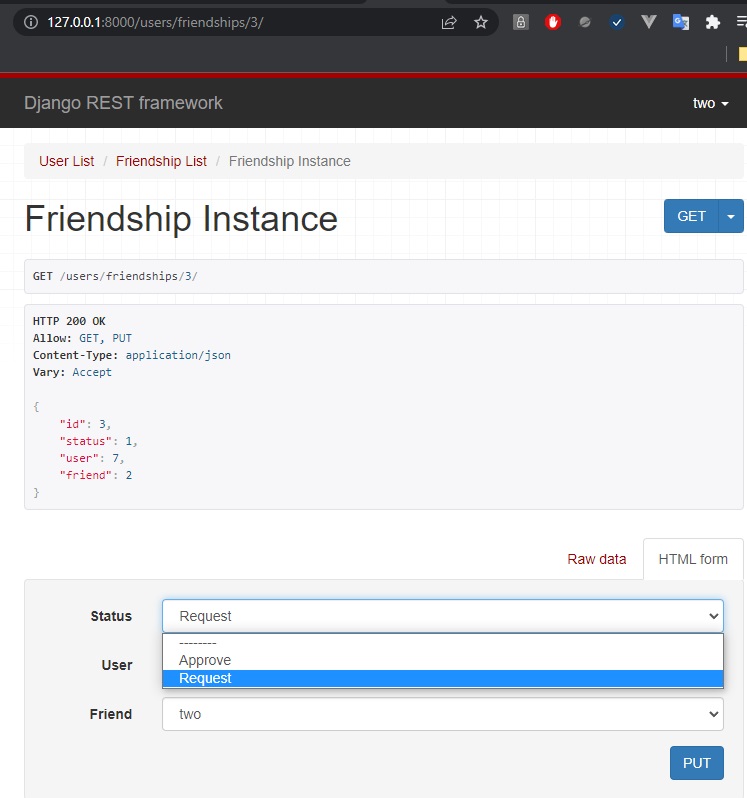
1. **Примеры использования.**
   1. Регистрация нового пользователя.

Необходимо послать POST запрос на /users/ с данными username и password.

* 1. Отправка заявки.

Необходимо послать POST запрос на /friendships/ , где friend это id целевого юзера, status и user будут преобразованы перед сохранением согласно методу 'def perform\_create’ во views.

* 1. Принять/отклонить заявку.

Для того, что принять или отклонить заявку, необходимо отправить PUT запрос на /users/friendships/id где id – это id конкретной заявки в друзья. У заявки есть поле status(None-нет значения, 0-пользователи в друзьях, 1 – заявка отправлена, но не подтверждена), необходимо изменить его на None (отклонить заявку) или 0 (подтверждена). 0 и 1 имеют обозначение, установленное в модели (APPROVE = 0, REQUEST = 1)

* 1. Получить список исходящих и входящих заявок, получить статус дружбы, получить список друзей.

Методы для получения данной информации прописаны в сериалайзере модели User. Информацию можно получить в списке всех User в /users/ или конкретного User /users/id, где

incoming\_requests - список входящих неподтверждённых заявок User’а (имеющих статус 1)

outgoing\_requests – аналогично, только для исходящих

status – статус дружбы, он показывает статус **относительно авторизованного пользователя API,** который делает запрос: (None-связи нет или пользователь API неавторизован, 0-друзья, 1- входящая заявка от него и -1 – исходящая заявка ему)

friends – показывает список друзей пользователя(то есть user’ов с подтверждёнными заявки). Хотелось бы отметить, что пользователи могут быть в друзьях по одной подтверждённой заявке или же по отправленным двум друг другу. В первом случае в базе существует одна запись, во втором две записи. Это повлекло за собой некоторое усложнение кода в методе get\_friends сериализатора User.



Что видит авторизованный пользователь Three:

Пользователь Three(id=3) имеет исходящую заявку пользователю Two. Заявка отражается в outgoing\_requests.

У пользователя Two можно увидеть эту же заявку в incoming\_requests и статус (-1 – исходящая заявка в контексте пользователя three)

Пользователь One и Two уже установили дружеские отношения.

Пользователь One и Two не отправляли друг другу заявок, статус null.

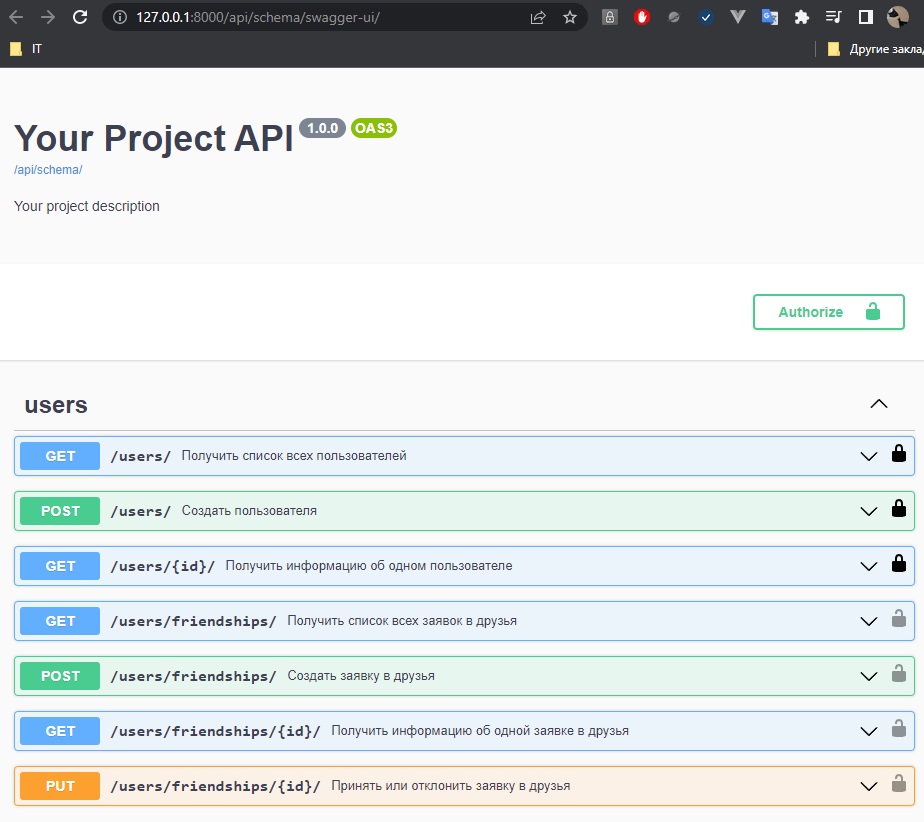
* 1. Встречные заявки.

При отправке двух встречных заявок друг другу их статус автоматически устанавливается в 0 (дружба). Заявки пропадают из incoming\_requests и outgoing\_requests, пользователи появляются друг у друга в друзьях.

* 1. Удалить пользователя из друзей.

Для удаления пользователя необходимо аннулировать статус исходящей заявки (метод PUT), если она есть. У входящей изменить статус на 1. Получается, что пользователь A удалил B, но пользователь B остался подписан на A.

1. **Примечания**

4.1 В папке с проектом есть файл «openapi-schema.yml», в человеко-читаем формате спецификацию можно открыть по адрессу <http://127.0.0.1:8000/api/schema/swagger-ui/>

4.2 В папке tests находятся файлы для запуска тестов, покрытие неполное, но основные части кода проверены. Для запуска тестов проще всего зайти через виртуальное окружение, в которое необходимо установить модули из файла requirements.txt, и выполнить команду python manage.py test. Или же как более сложный способ, зайти в контейнер, остановить сервер и запустить команду там.

4.3 Я постарался обойтись без метода DELETE, тк насколько понимаю, это желательная практика, не удалять записи из БД, а ставить в поле NULL.

*Спасибо за внимание!* ☺