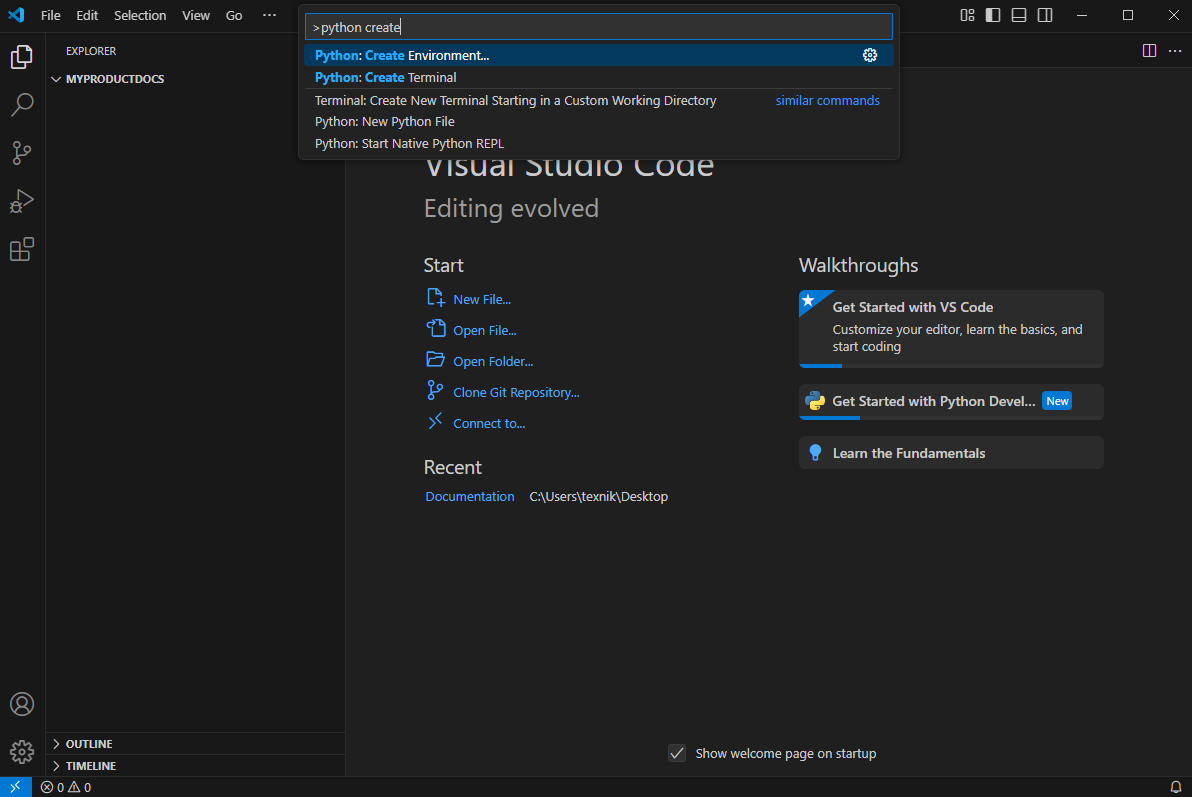
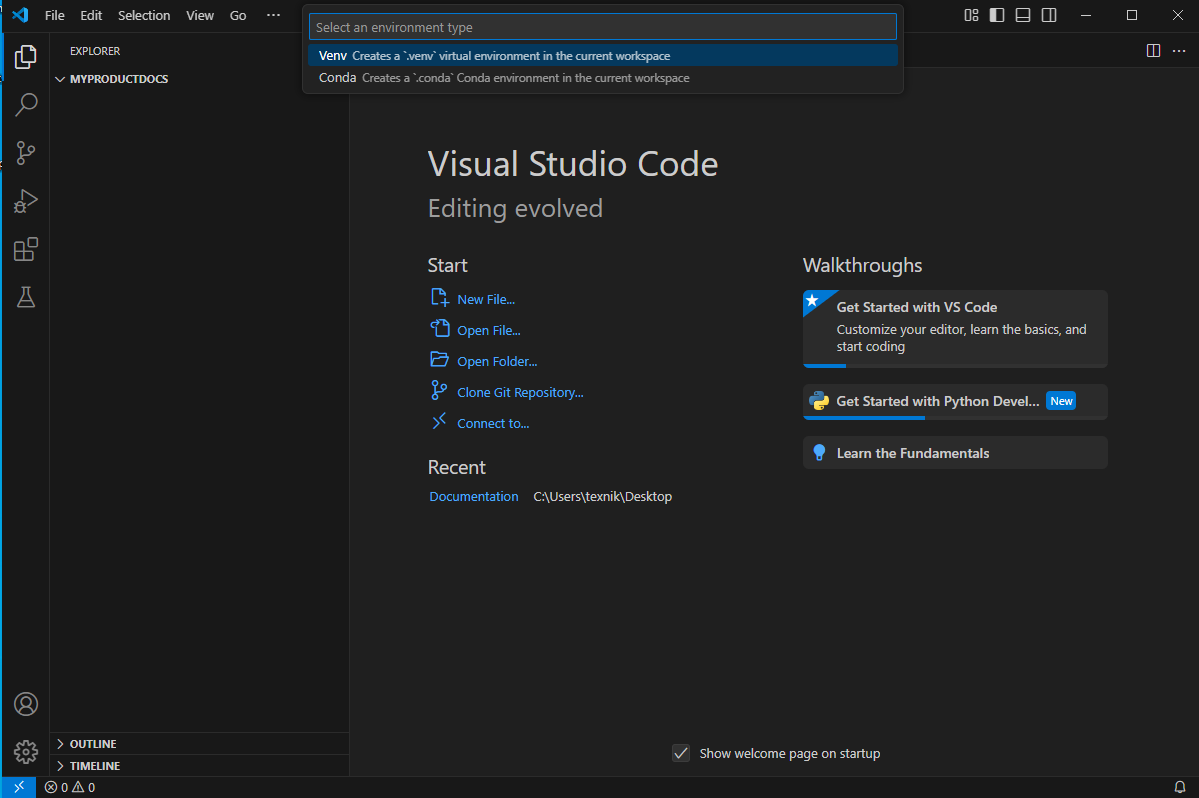
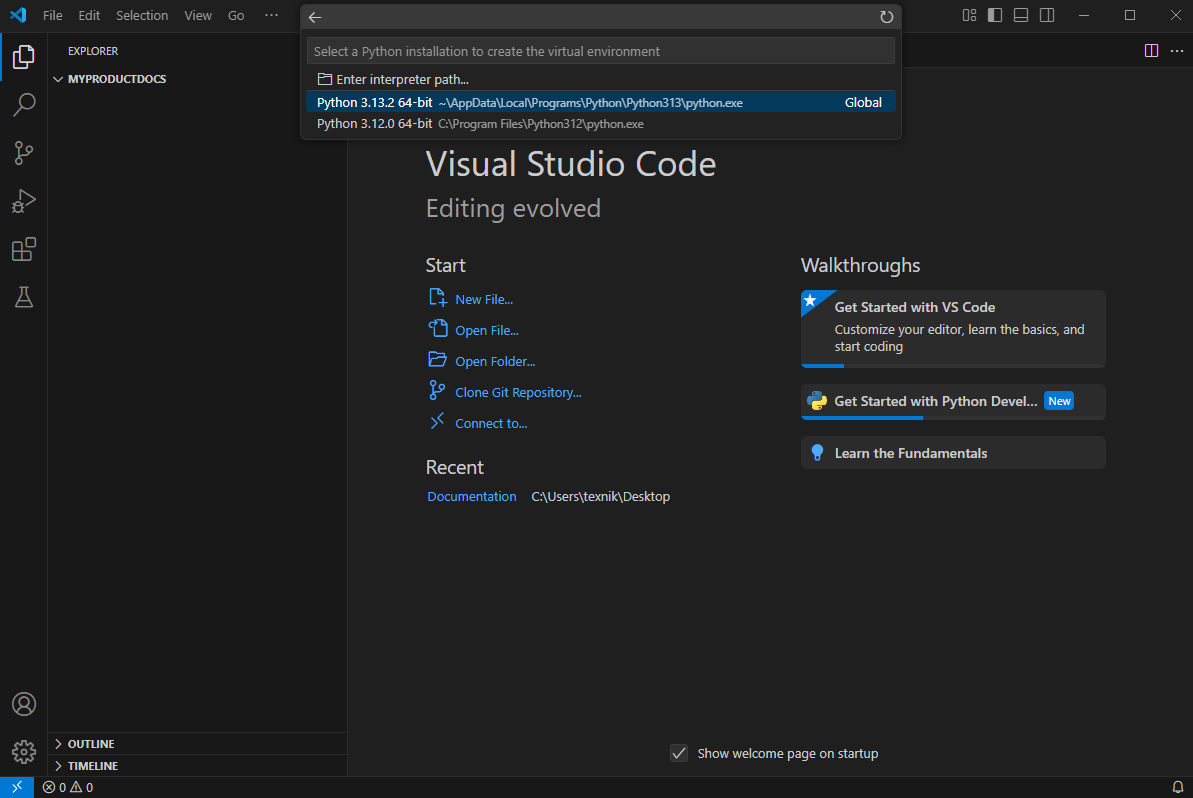
1. Создать VENV(виртуальное окружение)  
   Нажмите CTRL + SHIFT + P и напишите в этом окне

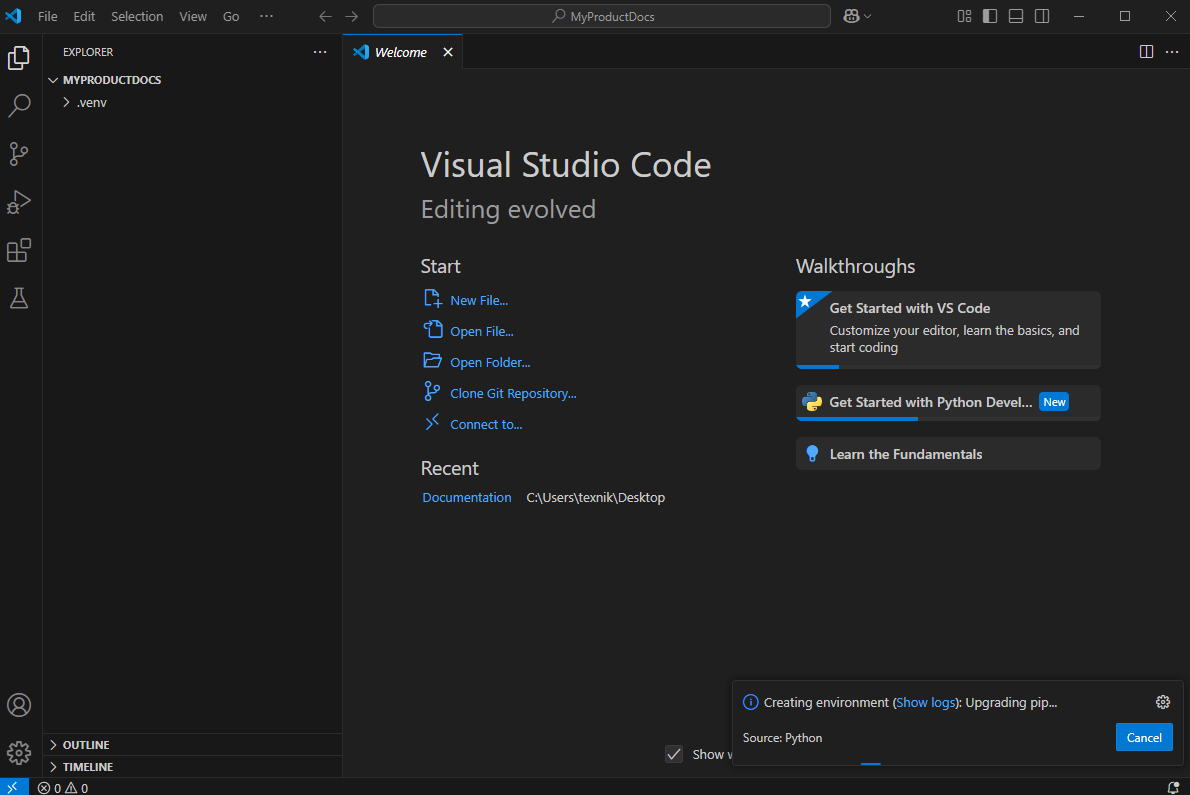
Python create environment  


Выберите VENV  
  


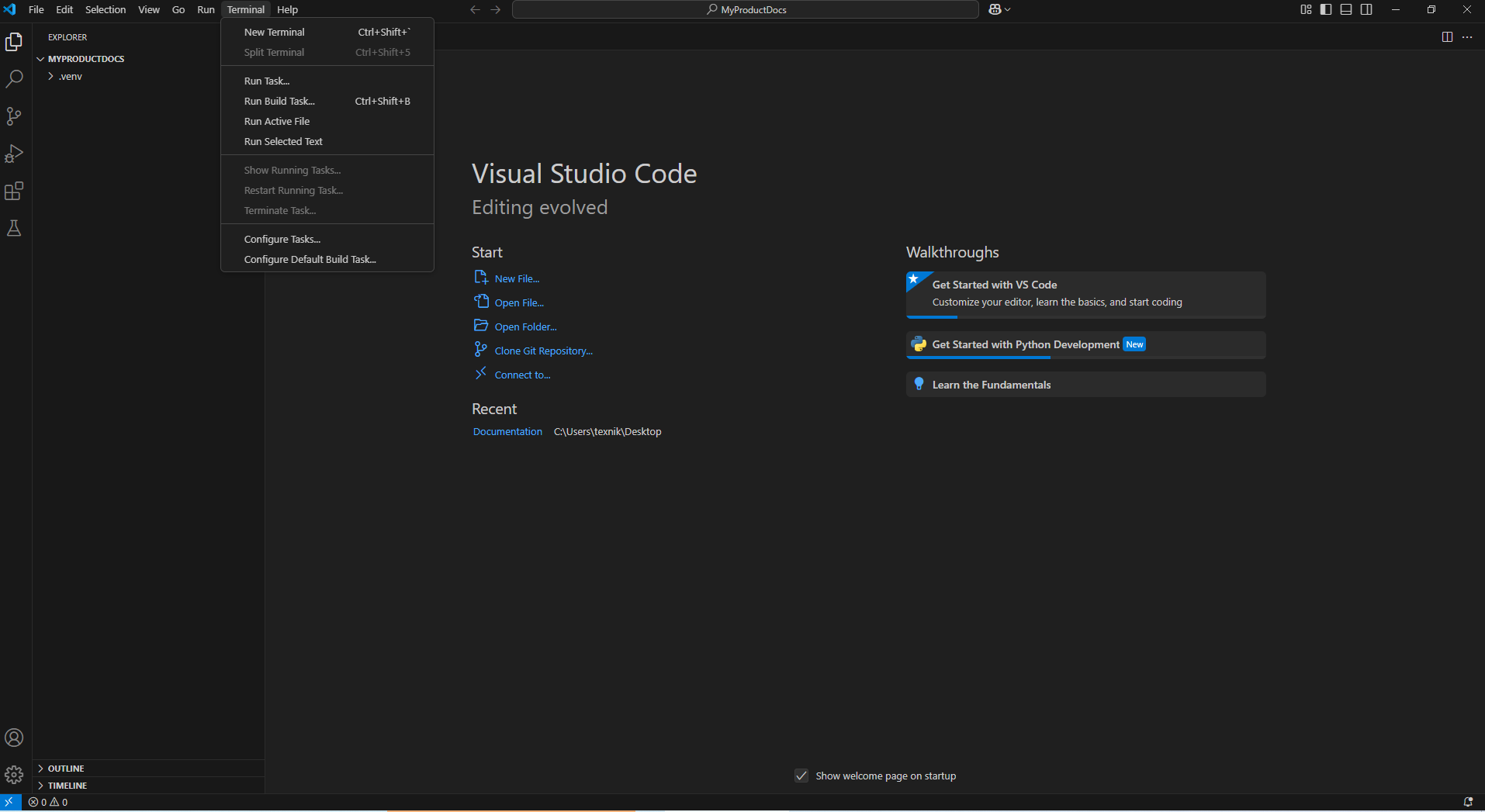
Выберите глобального интерпретатора



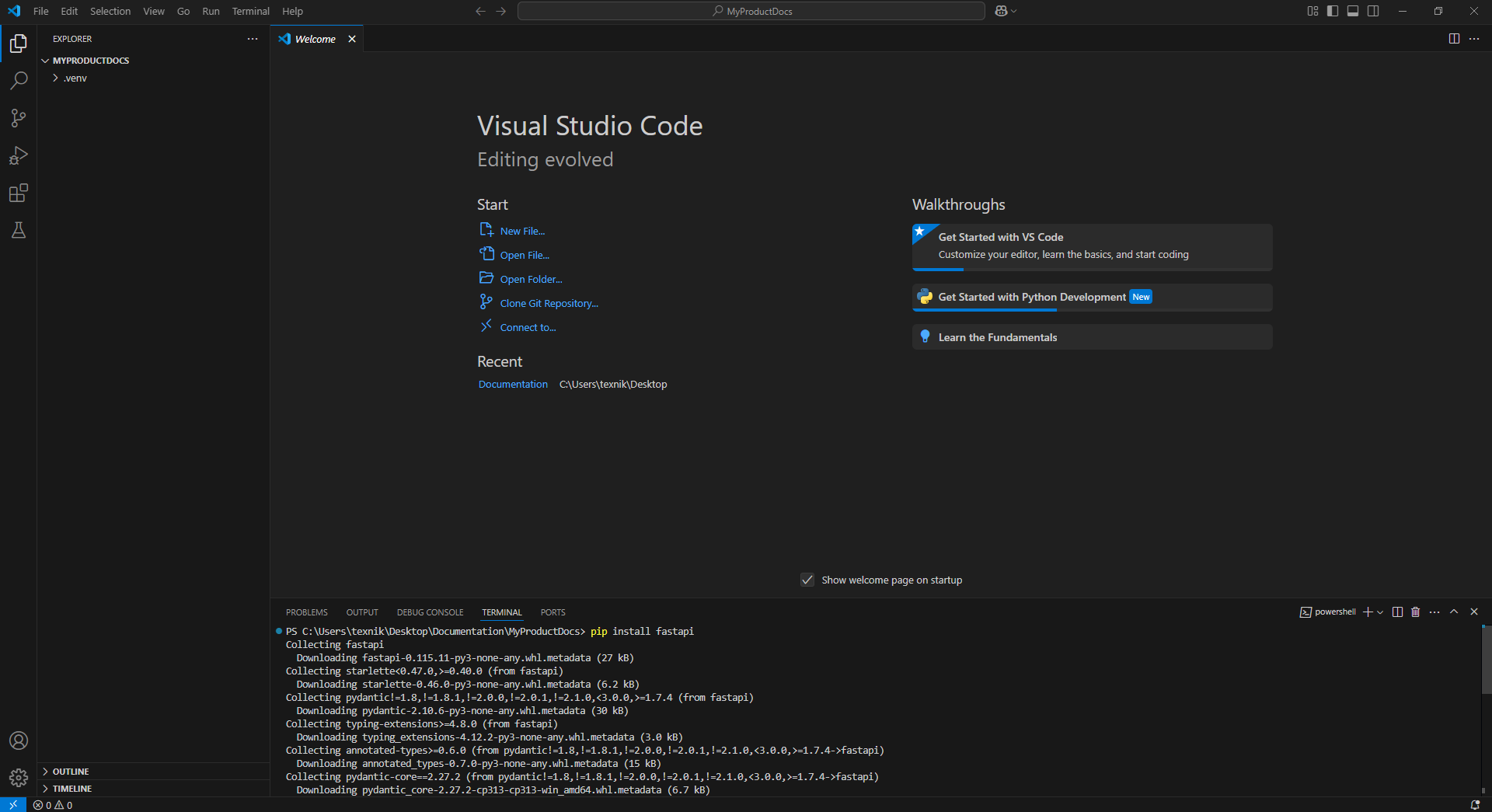
Начнется загрузка около минуты



Нажмите на Terminal и на кнопку New Terminal



В появишемся окне введите pip install fastapi



Начнется скачивание

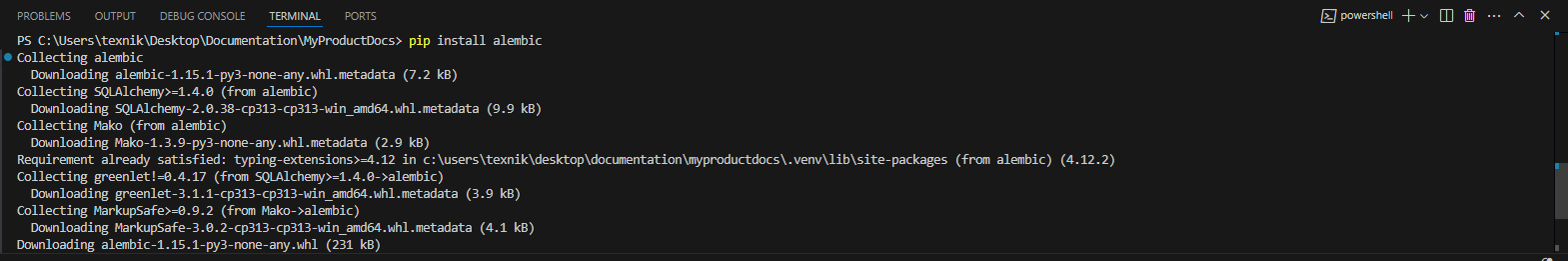
Потом введите

pip install alembic

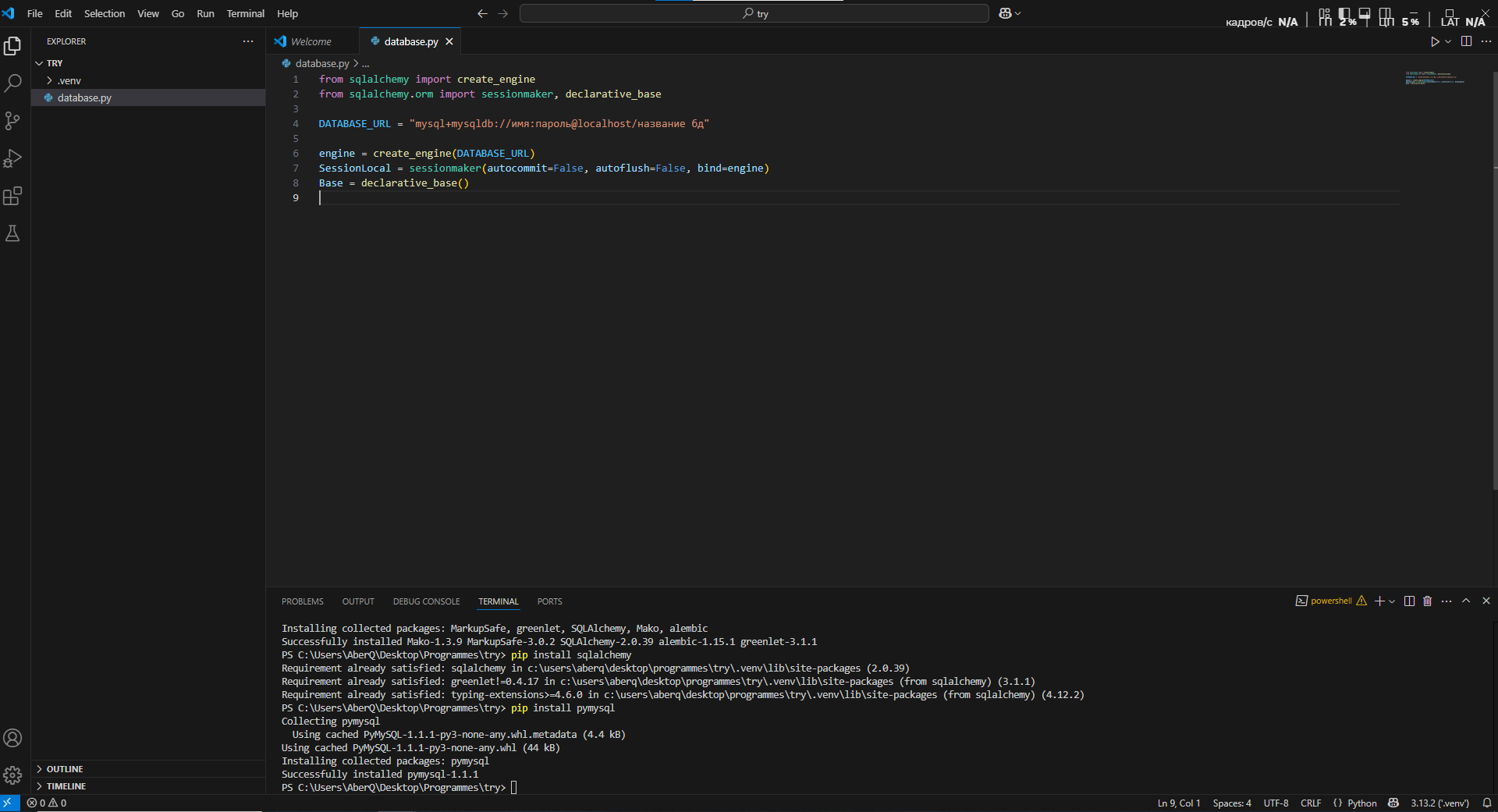
pip install pymysql

pip install uvicorn

и начнется загрузка



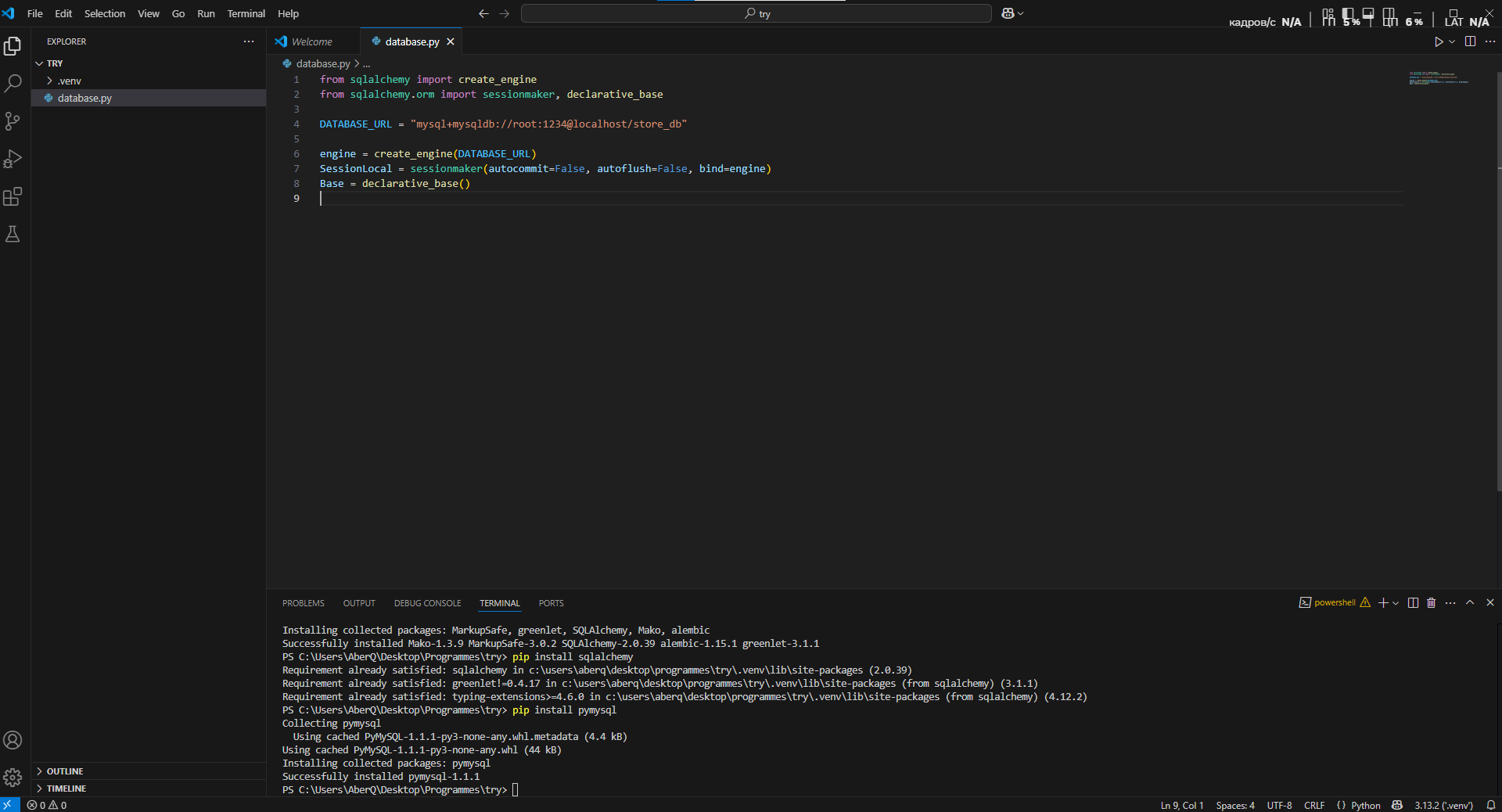
Теперь создадим файл database.py для того, чтоб создать связь с базой данных



ВНИМАНИЕ!!!

Учтите, что нужно заменить имя пользователя бд, пароль, имя бд на свои значения.

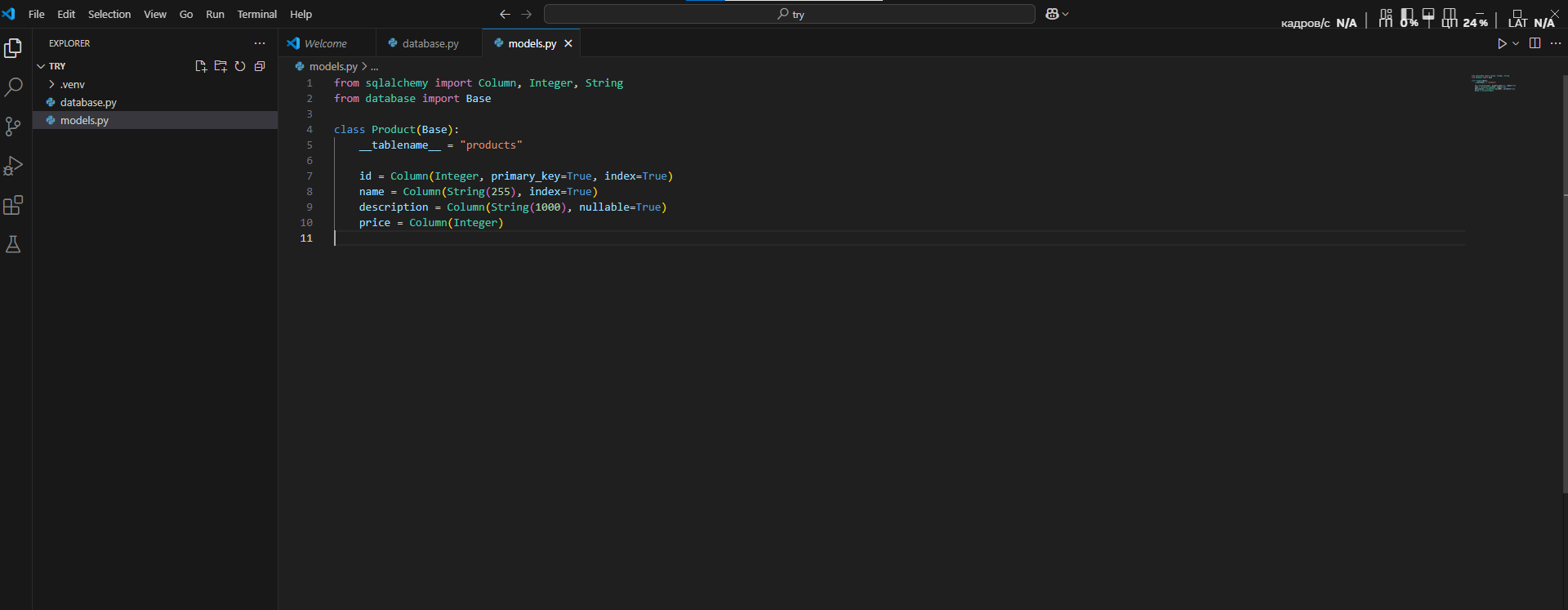
В моем случае я назвал свой аккаунт при скачивании MySQL root а пароль дал 1234. БД хочу назвать store\_db



Теперь давайте создадим файл models.py в нем мы будем хранить модели используемые в работе приложения.

Примечание:  
Модель — это описание структуры данных, используемое в FastAPI для обработки и валидации входных и выходных данных. Существуют два типа моделей:

1. Pydantic-модели – применяются для проверки, сериализации и десериализации данных, передаваемых в API.
2. SQLAlchemy-модели – используются для работы с базой данных, представляя таблицы и их поля.



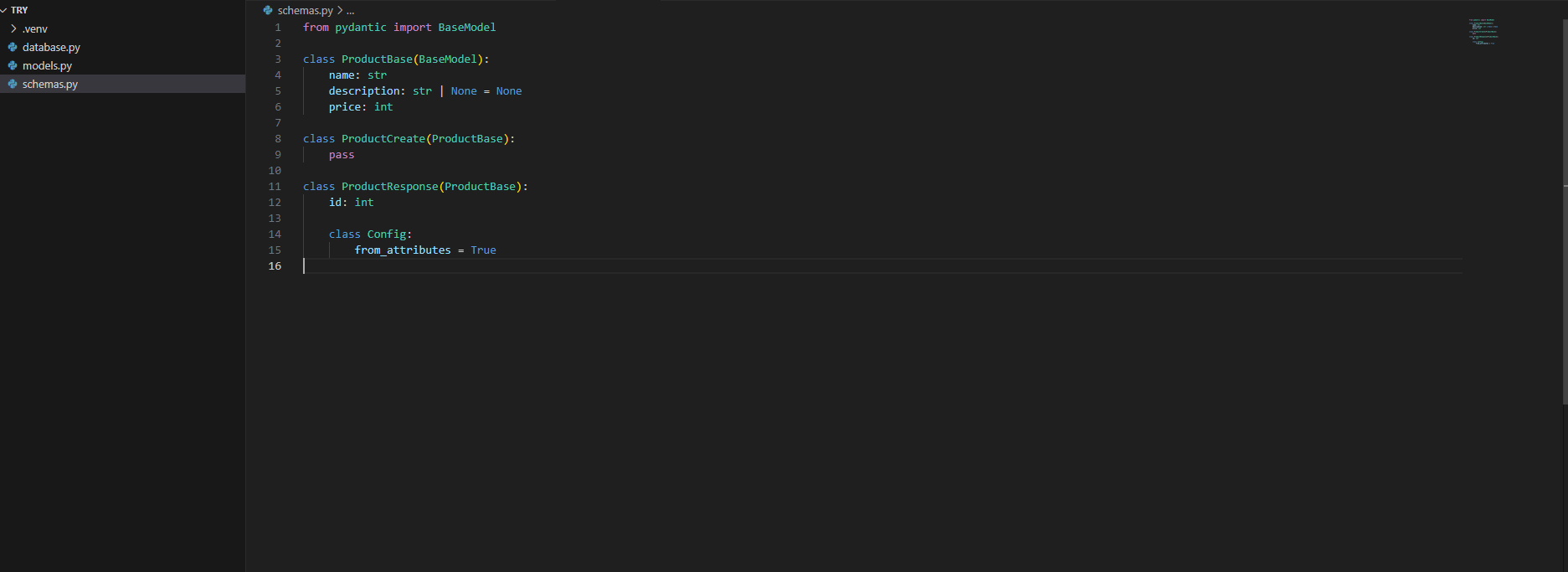
Мы указали поля, а также их типы в нашей базе данных благодаря этой модели.

Я предполагаю, что вы уже знакомы с полями типа STRING и INTEGER, а также знаете про primary\_key и foreign\_key.

Index=true означает, что мы добавляем это поле в индекс для значительного ускорения производительности базы данных.

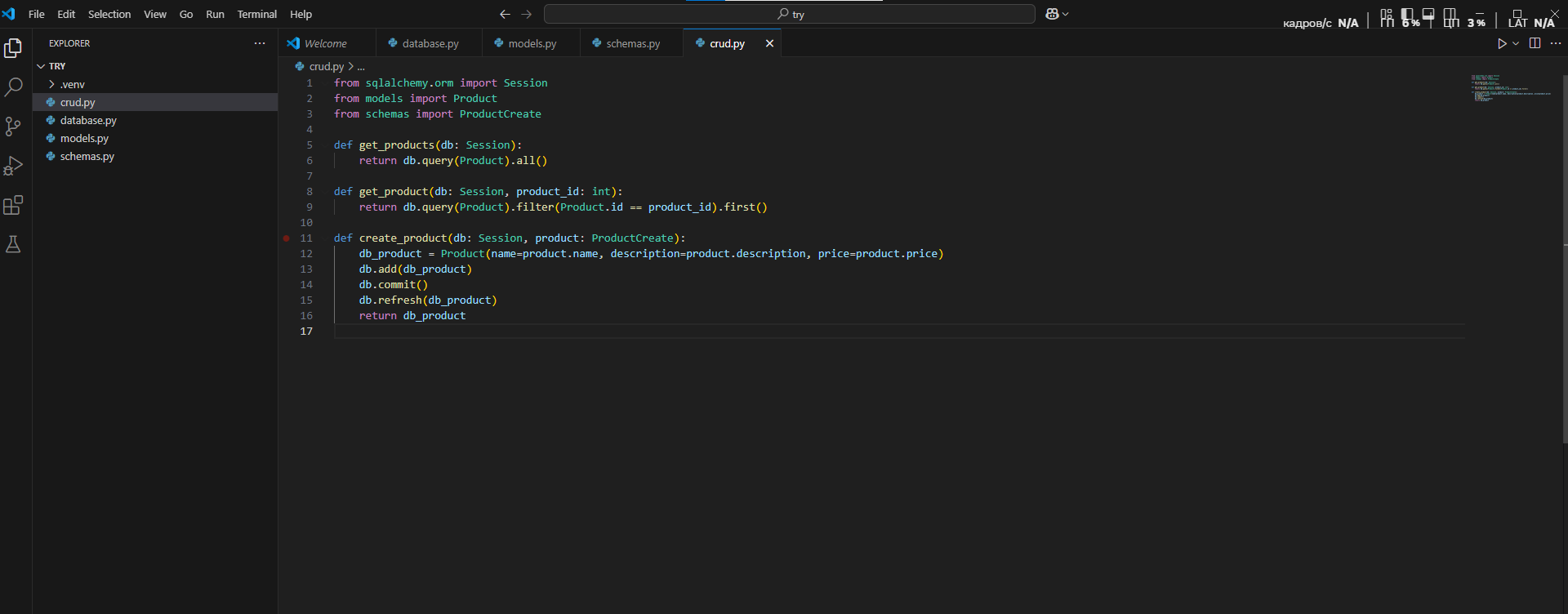
nullable=True означает, что мы допускаем возможность оставление поля пустым(без текста)

Теперь надо сделать schemas.py для создания pydantic-моделей и сериализации данных.



Теперь мы можем продумать логику работы с базой данных.

Создадим файл crud.py(от SQL-синтаксиса CREATE READ UPDATE DELETE) и создадим в нем необходимые функции для приложения.

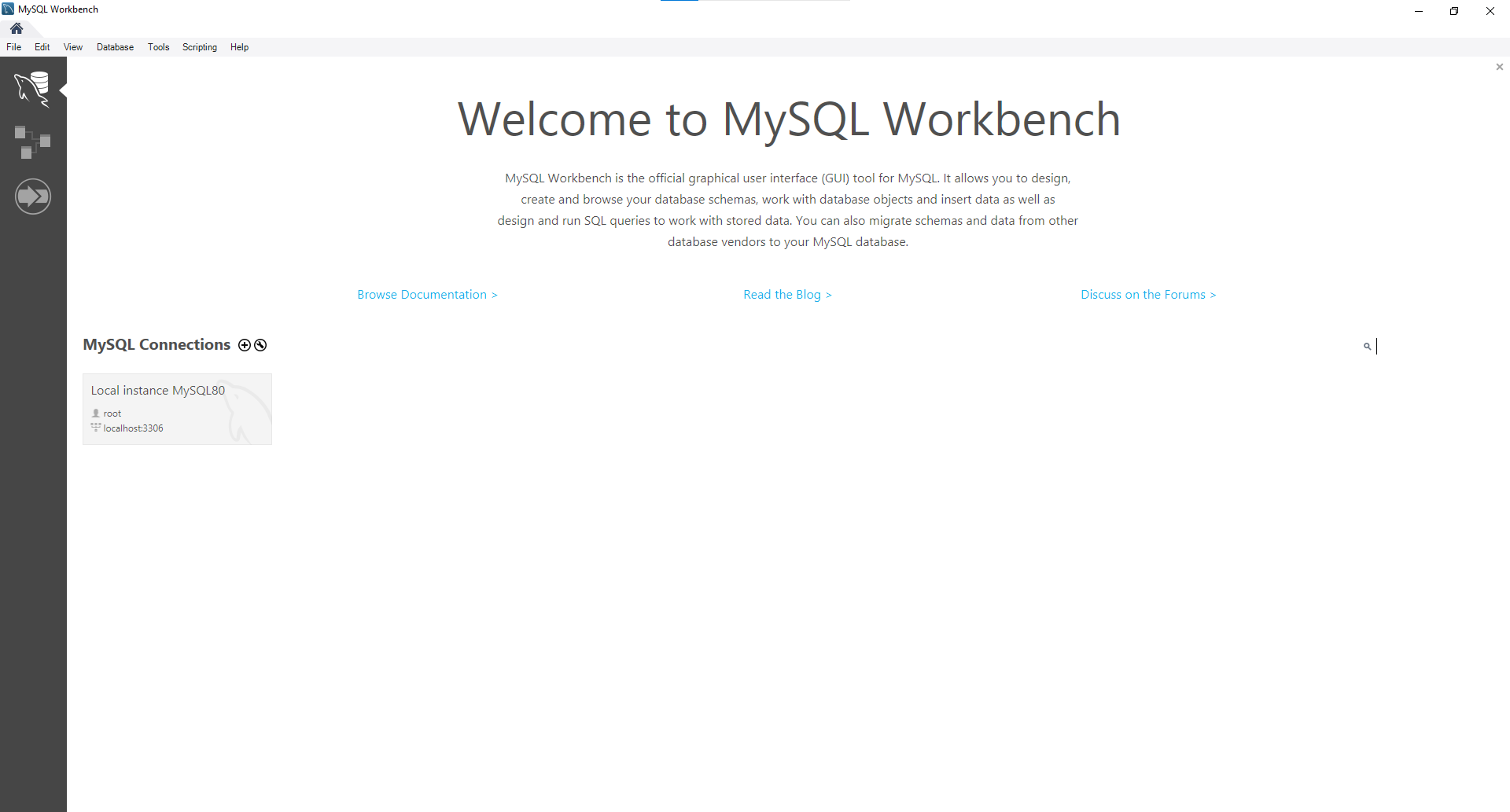


Мы создали 3 функции.

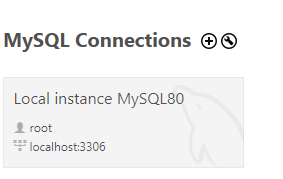
1. Get\_products позволяет получить все продукты из бд.
2. Get\_product позволяет получить данные о конкретном продукте через бд.
3. Create\_product позволяет создать продукт в бд.

Теперь когда мы придумали всю логику работы с бд, можно мигрировать изменения в MySQL

Давайте откроем MySQL Workbench

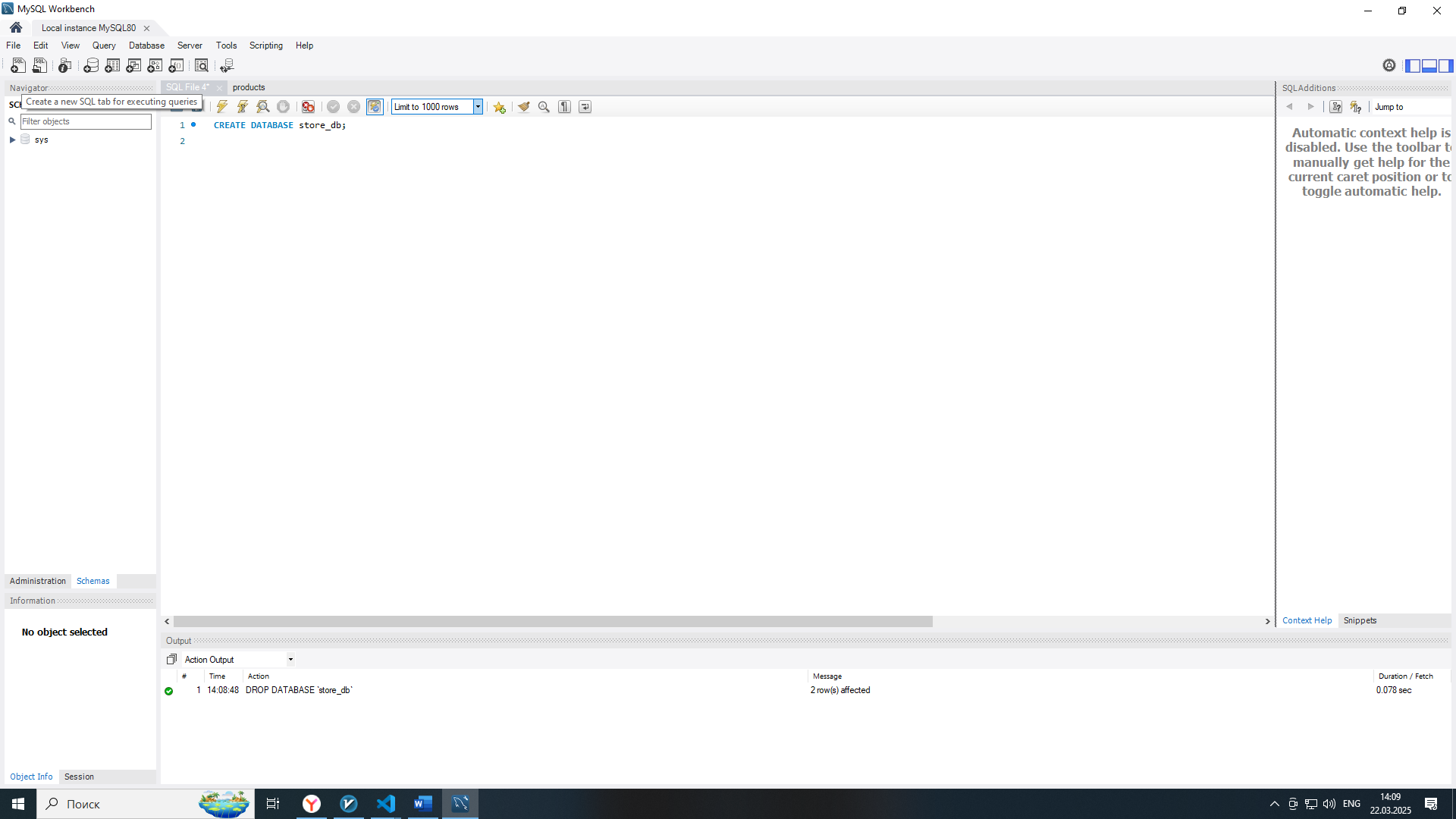


Перейдем по вкладке соединений

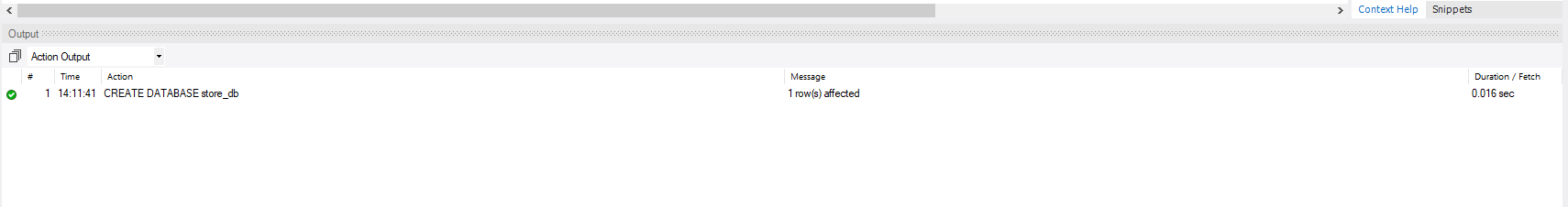


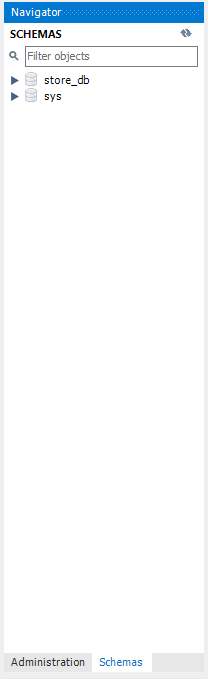
Примечание: здесь, кстати, написан ник вашего пользователя и локальный адрес.

Сначала создадим SQL-файл и введем команду для создания базы данных(используйте свое имя бд, если оно не store\_db)



Активируем и должны получить успешное создание.





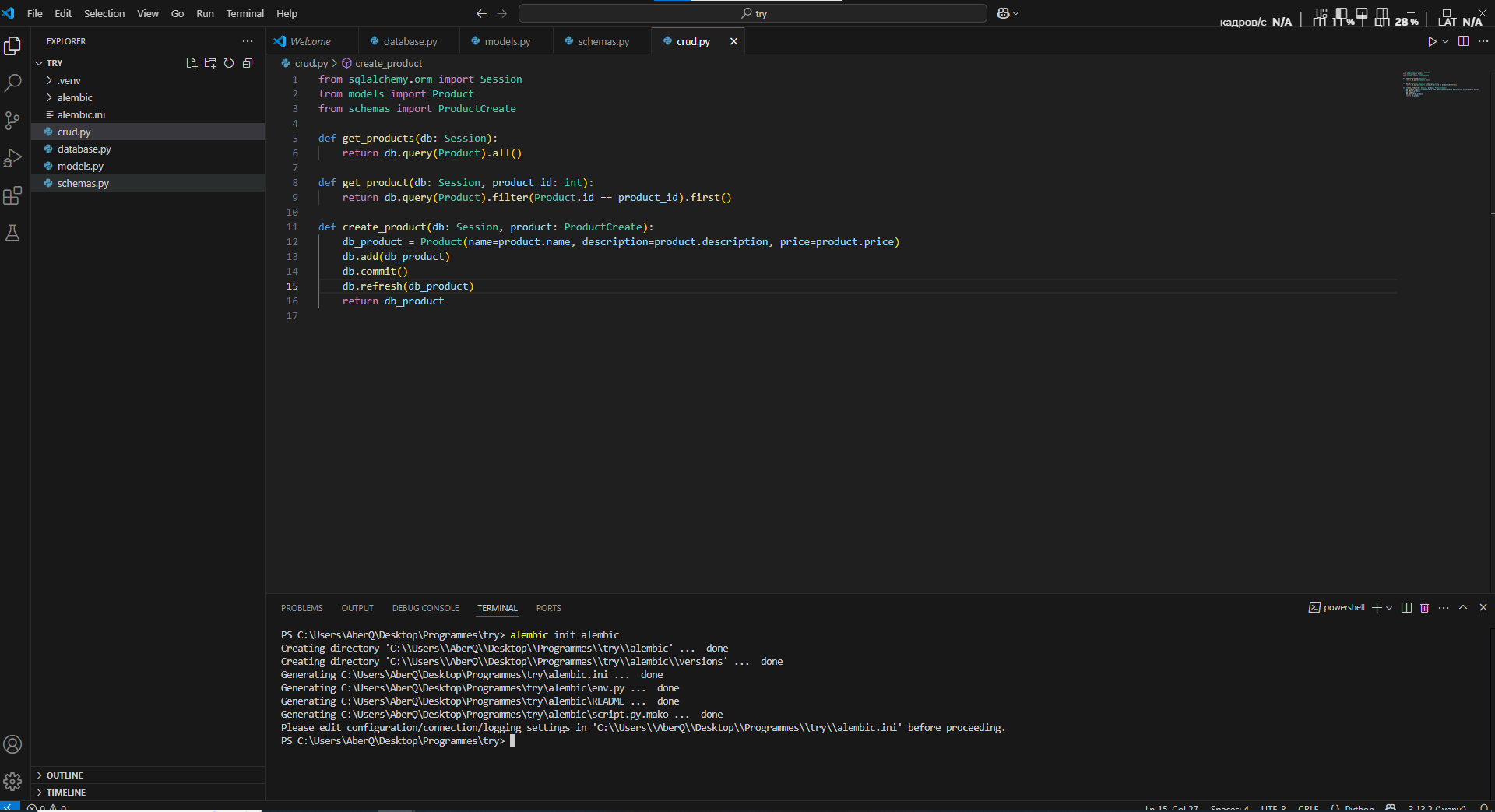
В левой навигационной панели по вкладке Schemas мы сможем найти нашу базу данных.

Она еще будет пустой, потому что мы не провели миграцию.

Давайте обратно перейдем в VS code и инициализируем Alembic.

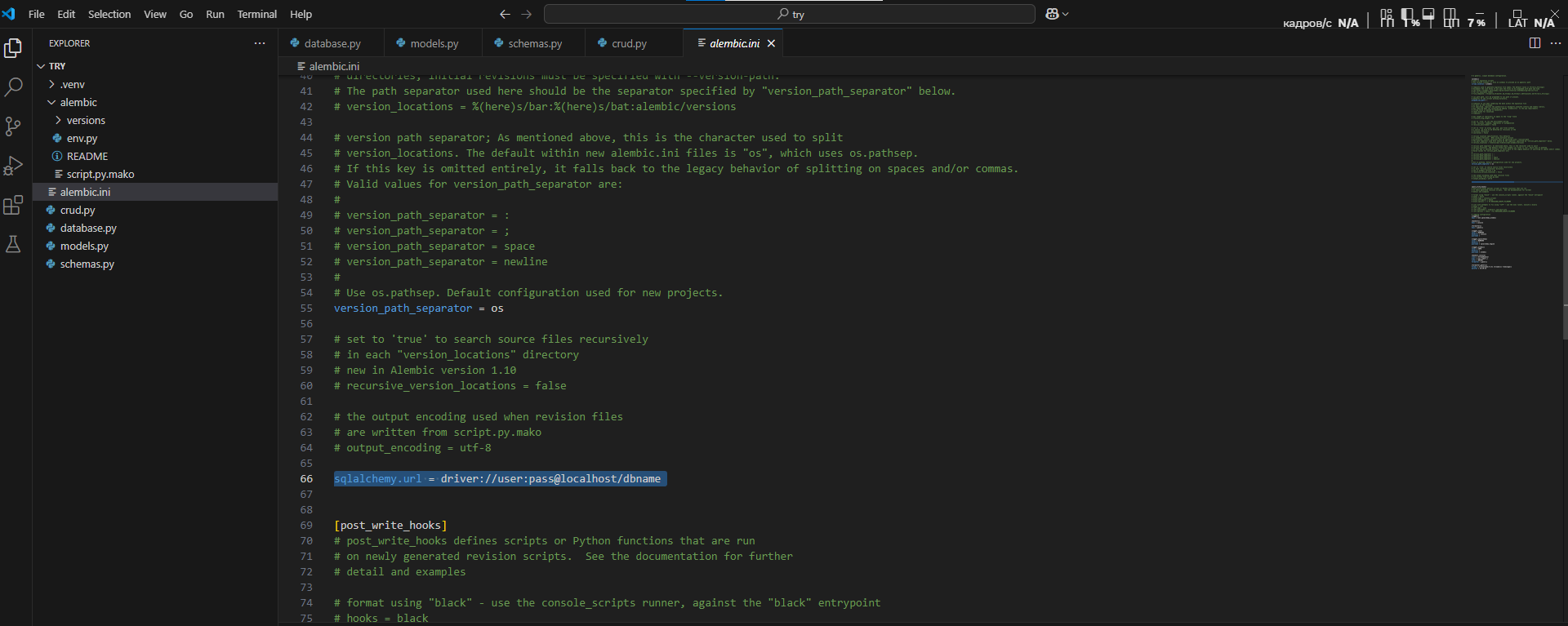
Alembic – это лучший инструмент для FastAPI позволяющий эффективно создавать и отслеживать миграции моделей в бд.

Пишем в терминале alembic init alembic



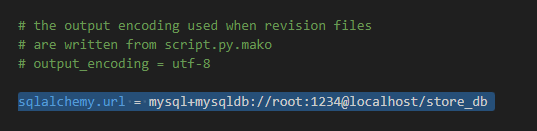
Получаем новые файлы alembic.ini и папку alembic. Здесь и будет наше основное взаимодействие с бд.

Давайте перейдем в alembic.ini и укажем адрес нашей бд.

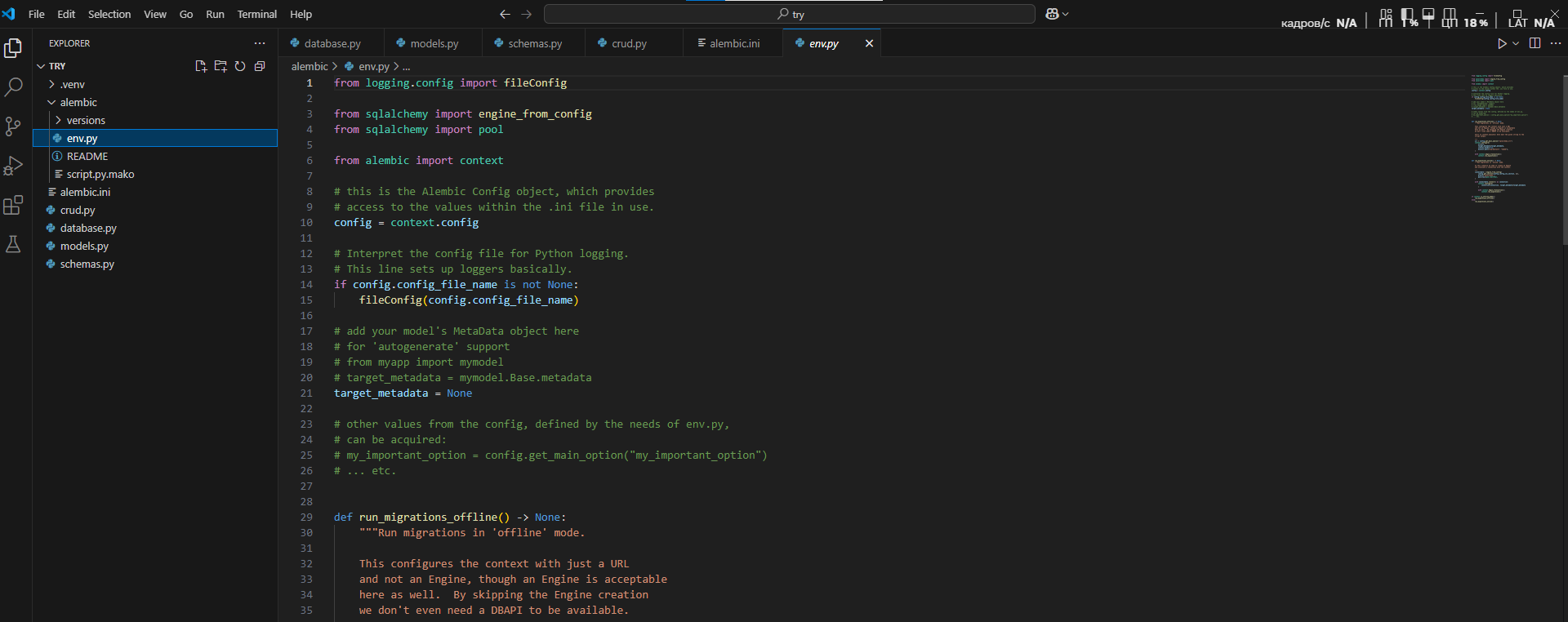


Найдите эту строчку в коде.

Замените ее значение на адрес нашей бд, который мы прописывали в database.py(не забудьте убрать кавычки, иначе будет ошибка)

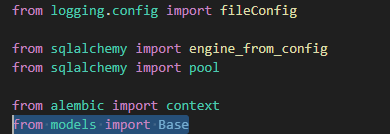


Теперь давайте зайдем в папку alembic и найдем файл env

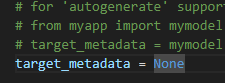


Здесь тоже нужно указать нашу базу данных.

Сначала добавьте import

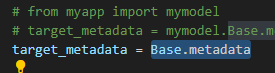


Теперь давайте найдем эту строчку



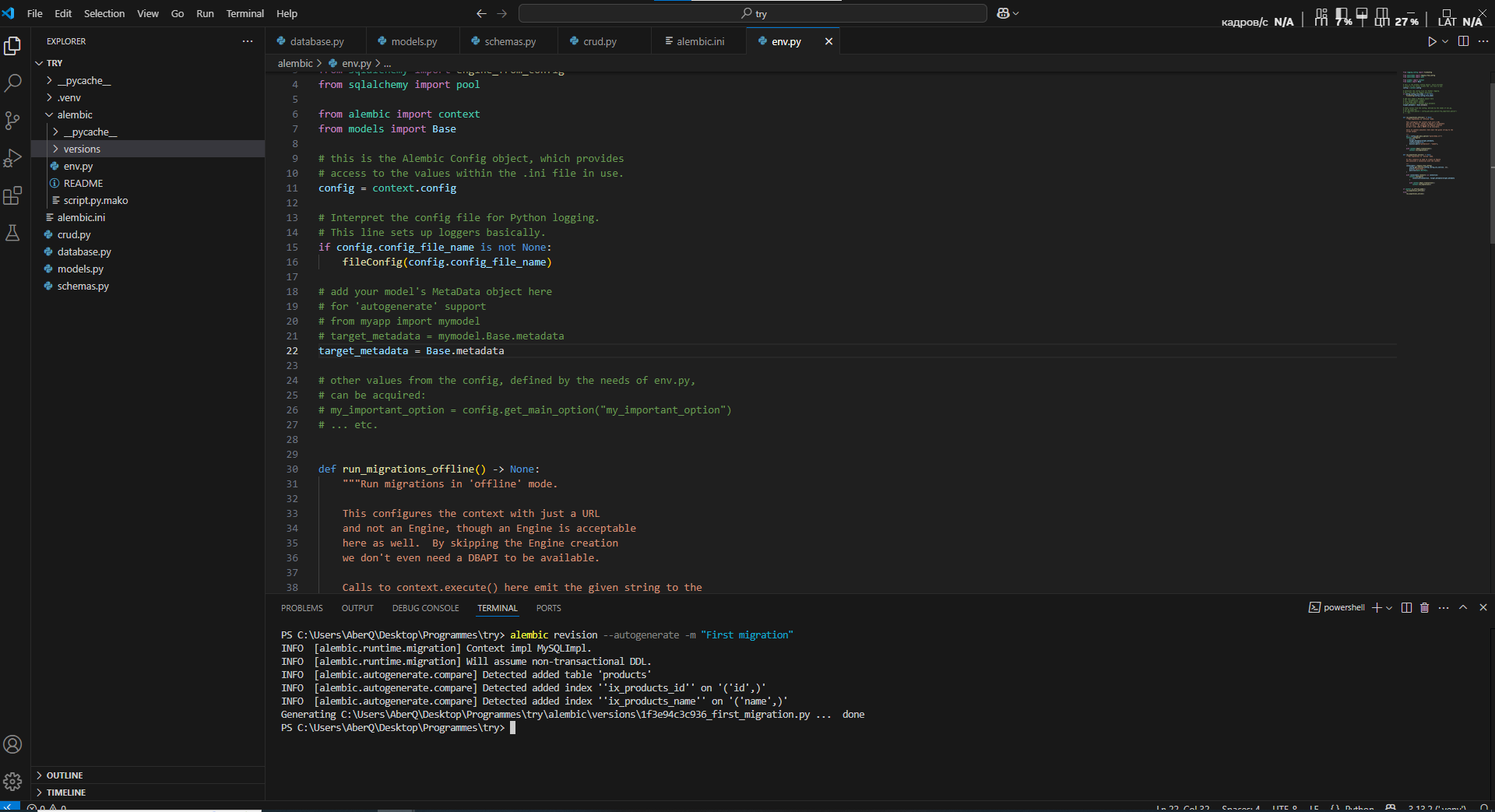
Здесь должна быть указана наша база данных, но как видите стоит None.

Давайте заменим на правильный вариант

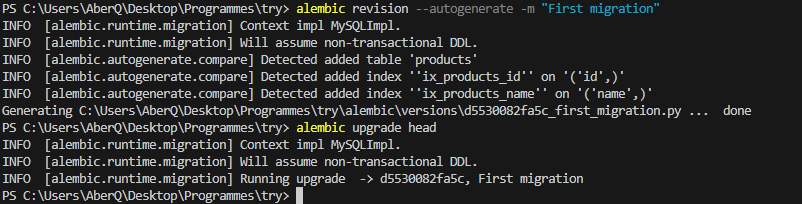


Теперь у нас есть возможность провести миграцию.

Пишем в терминале alembic revision --autogenerate -m "First migration"

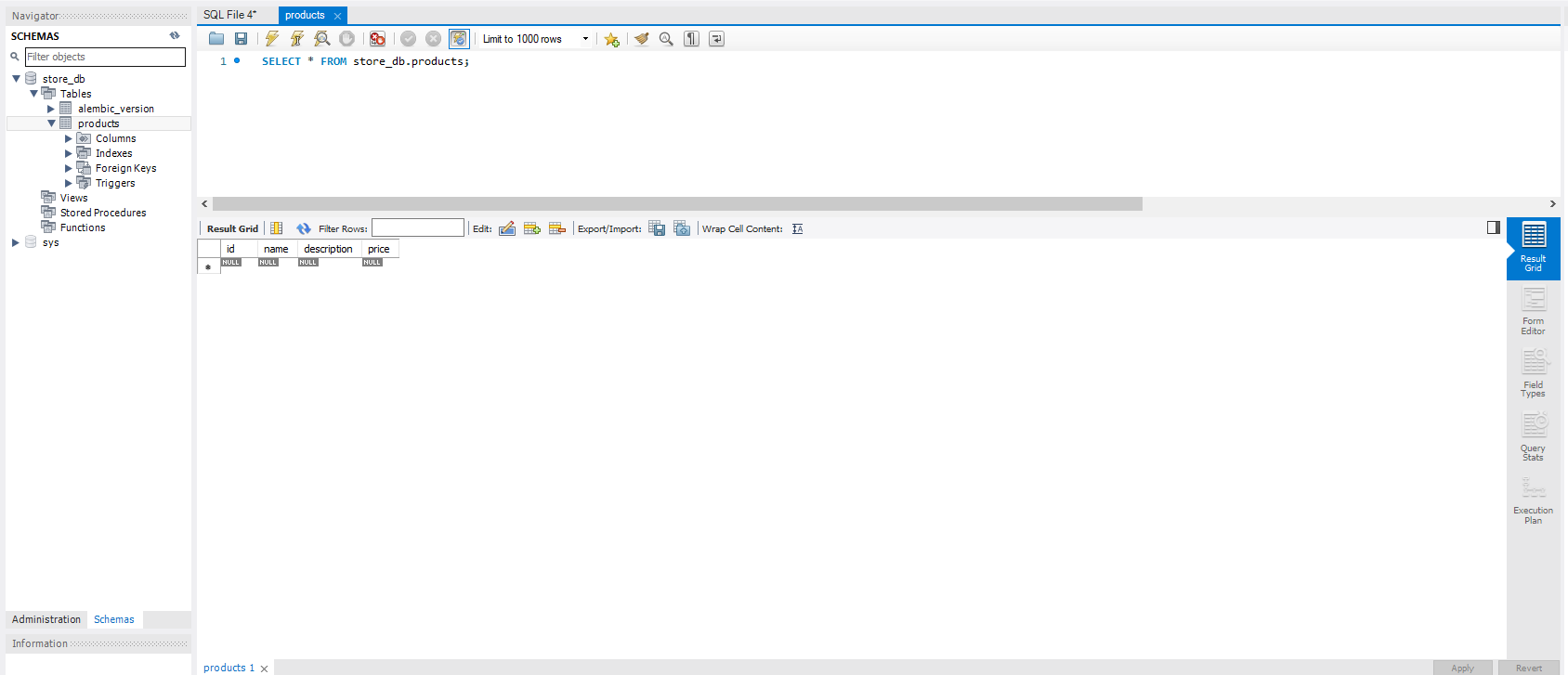


Отлично, миграция успешно подготовилась! Но она еще не вступила в силу, нам нужно для этого отдать команду alembic upgrade head



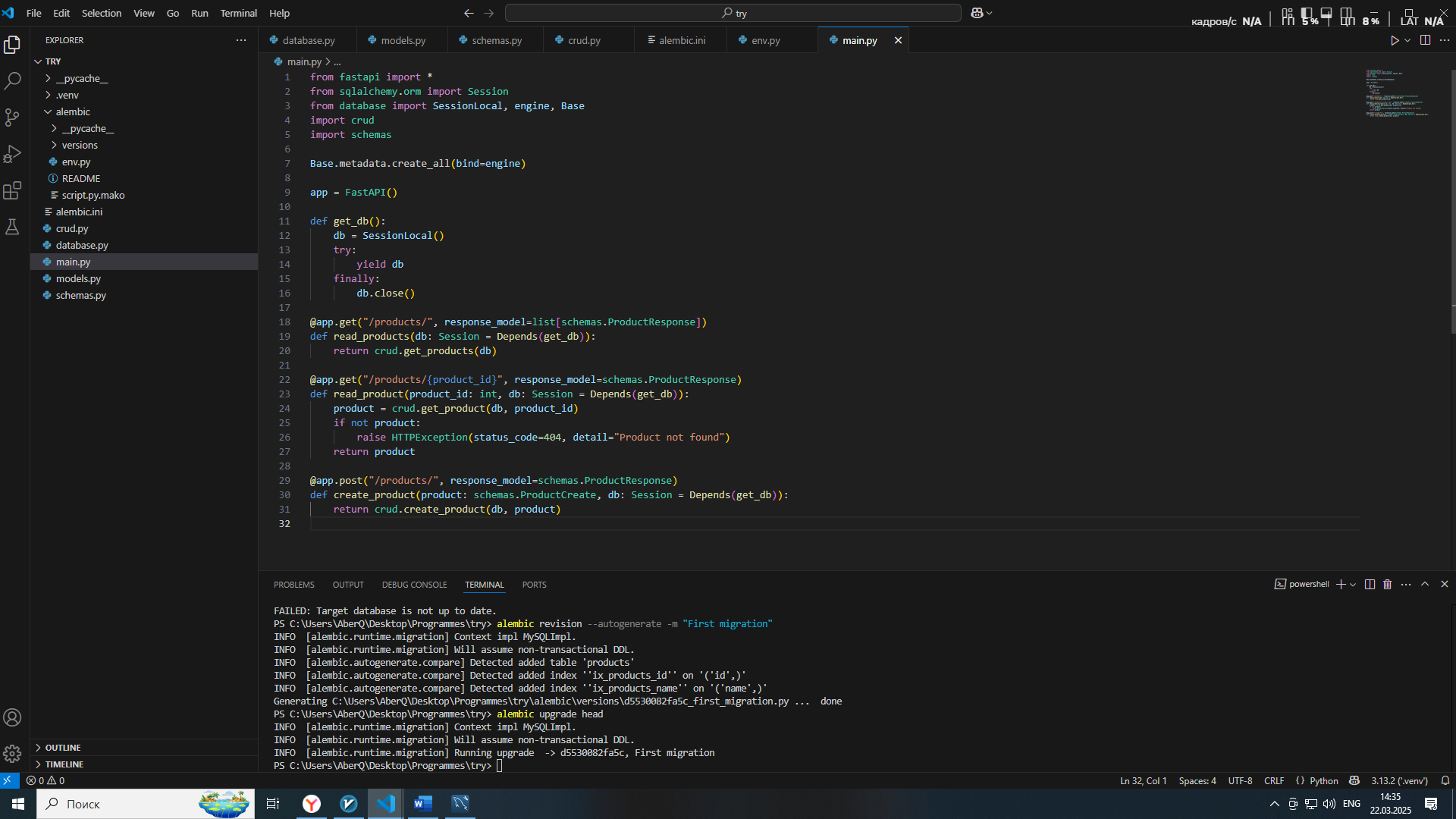
Отлично, все сработало!

Теперь в MySQL мы можем увидеть, что в нашей бд появились столбцы

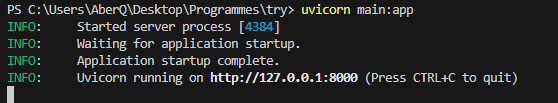


Мы создали основной каркас для нашего веб-приложения, теперь можно переходить к совмещению компонентов и основной реализации.

Давайте создадим файл main.py и пропишем эндпоинты нашего веб-приложения.



Теперь мы можем запустить приложение командой uvicorn main:app

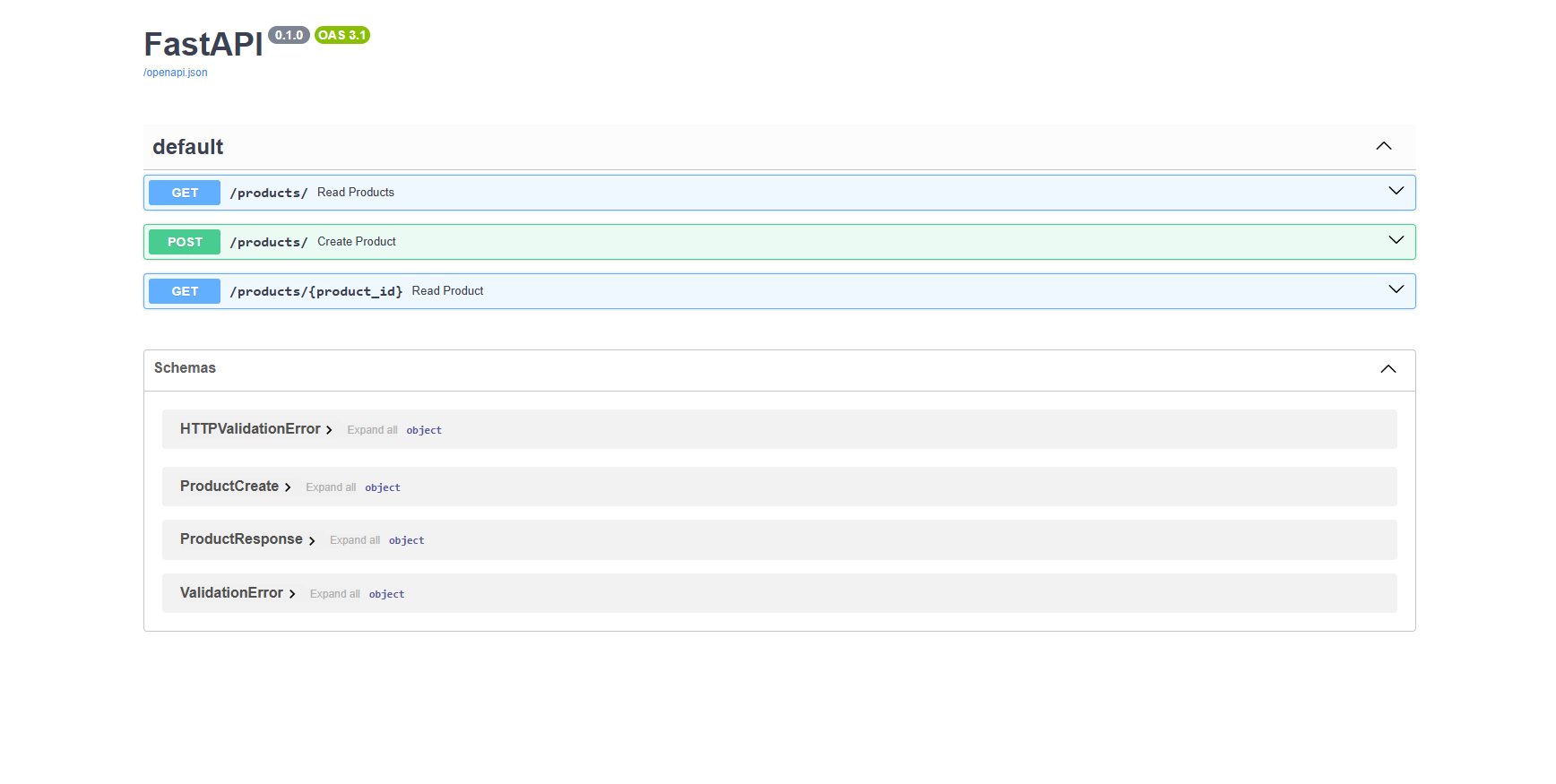


Приложение успешно запущено, давайте посмотрим. Зажмите CTRL и нажмите ЛКМ по адресу.

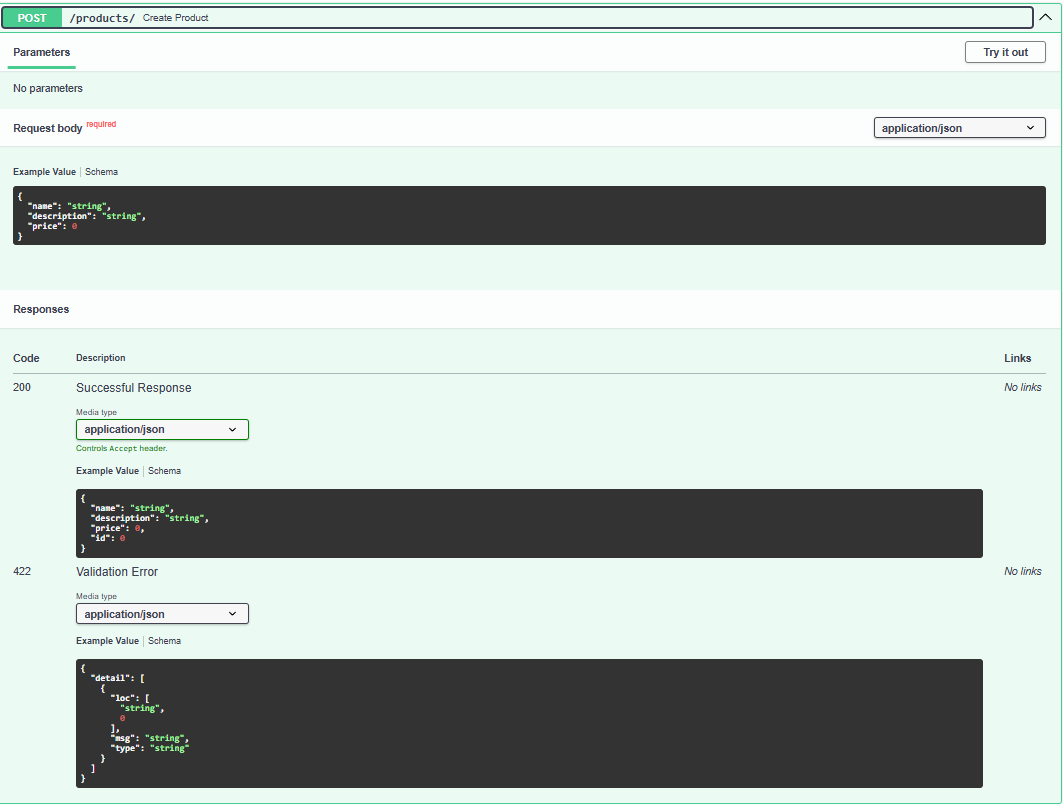


Ничего нет, потому что наши эндпоинты по другим адресам. Давайте попробуем перейти на этот адрес



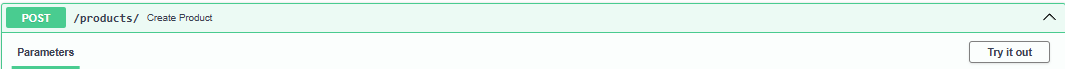


Это автоматически созданная интерактивная Swagger-документация по нашему API. Мощный инструмент позволяющий удобно тестировать свое приложение, а также помогающие другим разработчикам понимать структуру приложения.

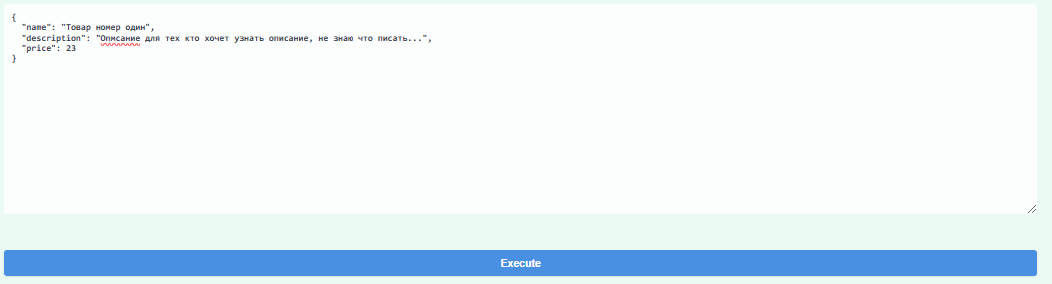
Можно нажать на эндпоинт и узнать какие ответы он дает.

Рекомендую почитать про возможности модификации Swagger-документации, это очень полезно в больших проектах.

Давайте попробуем использовать наше API внутри документации.

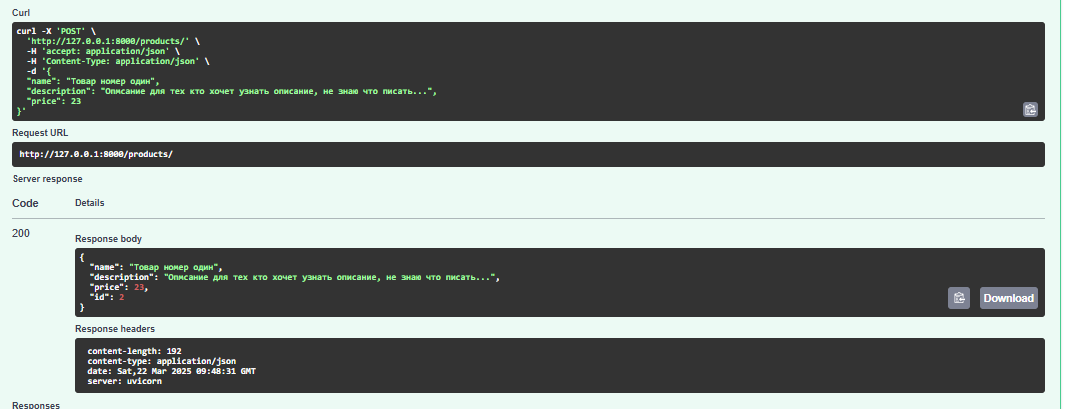


Нажмите кнопку try it out



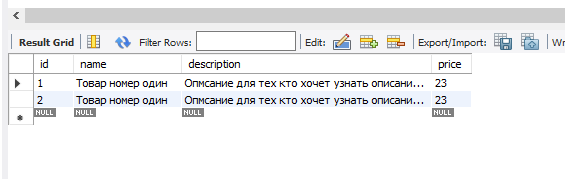
Введите значения, которыми хотите заполнить поля. Нажмите execute.

В результате получите ответ от сервера



Ответ 200 значит успешный, в результате характеристика нашего созданного продукта.

Можем проверить записаны ли объекты в бд.



Как видно, все успешно записано.

Попробуем API с GET-запросом. Все успешно работает!

