**Министерство науки и высшего образования Российской Федерации**

**ФГАОУ ВО «УрФУ имени первого Президента России Б. Н. Ельцина»**

ОТЧЕТ ПО ЗАДАНИЮ 3

Предмет: «основы web-api»

Выполнила:

Студент Катаева А. А.

**ФИО студента**

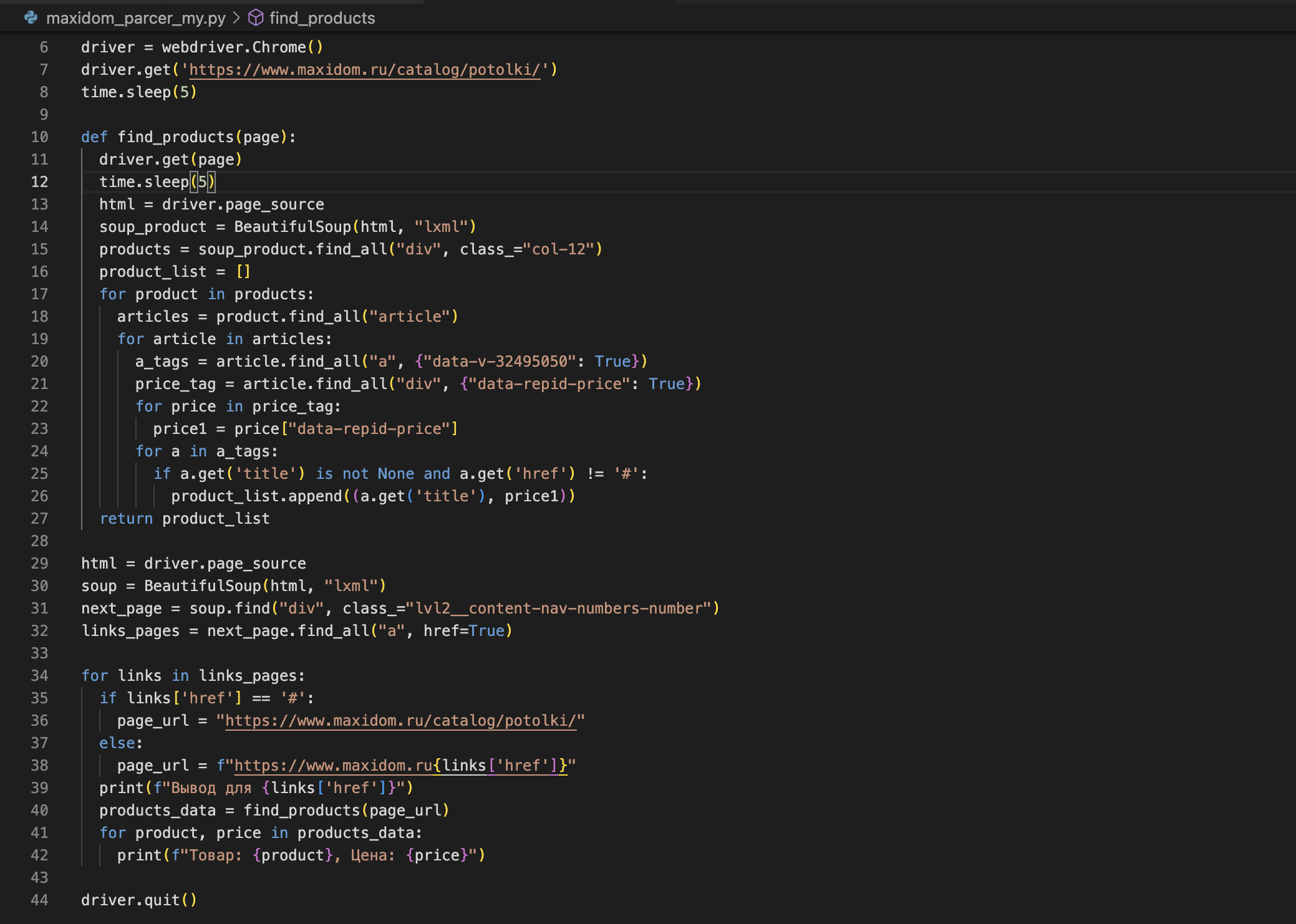
Специальность (направление подготовки) 09.03.01 Информатика и вычислительная техника

Группа РИ-220910

Екатеринбург 2024

Задание: необходимо создать API которое будет фоном парсить данные с сайта и сохранять их в БД. Доступ к данным нужно осуществить через RESTApi с возможность их редактирования и удаления. Все манипуляции с данными должны генерировать уведомления при помощи WebSocket.

Для данного задания был написан код парсера сайта maxidom.ru (см. рис. 1):

Рисунок. 1. Код парсера для одной категории

Данный парсер использует веб-драйвер, так как необходимые для парсинга атрибуты ссылок формируются JS- кодом и с помощью веб-драйвера можно как раз читать JS. Для начала мы запускаем веб-драйвер для главной страницы категории, длаее описываем функцию парсинга для любой страницы, сначала она ищет все div-контейнеры с описанием товара и получается: название товара и соответствующую ему цену, далее (внизу каждой страницы находятся навигационные ссылки) парсер ищет ссылки и добавляет их в список. Далее идет перебор ссылок, к каждой странице применяется функция поиска товаров и ссылок.

Приступаем к написанию API. Для начала опишем модель базы данных и инициализируем ее (см. рис 2):

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описаниеРисунок 2. Файл для базы данных

Определяем URL для подключения к базе данных (SQLALCHEMY\_DATABASE\_URL), далее создаем экземпляр асинхронного движка SQLAlchemy (engine). Создаем экземпляр фабрики сессий для асинхронных сессий (AsyncSessionLocal), параметры sessionmarker:

* Engine # движок
* class\_=AsyncSession # используем асинхронную версию Session
* expire\_on\_commit=False, # Отключение автоматического сохранения изменений при коммите

Определяем базовый класс модели: Base = declarative\_base()

Init\_db() – фсинхронная функция инициализации базы данных:

- engine.begin() создаёт транзакцию, которая начинается с получения соединения.

- conn — это объект соединения, который используется для выполнения операций с базой данных в рамках транзакции.

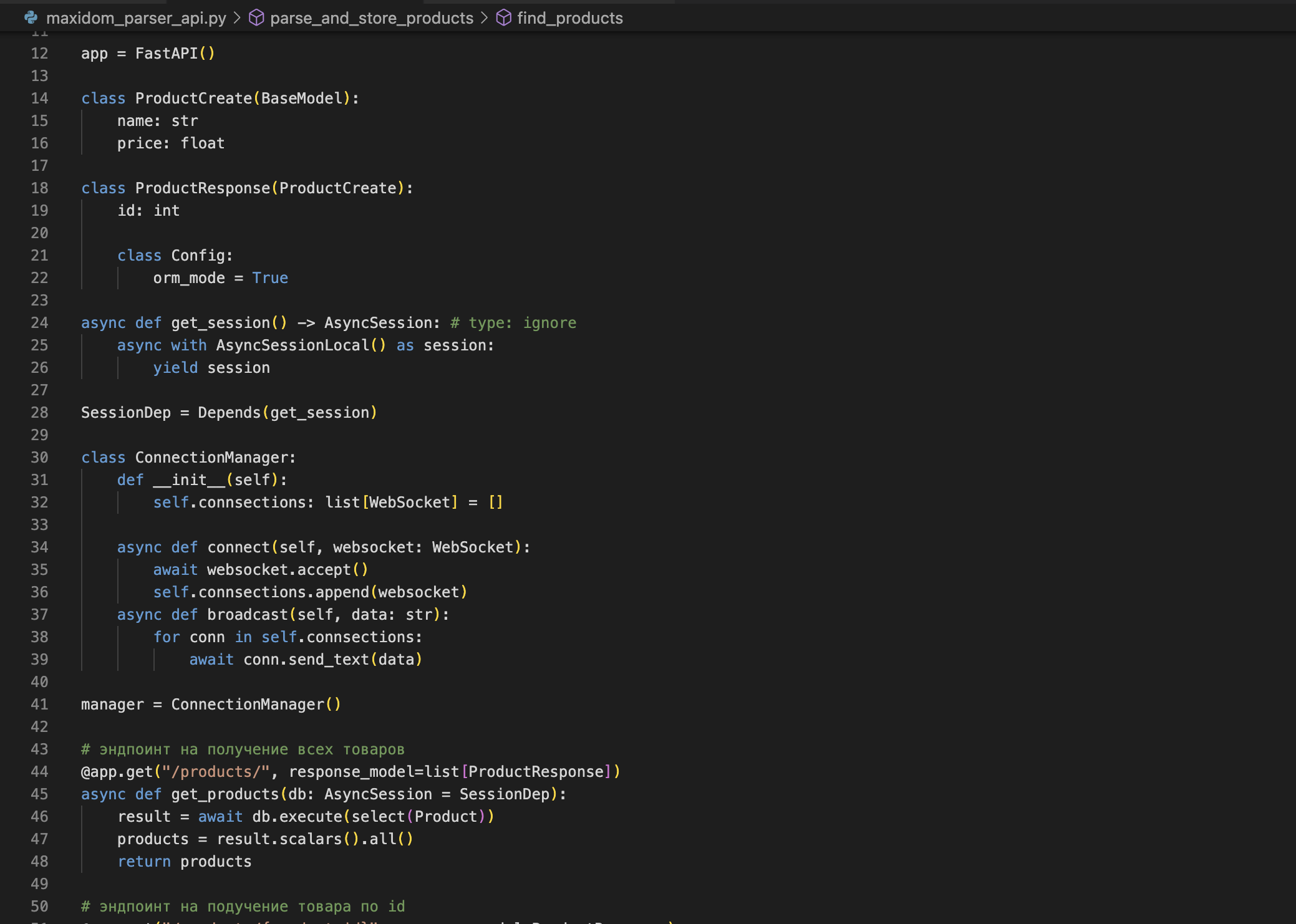
- run\_sync() позволяет выполнять синхронные функции в асинхронном контексте.

- Base.metadata.create\_all() – функция создания всех таблиц в базе данных , которые были определены в базах данных через классы моделей (которые наследуют Base).

close\_db() – функция разрыва соединения.

А в классе Product идет описание схемы таблицы, наследует класс Base.

API с использованием парсера и описанной выше модели базы данных, создано с помощью FastAPI (см. рис. 3):

Рисунок 3. API

app – экземпляр fastAPI();

Ниже следуют классы, для формата получаемых. Config – настраивает Pydantic (используется для валидации данных) для работы с объектно-реляционным отображением: orm\_mode -– автоматически преобразовывает объекты ORM в объекты ProductResponce и наоборот. ProcuctCreacte() – описывает структуру БД.

get\_session() – асинхронная функция для выделения сессии из фабрики сессий.

SessionDep – экземпляр сессии, который создается функцией (get\_session)

Далее описан класс для объекта, который будет хранить все подключения по веб-сокетам:

Connections – список соединений по веб-сокетам;

Connect – функция добавления всех подключений в список;

Websocket.accept – говорим клиенту, что готовы работать с ним, устанавливаем соединение;

Broadcast() – функция, которая обходит все подключения и отправляет каждому уведомление.

Manager – экземпляр класса ConnectionManager.

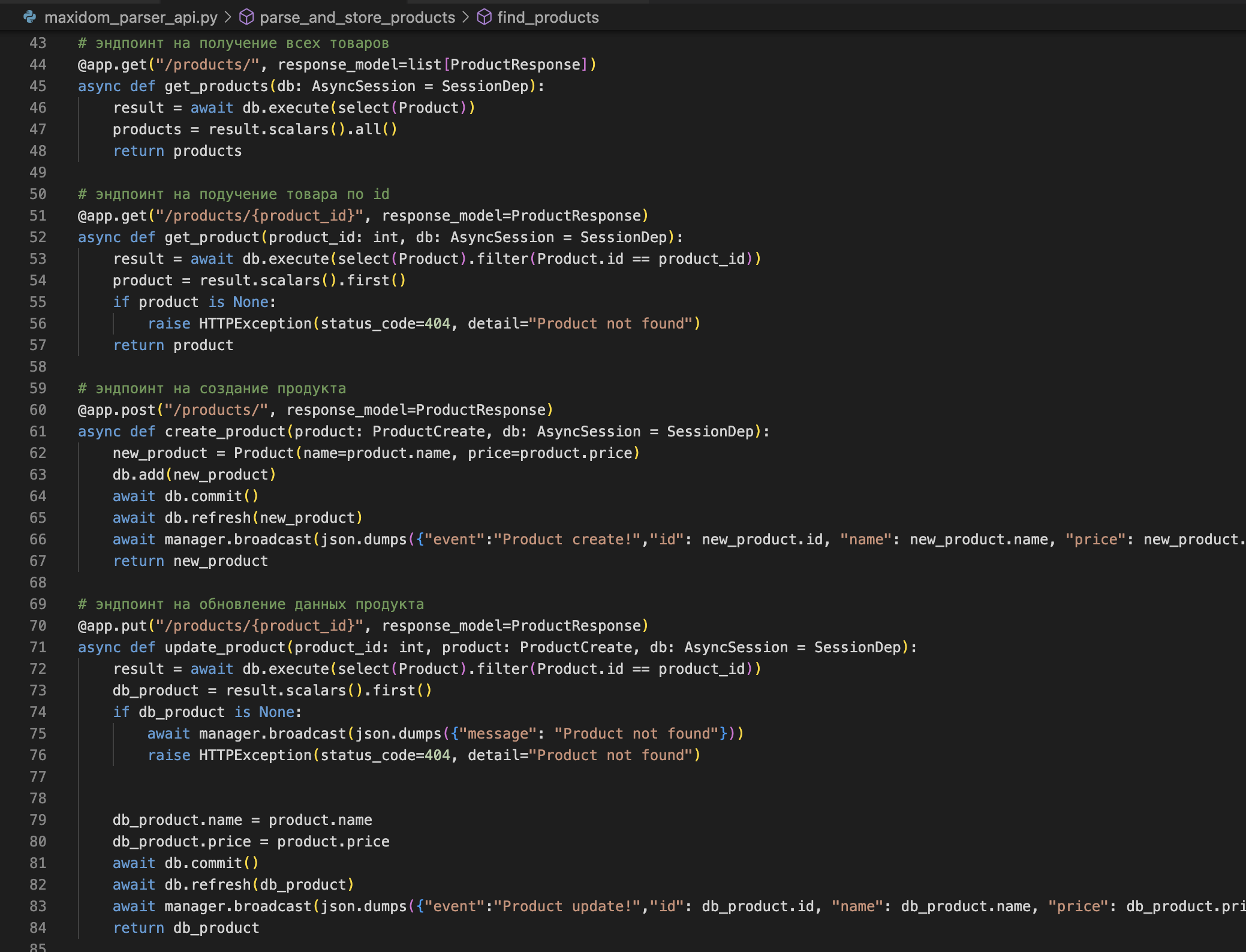
Дальше идет описание эндпоинтов апи, которые обернуты в декларативные функции fastAPI. (см. рис. 4 - 5):

Рисунок 4. Эндпоинты API

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 5. Эндпоинты API

В эндпоинтах, которые не только запрашивают данные, но и отправляют или изменяют их используются оповещения по веб-сокетам с соответствующими messages, преобразованными в json-формат.

Так же в апи описана функция парсера, которая добавляет информацию о полученных товарах в базу данных (см. рис. 6):

Рисунок 6. Функция парсера

Ниже идет функция, которая выполнятеся при запуске апи, и эндпоинта вебсокета (см. рис. 7):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеРисунок 7. Функция при запуске и эндпоинт веб-сокета

Можно заметить, что путь, по которому можно выполнить эндпоинт уже не является http. В данном эндпоинте мы устанавливаем подключение, получаем какое-то сообщение от клиента и выполняем соответсвующие действия: если в сообщении идет запрос продукта по id, то мы отправляем данные о нем (если продукт существует, в ином случае отправляем ошибку: Продукт не найден), если в сообщении нет запроса продукта, то просто отправляем полученное сообщение, длина которого в 10 раз больше. Так же при разрыве соединения будет выводиться сообщение: Client disconnected.

Перейдем к проверке работы апи, запускаем его специальной командой (см. рис. 8):

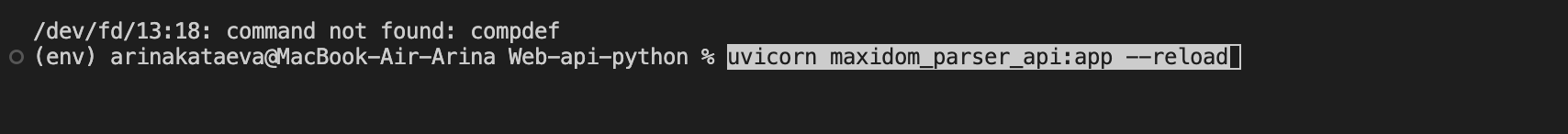


Рисунок 8. Команда запуска веб-апи

При запуске происходит инициализация БД и запускается парсер (см. рис. 9):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9. Подтверждение запуска парсера и создании таблицы

Появляется файл базы данных (см. рис. 10):

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10. Файл БД

Посмотрим содержимое файла БД (см. рис. 11):

Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, Редактирование

Автоматически созданное описаниеРисунок 11. Содержимое базы данных.

Видим, что данные полученные при парсинге страницы успешно добавляются в БД в необходимые столбцы.

Устанавливаем соединение по веб-сокету в приложении Postman (см. рис. 12):

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описаниеРисунок 12. Соединение веб-сокет

Так же загрузим страницу со свагером, чтобы проверить работу апи. Выполняем первый запрос (см. рис. 13):

Изображение выглядит как программное обеспечение, текст, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описаниеРисунок 13. Выполнение первого запроса

Первый запрос выполнился с кодом 200 – успешно. И вернул нам информацию о продуктах в БД. В консоли мы так же видим результат запроса (см. рис. 14):



Переходим ко 2-му запросу (см. рис. 15 - 16):

Изображение выглядит как программное обеспечение, текст, Значок на компьютере, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 15. Выполнение 2 запроса

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 16. Ответ после выполнения 2 запроса

Замечаем, что продукт успешно создан. Проверим сообщение по веб-сокету (см. рис. 17):

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описаниеРисунок 18. Уведомление веб-сокета

Выполняем следующий эндпоинт. Запросим информацию по только что созданному продукту (см. рис. 19):

Изображение выглядит как снимок экрана, текст, программное обеспечение, веб-страница

Автоматически созданное описаниеРисунок 19. Выполнение 3 эндпоинта

Изменим информацию с помощью следующего запроса о созданном товаре (см. рис. 20-21):

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 20. Запрос 4

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

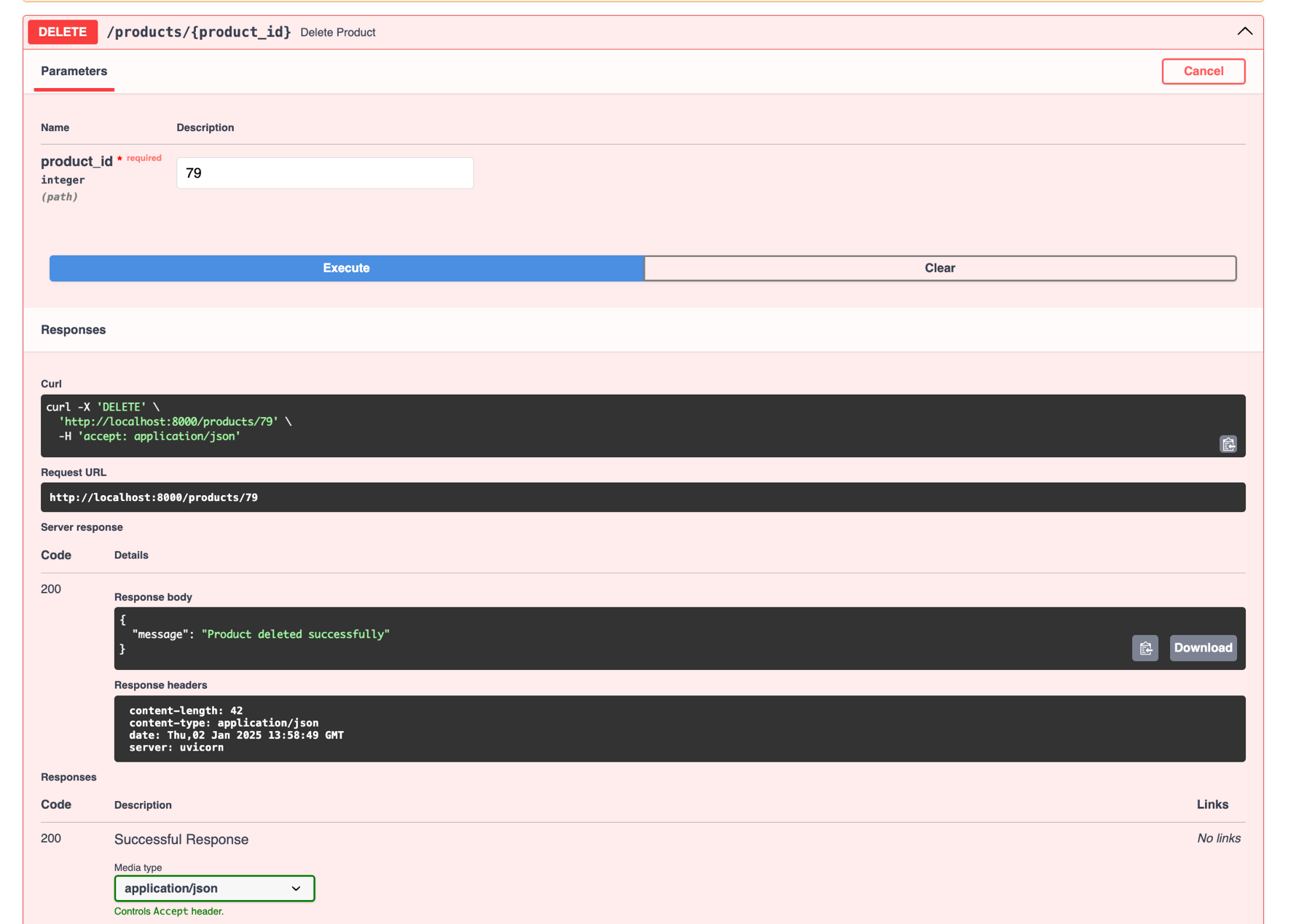
Автоматически созданное описаниеРисунок 21. Ответ 4 запроса

Изображение выглядит как программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение, текст

Автоматически созданное описаниеПроверим уведомления по веб-сокету (см. рис. 22):

Рисунок 22. Уведомления веб-сокета

Удаляем наш созданный продукт с помощью последнего запроса (см. рис. 23):

Рисунок 23. Удаление продукта

Теперь попробуем получить информацию по удаленному продукту, отправив сообщение от клиента (рис. 24):

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описаниеРисунок 24. Запрос отсутствующего продукта

Получили соответствующий ответ. Проверим, что произойдет, если попробовать изменить информацию о несуществующем продукте через сваггер (см. рис. 25-26):

Изображение выглядит как снимок экрана, текст, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 25. Ответ в сваггере

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описаниеРисунок 26. Уведомление в постмане

Таким образом была проверена работа написанного апи с подключенным веб-сокетом. Хорошо просматривается двухстороннее связывание с помощью веб-сокетов между клиентом и сервером. Все эндпоинты корректно работают и база данных успешно заполняется.

Ссылка на github с кодом: https://github.com/arinakataeva007/Web-api-python