|  |  |
| --- | --- |
|  | Министерство образования и науки Российской Федерации  Федеральное государственное автономное образовательное учреждение высшего образования  «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» |

Дополнительное профессиональное образование.

Цифровые кафедры УрФУ.

Курс “DevOps: Виртуализация и облачные вычисления для бизнеса”

Отчет

По итоговому заданию

**Создание парсера сайта**

Студент: Полякова Юлия

г. Екатеринбург

2025г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc187596761)

[ХОД РАБОТЫ 4](#_Toc187596762)

[1. Подготовка и создание базы данных для хранения всех товаров с использованием SQLite и SQLModel 4](#_Toc187596763)

[2. Создание парсера для получения товаров с сайта maxidom.ru с использованием httpx и BeautifulSoup 7](#_Toc187596764)

[3. Создание API с использованием веб-фреймворка FastAPI 10](#_Toc187596765)

[4. Подключение уведомлений при помощи WebSocket 10](#_Toc187596766)

[5. Подключение автоматического парсинга товаров с веб-страницы через заданный URL каждые 12 часов через API 11](#_Toc187596767)

[6. Создание ручного запуска парсинга товаров через API 12](#_Toc187596768)

[7. Добавление возможности изменить URL для автоматического парсинга через API 13](#_Toc187596769)

[8. Управление товарами в базе данных (добавление, удаление, обновление, получение списка товаров, получение одного товара) через API 14](#_Toc187596770)

[ВЫВОД 18](#_Toc187596771)

ВВЕДЕНИЕ

**Цель работы:**

Создание API, которое будет фоном парсить данные с сайта и сохранять их в БД. Доступ к данным осуществляется через REST API с возможностью их добавления, удаления, обновления, получения списка товаров, получение одного товара. Все манипуляции с данными генерируют уведомления при помощи WebSocket.

**Задачи:**

1. Подготовка и создание базы данных для хранения всех товаров с использованием SQLite и SQLModel;
2. Создание парсера для получения товаров с сайта maxidom.ru с использованием httpx и BeautifulSoup;
3. Создание API с использованием веб-фреймворка FastAPI;
4. Подключение уведомлений при помощи WebSocket;
5. Подключение автоматического парсинга товаров с веб-страницы через заданный URL каждые 12 часов через API;
6. Создание ручного запуска парсинга товаров через API;
7. Добавление возможности изменить URL для автоматического парсинга через API;
8. Управление товарами в базе данных (добавление, удаление, обновление, получение списка товаров, получение одного товара) через API.

ХОД РАБОТЫ

## Подготовка и создание базы данных для хранения всех товаров с использованием SQLite и SQLModel

**Описание задачи:**

Необходимо реализовать базу данных для хранения информации о товарах, включая их категории, названия и цены. Для этого использовалась библиотека SQLite в сочетании с SQLModel. Задача включала:

1) Разработку модели данных;

2) Инициализацию базы данных и создание таблицы для хранения товаров;

3) Реализацию функций для проверки существования продукта и сохранения данных.

**Реализованный функционал:**

1) Определение модели данных

Создана модель данных Product, которая соответствует таблице в базе данных. Она включает следующие поля:

id (тип Optional[int]): Автоматически генерируемый первичный ключ.

category (тип str): Категория продукта.

name (тип str): Название продукта.

price (тип float): Цена продукта.

Модель определена с использованием класса SQLModel, что обеспечивает интеграцию с ORM и автоматическое создание таблиц на основе схемы данных (Рисунок 1).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 1 – Модель данных.

2) Инициализация базы данных

Для работы с базой данных был создан файл products.db, подключение к которому осуществляется через SQLAlchemy Engine (Рисунок 2).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 2 – Создание файла .db и объекта для подключения к БД.

Функция create\_table выполняет создание таблицы, если она еще не существует, используя метаданные модели Product (Рисунок 3).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Создание таблицы.

3) Проверка существования записи

Функция product\_exists реализована для проверки наличия продукта в базе данных (Рисунок 4). Логика основана на SQL-запросе с фильтрацией по полям category, name и price.

Возвращает True, если продукт существует, иначе False.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 4 – Функция для проверки на наличие товара в БД.

4) Сохранение списка продуктов

Функция save\_product выполняет сохранение нескольких продуктов в базу данных одной транзакцией. Реализован механизм обработки ошибок: в случае попытки добавить дублирующиеся записи, транзакция откатывается, и выводится сообщение об ошибке (Рисунок 5).

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

Автоматически созданное описание

Рисунок 5 – Функция для сохранения товаров.

## Создание парсера для получения товаров с сайта maxidom.ru с использованием httpx и BeautifulSoup

**Описание задачи:**

Целью задачи было разработать асинхронный парсер для получения данных о товарах с веб-сайта maxidom.ru. Извлеченная информация сохраняется в базе данных. Задача включала:

1) Извлечение информации о категории, названии и цене товара;

2) Проверку на наличие дубликатов в базе перед сохранением;

3) Автоматический переход на следующую страницу, если она существует.

**Реализованный функционал:**

1) Асинхронное выполнение запросов

Использована библиотека httpx для асинхронной работы с HTTP-запросами, что ускоряет процесс парсинга, особенно при обработке нескольких страниц (Рисунок 6).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 6 – Асинхронные HTTP-запросы к веб-странице.

2) Парсинг данных с HTML-страницы

Для анализа и извлечения данных из HTML использован BeautifulSoup. Первым действием было создание объекта BeautifulSoup для дальнейшей работы (Рисунок 7).



Рисунок 7 – Создание объекта BeautifulSoup.

Следующие действия:

1) Извлечение категории товара через тег span[itemprop="name"] (Рисунок 8);



Рисунок 8 – Извлечение категории товаров.

2) Извлечение названия и цены каждого товара из тегов article.l-product, span[itemprop="name"], и span[itemprop="price"] (Рисунок 9).

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

Автоматически созданное описание

Рисунок 9 – Извлечение названий и цен товаров.

3) Проверка на дублирование

Перед добавлением каждого товара в базу данных выполняется проверка, существует ли запись с аналогичной категорией, названием и ценой. Это предотвращает дублирование данных (Рисунок 10).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Проверка на дублирование перед добавлением товара в БД.

4) Сохранение данных в БД

Для сохранения данных использована функция save\_product, которая добавляет продукты в базу данных одной транзакцией (Рисунок 11).



Рисунок 11 – Сохранение всех товаров в БД.

5) Обработка пагинации

Если на текущей странице существует ссылка на следующую, парсер автоматически переходит по ней для продолжения извлечения данных. Если ссылки нет, процесс парсинга завершается (Рисунок 12).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 12 – Проверка на пагинацию.

## Создание API с использованием веб-фреймворка FastAPI

Инициализация FastAPI-приложения и глобального URL для парсинга (Рисунок 13).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 13 – Инициализация.

## Подключение уведомлений при помощи WebSocket

Для уведомлений пользователей при взаимодействии с API реализован WebSocket.

Клиенты подключаются к эндпоинту /ws (Рисунок 14). Менеджер соединений (ConnectionManager) управляет подключением и отключением клиентов (Рисунок 15). Уведомления рассылаются через метод manager.broadcast (Рисунок 16).

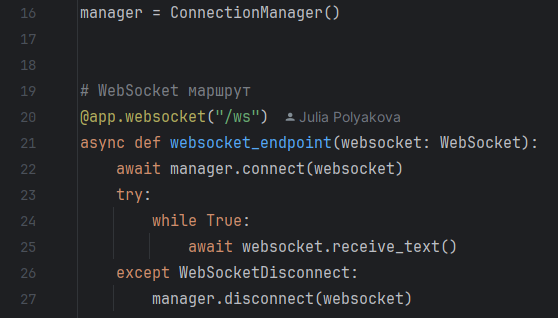


Рисунок 14 – Эндпоинт для подключения к WebSocket.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 15 – Менеджер соединений WebSocket.



Рисунок 16 – Пример уведомления.

## Подключение автоматического парсинга товаров с веб-страницы через заданный URL каждые 12 часов через API

Для реализации автоматического парсинга была создана фоновая задача run\_periodic\_parsing, которая запускается при старте приложения благодаря декоратору @app.on\_event("startup") (Рисунок 17).

Эта задача выполняет следующие действия:

1) Использует глобальную переменную current\_url как источник для парсинга данных;

2) Запускает функцию parse\_and\_save\_products, которая парсит товары с текущего URL и сохраняет их в базу данных;

3) После завершения цикла парсинга делает паузу на 12 часов перед следующим запуском.

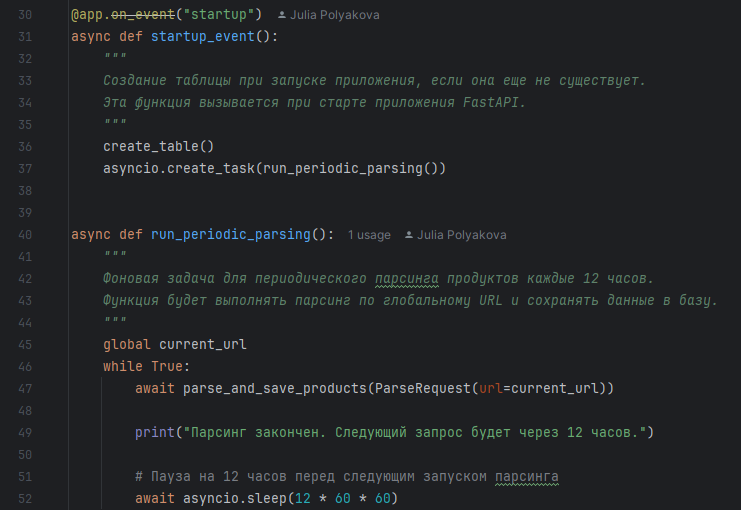


Рисунок 17 – Автоматический парсинг веб-страниц.

## Создание ручного запуска парсинга товаров через API

Для удобства добавлена возможность ручного запуска парсинга через эндпоинт /parse. Эндпоинт принимает объект ParseRequest (Рисунок 18), содержащий URL для парсинга. Внутри используется функция parser\_products, которая парсит товары с указанного URL и сохраняет их в базу данных (Рисунок 19).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 18 – Объект ParseRequest.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 19 – Эндпоинт для запуска парсера.

## Добавление возможности изменить URL для автоматического парсинга через API

Добавлен эндпоинт /set\_url, позволяющий менять URL для автоматического парсинга. Новый URL сохраняется в глобальной переменной current\_url, которая используется в фоновой задаче для парсинга (Рисунок 20).

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

Автоматически созданное описание

Рисунок 20 – Эндпоинт для смены глобального URL.

## Управление товарами в базе данных (добавление, удаление, обновление, получение списка товаров, получение одного товара) через API

1) Получение списка всех товаров:

Эндпоинт /products позволяет получить список всех товаров из базы данных. После получения данных пользователи, подключенные через WebSocket, получают уведомление о запросе (Рисунок 21).

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

Автоматически созданное описание

Рисунок 21 – Эндпоинт для получения всех товаров и проверка WebSocket.

2) Получение одного товара:

Эндпоинт /products/{product\_id} позволяет получить товар по его id из базы данных. После получения данных пользователи, подключенные через WebSocket, получают уведомление о запросе (Рисунок 22).

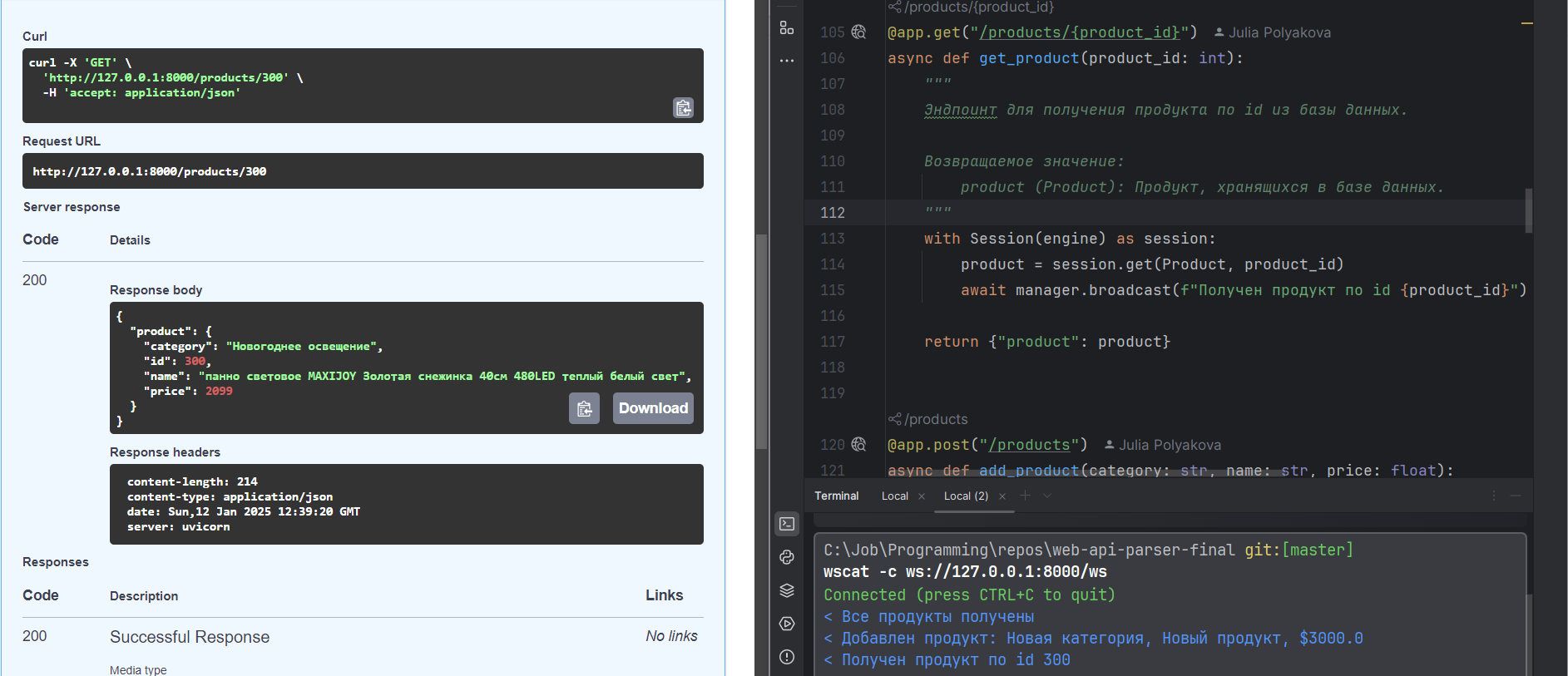


Рисунок 22 – Эндпоинт для получения одного товара и проверка WebSocket.

3) Добавление нового товара:

Эндпоинт /products принимает категорию, название и цену нового товара. Перед добавлением проверяет, существует ли продукт в базе. После успешного добавления отправляется уведомление через WebSocket (Рисунок 23).

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

Автоматически созданное описание

Рисунок 23 – Эндпоинт для добавления нового продукта и проверка WebSocket.

4) Обновление существующего товара:

Эндпоинт /products/{product\_id} позволяет обновлять свойства товара: категорию, название и цену. После обновления отправляется уведомление о редактировании всем клиентам WebSocket (Рисунок 24).

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

Автоматически созданное описание

Рисунок 24 – Эндпоинт для изменения продукта и проверка WebSocket.

5) Удаление товара:

Эндпоинт /products/{product\_id} удаляет товар по его ID. Если товар не найден, возвращается ошибка 404. После удаления отправляется уведомление (Рисунок 25).

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

Автоматически созданное описание

Рисунок 25 – Эндпоинт для удаления товара и проверка WebSocket.

ВЫВОД

В результате выполнения данной работы была создана полноценная система для парсинга данных с сайта, хранения их в базе данных и предоставления доступа через REST API. Реализация выполнена с использованием современных технологий, таких как FastAPI, SQLModel и WebSocket, что позволило создать масштабируемое, удобное и функциональное решение.

API поддерживает автоматический парсинг товаров каждые 12 часов, ручной запуск парсинга, управление URL для парсинга и все необходимые CRUD-операции с данными в БД. Встроенные уведомления через WebSocket обеспечивают прозрачность взаимодействия и позволяют клиентам в реальном времени получать информацию о выполненных операциях.