Министерство науки и высшего образования Российской Федерации
ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»
Кафедра «школа бакалавриата (школа)»

Отчет по итоговому заданию курса «Основы Web API» на тему

Разработка асинхронного backend-сервиса на FastAPI для работы с курсами валют

Преподаватель Свинцов Дмитрий Владимирович

Студент Швенк Милена Витальевна

Специальность (направление подготовки) 09.03.01 Информатика и вычислительная техника

Группа РИ-330913

Екатеринбург 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью выполненной работы являлось создание полноценного асинхронного backend-сервиса на базе FastAPI, способного обрабатывать данные валют в реальном времени. Главная задача заключалась в том, чтобы объединить несколько современных технологий: REST API, WebSocket, фоновую задачу с регулярным обновлением данных, асинхронную работу с базой данных через SQLAlchemy/SQLModel, а также интеграцию с брокером сообщений NATS.

В современном веб-разработке крайне важна высокая скорость и масштабируемость серверных приложений. Асинхронный подход позволяет обрабатывать множество запросов одновременно, не блокируя основной поток выполнения программы. Это особенно важно при работе с внешними API, такими как сервис Центробанка России, откуда берутся курсы валют.

В данной работе реализован сервис, который выполняет следующие ключевые функции: хранение данных о валютах в базе данных SQLite, предоставление REST API для создания, чтения, обновления и удаления записей, информирование клиентов в реальном времени через WebSocket и автоматическое обновление данных через фоновую задачу. Кроме того, реализована интеграция с NATS, что позволяет публиковать и получать события в отдельном канале.

1 REST API

Основная часть проекта реализована с помощью REST API, что обеспечивает стандартный способ взаимодействия клиента с сервером. Были реализованы следующие конечные точки (эндпоинты):

- GET /currencies — получение списка всех валют;
- GET /currencies/{id} — получение конкретной валюты по идентификатору;
- POST /currencies — создание новой записи;
- PATCH /currencies/{id} — обновление существующей записи;
- DELETE /currencies/{id} — удаление записи;
- POST /tasks/run — принудительный запуск фоновой задачи обновления валют.

Все маршруты построены с использованием асинхронного подхода через AsyncSession SQLAlchemy, что позволяет эффективно работать с базой данных без блокировки приложения. Для передачи данных клиенту используется схема CurrencyRead, которая конвертирует объекты модели в удобный формат с датами в строковом виде.

Особое внимание было уделено обработке ошибок: при попытке получить или изменить несуществующую валюту возвращается HTTP-ошибка 404, что соответствует REST-стандартам. Также реализована отправка уведомлений в WebSocket и публикация событий в NATS при создании, обновлении или удалении записей, что обеспечивает синхронизацию данных между различными частями системы в реальном времени.

2 WebSocket

Для реализации уведомлений в реальном времени был создан WebSocket-сервер по маршруту /ws. Его основная задача — уведомлять подключенные клиенты о событиях, происходящих на сервере, таких как создание, изменение или удаление валют, а также поступление новых данных, загружаемых фоновой задачей.

Клиенты подключаются к серверу по WebSocket-соединению и получают сообщения в формате JSON, содержащие информацию о типе события и актуальные данные. WebSocket реализован через класс WSManager, который хранит список активных соединений и умеет рассылать сообщения всем клиентам одновременно. При возникновении ошибок соединение корректно закрывается и удаляется из списка активных клиентов.

Проверка работы WebSocket выполнялась с помощью инструмента Postman. Для этого в Postman создаётся новый запрос типа WebSocket Request, после чего указывается адрес подключения ws://127.0.0.1:8000/ws. После успешного соединения клиент начинает получать сообщения автоматически при выполнении операций через REST API (создание, обновление или удаление валют), а также при срабатывании фоновой задачи обновления курсов. Таким образом, можно наглядно убедиться, что сервер корректно рассылает уведомления в реальном времени.

Использование WebSocket позволяет поддерживать актуальность данных на клиентской стороне без необходимости периодического опроса сервера. Это значительно снижает сетевую нагрузку и повышает отзывчивость интерфейса. Например, при изменении курса валют пользователи моментально получают обновлённые значения без перезагрузки страницы.

3 Фоновая задача и интеграция с внешним API

Фоновая задача является центральным элементом проекта, обеспечивая регулярное обновление данных валют. Она реализована в виде асинхронной функции update_currencies_periodically, которая запускается автоматически при старте приложения и может быть запущена вручную через POST /tasks/run.

Задача выполняет HTTP-запрос к внешнему API Центробанка России через библиотеку httpx, получает актуальные курсы валют и обновляет данные в локальной базе SQLite. Если курс валюты изменился, производится обновление записи с фиксацией времени изменения (updated_at). В случае добавления новой валюты создается новая запись.

После обновления данных фоновой задачей, сервис публикует сообщение в канал NATS и рассылает уведомление через WebSocket. Такой подход обеспечивает синхронизацию всех клиентов и других сервисов, подписанных на канал. Задержка между обновлениями составляет 120 секунд, но при необходимости её можно изменить.

4 Интеграция с NATS

Интеграция с брокером сообщений NATS была реализована для публикации и подписки на события обновления валют. Сервис подключается к NATS при старте приложения и подписывается на канал currencies.updates, что позволяет получать сообщения как от собственной фоновой задачи, так и от внешних источников.

При публикации событий (создание, изменение и удаление валют, а также обновление данных фоновой задачей) сообщения отправляются в указанный канал и одновременно рассылаются всем подключённым клиентам через WebSocket.

Дополнительно реализована обработка входящих сообщений из NATS: при получении внешнего события сервер логирует сообщение и передаёт его клиентам через WebSocket. На текущем этапе входящие события используются исключительно для логирования и уведомления клиентов, без внесения изменений в базу данных.

Корректная работа интеграции с NATS проверялась несколькими способами. Во-первых, сообщения отслеживались в терминале, где запущен сервер uvicorn, что позволяло убедиться в успешном получении событий. Во-вторых, уведомления отображались в Postman при активном WebSocket-подключении. Кроме того, для тестирования использовались отдельные терминалы с утилитой NATS:

- команда подписки на канал: ./nats.exe sub currencies.updates;
- команда ручной отправки сообщения: ./nats.exe pub currencies.updates '{"event":"rates_update","rates":{"USD":80.0,"EUR":90.0}}'.

При подписке на канал в отдельном терминале были видны все сообщения, публикуемые в NATS. В терминале отправки подтверждался факт успешной публикации сообщения, а в терминале с запущенным uvicorn отображалось полученное событие. Это же сообщение одновременно передавалось

подключённым клиентам и отображалось в Postman через WebSocket. Таким образом, было подтверждено, что сообщения корректно проходят полный цикл доставки: от публикации в NATS до получения сервером и рассылки клиентам, при этом данные, отправленные из вне только, логируются и не добавляются в базу данных.

5 Асинхронная работа с базой данных

Для хранения данных о валютах используется SQLite через SQLModel и SQLAlchemy в асинхронном режиме. Асинхронные сессии позволяют обрабатывать запросы к базе данных без блокировки основного потока, что особенно важно при множественных параллельных запросах и при работе фоновых задач.

Каждая таблица в базе данных автоматически создается при старте приложения через команду SQLModel.metadata.create_all. Модель Currency хранит основные поля: id, currency, rate, created_at и updated_at. Поля с датой создаются с помощью datetime.utcnow() и автоматически обновляются при изменении записи.

Такой подход позволяет обеспечивать целостность данных и историю изменений, а также легко интегрировать базу данных с другими компонентами системы, такими как WebSocket и NATS.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы был создан полнофункциональный асинхронный backend-сервис, который сочетает в себе REST API, WebSocket, фоновую задачу с обновлением данных и интеграцию с NATS.

Сервис обеспечивает:

- эффективное асинхронное взаимодействие с базой данных;
- возможность мгновенной рассылки обновлений клиентам через WebSocket;
- регулярное обновление курсов валют через внешний API;
- синхронизацию с другими сервисами через брокер сообщений NATS;
- возможность ручного запуска фоновой задачи.

В процессе выполнения задания были получены практические навыки разработки асинхронных приложений на FastAPI, работы с WebSocket-соединениями, реализации фоновых задач и интеграции с брокером сообщений NATS. Также были закреплены знания по асинхронной работе с базой данных и построению событийно-ориентированной архитектуры backend-сервисов.

ПРИЛОЖЕНИЕ А

(обязательное)

Ссылка на репозиторий

GitHub: MilenaShvenk/Currency-API

ПРИЛОЖЕНИЕ Б

(обязательное)

Скриншоты проверки работы сервиса

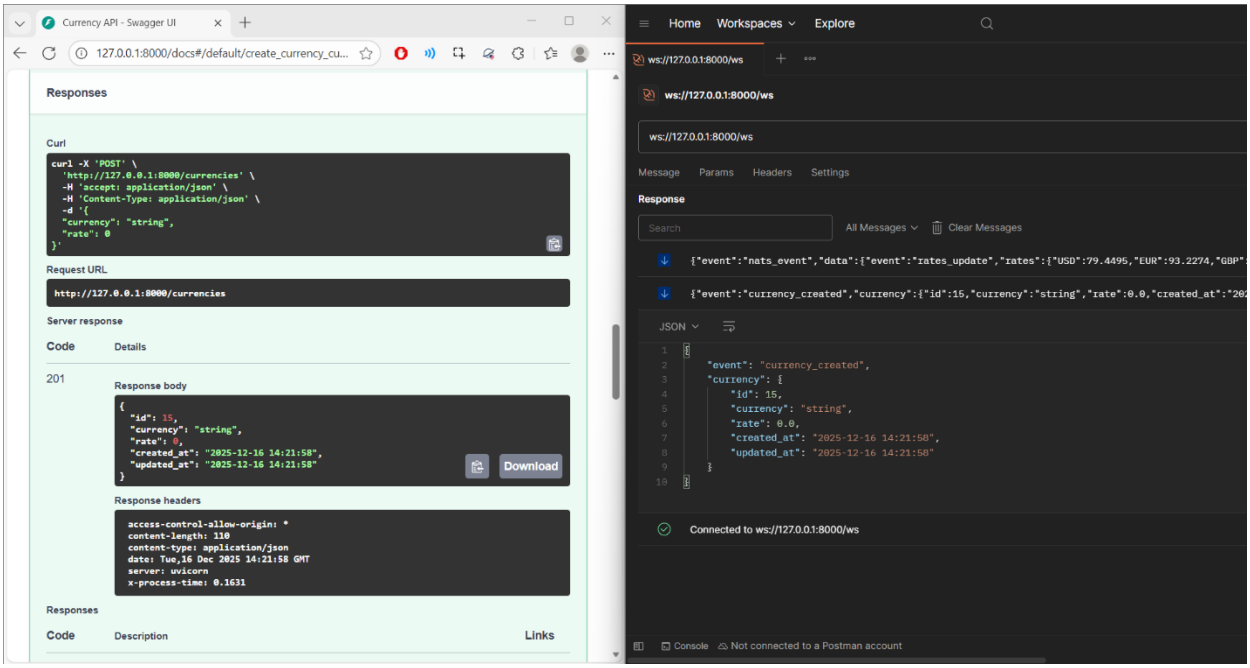

Рисунок 1 – Создание новой записи

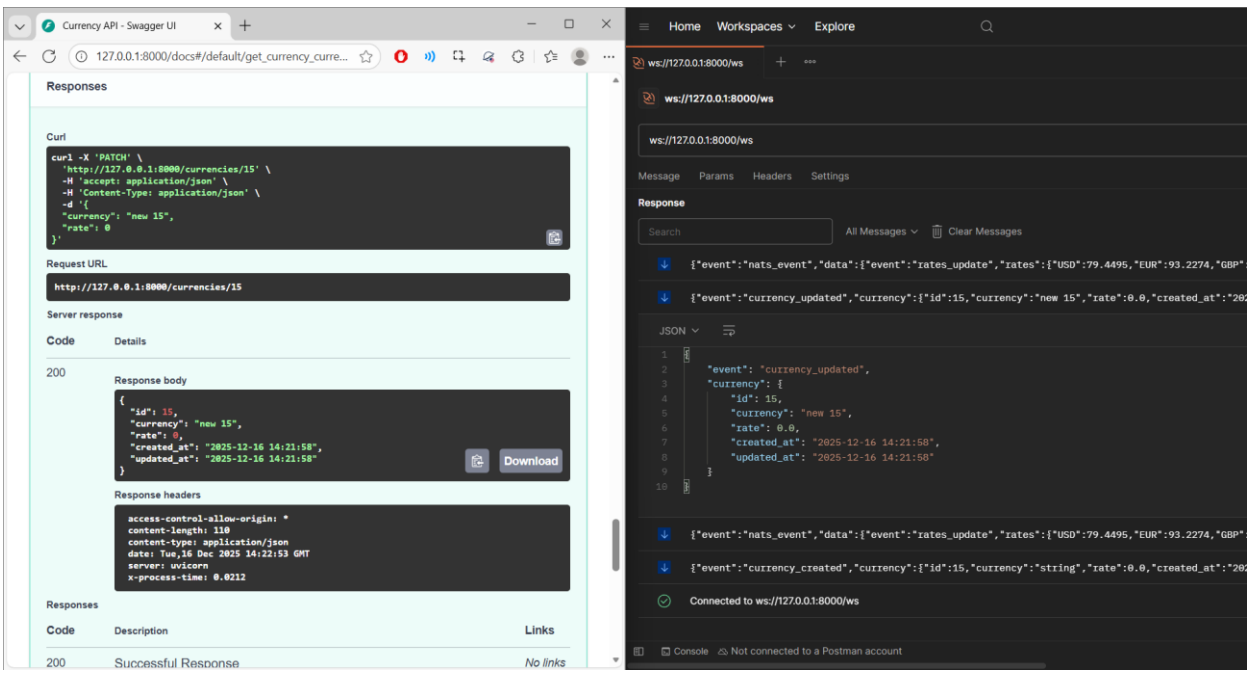

Рисунок 2 – Обновление существующей записи

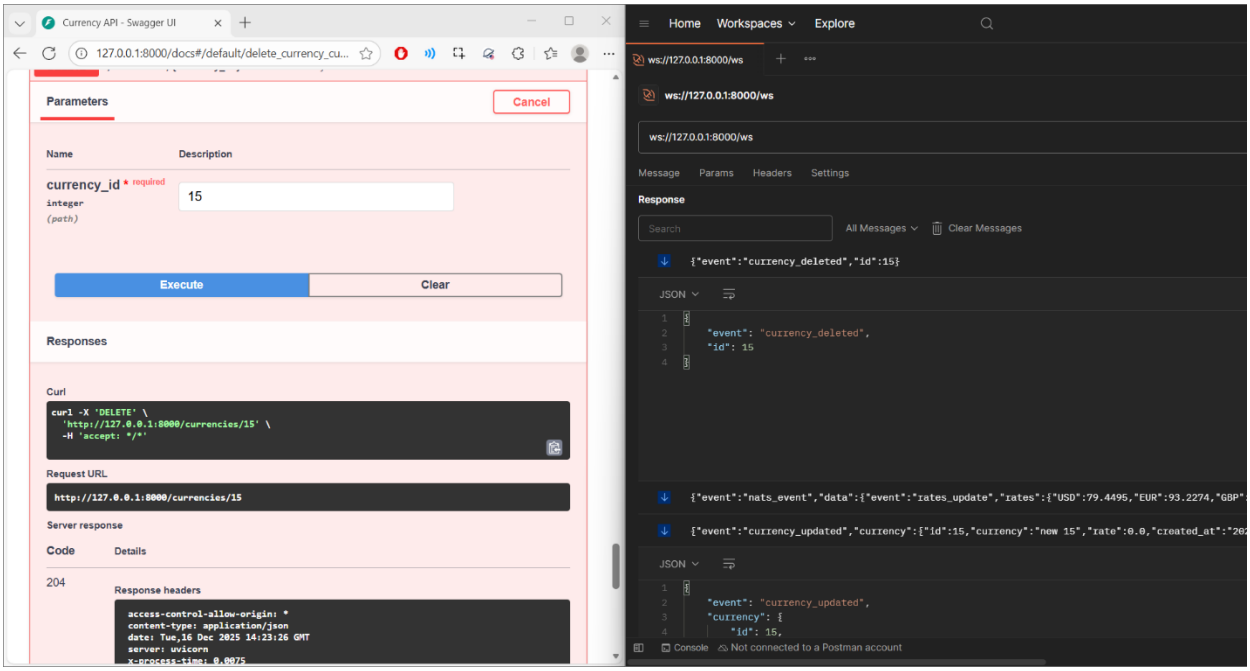

Рисунок 3 – Удаление записи

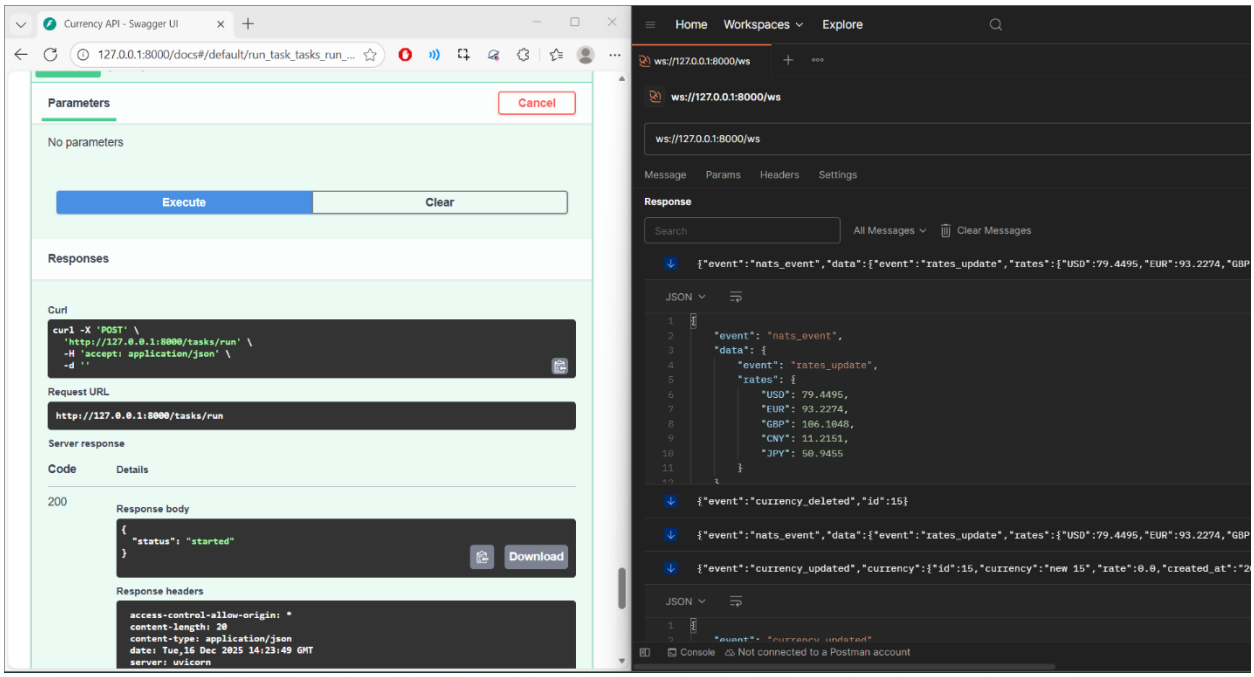

Рисунок 4 – Принудительный запуск фоновой задачи обновления валют