Async/Await Demo: API + Cache + DB

This project is a teaching demo that shows what `async`/`await` buys you in Python.

It starts 3 fake servers in one process: - API server (`127.0.0.1:9000`) - Cache server (`127.0.0.1:9001`) - Database server (`127.0.0.1:9002`)

The API receives a client request and needs several pieces of data: - user profile (cache, then DB fallback) - product info (cache, then DB fallback) - inventory (DB) - recommendations (DB)

It does this in two ways: 1. `sequential`: await each call one by one 2. `concurrent`: start independent calls together with `asyncio.create_task(...)` and `await asyncio.gather(...)`

The script runs both modes and prints timing so the speed difference is obvious.

Run

```
uv run demo_async_await.py
```

What to watch for

- Timestamped log lines from `API`, `CACHE`, and `DB`
- In sequential mode, DB calls happen one after another
- In concurrent mode, calls overlap and complete sooner
- Same final data, lower total latency