Programowanie zaawansowane

Mateusz Piróg 152773

Zadanie 5

Flask

Flask to mikroframework webowy napisany w Pythonie, stworzony z myślą o prostocie, elastyczności i rozszerzalności. Jego architektura opiera się na zasadzie "micro", co oznacza, że nie narzuca struktury projektu ani nie zawiera zbędnych komponentów, pozostawiając deweloperowi pełną kontrolę nad architekturą aplikacji. Flask jest szeroko wykorzystywany zarówno przez początkujących, jak i profesjonalistów w małych oraz średnich projektach webowych, REST API czy serwisach wewnętrznych.

Najważniejsze cechy:

- Minimalistyczne i intuicyjne API.
- Wbudowany serwer deweloperski i debugger.
- Obsługa routingu URL, szablonów (Jinja2) oraz żądań HTTP.
- Możliwość łatwej integracji z bazami danych i bibliotekami front-endowymi.
- Bogaty ekosystem rozszerzeń: m.in. Flask-Login (autoryzacja), Flask-Migrate (migracje baz danych), Flask-WTF (formularze).

Typowe zastosowania:

- Szybkie prototypowanie serwisów webowych.
- Małe i średnie REST API.
- Backend do aplikacji front-endowych (np. React, Vue).

Zalety:

- Niska bariera wejścia — idealny dla początkujących.
- Duża i aktywna społeczność, wiele gotowych rozwiązań i poradników.
- Łatwość testowania i debugowania.
- Duża elastyczność — brak narzuconej struktury projektu.

Wady:

- Brak natywnej obsługi asynchroniczności.
- Wymaga ręcznej konfiguracji przy większych projektach (brak ORM czy kontroli dostępu „out of the box”).

Oficjalna dokumentacja: https://flask.palletsprojects.com/

Repozytorium GitHub: https://github.com/pallets/flask

FastAPI

FastAPI to nowoczesny framework webowy służący do budowania wysokowydajnych interfejsów REST API w Pythonie. Jego główną zaletą jest pełna integracja z typowaniem statycznym Pythona oraz wykorzystanie asynchronicznych operacji (async/await), co umożliwia tworzenie skalowalnych i bezpiecznych aplikacji. Dzięki wykorzystaniu biblioteki Pydantic oraz standardu OpenAPI, FastAPI automatycznie generuje dokumentację interfejsu w postaci Swagger UI oraz Redoc.

Najważniejsze cechy:

- Automatyczne generowanie dokumentacji OpenAPI i interfejsu Swagger UI.
- Obsługa JSON Schema i walidacja danych wejściowych przy użyciu Pydantic.
- Wysoka wydajność dzięki wykorzystaniu Starlette (asynchronicznego frameworka).
- Zgodność ze standardami REST i JSON:API.
- Obsługa zależności (dependency injection), autoryzacji i middleware.

Typowe zastosowania:

- Tworzenie nowoczesnych REST API.
- Backend do aplikacji mobilnych i front-endowych.
- Aplikacje wymagające wysokiej wydajności i skalowalności (np. mikroserwisy).

Zalety:

- Wysoka wydajność i niskie zużycie zasobów.
- Automatyczna, interaktywna dokumentacja API.
- Typowanie i walidacja danych ograniczają liczbę błędów programistycznych.
- Asynchroniczność zwiększa możliwości obsługi wielu zapytań równocześnie.
- Współpraca z ORM (np. SQLAlchemy), serwerami ASGI i bazami NoSQL.

Wady:

- Nieco większa krzywa uczenia się w porównaniu do Flask.
- Mniejszy ekosystem w porównaniu do starszych frameworków.
- Wymaga znajomości asynchronicznego programowania, co dla części programistów może być barierą.

Oficjalna dokumentacja:

https://fastapi.tiangolo.com/

Repozytorium GitHub:

https://github.com/tiangolo/fastapi

Link do repozytorium: : https://github.com/Mateusz-Pirog/python-intro