# Wprowadzenie teoretyczne

- Endpoint
- Kody http
- JSON
- Request Body / Params
- JWT Token

# Flask Server

## Instalacja potrzebnych paczek

Do zainstalowania wszystkich potrzebnych paczek użyjemy pipenv, który powinien zainstalować wszystkie dependencje.

```
py -m pipenv install
```

## Schema (schema.sql)

- W pliku zdefiniowana jest prosta schema bazy danych, która pomoże nam w utrzymywaniu danych

## Init Database (init_db.py)

W celu stworzenia bazy danych należy odpalić komendę:

```
py -m init_db.py
```

lub

```
python init_db.py
```

Jeżeli wszystko się powiodło powinniśmy zobaczyć plik **database.db**.

Dodatkowo możemy otworzyć bazę przy użyciu rozszerzenia Database Client.

- Otwieramy **Database Client**
- Wybieramy **SQLite**
- Szukamy ścieżki do pliku **database.db** i klikamy **Connect**
- Otwieramy tabelke **Posts**

## Serwer (app.py)

Włączenie serwera:

```
py -m flask run
```

lub

```
flask run
```

# Wywoływanie zapytań do serwera

## Postman

- Pobierz program dostępny pod linkiem
- Zaimportuj plik **lab_04.postman_collection.json**
- Poeksperymentuj z zapytaniami i przeanalizuj odpowiedzi

## Rest Client

- Pobierz rozszerzenie VS Code Rest Client
- Otworz plik **collection.http**
- Poeksperymentuj z zapytaniami i przeanalizuj odpowiedzi
- Porównaj z programem **Postman**

# Debugowanie

## VS Python Extension

- Dodaj plik **launch.json** do folderu **.vscode**
- Zainstaluj rozszerzenia do pythona
  - https://marketplace.visualstudio.com/items?itemName=ms-python.python
  - https://marketplace.visualstudio.com/items?itemName=ms-python.debugpy
- Otwórz sekcję **Run and Debug**
- Kliknij przycisk **Run**
- Dodaj **breakpoint**

# Zadania [10 pkt]

Celem dzisiejszego laboratorium jest stworzenie analogicznego serwisu w **Node.js** Express dla konkretnej dziedziny (np. księgarnia).

Serwis 1 - obsługa książek [2.5 pkt]:

- posiada baze danych (np. SQLite), która przechowuje aktualny stan książek w bibliotece (books)
- posiada metody:
  - GET /api/books - pobranie wszystkich dostępnych książek (bez podania parametru)
  - GET /api/books/{ID_KSIAZKI} - pobranie jednej konkretnej książki (ID książki)
  - POST /api/books - dodanie nowej książki do księgarni
    - nazwa

        - autor
        - rok
        - zwraca ID_KSIAZKI
    - DELETE /api/books/{ID_KSIAZKI} - usunięcie książki ze sklepu (ID książki)

Serwis 2 - obsługa zamówień [2.5 pkt]:

- GET - /api/orders/{ID_UZYTKOWNIKA} - pobranie wszystkich zamówień użytkownika
- POST - /api/orders - dodanie zamówienia
    - ID_KSIAZKI
    - ilość
    - zwraca ID_ZAMOWIENIA
    - należy sprawdzić czy książka o danym ID_KSIAZKI istnieje
- DELETE - /api/orders/{ID_ZAMOWIENIA} - usunięcie zamówienia
- PATCH - /api/orders/{ID_ZAMOWIENIA}

Serwis 3 - obsługa użytkowników [2 pkt]:

- POST /api/register - założenie konta w księgarni (rejestracja)
    - E-mail
    - Hasło
    - zwraca ID_UZYTKOWNIKA
- POST /api/login - logowanie do księgarni
    - e-mail
    - hasło
- endpointy do edycji / dodawania / usuwania powinny być zabezpieczone (JWT token) [1 pkt]
- stworzenie kolekcji (Postman lub Http Client) zawierającej wszystkie endpointy [1 pkt]
- do obsługi bazy proszę użyć narzędzia typu ORM np. sequelize [1 pkt]