**Documentație - Aplicație Web Python OOP**

Proiectul de față constă într-o aplicație web asincronă, realizată în Python, care implementează arhitectura MVCS (Model–View–Controller–Service). Aceasta oferă o interfață web modernă pentru efectuarea de calcule matematice (ridicare la putere, factorial, Fibonacci), susținută de o infrastructură robustă de logare, autentificare și gestionare a sesiunilor. Interfața utilizator este construită cu Bootstrap, iar datele sunt gestionate într-o bază SQLite accesată asincron prin aiosqlite.

Tehnologii și Biblioteci Utilizate:  
Python 3.11+ — limbajul principal de programare  
Flask — framework web minimalist și extensibil  
Bootstrap — framework CSS pentru interfețe responsive  
SQLite — sistem de gestionare a bazelor de date relaționale  
SQLAlchemy — ORM pentru manipularea abstractă a datelor  
Pydantic — validarea și serializarea datelor de intrare  
aiosqlite — interacțiune asincronă cu baza de date SQLite

Structura Directorului ProiectuluiTemaPythonOOP/

├── app/  
│ ├── main.py # Inițializare Flask și înregistrare blueprint-uri  
│ ├── controllers/  
│ │ ├── auth\_controller.py # Gestionare autentificare  
│ │ ├── log\_controller.py # Vizualizare istoric activități  
│ │ └── math\_controllers.py # API pentru operații matematice  
│ ├── services/  
│ │ └── math\_services.py # Logica matematică principală  
│ ├── models/schemas.py # Definirea schemelor Pydantic  
│ ├── db/database.py # Conexiune și logare în SQLite  
│ └── utils/auth\_decorator.py # Decorator pentru rutare protejată  
├── templates/ # Șabloane HTML stilizate cu Bootstrap  
├── static/ # Fișiere statice: CSS, JS, imagini  
├── dockerfile # Configurație Docker pentru containerizare  
├── .env # Configurații de mediu (chei, mod debug)  
├── requirements.txt # Lista dependențelor Python  
├── logs.db # Baza de date locală

Fișierul .env conține parametrii de configurare utilizați de aplicație. Exemplu:  
SECRET\_KEY=supersecretkey  
FLASK\_ENV=development

Ghid de RulareLocal (dezvoltare)  
python3 -m venv .venv  
source .venv/bin/activate # Linux/MacOS  
.venv\Scripts\activate # Windows  
pip install -r requirements.txt  
python app/main.pyDocker (producție/testare)docker build -t tema-python-oop .  
docker run --env-file .env -p 5000:5000 tema-python-oopAcces aplicație: http://localhost:5000

Funcționalități1. AutentificareGET /login – Afișează formularul de autentificare  
POST /login – Procesează credențialele introduce  
GET /logout – Închide sesiunea utilizatorului current  
Rutele utilizează metoda GET, acceptând parametrii prin query string, și returnează răspunsuri JSON:  
- GET /pow?a=2&b=3 – Returnează 2 la puterea 3  
- GET /factorial?n=5 – Returnează factorialul lui 5  
- GET /fibonacci?n=7 – Returnează al 7-lea termen din șirul Fibonacci

Toate apelurile sunt logate asincron în baza de date, împreună cu datele de intrare și codul de răspuns.

Istoric activități:   
GET /logs – Vizualizează toate operațiile efectuate de utilizatorul autentificat

Descrierea Componentelor:   
\* main.py - Inițializează aplicația Flask, încarcă și înregistrează blueprint-urile definite în modulele de control, apoi pornește serverul.  
\* auth\_controller.py - Gestionează procesele de login și logout și gestionează sesiunile utilizatorilor.  
\* math\_controllers.py - Definește rutele asincrone /pow, /factorial, și /fibonacci, validează datele cu Pydantic și răspunde cu JSON. Fiecare apel este logat.  
\* math\_services.py - Implementări pure ale algoritmilor: pow, factorial, Fibonacci  
\* log\_controller.py - Oferă interfața web pentru afișarea activităților logate, accesibilă doar utilizatorilor autentificați.  
\* auth\_decorator.py - Decorator care impune autentificarea. Redirecționează către /login dacă sesiunea nu este validă.  
\* database.py - Inițializează conexiunea la baza de date și oferă funcția asincronă log\_request() pentru înregistrarea detaliată a fiecărei operații.