Haladó Programozás beadandó

Dinamikus honlap készítése Flask keretrendszerben

Móricz Pál
TKEX38

2025

T

TARTALOMJEGYZÉK

Bevezetés

A projekt egy Flask [1] alapokra épülő webalkalmazás, amely filmek keresését, megtekintését és a felhasználói interakciók kezelését valósítja meg. Az alkalmazás a TMDb [7] (The Movie Database) külső API-ját használja a filmek adatainak lekéréséhez, így a felhasználó részletes információkat érhet el a filmekről. A rendszer lehetőséget biztosít regisztrált felhasználók számára értékelések és kedvencek mentésére, amelyek egy SQLite [9] adatbázisban tárolódnak. A felhasználói felület Jinja2 [2] sablonok segítségével dinamikusan generálódik, így a megjelenés igazodik az aktuálisan betöltött adatokhoz. A projekt célja egy letisztult felépítésű, jól strukturált webalkalmazás létrehozása, amely egyaránt demonstrálja a Flask keretrendszer rugalmasságát és a modern webfejlesztés alapvető technikáit.

1. Felhasznált technológiák

Az alkalmazás alapját a Flask webkeretrendszer adja, amely gondoskodik a route-ok kezeléséről, a szerveroldali logikáról és az alkalmazás általános működéséről. A felhasználói hitelesítést a Flask-Login [6] modul valósítja meg, amely egyszerű és biztonságos munkamenet-kezelést biztosít. Az adatkezelés SQLAlchemy ORM [5]-en keresztül történik, amely lehetővé teszi a Python osztályok és az adatbázistáblák közötti egyértelmű kapcsolat kialakítását. A projekt SQLite [9] adatbázist használ, amely ideális választás kisebb, helyi fejlesztésű webalkalmazások számára. A külső adatforrás a TMDb API, amely film adatokat szolgáltat keresésekhez és részletes megjelenítéshez. A frontend elemei Jinja2 sablonmotorral generálódnak, amely dinamikusan állítja össze a HTML oldalakat az éppen elérhető adatok alapján. Mindezek mellett az alkalmazás Python nyelven íródott, és a HTTP kommunikációt a Requests [8] könyvtár kezeli.

Backend

- Flask [1] — könnyű, moduláris Python web framework, amely kezeli a route-okat és a szerveroldali logikát.
- Flask-Login [6] — felhasználói munkamenetek kezelése.
- SQLAlchemy [5] — ORM az adatbázis-modellekhez és SQL műveletekhez.
- SQLite [9] — helyi adatbázis a felhasználói adatok és értékelések tárolásához.
- TMDb API [7] — filmek keresése és részletes információk lekérése.

Frontend

- Jinja2 [2] — sablonmotor HTML oldalak dinamikus generálásához.
- HTML/CSS — az alkalmazás felhasználói felületének kialakításához.

Futtatáshoz szükséges telepíteni a következő dolgokat.

- flask
- flask_sqlalchemy
- flask_login
- werkzeug
- requests

1.1. Flask

Az érzelmek leolvasásához a DeepFace framework lett alkalmazva. Az arc detektálást a Haar Cascade végzi ezért DeepFace csak az arc kivágott részét elemzi. És ezekről az arcokról megtudja becsülni az illető korát és nemét. Kimentnek megadja a domináns érzelmet az ember arcán és egy eloszlást az összes érzelemről, amit érzékelni tudott. A választás azért a DeepFace-re esett mivel ugyan úgy, mint a Haar Cascade könnyen integrálható volt a projektbe és könnyen lehet vele tovább dolgozni.

1.2. Flask-Login

A Flask-Login a felhasználók hitelesítését és munkamenet-kezelését végzi az alkalmazásban. A modul biztonságos bejelentkezési folyamatot biztosít, és gondoskodik arról, hogy a hitelesített felhasználók adatai elérhetők legyenek a teljes alkalmazásban. A current_user [1] objektum segítségével bármely route könnyen megállapíthatja, ki van bejelentkezve, és szükség esetén korlátozhatja a hozzáférést bizonyos funkciókhoz.

1.3. SQLAlchemy

A projekt az SQLAlchemy ORM rendszert használja az adatbázissal való kommunikációra. Az ORM lehetővé teszi, hogy az adatbázis tábláit Python osztályok reprezentálják, így az SQL utasítások írása helyett objektumorientált módon történik az adatkezelés. A projekt SQLite adatbázist használ, amely teljesen kompatibilis az SQLAlchemy-vel, ugyanakkor az ORM használata lehetővé teszi, hogy később bármikor más adatbázisrendszerre is át lehessen térni.

1.4. SQLite

Az alkalmazás az SQLite adatbázist használja a felhasználói adatok, értékelések és kedvencek tárolására. Az SQLite egy fájlalapú adatbázisrendszer, amely nem igényel külön adatbázis-szervert. Ennek köszönhetően gyorsan beüzemelhető és fejlesztési környezetben igen hatékony. Az adatbázis egyetlen .db fájlban található, amely könnyen mozgatható, másolható vagy biztonsági mentéssel ellátható.

1.5. TMDb API

Az alkalmazás a The Movie Database (TMDb) nyilvános API-ját használja a filmek adatainak lekérésére. A TMDb kiváló minőségű, folyamatosan frissített adatbázist biztosít több millió filmmel és sorozattal. Az API JSON formátumban szolgáltat adatokat, amelyet a backend könnyen feldolgoz, majd továbbít a HTML sablonok felé. Ez az integráció biztosítja, hogy az alkalmazás mindig aktuális adatokat tudjon megjeleníteni a felhasználóknak.

1.6. Jinja2

A Jinja2 a Flask alapértelmezett sablonmotorja, amely lehetővé teszi dinamikus HTML oldalak létrehozását. A sablonok képesek Python változók fogadására, így a backendből érkező adatok könnyen megjeleníthetők a felhasználói felületen. A sablonöröklődés segítségével egyetlen központi base.html fájl hozható létre, amely meghatározza az oldal fő szerkezetét, míg az egyes oldalak csak a saját tartalmukat adják hozzá. Ez nagymértékben csökkenti a kódismétlést és megkönnyíti a weboldal karbantartását.

1.7. HTML/CSS

A projekt felhasználói felületét HTML és CSS segítségével valósítja meg. A HTML határozza meg az oldalak szerkezetét, például a fejléceket, filmkártyákat, űrlapokat és navigációs elemeket. A sablonok Jinja2 használatával dinamikusan töltődnek fel tartalommal, így a HTML szerkezet egyszerre statikus és adat vezérelt. A CSS biztosítja az alkalmazás vizuális megjelenését, beleértve a színeket, betűtípusokat és elrendezést. A stíluslapok segítségével a weboldal egységes és letisztult felületet kap, amely javítja a felhasználói élményt.

2. A projekt felépítése

A projekt egy modulárisan felépített Flask webalkalmazás, ahol a különböző funkciók jól elkülönített fájlokba és mappákba szerveződnek. Az alkalmazás szerveroldali logikáját a Flask biztosítja, a route-ok pedig külön Python fájlokba kerültek, így az egyes oldalakat és műveleteket könnyebb átlátni és karbantartani. A felhasználói felület Jinja2 sablonok segítségével valósul meg.

2.1. app.py

Az *app.py* az alkalmazás központi indítófájlja, amely létrehozza és konfigurálja a Flask alkalmazást. Ebben a modulban történik a konfigurációs beállítások betöltése, az adatbázis inicializálása és a Flask-Login integrálása, amely a felhasználók hitelesítését kezeli. A fájl egy context processort [3] is definiál, amely automatikusan elérhetővé teszi a sablonok számára a bejelentkezett felhasználót és az aktuális nyelvi beállítást. Az útvonalak külön blueprintekben kerültek létrehozásra, amelyeket itt regisztrál az alkalmazás. Az *app.py* gondoskodik továbbá az adatbázistáblák létrehozásáról.

2.2. config.py

A *config.py* fájl az alkalmazás konfigurációs beállításainak betöltéséért felelős. A benne található load_config() függvény a Flask alkalmazás példányát kapja meg, majd beolvassa a működéshez szükséges kulcsfontosságú értékeket külön fájlokból. Itt kerül betöltésre az alkalmazás titkos kulcsa, amely a sessionök és a hitelesítés biztonságát biztosítja, valamint a TMDb API kulcsa, amely a külső filmadatbázishoz való kapcsolódáshoz szükséges.

2.3. models.py

A *models.py* fájl az alkalmazás adatbázismodelljeit határozza meg SQLAlchemy segítségével. A modul három modellt tartalmaz: a *User*, *Rating* és *Favorite* táblákat. A *User* modell tárolja a felhasználók alapadatait, valamint a jelszó hash-elésre és ellenőrzésre is akalmaz. A *Rating* modell a felhasználók filmekhez adott értékeléseit rögzíti, egyedi megszorítással, hogy egy filmhez csak egy értékelés tartozhasson felhasználónként. A *Favorite* modell hasonló módon a kedvencek listáját kezeli, biztosítva, hogy egy film csak egyszer kerülhessen a felhasználó kedvencei közé.

2.4. recommender.py

A *recommender.py* az alkalmazás saját ajánló rendszerét valósítja meg. A TMDb API-t használja arra, hogy egy adott filmhez hasonló filmeket keressen, figyelembe veszi a felhasználónál beállított nyelvet is. A recommend_for_user() függvény a felhasználó kedvenc filmjei alapján ad ajánlásokat, a legelső kedvenc filmhez hasonló találatok listázásával.

2.5. tmdb.py

A *tmdb.py* fájl az alkalmazás TMDb API-val való kommunikációját valósítja meg. Ez a modul felelős az API-kulcs és az alap URL kezeléséért, a nyelvi beállítások alkalmazásáért, valamint a külső lekérések biztonságos végrehajtásáért. A safe_tmdb_request() függvény egységes módot biztosít az API-hívások kezelésére, hibakezeléssel és alap értékkel lett megoldva. A fájl több segédfüggvényt is tartalmaz, amelyek népszerű filmek, filmrészletek, műfajok vagy hasonló filmek lekérését végzik.

2.6. __init__.py

A *routes* mappa __init__.py fájlja a különböző útvonalakat tartalmazó modulok összefogására szolgál. A projekt route-jai külön fájlokban kaptak helyet — *main_routes*, *auth_routes* és *user_routes* — amelyek mind saját Blueprintként [4] működnek. A Blueprintek lehetővé teszik, hogy az alkalmazás útvonalai logikailag elkülönüljenek egymástól, így a projekt felépítése áttekinthetőbb és jobban karbantartható.

Az __init__.py feladata, hogy ezeket a Blueprint objektumokat importálhatóvá tegye az alkalmazás számára. Az __all__ lista segítségével meghatározza, mely komponensek érhetők el kívülről, így az *app.py* egyszerűen tudja regisztrálni az összes útvonalat.

2.7. auth_routes.py

Az *auth_routes.py* fájl az alkalmazás felhasználói hitelesítésért felelős útvonalait tartalmazza, külön Blueprint formájában. A modul három fő funkciót valósít meg: a regisztrációt, a bejelentkezést és a kijelentkezést. A regisztráció során a rendszer ellenőrzi, hogy a megadott felhasználónév vagy e-mail cím már létezik-e, majd új felhasználót hoz létre az adatbázisban. A bejelentkezés részben a felhasználó megadhat felhasználónevet vagy e-mail címet, és a rendszer jelszóellenőrzés után belépteti. A kilépés gomb megnyomásával kilépteti a rendszerből a felhasználót.

2.8. main_routes.py

A *main_routes.py* fájl az alkalmazás fő funkcióit megvalósító útvonalakat tartalmazza, külön Blueprintben szervezve. Ez a modul felelős a főoldal megjelenítéséért, amely keresés esetén a TMDb API-ból lekért találatokat jeleníti meg, egyébként pedig a népszerű filmek listáját mutatja. Tartalmazza a keresőhöz tartozó útvonalak a műfajok listáját, illetve a keresési eredményeket jelenítik meg, opcionális műfaj szerinti szűréssel. A fájl kezeli a filmrészletek oldalát is, amely a TMDb API-ból származó adatok alapján jeleníti meg a film információit, valamint összekapcsolja azokat a felhasználó adatbázisban tárolt értékeléseivel és kedvenceivel.

2.9. user_routes.py

A *user_routes.py* fájl a felhasználókhoz kapcsolódó műveletek útvonalait tartalmazza, külön Blueprint alatt. A modul kezeli a kedvencek megjelenítését, hozzáadását és eltávolítását, valamint a felhasználó saját értékeléseinek kezelését. A kedvencek és értékelések megjelenítésekor a rendszer a TMDb API segítségével tölti be a filmek adatait. fájl tartalmaz egy útvonalat az értékelések rögzítésére és törlésére, amelyek az adatbázisban tárolódnak.A modul része továbbá a személyre szabott ajánlások oldala is. Ez támogatja mind a TMDb API által kínált ajánlásokat, mind a projekt saját, egyszerű tartalom-alapú ajánlólogikáját. A felhasználó választhat a két mód között, és az eredmények a kedvencek listája alapján kerülnek meghatározásra.

2.10. Html fájlok

A *base.html* fájl az alkalmazás összes oldalának közös alapstruktúráját határozza meg. Ez a sablon tartalmazza a weboldal fő kereteit, például a navigációs sávot, a fejléceket, a CSS hivatkozásokat és az oldal általános elrendezését. A Jinja2 sablonöröklés segítségével minden más HTML oldal erre a fájlra épül, és csak a saját tartalmát helyezi el a kijelölt blokkokban.
A html sablonok tartalma:

- index.html – A főoldal sablonja, ahol a népszerű filmek vagy a keresési találatok jelennek meg.
- favorites.html – A felhasználó kedvencekhez hozzáadott filmjeinek listáját jeleníti meg.
- login.html – A bejelentkezési űrlap sablonja.
- register.html – A felhasználók regisztrációjához szükséges űrlap sablonja.
- movie_details.html – Egy kiválasztott film részletes adatait, értékeléseit és a felhasználóhoz kapcsolódó információkat jeleníti meg.
- my_ratings.html – A felhasználó által adott értékelések listáját mutatja.
- search.html – A keresőoldal sablonja, műfajok listájával és keresési mezővel.
- search_results.html – A keresési eredmények megjelenítésére szolgáló sablon.
- recommendations.html – A felhasználónak ajánlott filmek listáját jeleníti meg, választott ajánlási módtól függően.

3. Mesterséges Intelligencia alkalmazása a projektben

Mesterséges intelligencia alkalmazva volt a HTML és CSS fájlok legenerálásához. Akkor is alkalmazva volt, amikor nem sikerült megoldani egy error-t a programban több próbálkozás után se. A két fajta ajánló rendszer összekötéséhez is alkalmazva volt. Refaktorálásra is alkalmazva volt mivel egy ponton a kód csak egy app.py-ból állt és átláthatatlanná vált a kód.

A mesterséges intelligencia használata néha sokkal több munkát csinált ahelyett, hogy könnyebb lett volna a beadandó elkészítése.

Források

[1] Flask webkeretrendszer dokumentáció – https://flask.palletsprojects.com/

[2] Jinja2 sablonmotor dokumentáció – https://jinja.palletsprojects.com/

[3] Context processor dokumentáció – https://flask.palletsprojects.com/en/stable/templating/

[4] Flask Blueprint – https://flask.palletsprojects.com/en/stable/blueprints/

[5] SQLAlchemy ORM dokumentáció – https://docs.sqlalchemy.org/

[6] Flask-Login modul dokumentáció – https://flask-login.readthedocs.io/

[7] TMDb API dokumentáció dokumentáció – https://developer.themoviedb.org/

[8] Requests könyvtár dokumentáció – https://requests.readthedocs.io/

[9] SQLite adatbázis dokumentáció – https://www.sqlite.org/docs.html