Teodoro Grauso
Matricola: 0522501911
Professore: Luigi Di Biasi
Anno Accademico: 2024/2025

FilmFinder

Progetto di Basi di Dati II

Webapp per la gestione e ricerca di contenuti Netflix

Esplora migliaia di film, dalle ultime uscite ai grandi classici.

12 giugno 2025

Indice

Introduzione

FilmFinder è una webapp progettata per facilitare e digitalizzare la gestione delle informazioni riguardanti film e serie TV presenti sulla piattaforma Netflix. L'applicazione è pensata per gli utenti che possono registrarsi, accedere tramite login e organizzare la propria lista di contenuti preferiti attraverso un'interfaccia web semplice e intuitiva.

Obiettivi del progetto

Il progetto FilmFinder si propone di fornire una soluzione completa per la gestione personalizzata dei contenuti Netflix, implementando moderne tecnologie web e pratiche di sviluppo software.

Funzionalità principali

Le principali funzionalità offerte da FilmFinder includono:

- **Gestione utenti**: Registrazione e autenticazione degli utenti per un'esperienza personalizzata
- **Catalogo completo**: Ricerca e navigazione nel catalogo di film e serie TV con filtri avanzati
- **Lista preferiti**: Gestione della lista dei preferiti, con possibilità di aggiungere o rimuovere contenuti
- **Dettagli completi**: Visualizzazione di informazioni dettagliate su cast, registi e valutazioni

1 Dataset

Per la realizzazione del progetto è stato utilizzato il dataset **Netflix TV Shows and Movies** disponibile su Kaggle, che fornisce informazioni complete sui contenuti della piattaforma.

Il dataset è composto da due file principali:

`titles.csv`
: Contiene i titoli (film e serie TV) con metadati completi

`credits.csv`
: Contiene i dettagli del cast e dei registi associati ai titoli

1.1 Struttura del dataset titles.csv

Questo file include oltre **5.000 titoli unici** presenti su Netflix, con 15 colonne informative:

Campo	Descrizione
id	ID del titolo su JustWatch
title	Nome del titolo
show_type	Tipo di contenuto (TV show o movie)
description	Breve descrizione del contenuto
release_year	Anno di uscita
age_certification	Certificazione per età
runtime	Durata dell'episodio o del film
genres	Lista dei generi
production_countries	Lista dei paesi di produzione
seasons	Numero di stagioni (solo per TV show)
imdb_id	ID su IMDb
imdb_score	Punteggio IMDb
imdb_votes	Numero di voti su IMDb
tmdb_popularity	Popolarità su TMDb
tmdb_score	Punteggio su TMDb

Tabella 1: Struttura del file titles.csv

1.2 Struttura del dataset credits.csv

Contiene oltre **77.000 record** relativi ad attori e registi collegati ai titoli Netflix:

Campo	Descrizione
person_ID	ID della persona su JustWatch
id	ID del titolo su JustWatch
name	Nome dell'attore o regista
character_name	Nome del personaggio interpretato
role	Ruolo (ACTOR o DIRECTOR)

Tabella 2: Struttura del file credits.csv

Nota sui dati mancanti

I dati mancanti nel dataset originale sono stati sostituiti con la stringa `"No Data"` per garantire consistenza nell'elaborazione.

1.3 Arricchimento con immagini di copertina

Per migliorare l'esperienza utente, il dataset è stato arricchito con le immagini di copertina utilizzando le API di `imdbapi.dev`, che permettono di recuperare automaticamente le locandine ufficiali a partire dall'ID IMDb dei titoli.

Il dataset aggiornato, contenente il campo `cover_url`, è stato salvato come `titles_netflix_with_covers` nella cartella `/backend/database`.

1.4 Importazione in MongoDB

Per la gestione dei dati è stato creato un database **MongoDB** chiamato `filmfinder`, strutturato in tre collezioni principali:

Collezione	Contenuto	Documenti
`movie`	Dati dei titoli (film e serie TV)	5.000
`credits`	Informazioni su cast e registi	77.000
`users`	Profili e preferenze utenti	Variabile

Tabella 3: Struttura del database MongoDB

L'importazione è stata effettuata tramite *MongoDB Compass*, che consente di caricare direttamente i file CSV nelle collezioni.

2 Architettura e Tecnologie

La scelta delle tecnologie si è orientata verso soluzioni moderne e consolidate che richiedessero una curva di apprendimento contenuta, favorendo il riuso di componenti già disponibili per ridurre il codice da scrivere e velocizzare lo sviluppo.

Stack Tecnologico

Database: MongoDB + PyMongo
Backend: Flask (Python)
Frontend: React (JavaScript)
API: RESTful Services

Figura 1: Stack tecnologico utilizzato

2.1 PyMongo - Database Driver

PyMongo è il driver ufficiale per interfacciarsi con MongoDB utilizzando Python. Offre metodi semplici e diretti per effettuare connessioni, eseguire query e gestire documenti nel database.

Per centralizzare e gestire la connessione al database, è stata implementata una classe `Database` basata sul pattern *Singleton*:

```
from pymongo import MongoClient
from config import Config

class Database:
    _instance = None
    _client = None
    _db = None

    def __new__(cls):
        if cls._instance is None:
```

```
            cls._instance = super(Database, cls).__new__(cls)
        return cls._instance

    def connect(self):
        if self._client is None:
            self._client = MongoClient(Config.MONGODB_URI)
            self._db = self._client[Config.DATABASE_NAME]
            print(f"Connesso a MongoDB: {Config.DATABASE_NAME}")
        return self._db

    def get_collection(self, collection_name=None):
        if self._db is None:
            self.connect()
        collection_name = collection_name or Config.COLLECTION_NAME
        return self._db[collection_name]

    def close(self):
        if self._client:
            self._client.close()
            self._client = None
            self._db = None
```

Listing 1: Implementazione della classe Database

2.2 Backend con Flask

Il backend dell'applicazione è stato sviluppato con il framework **Flask**, una soluzione leggera e modulare che consente la creazione di API RESTful in modo semplice e scalabile.

2.2.1 Architettura MVC

La struttura del progetto segue il principio della separazione delle responsabilità (Separation of Concerns), suddividendo il codice in moduli chiari e riutilizzabili:

Models
: Definiscono le entità principali come `Movie` e `User`, descrivendo la struttura dei dati

Routes
: Contengono la definizione degli endpoint API e gestiscono le richieste HTTP

Services
: Implementano la logica applicativa e le operazioni complesse con il database

Controllers
: Coordinano il flusso dei dati tra routes e services

```
/backend
  /models
    movie.py          # Modello dati per i film
    user.py           # Modello dati per gli utenti
  /routes
    movie_routes.py   # Endpoint per la gestione film
    user_routes.py    # Endpoint per autenticazione e profili
  /services
    movie_service.py  # Logica di business per i film
    user_service.py   # Logica di business per gli utenti
  /controllers
```

```
    movie_controller.py
    user_controller.py
  app.py              # Applicazione Flask principale
  config.py           # Configurazioni
  database.py         # Gestione connessione database
```

Listing 2: Struttura del backend

2.3 Frontend con React

Per l'interfaccia utente è stata utilizzata **React**, libreria moderna per lo sviluppo di interfacce dinamiche e componenti riutilizzabili.

React è stato impiegato per:

- Gestire la navigazione tra le sezioni dell'applicazione (routing)
- Visualizzare le liste di film e serie TV con componenti ottimizzati
- Implementare funzionalità di registrazione, login e gestione preferiti
- Creare un'esperienza utente fluida e responsive

2.3.1 Flusso delle API

Il sistema implementa un'architettura RESTful con il seguente flusso:

1. Le **routes** ricevono le richieste HTTP dal frontend
2. I **controller** elaborano le richieste e invocano i **services**
3. I **services** eseguono la logica di business interagendo con i **models**
4. I dati elaborati vengono restituiti attraverso lo stesso percorso inverso

3 Presentazione dell'Applicazione

L'applicazione può essere avviata seguendo questi passaggi:

Avvio dell'applicazione

Backend:

```
cd backend
python app.py
```

Frontend:

```
cd frontend
npm start
```

3.1 Schermata di Login

L'accesso all'applicazione avviene attraverso una schermata di login intuitiva che consente l'autenticazione degli utenti registrati.

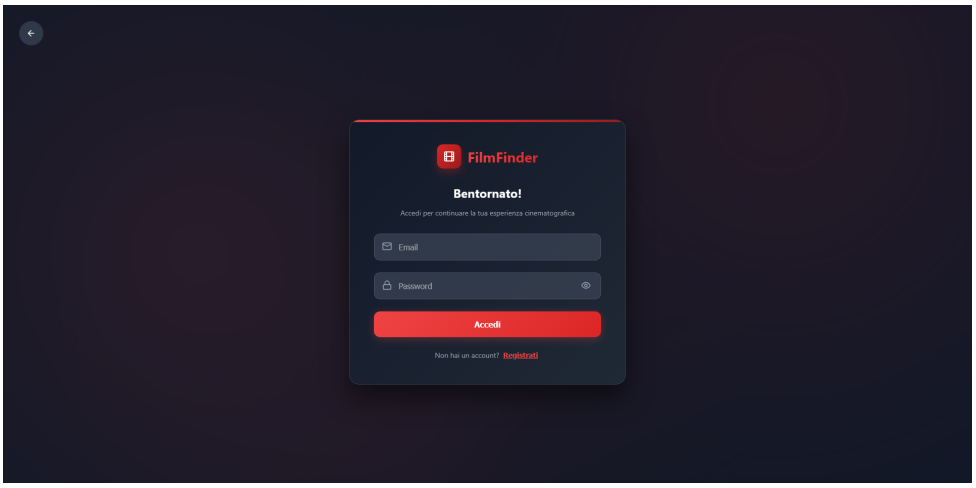

Figura 2: Schermata di login dell'applicazione

3.2 Homepage

Dopo l'autenticazione, l'utente accede alla homepage che presenta una panoramica organizzata dei contenuti disponibili.

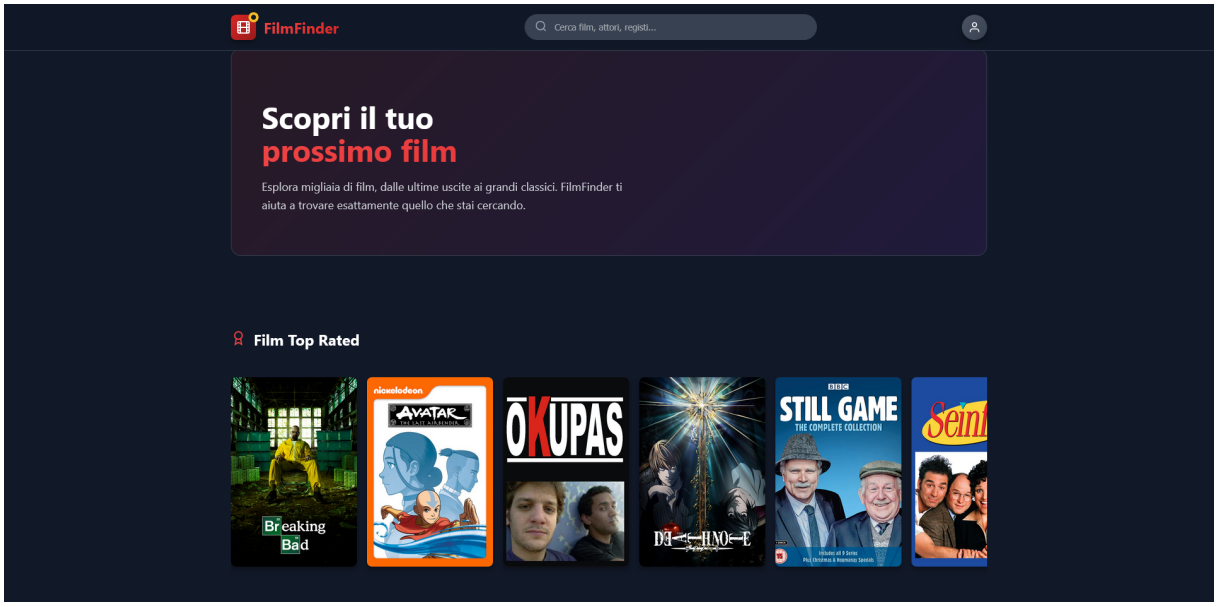

Figura 3: Homepage - Sezione principale con film in evidenza

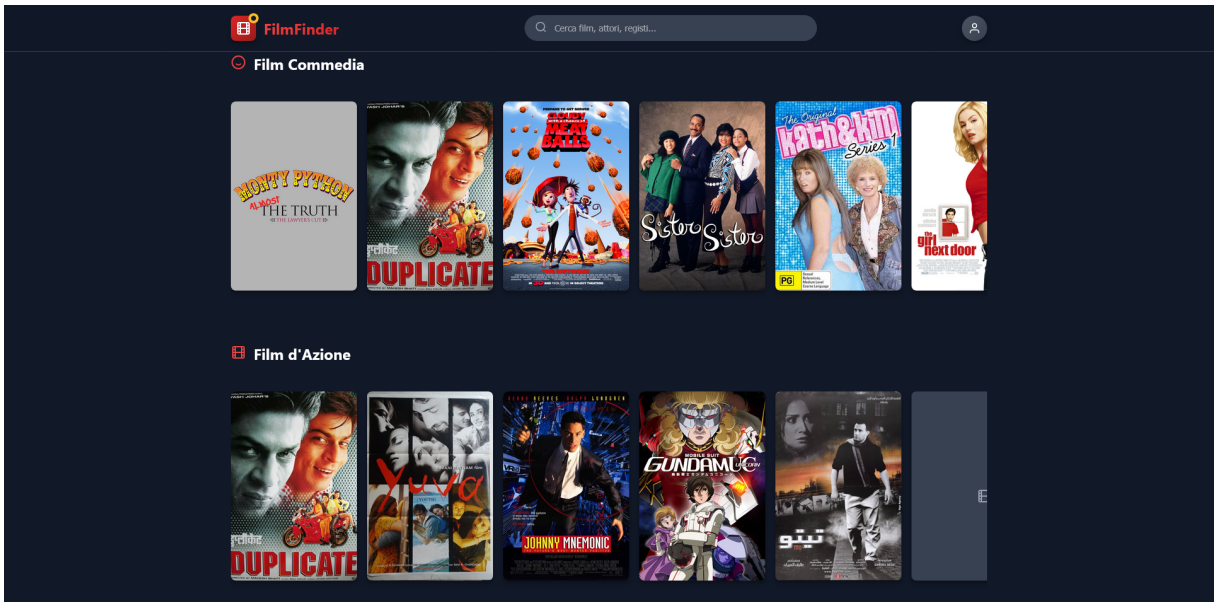

Figura 4: Homepage - Categorie tematiche e suggerimenti

La homepage offre diverse sezioni tematiche:

- **Top Ranking**: Titoli con i punteggi più alti
- **Categorie per genere**: Organizzazione per drama, action, comedy, ecc.
- **Nuove uscite**: Contenuti aggiunti recentemente
- **Suggerimenti personalizzati**: Basati sulle preferenze dell'utente

3.3 Dettagli del Film

Selezionando un titolo dalla homepage si accede alla pagina dei dettagli, che fornisce informazioni complete sul contenuto scelto.

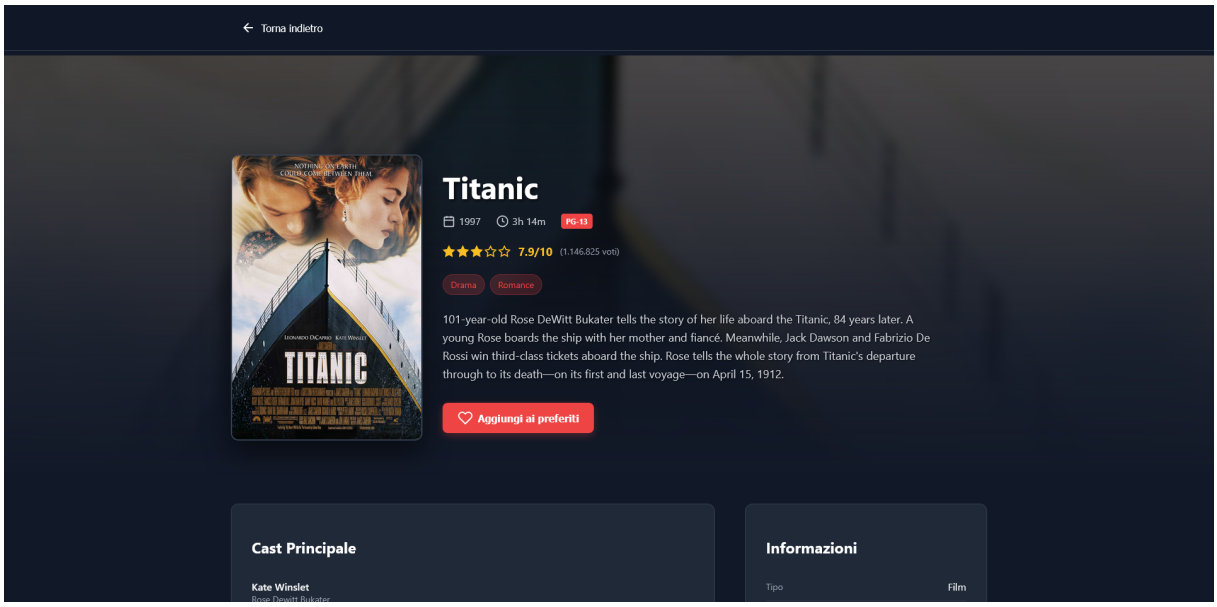

Figura 5: Pagina dettagli film con informazioni complete

Questa pagina include:

- Titolo, anno di uscita e durata
- Trama e descrizione dettagliata
- Valutazioni IMDb e TMDb
- Generi e paesi di produzione
- Pulsante per aggiungere/rimuovere dai preferiti

3.4 Cast e Crew

Attraverso una query con `LEFT OUTER JOIN`, vengono recuperati e visualizzati gli attori e i membri del cast associati al film.

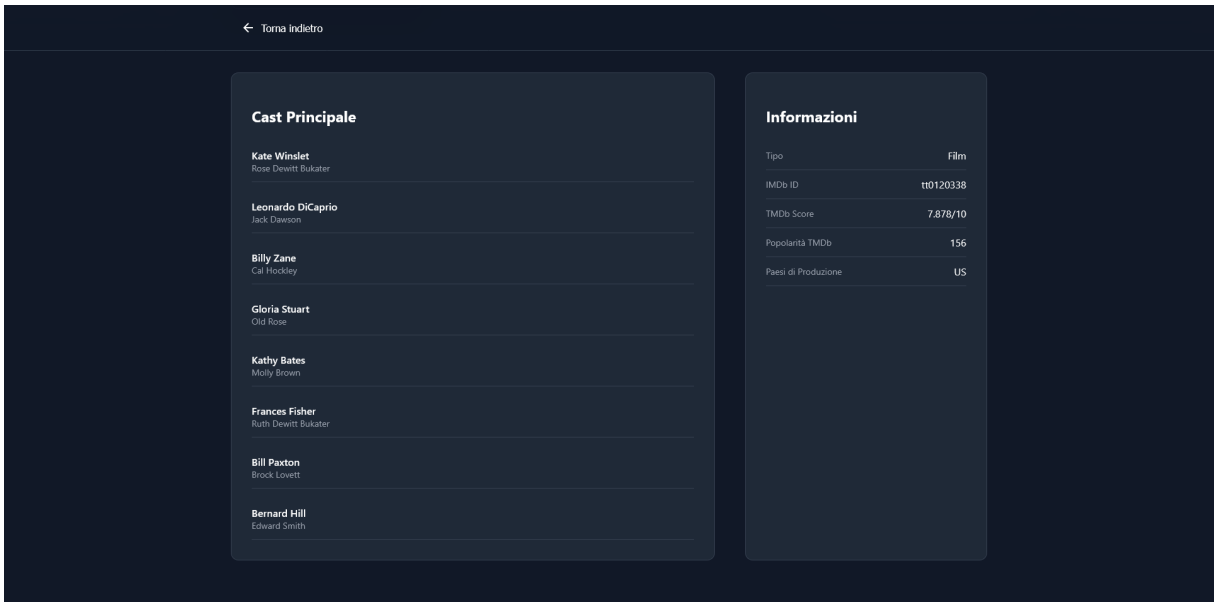

Figura 6: Sezione cast del film con dettagli sui personaggi

3.5 Profilo Utente

La sezione profilo permette agli utenti di gestire la propria collezione personale di film preferiti.

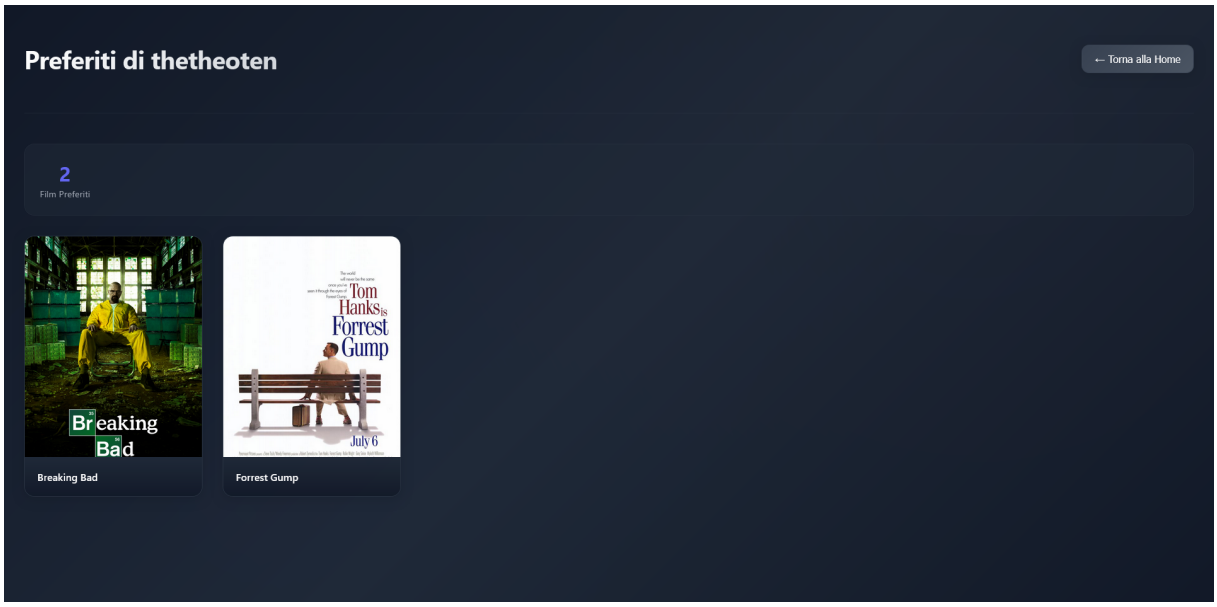

Figura 7: Sezione Profilo con la collezione dei film preferiti

Le funzionalità del profilo includono:

- Visualizzazione di tutti i film aggiunti ai preferiti
- Rimozione rapida dalla lista dei preferiti

- Accesso diretto ai dettagli di ogni film preferito
- Statistiche sulla propria collezione

Conclusioni

Il progetto FilmFinder rappresenta una soluzione completa per la gestione personalizzata dei contenuti Netflix, implementando tecnologie moderne e best practices dello sviluppo web.

L'architettura scelta ha permesso di creare un'applicazione scalabile e manutenibile, mentre l'interfaccia utente intuitiva garantisce un'esperienza d'uso ottimale per la scoperta e la gestione dei contenuti preferiti.

Le tecnologie utilizzate (MongoDB, Flask, React) si sono dimostrate efficaci per la realizzazione degli obiettivi del progetto, offrendo le basi per eventuali futuri sviluppi e miglioramenti.