

MOVIELIST

Luca Baldini & Lorenzo Boggiani

PANORAMICA

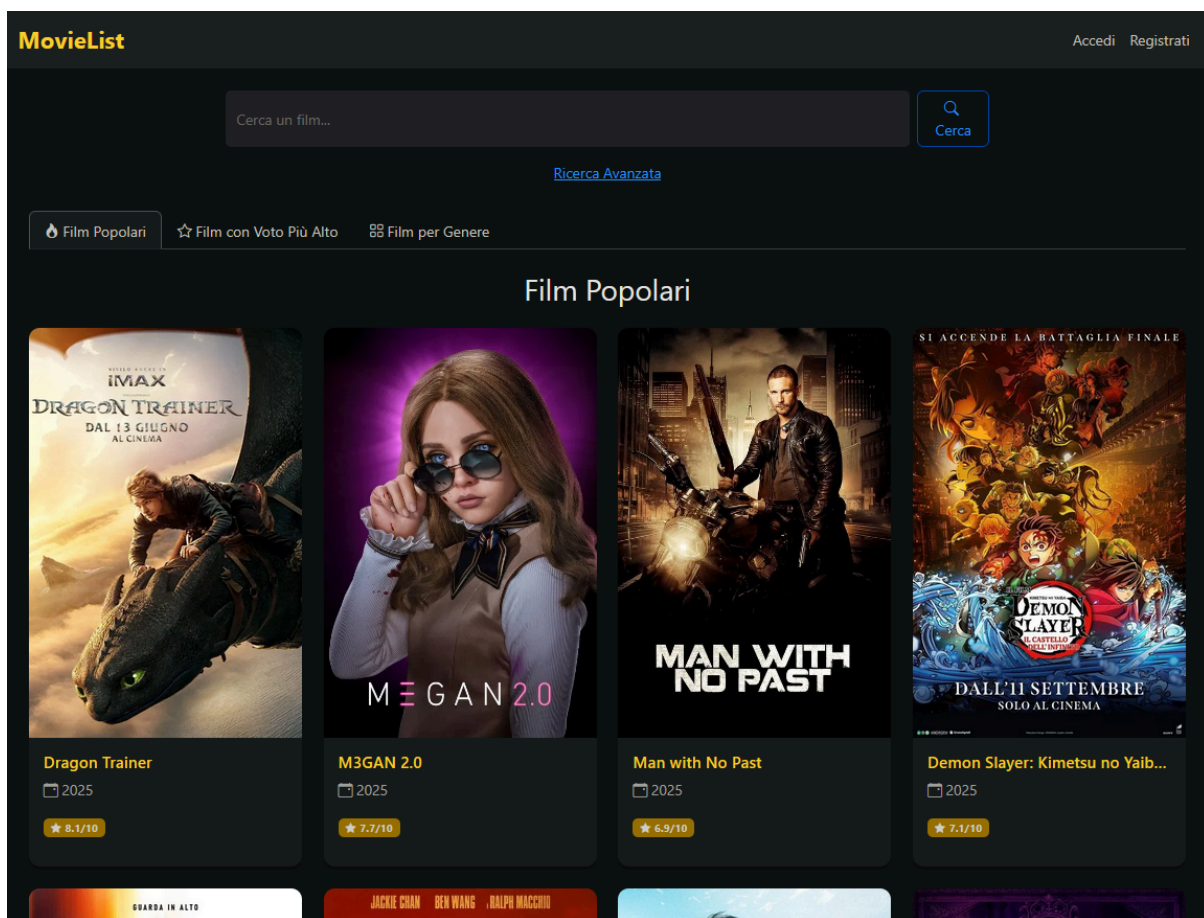
Abbiamo deciso di sviluppare un sito chiamato Movielist, che permette agli utenti registrati di annotare tra i preferiti una lista di film e di inserire commenti e valutazioni. Ci ha attratto la duplice funzionalità di community, grazie alla possibilità di commentare con recensioni visibili da tutti gli utenti, e la gestione dei preferiti che permette al singolo utente di appuntarsi i film già visti o da vedere.

Si è optato per un database esterno di film e relativi dettagli per limitare al massimo l'attività di inserimento contenuti e rendere pertanto il sito a manutenzione zero.

SPECIFICHE

- **Database:** PostgreSQL
- **Linguaggi:** JavaScript, CSS, Pug
- **Applicazioni ed Editor:** WebStorm, pgAdmin, GitHub
- **Tecnologie:** Frontend application, API, Database

STRUTTURA DEL SITO



La homepage presenta una lista di film aggiornata grazie all'accesso mediante API al sito TMDb. E' possibile per l'utente effettuare una registrazione mediante l'inserimento di "nome utente" e "password".

Registrazione

Nome utente

Password

[Registrati](#)

Hai già un account? [Accedi qui](#)

Una volta registrato sarà possibile per l'utente accedere con le proprie credenziali.

Per ogni utente (registrato o non) è possibile visualizzare la lista dei film attraverso filtri "Film Popolari", "Film con Voto Più Alto", "Film per Genere" oltre che effettuare una ricerca testuale.

Ricerca Avanzata

Titolo del film

Anno di uscita

Voto minimo: Qualsiasi


Generi

<input type="checkbox"/> Azione	<input type="checkbox"/> Avventura	<input type="checkbox"/> Animazione
<input type="checkbox"/> Commedia	<input type="checkbox"/> Crime	<input type="checkbox"/> Documentario
<input type="checkbox"/> Dramma	<input type="checkbox"/> Famiglia	<input type="checkbox"/> Fantasy
<input type="checkbox"/> Storia	<input type="checkbox"/> Horror	<input type="checkbox"/> Musica
<input type="checkbox"/> Mistero	<input type="checkbox"/> Romance	<input type="checkbox"/> Fantascienza
<input type="checkbox"/> televisione film	<input type="checkbox"/> Thriller	<input type="checkbox"/> Guerra
<input type="checkbox"/> Western		

[Cerca Film](#)

E' inoltre disponibile una ricerca avanzata eseguibile mediante opportuni filtri.

Queste funzionalità vengono gestite dal frontend da noi sviluppato, che effettua le ricerche mediante API verso il sito TMDB.



VIDEO ANCHE IN IMAX

DRAGON TRAINER

DAL 13 GIUGNO AL CINEMA

[Accedi per aggiungere ai preferiti](#)

[Torna alla Homepage](#)

Dragon Trainer





2025 ★ 8.1/10

Sulla selvaggia isola di Berk, dove vichinghi e draghi sono stati acerrimi nemici per generazioni, Hiccup è diverso dagli altri. Figlio geniale ma sottovalutato dal capo Stoick l'Immenso, Hiccup sfida secoli di tradizione stringendo un'inusolata amicizia con Sdentato, un temibile drago Furia Buia. Questo legame inaspettato rivela la vera natura dei draghi, mettendo in discussione le fondamenta stesse della società vichinga.

Generi

Fantasy Famiglia Azione

Cast Principale

 <p>Mason Thames Hiccup</p>	 <p>Nico Parker Astrid</p>	 <p>Gerard Butler Stoick</p>	 <p>Nick Frost Gobber</p>
---	--	---	---

Per ogni film è disponibile una scheda con dettagli e classificazione del film stesso, quali trama, generi, cast, trailer, raccomandazioni e recensioni; tali informazioni vengono estratte ed aggiornate mediante API verso il sito TMDB

Recensioni

Voto

Seleziona un voto

La tua recensione

Scrivi qui la tua recensione...

🚩

Publica recensione

★ 8/10

admin

23/07/2025

che bel film!

All'interno della scheda del film sono visualizzabili i commenti degli utenti registrati.

Un utente registrato, dopo l'accesso, all'interno della scheda del film ha anche la sezione digitabile per le recensioni.

MovieList

admin

I miei film preferiti

SI ACCENDE LA BATTAGLIA FINALE

Demon Slayer: Kimetsu no Yaiba - Il Castello dell'Infinito

Dettagli

Rimuovi

M3GAN 2.0

M3GAN 2.0

Dettagli

Rimuovi

MovieList

admin

Le mie recensioni

Dragon Trainer

8/10

Valutazione

che bel film!

23/07/2025

Vai al film

Elimina recensione

Per gli utenti registrati sono disponibili due ulteriori pagine contenenti le proprie recensioni e la lista dei film preferiti, da cui si può accedere ai dettagli del film o eventualmente rimuovere rispettivamente la recensione o il film dai preferiti

FUNZIONALITÀ

- Consultazione di un catalogo film estratto mediante API dal sito TMDb
- Possibilità agli utenti di registrarsi e, una volta registrati, di accedere da qualunque dispositivo
- Dopo il login è possibile per gli utenti:
 - Crearsi una lista di film preferiti attraverso il pulsante “Aggiungi ai Preferiti” disponibile su ogni film nella lista principale
 - Inserire una recensione relativa a un film nella sezione finale disponibile su ogni film cliccando su “Pubblica recensione”, dopo aver inserito voto (da 1 a 10) e/o recensione
- Consultazione della lista dei preferiti, da cui eventualmente rimuoverli
- Consultazione delle recensioni
- All'interno delle schede dei film è possibile visualizzare le recensioni degli altri utenti

PROGETTAZIONE DEL DATABASE

In questo progetto si è utilizzato PostgreSQL come database e PgAdmin come applicazione per configurazione, manutenzione e consultazione delle tabelle.

Sono state utilizzate 3 tabelle:

Users: la tabella contiene un record per ogni utente registrato, le colonne sono:

- L'identificativo è definito intero, univoco, progressivo e incrementale e viene gestito direttamente dal sistema.
- Lo Username è definito string ed è univoco
- La Password è definita string e viene memorizzata in maniera criptata mediante algoritmo di hashing con utilizzo di un salt, garantendo una moderna gestione delle password sul db (le password non possono essere ricavate grazie all'hashing e non è possibile nemmeno identificare password comuni grazie al salt)
- createdAt che contiene il timestamp di creazione dell'utente (registrazione)
- updatedAt che contiene il timestamp di ultima modifica dell'utente (attualmente non utilizzata in quanto non è presente alcuna funzionalità di aggiornamento dell'utente)

Tutti i campi sono Not NULL e inseriti al momento del censimento dell'utente

Users

×

General

Columns

Advanced

Constraints

Partitions

Parameters

Security

SQL











Inherited from table(s)

Select to inherit from...

▼

Columns

+

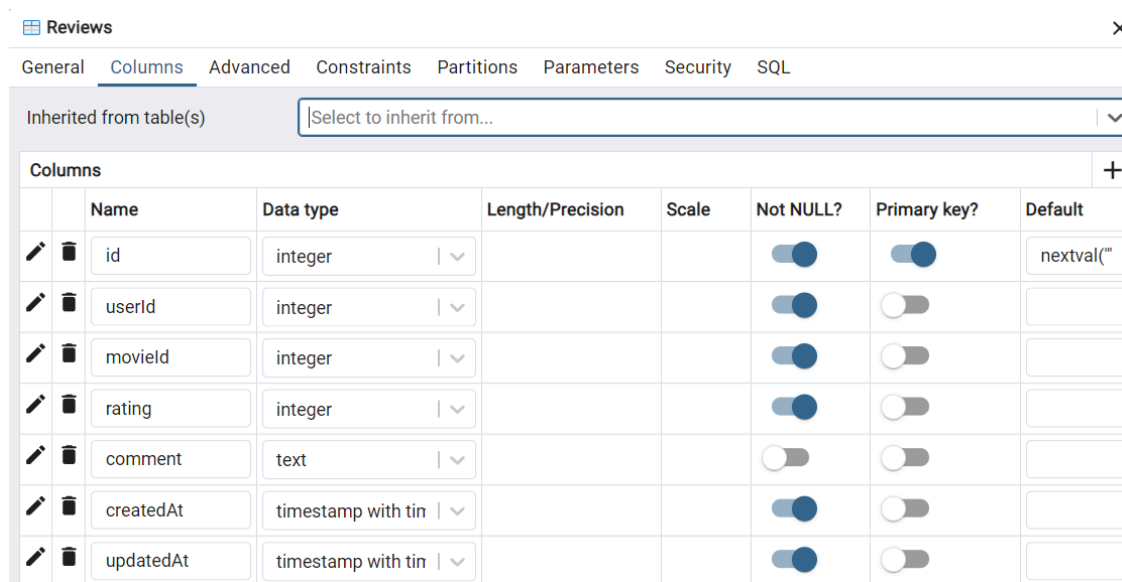
	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
 	id	integer ▼			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval(")
 	username	character varying ▼	255		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 	password	character varying ▼	255		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 	createdAt	timestamp with tin ▼			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 	updatedAt	timestamp with tin ▼			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Reviews: la tabella contiene un record per ogni recensione di un film, le colonne sono:

- L'identificativo è definito intero, univoco, progressivo e incrementale e viene gestito direttamente dal sistema.
- `userId` è definito intero, chiave esterna sulla tabella "**Users.id**" e corrisponde all'utente che ha lasciato la recensione
- `movieId` è definito intero e corrisponde al film su cui si è lasciata la recensione. Non è prevista una chiave esterna in quanto non è presente una tabella "Movie".
I film vengono recuperati mediante API TMDB, che di fatto sostituisce la tabella dei film
- `rating`: definito come intero, rappresenta il voto dato dall'utente al film
- `createdAt` che contiene il timestamp di creazione dell'utente (registrazione)
- `updatedAt` che contiene il timestamp di ultima modifica dell'utente (attualmente non utilizzata in quanto non è presente alcuna funzionalità di aggiornamento dell'utente)

Fin qui i campi sono tutti Not NULL e quindi obbligatori

- `comment`: definito come string, può essere NULL



	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	id	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval(")
	userId	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	movieId	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	rating	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	comment	text			<input type="checkbox"/>	<input type="checkbox"/>	
	createdAt	timestamp with tin			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	updatedAt	timestamp with tin			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Favourites: la tabella contiene un record per ogni film inserito tra i preferiti di un utente, le colonne sono:

- L'identificativo è definito intero, univoco, progressivo e incrementale e viene gestito direttamente dal sistema.
- `userId` è definito intero, chiave esterna sulla tabella "**Users.id**" e corrisponde all'utente che ha selezionato il film come preferito
- `movieId` è definito intero e corrisponde al film preferito.

Non è prevista una chiave esterna in quanto non è presente una tabella "Movie".

I film vengono recuperati mediante API TMDB, che di fatto sostituisce la tabella dei film

- `title`: titolo del film, definito character varying con lunghezza massima di 255 caratteri
- `createdAt` che contiene il timestamp di creazione dell'utente (registrazione)
- `updatedAt` che contiene il timestamp di ultima modifica dell'utente (attualmente non utilizzata in quanto non è presente alcuna funzionalità di aggiornamento dell'utente)

Fin qui i campi sono tutti Not NULL e quindi obbligatori

- posterPath: path dove è disponibile la locandina relativa ai film, definito character varying con lunghezza massima di 255 caratteri.

Può essere NULL in quanto non tutti i film hanno necessariamente una locandina

Favorites

General

Columns

Advanced

Constraints

Partitions

Parameters





























Security

SQL

Inherited from table(s)

Select to inherit from...

Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
		id	integer					nextval(")
		userId	integer					
		movieId	integer					
		title	character varying	255				
		posterPath	character varying	255				
		createdAt	timestamp with tin					
		updatedAt	timestamp with tin					

CSS

E' stata gestita la componente grafica del sito mediante un unico foglio di stile (style.css) in maniera separata dalle pagine logiche in modo da dare uniformità e una maggiore manutenibilità al sito.

```

style.css
1  /* === COLORI PERSONALIZZATI === */
2  :root {
3    --accent-color: #facc15;
4    --bg-dark: #121212;
5    --text-light: #f5f5f5;
6  }
7
8  /* === BODY & BACKGROUND === */
9  body {
10   background-color: var(--bg-dark);
11   color: var(--text-light);
12 }
13
14 /* === NAVBAR === */
15 .navbar-brand {
16   font-weight: bold;
17   font-size: 1.6rem;
18   color: var(--accent-color) !important;
19 }
20
21 .nav-link {

```


STRUTTURA DEL FILE:

1. Body & Background: configurazione grafica delle pagine e relativa struttura
2. Navigazione e Layout dinamico dei contenuti: Definisce stile per il logo, navbar, filmcards, pagine di ricerca, icona profilo e dei film preferiti
3. Form e pulsanti
4. L'utilizzo del CSS permette la riorganizzazione dinamica dei contenuti a fronte di eccesso mediante mobile o di display di dimensioni ridotte

JAVASCRIPT

Il linguaggio JavaScript (JS) è stato utilizzato per aggiungere interattività, dinamicità e logica client-side al sito, in particolare:

tmdb.js

Questo codice è un modulo Node.js che interagisce con l'API di TMDB (The Movie Database) per ottenere informazioni sui film. Utilizza axios per effettuare richieste HTTP e simple-statistics per calcoli di similarità. Le sue principali funzionalità includono:

getMovieGenres: recupera tutti i generi cinematografici disponibili.

getPopularMovies: ottiene la lista dei film più popolari del momento.

getMovieDetails: recupera i dettagli completi di un film, inclusi cast, regista, trailer YouTube, piattaforme di streaming in Italia e film simili (con calcolo della similarità).

searchMovies: permette di cercare film usando filtri avanzati (titolo, anno, voto minimo, generi).

getTopRatedMovies: ottiene i film con i voti più alti (con almeno 100 voti).

getMoviesByGenre: ottiene i film più popolari per un determinato genere.

getCastMemberDetails: recupera informazioni su un attore o regista, inclusa la sua filmografia ordinata per data.

Il modulo esporta tutte queste funzioni per essere usate in altre parti dell'applicazione.

```
JS tmdb.js x
1 // Importa i moduli necessari
2 const axios = require('axios');
3 const ss = require('simple-statistics'); // Per calcoli statistici
4
5 // Configura l'istanza di axios per l'API TMDB
6 // - Imposta l'URL base per tutte le richieste
7 // - Configura i parametri di default (API key e lingua italiana)
8 const tmdbApi = axios.create({
9   baseURL: 'https://api.themoviedb.org/3/',
10   params: {
11     api_key: 'eb1bd9da13ec5672dca46db23dd48f17',
12     language: 'it-IT',
13   }
14 });
15
16
17 /**
18  * Recupera la lista completa dei generi cinematografici da TMDB
```

index.js

Questo file definisce le rotte principali dell'app Express per un sito di film basato su TMDb. Le principali funzionalità sono:

`/`: mostra film popolari, votati meglio e divisi per generi.

`/search`: consente di cercare film con filtri (titolo, anno, voto, genere).

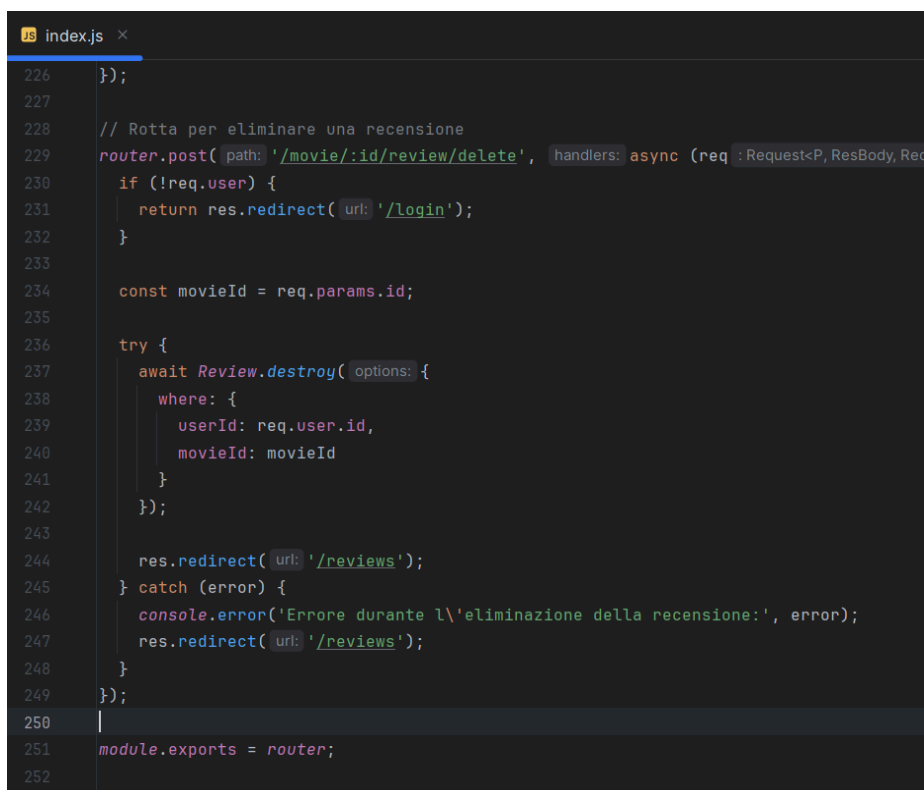
`/movie/:id`: mostra i dettagli di un film, le recensioni e se è tra i preferiti dell'utente.

`/movie/:id/review`: consente agli utenti loggati di lasciare una recensione.

`/movie/:id/review/delete`: permette di eliminare una propria recensione.

`/person/:id`: mostra biografia e filmografia di un attore o regista.

Utilizza moduli esterni per recuperare i dati TMDb relativi ai film.



```
226 });
227
228 // Rotta per eliminare una recensione
229 router.post( path: '/movie/:id/review/delete', handlers: async (req : Request<P, ResBody, Req
230   if (!req.user) {
231     return res.redirect( url: '/login');
232   }
233
234   const movieId = req.params.id;
235
236   try {
237     await Review.destroy( options: {
238       where: {
239         userId: req.user.id,
240         movieId: movieId
241       }
242     });
243
244     res.redirect( url: '/reviews');
245   } catch (error) {
246     console.error('Errore durante l\'eliminazione della recensione:', error);
247     res.redirect( url: '/reviews');
248   }
249 });
250
251 module.exports = router;
252
```

auth.js

Il file definisce le rotte per l'autenticazione e la gestione dell'utente mediante le funzionalità messe a disposizione da Express.js (estensione di Node.js).

Usa Passport.js per l'autenticazione locale e gestisce funzionalità come:

Autenticazione

`GET /login`, `POST /login`: mostra il form di login e autentica l'utente.

`GET /register`, `POST /register`: mostra il form di registrazione e crea un nuovo utente.

`GET /logout`: effettua il logout dell'utente.

Preferiti

POST /favorite/:movieId: aggiunge un film ai preferiti.

GET /favorites: mostra i film preferiti dell'utente loggato.

POST /favorite/remove/:movieId: rimuove un film dai preferiti.

Recensioni

GET /reviews: mostra tutte le recensioni scritte dall'utente, con i dettagli dei film.

Il file utilizza i modelli **User**, **Favorite** e **Review**, e funzioni TMDb per recuperare i dettagli dei film.

```
auth.js
1 // Importazione delle dipendenze necessarie
2 const express :e|()=> core.Express = require('express'); // Framework web
3 const passport :Authenticator|{...} = require('passport'); // Middleware di autenticazione
4 const LocalStrategy :function(Object, Function): vo...|{...} = require('passport-local').Strategy; // Strategia di autenticazione
5 const router :Router = express.Router(); // Router Express
6 const User :... = require('../models/user'); // Modello utente
7 const Favorite :... = require('../models/favorite'); // Modello preferiti
8 const Review :... = require('../models/review'); // Modello recensioni
9 const { getMovieDetails } = require('./tmdb'); // Funzioni API TMDb
10 const flash :function(any): function(any, a...|{...}) = require('connect-flash'); // Messaggi flash per notifiche
11
12 // Configurazione della strategia di autenticazione locale
13 passport.use(new LocalStrategy( options: async (username, password, done) :Promise<...> => {
14   try {
15     // Cerca l'utente nel database
16     const user = await User.findOne({ where: { username } });
17     if (!user) return done(null, false); // Utente non trovato
18     // Verifica la password
19     const isValid = await user.validatePassword(password);
20     if (!isValid) return done(null, false); // Password non valida
21     return done(null, user); // Autenticazione riuscita
22   } catch (err) {
23     return done(err); // Errore durante l'autenticazione
24   }
25 });
26
27 // Serializzazione dell'utente per la sessione
```

In **db.js** viene configurata una connessione a PostgreSQL usando il modulo **pg**. Viene creato un pool di connessioni con vari parametri

Mentre **users.js** definisce una rotta Express per **/users**. Quando viene effettuata una richiesta GET a **/users**, il server risponde con il testo **'respond with a resource'**

app.js

Il file importa il framework Express.js e lo utilizza per le funzionalità presenti sulla nostra web app

CONCLUSIONE

Col progetto Movielist è stato possibile realizzare un'applicazione, che permette di inserire i propri film preferiti e di lasciare una recensione per ognuno di essi.

Attraverso l'uso di tecnologie standard come CSS e JavaScript, l'integrazione con PostgreSQL per la gestione dei dati e della API, è stato possibile costruire una piattaforma accessibile, intuitiva e grazie alla API verso un sito di riferimento che si aggiorna automaticamente.

L'approccio adottato ha garantito una chiara separazione tra logica, presentazione e dati, rendendo il codice manutenibile e scalabile. L'utilizzo dell'API ha permesso di avere film sempre aggiornati, che aggiunte alla possibilità di gestire i preferiti e i commenti hanno reso un sito statico in una web app interattiva. Commenti e preferiti vengono memorizzati sul database PostgreSQL.

Dal punto di vista didattico, il progetto ha permesso di acquisire competenze concrete nello sviluppo frontend e nell'integrazione con servizi backend, utilizzo dell'API, simulando un flusso di lavoro realistico nel mondo dello sviluppo web.

Movielist rappresenta una base solida per un'applicazione "community" evoluta o semplicemente una dashboard in cui l'utente possa annotarsi film visti, da vedere o rivedere. Questa base è pronta per essere ampliata in futuro con funzionalità avanzate come verifica email in registrazione, notifiche push, "amicizie" tra utenti, forum e quant'altro.