Documentazione API REST – Piattaforma Streaming

Autore: Alessio Cattaneo

Introduzione

In questa documentazione viene descritto il funzionamento delle API REST sviluppate per una piattaforma streaming personale realizzata con Laravel. L'obiettivo è fornire un backend sicuro e con gestione ruoli, accessibile solo tramite API, senza interfaccia grafica, concentrandosi sulla logica di autenticazione, risorse e sicurezza degli endpoint.

Autenticazione e Sicurezza

L'autenticazione si basa su JWT (Web Token):

- L'utente si registra passando nome, cognome, sesso, **mail hashata** e password.
- Il login avviene tramite endpoint GET /accedi/{utente}/{hash} dove:
    - **utente** è la mail hashata (SHA512)
    - **hash** è la password cifrata secondo specifica.
- Se le credenziali sono corrette, il server genera un token JWT firmato, salvato nella sessione.
- Tutte le rotte protette richiedono il token JWT nell'header Authorization.

I permessi sono gestiti tramite middleware e ruoli:

- Ospite (guest): solo registrazione e login
- Utente: può vedere i contenuti, modificare profilo, aggiungere crediti
- Admin: può gestire ogni risorsa e gli utenti

Gestione dei Ruoli

Sono presenti tre livelli di accesso:

- **Ospite**: solo registrazione/login
- **Utente**: accesso ai contenuti e al proprio profilo
- **Amministratore**: può aggiungere, modificare e cancellare qualsiasi risorsa e gestire gli utenti

Ogni endpoint è protetto secondo il ruolo richiesto.

Risorse Gestite

Le principali risorse gestite sono:

- **Utenti**: dati anagrafici, login, crediti
- **Film**

- **Serie TV**
- **Episodi**
- **Categorie**

Ogni risorsa è soggetta a permessi differenti a seconda del ruolo.

Rotte Principali

Pubbliche

- POST /api/register/{utente} — Registrazione nuovo utente (con mail hashata)
    - Body: password, password_confirmation, nome, cognome, sesso
- GET /accedi/{utente}/{hash} — Login utente (mail e password hashate)

Protette (utente o admin, richiedono JWT)

- GET /api/me — Info utente autenticato
- PUT /api/me/update — Modifica profilo personale
- POST /api/me/add-credits — Aggiunta crediti
- POST /api/logout — Logout

Contenuti

- GET /api/movies — Visualizza tutti i film
- GET /api/movies/{id} — Dettagli film
- GET /api/series — Tutte le serie
- GET /api/series/{id} — Dettagli serie
- GET /api/episodes — Episodi
- GET /api/episodes/{id} — Dettaglio episodio
- GET /api/categories — Categorie
- GET /api/categories/{id} — Dettaglio categoria

Solo admin

- POST|PUT|DELETE /api/movies, /api/series, /api/episodes, /api/categories — CRUD risorse
- GET|PUT|DELETE /api/users — Gestione utenti

Comportamenti Differenziati per Ruolo

- **Guest:** riceve 401 Unauthorized sulle rotte protette
- **Utente:** accede alle risorse con i campi standard

- **Admin:** può accedere, modificare ed eliminare qualsiasi risorsa e vede eventuali dati extra (es. film archiviati)

Esempi di Body per le Richieste

Registrazione


```
{
  "nome": "Alessio",
  "cognome": "Cattaneo",
  "sesso": 1,
  "password": "Password123!",
  "password_confirmation": "Password123!"
}
```


La mail va inviata come parametro nella URL, già hashata (SHA512):
POST /api/register/{mail_hashata}

Login

Endpoint:
GET /accedi/{mail_hashata}/{password_hashata}

Aggiunta film


```
{
  "title": "Interstellar",
  "description": "Un viaggio epico nello spazio.",
  "year": 2014,
  "cover": "interstellar.jpg",
  "category_id": 1
}
```

Aggiunta episodio


```
{
  "title": "Il Sottosopra",
  "episode_number": 1,
  "season_number": 1,
```

"series_id": 1

}

Esempio di Flusso di Login e Autenticazione JWT

1. L'utente hash(a) la propria mail e password e chiama GET /accedi/{utente}/{hash}.
2. Il server verifica le credenziali, genera un JWT firmato.
3. Il token va inserito nell'header Authorization: Bearer ... nelle richieste successive.
4. Ogni richiesta autenticata verifica il token e carica i permessi dal DB.

Controlli di Sicurezza

- Ogni rotta verifica il ruolo tramite middleware (autenticazione, role:admin)
- Risposte sempre con errori chiari (401, 403) se non autorizzato
- Non è possibile accedere o modificare dati non autorizzati

Test

Tutte le rotte sono state testate con Postman:

- Registrazione e login
- Accesso alle risorse da utente
- Inserimento/modifica/eliminazione da admin
- Accesso non autorizzato gestito correttamente
- Flusso JWT dimostrato

Backup e File Consegnati

- Cartella progetto Laravel
- Backup SQL
- Documentazione PDF
- File con link alle API REST e al video dimostrativo

Conclusione

Queste API sono state progettate per essere sicure, scalabili ed estendibili, con una netta separazione dei permessi per ruolo e la sicurezza al centro di ogni operazione.