Disney Characters

Proiect Tehnologii Web

Lamba David-Alexandru

Tatomir Mihail-Cristian

Grupa:445B

Cuprins

1.Descrierea generala a proiectului

Disney Characters Management System este o aplicație web de tip Full-Stack (Angular & Spring Boot) dezvoltată cu scopul de a oferi o soluție digitală modernă pentru administrarea unei baze de date preexistente de personaje și producții Disney. Proiectul transformă un set de date static, furnizat sub forma unui script SQL complex, într-o platformă interactivă care permite utilizatorilor să vizualizeze, să filtreze și să modifice informațiile în timp real.

Sistemul este organizat pe mai multi piloni de funcționalitate, meniți să acopere tot fluxul de gestionare a datelor:

1. **Gestiunea Resurselor (CRUD):**
    - **Vizualizare:** Afișarea listei complete de personaje Disney într-un format tabelar organizat.
    - **Adăugare/Editare:** Formulare dinamice care permit introducerea de noi producții sau actualizarea celor existente (eroi, antagoniști, coloane sonore).
    - **Ștergere:** Posibilitatea de a elimina înregistrări, menținând integritatea bazei de date MySQL.
2. **Sistemul de Filtrare și Căutare Avansată:**
    - Permite utilizatorului să identifice rapid informații specifice folosind filtre pentru titlul filmului sau numele personajului principal. Această funcție este esențială pentru navigarea eficientă printre sutele de înregistrări din scriptul SQL primit.
3. **Securitate și Control Acces:**
    - Implementarea unui modul de autentificare (Login/Register) bazat pe token-uri de securitate (JWT).
    - Diferențierea drepturilor: utilizatorii autentificați pot gestiona datele, în timp ce interfața este protejată împotriva accesului neautorizat.
4. **Arhitectura de Comunicare (REST API):**
    - Utilizarea Spring Boot pentru a crea puntea de legătură între baza de date MySQL și interfața Angular, asigurând un transfer de date rapid și sigur.

2. Tehnologiile utilizate

Frontend

Technology	Version	Purpose
Angular	17	Frontend framework
TypeScript	5.2	Type-safe JavaScript
RxJS	7.8	Reactive programming
Standalone Components	-	Modern Angular architecture
HTTP Interceptors	-	Auto JWT injection
Route Guards	-	Access control

Backend

Technology	Version	Purpose
Java	17	Programming language
Spring Boot	3.1.5	Application framework
Spring Security	Latest	JWT authentication & authorization
Spring Data JPA	Latest	Database ORM
MySQL	8.0+	Database
Maven	3.6+	Build tool
Lombok	Latest	Reduce boilerplate
JJWT	0.11.5	JWT token handling
Gson	Latest	JSON processing

3. Structura datelor - prezentarea campurilor din setul de date

```
{
  "name": "Character Name",
  "url": "https://api.disneyapi.dev/characters/123",
  "image": "...",
  "films": ["Film 1", "Film 2"],
  "shortFilms": ["Short Film"],
  "tvShows": ["TV Show 1"],
  "videoGames": ["Game 1"],
  "parkAttractions": ["Attraction 1"],
  "allies": ["Ally 1"],
  "enemies": ["Enemy 1"]
}
```

Informațiile despre personaje sunt stocate într-o singură coloană de tip JSON într-o tabelă numită data.

Structura datelor pentru un personaj este următoarea:

1. name: Numele personajului
2. url: Adresa URL a API-ului pentru personaj
3. image: Imaginea personajului (în format base64)
4. films: O listă de filme în care apare personajul
5. shortFilms: O listă de scurtmetraje
6. tvShows: O listă de emisiuni TV
7. videoGames: O listă de jocuri video
8. parkAttractions: O listă de atracții din parcuri tematice
9. allies: O listă de aliați
10. enemies: O listă de inamici

Așadar, câmpurile pentru un personaj sunt cheile din acest obiect JSON. Tabela data din baza de date are doar două coloane: id și data (care conține obiectul JSON de mai sus).

4.Utilizarea inteligentei artificiale - detalierea modului in care inteligenta artificiala a fost folosita pentru realizarea proiectului

În dezvoltarea sistemului de gestiune pentru personajele Disney, Inteligența Artificială (IA) a fost integrată ca un asistent tehnic avansat, având un rol de suport în procesele decizionale și de execuție. Utilizarea IA nu a înlocuit efortul de programare, ci l-a optimizat, oferind soluții la probleme tehnice complexe și accelerând implementarea componentelor repetitive.

IA a funcționat ca un partener de dialog pentru validarea structurii proiectului. În faza inițială, am utilizat modelele de limbaj pentru a primi sfaturi privind cea mai bună modalitate de a integra scriptul SQL primit în logica de backend.

- **Sfaturi pentru Mapare:** IA a oferit recomandări despre cum să structurez entitățile Java astfel incat să fie compatibile cu specificațiile JPA.
- **Arhitectura Angular:** Am solicitat sfaturi privind organizarea componentelor pentru a respecta bunele practici de modularizare, asigurând o separare clară între serviciile de date și elementele vizuale de interfață.

Pentru a crește viteza de dezvoltare, am delegat către IA sarcini care implicau cod repetitiv permițându-mi să mă concentrez pe logica de business a aplicației:

- **Generarea de Cod Repetitiv:** IA a fost utilizată pentru generarea automată a metodelor standard de tip Getter, Setter, și a structurilor de bază pentru Controller-ele REST. Acest lucru a redus considerabil timpul alocat scrierii manuale a codului care nu implică logică complexă.
- **Configurări Maven și Dependențe:** Am apelat la IA pentru a identifica versiunile compatibile ale librăriilor necesare (Spring Security, JWT, MySQL Driver), evitând astfel conflictele de versiuni care ar fi putut apărea în fișierul pom.xml.

Unul dintre cele mai importante roluri ale IA a fost în etapa de testare și eliminare a bug-urilor. În momentele în care codul nu producea rezultatul așteptat, IA a servit ca instrument de diagnosticare:

- **Analiza Stack Trace-ului:** În cazul erorilor de rulare (precum *Hibernate Mapping Exceptions* sau *CORS Policy Errors*), am utilizat IA pentru a interpreta mesajele de eroare și pentru a găsi soluții specifice mediului meu de dezvoltare.

- **Optimizarea SQL-ului:** Deoarece am lucrat cu un script SQL preexistent, IA m-a ajutat să depanez interogările care nu returnau datele corecte sau care aveau performanțe scăzute, sugerând indexări sau modificări ale sintaxei JPQL.

Spre finalul proiectului, am utilizat IA pentru a primi sugestii de "refactoring" – optimizarea codului existent pentru a fi mai curat și mai ușor de întreținut. Aceasta a identificat bucăți de cod care puteau fi simplificate și a oferit sfaturi pentru gestionarea mai sigură a excepțiilor, crescând astfel robustetea generală a sistemului.

5. Concluzii - concluzii personale, dificultati intampinate si solutii implementate

Finalizarea proiectului **Disney Characters Management System** a reprezentat o etapă esențială în procesul nostru de învățare, oferindu-ne ocazia de a colabora pentru a construi o aplicație Full-Stack modernă. Pornind de la o bază de date stabilă și ajungând la o interfață utilizator interactivă, am reușit să integrăm fluxuri de date complexe într-o soluție unitară.

Acest proiect ne-a permis să înțelegem importanța unei arhitecturi bine definite și a comunicării constante în cadrul unei echipe de dezvoltare. Lucrul cu un stack tehnologic format din Angular, Spring Boot și MySQL ne-a forțat să privim aplicația ca pe un sistem integrat, unde fiecare componentă trebuie să se sincronizeze perfect cu celelalte. Am învățat că succesul unui proiect software depinde nu doar de scrierea codului, ci de modul în care gestionăm fluxul de informație între server și client. Transformarea scriptului SQL static într-un instrument de gestiune dinamic a fost o experiență valoroasă care ne-a testat abilitățile de analiză și implementare.

Pe parcursul implementării, ne-am confruntat cu o serie de obstacole tehnice care au necesitat sesiuni comune de debugging și studiu aprofundat:

- **Complexitatea Structurii Angular:** O dificultate majoră a fost adaptarea la arhitectura modulară și riguroasă a framework-ului Angular. Gestionarea componentelor, a serviciilor și a rutelor, menținând în același timp o structură curată a fișierelor, a reprezentat o provocare pentru noi. Organizarea modulelor pentru a asigura o încărcare eficientă și gestionarea stării aplicației au fost procese care au necesitat o atenție sporită.
- **Erori de Comunicare Frontend-Backend:** Conectarea aplicației Angular cu API-ul Spring Boot a generat cele mai multe probleme de integrare. Ne-am lovit de erori

frecvente de tipul **CORS (Cross-Origin Resource Sharing)**, care blocau transferul de date din motive de securitate între porturile de dezvoltare. De asemenea, am întâmpinat dificultăți în sincronizarea formatului JSON trimis de server cu interfețele TypeScript din frontend.

- **Gestiunea Sesiunilor și Securitatea:** Implementarea și sincronizarea token-urilor JWT între cele două medii a fost dificilă, mai ales în ceea ce privește protejarea rutelor și menținerea utilizatorului autentificat după reîmprospătarea paginii.

Fiecare barieră tehnologică a fost depășită prin aplicarea unor soluții tehnice standardizate:

- **Modularizarea și Serviciile Angular:** Pentru a simplifica structura proiectului, am extras logica de comunicare în servicii dedicate (CharacterService). Această abordare ne-a permis să păstrăm componentele vizuale cât mai simple, delegând responsabilitatea datelor către servicii specializate.
- **Configurarea Globală CORS și Proxy:** Pentru a elimina erorile de comunicare, am implementat o clasă de configurare în Spring Boot care permite în mod explicit cererile de la portul local al Angular-ului. Am utilizat, de asemenea, Observables pentru a gestiona fluxurile de date asincrone, asigurându-ne că interfața rămâne responsivă indiferent de timpul de răspuns al serverului.
- **Utilizarea Interceptorilor HTTP:** Am creat interceptori în Angular care injectează automat token-ul de securitate în header-ul fiecărei cereri către backend. Această soluție a eliminat codul repetitiv și a securizat întreaga comunicare a aplicației noastre.