UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

TV Shows Tracker

Name: Handaric Cosmin
Group: 30235

Table of Contents

Deliverable 1

Project Specification

Aplicație web destinată comunității amatoare de seriale ce oferă, în principiu două servicii importante: serviciul de tracking și social media. Prin serviciul de tracking utilizatorii pot să-și gestioneze serialele pe care le urmăresc sau doresc să le urmărească, iar prin cel de social media fanii vor putea să-și împărtășească reacțiile și opiniile la diferite episoade.

Functional Requirements

Principalele funcționalități ale proiectului sunt împărțite în 4 categorii, sub formă de assignment-uri:

Assignment 1:

- Înregistrare utilizator nou (Register);
- Login și logout;
- Folosirea API-ul de la TMDB pentru inserarea de seriale și episoade în baza de date;
- Implementarea operațiilor CRUD pentru admin (Admin panel);
- Editarea detaliilor din profil (*Display name*, *About me*, *Birthday*, *Gender*, *Social media*).

Assignment 2:

- Căutare seriale noi și posibilitatea de a adăuga seriale în lista *To Watch*;
- Integrare YouTube Data API pentru a vizualiza trailere la seriale;
- Adăugare serial în listele: *Favorites*/*Stopped*;
- Marcare episod și actualizare liste: *Watching*/*Up-To-Date*/*Finished*.

Assignment 3:

- Căutare prieteni și trimitere cerere de prietenie;
- Acceptare/respingere cerere de prietenie;
- Vizualizare profil prieten (un user poate sa vada listele unui alt user doar daca e prieten cu acesta);
- Evaluare episod și adăugarea unui comentariu de către user;
- Vizualizare profil prieten (un user poate sa vada listele unui alt user doar daca e prieten cu acesta).

Assignment 4:

- Posibilitatea de a aprecia un comentariu;
- Organizarea comentariilor de la un episod pe bază aprecierilor;
- Raportare bug/comentariu (spoiler alert, comentariu ofensator);
- Exportare date XML - rapoarte;
- Notificări prin email: recuperare parolă, contul a fost creat cu succes, comentariul tău a fost etichetat ca spoiler, comentariul tău a fost șters de către user sau admin (comentariu ofensator/spam/probleme de drepturi de autor).

Use Case Model

Use Cases Identification

Use case: Adăugarea unui nou serial în baza de date
Level: user-goal
Primary Actor: Admin
Main success scenario:

- ✓ se caută serialul pe TMDB pentru a afla ID-ul;
- ✓ se introduce ID-ul respectiv într-un search bar;
- ✓ se preiau informațiile serialului și episoadelor din TMDB;
- ✓ se face confirmarea adăugării;
- ✓ serialul și episoadele au fost adăugate în baza de data a aplicației.

Extensions:

- serialul există deja în baza de date;
- nu se face confirmare adăugării;
- se introduce alt ID înainte de confirmare;
- probleme la conexiunea cu API-ul.

Use case: Register
Level: sub-function
Primary Actor: User (neînregistrat)
Main success scenario:

- ✓ se accesează pagina de înregistrare;
- ✓ se introduc datele de înregistrare: username, email, parolă și confirmare parolă;
- ✓ se trimite cererea de înregistrare;
- ✓ user-ul este înregistrat;
- ✓ se face redirecționarea către pagina de login.

Extensions:

- username-ul sau email-ul există deja în baza de date;
- email-ul nu are formatul corect;
- câmpurile parolă și confirmare parolă nu corespund;
- parola conține exclusiv caractere numerice, are mai puțin de 8 caractere sau este o parolă comună.

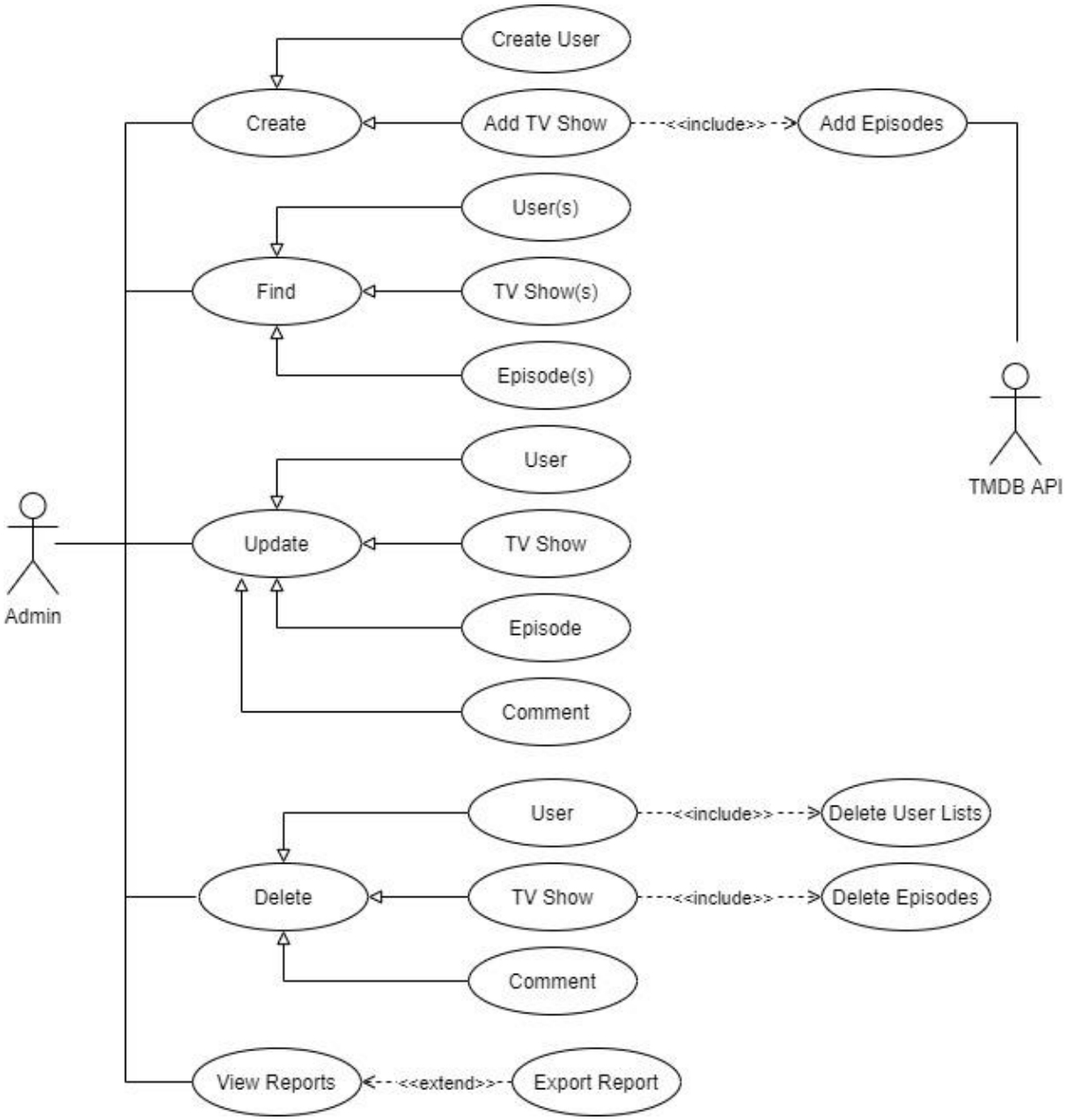
Use case: Adăugarea unui comentariu
Level: user-goal
Primary Actor: User (înregistrat)
Main success scenario:

- ✓ se marchează episodul ca fiind episod văzut;
- ✓ se accesează rubrica de comentarii;
- ✓ se adaugă un comentariu la episod.

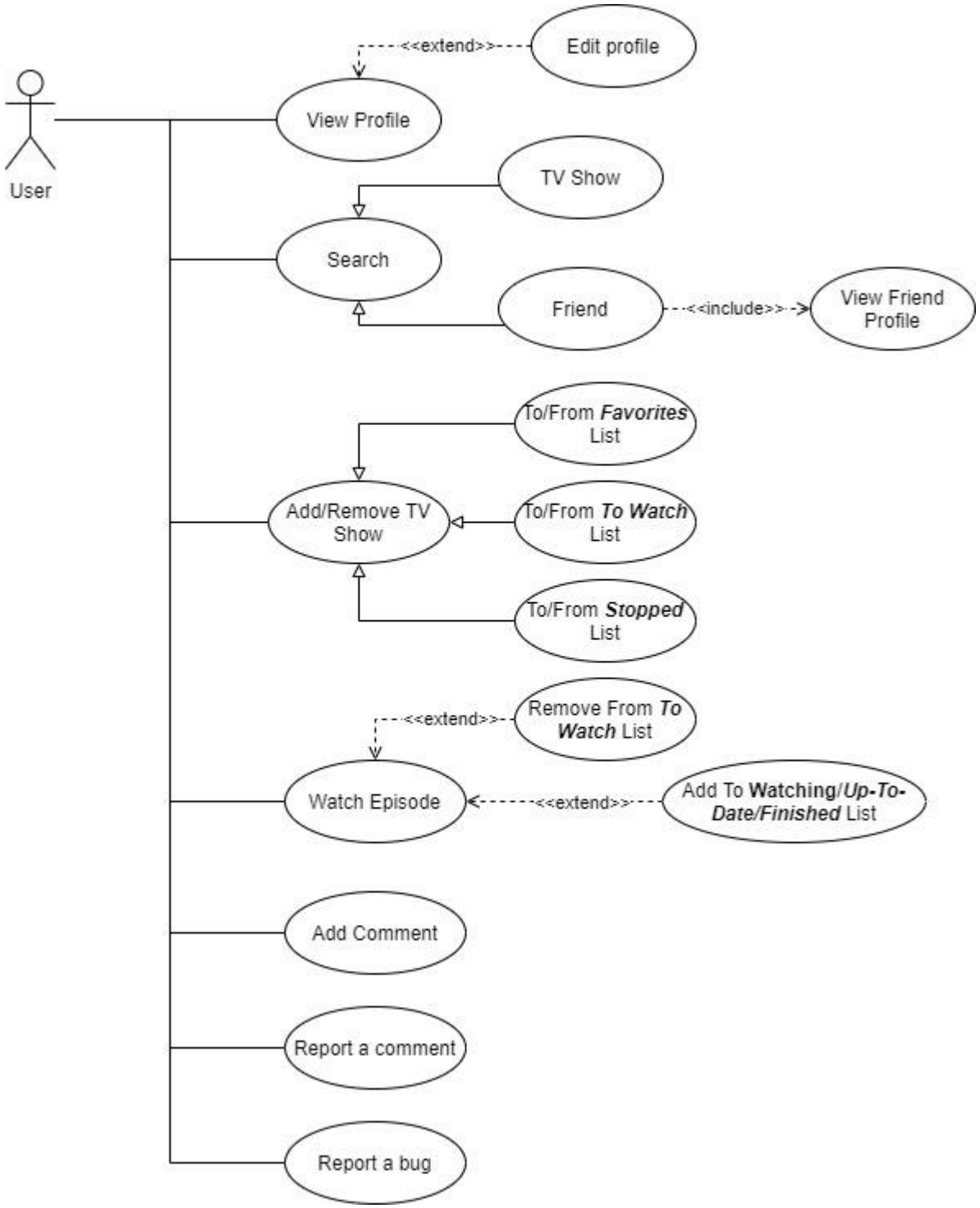
Extensions:

- comentariul poate fi marcat ca fiind *spoiler alert* de către admin;
- comentariul poate fi șters din cauza conținutului ofensator.

UML Use Case Diagrams



Use Case Diagram – Admin



Use Case Diagram – User

Supplementary Specification

Non-functional Requirements

1. **Portability**. Fără prea multe modificări aplicația poate fi utilizată pe orice sistem de operare. Deoarece aplicația este o aplicație web, aceasta nu necesită un anumit sistem de operare; singurele cerințe ar fi existența unei conexiuni la internet și a unui browser.
2. **Security**. Aplicație trebuie să protejeze datele utilizatorilor. Acest lucru va fi asigurat de către Spring Security care oferă suport pentru procesele de autentificare și autorizare și protecție împotriva a diferite atacuri. In cazul acestei versiuni a aplicației se va folosi Spring Security pentru criptarea parolelor în baza de date.
3. **Scalability**. Site-ul se va putea adapta la o creștere a numărului de utilizatori, iar folosirea librării React pentru implementarea frontend-ului va sprijini această proprietate.
4. **Usability**. Având în vedere serviciile oferite de această aplicație, aceasta trebuie să ofere un design cât mai plăcut și ușor de interpretat pentru utilizator.

Design Constraints

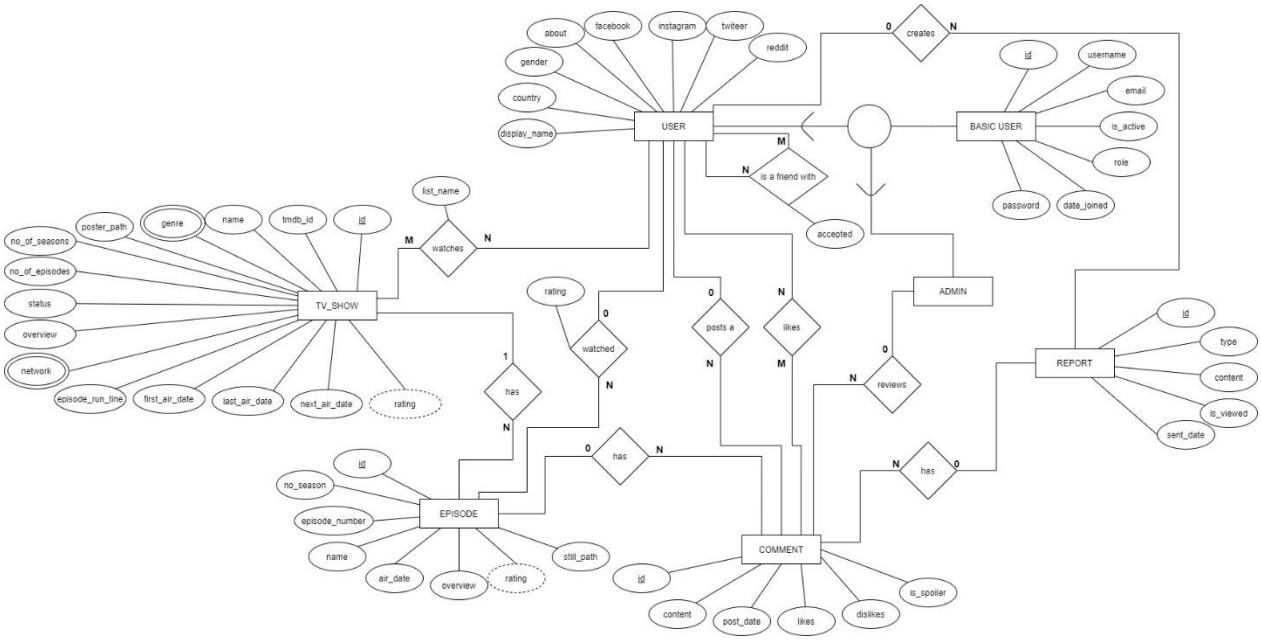
1. Aplicația poate rula doar pe *localhost*.
2. Dependințe de următoarele API-uri:
   - TMDB: pentru adăugarea și actualizarea serialelor și episoadelor din baza de date a aplicației;
   - YouTube Data: pentru vizualizarea de trailere la seriale.

Glossary

- *TMDB* = The Movie Database;
- *To Watch* = lista de seriale pe care user-ul vrea să le vadă;
- *Favorites* = liste de seriale preferate;
- *Stopped* = liste de seriale pe care user-ul nu mai vrea să le vadă;
- *Watching* = lista de seriale pe care user-ul le urmărește;
- *Up-To-Date* = lista de seriale pe care user-ul le urmărește și se află la zi;
- *Finished* = lista de seriale terminate.

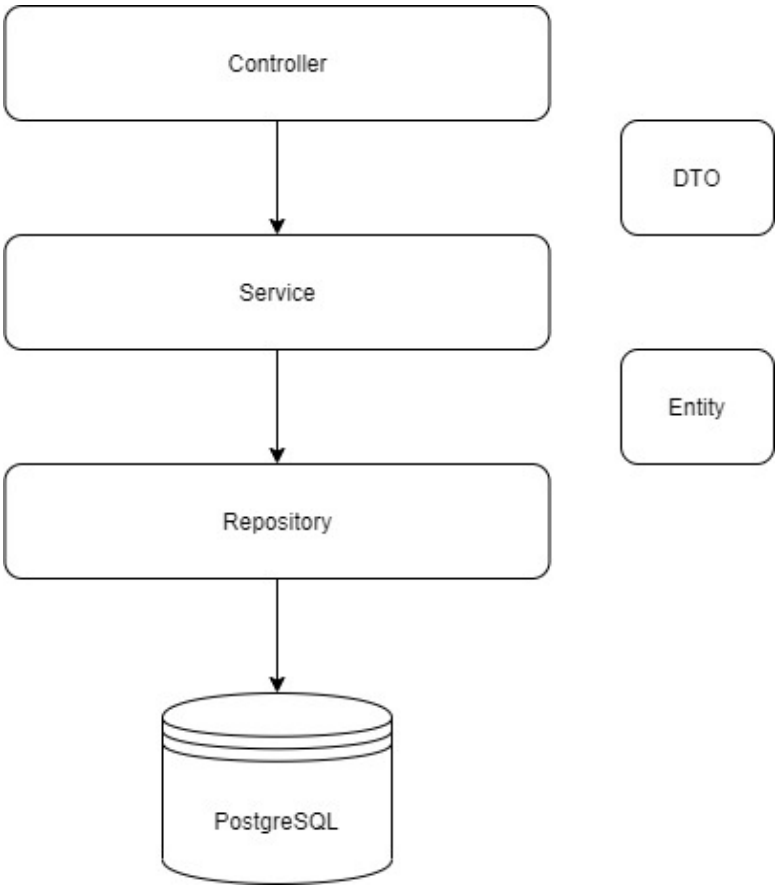
Deliverable 2

Domain Model

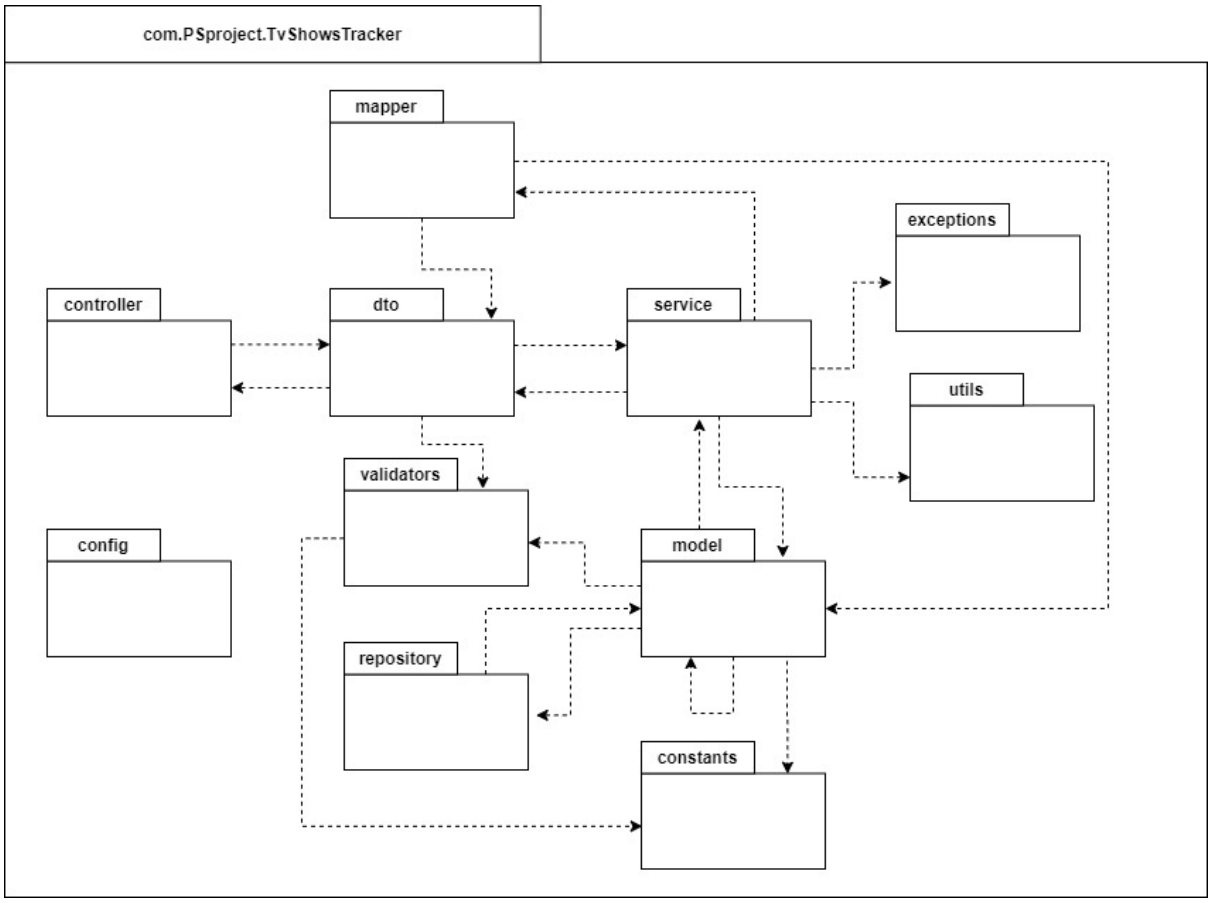


Architectural Design

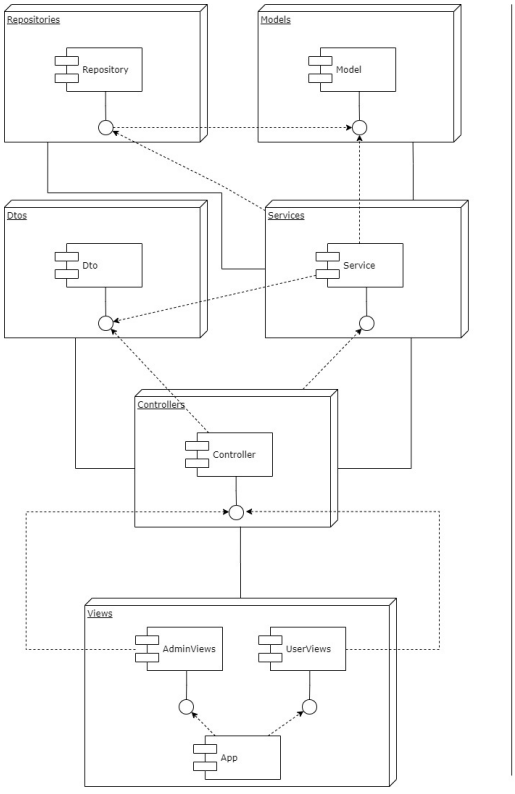
Conceptual Architecture

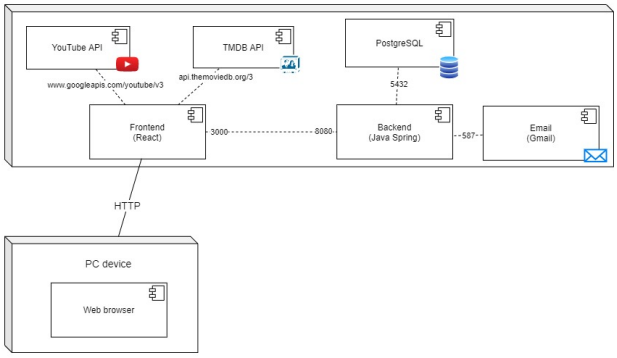


Package Design



Component and Deployment Diagram

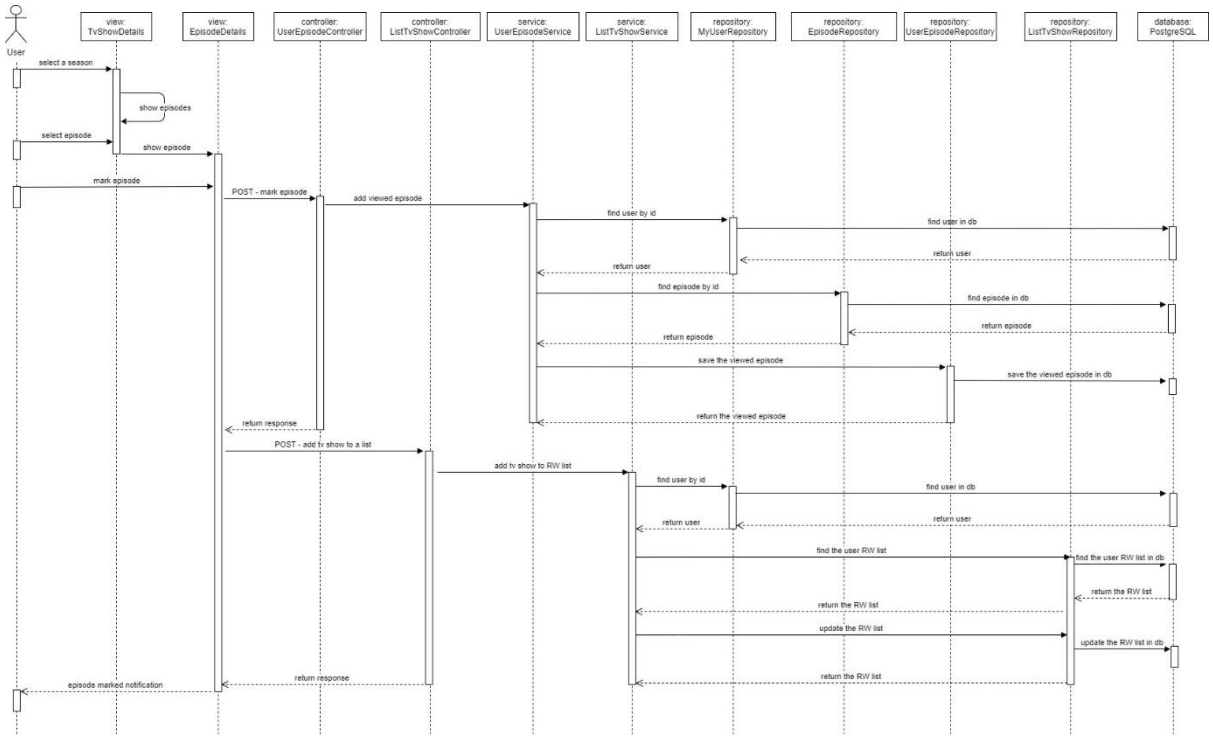




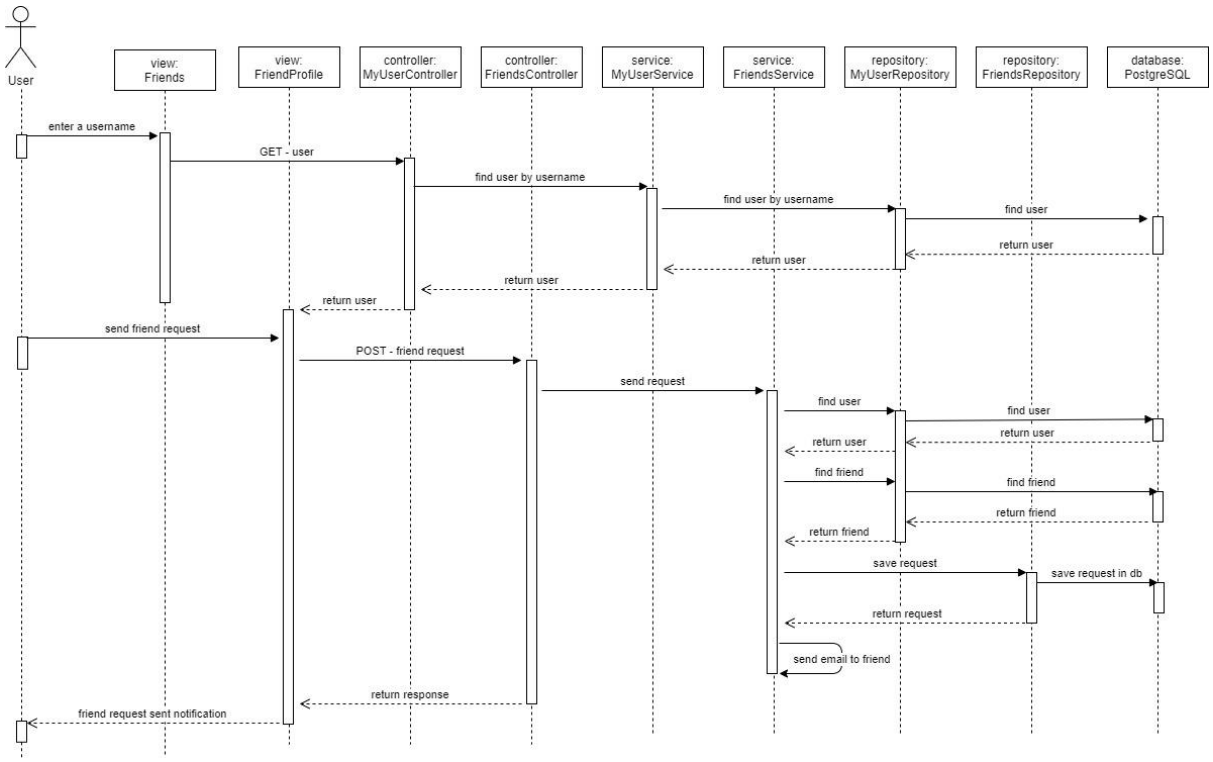
Deliverable 3

Design Model

Dynamic Behavior

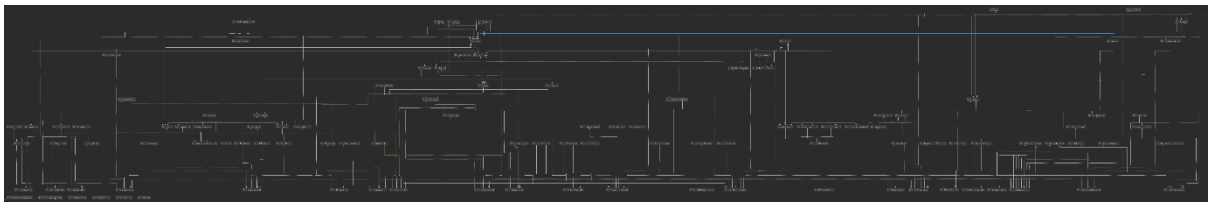


Marcare episod și actualizare listă *Recently Watched*

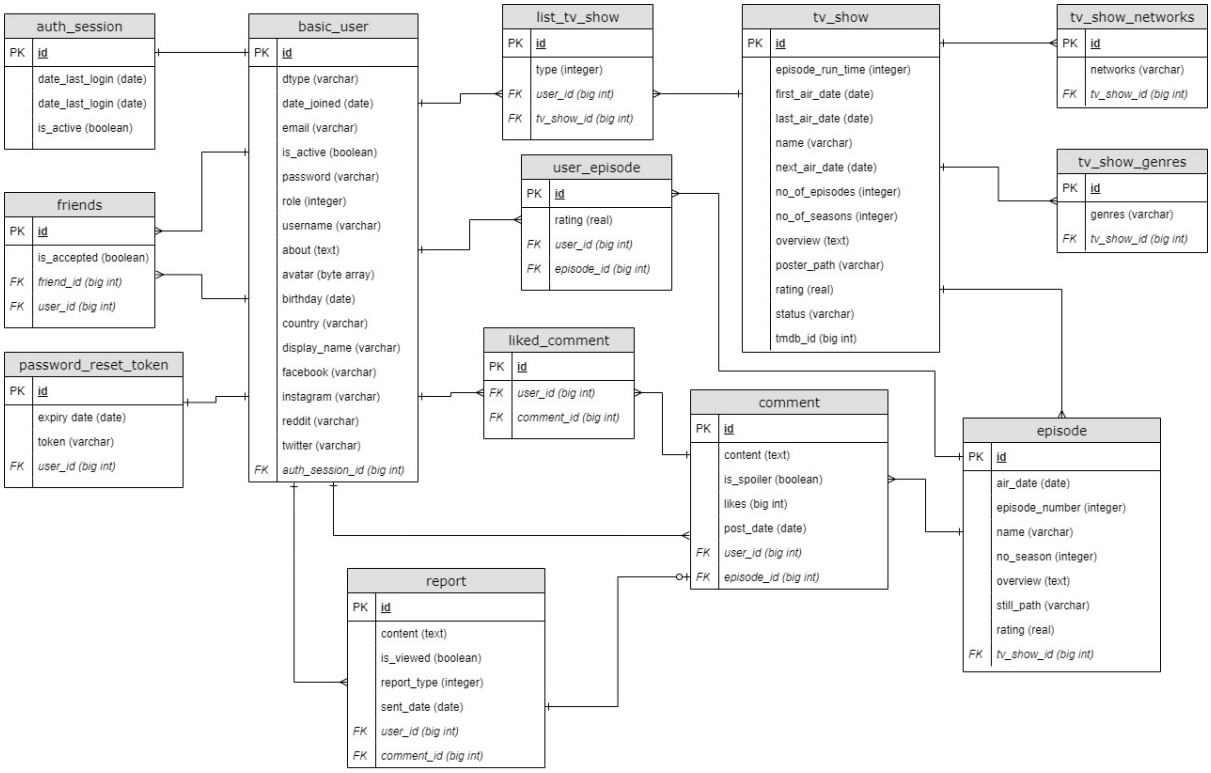


Căutare utilizator, trimitere cerere de prietenie și notificare cerere nouă de prietenie

Class Diagram



Data Model



System Testing

1. `void givenExistingTmdbId_whenFindByTmdbId_thenFindOne()`
   - se verifică existența unui obiect de tip *TvShow* în funcție de variabila instanță *tmdbID*;
   - se dă un ID a unui serial din baza de date celor de TMDB care este deja atribuit unui obiect;
   - se caută obiectul în funcție de valoarea variabilei;
   - se găsește obiectul și se returnează.
2. `void givenNonExistingTmdbId_whenFindByTmdbId_thenThrowException()`
   - se verifică cazul în care se caută un obiect *TvShow* după variabilă instanță *tmdbID*;
   - se dă un ID a unui serial din baza de date celor de TMDB care nu este atribuit unui obiect;
   - se caută obiectul în funcție de valoarea variabilei;
   - nu se găsește obiectul și se returnează *null*;
   - se aruncă o excepție de tip *NullPointerException*.

3. `void createTvShow()`
   - se crează un nou obiect de tip *TvShow*;
   - se apelează constructorul fără nici un parametru;
   - se setează numele și id-ul din TMDB al obiectului creat;
   - se adăugă obiectul;
   - verificăm că obiectul adăugat nu este *null*;
   - se verifică dacă s-a adăugat obiectul corect prin verificarea celor două câmpuri: *name* și *tmdbID*.
4. `void deleteTvShow()`
   - se șterge un nou obiect de tip *TvShow*;
   - se ia un obiect deja creat și se șterge;
   - verificăm că obiectul șters nu este *null*;
   - se verifică dacă s-a șters obiectul corect prin verificarea celor două câmpuri pe care le cunoșteam: *name* și *tmdbID*.

Future Improvements

- Actualizarea automată (la un interval de timp) a serialelor când apare un episod nou.
- Distribuirea comentariilor pe rețele de socializare.
- Replay la comentarii.
- Postare comentarii cu imagini/gifs.
- Chat.
- Implementarea unui sistem de recomandări.

Conclusion

Realizarea acestui proiect a reprezentat un mod optim și eficient de a învăța atât procesul de proiectare și implementare unei aplicații web, cât și folosirea framework-ul Spring pe partea backend și bibliotecii open-source ReactJS pe partea de frontend.

Partea de backend development m-a ajutat să îmi consolidez cunoștințele de Java (folosirea de stream-uri, design patter pentru generarea de fișiere xml/txt, integrarea de email-uri) și să aflu modul de implementare a părții de server a aplicații (folosirea de controllere pentru a request-uri, de service-uri pentru prelucrarea request-urilor, de repository-uri pentru accesul la baza de date, de modeluri pentru reprezentarea entităților, de dto-uri pentru transferul de date, de websockets pentru comunicare full-duplex și de Swagger pentru a descrierea metodelor de request pentru server de către client).

Partea de frontend development a fost o inițiere către JavaScript, JSX și framework-uri de CSS (SemanticUI, MaterialUI) și o oportunitatea de a vedea cum se implementează partea de client a aplicației și de a folosi diverse API-uri (TMDB, YouTube).

Bibliography

[1] Many-To-Many Relationship in JPA: https://www.baeldung.com/jpa-many-to-many
[2] Hibernate Many to Many Annotation Tutorial: https://www.baeldung.com/hibernate-many-to-many
[3] One-to-One Relationship in JPA: https://www.baeldung.com/jpa-one-to-one
[4] Hibernate Tips: How to persist a List of Strings as an ElementCollection: https://thorben-janssen.com/hibernate-tips-elementcollection/
[5] TMDB API: https://developers.themoviedb.org/3
[6] YouTube API: https://developers.google.com/youtube/v3
[7] SematicUI: https://semantic-ui.com
[8] MaterialUI: https://material-ui.com
[9] Template Login page: https://material-ui.com/getting-started/templates/
[10] React File Upload/Download example with Spring Boot Rest Api: https://bezkoder.com/react-file-upload-spring-boot/
[11] Get image from Java spring to show in React: https://stackoverflow.com/questions/61321790/get-image-from-java-spring-to-show-in-react
[12] Guide to Spring Email: https://www.baeldung.com/spring-email
[13] Spring Security – Reset Your Password: https://www.baeldung.com/spring-security-registration-i-forgot-my-password
[14] Encode, Decode, Validate using BCryptPasswordEncoder in Spring Boot Security: https://www.yawintutor.com/encode-decode-using-bcryptpasswordencoder-in-spring-boot-security/