UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Filmento

Name: Gligor Mihai
Group: 302310

Table of Contents

Deliverable 1

Project Specification

[Present the project specification.]

Filmento este un site pentru iubitorii de filme care permite navigarea intre cele mai populare titluri ale momentului si creearea unei liste de favorite prin care utilizatorul poate avea in acelas loc toate operele cinematografice preferate. De asemenea acesta permite si publicarea unei opinii prin intermediul sistemului de rating si prin scrierea unui review. Ideea proiectului este asemanatoare cu aplicatia/site ul IMDB .

Functional Requirements

[Present the functional requirements.]

- Inregistrare : pentru ca user ul sa poata folosii site ul filmento , acesta trebuie sa aiba mai intai un cont , acest lucru fiind usor realizabil prin intermediul paginii de access . Mai intai clientul trebuie sa introduca un email valid , dupa care sa introduca parola Dorita si sa o repete pentru a se asigura ca a introdus parola correct.
- Login : aceasta functionalitate permite accesul clientului la aplicatia filmento si se realizeaza prin intermediul paginii de access prin introducere email-ului si a parolei corespunzatoare contului .
- List : Fiecare utilizator , odata ajuns pe pagina unui anumit film poate sa il adauge la lista lui personala de filme care va fii salvata in baza de date .
- Rating : utilizatorii vor putea sa “aprobe” sau “ dezaprobe” un film. Alegerea acestora va fii salvata intr-o baza de date si va fi afisat procentul de oameni care au aprobat pozitiv filmul resspectiv ,astfel userul avand access la opinia publicului cu privire la film
- Review : utilizatorii vor putea scrie cate un review pentru fiecare film care va fii salvat public pe pagina acestuia .De asemenea acesta va putea vizualiza review urile altor useri
- Movie Browsing : utilizatorii vor putea vizualiza un catalog intreg de filme , fiecare cu pagina lui individuala care contine informatii despre acesta.
- Movie Searching: utilizatorii vor avea posibilitate de a sorta filmele in functie de genul lor, sau de a le cauta dupa nume.

Use Case Model 1

Use Cases Identification

Use-Case: adauga filmul la lista

Level: adaugarea unui film in lista de filme preferate

Primary Actor: user inregistrat

Main success scenario: clientul adauga cu success un film in lista personala

Extensions:

Use Case Model 2

Use Cases Identification

Use-Case: adaugarea unui review

Level: expunerea opiniei cu privire la film in mod public

Primary Actor: user inregistrat

Main success scenario: clientul scrie si incarca cu success un review pe pagina filmului

Extensions:

Use Case Model 3

Use Cases Identification

Use-Case: cautarea unui film

Level: gasirea unui anumit film in baza de date a site ului

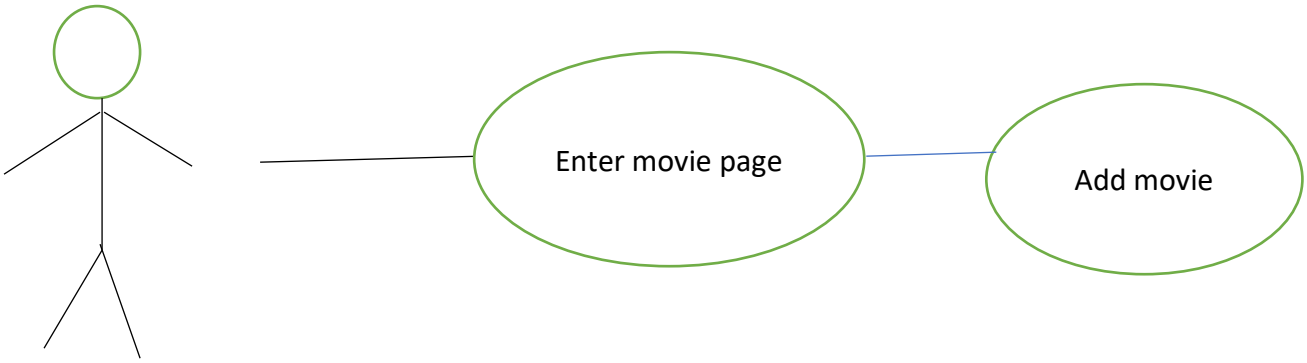
Primary Actor: user inregistrat

Main success scenario: clientul gaseste cu success filmul dorit

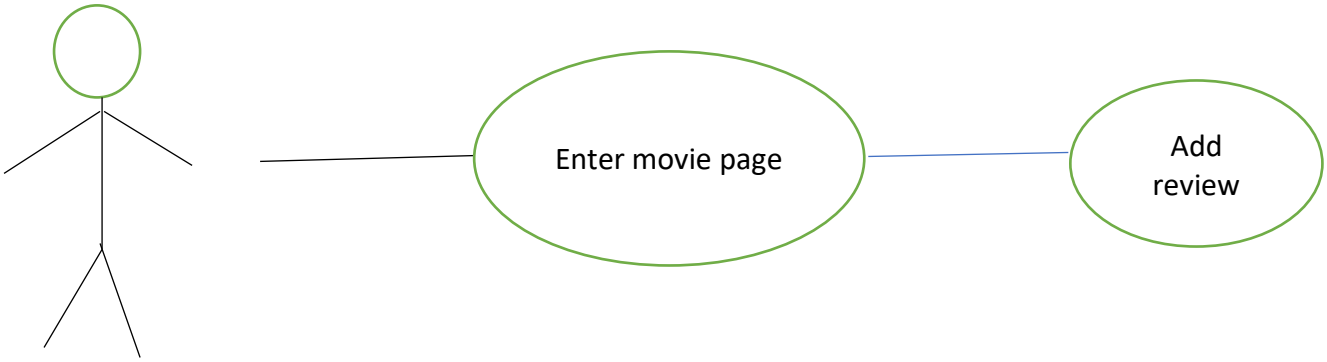
Extensions: gasirea filmului atat prin cautarea numelui cat si dupa gen

UML Use Case Diagrams

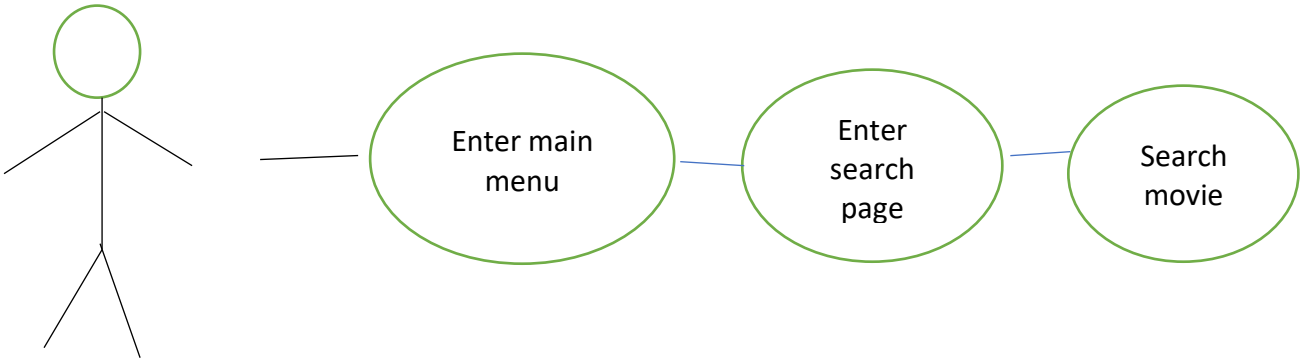
Use case 1

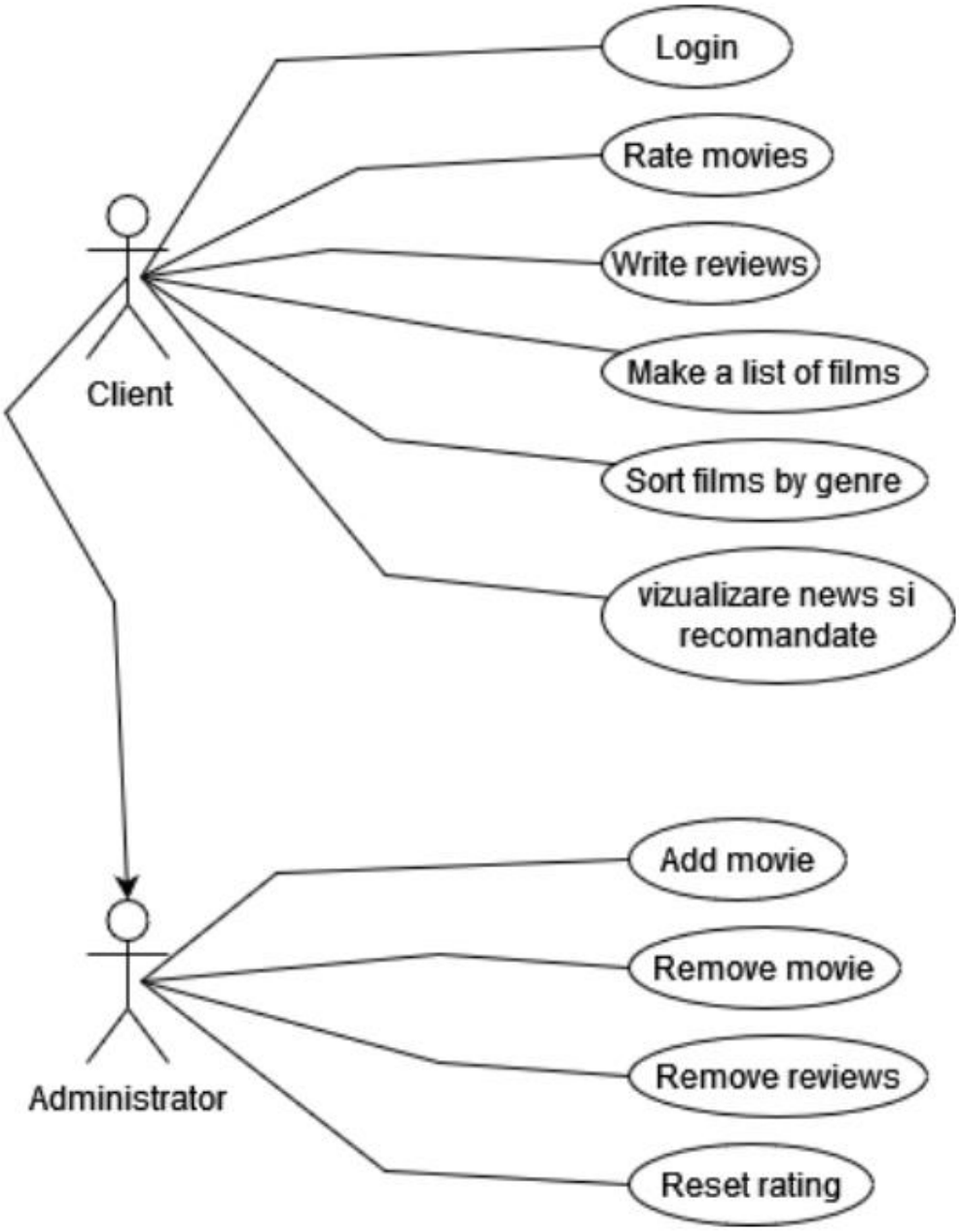


Use case 2



Use case 3





Supplementary Specification

Non-functional Requirements

[Choose 4 NF for your system, describe them and explain why these NF are suitable for your implementation.]

- Performance: Se refera la eficienta viteza si acuratetea cu care aplicatia isi executa functionalitatile . Acest requirement este necesar intr-o aplicatie care lucreaza cu o baza de date care este populate cu elemente complexe precum listele de filme, deoarece , in momentul in care browsing ul intre diferitele filme necesita ca utilizatorul sa astepte incarcarea paginii acesta va cauta o alta aplicatie.

- Transparency: Se refera la actionarea si functionarea intr-una asa mod incat toate functionalitatile sa fie la vedere si sa existe o comunicare onesta si o procesarea a datelor clara si agreeata atat de utilizator cat si de developer.Avand in vedere ca datele necesare unei aplicatii de browsing a filmelor nu sunt sensibile , transparenta va fii un requirement usor de implementat.
- Usability : se refera la capacitatea site ului de a isi indeplinii sarcinile intr-o maniera intuitiva , astfel incat utilizatorul sa poata sa foloseasca aplicatia pentru necesitatile lui , intr-o maniera atat eficienta cat si placuta. Acest requirement este usor implementabil prin intermediul interfetei intuitive a aplicatiei. Este necesar pentru o navigare usoara si placuta in lumea filmelor.
- Flexiblility : se refera la capacitatea aplicatiei de a se descurca cu viitoare schimbari. Acest requirement este necesar aplicatiei deoarece aceasta trebuie updatata frecvent cu noi filme, de asemenea exista multiple viitoare implementari de functionalitati care vor putea imbunatatii calitatea experientei utilizatorului/

Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

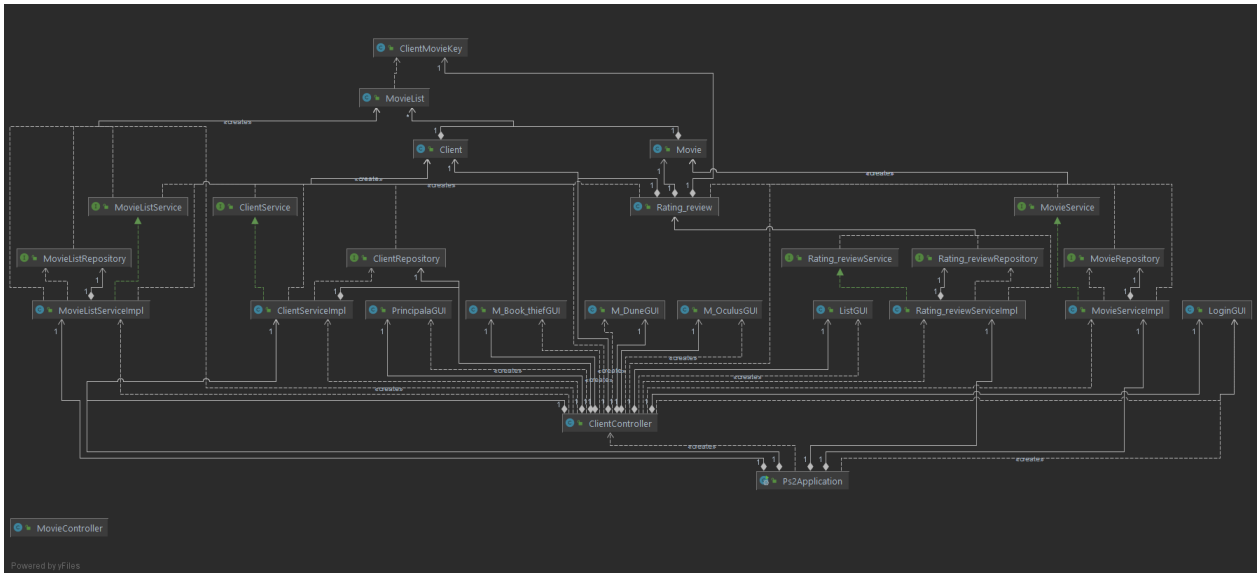
Glossary

[Present the noteworthy terms and their definition, format and validation rules if appropriate.]

Deliverable 2

Domain Model

[Define the domain model and create the conceptual class diagrams]



Architectural Design

Conceptual Architecture

[Define the system's conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]

Arhitectura layered , este organizarea structurii proiectului în patru categorii principale: Prezentare, aplicație, domeniu și infrastructură. Fiecare dintre straturi conține obiecte legate de preocuparea specială pe care o reprezintă.

presentation layer conține toate clasele responsabile pentru prezentarea UI utilizatorului final sau trimiterea răspunsului înapoi la client (în cazul în care operăm adânc în back-end).

aplication layer conține toată logica cerută de aplicație pentru a-și îndeplini cerințele funcționale și, în același timp, nu face parte din regulile domeniului.

domain layer reprezintă domeniul de bază, constând în principal din entități de domeniu și, în unele cazuri, servicii. Regulile de afaceri, cum ar fi invarianții și algoritmii, ar trebui să rămână toate în acest strat.

infrastructure layer (cunoscut și sub numele de stratul de persistență) conține toate clasele responsabile pentru a face lucrurile tehnice, cum ar fi persistența datelor în baza de date, cum ar fi DAO-uri, depozite sau orice altceva pe care îl utilizați.

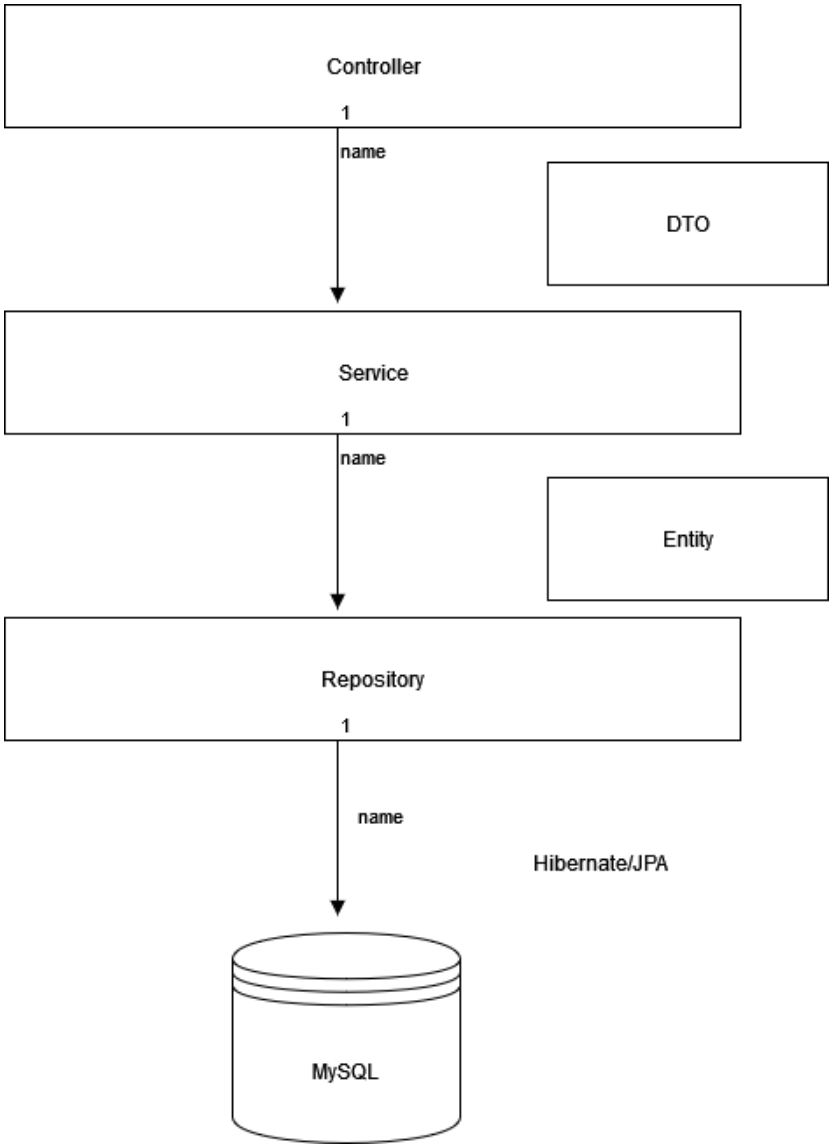
Avantaje

Simplitate – conceptul este foarte ușor de învățat și vizibil în proiect la prima înțelegere.

Consecvent în diferite proiecte – straturile și, prin urmare, organizarea generală a codurilor este cam aceeași în fiecare proiect stratificat.

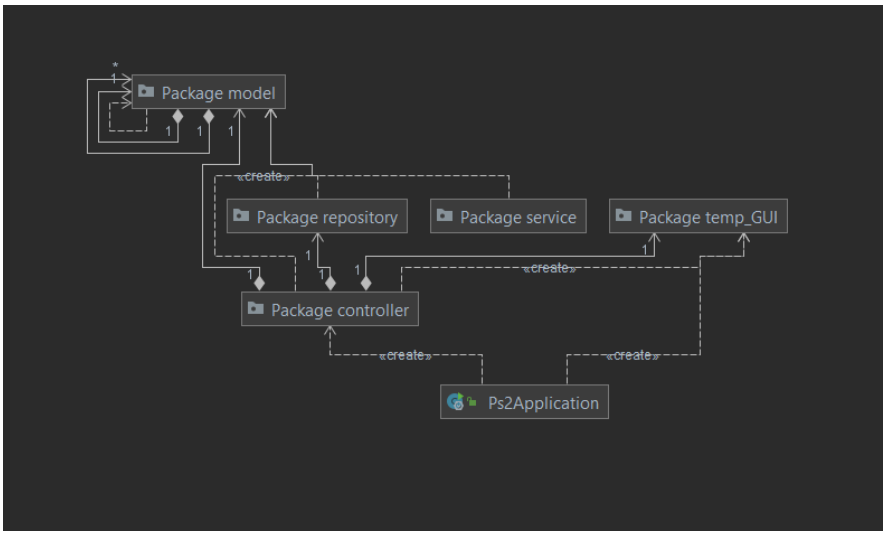
Separarea garantată a preocupărilor - doar preocupările care au un strat și până la punctul în care respectați regulile arhitecturii stratificate, dar este foarte ușor cu organizarea codului pe care o implică.

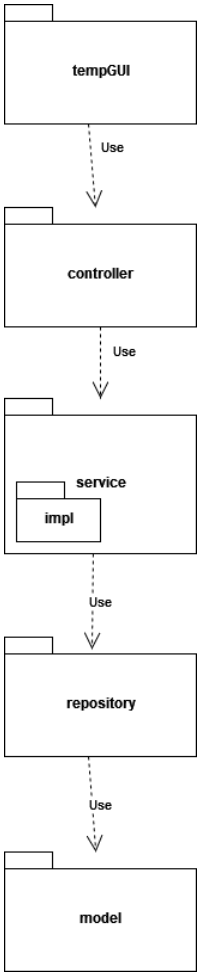
Browsability din punct de vedere tehnic – atunci când doriți să schimbați ceva în unele / toate obiectele de un anumit tip, acestea sunt foarte ușor de găsit și sunt ținute împreună.



Package Design

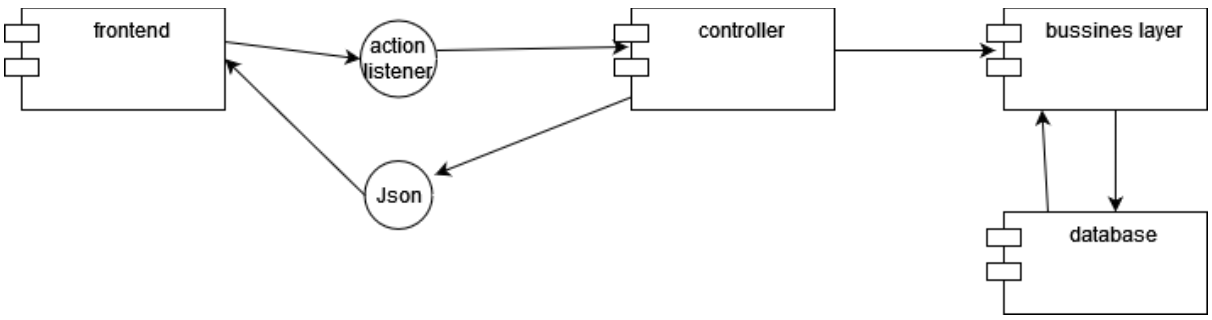
[Create a package diagram]

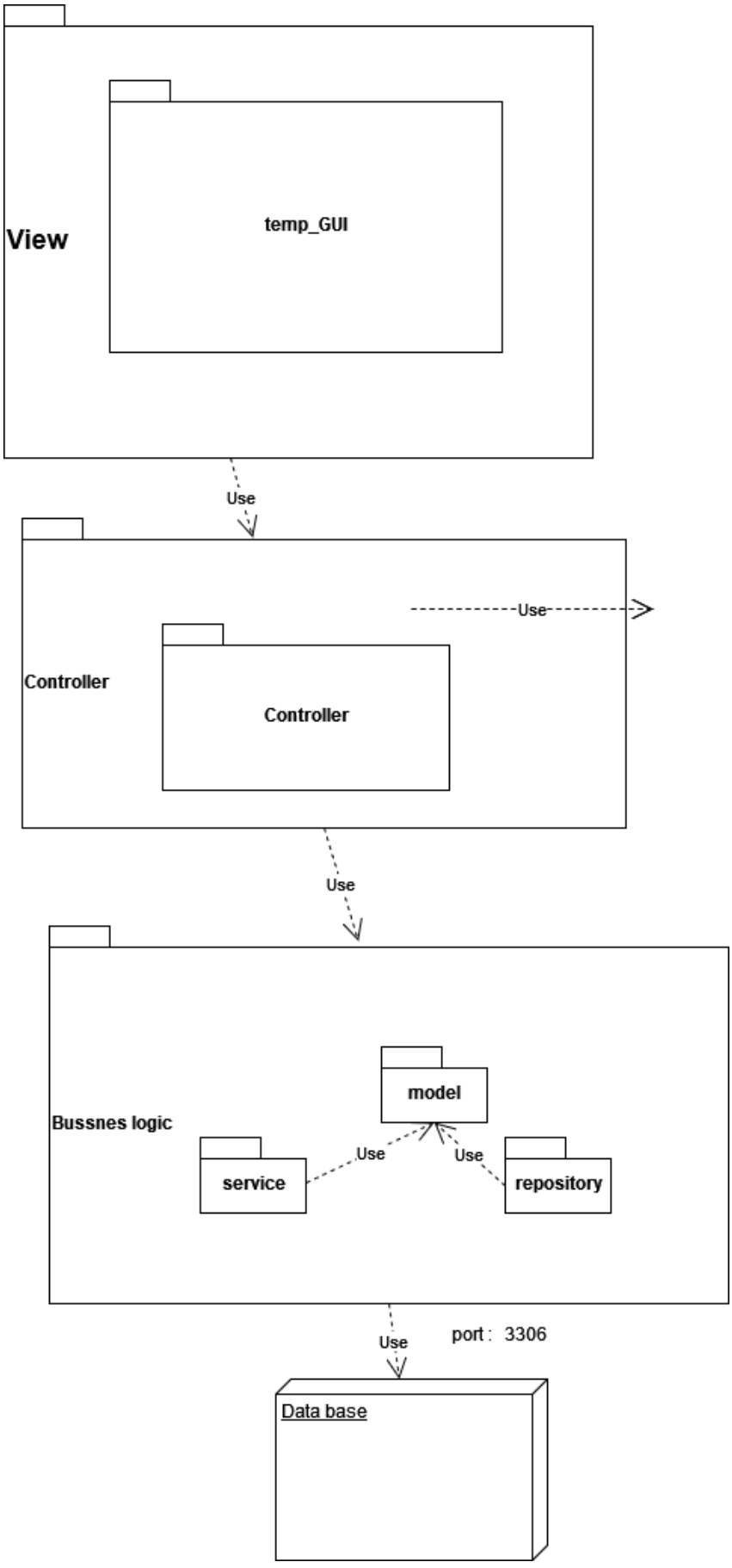




Component and Deployment Diagram

[Create the component and deployment diagrams.]



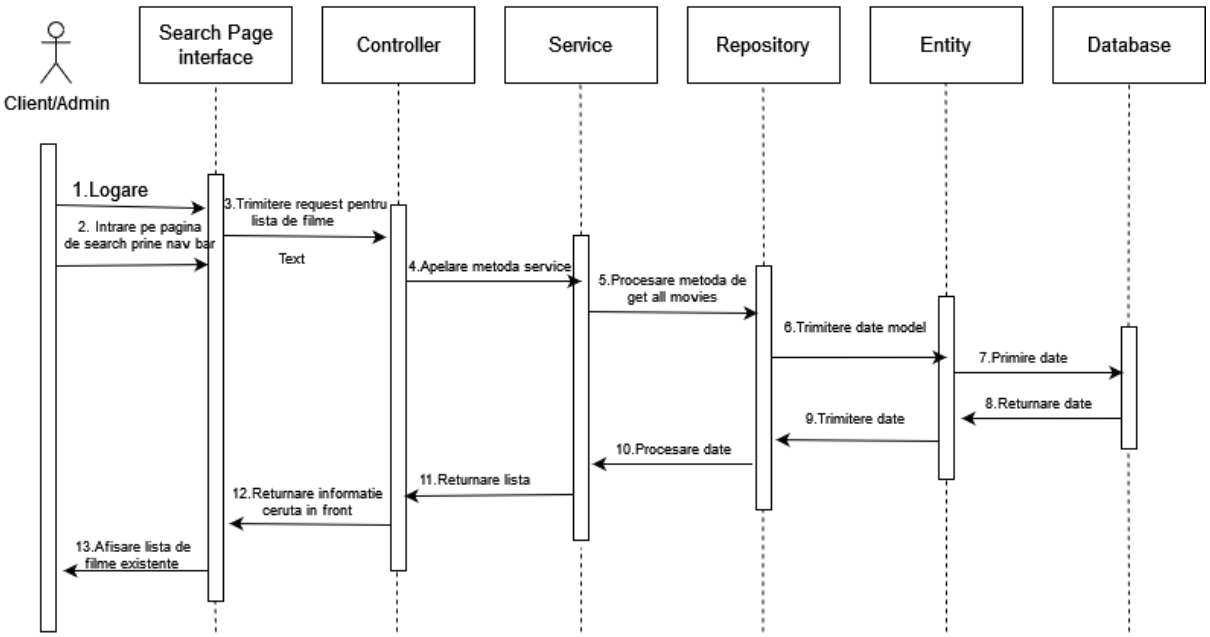

View
temp_GUI
Use
Use
Controller
Controller
Use
Bussnes logic
model
service
Use
Use
repository
Use
port : 3306
Data base

Deliverable 3

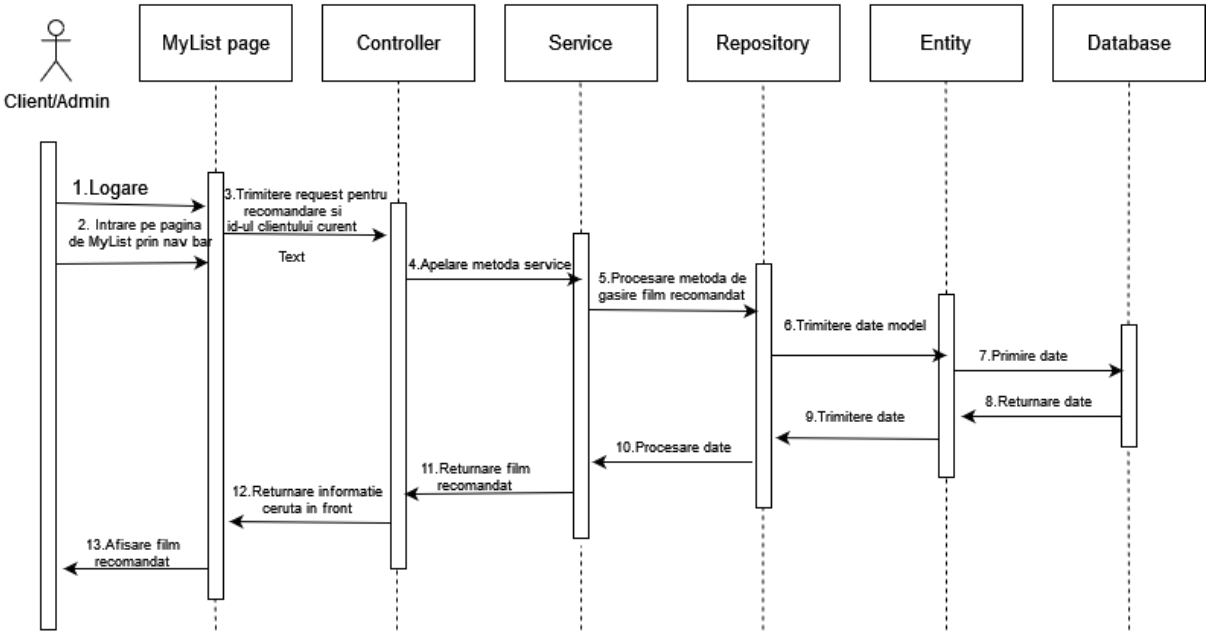
Design Model

Dynamic Behavior

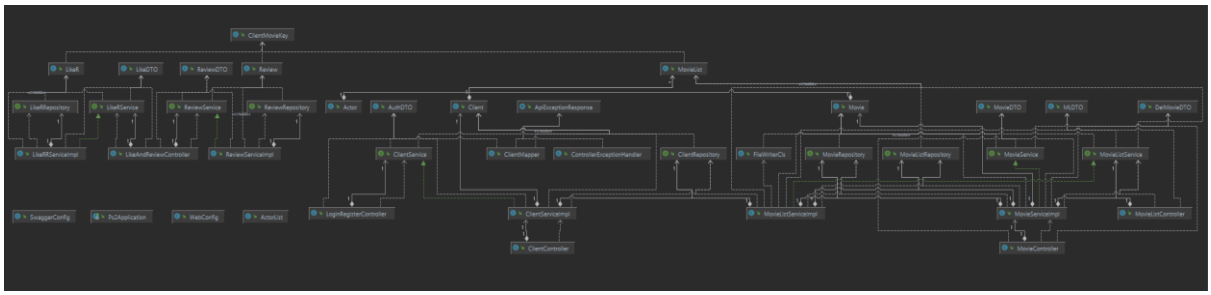
- Vizualizare filme existente

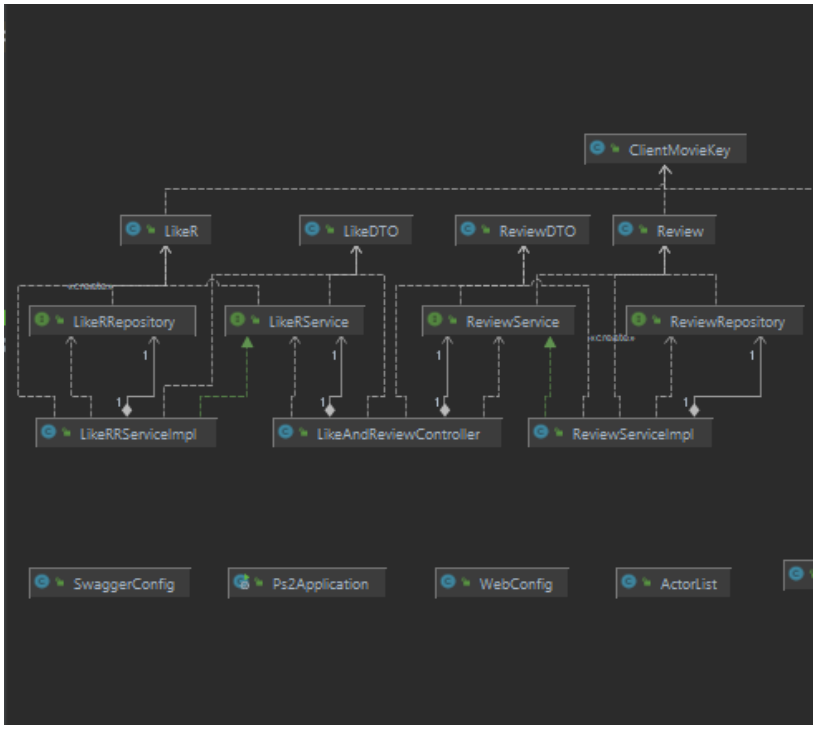


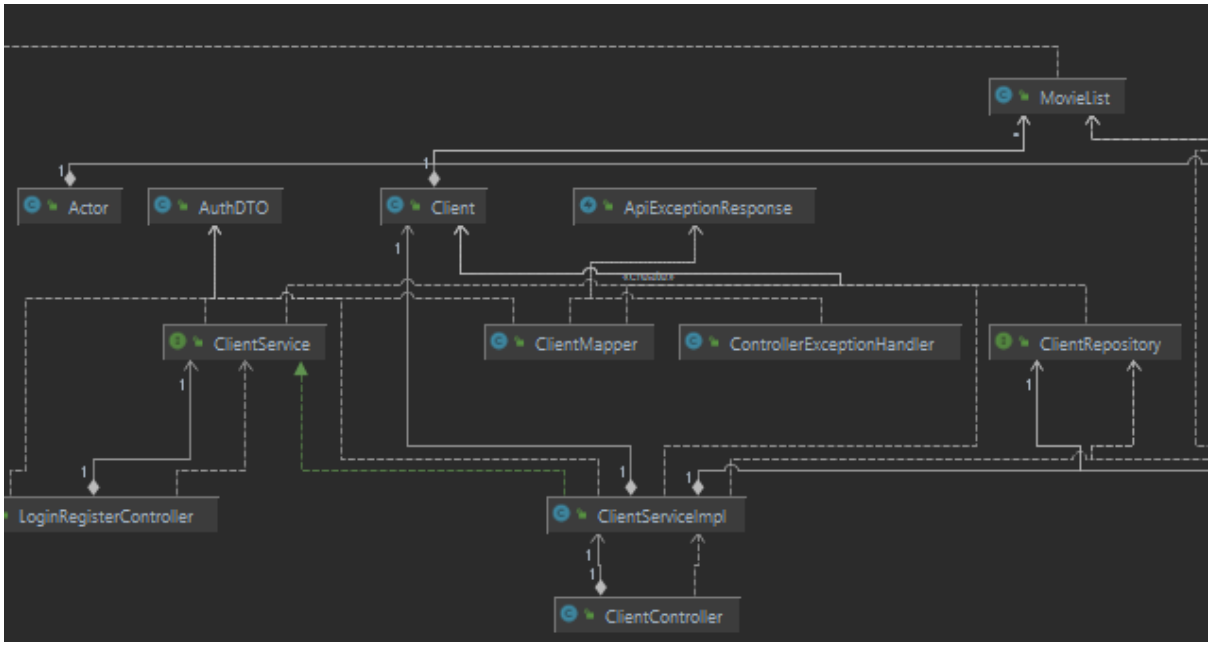
- Recomandare film

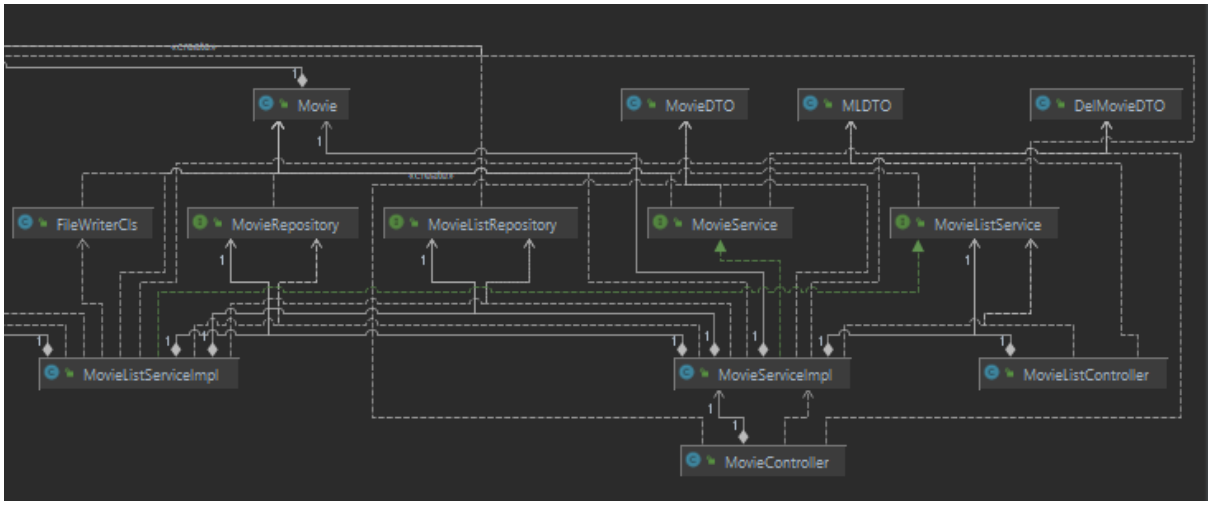


Class Diagram

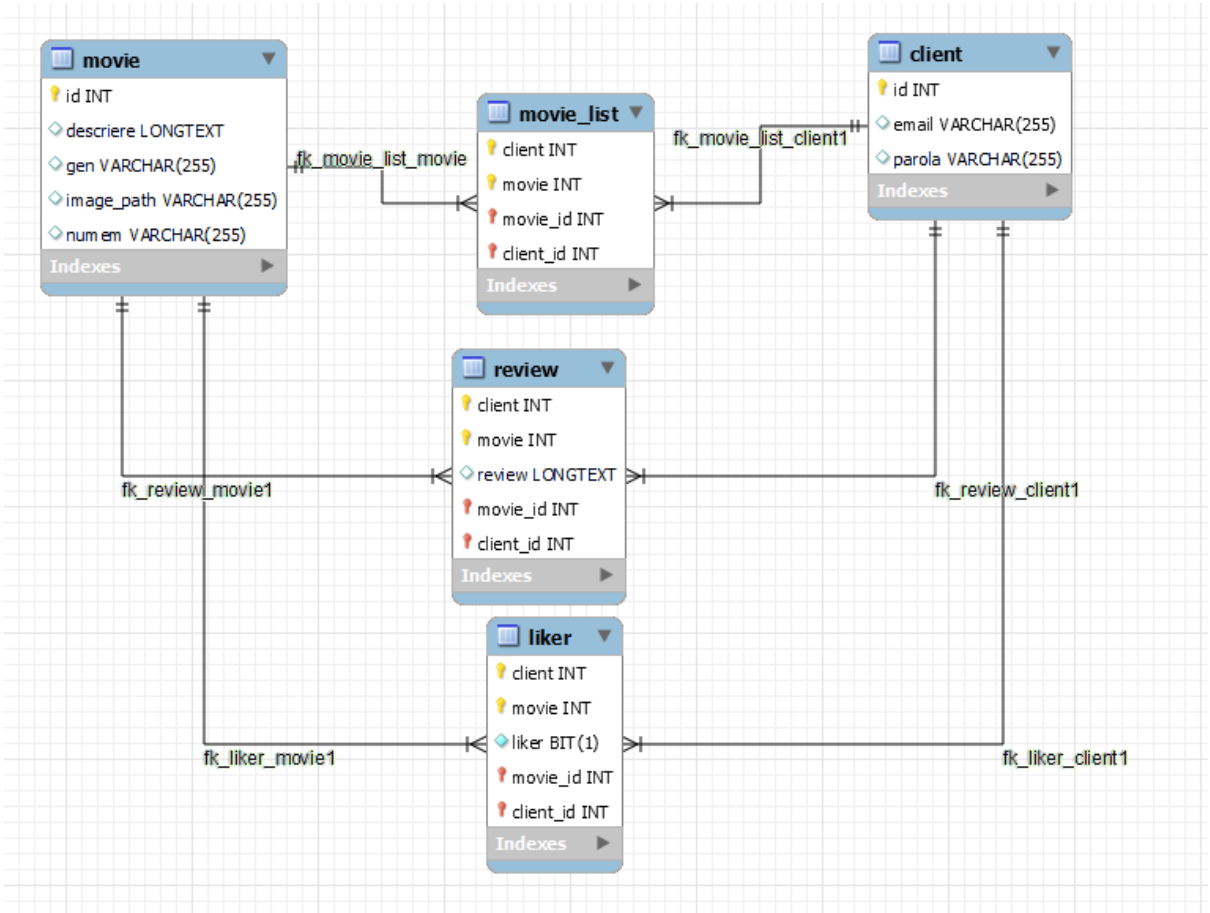







Movie
MovieDTO
MLDTO
DelMovieDTO
FileWriterCls
MovieRepository
MovieListRepository
MovieService
MovieListService
MovieListServiceImpl
MovieServiceImpl
MovieListController
MovieController

Data Model


movie
id INT
descriere LONGTEXT
gen VARCHAR(255)
image_path VARCHAR(255)
num em VARCHAR(255)
Indexes
fk_movie_list_movie
movie_list
client INT
movie INT
movie_id INT
client_id INT
Indexes
fk_movie_list_client1
client
id INT
email VARCHAR(255)
parola VARCHAR(255)
Indexes
review
client INT
movie INT
review LONGTEXT
movie_id INT
client_id INT
Indexes
fk_review_movie1
fk_review_client1
liker
client INT
movie INT
liker BIT(1)
movie_id INT
client_id INT
Indexes
fk_liker_movie1
fk_liker_client1

System Testing

Testarea unitară s-a impus în ultima perioadă în dezvoltarea proiectelor scrise în limbajul Java şi numai, pe măsura apariţiei unor utilitare gratuite de testare a claselor, care au contribuit la creşterea vitezei de programare şi la micşorarea drastică a numărului de bug-uri. Cel mai folosit utilitar pentru testarea unitară a claselor Java este JUnit

Am efectuat cu success 3 teste asupra entitatii Movie :

```
@Test
public void findByName()
```

```
@Test
public void addMovie()
```

```
@Test
public void deleteMovie()
```

Pentru realizarea testelor am utilizat Framework-ul Mockito. Mocking ul este un proces de creeare a unei clone pentru obiectere reale. Cu alte cuvinte,
mocking este o tehnică de testare în care obiectele clonate sunt folosite în loc de obiecte reale în scopuri de testare. Obiectele clonate oferă o ieșire falsa pentru o anumită intrare falsa transmisă acestuia.

Future Improvements

[Present some features that apply to the application scope.]

Conclusion

Bibliography