An online database for information on movies, tv series and celebrities.

Analysis and Design Document

Student:Giurgiu Diana-Ioana

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <24/04/2019> | 1.0 | Iteration 1.1 | Giurgiu Diana |
| 08/05/2019 | 2.0 | Iteration 1.2 | Giurgiu Diana |
| 22/05/2019 | 3.0 | Iteraton 1.1 revised | Giurgiu Diana |
| 30/05/2019 | 4.0 | Iteration 1.2 revised | Giurgiu Diana |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

This application is intended to be a database for Movies, Tv Shows, Celebrities(Actors) where a User can post comments in the movies/tv Shows they have watched or give a rating to movies/Tv Show. The comments are checked afterwards by the moderators and if they identify any illegal words(ex: swearing) the comment will be removed and depending on the rules they have crossed their penalties can vary from being mute for a period of time to their account being removed.

# Elaboration – Iteration 1.1

# Domain Model

In software engineering, a domain model is a conceptual model of the domain that incorporates both behaviour and data.

In ontology engineering, a domain model is a formal representation of a knowledge domain with concepts, roles, datatypes, individuals, and rules, typically grounded in a description logic.

The domain model of this application includes the following :

* Admin
* User
* Moderators
* Movie
* Tv Shows
* Actors
* Anon

The admin has action over user and moderators and also over movies, tv shows and actors (those are not mentioned in the image because the relationships would have become too chaotic and hard to understand).

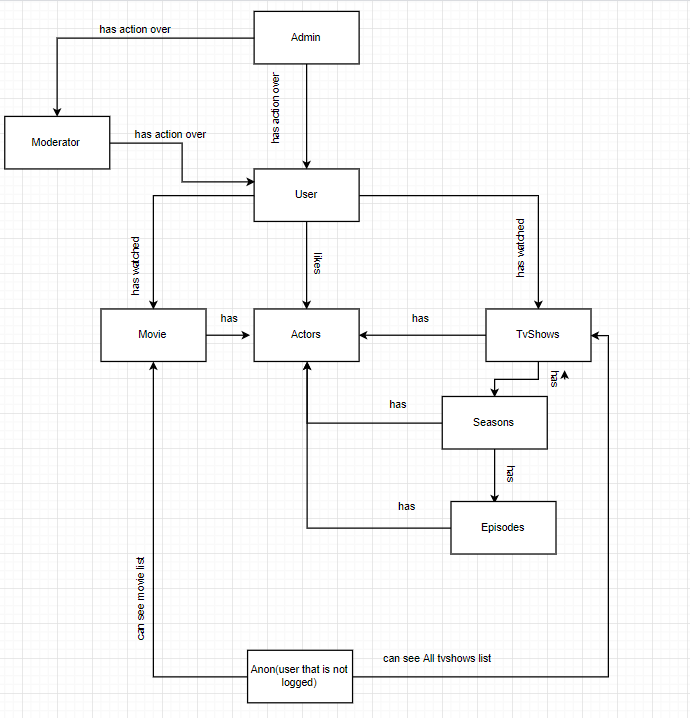
The moderator has action over User and also over the movies and tv shows and actors categories, they have the role to check if the facts written are correct and to correct eventual errors.

The users are the ones that can use the application and can get recommendation on movies or tv shows based on the movies and tv shows watched before. A user can get a recommendation for movies or for tvShows. He can select the movies/tvshows that he has watched.

The Anon can view the list of movies and tv shows and can log in to create an account in order to gain access to the recommendation feature.

Concerning the relationships:

* Mapped One to many
  + Seasons-Episodes
* Mapped Many to many
  + Tvshows-actors
  + Movies-actors
  + Movies- User
  + TvShows- User
  + TvShows- Seasons

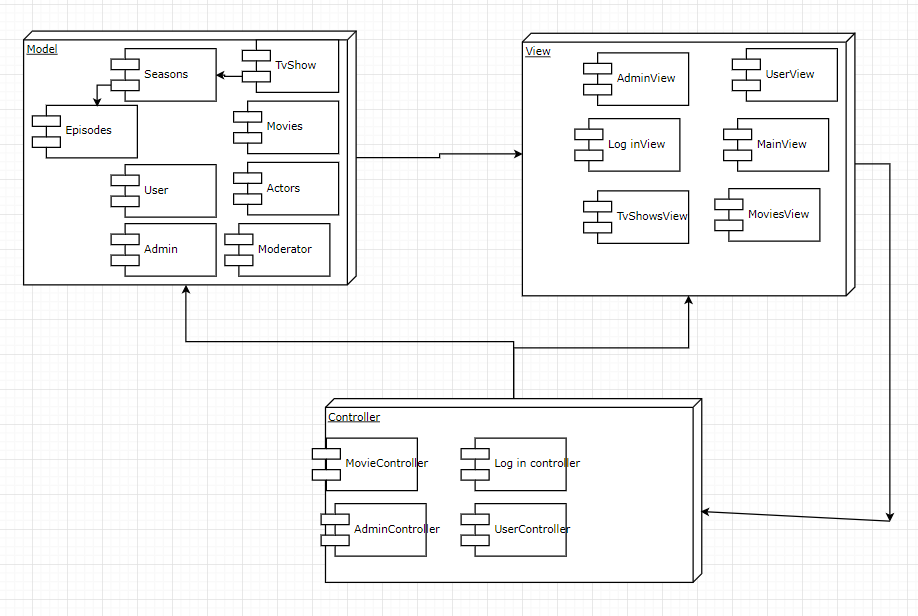


# Architectural Design

## Conceptual Architecture

Chosen architecture is MCV implemented in java and spring.

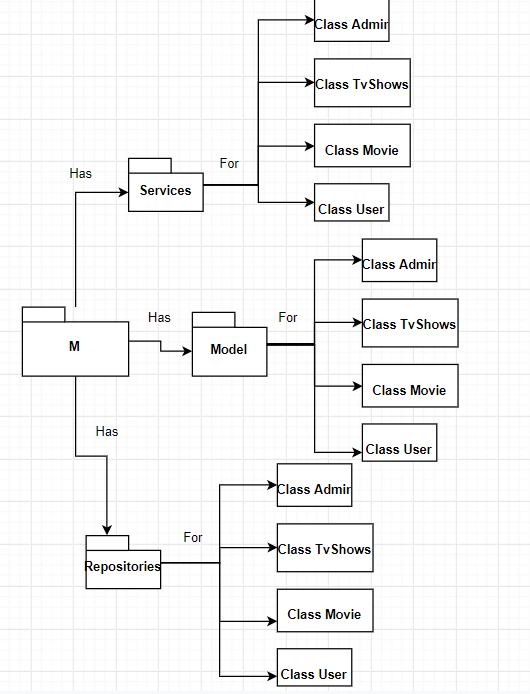
MVC stand for Model-View-Controller and each package has the corresponding classes or packages. Below are the diagrams.



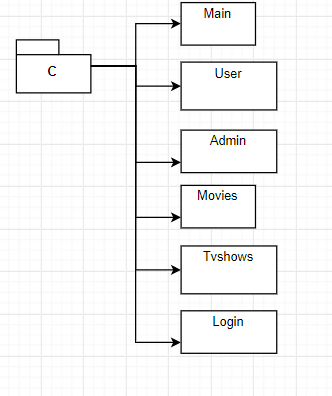
## Package Design

The packages incorporated in my application are mapped by the layers they are part of:

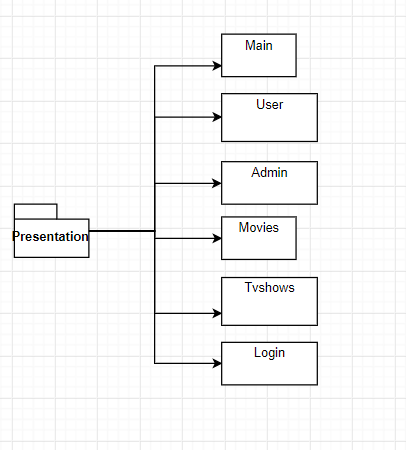
* The M (model)package it contains the database connection and all the classes for the model, the JPA repositories and the services that contain actions like CRUD, findAllMoviesByName etc.



* The C(Controller) package.
  + It contains all the methods necessary in order to make the connection between the database and the User Interface.

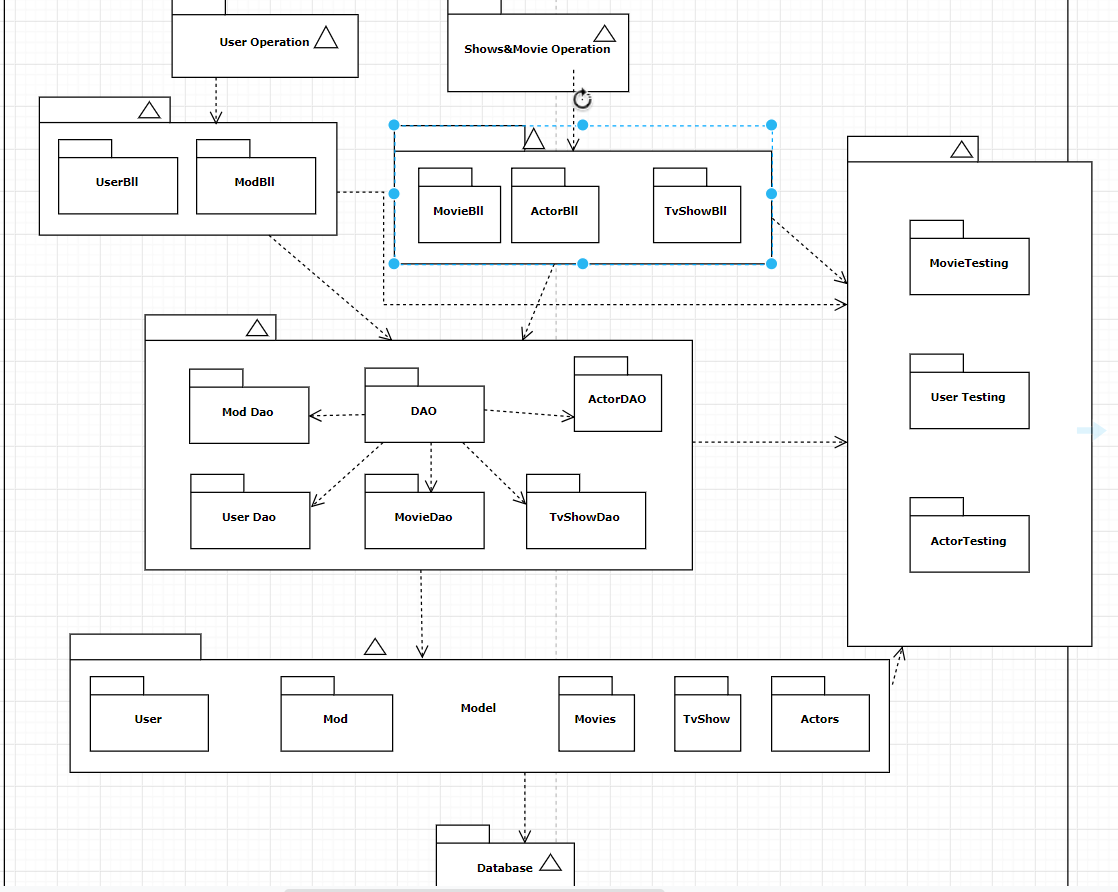


* The last package, the presentation package has all the interfaces for the user access. Interfaces are under the form of HTML pages.



## Component and Deployment Diagrams

The component diagram:

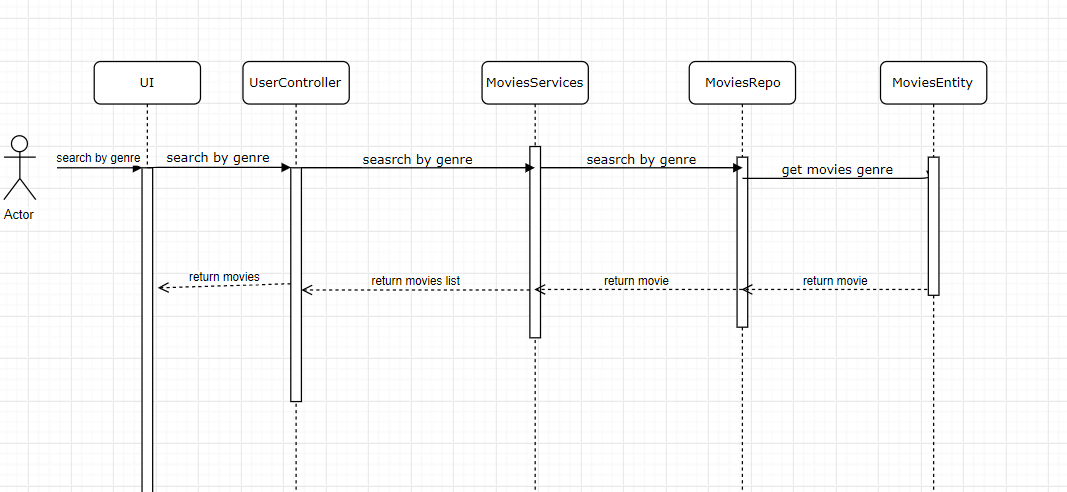


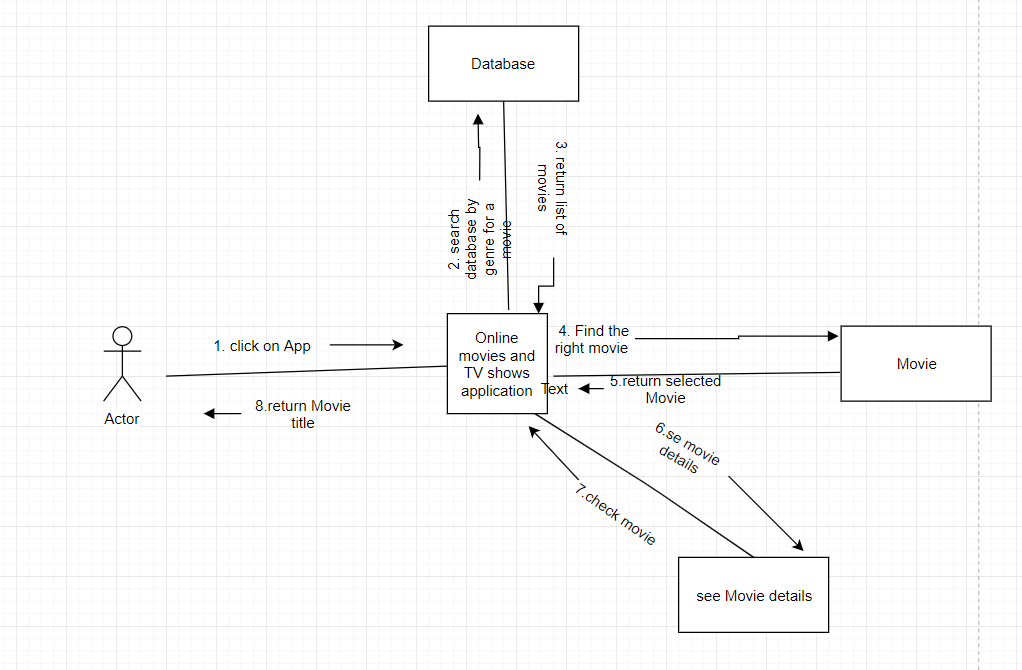
# Elaboration – Iteration 1.2

# Design Model

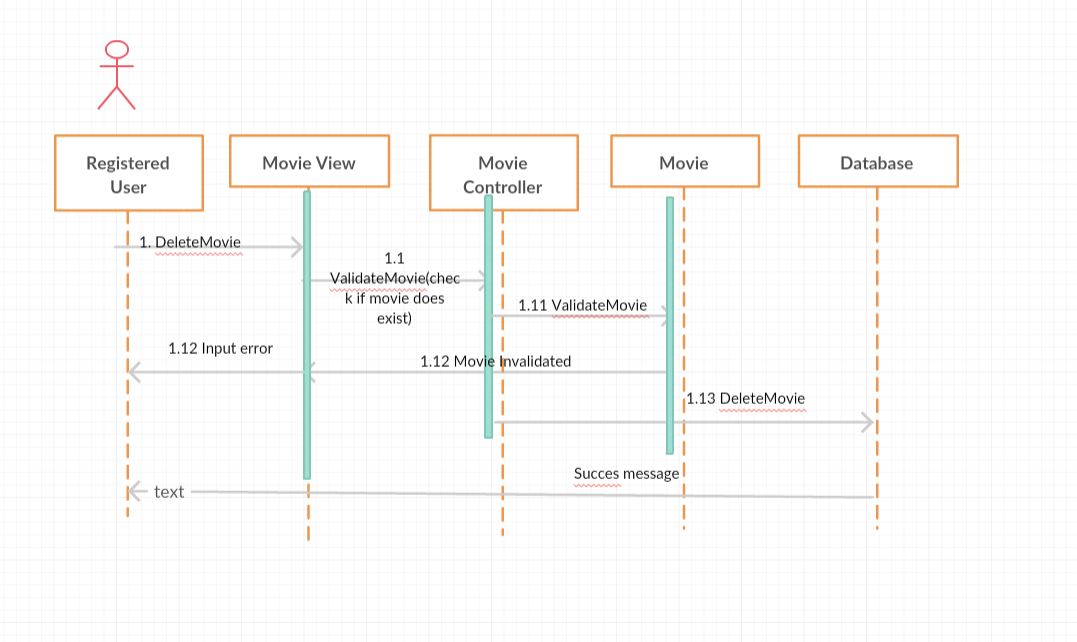
## Dynamic Behavior

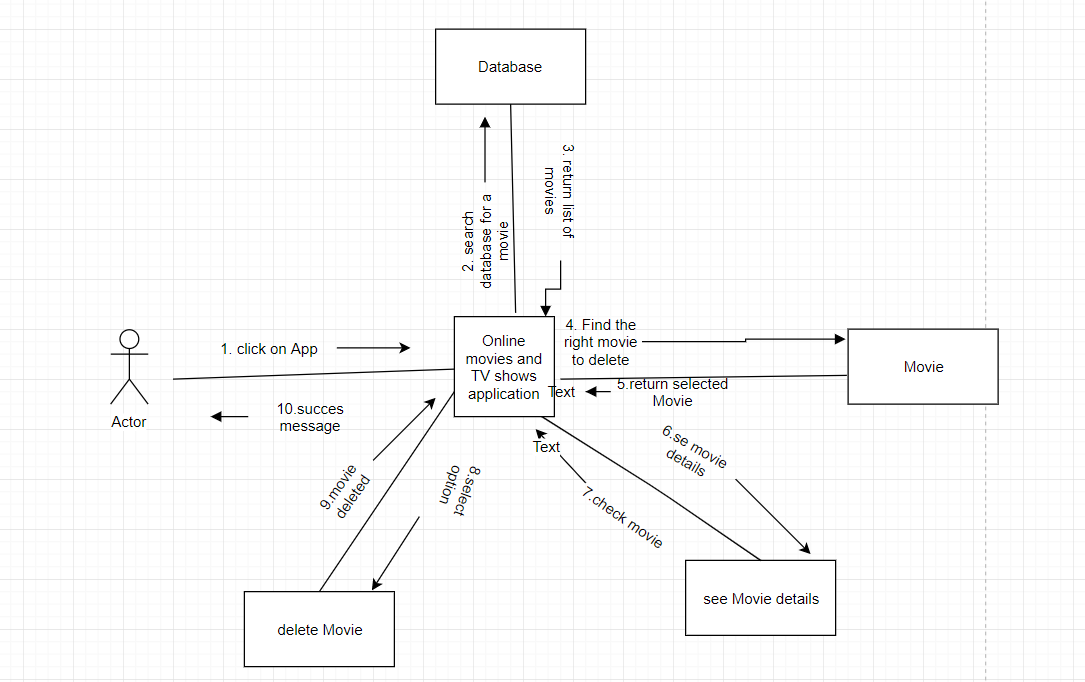
Searching for a movie by genre



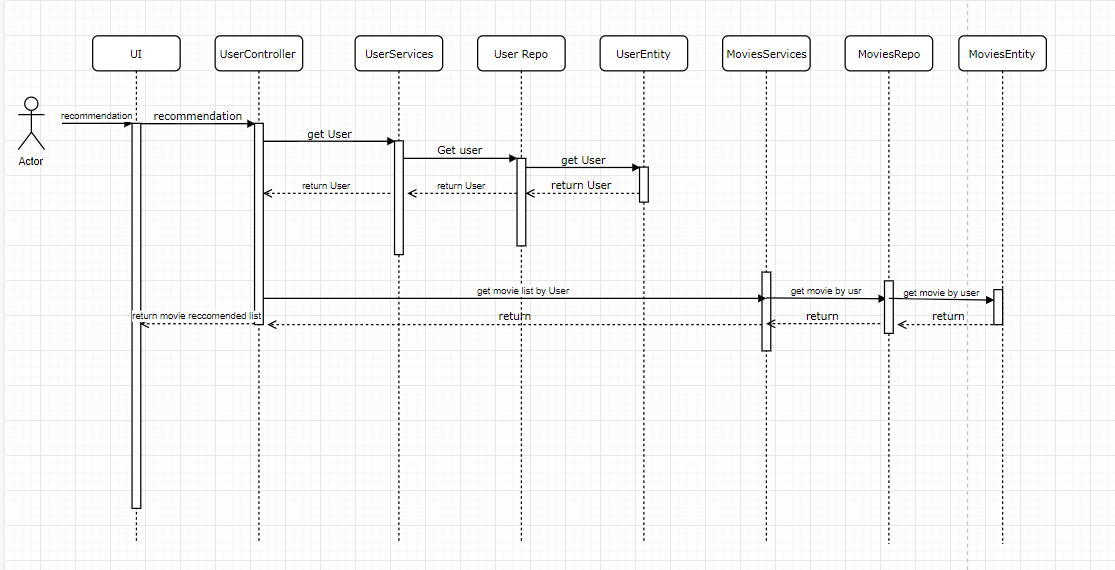


Deleting a Movie by a registered user





Getting recommended movies:

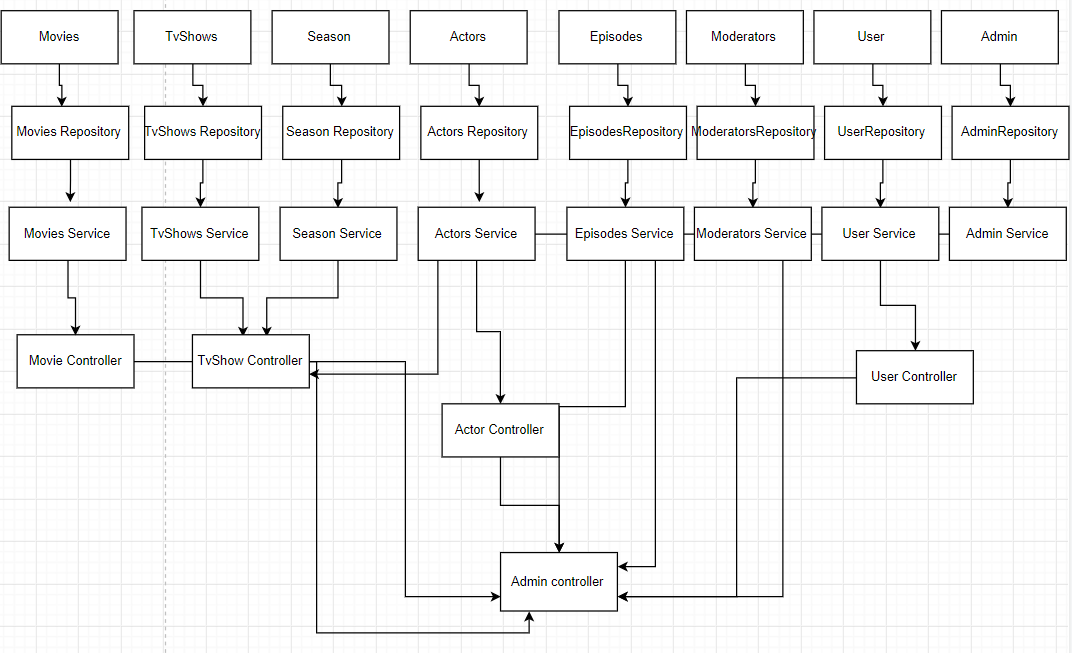


## Class Design

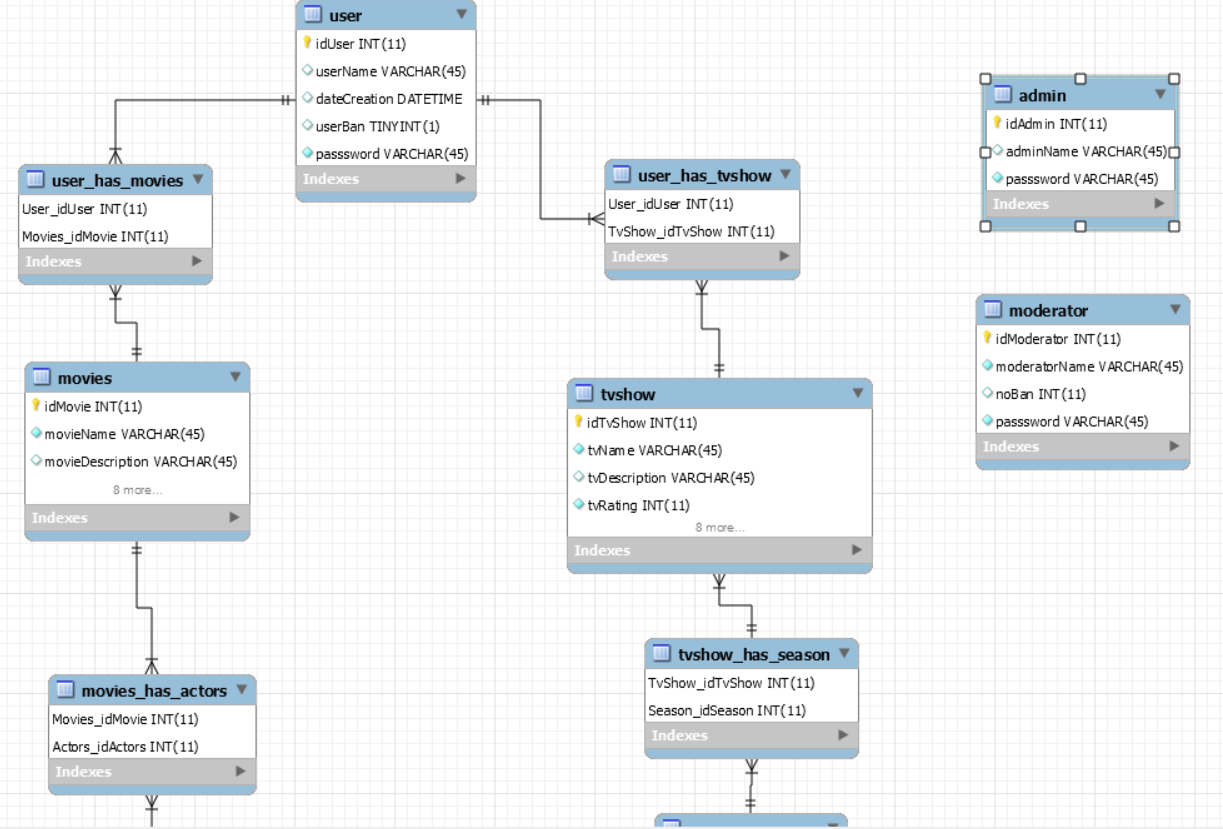
The classes I have designed follow the MVC pattern in which the DAO classes are in a separated package from the BLL and the MODEL.

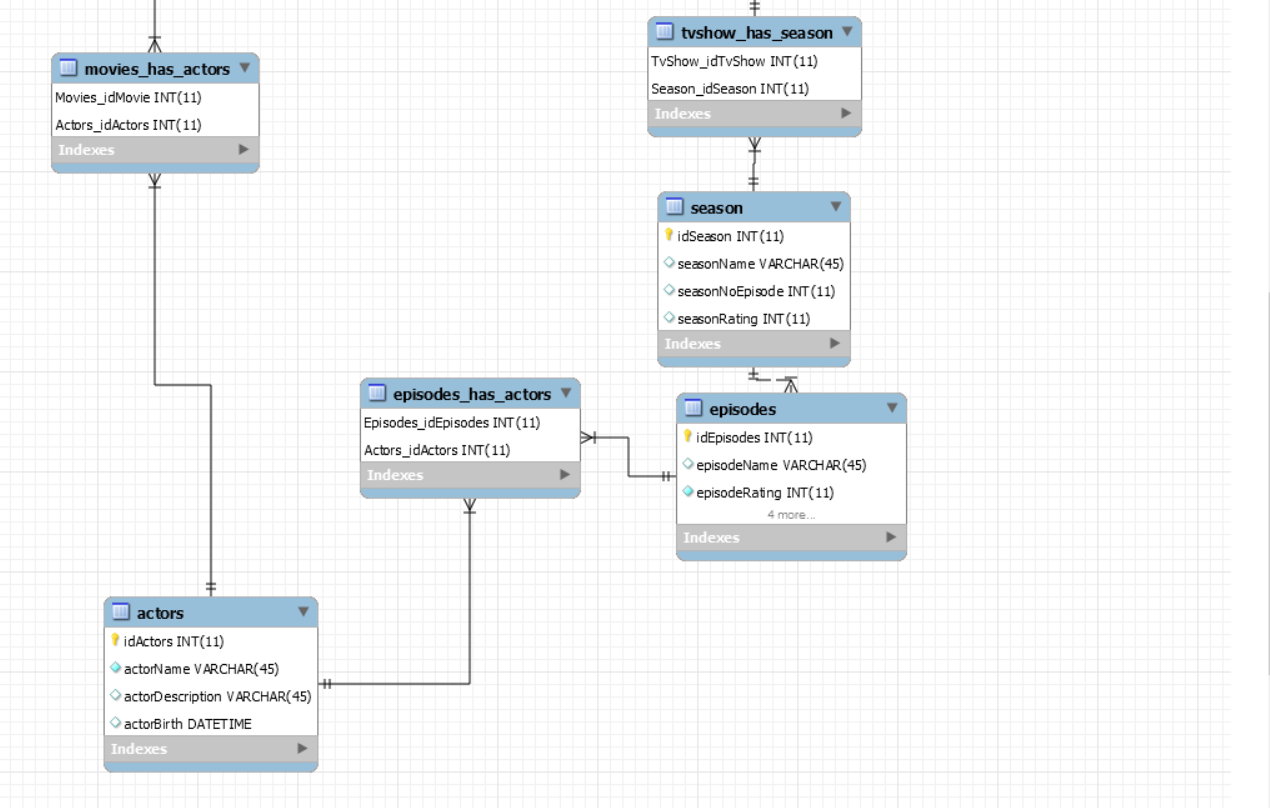
The M package contains packages with the definitions for the tables and the JPA and services necessary.

* Movies:
* TvShows
* Actors
* Admin
* User
* Moderator
* Season
* Episodes



# Data Model





# Unit Testing

The testing was done with JUNIT testing.

The testing cases that were run were:

* CRUD on user
* CRUD on movies
* CRUD on tvShows
* CRUD on seasons
* CRUD on episodes
* CRUD on moderators
* CRUD on admin
* CRUD on actors
* Relationship one to many season-episodes
* Relationship many to many tvShows-seasons
* Relationship many to many movies-users

# Elaboration – Iteration 2

# Architectural Design Refinement

I have modified the domain model by adding a diagram that represents my project and the relationship between the classes.

Furthermore, I have completed the package design classes and added the correct diagrams for the package design classes.

I have explained the features my application has and The purpose of the application.

I have added a new diagram for the

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

There was no need to modify the UML class diagram.

# Construction and Transition

# System Testing

I have applied the testing by creating a testing method for each of the above mentioned testing cases and I have teste separately each one of them in hopes of correcting if there are any issues. The testing has helped me to identify early on issues in the code so that they will not appear further down the implementation.

# Future improvements

Future improvements would consist in a email delivered service information regarding the apparition of new movies that fit the genre of the user. Another possibility would be to make a tied in cinema purchase application through which the user can buy or reserve tickets online.

# Bibliography

<https://www.draw.io/>

<https://start.spring.io/>

<https://www.callicoder.com/hibernate-spring-boot-jpa-one-to-many-mapping-example/>

<https://www.callicoder.com/categories/spring-boot/page/3/>

<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

<https://www.imdb.com/>