Contribution

Front End: Zygimantas

Server: Zygimantas

MySql: Zygimantas

Rest: Zygimantas, Daniel, Emil

Façade: Daniel, Zygimantas

Entities: Daniel, Plamen

Tests: Plamen

Movie List Application

The idea is to have a database of movie information, and an entire community built around it.

Each user would have their own personal list of the movies they have watched and the movies they plan to watch in the future, this would be coupled with a way for each person to personal rate the movie they just watched. This information should be available for other users so you are able to look at other users and compare your own tastes in movies to others.

On the movie page itself there should be reviews by the users and we could also implement that we go out and gather professional critics reviews of the movie as well. We would also on the movie page have recommendations by other users to movies that are similar to the selected movie put together with a box of text with how they are similar and why you should watch this.

We would also have a friend function so you could easily re find users with a similar taste to you. We also want to implement a suggest a movie function that takes the movies you have watched and find a similar one to the ones you have liked, for this function you should be able to give it a genre as well to narrow down the result list.

We would also like to implement a system for finding people with similar taste so you could find another user, who have watched and liked the same movies as you have.

In the end, we might also want to implement a chat system and a forum for people to discuss the movies or film news and similar things but that is on a lower priority to the previous features.

Our current idea consists of the following user stories:

Admins should be able to add a movie to the database

Admins should be able to Edit/Update a movie.

Users should be able to add a movie to their movielist

Users should be able to add a review to the moviepage

Users should be able to add a recommendation to the moviepage

Users should be able to add other users to their friendslist

Admins should be able to update a moviepage

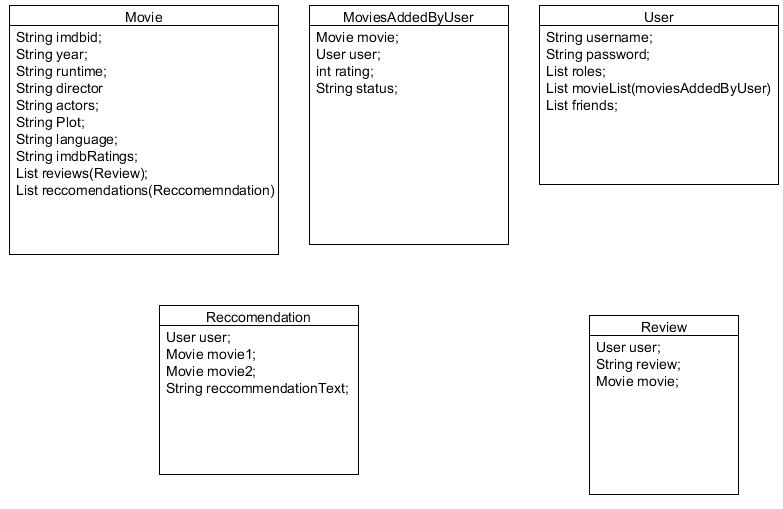
Users should be able to receive a list of movies that are suitable to their tastes

Users should be able to get a list of movies they might like for a specific genre

Users should be able to find other users with similar tastes in movies.

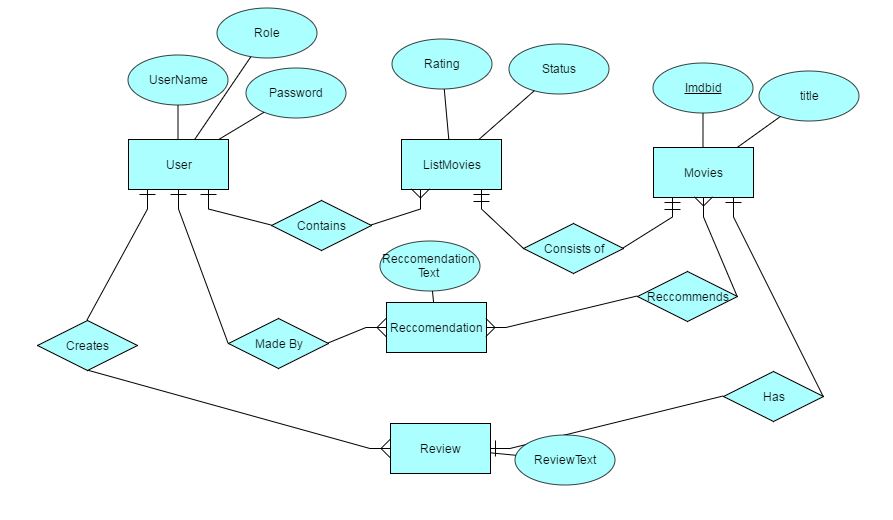
***Daniel.***

Our current UML Diagram of Entity Classes:



***Daniel.***

ER-Diagram



***Daniel.***

Description of MovieFacade

We have made a new Facade called "MovieFacade" where we have various methods for dealing with our information. We have two different methods for getting movies, the first method gets a movie by its name and the second gets it by the ID.   
We have done it this way as some movies may have the same name, and therefore searching by name of the movie that the user wishes to get is not the correct one, they can therefore get it by ID, which is unique for each movie and will ensure the user getting the info for the specific movie.

We also have 2 methods for creating movies, which also works with name and ID, these methods add movies into our Database. As this is only done on the Admin side, then we could just stay with the method of adding movies by ID only, since that will ensure the correct movie added every time, but we still implemented it to make sure movies could also be added with the title of the movie. It could probably just be a "test" method, but we still felt it was a good thing to have as an Admin.

We have an update method where it's possible for the Admin to change information about a movie. It could be that a movie is added before it's released and therefore have missing information, which would then need to be updated. This will be for the admin only.

We talked together and figured that we wouldn't need a delete method, as all the movies we get are existing movies and will therefore not need to be deleted again.

***Emil.***

Rest package – Movies class

In our Movies class in the rest package we have methods for getting and creating movies, we have also assigned roles. The roles are assigned for those methods which is meant for admins only, so the create methods. This will ensure that the user is not able to create movies in the database, but only the admins. The first two methods are "@GET" as we're getting information from our database to view for the user,  
the two next methods are "@POST" since we're putting data into our database from another place "OMDB" and the last method is "@PUT" as we're changing already existing data in our database.

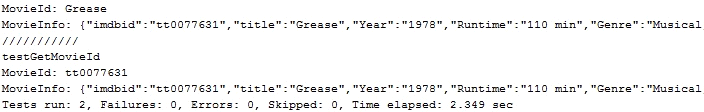
***Emil.***

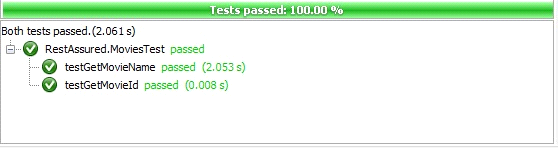
We tested the functionality of our facade and the connection to the persistence. We did that briefly by instantiating a test database and using 2 tests, to try out our methods for first getting a movie query and then pushing a movie object to the database.

***Plamen.***

Rest assured tests:

We test to see if our rest works correctly by checking if we can find a movie that already exists in the database by giving it the test the name of the movie. We do this by going into rest and call the methods for getMovieName and getMovieId and then giving it a name or an ID for an existing movie. We then check if we get all the data back from the movie.



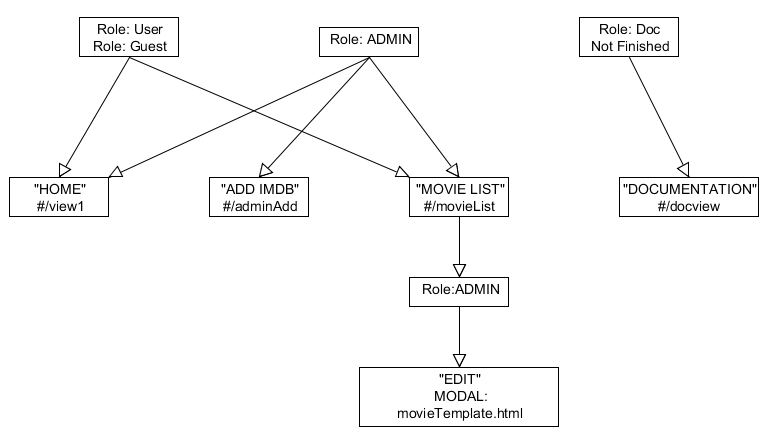


We do a System out Println to make see if we get all the info about the movie back from the JsonObject, but we only check for the Title or the ID of the movie in our expected result.

***Emil.***

Navigation Diagram

Because we are using AngularJS NG-VIEW and avoiding making redirect pages the Navigation Diagram is very simple.



*Zygimantas.*