Movielister

Grupp: Beta

Medlemmar:

Vickie Dam
- **900122-4220**
- **vickie@student.chalmers.se**

Alexander Ekstrand
- **920102-4552**
- **aleeks@student.chalmers.se**

Magnus Rising
- **920311-2934**
- **rising@student.chalmers.se**

Vidar Åsberg
- **930929-1590**
- **vidara@student.chalmers.se**

Generell beskrivning:

Vår webbapplikation är tänkt som ett bibliotek av filmer. Där du som användare ska kunna söka efter filmer efter namn. Klickar man in på en film presenteras man med ytterligare information om den. Informationen är tagen från sidan Rotten Tomatoes. Är man inloggad via Twitter kan man även skapa sina egna listor med t.ex bilfilmer eller favoritfilmer.
Sidan är tänkt för personer som har svårt för att komma ihåg sedda filmer.

Use cases:

- Söka efter filmer
- Se informationen om en film
- Se publika listor
- Inloggning
- Skapa egen lista med filmer
- Lägga till film i en lista
- Ta bort en film från en lista

Användarroller

Man är antingen inloggad som en användare eller inte. När man är inloggad kan man skapa nya listor, lägga till filmer i listan, ta bort filmer från listan och även ta bort hela listan. Användaren kan sätta listorna som privata eller publika. Med privata listor innerbär det att den listan bara ses av användaren som har skapat den, medans publika listor kan ses av alla. Vid borttagning av användaren kommer listorna skapade av just den användaren att tas bort och försvinna från databasen.

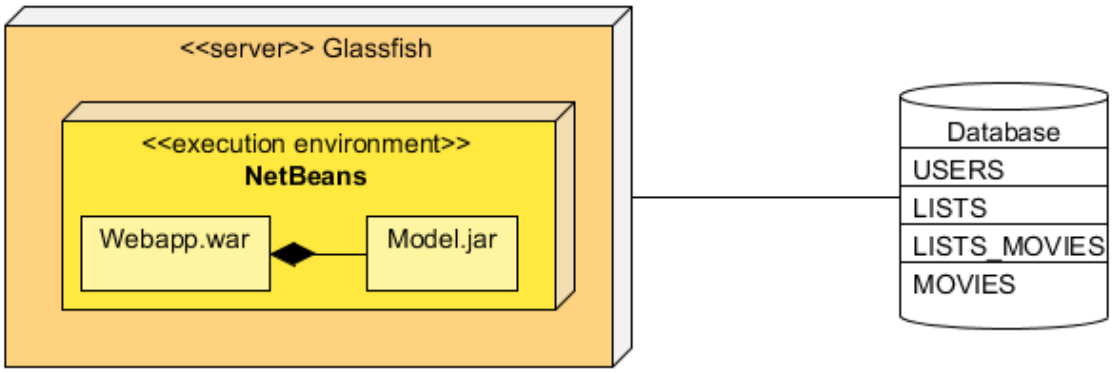
Teknisk design av systemet:

- UML (se sista sidan eller bifogat i git repo)

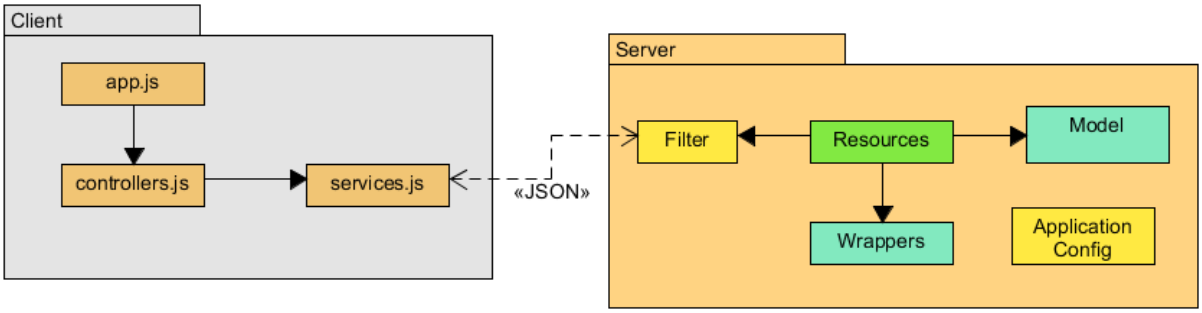
Approach: Vi använder oss av RESTful tjänsten.

I vår persistenslager, finns DAO- och entitetklasserna som är skrivet i java. I weblagret har vi den grafikska delen för klienten där html, css och javascript används. Vi använder oss även utav AngularJS för lättare hantering av värden.

En enkel representation av vårt system som är kopplad till databasen.



Kommunikation mellan servern och klienten



För inloggning använder vi oss av oauth och ett filter som skyddar anropet för resurserna. Kommunikationen mellan servern och klienten sker via resurser på klientsidan. Resurserna skickar över information som ett Json-objekt som fångas upp av services som är skrivet i javascript. Det grafiska som är skrivet i html kan då använda sig av de ”scope” värdena som vi har skapat med hjälp av AngularJS och visa upp det på sidan. Värdena kan vara listor av filmer eller bara filmer.

Modules:

Vi använder oss utav en sida (Rotten Tomatoes) där vi från deras databas kan söka upp filmer vi vill lägga till i vår filmlista. Authorisering sker med hjälp av oauth där vi loggar in med Twitter.

Att lägga in filmer är bara möjligt om man är inloggad, annars visas bara publika listor som en användare har skapat och gjort den publikt.
Vi har följande paket:

- **Model**
  - **core** - innehåller entiteter, så som Movie, MovieList, User.
    - Movie - En film som innehåller titel och utsläppsdatum.
    - MovieList - En lista med filmer som kan skapas av användaren.
    - User - Användare som skapas via Twitter (oauth).
  - **persistence** - innehåller abstrakta klasser för persistenslagret (se uml).
  - **dao** - innehåller “data access object” klasser för databasen och persist mekanism.
- **Webapp**
  - I **WEB-INF** finns en urlrewrite som städar upp urlerna utan att klienten märker det. Detta gör så att hela sidan kan indexeras av sökmotorer då de inte kör javascript.
  - **auth** – Sköter login med OAuth. Vi loggar in genom Twitter och dirigerar till en callback servlet där vi har fått en nyckel av Twitter. I sessionen kan vi sätta användarnamnet som kan användas av klienten för att autentisera resursen.
  - **resource** – innehåller resurser och wrap-klasser för att kommunicera med servern.
  - AngularJS – Används av klienten

RESTful api

POST

----------UserRegistryResource----------
createNewList - api/users/{nickname}/lists
-- Skapa en ny lista till användaren

addMovie - api/users/{nickname}/lists/{id}
-- Lägg till en ny film i listan

PUT

----------UserRegistryResource----------
updateList - api/users/{nickname}/lists/{id}
-- Lägg till en ny film i listan

DELETE

----------UserRegistryResource----------
deleteList - api/users/{nickname}/lists/{listId}
-- Ta bort listan från användaren

removeMovie - api/users/{nickname}/lists/{id}/{movie}
-- Ta bort listan från användaren

GET

----------InitDataResource----------
getJson -api/init
-- Fyller databasen med data för att testköra

----------UserRegistryResource----------
findAllLists - api/users/{nickname}/lists
-- Hitta alla listor som användaren har

findList -api/users/{nickname}/lists/{id}
-- Hitta den specifika listan med id

----------SessionResource----------
getSession -api/session
-- Hämta nuvarande användare för denna session

----------ListCatalogueResource----------
findAllPublic -api/lists
-- Hämta alla listor som är publika

findPublic -api/lists/{id}
-- Hämta den specifika listan, hittar bara om den är publikt

Umlet kommer även att finnas som bifogat fil i git repot