FINAL PROJECT

Movie Website

Advanced Web and Javascript And API Management project

yang mengwei
yangmegnwei1997@126.com

1. Basic information

The front end: Angular2

The backend database: PostgreSQL and mango dB databases

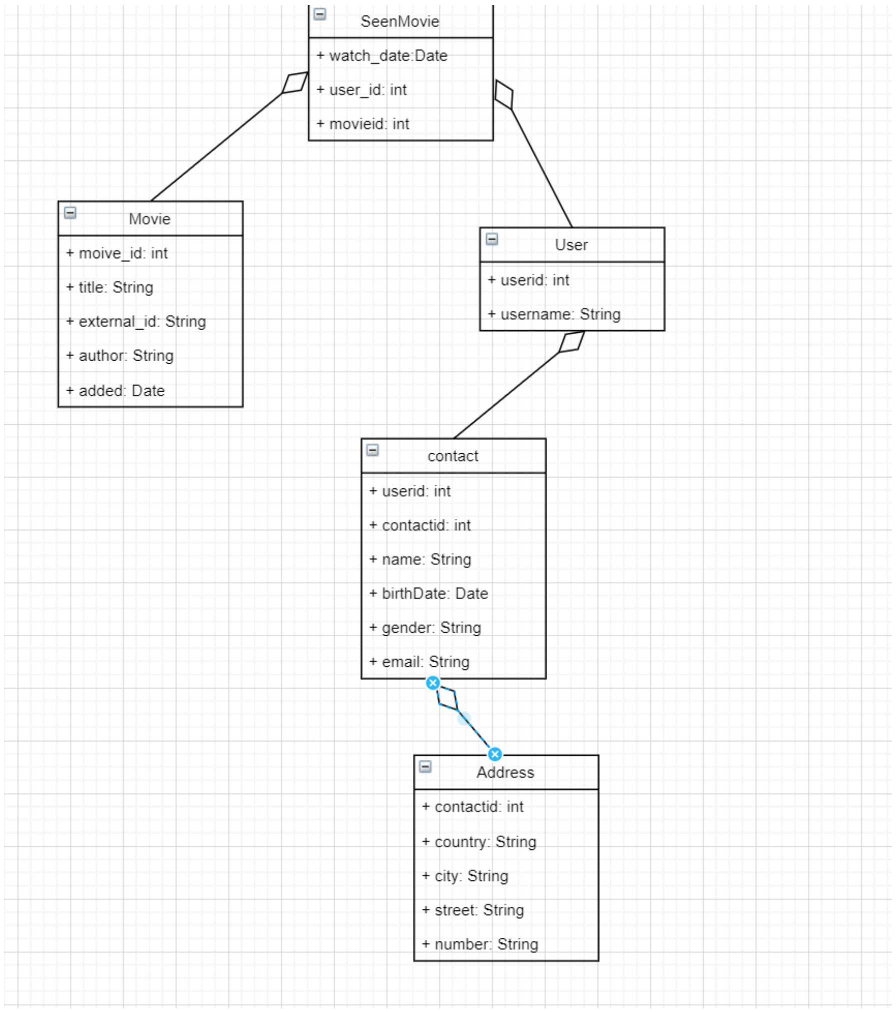
Back-end Language: java and express

2. Data structure:

2.1. PostgreSQL:

The data stored in PostgreSQL: movie information, and user information, the main tables are movie, user, seenmovie, address, contact,

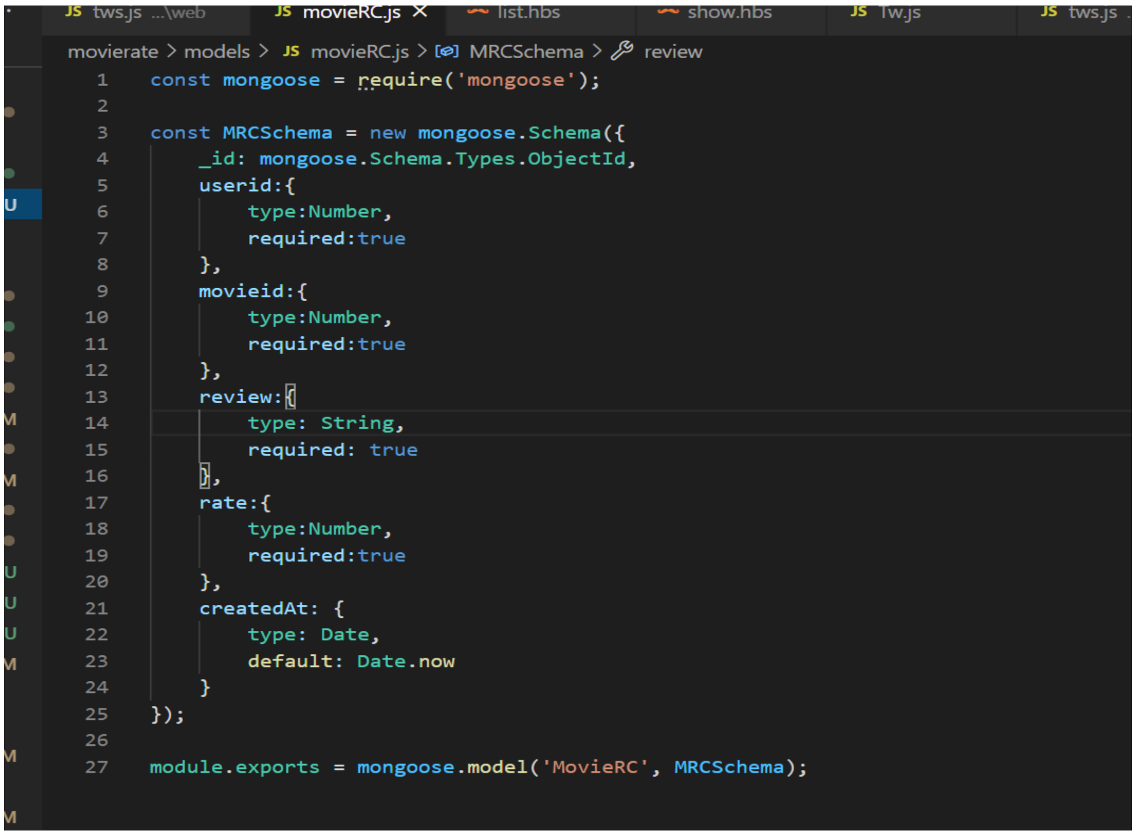
The data stored in mango dB is the user's comments and scoring of the movie. The table structure is userid, moveid



The PostgreSQL database is implemented in java, which is a jar project used to create and implement operation statements in the database grid and create API

2.2. Mango dB

Mango dB uses node.js for the back end and express framework, and the front end uses Angular.js



```
const mongoose = require('mongoose');

const MRCSchema = new mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    userid:{
        type:Number,
        required:true
    },
    movieid:{
        type:Number,
        required:true
    },
    review:{
        type: String,
        required: true
    },
    rate:{
        type:Number,
        required:true
    },
    createdAt: {
        type: Date,
        default: Date.now
    }
});

module.exports = mongoose.model('MovieRC', MRCSchema);
```

3. Page Design

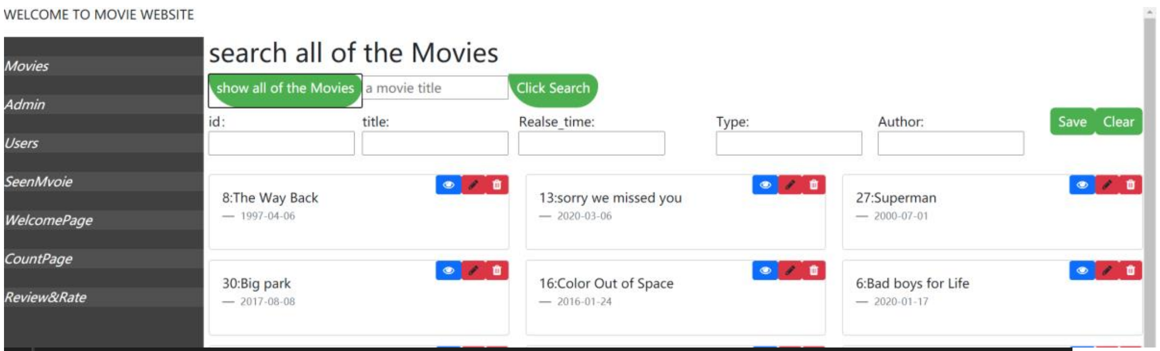
My page is divided into 7 parts, which respectively fulfill the remaining 4 requirements required by the teacher. (Regarding the login function, because I don’t know how to implement user login in java in Angular, so only the login page is implemented, but the login function is not really implemented.)

3.1. Movie Page

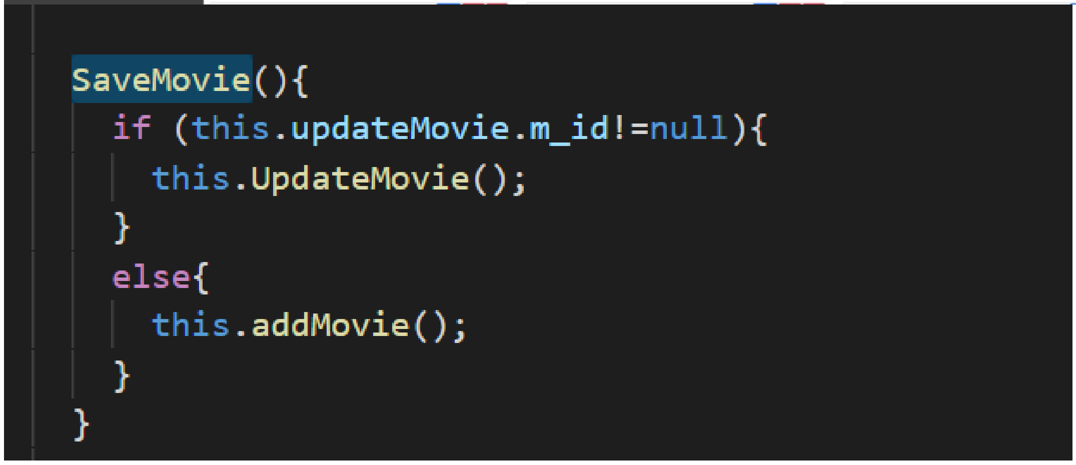
In this Page, I finish some operations about movie: add, search all of movies, search a single movie, delete movies and update a movie.
The SQL sentences in the movieDAO .

1) add and update movie

you can input these values and click add movie, if you do not add id ,click save means add a new movie. if you click pen icon, the data will upload on these inputs, you can update this movie. And also click save button. if there is a number in id, click save button means update the movie data.



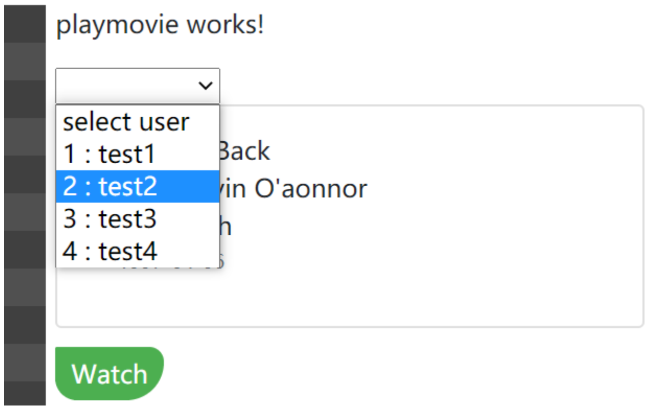
8:The Way Back — 1997-04-06
13:sorry we missed you — 2020-03-06
27:Superman — 2000-07-01
30:Big park — 2017-08-08
16:Color Out of Space — 2016-01-24
6:Bad boys for Life — 2020-01-17



```
SaveMovie(){
  if (this.updateMovie.m_id!=null){
    this.UpdateMovie();
  }
  else{
    this.addMovie();
  }
}
```

2) Watch movie (Play film mock).

When you are ready to watch a movie, please click the “eye” icon, then it will jump another page.

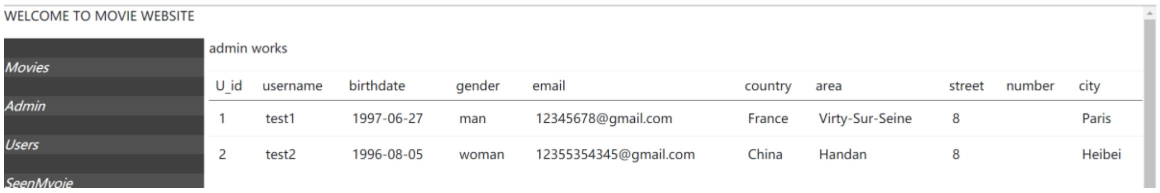


choose a user name and click watch button, then make sure that you are already to watch the movie (This corresponds to the third requirement of the document requirement, play film (mock))

3) delete movie

if you click rubbish icon, then you will delete the movie.

3.2. Admin Page

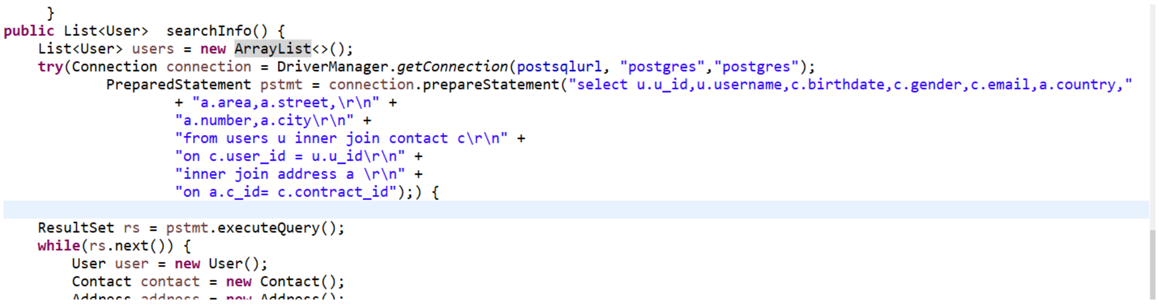


WELCOME TO MOVIE WEBSITE

admin works

U_id	username	birthdate	gender	email	country	area	street	number	city
1	test1	1997-06-27	man	12345678@gmail.com	France	Virty-Sur-Seine	8		Paris
2	test2	1996-08-05	woman	12355354345@gmail.com	China	Handan	8		Heibei

these are user information. there are three tables by using SQL sentences.



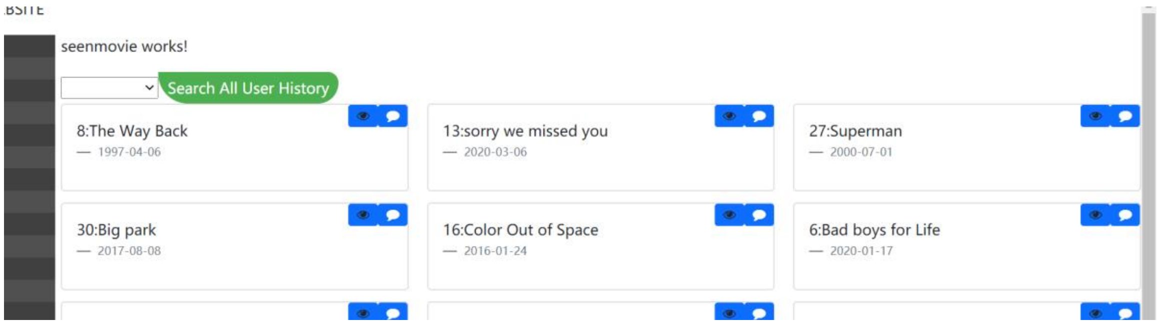
```
    }
public List<User>  searchInfo() {
    List<User> users = new ArrayList<>();
    try(Connection connection = DriverManager.getConnection(postsqlurl, "postgres","postgres");
            PreparedStatement pstmt = connection.prepareStatement("select u.u_id,u.username,c.birthdate,c.gender,c.email,a.country,"
                    + "a.area,a.street,\r\n" +
                    "a.number,a.city\r\n" +
                    "from users u inner join contact c\r\n" +
                    "on c.user_id = u.u_id\r\n" +
                    "inner join address a \r\n" +
                    "on a.c_id= c.contract_id");) {

    ResultSet rs = pstmt.executeQuery();
    while(rs.next()) {
        User user = new User();
        Contact contact = new Contact();
```

3.3. User page (search the last seen movie)

In user page, you can search the user last seen movies by their id

3.4. SeenMovie Page (Rate film)

In this page, you could search all of users' history, and choosing a user and watch a movie and make a rate to a movie.



seenmovie works!

8:The Way Back — 1997-04-06

13:sorry we missed you — 2020-03-06

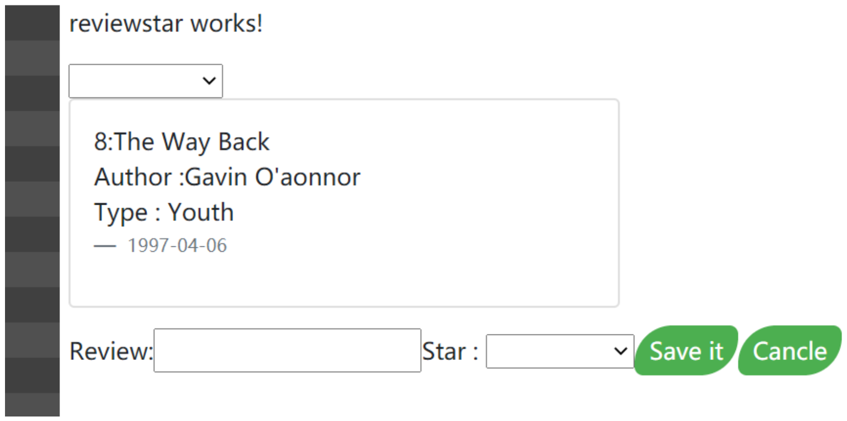
27:Superman — 2000-07-01

30:Big park — 2017-08-08

16:Color Out of Space — 2016-01-24

6:Bad boys for Life — 2020-01-17

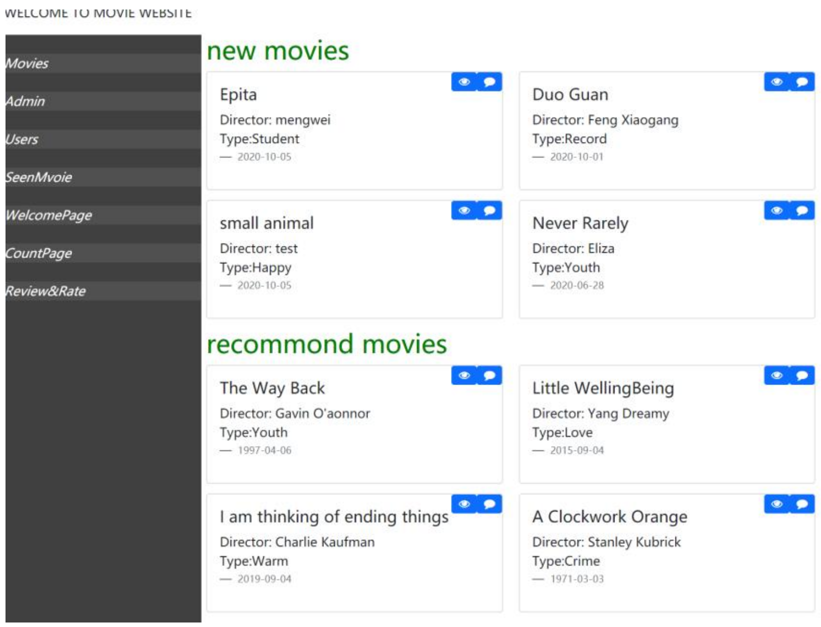
When you click the message icon, it will jump another Page -review Star page, choose a user and according to the movie introduction to make a review and star. There are 5 levels. (This corresponds to the fourth requirement of the document requirement,---rate film)


reviewstar works!

8:The Way Back
Author :Gavin O'aonnor
Type : Youth
— 1997-04-06

Review: Star :

3.5. Welcome page (welcome page with last seen films, new movies and recommendations)


WELCOME TO MOVIE WEBSITE

new movies

Epita
Director: mengwei
Type:Student
— 2020-10-05

Duo Guan
Director: Feng Xiaogang
Type:Record
— 2020-10-01

small animal
Director: test
Type:Happy
— 2020-10-05

Never Rarely
Director: Eliza
Type:Youth
— 2020-06-28

recommond movies

The Way Back
Director: Gavin O'aonnor
Type:Youth
— 1997-04-06

Little WellingBeing
Director: Yang Dreamy
Type:Love
— 2015-09-04

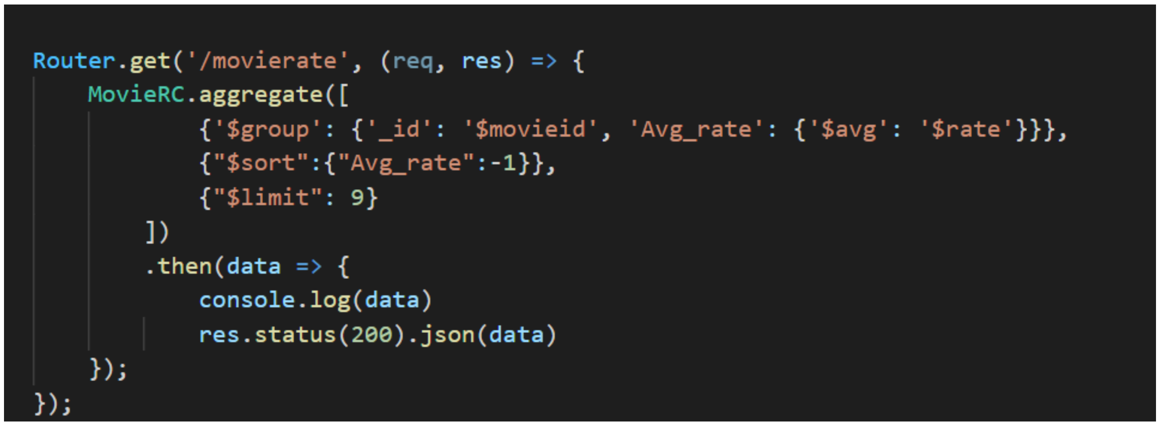
I am thinking of ending things
Director: Charlie Kaufman
Type:Warm
— 2019-09-04

A Clockwork Orange
Director: Stanley Kubrick
Type:Crime
— 1971-03-03

The part of new movies is according to the movie's realsed-time by using SQL sentence in PostgreSQL .

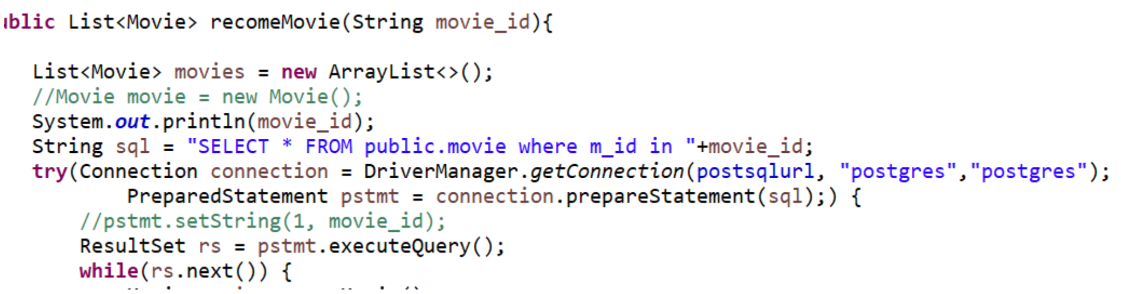
The part of recommend movies is according to the movies' average star in the mango dB database.

1.back-end code:



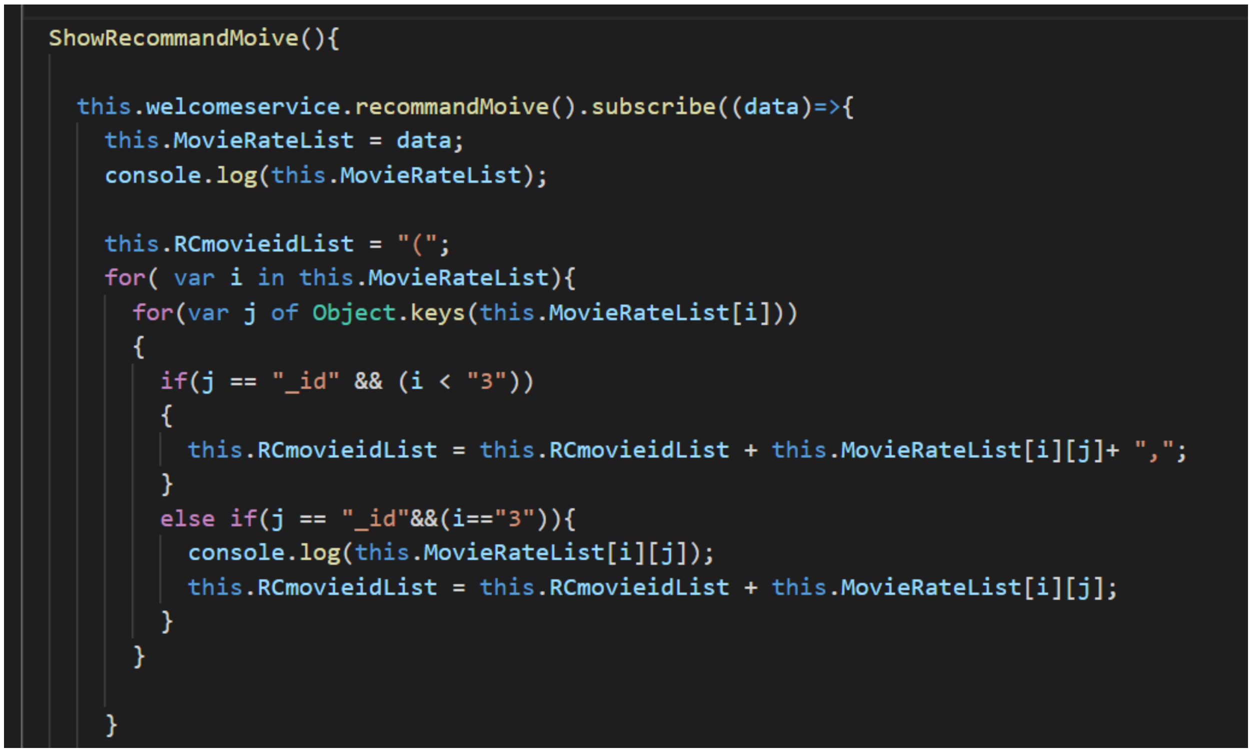
```
Router.get('/movierate', (req, res) => {
    MovieRC.aggregate([
            {'$group': {'_id': '$movieid', 'Avg_rate': {'$avg': '$rate'}}},
            {"$sort":{"Avg_rate":-1}},
            {"$limit": 9}
        ])
        .then(data => {
            console.log(data)
            res.status(200).json(data)
    });
});
```

it will give a list of movies id ,then in the java ,we use SQL sentences to find the movies by movie_id .



```
ıblic List<Movie> recomeMovie(String movie_id){

  List<Movie> movies = new ArrayList<>();
  //Movie movie = new Movie();
  System.out.println(movie_id);
  String sql = "SELECT * FROM public.movie where m_id in "+movie_id;
  try(Connection connection = DriverManager.getConnection(postsqlurl, "postgres","postgres");
          PreparedStatement pstmt = connection.prepareStatement(sql);) {
      //pstmt.setString(1, movie_id);
      ResultSet rs = pstmt.executeQuery();
      while(rs.next()) {
```

2.front-end code:

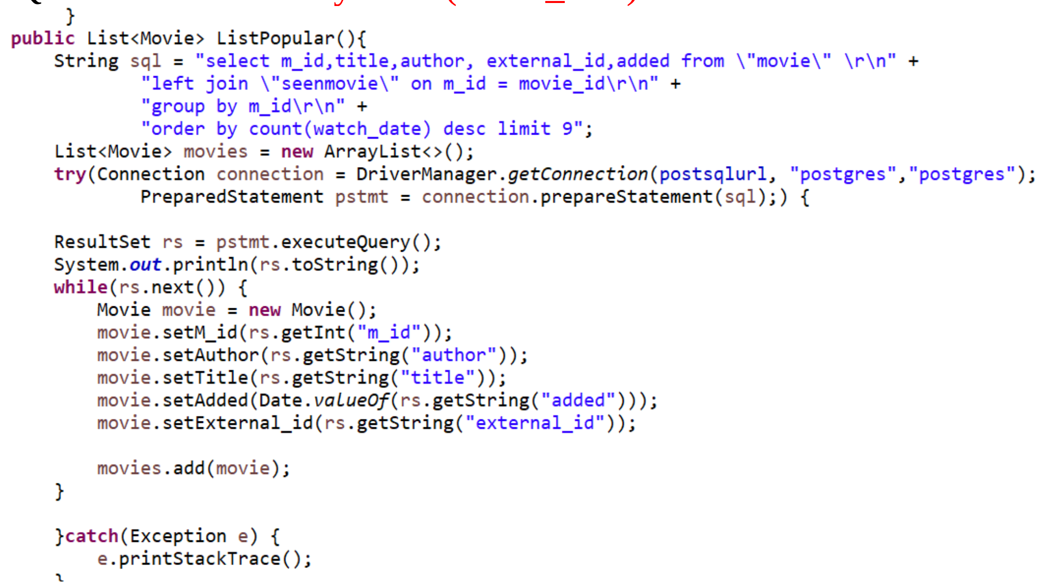


```
  ShowRecommandMoive(){

    this.welcomeservice.recommandMoive().subscribe((data)=>{
      this.MovieRateList = data;
      console.log(this.MovieRateList);

      this.RCmovieidList = "(";
      for( var i in this.MovieRateList){
        for(var j of Object.keys(this.MovieRateList[i]))
        {
          if(j == "_id" && (i < "3"))
          {
            this.RCmovieidList = this.RCmovieidList + this.MovieRateList[i][j]+ ",";
          }
          else if(j == "_id"&&(i=="3")){
            console.log(this.MovieRateList[i][j]);
            this.RCmovieidList = this.RCmovieidList + this.MovieRateList[i][j];
          }
        }

      }
```

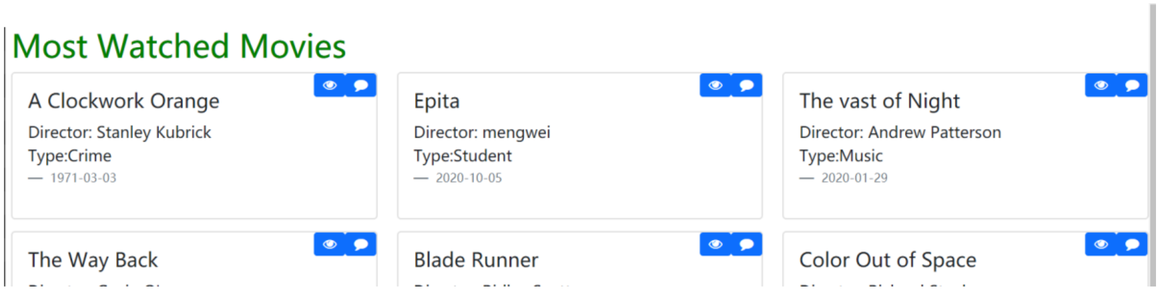
3.6. CountPage ((10 most popular movies, 10 most viewed movies).

In this countPage, there are two parts, one is most watched movies, the other is popular movie (it is same with recommend movie part),I made 9 because it is neat in the page).

the most watched movie is to calculate the watch times of movies by using SQL sentences order by count(watch_date)



```
    }
public List<Movie> ListPopular(){
    String sql = "select m_id,title,author, external_id,added from \"movie\" \r\n" +
            "left join \"seenmovie\" on m_id = movie_id\r\n" +
            "group by m_id\r\n" +
            "order by count(watch_date) desc limit 9";
    List<Movie> movies = new ArrayList<>();
    try(Connection connection = DriverManager.getConnection(postsqlurl, "postgres","postgres");
            PreparedStatement pstmt = connection.prepareStatement(sql);) {

    ResultSet rs = pstmt.executeQuery();
    System.out.println(rs.toString());
    while(rs.next()) {
        Movie movie = new Movie();
        movie.setM_id(rs.getInt("m_id"));
        movie.setAuthor(rs.getString("author"));
        movie.setTitle(rs.getString("title"));
        movie.setAdded(Date.valueOf(rs.getString("added")));
        movie.setExternal_id(rs.getString("external_id"));

        movies.add(movie);
    }

    }catch(Exception e) {
        e.printStackTrace();
```



Most Watched Movies

A Clockwork Orange
Director: Stanley Kubrick
Type:Crime
— 1971-03-03

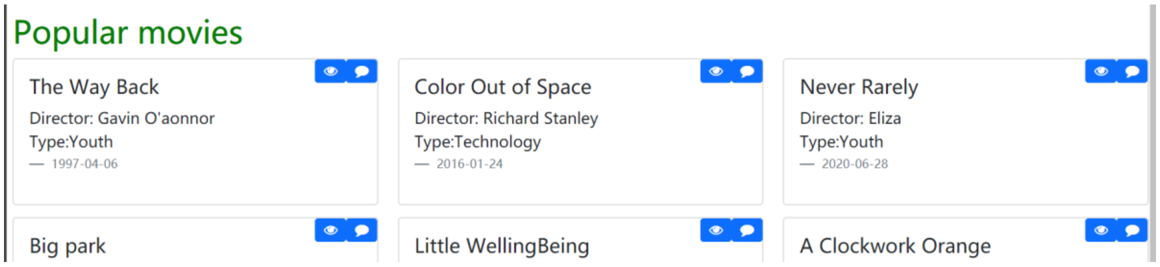
Epita
Director: mengwei
Type:Student
— 2020-10-05

The vast of Night
Director: Andrew Patterson
Type:Music
— 2020-01-29

The Way Back

Blade Runner

Color Out of Space



Popular movies

The Way Back
Director: Gavin O'aonnor
Type:Youth
— 1997-04-06

Color Out of Space
Director: Richard Stanley
Type:Technology
— 2016-01-24

Never Rarely
Director: Eliza
Type:Youth
— 2020-06-28

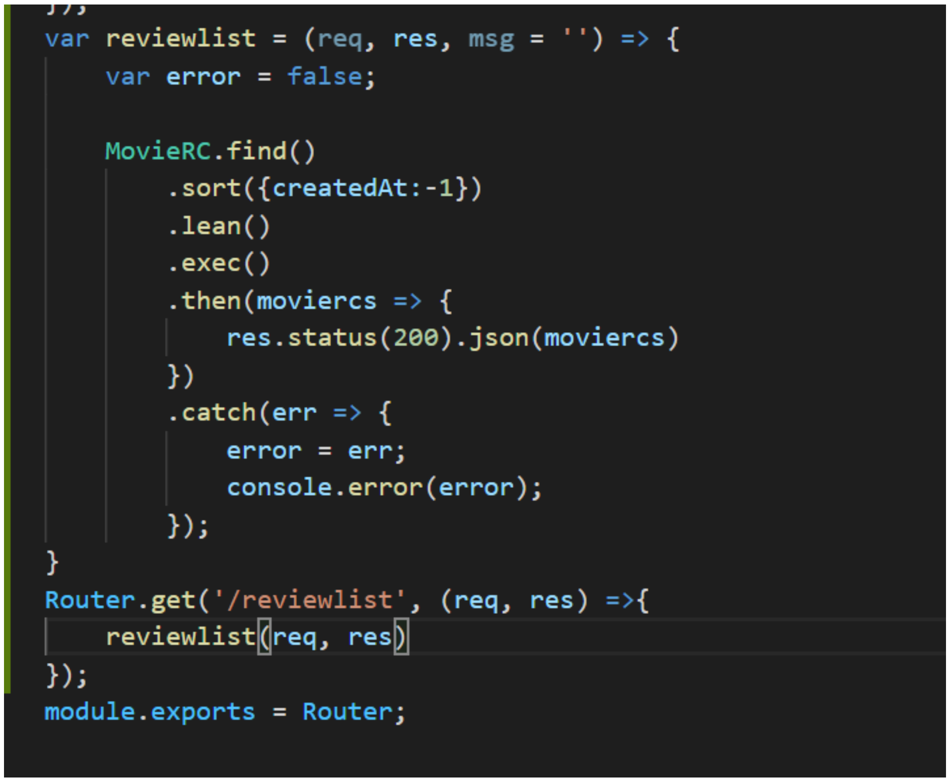
Big park

Little WellingBeing

A Clockwork Orange

3.7. Review Rate page

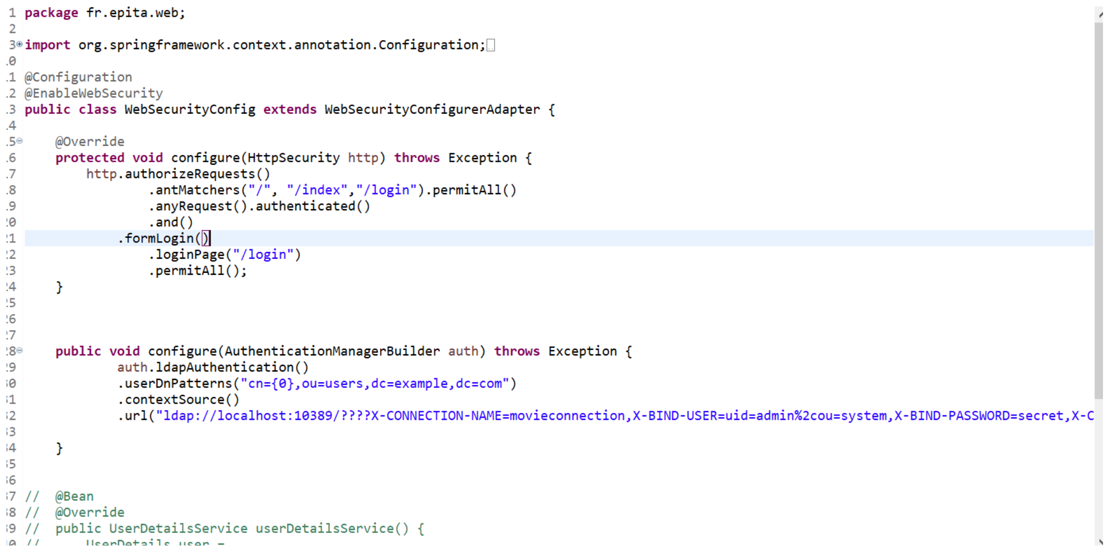
In this page , you could search all of users review and stars of each movie. it is done in mongo dB.



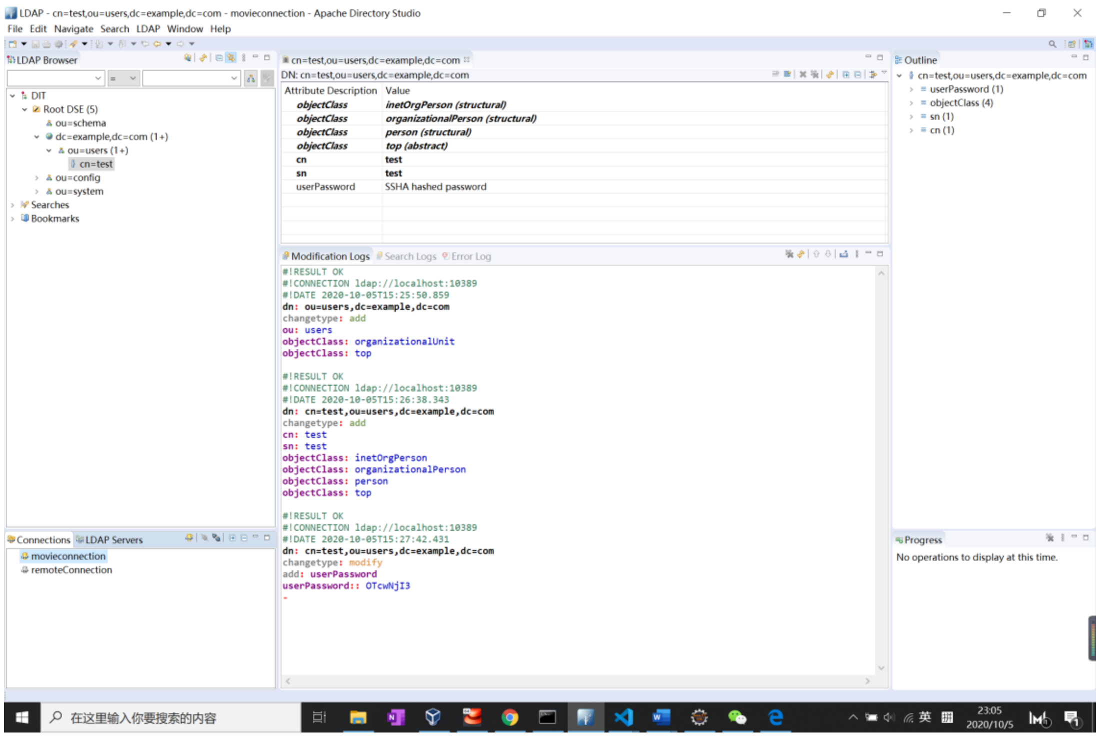
```
});
var reviewlist = (req, res, msg = '') => {
    var error = false;

    MovieRC.find()
        .sort({createdAt:-1})
        .lean()
        .exec()
        .then(moviercs => {
            res.status(200).json(moviercs)
        })
        .catch(err => {
            error = err;
            console.error(error);
        });
}
Router.get('/reviewlist', (req, res) =>{
    reviewlist(req, res)
});
module.exports = Router;
```

3.8. login in Page (login in)

login Page uses the ApachDS to set username and password



```
package fr.epita.web;

import org.springframework.context.annotation.Configuration;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .antMatchers("/", "/index","/login").permitAll()
                .anyRequest().authenticated()
                .and()
            .formLogin()
                .loginPage("/login")
                .permitAll();
    }



    public void configure(AuthenticationManagerBuilder auth) throws Exception {
            auth.ldapAuthentication()
            .userDnPatterns("cn={0},ou=users,dc=example,dc=com")
            .contextSource()
            .url("ldap://localhost:10389/????X-CONNECTION-NAME=movieconnection,X-BIND-USER=uid=admin%2cou=system,X-BIND-PASSWORD=secret,X-C

    }


//  @Bean
//  @Override
//  public UserDetailsService userDetailsService() {
//      UserDetails user =
```



DN: cn=test,ou=users,dc=example,dc=com

Attribute Description	Value
objectClass	*inetOrgPerson (structural)*
objectClass	*organizationalPerson (structural)*
objectClass	*person (structural)*
objectClass	*top (abstract)*
cn	test
sn	test
userPassword	SSHA hashed password

```
#!RESULT OK
#!CONNECTION ldap://localhost:10389
#!DATE 2020-10-05T15:25:50.859
dn: ou=users,dc=example,dc=com
changetype: add
ou: users
objectClass: organizationalUnit
objectClass: top

#!RESULT OK
#!CONNECTION ldap://localhost:10389
#!DATE 2020-10-05T15:26:38.343
dn: cn=test,ou=users,dc=example,dc=com
changetype: add
cn: test
sn: test
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top

#!RESULT OK
#!CONNECTION ldap://localhost:10389
#!DATE 2020-10-05T15:27:42.431
dn: cn=test,ou=users,dc=example,dc=com
changetype: modify
add: userPassword
userPassword:: OTcwNjI3
-
```

4. Experience

By this project, I am totally understanding how to separate the front and back ends and more familiar with Angular and Spring MVC. I also understood how to built a website in a real environment, but about the login in and using Apiman, I should practice and learn how to use it in my project later.