TECHNICAL DOCUMENTATION

SHENGYI ZHOU

05-10-2020

Introduction 2
Structure 2
  Databases 2
  Front-end 4
  Back-end 5
Features 6
  User authentication 6
  Welcome page 6
  Play film 7
  Rate film 7
  Compute stats 8
  Api management 8

Introduction

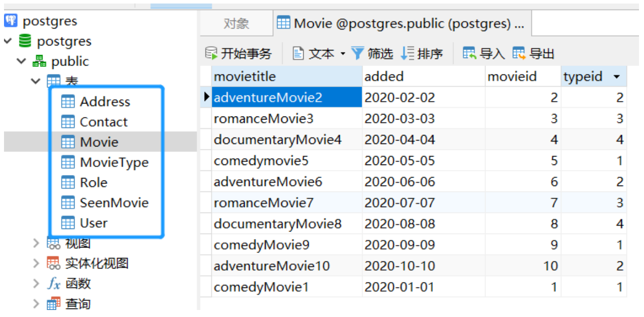
This is a web app about movies. Users can log in to watch movies, rate and comment. Administrator-level users can also add, delete, and modify movies. It’s necessary to login before any operation.

Databases : Postgresql (Docker) & MongoDB
Front-end : Angular2
Back-end: Express
Features: User authentication, Welcome page, Play film, Rate film, Compute stats, API gateway

Structure

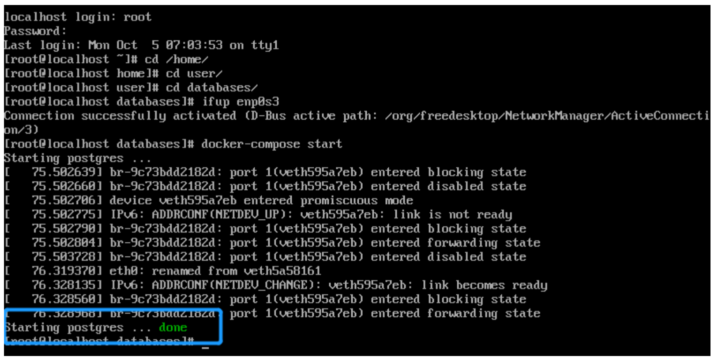
Databases

Postgresql (Docker):



movietitle	added	movieid	typeid
adventureMovie2	2020-02-02	2	2
romanceMovie3	2020-03-03	3	3
documentaryMovie4	2020-04-04	4	4
comedymovie5	2020-05-05	5	1
adventureMovie6	2020-06-06	6	2
romanceMovie7	2020-07-07	7	3
documentaryMovie8	2020-08-08	8	4
comedyMovie9	2020-09-09	9	1
adventureMovie10	2020-10-10	10	2
comedyMovie1	2020-01-01	1	1

Image pgdb



```
localhost login: root
Password:
Last login: Mon Oct  5 07:03:53 on tty1
[root@localhost ~]# cd /home/
[root@localhost home]# cd user/
[root@localhost user]# cd databases/
[root@localhost databases]# ifup enp0s3
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnecti
on/3)
[root@localhost databases]# docker-compose start
Starting postgres ...
[   75.502639] br-9c73bdd2182d: port 1(veth595a7eb) entered blocking state
[   75.502660] br-9c73bdd2182d: port 1(veth595a7eb) entered disabled state
[   75.502706] device veth595a7eb entered promiscuous mode
[   75.502775] IPv6: ADDRCONF(NETDEV_UP): veth595a7eb: link is not ready
[   75.502790] br-9c73bdd2182d: port 1(veth595a7eb) entered blocking state
[   75.502804] br-9c73bdd2182d: port 1(veth595a7eb) entered forwarding state
[   75.503728] br-9c73bdd2182d: port 1(veth595a7eb) entered disabled state
[   76.319370] eth0: renamed from veth5a58161
[   76.328135] IPv6: ADDRCONF(NETDEV_CHANGE): veth595a7eb: link becomes ready
[   76.328560] br-9c73bdd2182d: port 1(veth595a7eb) entered blocking state
[   76.328968] br-9c73bdd2182d: port 1(veth595a7eb) entered forwarding state
Starting postgres ... done
[root@localhost databases]# _
```

Image Docker

This is where data about movies and users are stored, and it is also where user login information is stored. The relationship between tables is constructed through foreign keys, ex: we have typeid in “Movie” table, which is the PR key of “MovieType”.

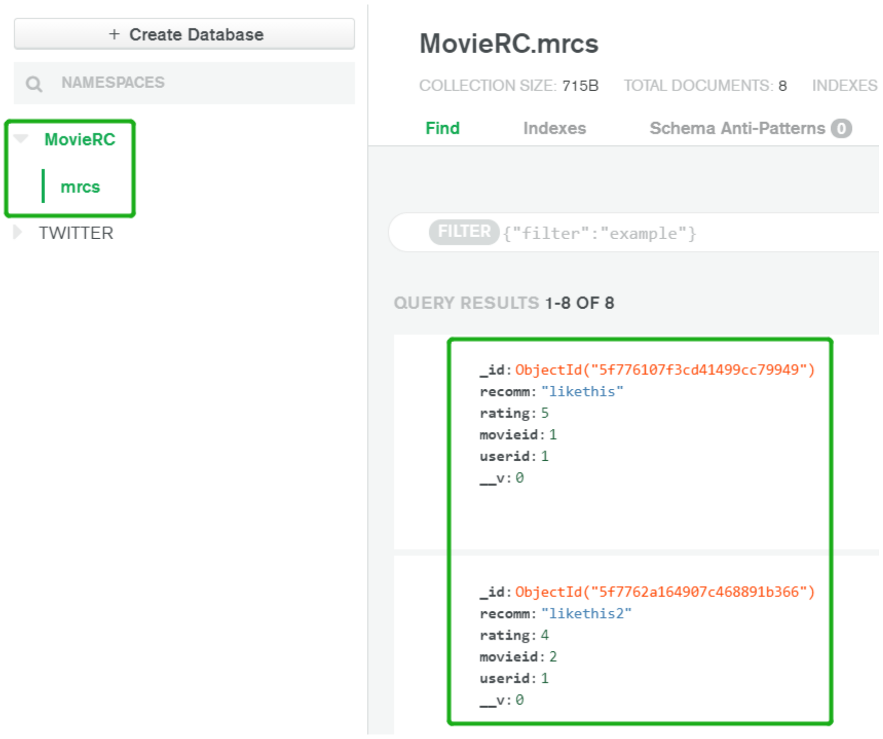
The primary key of each table is automatically added as showed above.

This database server is in docker.

```
movieid integer NOT NULL DEFAULT nextval('"Movie_movieid_seq"'::regclass),
typeid integer,
CONSTRAINT "Movie_pkey" PRIMARY KEY (movieid),
CONSTRAINT typeid FOREIGN KEY (typeid)
    REFERENCES public."MovieType" (typeid) MATCH SIMPLE
```

Image PR & FR keys

Mongodb:

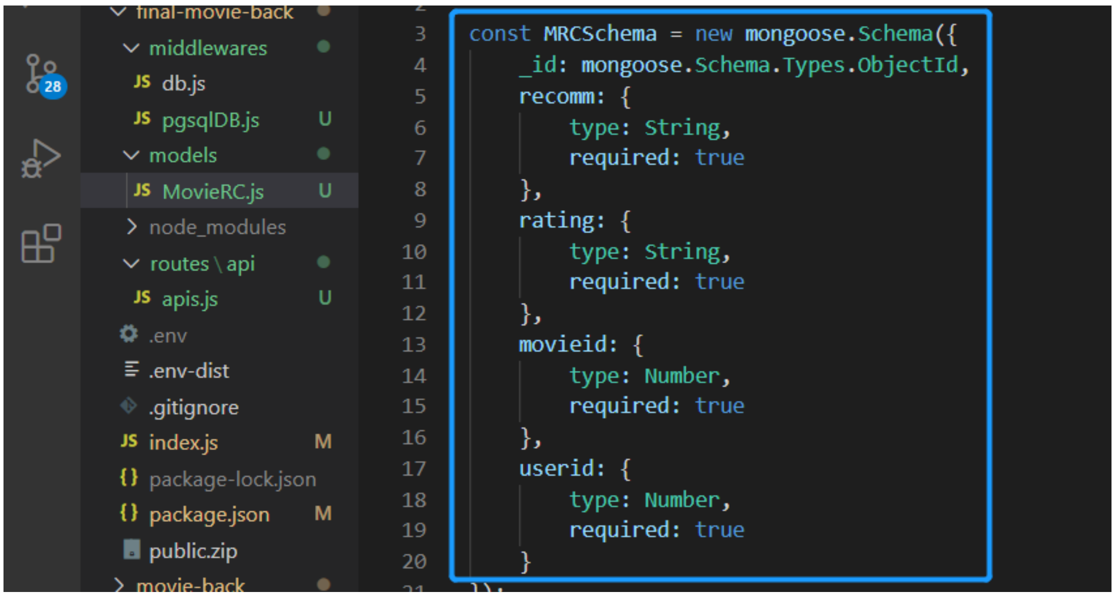


```
{"filter":"example"}
```

```
_id: ObjectId("5f776107f3cd41499cc79949")
recomm: "likethis"
rating: 5
movieid: 1
userid: 1
__v: 0

_id: ObjectId("5f7762a164907c468891b366")
recomm: "likethis2"
rating: 4
movieid: 2
userid: 1
__v: 0
```

Image mgdb

I store the ratings and reviews of the movie here. The database structure is defined in the backend, and _id is automatically generated

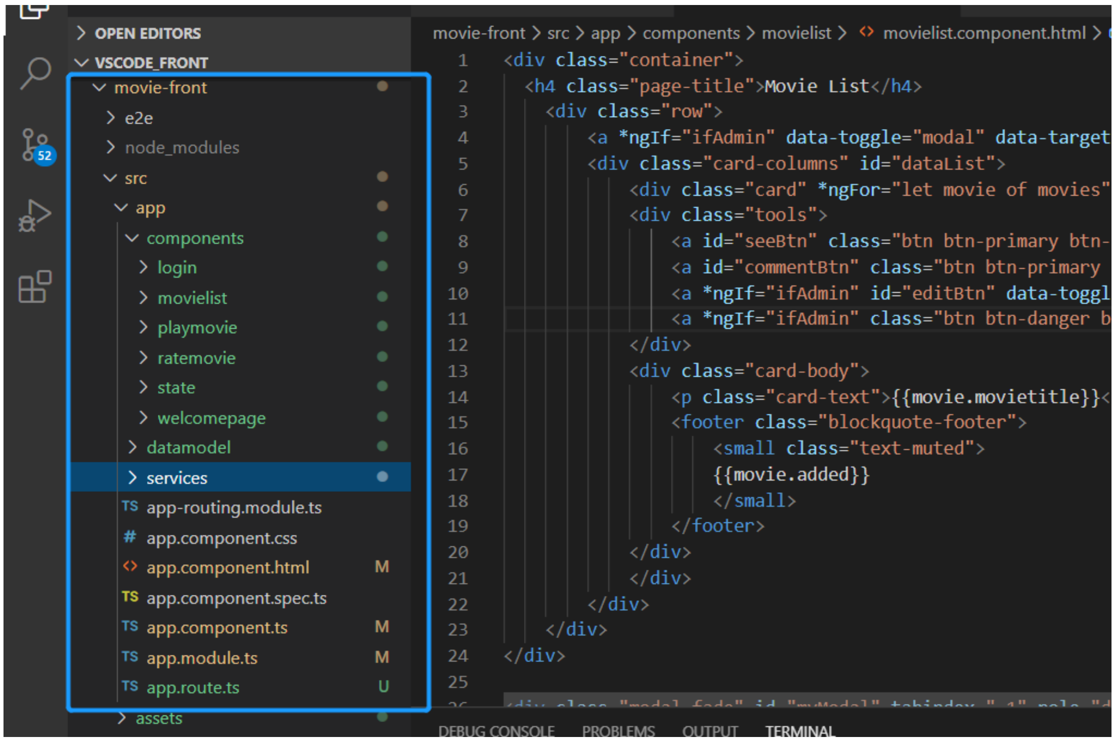


```
const MRCSchema = new mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    recomm: {
        type: String,
        required: true
    },
    rating: {
        type: String,
        required: true
    },
    movieid: {
        type: Number,
        required: true
    },
    userid: {
        type: Number,
        required: true
    }
```

Image mgdb structure

Front-end

Angular2



```
movie-front
  e2e
  node_modules
  src
    app
      components
        login
        movielist
        playmovie
        ratemovie
        state
        welcomepage
      datamodel
      services
      app-routing.module.ts
      app.component.css
      app.component.html        M
      app.component.spec.ts
      app.component.ts          M
      app.module.ts             M
      app.route.ts              U
    assets
```

movie-front > src > app > components > movielist > movielist.component.html

```html
<div class="container">
  <h4 class="page-title">Movie List</h4>
   <div class="row">
       <a *ngIf="ifAdmin" data-toggle="modal" data-target
       <div class="card-columns" id="dataList">
           <div class="card" *ngFor="let movie of movies"
           <div class="tools">
               <a id="seeBtn" class="btn btn-primary btn-
               <a id="commentBtn" class="btn btn-primary
               <a *ngIf="ifAdmin" id="editBtn" data-toggl
               <a *ngIf="ifAdmin" class="btn btn-danger b
           </div>
           <div class="card-body">
               <p class="card-text">{{movie.movietitle}}<
               <footer class="blockquote-footer">
                   <small class="text-muted">
                   {{movie.added}}
                   </small>
               </footer>
           </div>
           </div>
       </div>
   </div>
</div>
```

Image front-end

In order to facilitate the display of data, I used angular2 as the front end. But I found a flaw, I will explain it in the “Api management” section.

The top menu bar is generated from “app.component”, uses “RouterLink” for navigation. Each option in the menu corresponds to a component.

In addition to components, there are also “datamodel” and “services” in the front end. The “components” will call the method in the “services”, and then the ”services” send the requests to the back-end through the API. About “withCredentials” (in Image request), I will explain it in the “User authentication”.

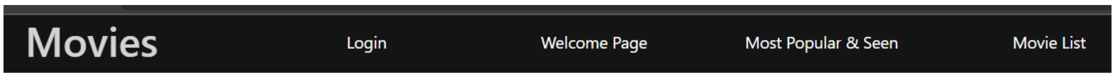
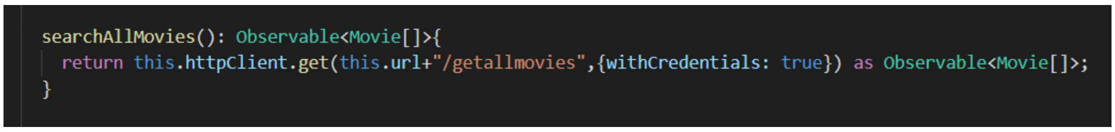


Image top menu bar

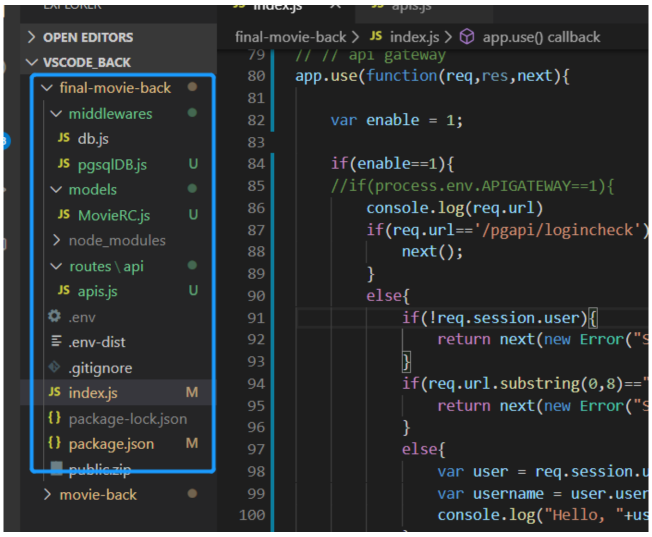


```typescript
searchAllMovies(): Observable<Movie[]>{
  return this.httpClient.get(this.url+"/getallmovies",{withCredentials: true}) as Observable<Movie[]>;
}
```

Image request

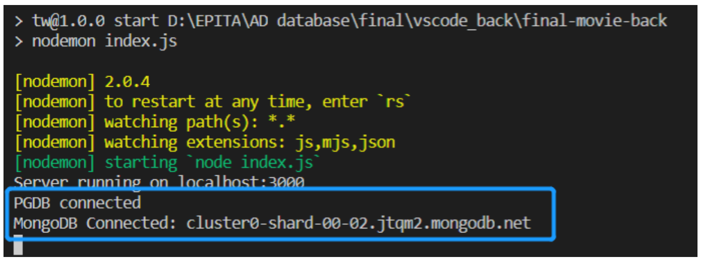
Back-end

Express



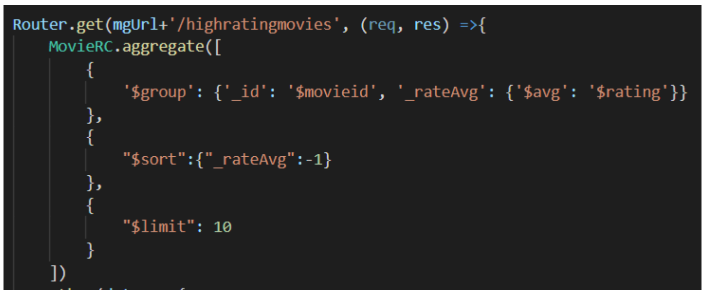
```
// // api gateway
app.use(function(req,res,next){

    var enable = 1;

    if(enable==1){
    //if(process.env.APIGATEWAY==1){
        console.log(req.url)
        if(req.url=='/pgapi/logincheck')
            next();
        }
        else{
            if(!req.session.user){
                return next(new Error("S
            }
            if(req.url.substring(0,8)=="
                return next(new Error("S
            }
            else{
                var user = req.session.u
                var username = user.user
                console.log("Hello, "+us
```

Image back-end



```
> tw@1.0.0 start D:\EPITA\AD database\final\vscode_back\final-movie-back
> nodemon index.js

[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server running on localhost:3000
PGDB connected
MongoDB Connected: cluster0-shard-00-02.jtqm2.mongodb.net
```

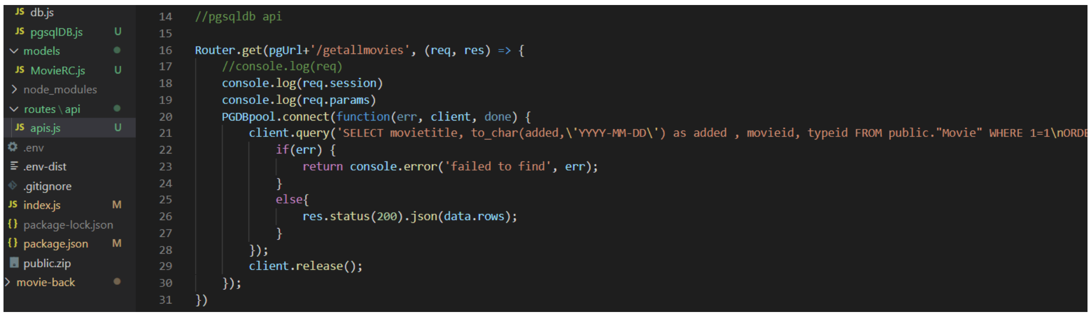
Image database connection



```
Router.get(mgUrl+'/highratingmovies', (req, res) =>{
    MovieRC.aggregate([
        {
            '$group': {'_id': '$movieid', '_rateAvg': {'$avg': '$rating'}}
        },
        {
            "$sort":{"_rateAvg":-1}
        },
        {
            "$limit": 10
        }
    ])
```

Image MGDB query

I used express on the back end to connect to the two databases.
For PGDB, I used “Router” to simply excute sql query.
For MGDB, we need to use a specific queries.



```
//pgsqldb api

Router.get(pgUrl+'/getallmovies', (req, res) => {
    //console.log(req)
    console.log(req.session)
    console.log(req.params)
    PGDBpool.connect(function(err, client, done) {
        client.query('SELECT movietitle, to_char(added,\'YYYY-MM-DD\') as added , movieid, typeid FROM public."Movie" WHERE 1=1\nORDE
            if(err) {
                return console.error('failed to find', err);
            }
            else{
                res.status(200).json(data.rows);
            }
        });
        client.release();
    });
})
```

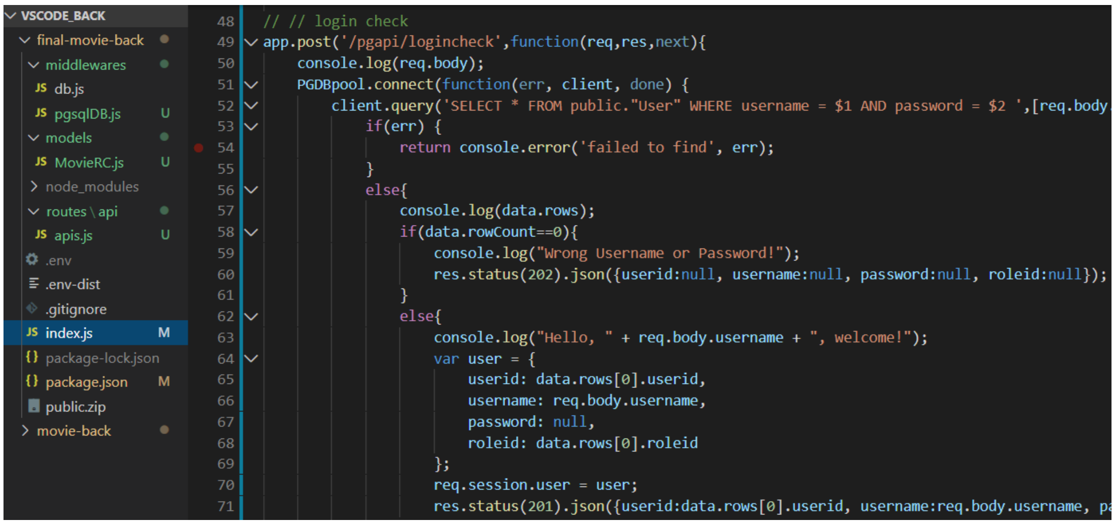
Image Router & sql query

Features

User authentication

The process of user login is mainly like this. The user enters the user name and password in the fore-end, and the data is received in the back-end and compared with the data in the database. If they are the same, put user data in the session and return, if they are inconsistent, an error will be reported.

After that, each request from the front-end, the session data will be added through “withCredentials: true”. The back-end will determine whether the user is logged in and has the right to use the API based on the session.



```
// // login check
app.post('/pgapi/logincheck',function(req,res,next){
    console.log(req.body);
    PGDBpool.connect(function(err, client, done) {
        client.query('SELECT * FROM public."User" WHERE username = $1 AND password = $2 ',[req.body
            if(err) {
                return console.error('failed to find', err);
            }
            else{
                console.log(data.rows);
                if(data.rowCount==0){
                    console.log("Wrong Username or Password!");
                    res.status(202).json({userid:null, username:null, password:null, roleid:null});
                }
                else{
                    console.log("Hello, " + req.body.username + ", welcome!");
                    var user = {
                        userid: data.rows[0].userid,
                        username: req.body.username,
                        password: null,
                        roleid: data.rows[0].roleid
                    };
                    req.session.user = user;
                    res.status(201).json({userid:data.rows[0].userid, username:req.body.username, pa
```

Image login function in “index.js”

Welcome page

Last seen films

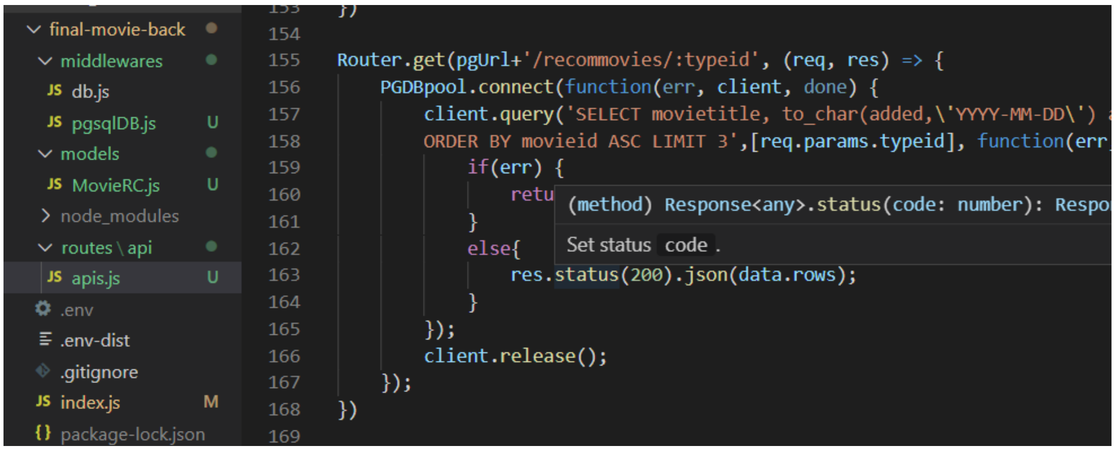
The last seen films are simply based on “SeenMovies” database.

New movies

I arrange the data in the database in reverse order, and take the first three of them.

Recommendations

According to the movies that the user has watched, I analyzed the user’s favorite movie genre and showed the recommended movies based on this genre (send the movie type as a param to the back-end).



```
})

Router.get(pgUrl+'/recommovies/:typeid', (req, res) => {
    PGDBpool.connect(function(err, client, done) {
        client.query('SELECT movietitle, to_char(added,\'YYYY-MM-DD\') 
        ORDER BY movieid ASC LIMIT 3',[req.params.typeid], function(err
            if(err) {
                retu (method) Response<any>.status(code: number): Respo
            }
            else{     Set status code .
                res.status(200).json(data.rows);
            }
        });
        client.release();
    });
})
```

Image function recommend films

Play film

After the user clicks the blue eye icon in the upper right corner of the movie tab, the page will jump to watch the movie. After clicking the play button, the “userid”, “movieid” and current time will be saved in the “SeenMovies” table.

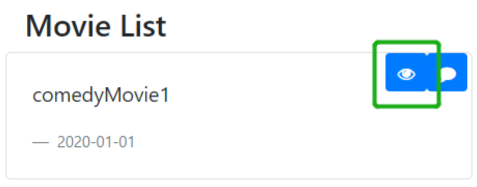


Image watch movie

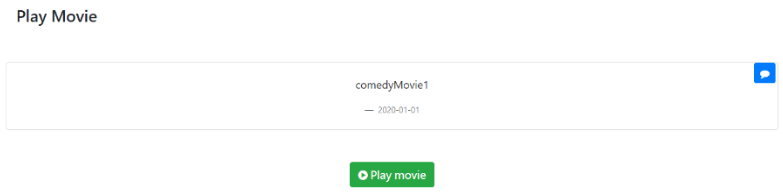


Image play movie page

Rate film

Same as the previous function, after the user clicks the blue comment icon in the upper right corner of the movie tab, the page will jump to rate & recomment the movie. After clicking the save button, the “userid”, “movieid”, rating and recommendation will be saved in the MongoDB.

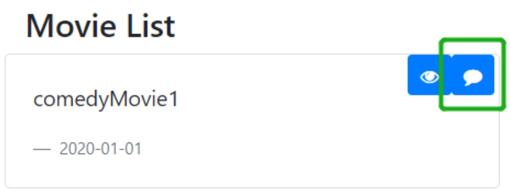


Image rate movie

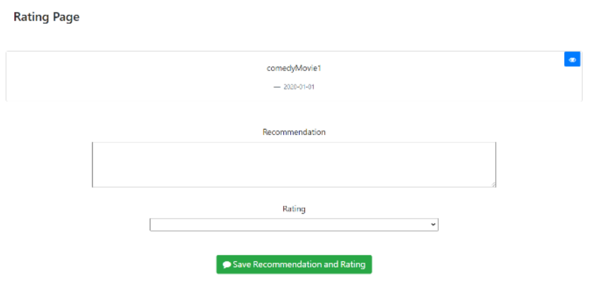


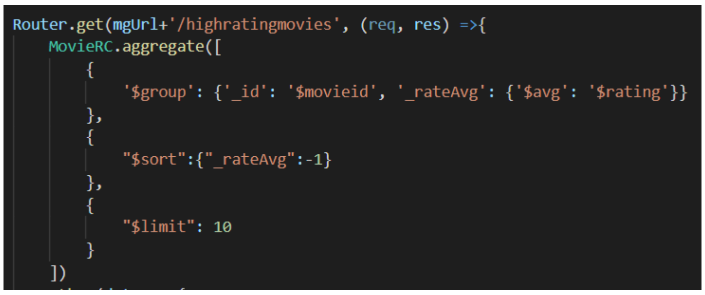
Image rate movie page

Compute stats

10 most popular movies

Since movie ratings are stored in mongodb, we need to use a special query to get movie ratings. “$group”, ”$avg”, “$sort”, “$limit” help me to get the average rating of each movie.

Then I select the first ten of them and get the movie INFO from “Movie” table in PGDB.



```
Router.get(mgUrl+'/highratingmovies', (req, res) =>{
    MovieRC.aggregate([
        {
            '$group': {'_id': '$movieid', '_rateAvg': {'$avg': '$rating'}}
        },
        {
            "$sort":{"_rateAvg":-1}
        },
        {
            "$limit": 10
        }
    ])
```

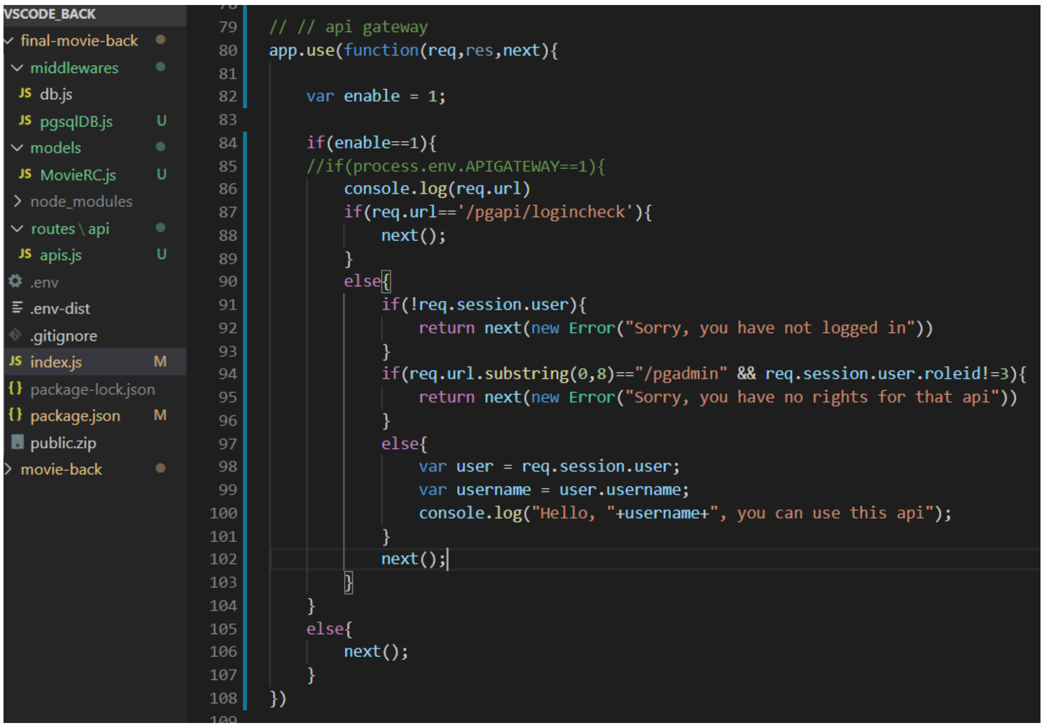
Image MongoDB query

10 most viewed movies

It’s much more simple than the previous one. I just count the data from the “SeenMovie” and get the movie INFO by the FR key movie_id.

Api management

For the API management, since I chose to create my own session as login, I didn’t use Apiman, instead of which, I create a simple API Gateway in the back-end part.



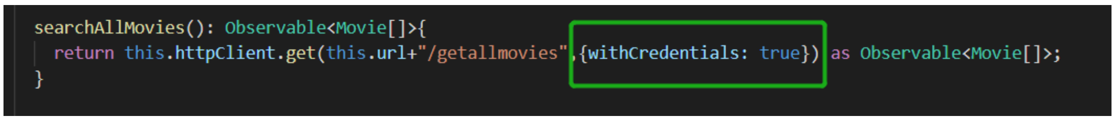
```
// // api gateway
app.use(function(req,res,next){

    var enable = 1;

    if(enable==1){
    //if(process.env.APIGATEWAY==1){
        console.log(req.url)
        if(req.url=='/pgapi/logincheck'){
            next();
        }
        else{
            if(!req.session.user){
                return next(new Error("Sorry, you have not logged in"))
            }
            if(req.url.substring(0,8)=="/pgadmin" && req.session.user.roleid!=3){
                return next(new Error("Sorry, you have no rights for that api"))
            }
            else{
                var user = req.session.user;
                var username = user.username;
                console.log("Hello, "+username+", you can use this api");
            }
            next();
        }
    }
    else{
        next();
    }
})
```

Image API gateway

First, we check whether the request from the front-end is a login request. If it is, we do not perform session verification and directly let the request pass.

If the request is not a login request, but some request from the administrator, we will verify whether the user in the session is an administrator (check “roleid” in the ”session.user”).

The last case is a request from an ordinary user. We only need to check whether there is a user in the session to determine whether the request is passed.



```
searchAllMovies(): Observable<Movie[]>{
  return this.httpClient.get(this.url+"/getallmovies",{withCredentials: true}) as Observable<Movie[]>;
}
```

Image withCredentials

The session data will be added in the request through “withCredentials: true”. But I found that, this only works for the GET request. And the people online have the same problem like me: we can’t pass the session through the POST request.

Reference: https://github.com/angular/angular/issues/24283

So, I changed all requests to GET methods, which put the params in the URL. I know that it will cause some security problems, but unfortunately I have no choice.

Thank you for reading !