电影售票系统详细设计报告

目录

[电影售票系统详细设计报告 1](#_Toc74767880)

[1 引言 3](#_Toc74767881)

[1.1标识 3](#_Toc74767882)

[1.2系统概述 3](#_Toc74767883)

[1.3文档概述 4](#_Toc74767884)

[1.4功能需求 4](#_Toc74767885)

[1.5基线 4](#_Toc74767886)

[2.引用文件 5](#_Toc74767887)

[3.详细系统设计 5](#_Toc74767888)

[3.1基本功能 5](#_Toc74767889)

[3.2附加功能 5](#_Toc74767890)

[3.3 基本设计概念和处理流程 6](#_Toc74767891)

[4.数据描述 6](#_Toc74767892)

[5.程序设计说明 13](#_Toc74767893)

[5.1 部分主体代码 13](#_Toc74767894)

[5.2功能 55](#_Toc74767895)

[5.3性能 56](#_Toc74767896)

[5.4输入项 56](#_Toc74767897)

[5.5输出项 56](#_Toc74767898)

[5.6算法 56](#_Toc74767899)

[5.7流程逻辑 58](#_Toc74767900)

[6.接口： 61](#_Toc74767901)

[7.注释设计 61](#_Toc74767902)

[8.限制条件 61](#_Toc74767903)

[9.测试计划 62](#_Toc74767904)

[10.尚未解决的问题 62](#_Toc74767905)

[11.注解 62](#_Toc74767906)

[12.总结 62](#_Toc74767907)

# 引言

现在，随着时代的发展，人们现在的生活节奏越来越快，生活工作压力也随之变大。很多人都在寻找着适合自己的减压方式，让自己的生活工作压力有所缓解。毋庸置疑的是，看电影是一种不错的解压方式，无论是缓解自己的负面情绪，还是让自己的状态更加完美，看电影都可以做到。

现在互联网的发展十分迅速，互联网的功能也十分强大。借助互联网，让我们看电影不再那么麻烦。现在网上有许多电影售票系统，可以说都是互联网崛起的功劳。许多售票系统的功能都大致相同，电影的管理，电影的搜索以及电影售票的功能。在线电影售票系统乘着互联网的东风，让我们看电影的方式越来越多样化，也让我们看电影的选择越来越多样化。这个在线电影售票系统主要就是实现电影售票的基本功能，借助互联网，实现电影的排片以及一些基本的搜索功能。当然，这个电影售票系统也能够实现选座功能。

## 1.1标识

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文件状态：  [ ]草稿  [ √ ]正式发布  [ ]正在修改 | 报告编号 | 电影售票系统 | | |
| 当前版本 | V1.0 | | |
| 编写人 | 本小组成员 | 编写日期 | 2020-6-9 |
| 审批人 | 本小组成员 | 审批日期 | 2019-6-9 |
| 保密级别 | 3 | | |

## 1.2系统概述

在线电影售票系统主要是利用互联网的优势，结合一些现有的网上电影售票系统，提供更加人性化的电影售票操作，包括座位的显示以及选择等等。不仅如此，该系统还提供一系列的基本操作，包括影片的评价，还有评论、用户推荐这些功能。当然，这个系统还增加了搜索操作，可以根据影片名称等进行搜索。

编写在线电影售票系统详细设计说明书的目的在于，从详细设计的角度明确在线电影售票系统项目的控件、功能范围以及它的设计思路，使系统开发人员和产品管理人员明确产品功能，可以有针对性的进行系统开发、测试、验收等各方面的工作。

，

读者：项目经理、概要设计人员，编码人员，测试人员。

## 1.3文档概述

在线电影售票系统总共有两个管理系统，后台管理系统和前端服务系统。后台管理系统主要是对用户基本信息进行管理，对用户的信息进行加密等等。后台管理系统是整个系统的安全系统，这个系统会对用户的信息进行严格保护，防止用户的信息泄露。该系统还是信息收集系统，这个系统会对用户的喜好数据进行收集并且分析，然后进行歌曲推荐。这个系统是连接网络的桥梁，这个系统会向前端系统发送必要的数据。而前端系统主要功能是接受来自后台系统的数据，并且进行展示。后台系统会根据用户的操作，包括点击输入等等，将一些必要的数据发送给后台，让后台访问数据库将对应的数据返回。也就是说，前端服务系统是用户和后台的桥梁，它的存在是让用户可以更加方便的获取想要的数据，并且隔绝用户直接的访问数据，这样有效的保护了数据，让数据以及一些基本信息更加安全。

本系统结合自己看电影时的经验，让这个系统更加贴合用户，让这个系统更加人性化。

## 1.4功能需求

1. 系统中有多个电影院，系统管理员可以完成电影院的维护，系统管理员可以为每个电影院指派1个电影院管理员；

2. 电影院管理员定期维护本电影院即将上映的电影信息；

4. 网民可以根据时间、电影名称、电影院名称进行查询，查询到自己中意的电影后，注册的网民可以在网上完成订票，并进行网上支付；

5. 系统能够对指定时间、电影院、电影名字进行统计分析，以便分析出受欢迎的电影

## 1.5基线

1**.** 在线电影售票系统软件开发计划书 V1.0

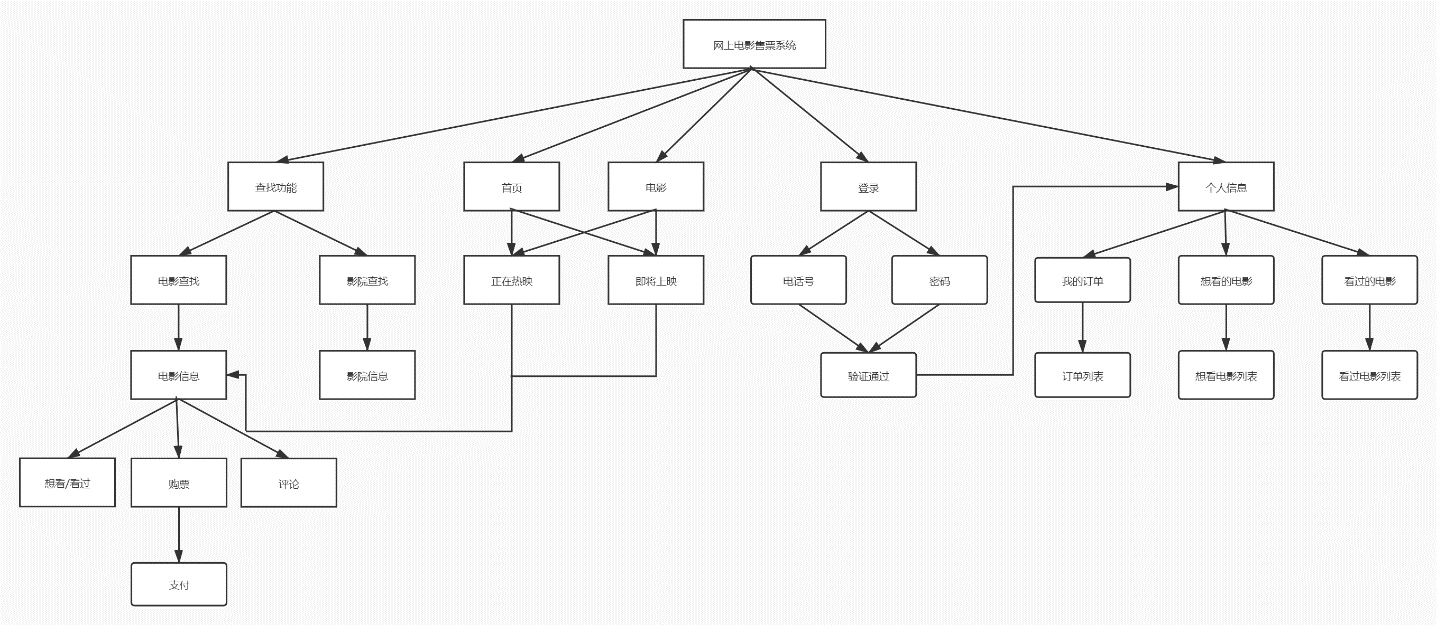
2. 在线电影售票系统需求分析报告 V1.0

3. 在线电影售票系统软件测试计划 V1.0

4. 在线电影售票系统概要设计说明书 V1.0

# 2.引用文件

# 3.详细系统设计



## 3.1基本功能

(a) 基于web的电影购票网站

(b) 注册登录用户可在线浏览热映电影信息，电影排行榜，;

(c) 注册登录用户查看附近影院，支持在线购票，实时支付等；

(d) 用户在观影结束后，可以对影院环境以及影片质量进行评价

(e) 系统UI界面简洁大方，操作简单，实时更新电影影院数据。

(f) 具有良好的平台可移植性，支持多人同时在线访问。

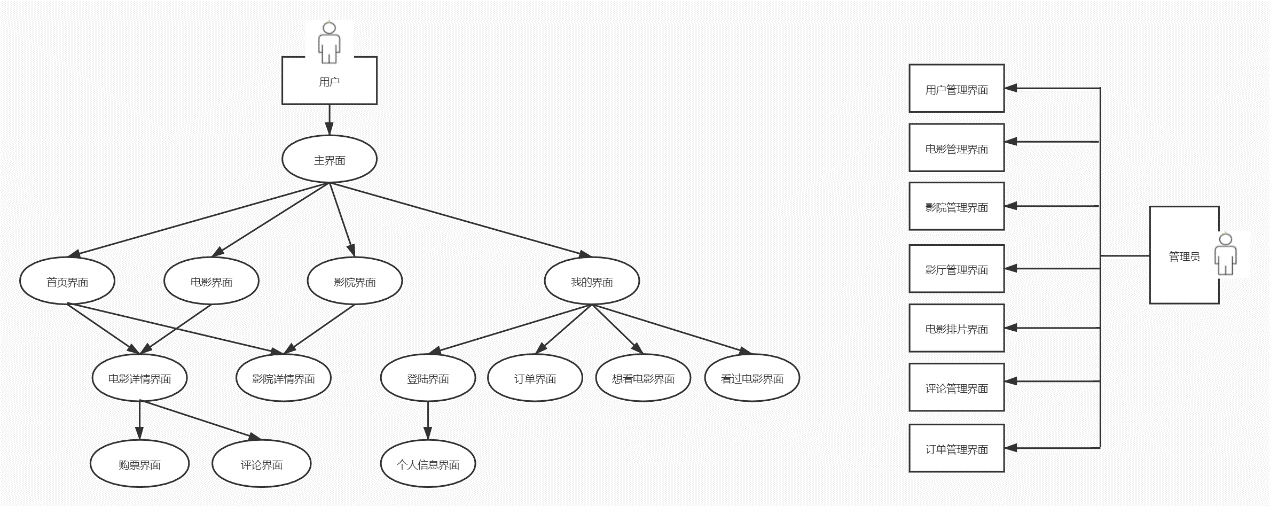
(g) 网页刷新时间等待时间不超过2s；用户能够流畅使用各项功能。

## 3.2附加功能

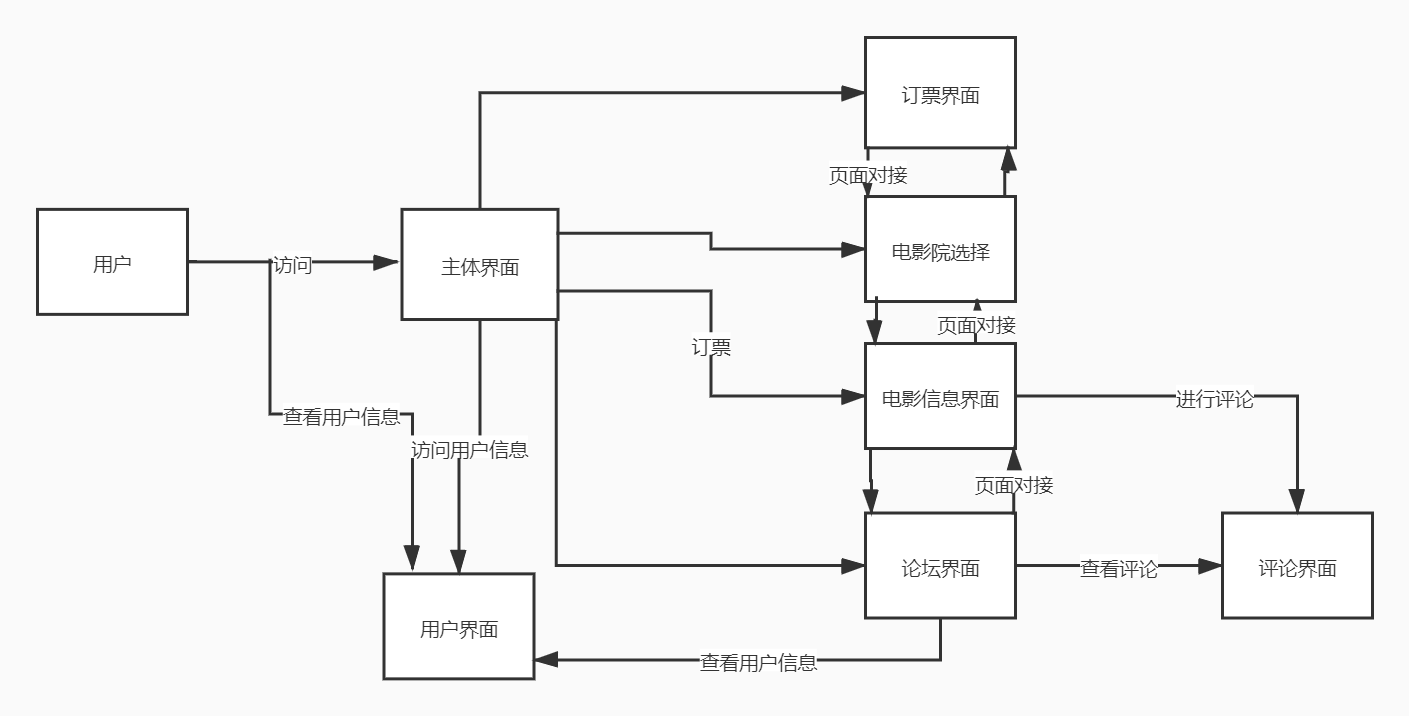
1. 安全与保密功能系统涉及用户私密信息，对于整个系统，需要完整的权限控制，每个用户分配特定的用户视图所应有的访问权限，防止某些人恶意的攻击系统，修改原始记录。设置必要的异常处理机制，比如当出现系统未知的操作时。也会弹出相应的错误提示等，以尽可能地保证用户体验。
2. 客户账号管理功能：管理人员能够列表显示用户的帐号信息、禁用或删除选定的用户帐号，在收到客户请求之后也可更改客户的账号信息。
3. 通知功能：管理人员可通过软件内消息向用户发送通知。
4. 电子邮件/短信服务：管理人员能够在客户账号信息出现异常时通过绑定的电子邮箱/手机号码向用户发送邮件/短信提醒。
5. 评论维护功能：对于违规评论，用户可向管理人员发送反馈，反馈人数达到一定程度即向管理人员显示该条消息，管理人员可以对其进行删除并发送消息通知该评论所属用户

## 3.3 基本设计概念和处理流程

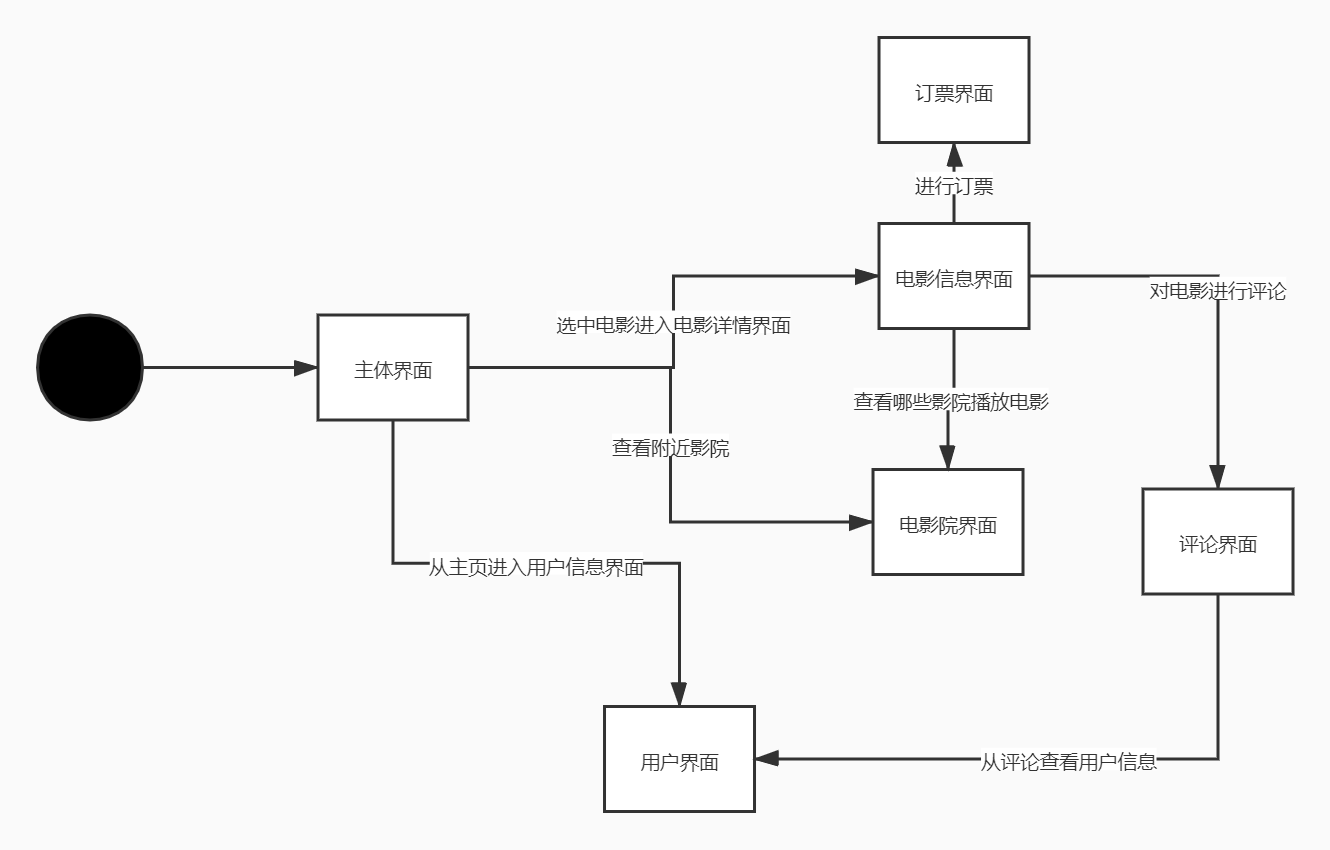
该网络售票系统对应着一个Web类型的解决方案。包括以下几大模块：



# 4.数据描述



界面分工：



后台数据表情况：

-- MySQL dump 10.13 Distrib 8.0.23, for Win64 (x86\_64)

--

-- Host: localhost Database: db\_film

-- ------------------------------------------------------

-- Server version 8.0.23

--

-- Table structure for table `t\_admin`

--

DROP TABLE IF EXISTS `t\_admin`;

CREATE TABLE `t\_admin` (

`admin\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '管理员id',

`name` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '管理员姓名',

`password` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '管理员密码',

`sex` varchar(10) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '管理员性别',

`phone` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '管理员联系电话',

`birthday` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '出生年月日',

`avatar` varchar(255) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '管理员头像',

PRIMARY KEY (`admin\_id`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

--

-- Table structure for table `t\_cinema`

--

DROP TABLE IF EXISTS `t\_cinema`;

CREATE TABLE `t\_cinema` (

`cinema\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '影院id',

`cinema\_name` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影院名称',

`cinema\_phone` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影院电话',

`province` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影院所在省份',

`city` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影院所在市',

`county` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影院所在区县',

`specified\_address` varchar(255) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影院详细地址',

PRIMARY KEY (`cinema\_id`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

--

-- Table structure for table `t\_comment`

--

DROP TABLE IF EXISTS `t\_comment`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

CREATE TABLE `t\_comment` (

`comment\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '评论id',

`user\_id` int unsigned DEFAULT NULL COMMENT '用户id',

`movie\_id` int unsigned DEFAULT NULL COMMENT '电影id',

`user\_score` int unsigned DEFAULT NULL COMMENT '用户评分',

`comment\_content` text CHARACTER SET utf8 COLLATE utf8\_unicode\_ci COMMENT '用户评论内容',

`comment\_date` datetime(6) DEFAULT NULL COMMENT '评论日期',

`support\_num` int unsigned DEFAULT NULL COMMENT '点赞数',

`is\_pass` int DEFAULT '0' COMMENT '评论是否通过审核',

`support\_user` text CHARACTER SET utf8 COLLATE utf8\_unicode\_ci COMMENT '点赞用户数组',

PRIMARY KEY (`comment\_id`) USING BTREE,

KEY `t\_comment\_ibfk\_1` (`user\_id`) USING BTREE,

KEY `t\_comment\_ibfk\_2` (`movie\_id`) USING BTREE,

CONSTRAINT `t\_comment\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `t\_user` (`user\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `t\_comment\_ibfk\_2` FOREIGN KEY (`movie\_id`) REFERENCES `t\_movie` (`movie\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=59 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `t\_hall`

--

DROP TABLE IF EXISTS `t\_hall`;

CREATE TABLE `t\_hall` (

`hall\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '影厅id',

`cinema\_id` int unsigned DEFAULT NULL COMMENT '影院id',

`name` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影厅名称',

PRIMARY KEY (`hall\_id`) USING BTREE,

KEY `t\_hall\_ibfk\_1` (`cinema\_id`) USING BTREE,

CONSTRAINT `t\_hall\_ibfk\_1` FOREIGN KEY (`cinema\_id`) REFERENCES `t\_cinema` (`cinema\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=41 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `t\_movie`

--

DROP TABLE IF EXISTS `t\_movie`;

CREATE TABLE `t\_movie` (

`movie\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '电影id',

`name` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '电影名称',

`poster` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '电影海报',

`director` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '导演',

`actor` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '主演',

`movie\_long` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '电影片长',

`language` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '电影语言版本',

`intro` text CHARACTER SET utf8 COLLATE utf8\_unicode\_ci COMMENT '电影简介',

`type` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '电影类型',

`public\_date` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '电影上映时间',

`wish\_num` int unsigned DEFAULT '0' COMMENT '想看人数',

`score` decimal(4,1) DEFAULT NULL COMMENT '电影综合评分',

PRIMARY KEY (`movie\_id`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=39 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `t\_order`

--

DROP TABLE IF EXISTS `t\_order`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `t\_order` (

`order\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '订单id',

`user\_id` int unsigned DEFAULT NULL COMMENT '用户id',

`schedule\_id` int unsigned DEFAULT NULL COMMENT '影院id',

`order\_phone` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '下单手机',

`order\_date` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户下单时间',

`ticket\_num` int unsigned DEFAULT NULL COMMENT '购买电影票数',

`ticket\_total\_price` decimal(10,2) unsigned DEFAULT NULL COMMENT '电影票单价',

`order\_seat\_info` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '座位信息',

`pay\_type` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '支付方式',

`phone\_code` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '手机取票码',

PRIMARY KEY (`order\_id`) USING BTREE,

KEY `t\_order\_ibfk\_1` (`user\_id`) USING BTREE,

KEY `t\_order\_ibfk\_2` (`schedule\_id`) USING BTREE,

CONSTRAINT `t\_order\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `t\_user` (`user\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `t\_order\_ibfk\_2` FOREIGN KEY (`schedule\_id`) REFERENCES `t\_schedule` (`schedule\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=102 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `t\_schedule`

--

DROP TABLE IF EXISTS `t\_schedule`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `t\_schedule` (

`schedule\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '排片id',

`movie\_id` int unsigned DEFAULT NULL COMMENT '电影id',

`cinema\_id` int unsigned DEFAULT NULL COMMENT '影院id',

`hall\_name` varchar(50) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '影厅名字',

`show\_date` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '放映日期',

`show\_time` varchar(10) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '放映时间',

`price` decimal(10,2) DEFAULT NULL COMMENT '电影售价',

`seat\_info` text CHARACTER SET utf8 COLLATE utf8\_unicode\_ci COMMENT '座位信息',

PRIMARY KEY (`schedule\_id`) USING BTREE,

KEY `hall\_id` (`hall\_name`) USING BTREE,

KEY `t\_schedule\_ibfk\_1` (`movie\_id`) USING BTREE,

KEY `t\_schedule\_ibfk\_2` (`cinema\_id`) USING BTREE,

CONSTRAINT `t\_schedule\_ibfk\_1` FOREIGN KEY (`movie\_id`) REFERENCES `t\_movie` (`movie\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `t\_schedule\_ibfk\_2` FOREIGN KEY (`cinema\_id`) REFERENCES `t\_cinema` (`cinema\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=102 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `t\_user`

--

DROP TABLE IF EXISTS `t\_user`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `t\_user` (

`user\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '用户id',

`user\_name` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户名',

`avatar` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户头像',

`password` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户密码',

`phone` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户手机号码',

`sex` varchar(10) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户性别',

`birthday` varchar(20) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户出生年月日',

`sign` varchar(100) CHARACTER SET utf8 COLLATE utf8\_unicode\_ci DEFAULT NULL COMMENT '用户签名',

PRIMARY KEY (`user\_id`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=55 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Table structure for table `t\_wishmovie`

--

DROP TABLE IF EXISTS `t\_wishmovie`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `t\_wishmovie` (

`wishMovie\_id` int unsigned NOT NULL AUTO\_INCREMENT COMMENT '想看电影id',

`user\_id` int unsigned DEFAULT NULL COMMENT '用户id',

`movie\_id` int unsigned DEFAULT NULL COMMENT '电影id',

PRIMARY KEY (`wishMovie\_id`) USING BTREE,

KEY `t\_wishmovie\_ibfk\_1` (`user\_id`) USING BTREE,

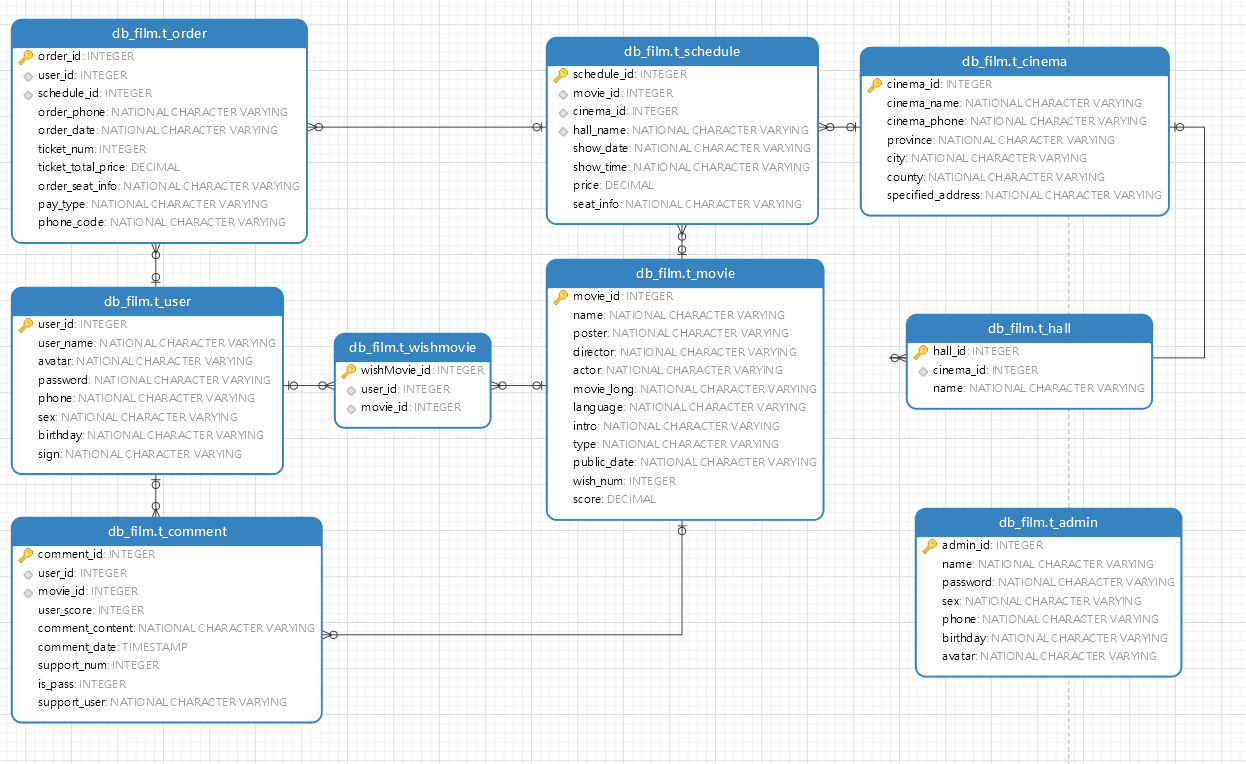
KEY `t\_wishmovie\_ibfk\_2` (`movie\_id`) USING BTREE,

CONSTRAINT `t\_wishmovie\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `t\_user` (`user\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `t\_wishmovie\_ibfk\_2` FOREIGN KEY (`movie\_id`) REFERENCES `t\_movie` (`movie\_id`) ON DELETE CASCADE ON UPDATE CASCADE

) ENGINE=InnoDB AUTO\_INCREMENT=119 DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci ROW\_FORMAT=DYNAMIC;

-- Dump completed on 2021-05-26 19:37:39



# 5.程序设计说明

## 5.1 部分主体代码

const express = require('express');

const router = express.Router();

const conn = require('../db/db');

const svgCaptcha = require('svg-captcha');

const util = require('../util/util');

const multer = require('multer');

// 用户API

let user = {};

/\* GET home page. \*/

router.get('/', function(req, res, next) {

res.render('index', { title: 'Express' });

});

//获取手机验证码

router.get('/api/getPhoneCode', function(req, res) {

let phone = req.query.phone;

let phoneCode = util.randomCode(4);

if (!phoneCode) {

res.json({error\_code:1,message:'获取验证码失败'});

}else{

user[phone] = phoneCode;

res.json({success\_code:200,data:phoneCode});

}

});

//获取图形验证码

router.get('/api/captcha', function (req, res) {

let captcha = svgCaptcha.create({

size: 4, // size of random string

ignoreChars: '0o1i',// filter out some characters like 0o1i

noise: 3 ,// number of noise lines

color: true, //, characters will have distinct colors instead of grey, true if background option is set

background: '#fff', // background color of the svg image

});

//保存图形验证码文本

req.session.captcha = captcha.text.toLowerCase();

//返回验证码数据

res.type('svg');

res.status(200).send(captcha.data);

});

//手机登录

router.post('/api/phoneLogin',function(req, res){

let phone = req.body.phone;

let phoneCode = req.body.phoneCode;

//判断手机验证码是否正确

if (user[phone]===phoneCode) {

let sqlStr = 'SELECT \* from t\_user WHERE phone = ? LIMIT 1 ;';

conn.query(sqlStr,[phone],(error,result,field)=>{

if (error){

res.json({error\_code: 1,message: '请求数据失败'})

} else{

result = JSON.parse(JSON.stringify(result));

//用户存在

if (result[0]){

req.session.userId = result[0].user\_id;

res.cookie('user\_id',result[0].user\_id);

res.json({success\_code: 200});

} else{

//用户不存在

let sqlStr = 'INSERT INTO t\_user(user\_name,phone,avatar,password) VALUES(?,?,?,?)';

let avatarSrc = '/images/avatar/monkey.png';

conn.query(sqlStr,[phone,phone,avatarSrc,123456],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'创建用户失败'});

} else{

res.cookie('user\_id',result.insertId);

res.json({success\_code:200})

}

})

}

}

})

} else{

res.json({error\_code:1,message:'验证码不正确'})

}

});

//密码登录

router.post('/api/pwdLogin',function(req,res){

let name = req.body.userName;

let pwd = req.body.password;

let captcha = req.body.captcha;

//判断验证码是否正确

if (captcha.toLowerCase()!==req.session.captcha){

res.json({error\_code:1,message:'验证码不正确'})

} else{

delete req.session.captcha;

let sqlStr = 'SELECT \* from t\_user WHERE user\_name =? LIMIT 1 ;'

conn.query(sqlStr,[name],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'查询用户失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

if (result[0].password===pwd){

//保存用户id

req.session.userId = result[0].user\_id;

res.cookie('user\_id',result[0].user\_id);

res.json({success\_code:200})

} else{

res.json({error\_code:1,message:'密码错误'});

}

} else{

res.json({error:1,message:'用户不存在'});

}

}

})

}

});

//获取用户信息

router.get('/api/getUserInfo',function(req,res){

let userId = req.query.userId;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'获取用户信息失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result[0]})

} else{

res.json({error\_code:1,message:'用户信息不存在'});

}

}

})

}

});

//更新用户头像

router.post('/api/updateUserAvatar',function(req,res){

let {

userId,

avatar,

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET avatar = ? WHERE user\_id = ?;';

conn.query(sqlStr,[avatar,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户头像失败'});

} else{

res.json({success\_code:200});

}

});

}

})

}

});

//更新用户名

router.post('/api/updateUserName',function(req,res){

let {

userId,

userName,

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

sqlStr = 'SELECT \* FROM t\_user WHERE user\_name = ? AND user\_id <> ? LIMIT 1 ;';

conn.query(sqlStr,[userName,userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'用户名已存在！'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET user\_name = ? WHERE user\_id = ?;';

conn.query(sqlStr,[userName,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户名失败'});

} else{

res.json({success\_code:200});

}

});

}

}

});

}

})

}

});

//更新用户性别

router.post('/api/updateUserSex',function(req,res){

let {

userId,

sex,

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET sex = ? WHERE user\_id = ?;';

conn.query(sqlStr,[sex,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户性别失败'});

} else{

res.json({success\_code:200});

}

});

}

})

}

});

//更新用户生日

router.post('/api/updateUserBirthday',function(req,res){

let {

userId,

birthday,

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET birthday = ? WHERE user\_id = ?;';

conn.query(sqlStr,[birthday,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户生日失败'});

} else{

res.json({success\_code:200});

}

});

}

})

}

});

//更新用户签名

router.post('/api/updateUserSign',function(req,res){

let {

userId,

sign,

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET sign = ? WHERE user\_id = ?;';

conn.query(sqlStr,[sign,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户签名失败'});

} else{

res.json({success\_code:200});

}

});

}

})

}

});

//更新用户信息

router.post('/api/updateUserInfo',function(req,res){

let {

userId,

userName,

avatar,

password,

sex,

sign,

birthday

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

sqlStr = 'SELECT \* FROM t\_user WHERE user\_name = ? AND user\_id <> ? LIMIT 1 ;';

conn.query(sqlStr,[userName,userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'用户名已存在！'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET user\_name = ?,avatar = ?,password = ?,sex = ? ,birthday = ?,sign = ?WHERE user\_id = ?;';

conn.query(sqlStr,[userName,avatar,password,sex,birthday,sign,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户信息失败'});

} else{

res.json({success\_code:200})

}

});

}

}

});

}

})

}

});

//加载电影列表

router.get('/api/getMovieList',function (req,res) {

let sqlStr = 'SELECT \* FROM t\_schedule INNER JOIN t\_movie ON t\_schedule.movie\_id = t\_movie.movie\_id;';

conn.query(sqlStr,(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'获取电影列表失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result.length){

result = result.filter((value)=>{

return new Date(value.show\_date+','+value.show\_time)-new Date()>0;

});

for(let i = 0; i < result.length; i++) {

for(let j = i+1; j < result.length; j++) {

if(result[i]['movie\_id'] === result[j]['movie\_id']){

result.splice(j, 1); j = j-1;

}

}

}

res.json({success\_code:200,data:result})

} else{

res.json({error\_code:1,message:'电影列表为空'})

}

}

});

});

//加载电影详细信息

router.get('/api/getMovieDetail',function (req,res) {

let movieId = req.query.movieId;

let sqlStr = 'SELECT \* FROM t\_movie WHERE movie\_id = ? LIMIT 1;';

conn.query(sqlStr,[movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'获取电影信息失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result})

} else{

res.json({error\_code:1,message:'该电影不存在'})

}

}

});

});

//是否想看电影

router.post('/api/isWishMovie',function (req,res) {

let {

userId,

movieId

} = req.body;

let sqlStr = 'SELECT \* FROM t\_wishmovie WHERE user\_id = ? AND movie\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId,movieId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200});

} else {

res.json({error\_code:1,message:'不想看'});

}

}

})

});

//想看电影

router.post('/api/wishMovie',function (req,res) {

let {

userId,

movieId

} = req.body;

let sqlStr = 'INSERT INTO t\_wishmovie(user\_id,movie\_id) VALUES(?,?)';

conn.query(sqlStr,[userId,movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'操作失败'});

} else{

let sqlStr = 'SELECT wish\_num from t\_movie WHERE movie\_id = ? LIMIT 1;';

conn.query(sqlStr,[movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

//更新数据库

let sqlStr = 'UPDATE t\_movie SET wish\_num = ? WHERE movie\_id = ?;';

conn.query(sqlStr,[result[0].wish\_num+1,movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新信息失败'});

} else{

res.json({success\_code:200})

}

})

}

}

})

}

})

});

//取消想看电影

router.post('/api/cancelWishMovie',function (req,res) {

let {

userId,

movieId

} = req.body;

let sqlStr = 'DELETE FROM t\_wishmovie WHERE user\_id = ? AND movie\_id =? ;';

conn.query(sqlStr,[userId,movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'操作失败'});

} else{

let sqlStr = 'SELECT wish\_num from t\_movie WHERE movie\_id = ? LIMIT 1;';

conn.query(sqlStr,[movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

//更新数据库

let sqlStr = 'UPDATE t\_movie SET wish\_num = ? WHERE movie\_id = ?;';

conn.query(sqlStr,[result[0].wish\_num-1,movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新信息失败'});

} else{

res.json({success\_code:200})

}

})

}

}

})

}

})

});

//获取当前用户评论

router.get('/api/getUserComment',function (req,res) {

let {

userId,

movieId

} = req.query;

let sqlStr = 'SELECT \* FROM t\_comment WHERE user\_id = ? AND movie\_id= ? LIMIT 1;';

conn.query(sqlStr,[userId,movieId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result[0]});

} else{

res.json({error\_code:1,message:'用户未评论'});

}

}

})

});

//获取所有用户通过审核的评论

router.get('/api/getAllUserPassComment',function (req,res) {

let {

movieId

} = req.query;

let sqlStr = 'SELECT \* FROM t\_user user INNER JOIN t\_comment comment ON user.user\_id = comment.user\_id WHERE comment.movie\_id = ? AND comment.is\_pass = ? ;';

conn.query(sqlStr,[movieId,1],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result){

res.json({success\_code:200,data:result});

} else{

res.json({error\_code:1,message:'未评论'});

}

}

})

});

//更新用户评论

router.post('/api/updateUserComment',function (req,res) {

let{

userId,

movieId,

score,

commentContent,

commentDate

} = req.body;

let sqlStr = 'SELECT comment\_id from t\_comment WHERE user\_id = ? AND movie\_id = ? LIMIT 1';

conn.query(sqlStr,[userId,movieId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

//评论存在

//更新

let sqlStr = 'UPDATE t\_comment SET user\_score = ?, comment\_content = ?, comment\_date = ?,support\_num = ?,is\_pass = ?,support\_user = ? WHERE comment\_id = ? ;';

conn.query(sqlStr,[score,commentContent,commentDate,0,0,undefined,result[0].comment\_id],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新评论失败'});

} else{

res.json({success\_code:200})

}

})

} else{

//评论不存在

let sqlStr = 'INSERT INTO t\_comment(user\_id,movie\_id,user\_score,comment\_content,comment\_date,support\_num,is\_pass) VALUES(?,?,?,?,?,?,?)';

conn.query(sqlStr,[userId,movieId,score,commentContent,commentDate,0,0],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'操作失败'});

} else{

res.json({success\_code:200})

}

})

}

}

})

});

//获取当前评论

router.get('/api/getCommentByID',function (req,res) {

let {

commentId

} = req.query;

let sqlStr = 'SELECT \* FROM t\_comment WHERE comment\_id = ? LIMIT 1;';

conn.query(sqlStr,[commentId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'操作失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result[0]});

} else{

res.json({error\_code:1,message:'用户未评论'});

}

}

})

});

//更新当前评论的用户点赞

router.post('/api/updateUserSupport',function (req,res) {

let {

commentId,

supportNum,

supportUser

} = req.body;

let sqlStr = 'UPDATE t\_comment SET support\_num = ? , support\_user = ? WHERE comment\_id = ?;';

conn.query(sqlStr,[supportNum,supportUser,commentId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'操作失败'});

} else{

res.json({success\_code:200});

}

})

});

//加载影院列表

router.get('/api/getCinemaList',function (req,res) {

let sqlStr = 'SELECT \* FROM t\_schedule INNER JOIN t\_cinema ON t\_schedule.cinema\_id = t\_cinema.cinema\_id INNER JOIN t\_movie ON t\_schedule.movie\_id = t\_movie.movie\_id;';

conn.query(sqlStr,(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'获取影院列表失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result.length){

result = result.filter((value)=>{

return new Date(value.show\_date+','+value.show\_time)-new Date()>0;

});

for(let i = 0; i < result.length; i++) {

for(let j = i+1; j < result.length; j++) {

if(result[i]['cinema\_id'] === result[j]['cinema\_id']){

result.splice(j, 1); j = j-1;

}

}

}

}

res.json({success\_code:200,data:result})

}

});

});

//加载当前影院详细信息

router.get('/api/getCurrentCinemaDetail',function (req,res) {

let cinemaId = req.query.cinemaId;

let sqlStr = 'SELECT \* FROM t\_cinema WHERE cinema\_id = ? LIMIT 1;';

conn.query(sqlStr,[cinemaId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'获取当前影院信息失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result[0]})

} else{

res.json({error\_code:1,message:'影院不存在'})

}

}

});

});

//加载当前影院排片

router.get('/api/getCurrentCinemaMovieSchedule',function (req,res) {

let cinemaId = req.query.cinemaId;

console.log(cinemaId);

let sqlStr = 'SELECT \* FROM t\_schedule WHERE cinema\_id = ?;';

conn.query(sqlStr,[cinemaId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'获取当前影院排片信息失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result){

let tempMovieArr = [];

result.forEach((value)=>{

if (new Date()-new Date(value.show\_date+','+value.show\_time)<=0){

tempMovieArr.push(value.movie\_id);

}

});

tempMovieArr = Array.from(new Set(tempMovieArr));

let movieArray = [];

let movieScheduleArray = [];

tempMovieArr.forEach((value)=>{

sqlStr = 'SELECT \* FROM t\_movie WHERE movie\_id = ? LIMIT 1;';

conn.query(sqlStr,[value],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

movieArray.push(result[0]);

}

}

});

sqlStr = 'SELECT \* FROM t\_schedule schedule INNER JOIN t\_movie movie ON schedule.movie\_id = movie.movie\_id WHERE schedule.movie\_id = ? AND schedule.cinema\_id = ?;';

conn.query(sqlStr,[value,cinemaId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result){

movieScheduleArray.push(result);

}

}

})

});

setTimeout(()=>{

res.json({

success\_code:200,

data:{

hasMovieInfo:movieArray,

movieScheduleInfo:movieScheduleArray,

}

});

},500);

} else{

res.json({error\_code:1,message:'当前影院排片信息为空'})

}

}

});

});

//加载当前影院详细信息

router.get('/api/getScheduleById',function (req,res) {

let scheduleId = req.query.scheduleId;

let sqlStr = 'SELECT \* FROM t\_schedule WHERE schedule\_id = ? LIMIT 1;';

conn.query(sqlStr,[scheduleId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'获取当前排片信息失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result[0]})

} else{

res.json({error\_code:1,message:'排片信息不存在'})

}

}

});

});

//加载当前影院详细信息

router.post('/api/updateScheduleSeat',function (req,res) {

let {

scheduleId,

seatInfo

} = req.body;

let sqlStr = 'UPDATE t\_schedule SET seat\_info = ? WHERE schedule\_id = ? LIMIT 1;';

conn.query(sqlStr,[seatInfo,scheduleId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'更新排片座位信息失败'})

} else{

res.json({success\_code:200,data:result[0]})

}

});

});

//加载当前电影排片

router.get('/api/getCurrentMovieSchedule',function (req,res) {

let movieId = req.query.movieId;

let sqlStr = 'SELECT \* FROM t\_schedule WHERE movie\_id = ?;';

conn.query(sqlStr,[movieId],(error,result,field)=>{

if (error){

console.log(error);

res.json({error\_code:1,message:'获取当前影院排片信息失败'})

} else{

result = JSON.parse(JSON.stringify(result));

if (result){

let tempDateArr = [];

result.forEach((value)=>{

if (new Date()-new Date(value.show\_date+','+value.show\_time)<=0){

tempDateArr.push(value.show\_date);

}

});

tempDateArr = Array.from(new Set(tempDateArr));

tempDateArr.sort((a,b)=>{

return new Date(a)-new Date(b);

});

let cinemaArray = [];

let cinemaScheduleArray = [];

tempDateArr.forEach((value)=>{

sqlStr = 'SELECT \* FROM t\_schedule WHERE show\_date = ?';

conn.query(sqlStr,[value],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result){

cinemaArray.push(result);

}

}

});

sqlStr = 'SELECT \* FROM t\_schedule schedule INNER JOIN t\_cinema cinema ON schedule.cinema\_id = cinema.cinema\_id WHERE schedule.movie\_id = ? AND schedule.show\_date = ?;';

conn.query(sqlStr,[movieId,value],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result){

cinemaScheduleArray.push(result);

}

}

});

});

setTimeout(()=>{

res.json({

success\_code:200,

data:{

hasCinemaInfo:cinemaArray,

cinemaScheduleInfo:cinemaScheduleArray,

}

});

},500);

} else{

res.json({error\_code:1,message:'当前电影排片信息为空'})

}

}

});

});

//根据名字匹配电影

router.get('/api/matchMovieByName',function (req,res) {

let movieName = req.query.movieName;

let sqlStr = 'SELECT \* FROM t\_schedule INNER JOIN t\_movie ON t\_schedule.movie\_id = t\_movie.movie\_id WHERE name LIKE ?;';

conn.query(sqlStr,['%'+movieName+'%'],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result.length){

result = result.filter((value)=>{

return new Date(value.show\_date+','+value.show\_time)-new Date()>0;

});

for(let i = 0; i < result.length; i++) {

for(let j = i+1; j < result.length; j++) {

if(result[i]['movie\_id'] === result[j]['movie\_id']){

result.splice(j, 1); j = j-1;

}

}

}

}

res.json({success\_code:200,data:result});

}

});

});

//根据名字匹配影院

router.get('/api/matchCinemaByName',function (req,res) {

let cinemaName = req.query.cinemaName;

let sqlStr = 'SELECT \* FROM t\_schedule INNER JOIN t\_cinema ON t\_schedule.cinema\_id = t\_cinema.cinema\_id INNER JOIN t\_movie ON t\_schedule.movie\_id = t\_movie.movie\_id WHERE cinema\_name LIKE ?;';

conn.query(sqlStr,['%'+cinemaName+'%'],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result.length){

result = result.filter((value)=>{

return new Date(value.show\_date+','+value.show\_time)-new Date()>0;

});

for(let i = 0; i < result.length; i++) {

for(let j = i+1; j < result.length; j++) {

if(result[i]['cinema\_id'] === result[j]['cinema\_id']){

result.splice(j, 1); j = j-1;

}

}

}

}

res.json({success\_code:200,data:result});

}

});

});

//用户下单

router.post('/api/order',function (req,res) {

let {

userId,

scheduleId,

orderPhone,

orderDate,

ticketNum,

totalPrice,

orderSeatInfo,

payType

} = req.body;

let phoneCode = util.randomCode(6);

let sqlStr = 'INSERT INTO t\_order(user\_id,schedule\_id,order\_phone,order\_date,ticket\_num,ticket\_total\_price,order\_seat\_info,pay\_type,phone\_code) VALUES(?,?,?,?,?,?,?,?,?); ';

conn.query(sqlStr,[userId,scheduleId,orderPhone,orderDate,ticketNum,totalPrice,orderSeatInfo,payType,phoneCode],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200,data:{phone\_code:phoneCode}});

}

});

});

//获取个人订单信息

router.get('/api/getOrderByUserId',function (req,res) {

let userId = req.query.userId;

let sqlStr = 'SELECT \* FROM t\_order INNER JOIN t\_schedule ON t\_order.schedule\_id = t\_schedule.schedule\_id INNER JOIN t\_movie ON t\_movie.movie\_id = t\_schedule.movie\_id INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_schedule.cinema\_id WHERE user\_id = ?;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result});

}

});

});

//获取个人想看电影

router.get('/api/getWishMovieByUserId',function(req,res){

let userId = req.query.userId;

let sqlStr = 'SELECT \* FROM t\_wishmovie INNER JOIN t\_movie ON t\_wishmovie.movie\_id = t\_movie.movie\_id WHERE user\_id = ?;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result});

}

})

});

//获取个人看过的电影

router.get('/api/getIsWatchedMovieByUserId',function(req,res){

let userId = req.query.userId;

let sqlStr = 'SELECT \* FROM t\_comment INNER JOIN t\_movie ON t\_comment.movie\_id = t\_movie.movie\_id WHERE user\_id = ? AND is\_pass = 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result});

}

})

});

//管理员API

//获取当前页用户

//密码登录

router.post('/api/admin/login',function(req,res){

let name = req.body.name;

let password = req.body.password;

let sqlStr = 'SELECT \* FROM t\_admin WHERE name = ? LIMIT 1 ;';

conn.query(sqlStr,[name,password],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'查询用户失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

if (result[0].password===password){

//保存用户id

req.session.adminId = result[0].admin\_id;

res.cookie('admin\_id',result[0].admin\_id);

res.json({success\_code:200})

} else{

res.json({error\_code:1,message:'密码错误'});

}

} else{

res.json({error:1,message:'用户不存在'});

}

}

})

});

//获取用户信息

router.get('/api/admin/getAdminInfo',function(req,res){

let adminId = req.query.adminId;

let sqlStr = 'SELECT \* FROM t\_admin WHERE admin\_id = ? LIMIT 1 ;';

conn.query(sqlStr,[adminId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'查询用户失败'});

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({success\_code:200,data:result[0]});

} else{

res.json({error:1,message:'用户不存在'});

}

}

})

});

router.get('/api/admin/getCurrentPageUser',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_user WHERE user\_name LIKE ? ORDER BY user\_id;';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

})

sqlStr = 'SELECT \* FROM t\_user WHERE user\_name LIKE ? ORDER BY user\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

var datatime = './public/images/avatar/';

//将图片放到服务器

var storage = multer.diskStorage({

// 如果你提供的 destination 是一个函数，你需要负责创建文件夹

destination: datatime,

// //给上传文件重命名，获取添加后缀名

filename: function (req, file, cb) {

cb(null, new Date().getTime()+'.jpg');

}

});

var upload = multer({

storage: storage

});

// let upload = multer({dest:'./public/images/avatar'}).any();

router.post('/api/admin/upLoadImg',upload.any(),function (req,res) {

res.json({success\_code:200,data:req.files});

console.log(req.files);

});

//更新用户信息

router.post('/api/admin/updateUserInfo',function(req,res){

let {

userId,

userName,

avatar,

password,

sex,

phone,

sign,

birthday

} = req.body;

if (userId){

let sqlStr = 'SELECT \* from t\_user WHERE user\_id = ? LIMIT 1;';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'用户不存在'});

} else{

let sqlStr = 'SELECT \* FROM t\_user WHERE user\_name = ? AND user\_id <> ? LIMIT 1;';

conn.query(sqlStr,[userName,userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'用户名已存在！'});

} else{

sqlStr = 'SELECT \* FROM t\_user WHERE phone = ? AND user\_id <> ? LIMIT 1;';

conn.query(sqlStr,[phone,userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'手机号码已注册！'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_user SET user\_name = ?,avatar = ?,password = ?,sex = ? ,phone = ?,birthday = ?,sign = ? WHERE user\_id = ?;';

conn.query(sqlStr,[userName,avatar,password,sex,phone,birthday,sign,userId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新用户信息失败'});

console.log(error);

} else{

res.json({success\_code:200})

}

});

}

}

})

}

}

});

}

})

}

});

//删除用户信息

router.post('/api/admin/deleteUserInfo',function(req,res){

let {

userId

} = req.body;

if (userId){

let sqlStr = 'SELECT t\_schedule.schedule\_id, t\_schedule.seat\_info,order\_seat\_info FROM t\_order INNER JOIN t\_schedule ON t\_order.schedule\_id = t\_schedule.schedule\_id WHERE user\_id = ?';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

console.log(result);

if (result){

result.forEach((value)=>{

let tempArr = [];

value.seat\_info = JSON.parse(value.seat\_info);

value.order\_seat\_info = JSON.parse(value.order\_seat\_info);

value.seat\_info.forEach((v)=>{

if(value.order\_seat\_info.indexOf(v)===-1){

tempArr.push(v);

}

});

tempArr = JSON.stringify(tempArr);

sqlStr = 'UPDATE t\_schedule SET seat\_info = ? WHERE schedule\_id = ?;';

conn.query(sqlStr,[tempArr,value.schedule\_id],(error,result,field)=>{

if (error){

console.log(error);

}

})

})

}

sqlStr = 'DELETE FROM t\_user WHERE user\_id =?';

conn.query(sqlStr,[userId],(error,result,field)=>{

if (error){

console.log(error);

} else {

res.json({success\_code:200});

}

})

}

})

}

});

//添加用户信息

router.post('/api/admin/addUserInfo',function(req,res){

let {

userName,

avatar,

phone,

password,

sex,

sign,

birthday

} = req.body;

if (!avatar){

avatar = '/images/avatar/monkey.png'

}

let sqlStr = 'SELECT \* FROM t\_user WHERE user\_name = ? LIMIT 1;';

conn.query(sqlStr,[userName],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'用户名已存在！'});

} else{

sqlStr = 'SELECT \* FROM t\_user WHERE phone = ? LIMIT 1';

conn.query(sqlStr,[phone],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'手机号码已注册！'});

} else{

sqlStr = 'INSERT INTO t\_user(user\_name,avatar,phone,password,sex,sign,birthday) VALUES(?,?,?,?,?,?,?);';

conn.query(sqlStr,[userName, avatar,phone, password, sex, sign, birthday],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

})

}

}

});

}

}

});

});

//获取当前页电影

router.get('/api/admin/getCurrentPageMovie',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_movie WHERE name LIKE ? ORDER BY movie\_id';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

});

sqlStr = 'SELECT \* FROM t\_movie WHERE name LIKE ? ORDER BY movie\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

//更新电影信息

router.post('/api/admin/updateMovieInfo',function(req,res){

let {

movieId,

movieName,

poster,

director,

actor,

long,

type,

language,

publicDate,

intro

} = req.body;

let sqlStr = 'SELECT \* FROM t\_movie WHERE name = ? AND movie\_id <> ? LIMIT 1;';

conn.query(sqlStr,[movieName,movieId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'电影名已存在！'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_movie SET name = ?,poster = ?,director = ?,actor = ? ,movie\_long = ?,type = ?,language = ?,public\_date = ?,intro = ? WHERE movie\_id = ?;';

conn.query(sqlStr,[movieName,poster,director,actor,long,type,language,publicDate,intro,movieId],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200})

}

})

}

}

});

});

//添加电影信息

router.post('/api/admin/addMovieInfo',function(req,res){

let {

movieName,

poster,

director,

actor,

long,

type,

language,

publicDate,

intro

} = req.body;

let sqlStr = 'SELECT \* FROM t\_movie WHERE name = ? LIMIT 1;';

conn.query(sqlStr,[movieName],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'电影名已存在！'});

} else{

let sqlStr = 'INSERT INTO t\_movie(name,poster,director,actor,movie\_long,type,language,public\_date,intro) VALUES(?,?,?,?,?,?,?,?,?);';

conn.query(sqlStr,[movieName,poster,director,actor,long,type,language,publicDate,intro],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

})

}

}

});

});

datatime = './public/images/movie/';

//将图片放到服务器

storage = multer.diskStorage({

// 如果你提供的 destination 是一个函数，你需要负责创建文件夹

destination: datatime,

// //给上传文件重命名，获取添加后缀名

filename: function (req, file, cb) {

cb(null, new Date().getTime()+'.jpg');

}

});

upload = multer({

storage: storage

});

router.post('/api/admin/upLoadMovieImg',upload.any(),function (req,res) {

res.json({success\_code:200,data:req.files});

console.log(req.files);

});

//删除电影信息

router.post('/api/admin/deleteMovieInfo',function(req,res){

let {

movieId

} = req.body;

let sqlStr = 'DELETE FROM t\_movie WHERE movie\_id =?';

conn.query(sqlStr,[movieId],(error,result,field)=>{

if (error){

console.log(error);

} else {

res.json({success\_code:200});

}

})

});

//获取当前页影院

router.get('/api/admin/getCurrentPageCinema',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_cinema WHERE cinema\_name LIKE ? ORDER BY cinema\_id ;';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

});

sqlStr = 'SELECT \* FROM t\_cinema WHERE cinema\_name LIKE ? ORDER BY cinema\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

//更新影院信息

router.post('/api/admin/updateCinemaInfo',function(req,res){

let {

cinemaId,

cinemaName,

cinemaPhone,

address

} = req.body;

if (cinemaId){

let sqlStr = 'SELECT \* from t\_cinema WHERE cinema\_id = ? LIMIT 1;';

conn.query(sqlStr,[cinemaId],(error,result,field)=>{

if (error){

console.log(error);

} else{

sqlStr = 'SELECT \* FROM t\_cinema WHERE cinema\_name = ? AND cinema\_id <> ? LIMIT 1 ;';

conn.query(sqlStr,[cinemaName,cinemaId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'影院名已存在！'});

} else{

//更新数据库

let sqlStr = 'UPDATE t\_cinema SET cinema\_name = ?,cinema\_phone = ?,specified\_address = ? WHERE cinema\_id = ?;';

conn.query(sqlStr,[cinemaName,cinemaPhone,address,cinemaId],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新影院信息失败'});

console.log(error);

} else{

res.json({success\_code:200})

}

})

}

}

});

}

})

}

});

//删除影院信息

router.post('/api/admin/deleteCinemaInfo',function(req,res){

let {

cinemaId

} = req.body;

if (cinemaId){

let sqlStr = 'DELETE FROM t\_cinema WHERE cinema\_id =?';

conn.query(sqlStr,[cinemaId],(error,result,field)=>{

if (error){

console.log(error);

} else {

res.json({success\_code:200});

}

})

}

});

//添加影院信息

router.post('/api/admin/addCinemaInfo',function(req,res){

let {

cinemaName,

cinemaPhone,

address

} = req.body;

sqlStr = 'SELECT \* FROM t\_cinema WHERE cinema\_name = ? LIMIT 1 ;';

conn.query(sqlStr,[cinemaName],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'影院名已存在！'});

} else{

let sqlStr = 'INSERT INTO t\_cinema(cinema\_name,cinema\_phone,specified\_address) VALUES(?,?,?);';

conn.query(sqlStr,[cinemaName,cinemaPhone,address],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

})

}

}

});

});

//获取当前页评论

router.get('/api/admin/getCurrentPageComment',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_comment INNER JOIN t\_movie ON t\_comment.movie\_id = t\_movie.movie\_id INNER JOIN t\_user ON t\_user.user\_id=t\_comment.user\_id WHERE t\_movie.name LIKE ? ORDER BY comment\_id;';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

});

sqlStr = 'SELECT \* FROM t\_comment INNER JOIN t\_movie ON t\_comment.movie\_id = t\_movie.movie\_id INNER JOIN t\_user ON t\_user.user\_id=t\_comment.user\_id WHERE t\_movie.name LIKE ? ORDER BY comment\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

//通过当前评论

router.post('/api/admin/passCurrentComment',function (req,res) {

let commentId = req.body.commentId;

let movieId = req.body.movieId;

if (commentId){

let sqlStr = 'UPDATE t\_comment SET is\_pass = 1 WHERE comment\_id = ?;';

conn.query(sqlStr,[commentId],(error,result,field)=>{

if (error){

console.log(error);

} else{

sqlStr = 'SELECT user\_score FROM t\_comment WHERE movie\_id = ? AND is\_pass = ?;';

conn.query(sqlStr,[movieId,1],(error,result,field)=>{

if (error){

console.log(error);

} else {

result = JSON.parse(JSON.stringify(result));

let avgScore = 0;

if (result.length){

result.forEach((val)=>{

avgScore += (Number(val.user\_score));

});

avgScore = (avgScore/Number(result.length)).toFixed(1);

}

sqlStr = 'UPDATE t\_movie SET score = ? WHERE movie\_id = ?;';

conn.query(sqlStr,[avgScore,movieId],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

})

}

});

}

})

}

});

//删除当前评论

router.post('/api/admin/deleteCurrentComment',function (req,res) {

let commentId = req.body.commentId;

let movieId = req.body.movieId;

if (commentId){

let sqlStr = 'DELETE FROM t\_comment WHERE comment\_id = ?;';

conn.query(sqlStr,[commentId],(error,result,field)=>{

if (error){

console.log(error);

} else{

sqlStr = 'SELECT user\_score FROM t\_comment WHERE movie\_id = ? AND is\_pass = ?;';

conn.query(sqlStr,[movieId,1],(error,result,field)=>{

if (error){

console.log(error);

} else {

result = JSON.parse(JSON.stringify(result));

let avgScore = 0;

if (result.length){

result.forEach((val)=>{

avgScore += (Number(val.user\_score));

});

avgScore = (avgScore/Number(result.length)).toFixed(1);

}

sqlStr = 'UPDATE t\_movie SET score = ? WHERE movie\_id = ?;';

conn.query(sqlStr,[avgScore,movieId],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

})

}

});

}

})

}

});

//获取当前页订单

router.get('/api/admin/getCurrentPageOrder',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_order INNER JOIN t\_schedule ON t\_order.schedule\_id = t\_schedule.schedule\_id INNER JOIN t\_movie ON t\_movie.movie\_id = t\_schedule.movie\_id INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_schedule.cinema\_id INNER JOIN t\_user ON t\_order.user\_id = t\_user.user\_id WHERE t\_movie.name LIKE ? ORDER BY order\_id;';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

});

sqlStr = 'SELECT \* FROM t\_order INNER JOIN t\_schedule ON t\_order.schedule\_id = t\_schedule.schedule\_id INNER JOIN t\_movie ON t\_movie.movie\_id = t\_schedule.movie\_id INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_schedule.cinema\_id INNER JOIN t\_user ON t\_order.user\_id = t\_user.user\_id WHERE t\_movie.name LIKE ? ORDER BY order\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

//删除订单

router.post('/api/admin/deleteOrder',function(req,res){

let {

orderId,

scheduleId,

orderSeatInfo

} = req.body;

if (orderId){

let sqlStr = 'SELECT seat\_info FROM t\_schedule WHERE schedule\_id = ?';

conn.query(sqlStr,[scheduleId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = result[0].seat\_info;

result = JSON.parse(result);

orderSeatInfo = JSON.parse(orderSeatInfo);

console.log(typeof result);

console.log(typeof orderSeatInfo);

let tempArr = [];

result.forEach((value)=>{

if (orderSeatInfo.indexOf(value)===-1){

tempArr.push(value);

}

});

console.log(tempArr);

result = JSON.stringify(tempArr);

sqlStr = 'UPDATE t\_schedule SET seat\_info = ? WHERE schedule\_id = ?;';

conn.query(sqlStr,[result,scheduleId],(error,result,field)=>{

if (error){

console.log(error);

} else{

sqlStr = 'DELETE FROM t\_order WHERE order\_id =?';

conn.query(sqlStr,[orderId],(error,result,field)=>{

if (error){

console.log(error);

} else {

res.json({success\_code:200});

}

})

}

})

}

})

}

});

//获取当前页影厅

router.get('/api/admin/getCurrentPageHall',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_hall INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_hall.cinema\_id WHERE t\_cinema.cinema\_name LIKE ? ORDER BY hall\_id;';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

});

sqlStr = 'SELECT \* FROM t\_hall INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_hall.cinema\_id WHERE t\_cinema.cinema\_name LIKE ? ORDER BY hall\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

//删除影厅

router.post('/api/admin/deleteHall',function(req,res){

let {

cinemaId,

hallName

} = req.body;

if (cinemaId){

let sqlStr = 'SELECT schedule\_id FROM t\_schedule WHERE cinema\_id = ? AND hall\_name = ?;';

conn.query(sqlStr,[cinemaId,hallName],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

console.log(result);

if (result.length){

result.forEach((value)=>{

sqlStr = 'DELETE FROM t\_schedule WHERE schedule\_id = ?;';

conn.query(sqlStr,[value.schedule\_id],(error,result,field)=>{

if (error){

console.log(error);

}

})

})

}

sqlStr = 'DELETE FROM t\_hall WHERE cinema\_id = ? AND name = ?';

conn.query(sqlStr,[cinemaId,hallName],(error,result,field)=>{

if (error){

console.log(error);

} else {

res.json({success\_code:200});

}

})

}

});

}

});

//更新影厅信息

router.post('/api/admin/updateHallInfo',function(req,res){

let {

hallId,

cinemaId,

hallOldName,

hallNewName

} = req.body;

if (cinemaId){

let sqlStr = 'SELECT \* FROM t\_hall WHERE name = ? AND cinema\_id = ? AND hall\_id <> ? LIMIT 1;';

conn.query(sqlStr,[hallNewName,cinemaId,hallId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'该影院的影厅已存在！'});

} else{

sqlStr = 'UPDATE t\_schedule SET hall\_name = ? WHERE cinema\_id = ? AND hall\_name = ?';

conn.query(sqlStr,[hallNewName,cinemaId,hallOldName],(error,result,field)=>{

if (error){

console.log(error);

} else{

//更新数据库

let sqlStr = 'UPDATE t\_hall SET name = ? WHERE cinema\_id = ? AND name = ?;';

conn.query(sqlStr,[hallNewName,cinemaId,hallOldName],(error,result,field)=>{

if (error){

res.json({error\_code:1,message:'更新影影厅信息失败'});

console.log(error);

} else{

res.json({success\_code:200})

}

})

}

});

}

}

});

}

});

//获取所有影院

router.get('/api/admin/getAllCinema',function(req,res){

let sqlStr = 'SELECT cinema\_id,cinema\_name FROM t\_cinema;';

conn.query(sqlStr,(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result});

}

})

});

//添加影厅信息

router.post('/api/admin/addHallInfo',function(req,res){

let {

cinemaId,

hallName

} = req.body;

let sqlStr = 'SELECT \* FROM t\_hall WHERE name = ? AND cinema\_id = ? LIMIT 1;';

conn.query(sqlStr,[hallName,cinemaId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

if (result[0]){

res.json({error\_code:1,message:'该影院的影厅已存在！'});

} else{

let sqlStr = 'INSERT INTO t\_hall(cinema\_id,name) VALUES(?,?);';

conn.query(sqlStr,[cinemaId,hallName],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

});

}

}

});

});

//获取当前页电影

router.get('/api/admin/getCurrentPageMovieSchedule',function(req,res){

let {

currentPage, pageSize,input

} = req.query;

let start = Number((currentPage-1)\*pageSize);

pageSize = Number(pageSize);

let sqlStr = 'SELECT \* FROM t\_schedule INNER JOIN t\_movie ON t\_schedule.movie\_id = t\_movie.movie\_id INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_schedule.cinema\_id WHERE t\_movie.name LIKE ? ORDER BY schedule\_id ';

let total;

conn.query(sqlStr,["%"+input+"%"],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

total = result.length;

}

});

sqlStr = 'SELECT \* FROM t\_schedule INNER JOIN t\_movie ON t\_schedule.movie\_id = t\_movie.movie\_id INNER JOIN t\_cinema ON t\_cinema.cinema\_id = t\_schedule.cinema\_id WHERE t\_movie.name LIKE ? ORDER BY schedule\_id LIMIT ?,?;';

conn.query(sqlStr,["%"+input+"%",start,pageSize],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result,total:total});

}

})

});

//获取所有电影

router.get('/api/admin/getAllMovie',function(req,res){

let sqlStr = 'SELECT \* FROM t\_movie;';

conn.query(sqlStr,(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result});

}

})

});

//获取影院的影厅

router.get('/api/admin/getHallByCinemaId',function(req,res){

let cinemaId = req.query.cinemaId;

console.log(cinemaId);

let sqlStr = 'SELECT hall\_id,name FROM t\_hall WHERE cinema\_id = ?;';

conn.query(sqlStr,[cinemaId],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

console.log(result);

res.json({success\_code:200,data:result});

}

})

});

//获取排片的某天的时间段安排

router.get('/api/admin/getHasScheduleDateTime',function(req,res){

let {

cinemaId,hallName,showDate

} = req.query;

let sqlStr = 'SELECT show\_time FROM t\_schedule WHERE cinema\_id = ? AND hall\_name = ? AND show\_date = ?;';

conn.query(sqlStr,[cinemaId,hallName,showDate],(error,result,field)=>{

if (error){

console.log(error);

} else{

result = JSON.parse(JSON.stringify(result));

res.json({success\_code:200,data:result});

}

})

});

//添加影院信息

router.post('/api/admin/addScheduleInfo',function(req,res){

let {

movieId,cinemaId,hallName,showDate,showTime,price

} = req.body;

let sqlStr = 'INSERT INTO t\_schedule(movie\_id,cinema\_id,hall\_name,show\_date,show\_time,price) VALUES(?,?,?,?,?,?);';

conn.query(sqlStr,[movieId,cinemaId,hallName,showDate,showTime,price],(error,result,field)=>{

if (error){

console.log(error);

} else{

res.json({success\_code:200});

}

})

});

//删除影厅

router.post('/api/admin/deleteMovieSchedule',function(req,res){

let {

scheduleId

} = req.body;

if (scheduleId){

let sqlStr = 'DELETE FROM t\_schedule WHERE schedule\_id = ?';

conn.query(sqlStr,[scheduleId],(error,result,field)=>{

if (error){

console.log(error);

} else {

res.json({success\_code:200});

}

})

}

});

module.exports = router;

## 5.2功能

实现以下全部功能。

1. 注册登入功能：

在登入页面（注册页面）登入账号（注册账号）即可。有管理员和普通用户两种登入方式，管理员登入可以对电影排版，哪个电影可以上映，在哪里上映等做出修改。

1. 买票功能：

登入之后，在首页的正在热映页面可以看到如今正在热卖的电影，可能会快速找到自己喜爱的电影。

我们也可以点击首页的右方电影按钮，然后在里面可以查看如今所有在映的电影。我们也可以在影院页面查看是否有自己喜欢的电影，每个影院都有自己上映的电影的清单。

最后我们可以直接在首页上方的搜索界面搜索想要看的电影。

在上述所有操作中只要找到想要购买的电影票，都可以在当前页面购买。

1. 搜索功能：

在首页上方直接搜索，可以搜索自己想看的电影，或者自己心仪的电影院。

1. 预购功能：

在首页的即将上映页面又即将上映的电影，我们可以在里面进行预购。

1. 查看订单功能：

在右下方“我的”界面可有“我的订单”功能，在这里我们可以看到自己的所有历史订单，清楚自己的每一笔消费。

1. 想看的电影：

也是在右下方“我的”界面右边“想看的电影”可以在那里写入想看的电影。

1. 看过的电影：

同是“我的”界面，我们可以查看自己所有看过的电影，留下自己的点点滴滴。

1. 评论功能：

可以再电影下进行评论，谈谈自己对电影的感受等。

## 5.3性能

精确度较高，且及时响应程度高

## 5.4输入项

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 输入方式 | 输入媒介 | 输入来源 |
| Username | Username | Varchar | 用户输入 | 键盘 | TextBox1 |
| Password | Password | Varchar | 用户输入 | 键盘 | TextBox2 |
| Moviename | Moviename | Varchar | 参数传递 | 系统 | netSearchList.js |

## 5.5输出项

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 标识 | 数据类型 | 输出方式 | 输出来源 |
| Username | Username | Varchar | 写入数据 | TextBox1 |
| Password | Password | Varchar | 写入数据 | TextBox2 |
| Movie | Movie | Varchar | 写入数据 | netSearchList.js |

## 5.6算法

推荐算法简单说就是在海量的用户数据（行为记录等）中对用户进行划分，对同一群体的用户推荐其他用户喜欢的电影。

这其中需要给电影分类并建立评分细则、建立用户模型、寻找相似用户，基于用户的行为数据将电影分类匹配

将电影分成三个部分：系统推荐 、个人喜好推荐 、同好推荐 。

1. 从准确性、多样性角度分析

系统推荐电影

准确性低、多样性高

多样性高能为用户带来新鲜感，如果发现了一个从未听过但特别喜欢的影片，会带来惊喜感，调动用户正面情绪。

可是由于准确性低，很可能新影片很不被用户喜欢，所以在系统推荐电影在播放界面设置“删除”、“下一个”两个按键便于用户切换电影。

个人喜好推荐

准确性高、多样性低

准确性高使得每日推荐的20个电影比较好的满足用户口味，但是存在电影类型单一化的问题，因此设置了播放列表以提供用户浏览、操作的权利，弥补曲目单一化带给用户的失望。

同好推荐

准确性中、多样性中

同好推荐 有别于其他两个个性化推荐功能，它准确性多样性的阈值不只是由算法决定的，更多的是它功能形式所决定的。先把功能的面向对象分为两类，一类是用户，一类是UGC片单，系统分别为片单和用户加标签以提高准确度，由于UGC片单是由很多用户创建，所以UGC片单就具有多样性，两者糅合从而保证了准确度和多样性共存。

2. 从操作流程上分析

三个功能从看见功能按键到最终获得推荐曲目的步骤：

看见系统推荐电影>点击系统推荐电影>获取电影

看见个人喜好推荐 >点击个人喜好推荐 >看见推荐列表>筛选喜欢曲目>点击喜欢曲目>获取电影

看见同好推荐 >点击同好推荐 >跳转片单页面>发现类型标签>筛选类型标签>点击类型标签>看见标签下的同好推荐 >筛选片单>点击片单>浏览片单列表>筛选喜欢电影>点击喜欢电影>获取电影

可以发现三种方式获取推荐电影的操作流程由简入繁。

3. 从用户使用阶段分析

三个功能对应着三种用户阶段：

系统推荐电影-新用户：

系统推荐电影位于个页黄金位置，新用户初次体验的产品功能时大概率点击这个按键，所以要简化用户使用流程，用户在快速感受产品个性化推荐的魅力后才产生继续了解其他功能的欲望。

个人喜好推荐 -普通用户：

新用户使用系统推荐电影过后需要不一样的体验来满足个性化需求。

每日20个电影推荐对用户来说是可预知的，20个上限的设定给用户物以稀为贵的感觉，会珍惜每日的推荐，而每日更新无法回看以往推荐的设定，会让用户觉得一天不看就错过了什么的紧迫感。

推荐算法设定了基于不同用户行为的权重，“想看”最高，评论、搜索、分享其次

同好推荐 -深度用户：

片单是云电影连接个性化推荐和社交的重要桥梁，同好推荐 是个性化推荐功能最后一环。

在深度体验了同好推荐 之后，用户会得到片单可被分享和推荐的认识，很可能会产生自建片单的冲动。

而片单在云电影中具有社交属性，用户可以互相收藏、评论、分享片单，而且片单在个人主页中也反映了个人电影风格，让用户能够更好地展现自己给他人。

4. 从参与元素分析

系统推荐电影：系统

个人喜好推荐 ：系统+自己

同好推荐 ：系统+自己+其他用户

欧氏距离vs余弦相似度

在线下导购时代，导购员会通过系统的话术掌握消费者的情况，来推荐商品。

类比导购员推销时的思维逻辑，我们可以得到电影推荐算法需要解决的三个核心问题：

将用户信息转化为用户类型；

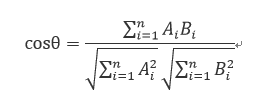
我们怎么量化两个事物之间的相似度呢？常见的方法是利用欧式距离和余弦相似度。利用欧式距离时，我们把A、B用户看做两点，用两点间距离表示二者相似度。使用余弦相似度时，则把二者看成同一坐标系下的两个向量。两个向量间夹角大小反应出他们的相似度，夹角越小则相似度越大。二维空间向量表示为r(x1,x2),多维空间向量表示为r(x1,x2…,xn) 比如，假设用户有5个维度

对喜剧的喜欢程度（1~5分），对动作片的喜欢程度（1~5分），对爱情片的喜欢程度（1~5分），对恐怖片的喜欢程度（1~5分），对科幻片的喜欢程度（1~5分）。

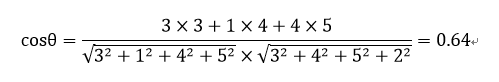
用户A：对喜剧的喜欢程度3，对动作片的喜欢程度1，对爱情片的喜欢程度4，对恐怖片的喜欢程度5，对科幻片喜欢程度0，用户A可以用向量表示为r\_A (3,1,4,5,0)

一个用户B：对喜剧的喜欢程度3，对动作片的喜欢程度4，对爱情片的喜欢程度5，对恐怖片的喜欢程度0，对科幻片的喜欢程度2，用户B可以用向量表示为r\_B (3,4,5,0,2)

对于向量A和B而言，他们的在多维空间的夹角可以用向量余弦公式计算：

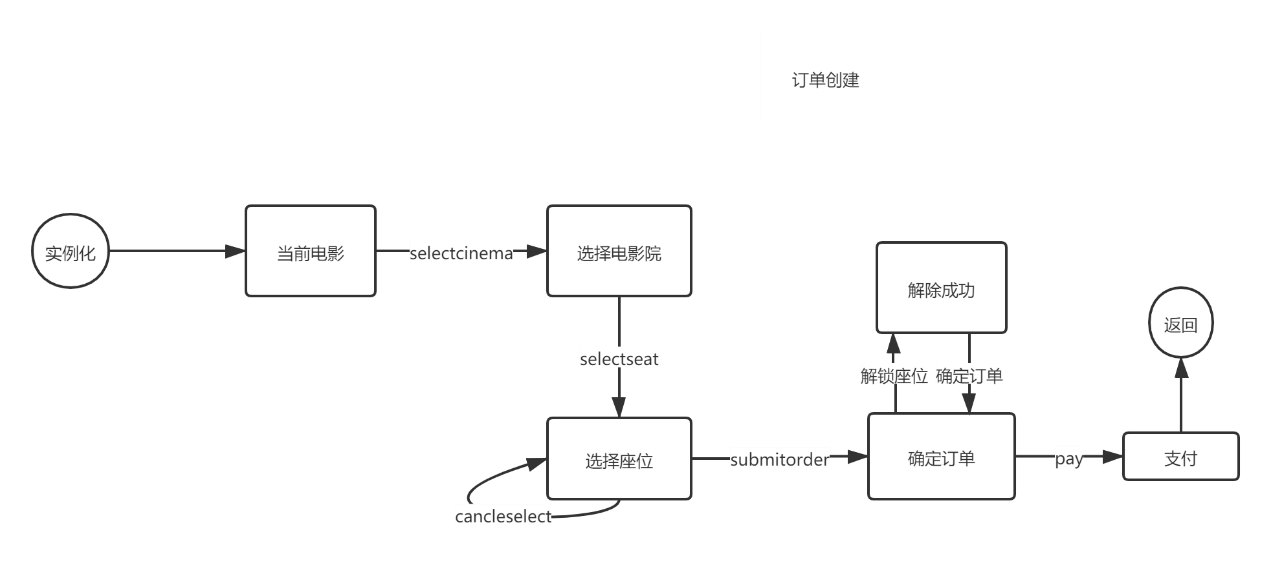


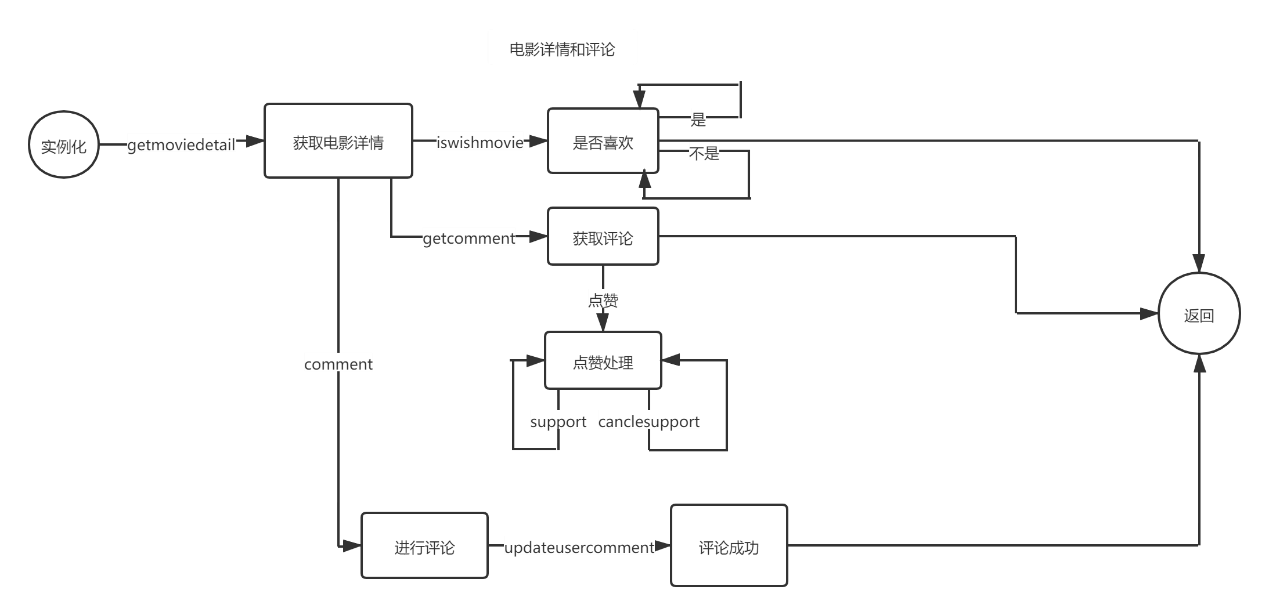
余弦相似度取值在0到1之间，0代表完全正交，1代表完全一致。那么用户A和B的相似度计算：

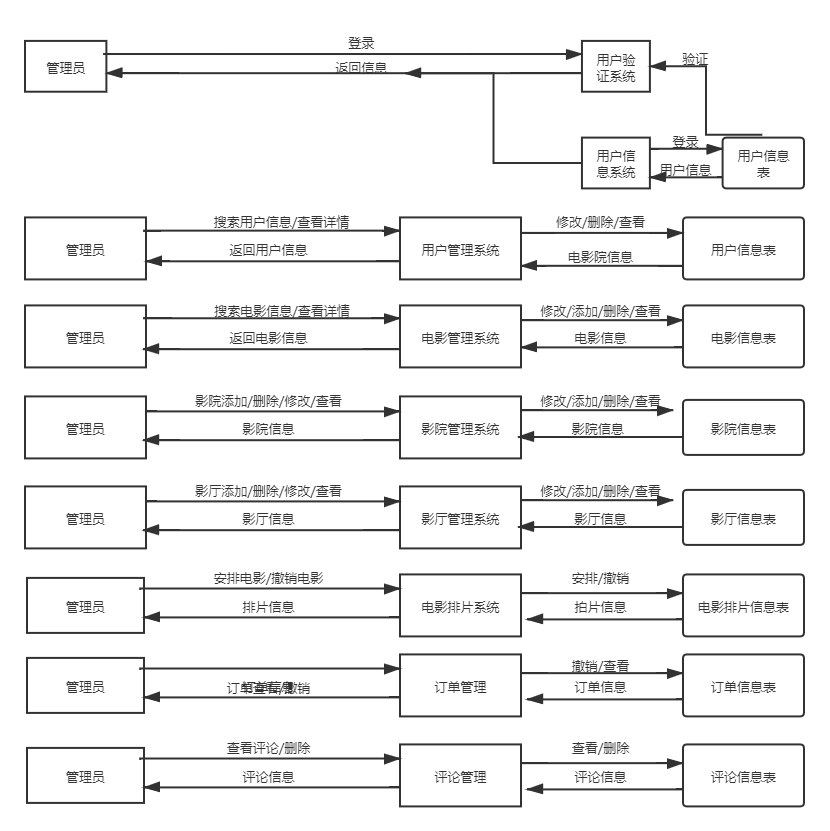


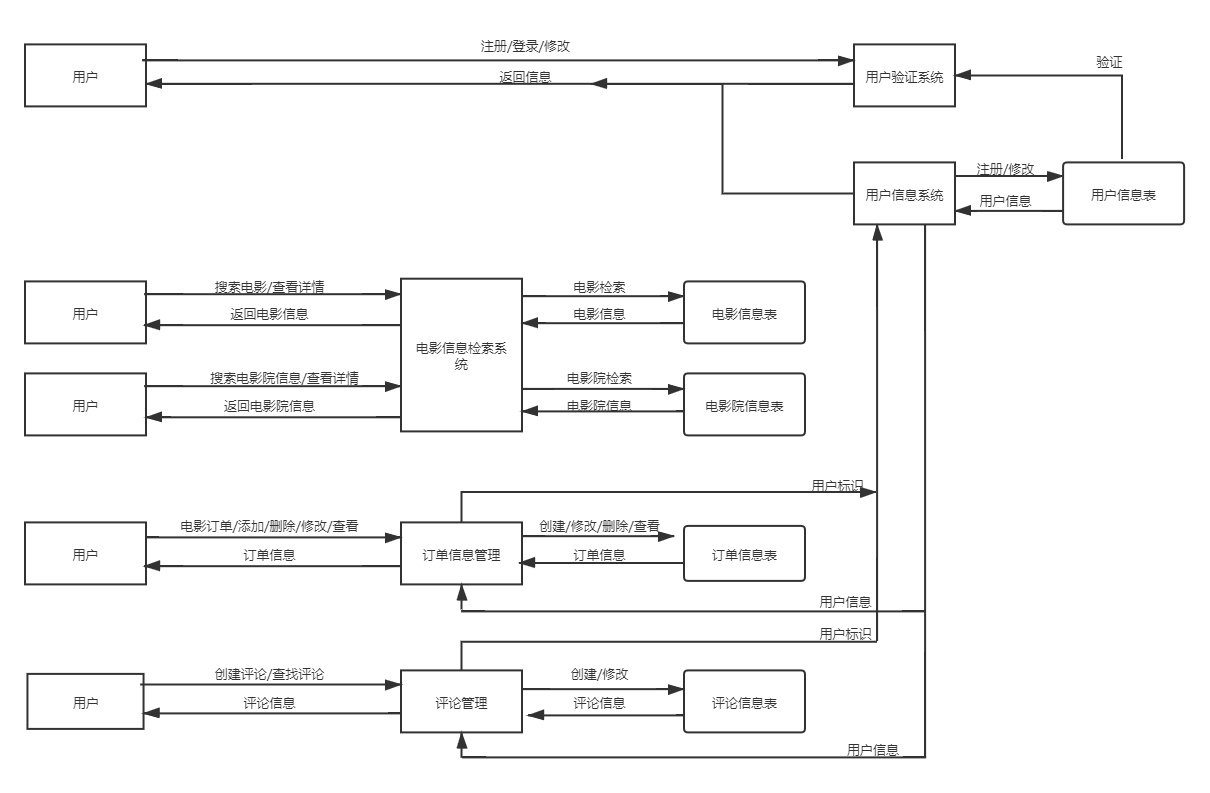
即代表了两个用户电影偏好的相似程度。

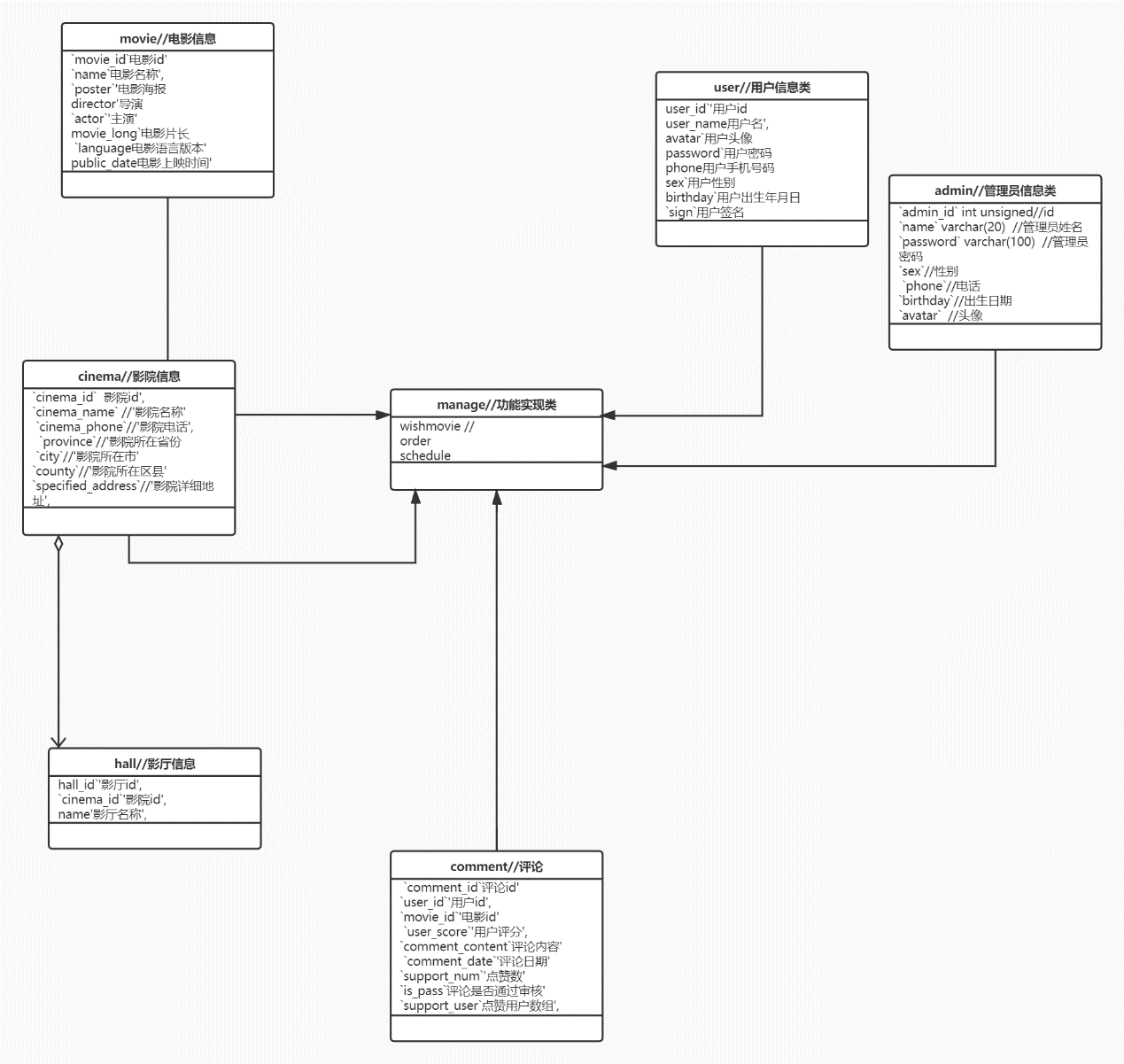
## 5.7流程逻辑





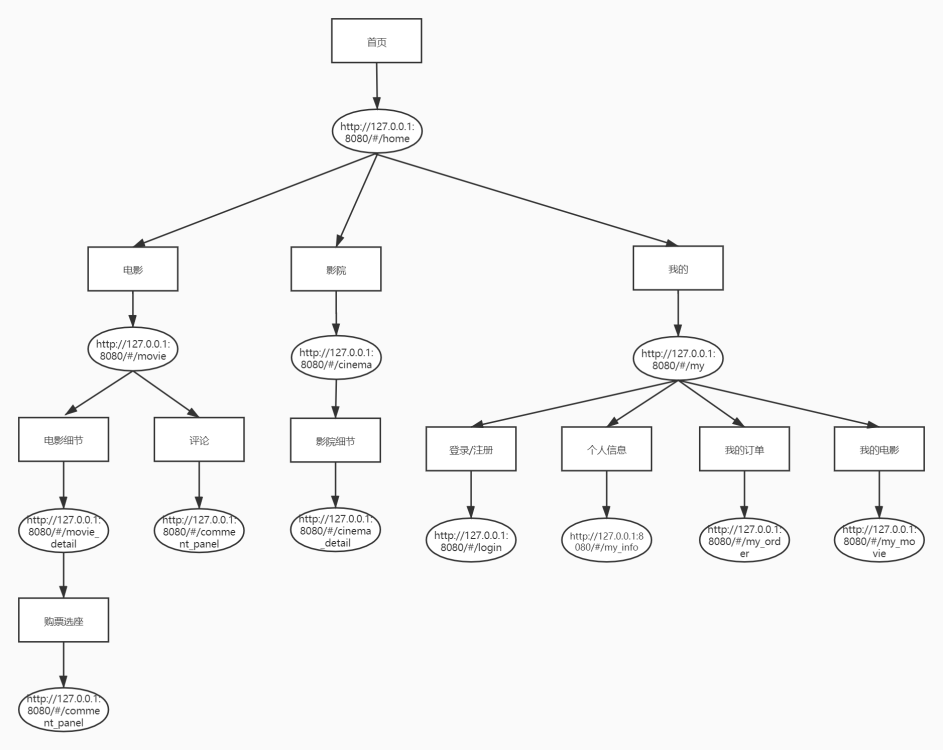






# 6.接口：

接口图：



# 7.注释设计

源文件外部对各部件组成功能进行说明。

函数开始前应进行注释，列出：函数的目的/功能 等

# 8.限制条件

后端只在windows系统下进行开发和测试，在其他系统可能出现无法运行或出现异常的情况。

前端可以在移动端访问，也可以在windows端访问。

# 9.测试计划

技术要求：在windows系统下，安装多个浏览器，如chrome 火狐 ie edge 360 qq浏览器等市面常用浏览器，分别使用它们打开测试项目，观察是否能正常运行。

输入数据：进行搜索，注册，选座，看评论，看电影信息和海报界面看能否正常显示。

预期结果：预期在这些浏览器下都能正常完成操作。

# 10.尚未解决的问题

# 11.注解

参考资料

[1] 《Vue.js从入门到项目实战》 胡同江编著，清华大学出版社

[2] 《Vue.js前端开发技术》， 王凤丽, 豆连军编著 人民邮电出版社

[3][美]Shari Lawrence Pfleeger [加]Joanne M.Atlee 著，《软件工程(第4版)》，人民邮电出版社

[4]刘竹林 编著，《软件工程案例开发与实践》，清华大学出版社

# 12.总结

基于vue.js的在线电影售票系统是一种精简，快捷，方便的实用性平台。该系统不仅可用于影院以及电影的在线管理，也能够记录用户信息和评论、初步实现社区化信息交流。网站式的售票平台使得用户和电影资源的管理更加合理化，信息化。买票的同时还具有电影搜索、电影评价、选座、等功能，可以实现较为灵活的网站内容切换和文本信息管理。

基于vue.js的在线电影售票系统是一个较为整合的系统，它涉及到管理功能和售卖功能等。在这次设计中，小组完成了用户注册/登录，相关数据处理，实时排片等基本功能。通过这次设计，小组对html、JavaScipt和Vue等技术应用方面有了更深入的认识和理解，取得了更大的进步。