



本科毕业设计

|  |  |
| --- | --- |
| 题 目 | 影片销售渠道管理平台的设计与实现 |
| 学生姓名 | 池昌澎 |
| 学 号 | 1631508141051 |
| 院 部 | 电子信息科学学院 |
| 年 级 | 2015级 |
| 专 业 | 信息管理与信息系统 |
| 指导教师 | 陈端芝 |
| 职 称 | 副教授 |
| 完成日期 | 2019年5月20日 |

福建江夏学院本科毕业论文诚信声明书

本人郑重声明：

兹提交的毕业论文《 影片销售渠道管理平台的设计与实现》，是本人在指导老师 陈端芝 的指导下独立研究、撰写的成果；论文未剽窃、抄袭他人的学术观点、思想和成果，未篡改研究数据，论文中所引用的文字、研究成果均已在论文中以明确的方式标明；在毕业论文工作过程中，本人恪守学术规范，遵守学校有关规定，依法享有和承担由此论文产生的权利和责任。

福建江夏学院本科毕业论文

指导教师承诺保证书

本人郑重承诺：

我已按有关规定对本篇毕业论文的选题与内容进行了认真指导和审核，且提交的毕业论文终稿与上传至“大学生论文管理系统”检测的电子文档相吻合，未发现弄虚作假、抄袭的现象，本人愿承担指导教师的相关责任。

# 摘 要

国内电影一直处在稳定发展状态的同时，也与国外电影的距离越拉越近。由此随着观影人数的越来越多，对影院影片的信息化管理成为了必不可少的一个步骤和目标。建立一套科学的、具有可操作性的高效的电影院信息管理服务体系，对于提高工作效率和服务质量具有重要作用。电影院在计算机上建立信息管理系统不仅方便了消费者，而且更大意义的方便了影院的工作人员。

影院销售渠道管理平台系统依据的是ECMAScript6标准开发，使用到NodeJS，Vue并采用Koa, Vue-element, e-Charts, cheerio框架；在此基础上，使用了不同于MySQL, Oracle等关系型数据库的NoSQL（非关系型数据库）MongoDB。该系统技术栈特点是全部使用JavaScript进行开发，保障了系统具有动态，高效，易于交互以及易于维护修改的特点。

此系统能够对影院影厅的信息和电影排片进行管理，在电影放映结束后能够对所销售的数据进行导入。之后系统会对所导入的数据进行统计处理，将每场电影销售数据进行统一格式化，最后将售票情况，区域售票情况，排片情况，销售渠道等信息返回给用户。

关键词：影院管理， 非关系型数据库， JavaScript

ABSTRACT

Since 21st century, in parallel with the stabilizing development of Chinese movie, the distance with International is getting closer. As the audience increases, the management of information technology has become an indispensable part. With the management and reveal of cinema and film schedule, Simultaneously, that System can run statistics on each data, for instance, sales data, regional sales data, film time schedules, sales channels and many more, then that system can show each data, graph and chart.

The Cinema Sales Channels Management System is developed with ECMAScript6 standard. The system running by NodeJS and Vue is use Koa, Vue-element, e-Charts, cheerio and another framework, furthermore, its database is using MongoDB in different from other Relational Database like MySQL and Oracle. The feature of that system is all used JavaScript developing, and guarantee it can be dynamic, efficient, easy interaction and easy maintaining.

**Keywords**: Cinema Management， Not Only SQL， JavaScript

目录

[1 绪论 1](#_Toc956)

[1.1 选题背景 1](#_Toc3615)

[1.2 课题目标 1](#_Toc2675)

[2 理论知识与开发技术简介 2](#_Toc9771)

[2.1 UML建模语言 2](#_Toc14169)

[2.2 MVC架构简介 2](#_Toc21369)

[2.3 SpringBoot简介 2](#_Toc23053)

[2.4 mybatis 简介 3](#_Toc6057)

[2.5 easyUI 简介 3](#_Toc1086)

[3 需求分析 4](#_Toc16883)

[3.1 用户需求概述 4](#_Toc25445)

[3.2 需求分析 4](#_Toc136)

[3.3 系统UML用例图 5](#_Toc12583)

[4 个人图书馆系统设计 7](#_Toc11761)

[4.1 系统总体结构概述 7](#_Toc4970)

[4.2 数据库设计 7](#_Toc24023)

[4.3 系统功能模块设计 14](#_Toc18722)

[5 个人图书馆系统开发与实现 16](#_Toc13988)

[5.1 首页大厅 16](#_Toc19876)

[5.2 用户注册 16](#_Toc16665)

[5.3 用户管理短篇文章 17](#_Toc8649)

[5.4 用户管理长篇小说 18](#_Toc30579)

[5.5 通讯录 23](#_Toc1541)

[5.6 用户信息管理 24](#_Toc24733)

[5.7 搜索 24](#_Toc31352)

[6 个人图书管系统功能测试 28](#_Toc23833)

[6.1 用户注册测试 28](#_Toc5125)

[6.2 馆藏模块测试 29](#_Toc2899)

[6.3 信息管理模块测试 31](#_Toc14414)

[6.4 搜索模块测试 31](#_Toc7268)

[6.5 用户管理模块测试 32](#_Toc32732)

[6.6 审核模块测试 32](#_Toc11158)

[后记 34](#_Toc25177)

[参考文献 35](#_Toc4356)

[致谢 36](#_Toc24815)

# 绪论

## 选题背景

电影，是由活动照相术和幻灯放映结合发展起来的一种现代艺术，有着复杂繁多的科系。随着技术力的发展，诞生出越来越多的优秀电影。同时，随着我国的物质生活水平提高以及更趋向娱乐化生活，观看电影也成为了广大群众不可或缺的娱乐活动。借助该影院销售渠道管理系统用户可轻松地使得数字影院系统优化，影院的运作更加简便和自动化。

## 1.2 课题目标

该影片销售渠道管理平台的设计是面向影院管理人员的，其初衷是为了帮助影院管理人员更加清楚清晰的进行影院的管理以及对当前档期的影片进行排片以及将众多影院数据进行统计使得管理人员能够清楚清晰的分析出在某地区某类影片的受欢迎程度。在完成数据库设计以及后台程序代码的同时也保持前台页面良好的页面和交互体验。

# 2 理论知识与开发技术简介

本系统采用MVC架构模式，该架构意于其将系统分为三个部分：

1、Model（模型） 是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。

2、View（视图） 是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。

3、Controller（控制器） 是应用程序中处理用户交互的部分。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

前端框架选择了代表MVVC模式的Vue，其是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。因其在去年所更新的2.0版本提供了Vue单向数据流的解决方案Vuex, 并且她的生态成熟且具有庞大的开发者群体，上手成本低且开发效率高也是选择她的理由。相比较之下同样的前端框架Angular和React会更加庞大没有Vue如此亲民。

后台所选择的是NodeJS的KOA框架，该框架是NodeJS框架的后起之秀，但对于在2013年就开始成为热门的Express等框架，她是NodeJS中最为小巧，且富有表现力的框架，并且通过利用ES7的新特性async函数，也丢弃了回调函数同时也能更积极的处理错误。并且因为是基于ES6(ECMA2015)的核心generator，KOA具有更加健壮的基石。这也是选择她而放弃Express, Meteor等优秀框架的理由。

数据库使用的是非传统的NoSQL非关系型数据库MongoDB。NoSQL全称Not Only SQL意即"不仅仅是SQL"，泛指当下的所有非关系型数据库，其相对于老牌的数据库像MySQL, Oracle等，她没有声明性查询语言，没有预定义的模式，她更自由且更加的开放。支持的数据结构非常松散，是类似json的bjson格式(BJSON全称：Binary JSON, BJSON把文件的二进制格式的数据直接保存到MongoDB的文档结构中)，因此可以存储比较复杂的数据类型[1]。MongoDB是一个基于分布式文件存储的数据库，是非常典型的NoSQL。其操作也是基于ECMAScript。

在此之上，又使用了页面交互框架Vue-element-UI，该框架在各大网站例如阿里云、腾讯云等网站都有使用，因为其优秀的交互反馈以及对简易性，在避免自己造轮子的同时也保持了页面的优雅精致。

而服务端的cheerio插件是jQuery核心功能的一个快速灵活而又简洁的实现，主要是为了用在服务器端需要对DOM进行操作的地方，亦可在服务端的配合下成为爬虫。在本系统中，她最大的作用是实现对电影数据库网站数据的爬取。

在展示数据方面上则是用E-Charts插件，她是一个使用 JavaScript 实现的开源可视化库，底层依赖轻量级的矢量图形库 Z-Render，提供直观，交互丰富，可高度个性化定制的数据可视化图表。

由此该系统的技术栈，前端Vue，后台KOA，数据库MongoDB，其最大的特点是是全部由JavaScript进行操作，具有统一，明晰，易读，轻量，扩展性高，易于维护等特点。

# 3 系统分析

## 3.1 系统功能需求

该系统的主要是面向影片销售方，主要向影院方提供了对影院和电影排片一系列管理的功能。影院方可以在首页获得到最近电影的上映信息、销售用户总览以及当日排片，之后，可以对自己影院的信息和影厅的信息进行修改编辑和管理，对电影的编排进行管理操作，在对上映结束后的电影可以进行销售数据导入。导入成功后台将会对数据进行统计分析操作，然后将所产生的结果以图表表格的形式展示，在此之上，该系统也向用户提供了条件筛选的功能，用户能从时间，地区，省份，城市来进行条件筛选。

整个系统分为用户（影院方）单个角色。

其中系统由本系统后台进行负责，完成对数据的收集以及处理。

对于用户的功能划分为几个相互关联的模块，它们都具有独立的界面，其中包括：

1. 信息总览模块：主要是对当前影片票房数据的概览以及对站外新闻文章的抓取展示，其中用户点击站外新闻会将跳转到该新闻的原网站。
2. 影院管理模块：包含了影院信息的管理、影院名下影厅的管理。主要是对影院以及影厅的信息进行的增加，删除，修改，查询操作。
3. 电影编排管理模块：本系统采用手动编排模式，包含了对电影排片信息操作，电影上映状态修改，电影售票信息导入操作。其中电影的数据更新会在后台自动处理，会从公用电影接口上获得最近上映电影的更新。若用户在同个时间段在同个影院和影厅有重复编排的操作会被拒绝。
4. 数据统计模块：再对所上映的电影进行数据的导入后，系统会在后台自动地对数据进行统计分析，并以图表表格的形式展示，并具有条件筛选的功能。

3.2 系统UML用例图

用例图将在用户的角度上描述该用户所应具有的功能。

影院方用户可以管理影院及影厅信息，编辑电影排片数据，导入销售数据，查看数据统计，用例图如图3-1所示。

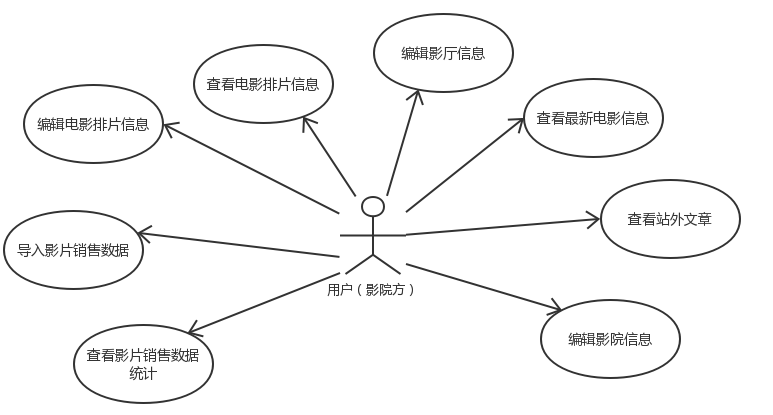


图3-1 用户用例图

# 4系统设计

在分析好系统需求之后，开始进行系统的设计。系统设计主要是针对系统功能模块的设计，系统设计是用户需求的最直接体现。

## 4.1 系统总体结构概述

该影片销售渠道管理系统的目标是帮助影院方能够更方便快捷的管理影院以及影片数据，并且对数据进行更系统化的统计，以此来提高影院管理方的工作效率以及分析数据的准确性。其系统总体结构框图如图4-1所示。

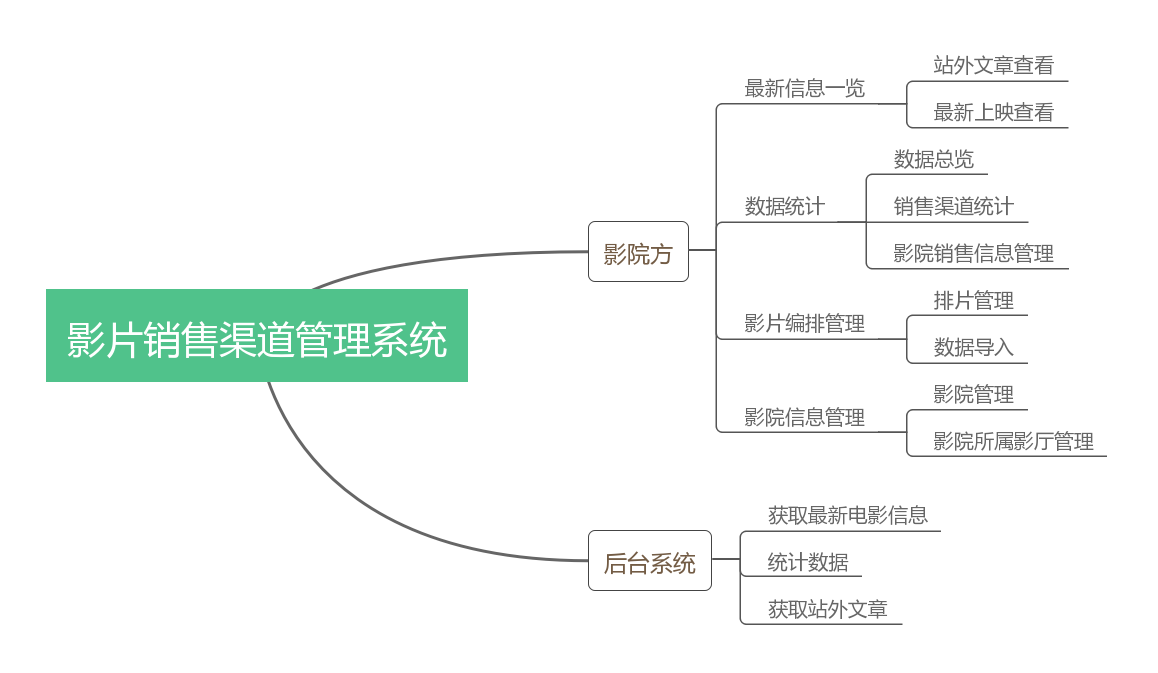


图4-1 影片销售渠道管理系统结构图

## 4.2 数据库设计

本系统采用MongoDB 4.0，该数据库是使用文档格式的非关系型数据库，但在此之上，相比于Redis, SQLite等典型非关系型数据库，是功能最丰富，使用最像关系数据库的。

不同于 RDBMS（关系型数据库）的形式，MongoDB中基本的概念是域(field)，文档(document)，集合(collection)，数据库(database)，分别对应RDBMS的数据字段（column），数据记录行（row），数据库表（table），数据库（database）。

并且NoSQL无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。该特点赋予了NoSQL数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。一般MySQL使用Query Cache，每次表的更新Cache就失效，是一种大粒度的Cache，在针对WEB2.0等交互频繁的应用中，Cache性能不高。而NoSQL的Cache是记录级的，是一种细粒度的Cache，所以NoSQL在这个层面上来说就要性能高很多了[2]。

但带来性能的同时也产生了难以进行多集合关联查询的缺点，并且无法像关系型数据库一样进行表连接。为了克服这一缺点，必须在数据引入创建编辑时尽心逻辑缜密的处理。在保存数据时，就可以将未来可能会使用到的数据进行额外处理。比如在导入单场电影的销售数据时，后台处理就额外地将N条的销售数据进行整理并保存于另一个易于查询的聚合当中。

### 4.2.1 数据库集合设计

在本系统中，每一个用户对应一个单独的数据库。并且拥有一个公共数据库是用以整理全局数据。其关系如图4-2-1

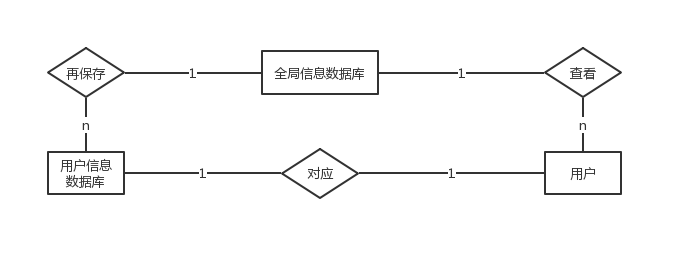


图4-2-1 数据库关系E-R图

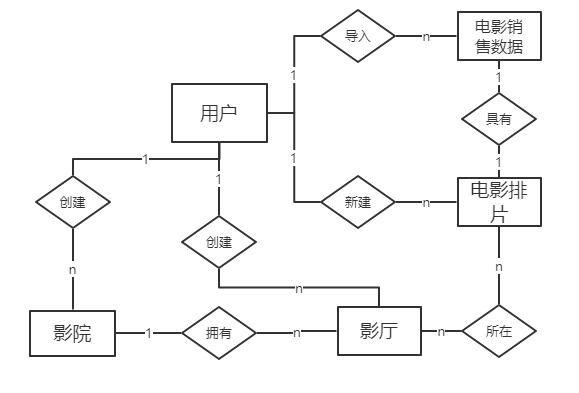
在用户数据数据库中，其中有影院信息、影厅信息、电影编排信息、电影销售导入信息，这一些集合是用户会直接操作到的。其中该数据库的总E-R图如图4-2-2

图4-2-2 用户数据数据库E-R图

在此用户导入电影销售数据时，所导入的电影销售数据是按照销售一人一条的数据格式，在导入成功后后台系统会自动将该场电影的数据聚合成新的单个的文档。该类图如图4-2-3

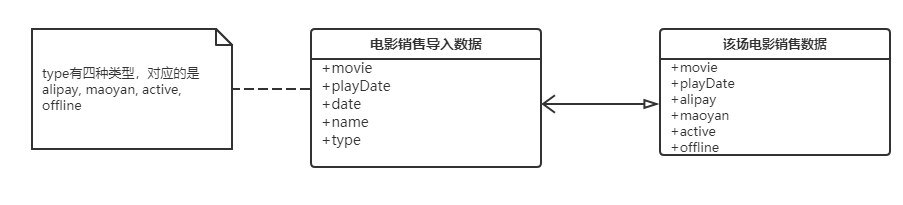


图4-2-3

电影销售导入数据集合中有电影（movie）、播放时间（playDate）、购买渠道（type）三个非空值的域，其中电影和播放时间是对应着该场电影销售数据的域，类似于RDBMS的主键。

## 4.3 系统功能模块设计

下面对图4-1的功能模块再进行详细设计。

后台系统主要功能包括获取最新电影数据，统计电影销售数据，获取站外文章，下文以获取最新电影数据为例，其他功能与之类似就不一一赘述。

获取最新电影数据： 在用户进入首页时，系统会连接到外部影片数据库，在此同时会唤醒调用爬虫接口。然后应用爬虫接口获取外部影片数据库系统的数据。此时爬虫接口会获取到未经处理的html格式数据，将其返回到系统。系统再对该数据进行格式化，将其转化为易读的json数据返回给用户。如图4-20。

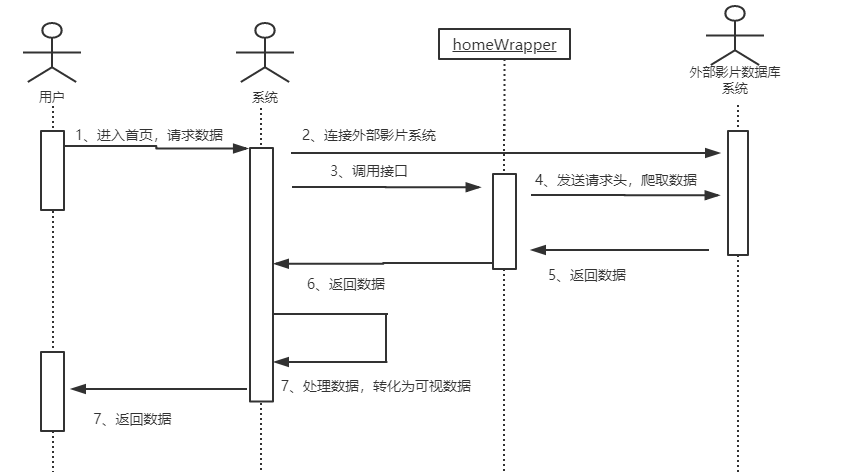


图4-3-1 管理员审核的顺序图

用户角色主要功能为：影片上映信息一览，影院影厅信息管理，影片排片管理，电影销售数据统计查询，下文以影片排片管理中的排片管理为例，其他功能就不一一详述。

新增影片排片：当用户进入到此模块时，系统先会向用户展示所有电影排片信息，当用户新增排片时，系统会将现在可以上映的电影名称显示在选择框。同时获取所有影院及影厅数据，形成一个级联选择器。之后向系统提交表单数据，系统将数据格式化为数据库可保存格式后判断是否在同一时间段有同一影厅有重复排片，若数据无误将数据存入数据库，其顺序流程图如图4-3-2所示。

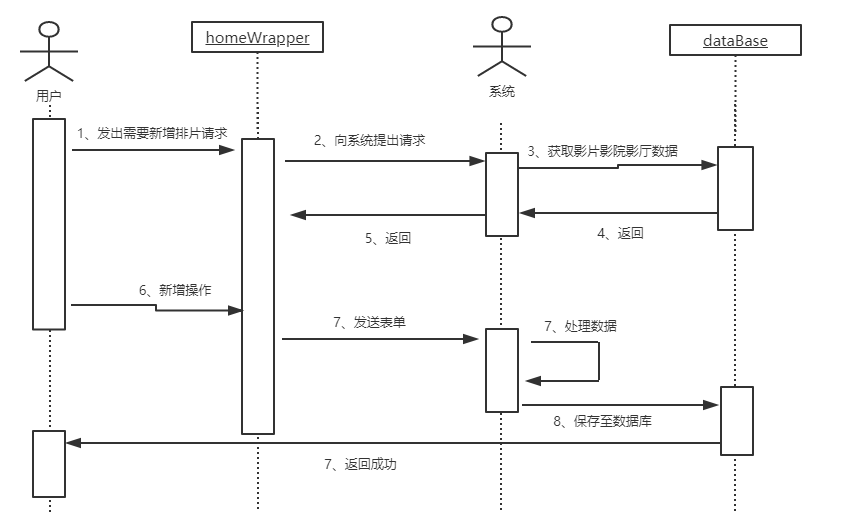


图4-3-2 新增排片顺序图

# 5 影院营销渠道管理系统开发与实现

本章介绍了影院营销渠道管理系统的功能。

## 5.1 总览页

如图5-1，该页面主要是向用户展示近期的数据，其中包括从外部网站获得的影评文章，用户点击会跳转至该文章的原地址；当日销售数据及总销售数据，其中含有新增 场次，新增的观众，新售出的票数；最新可上映的电影，通过外部电影数据库系统获取相关数据展示；以及当日的电影排片。

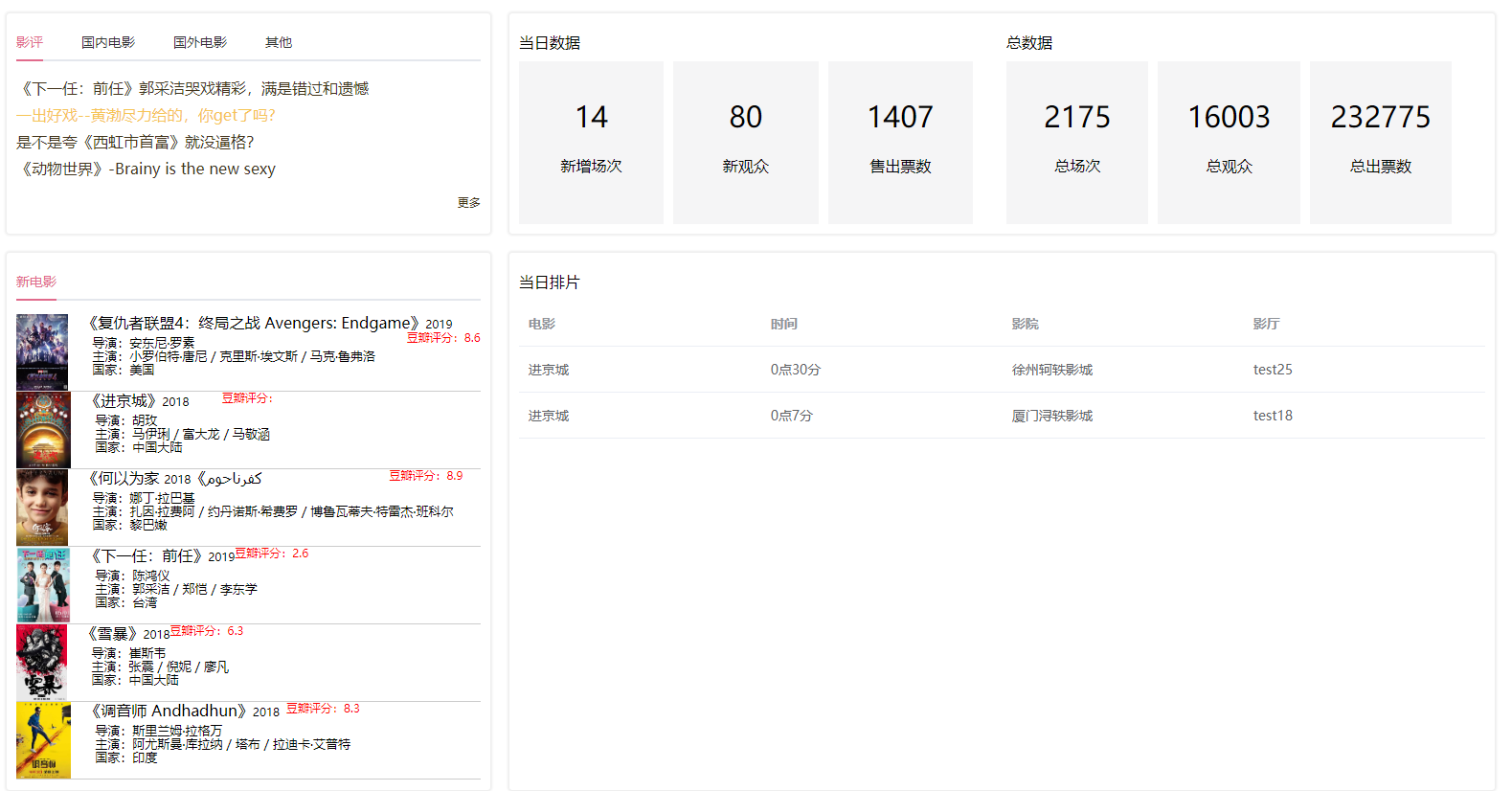


图5-1 影院销售总览主要界面

其中，对新电影信息获取的代码如下，其算法原理为当用户完成登入后，会直接先调用该接口，接口被调用时会使用node.js中的客户端请求代理模块superagent，向豆瓣电影网站发送html访问请求，随即豆瓣电影网站将html数据返回到系统后台，再通过cheerio模块抓取页面信息存入回传函数$()中，再对回传函数$进行遍历，将有用的信息存入预先准备好的空数组变量，最后将其返回到前台页面处理。

router.get('/newmovie', async (ctx, next) => {

// 当该接口被调用时

// 会用客户端请求代理模块 superagent 向豆瓣电影网发送请求

superagent.get('https://movie.douban.com/')

.end(function (err, sres) {

// 常规的错误处理

if (err) {

return err

}

let $ = cheerio.load(sres.text, {

decodeEntities: false

})

// 使用第三方模块cheerio将数据格式化放入回传函数$中

nm = [] // 定义一个数组变量mn用于存放接收到的数据

$('.ui-slide-item').each((idx, element) => {

// 选择$当中class=’ui-slide-item’的element代码

// 对其数据进行遍历，将每行数据中正确的信息存入数组变量nm

if (idx <= 5) {

let $element = $(element)

// console.log($('img')[idx].attribs.src)

nm.push({

img: $('img')[idx].attribs.src,

title: $element.data().title,

rate: $element.data().rate,

release: $element.data().release,

region: $element.data().region,

director: $element.data().director,

actors: $element.data().actors

})

}

})

})

ctx.body = nm // 向前台返回数组nm

})

## 5.2 影院管理

如图5-2，该模块用于对数据库中的影院进行修改编辑删除等操作。并且可由地区进行筛选。在此界面用户还能对影院当中的影厅进行预览操作。

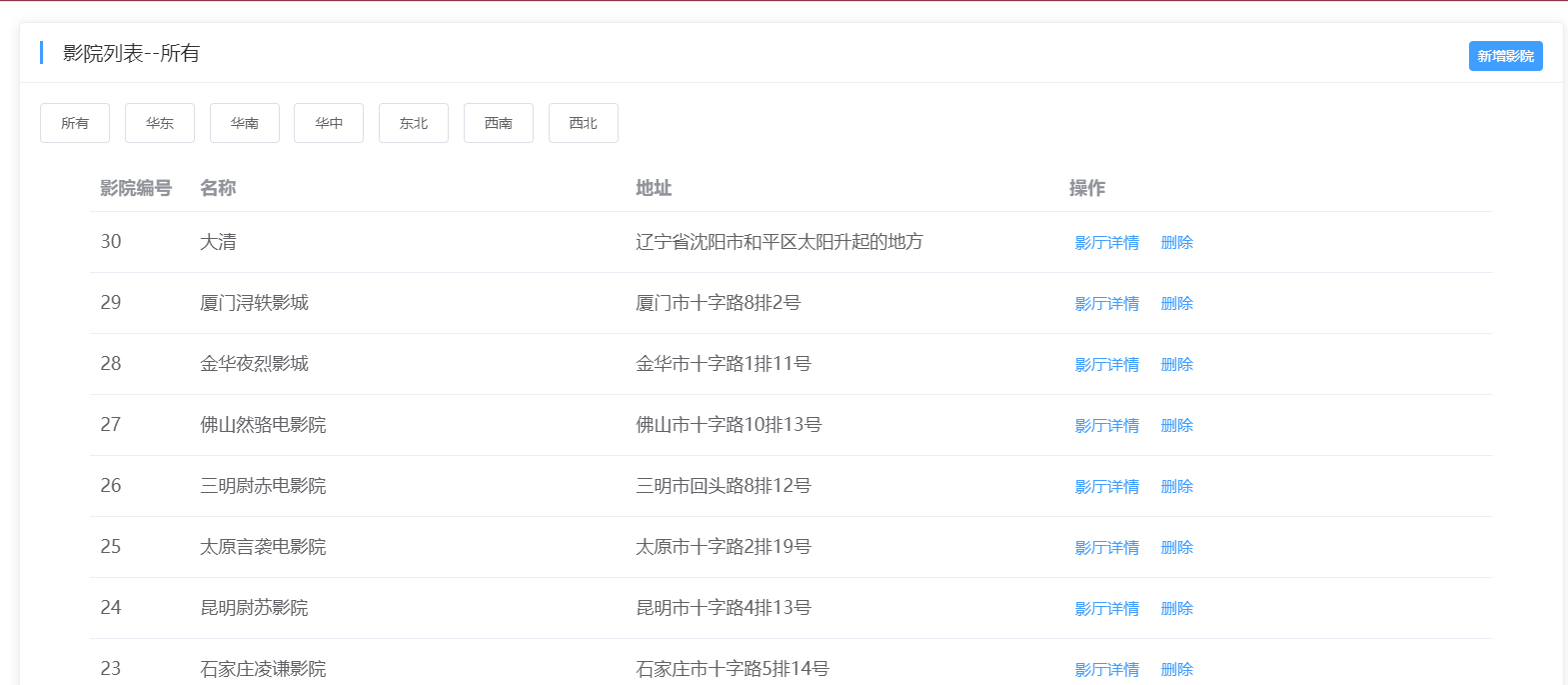


图5-2 影院管理主要界面

影院的添加界面如图5-3，用户点击影院管理界面中的新增影院界面就会跳出该窗口，需要用户填写影院名称、以及影院地址，并且影院地址通过使用jQuery完成3级级联查询。三级级联查询的设计思路是页面加载时，从后台获取第一级select的数据，第二级select的选项数据根据第一级select值的改变再动态更新，第三级select的选项数据再根据第二级select值的改变再做动态更新。

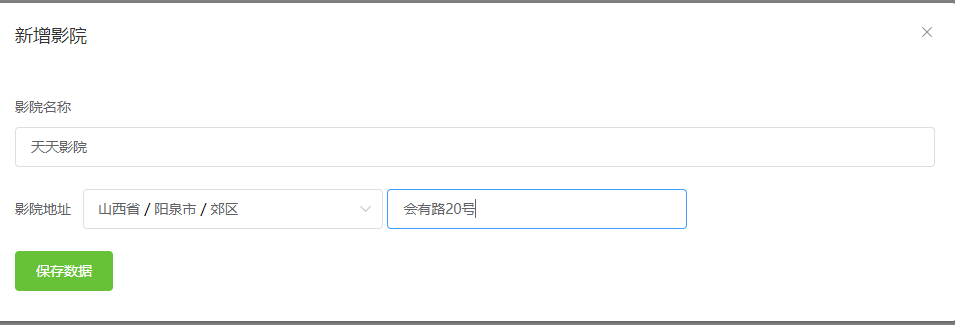


图5-3 增加影院主要界面



图5-4 影厅详情主要界面

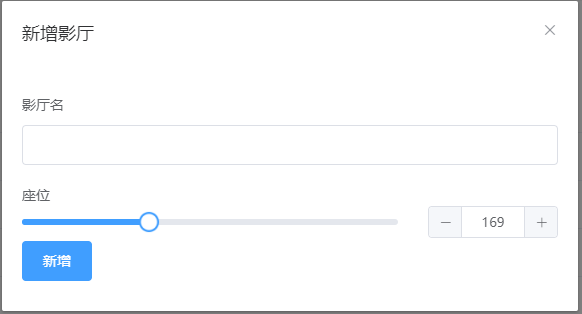


图5-5 新增影厅主要界面

影厅管理操作如图5-4，5-5所示。在影院管理界面点击查看影厅，会显示该行影院当中的影厅数据，同时可以对其进行修改。

## 5.3 排片管理

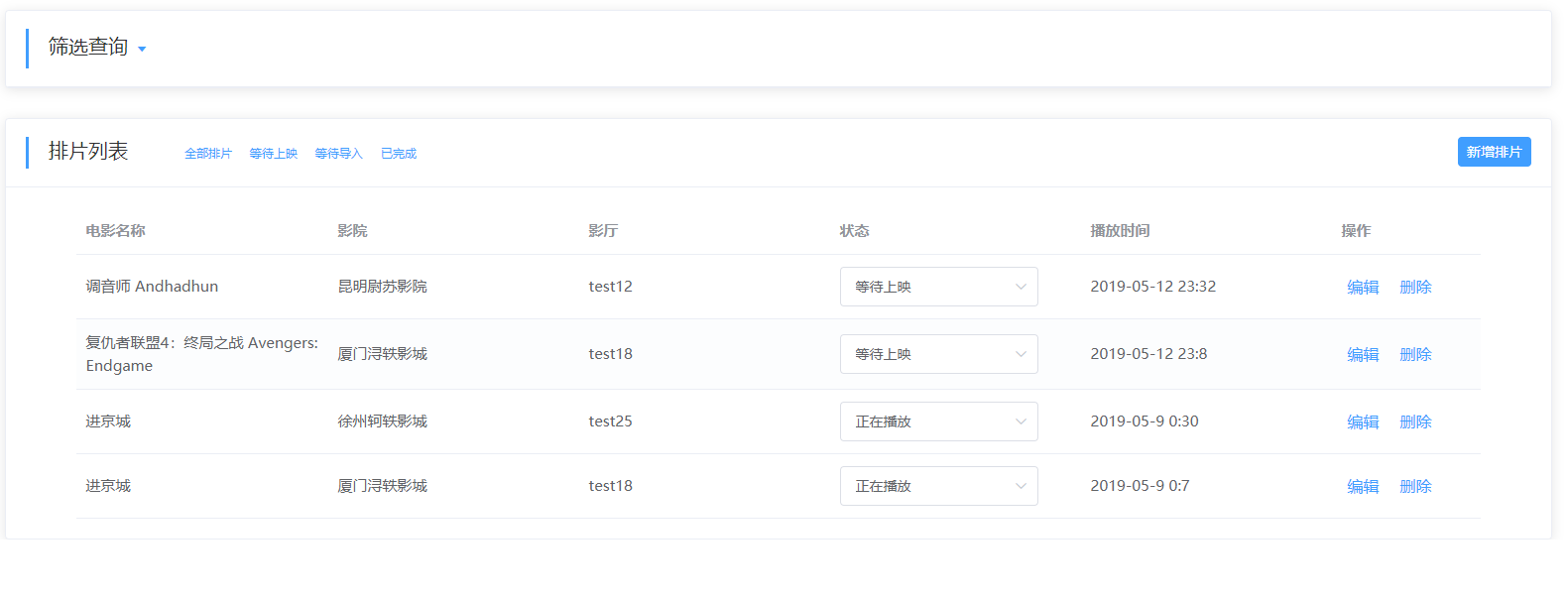


图5-6 排片管理主要界面

排片管理模块界面如图5-6，其中筛选界面展开如图5-7，其可对排片的电影名称，地点，时间进行筛选，用于更灵活的数据展示。

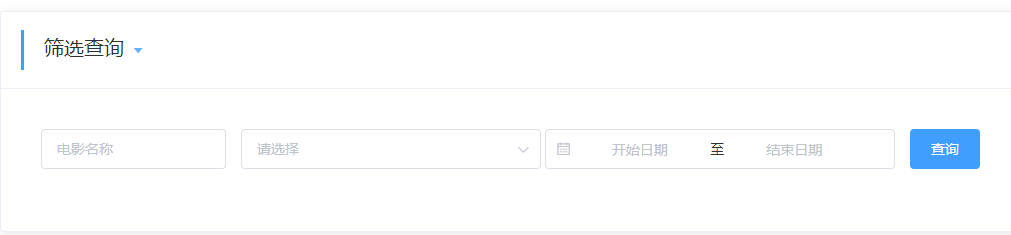


图5-7 筛选界面



图5-8 新增排片界面

新增排片界面如图5-8，用户需要输入电影名称，选择时间，选择影厅。同时影厅和影院也同时进行了级联查询，并且点击电影名称输入框时会自动列出输入建议，其中输入建议使用到了javascript的第三方插件autocomplete，她是一个可带输入建议的输入框组件，fetch-suggestions 是一个返回输入建议的方法属性，如 querySearch(queryString, cb)，在该方法中你可以在你的输入建议数据准备好时通过 cb(data) 返回到 autocomplete 组件中。其代码如下

querySearch(qs, cb) {

let data = this.data

let result = qs ? data.filter(this.createFilter(qs)) : data

// 调用 callback 回调函数返回建议列表的数据

cb(result)

},

并且对电影进行编排的同时会对当前时间进行判断，若排片时间设定在当前时间以前，则会拒绝排片。且在用户进入该页面时也会自动判断修改电影播放状态。在结束放映后，会自动将状态转变为已结束并可以导入销售数据。导入销售数据界面如图5-9，其中只能导入格式为json的文件，并且导入格式不正确或数据异常则会将拒绝该数据导入。

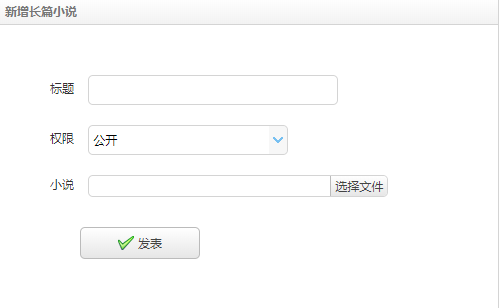
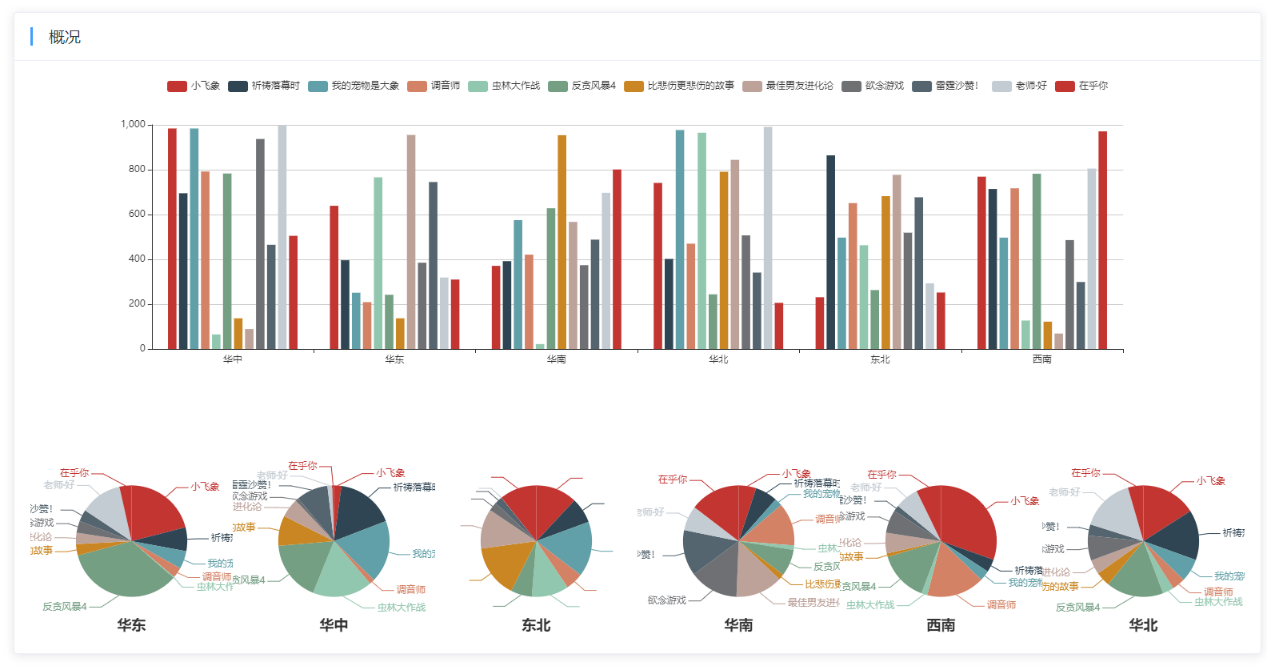


图5-9 导入数据界面

## 5.4 销售统计模块

统计模块中包括销售概况统计，这个统计是针对所有地区的对所有电影销量进行的统计以及数据可视化工作；销售渠道统计是对于某地区，甚至到某个影院某部电影的关于销售渠道的统计以及数据可视化统计；影院销量则是针对某地区某电影院所进行的数据统计可视化工作，以方便更好地进行系统化管理。其中可视化和统计绘图是使用E-chart框架，免去了自己造轮子的麻烦。其中展示图如图5-10所示





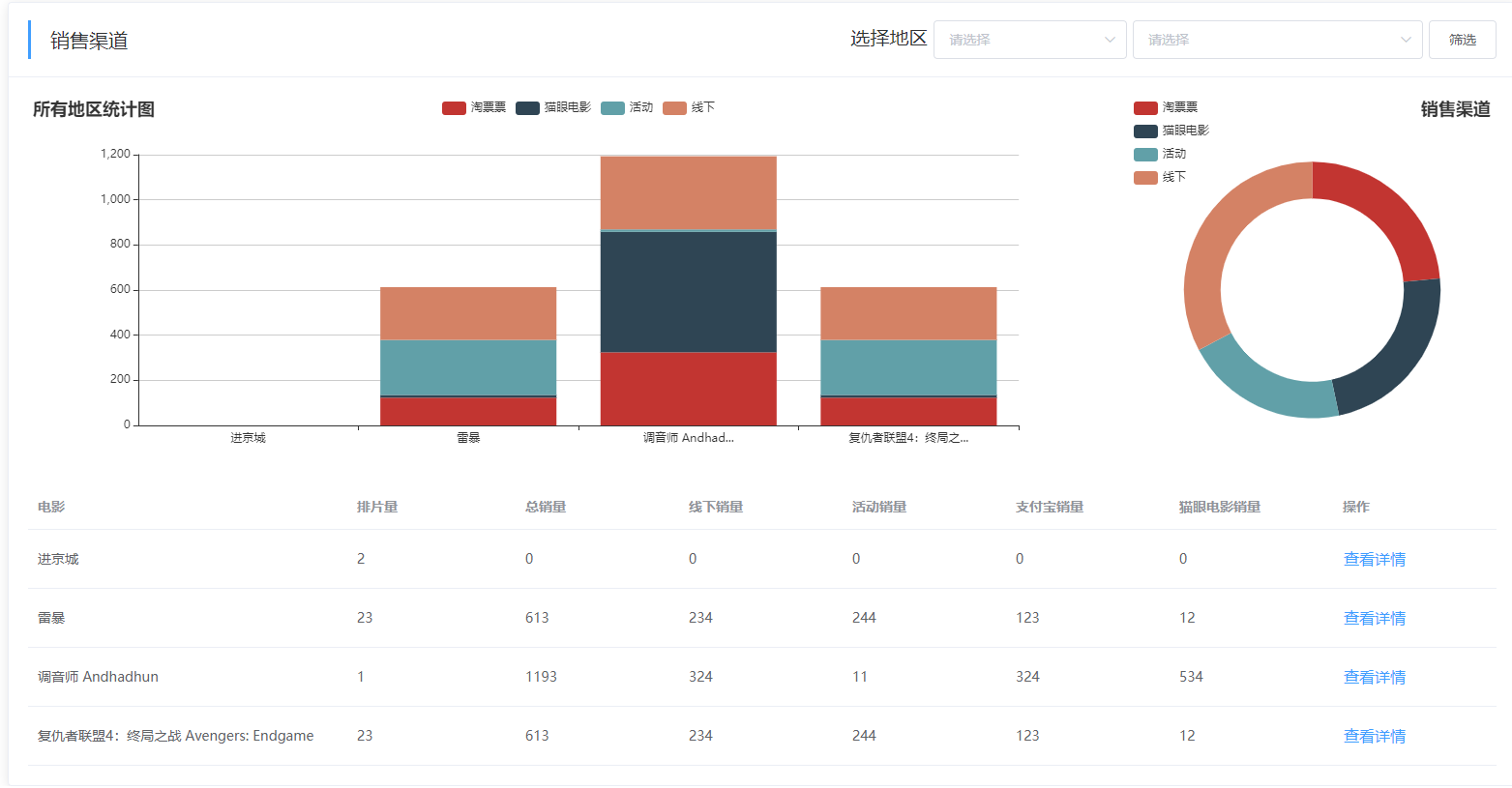


图5-10 数据统计展示界面

在该界面中，同样可以对数据进行筛选，条件筛选是在前台界面层实现，其代码是：

if (param.search) { // 判断是否筛选

let check = [] // 定义数组用于临时存放处理的数据

if (param.movie) { // 判断电影名称条件

// 电影名称

result.forEach(item => {

if (item.mname === param.movie) {

check.push(item)

} // 对之前的数据遍历，进行筛选

})

} else {

check = result

}

if (param.city || param.province) {

// 判断地区

let cinema

let area = []

if (param.city) {

cinema = await Cinema.find({city: param.city})

} else {

cinema = await Cinema.find({province: param.province})}

// console.log(check)

cinema.forEach(cinema => {

check.forEach(item => {

if (item.cinema === cinema.cinema) {

area.push(item) }

})

})

check = area

}

if (param.star && param.end) {

let area = []

check.forEach(item => {

let ptime = new Date(item.ptime)

let star = new Date(param.star)

let end = new Date(param.end)

if (ptime.getTime() >= star.getTime() && ptime.getTime() <= end.getTime()) {

area.push(item)

}

})

check = area

}

res = check

但当筛选条件中没有数据时则会在界面显示无数据，如图5-11。



图5-11 无筛选结果

6 个人图书管系统功能测试

## 6.1 影院管理测试

如图6-1以增加影院为例，添加影院会对地址和影院名称进行检查，当影院名称与数据库中有重复则会返回错误提示。

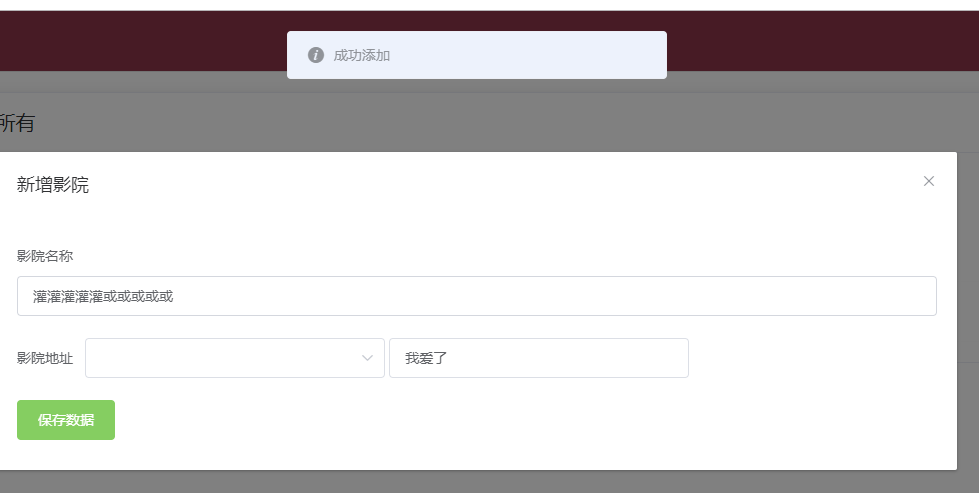


图6-1 用户注册测试图

图6-2对已存在影院名进行添加，弹出错误提示



图6-2 影院重复测试图

## 6.2 影片添加模块

用户在该界面可以对影片的编排进行编辑。在新增排片页面，激活电影名称输入框会自动弹出输入提示，如图6-3



图6-3 弹出输入建议

时间选择器如图6-4，在当中可对时间进行选择，但是将电影时间排在当天之前则会弹出错误。如图6-5，没法对当前时间以前进行排片。

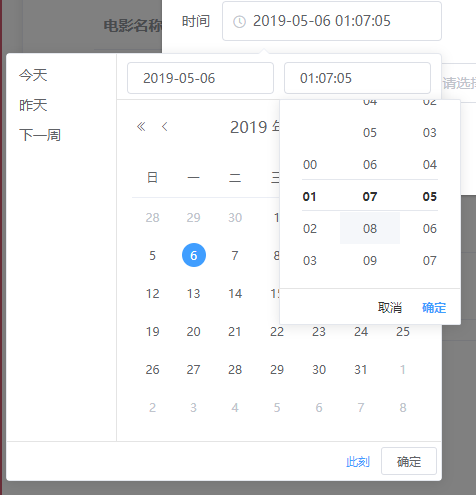




图6-4 时间选择器 图6-5

当对当天排片进行新增时会影响到首页的当天排片的数据，如图6-6，当当日没有排片时会在首页显示没有数据。添加当天排片后首页会出现对应的数据。



图6-6 当天没有数据的首页

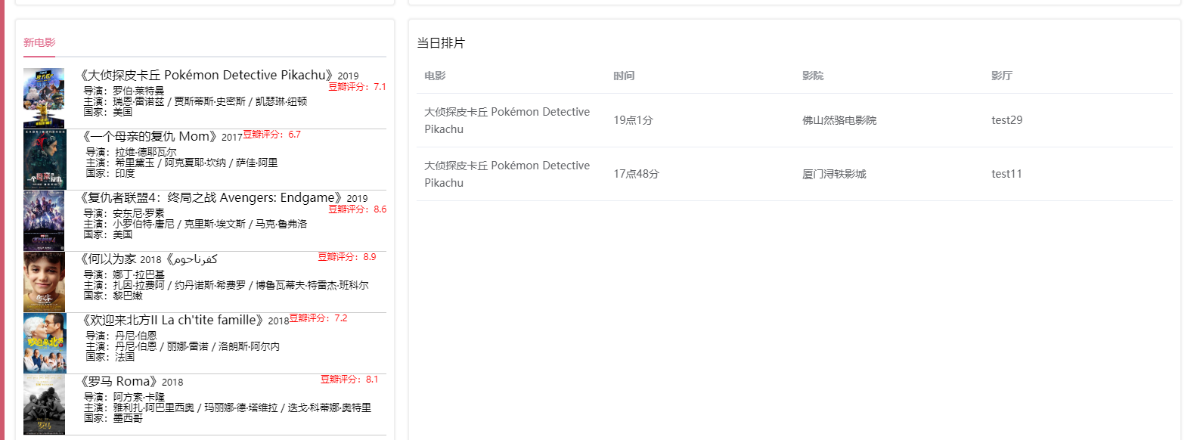


图6-7 添加当天数据后的首页

并且用户在进入该页面时系统会对当前的表格信息进行一次判断，若当前时间已超过一些电影的播放时间，则会将这些被超过的电影播放状态改为正在放映，若当前时间超过电影播放的结束时间，则会将状态改为已结束，并且在按键会显示导入数据。

筛选数据成功结果如图6-8

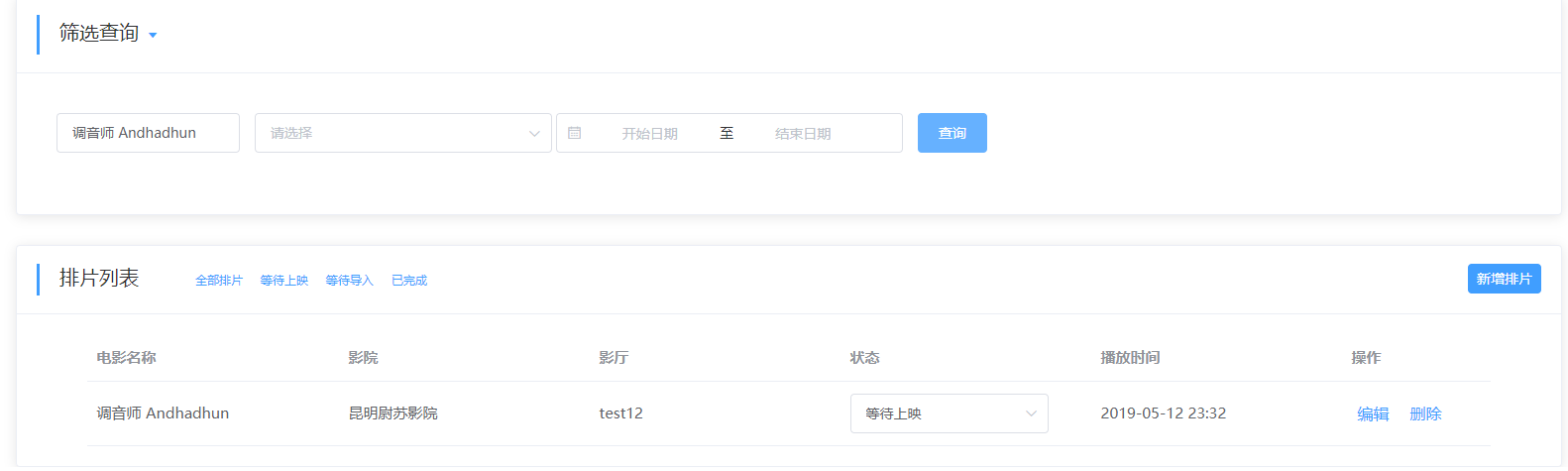


图6-8 筛选数据结果

# 后记

在该系统的制作中，其中遇到了很多困难，也感谢导师为我的指点。前期所分析的需求也基本都完成，测试也沟通过，但在该系统也存在许多问题，比如说数据库的选用，在这样数据联系比较紧密的系统当中，选用关系型数据库往往是能更好的用上数据库系统提供的资源，这是刚开始开发系统时没有考虑周全的问题，又因为这样从而对前台页面对数据要进行大量的处理导致页面载入速度及内存占用提高了许多。所以在经过这次的系统设计实现后，我也对数据库与非关系型数据库的使用区别有了更深的理解。由于在开发时对语言掌握不是很熟练，在一些地方也有重复出错的情况。在以后的开发当中还是得先将基础打牢固，这样才能建成更高的楼。

# 参考文献

1. 黄贤立. NoSQL非关系型数据库的发展及应用初探[J]. 福建电脑, 2010, 26(7):30-30.
2. Kristina Chodorow, Michael Dirolf. MongoDB权威指南[M]. 2011.3
3. 刘璐. 关系型与非关系型数据库对比分析[J]. 明日风尚, 2017(5).
4. 陶国荣. jQuery权威指南(第2版)[J]. 中国科技信息, 2013(24):132-132.
5. 王金龙, 宋斌, 丁锐. Node.js:一种新的Web应用构建技术[J]. 现代电子技术, 2015, v.38;No.437(6):70-73.
6. 李巍. 基于MongoDB的电网信息三维可视化系统数据管理[J]. 计算机与数字工程, 2019, 47(1):248-252.
7. 徐旭平. 基于MongoDB的元数据管理研究[J]. 信息技术, 2018, v.42；No.321(8):95-101.
8. MongoDB的存储与查询策略优化与应用[D]. 武汉邮电科学研究院, 2018.
9. 杨柳. 试析Web前端开发[J]. 电脑编程技巧与维护, 2018(1):135-137.
10. 于春娜, 王晨升, 杨光,等. Web前端MVC框架的意义研究[J]. 产业与科技论坛, 2014(1):52-53.
11. 高鹏. 新媒体领域中非关系型数据库的选择[J]. 广播与电视技术, 2013, 40(5):33-0.

|  |
| --- |
|  |

# 致谢

我首先要感谢我的指导老师、福建江夏学院电子信息科学学院的陈端芝老师。陈老师对我的系统和论文给予了巨大的帮助，在需求分析和系统功能改进方面悉心指导，提出了许多有建设性的意见，这也是我的系统和论文得以顺利完成的关键。同时，还要感谢电子信息科学学院信息管理与信息系统专业的授课老师们和所有同学们，大家在学习中互相帮助，互相学习，共同度过了一段完美难忘的时光。