



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。建立及规划一个严谨的、具有高效便捷度和可操作性的电影销售管理渠道系统，能够大幅度地增加工作效率和减少重复工作的时间。并且影院在建立标准规范化的信息管理系统，不仅能方便影院日常的工作，也能通过统计的数据也能帮助管理人员获取信息。

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（非关系型分布式数据库）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。

本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 reveal of cinema 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 MVVC架构 2](#_Toc14169)

[2.2 Vue前台框架 2](#_Toc21369)

[2.3 KOA后台框架 2](#_Toc23053)

[2.4 MongeDB非关系型数据库 3](#_Toc6057)

[2.5 其他插件 3](#_Toc1086)

[3 需求分析 4](#_Toc16883)

[3.1 系统功能需求 4](#_Toc25445)

[3.2 系统业务流程图 4](#_Toc136)

[3.3 系统UML用例图 5](#_Toc12583)

[4系统设计 6](#_Toc11761)

[4.1 系统总体结构概述 6](#_Toc4970)

[4.2 数据库设计 6](#_Toc24023)

[4.3 系统功能模块设计 9](#_Toc18722)

[5 影院营销渠道管理系统开发与实现 11](#_Toc13988)

[5.1 总览页面 11](#_Toc19876)

[5.2 影院管理 13](#_Toc16665)

[5.3 排片管理 14](#_Toc8649)

[5.4 信息数据统计 16](#_Toc30579)

[6 5 影院营销渠道管理系统功能测试 20](#_Toc23833)

[6.1 影院管理测试 20](#_Toc5125)

[6.2 影片添加模块 21](#_Toc2899)

[6.3 导入数据模块 23](#_Toc14414)

[后记 25](#_Toc25177)

[参考文献 26](#_Toc4356)

[致谢 27](#_Toc24815)

# 绪论

## 选题背景

电影是在照相机出现后，用照片与幻灯放映相结合发展成的一种艺术手段，其在中国的发展就可以追溯到上个世纪初的民国电影《定军山》。伴随着技术力与科技的发展，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。

随着经济发展的加速，市场上的竞争形势也日趋激烈，各种各样的中小企业也会面对更为激烈的市场竞争和更加开放的市场环境，在市场当中既有机遇又有挑战。同时也随着科技进步，信息化也成为了对一个企业发展和在市场竞争中不可缺少的一步。如何审时度势，对市场走向进行正确的判断。对于影片渠道营销上来说，如何对影片编排安排宣传，如何在现在大环境中依靠数据进行正确的抉择，使影院能够在市场竞争中能够占有一席之地，是每个影片管理者需要思考的问题。

## 1.2 课题目标

本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

2.1 MVVC架构模式

该模式是从MVC模式演变来的，有所不同的是，MVVC的核心是“数据模型双向绑定”， 在MVC模式中的界面层View和控制层model是没有直接联系的，而MVVC是通过介于MVC模式中V层和M层的ViewModel层进行数据交互，最大的特点是View层也能很轻松地操作后台数据以及当数据发生变化时View层也能将变化立即渲染到DOM上。

2.2 Vue前台框架

前端框架选择了代表MVVC模式的Vue，Vue的开发核心思想是MVVC的双向绑定，并且提供了一系列对数据处理的便捷方法。和其他由上而下的前端开发框架不同，Vue 的特性就表现在其开发过程时是自底向上逐层应用。并且Vue 的核心库是对界面层进行操作，在MVVC模式中，能够发挥到最大的作用。同时因为她小巧、扩展性强，她的用户群体也为她提供了各式各样的第三方库，更易于上手操作。

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。

2.3 KOA后台框架

后台所选择的是基于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)提出的中心支撑函数generator，也使得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。

2.4 MongoDB数据库

数据库使用的是非传统的NoSQL非关系型数据库MongoDB。NoSQL其意为不是数据库但也不只是数据库即Not Only SQL，其相对于老牌的数据库像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。她能够支持松散的数据格式Bjson，它是与Json较为相似的bjson格式。BJSON能够把将文件转化为二进制格式，然后将其保存到MongoDB文档结构中，因此也使得比较复杂的数据类型能够得以保存]。分布式数据库MongoDB，是非常典型的NoSQL。其操作也是基于ECMA。

2.5 其他技术

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。

服务端基于NodeJS的cheerio插件能够在服务端实现前端语言jQuery核心功能，能够在服务端对其他系统网站发送请求，对返回的DOM数据进行操作，由此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 语言实现的将数据可视化的工具库，她能够通过代码实现对数据进行可视化操作，并且轻量，扩展性高，具有高度个性定制化的特点。

由此该系统的技术栈，其最大的特点是是全部由JavaScript进行操作，具有统一，明晰，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。

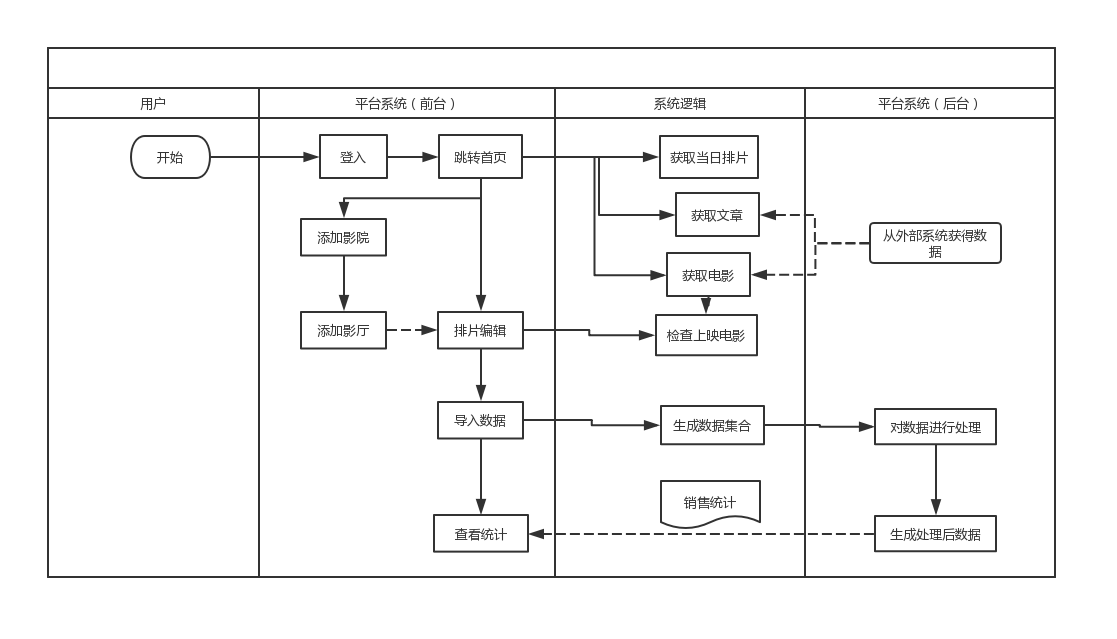
其中系统由本系统后台进行负责，完成对数据的收集以及处理。

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。主要是对影院的信息进行操作，在此之上也能够对影院所包括的影厅进行操作。
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 系统业务流程图

本系统是单用户系统，其系统业务流程图如图3-1



3-1 业务流程图

3.3 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。

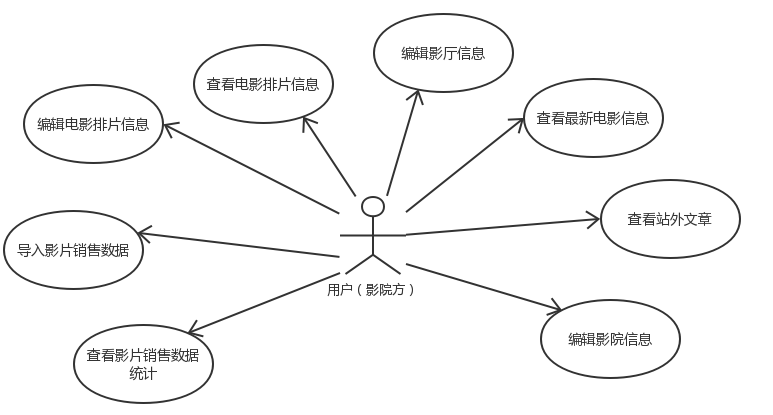


图3-2 用户用例图

# 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。

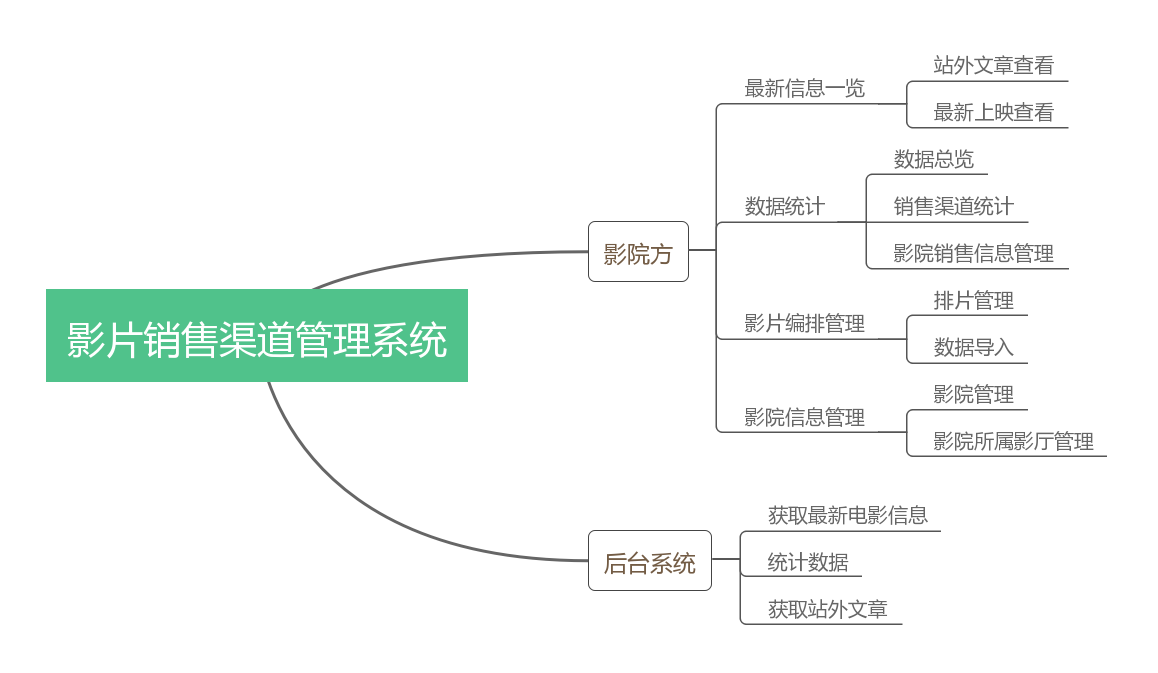


图4-1 影片销售渠道管理系统结构图

## 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)，它们分别对应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不用在存储之前为存储的数据建立字段，可以随时存储自定义的数据格式。因此，NoSQL数据库的读写性能都特别强，尤其是在大量数据下的表现格外优秀。这是因为她的无关系型，数据库的结构就会变得简单，不像Mysql等关系型数据库每次一张表的更新会对其他表产生一定的影响并且导致缓存cache失效，导致性能效率降低。而NoSQL的Cache是以日志进行记录，相当优雅轻量简洁。

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

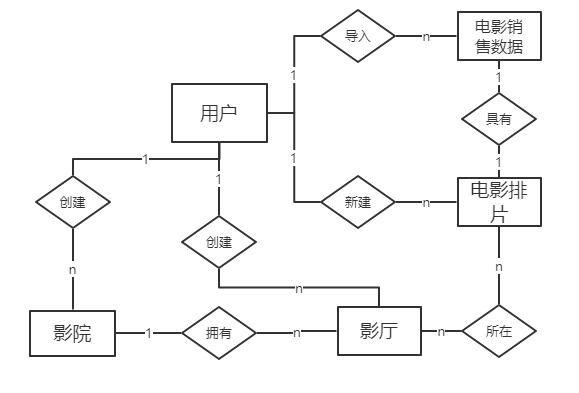
在用户数据数据库中，其中有影院信息、影厅信息、电影编排信息、电影销售导入信息，这一些集合是用户会直接操作到的。其中该数据库的总E-R图如图4-2-2

图4-2-1 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4-2-2

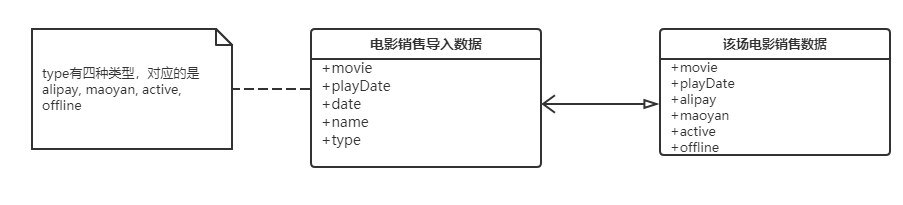


图4-2-3

电影销售导入数据集合中有电影（movie）、播放时间（playDate）、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的外键。

为了避免NoSQL的短板关联搜索的反复使用，会将一些数据未来会需要关联的信息提前写入文档，以电影为例，它的文档字段如下

{

"\_id":"5ccefd6324010ae2b07ed9c4", // \_id mongodb每条文档都具有的主键

"hall":"test12", // 影厅

"mname":"调音师 Andhadhun", // 电影名称

"ptime":"2019-05-12T15:12:32.577Z", //播放时间

"cinema":"昆明尉苏影院", // 播放影院

"status":"等待上映" // 状态

}

该文档字段中的hall（影厅），cinema（影院）是在保存该条数据时为了能够避免关联搜索就直接将播放地点信息写入该文档。

## 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。

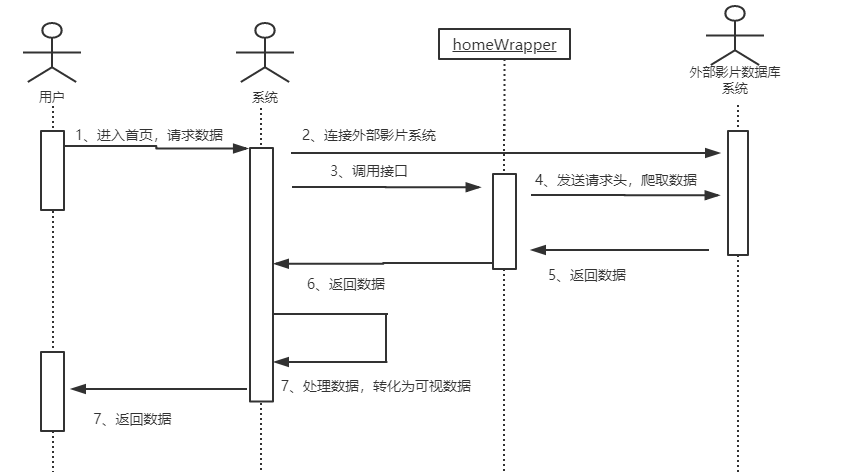


图4-3-1 管理员审核的顺序图

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。

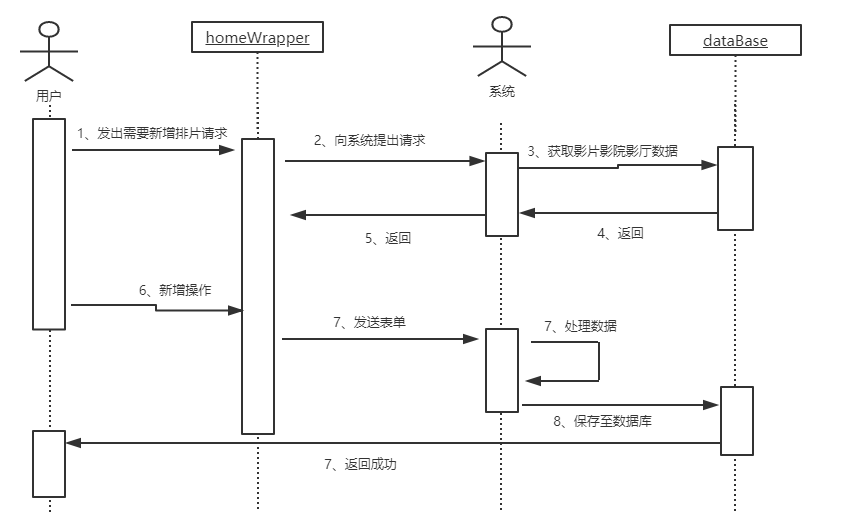


图4-3-2 新增排片顺序图

# 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。

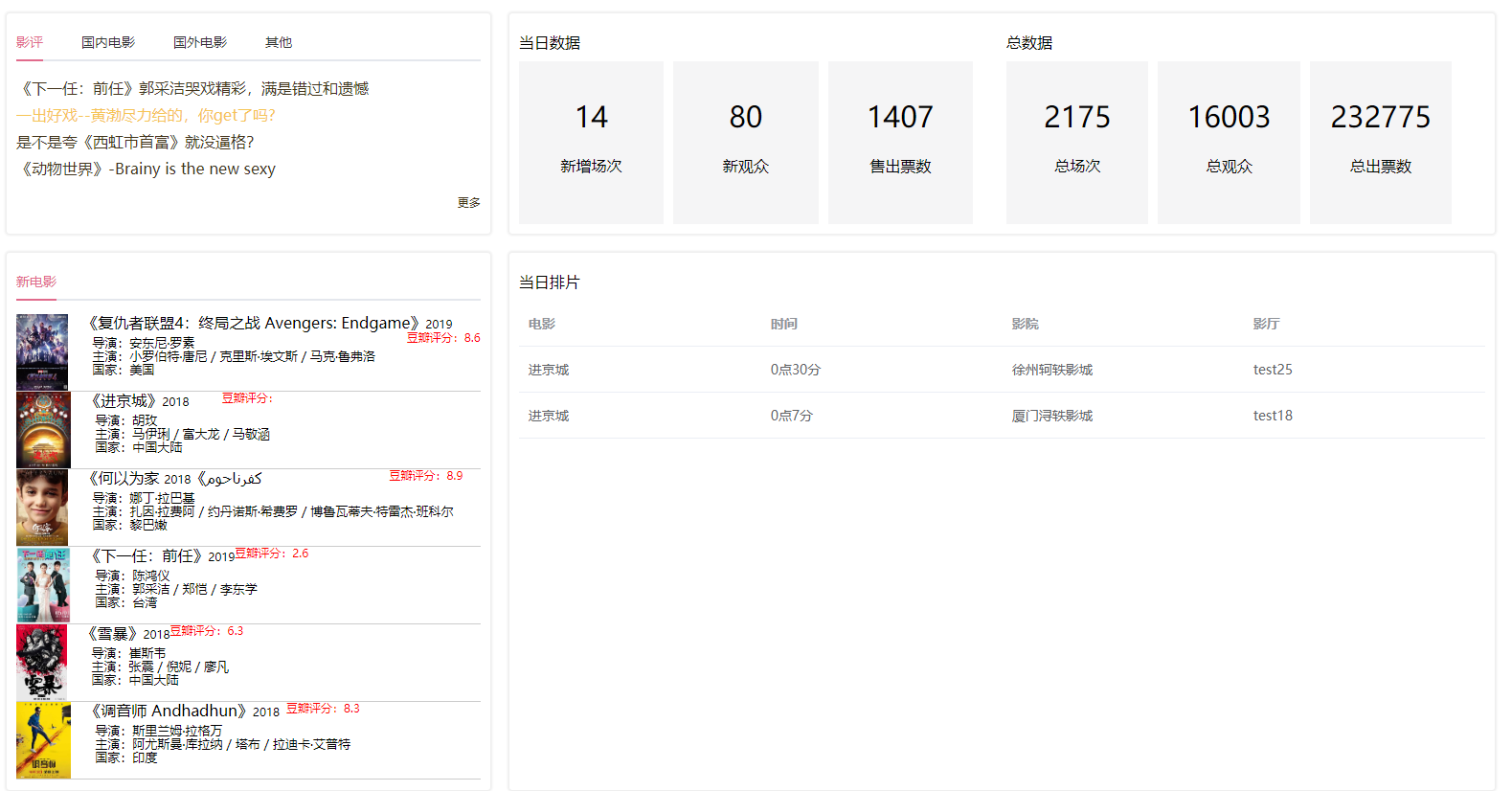


图5-1 影院销售总览主要界面

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。

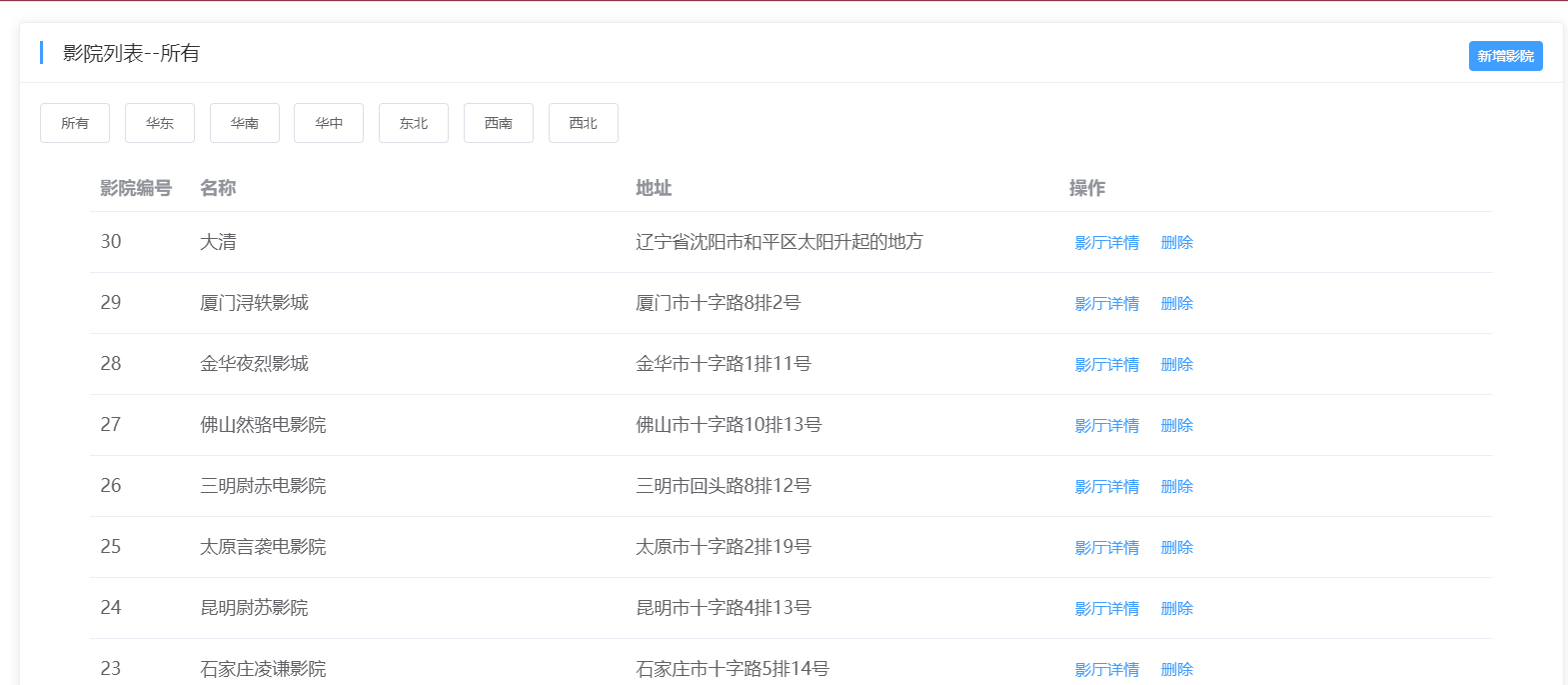


图5-2 影院管理主要界面

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，从后台获取第一级选择器的选择数据，第二级选择器的选项数据会根据第一级选择值的改变再进行动态更新，第三级选择器的选项数据再根据第二级选择值的改变再做动态更新。

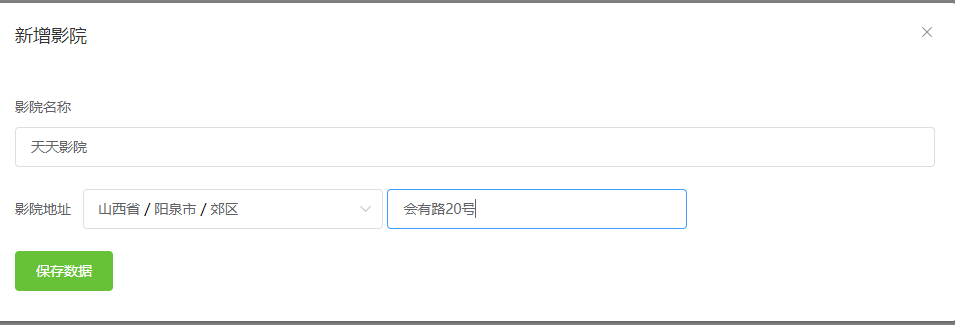


图5-3 增加影院主要界面



图5-4 影厅详情主要界面

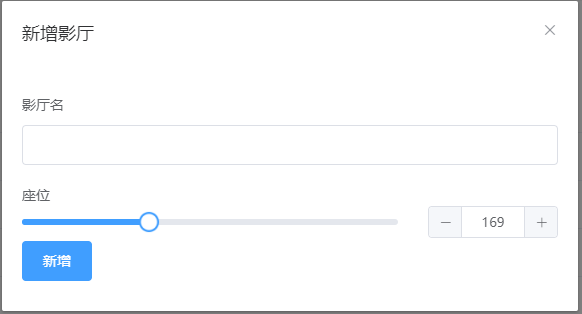


图5-5 新增影厅主要界面

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

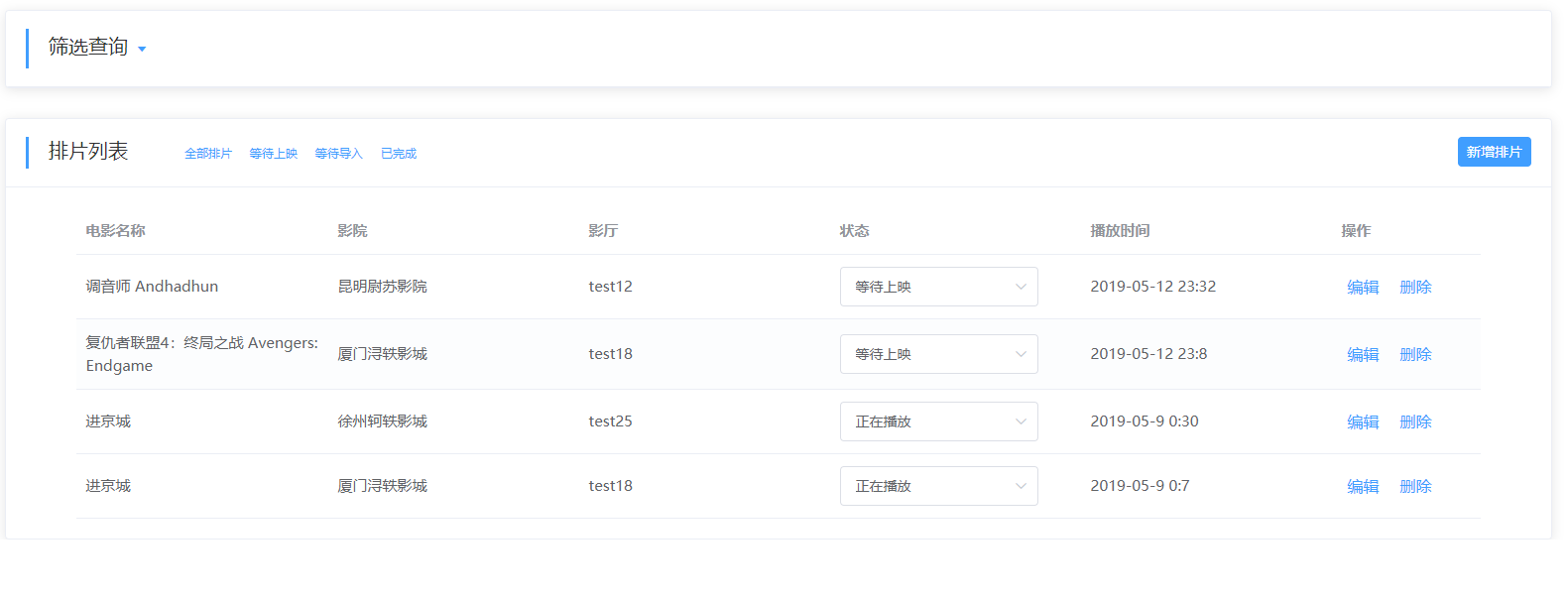


图5-6 排片管理主要界面

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。

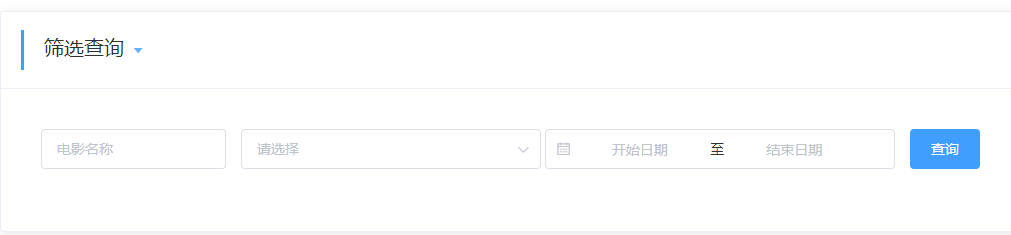


图5-7 筛选界面



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的第三方库autocomplete，她能在当用户激活输入框时弹出输入建议，。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)

},

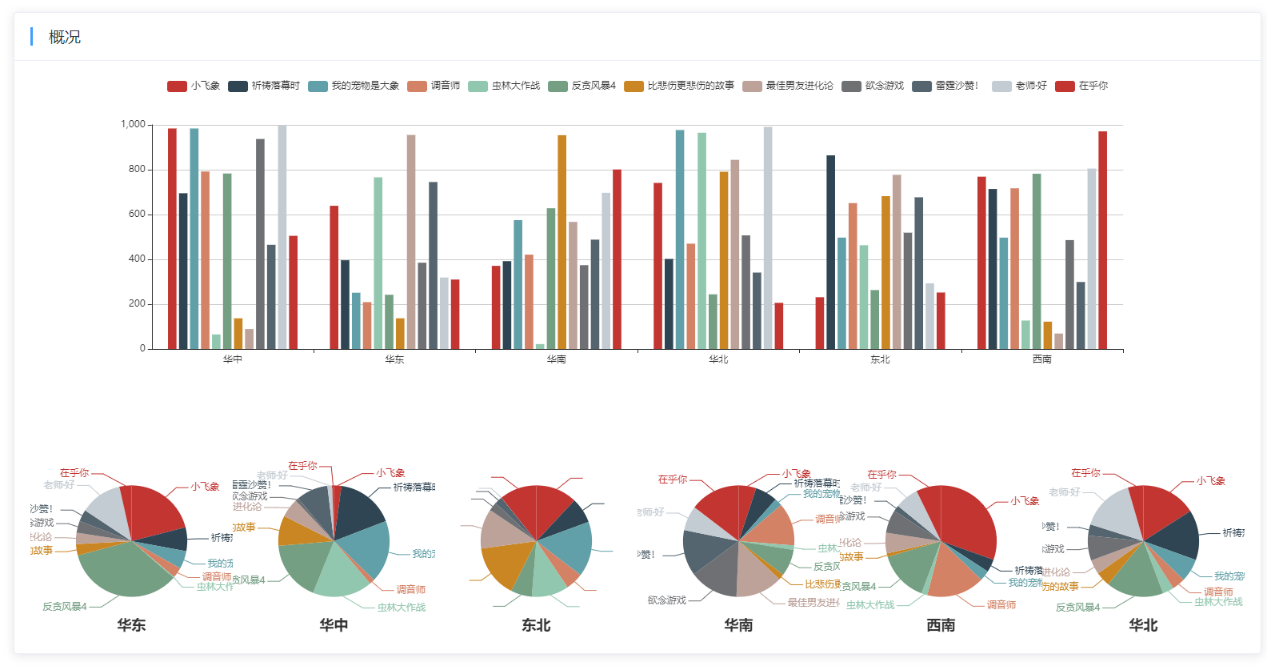
并且对电影进行编排的同时会对当前时间进行判断，若排片时间设定在当前时间以前，则会拒绝排片。且在用户进入该页面时也会自动判断修改电影播放状态。在结束放映后，会自动将状态转变为已结束并可以导入销售数据。导入销售数据界面如图5-9，其中只能导入格式为json的文件，并且导入格式不正确或数据异常则会将拒绝该数据导入。



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





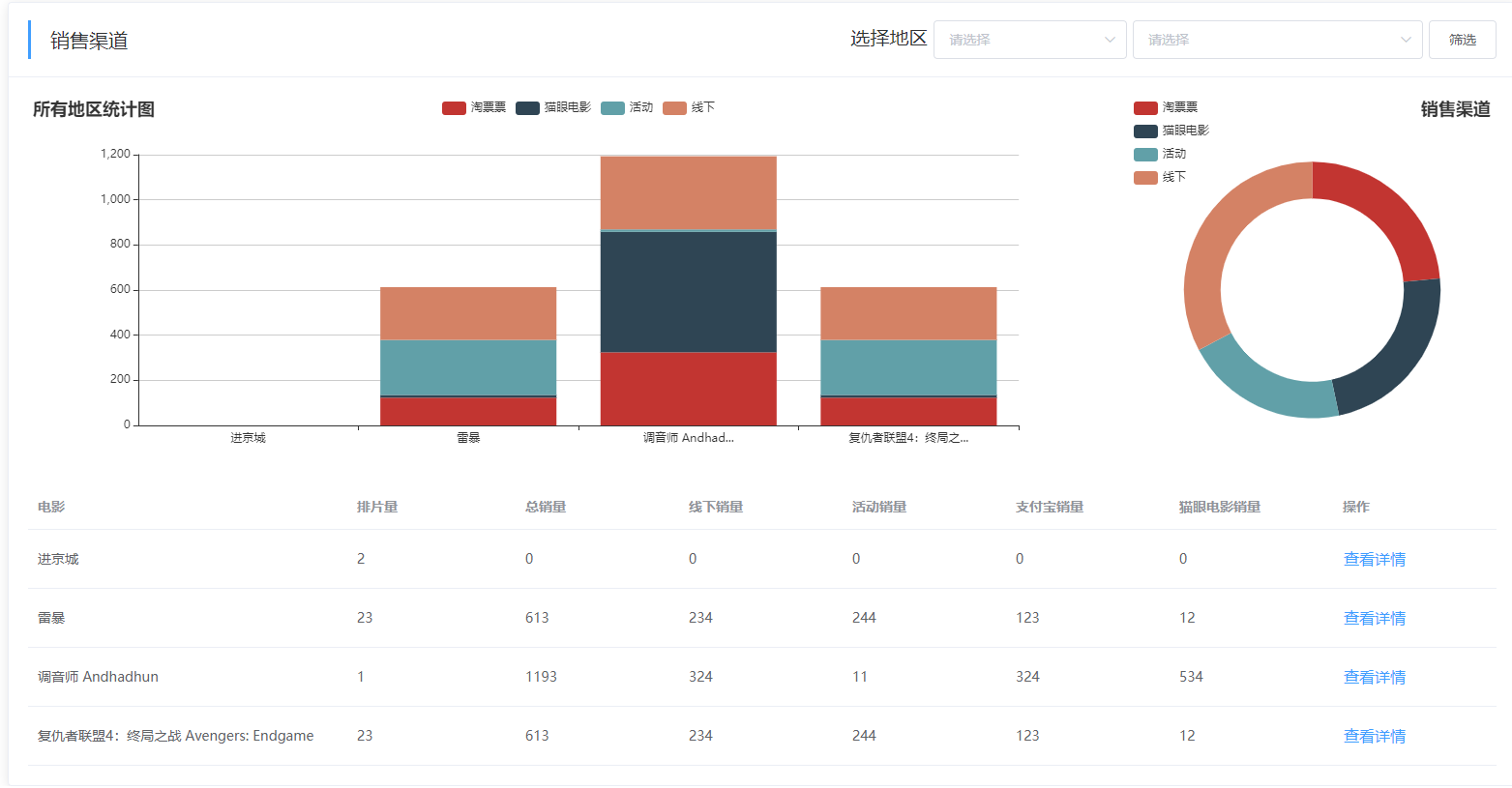


图5-10 数据统计展示界面

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。

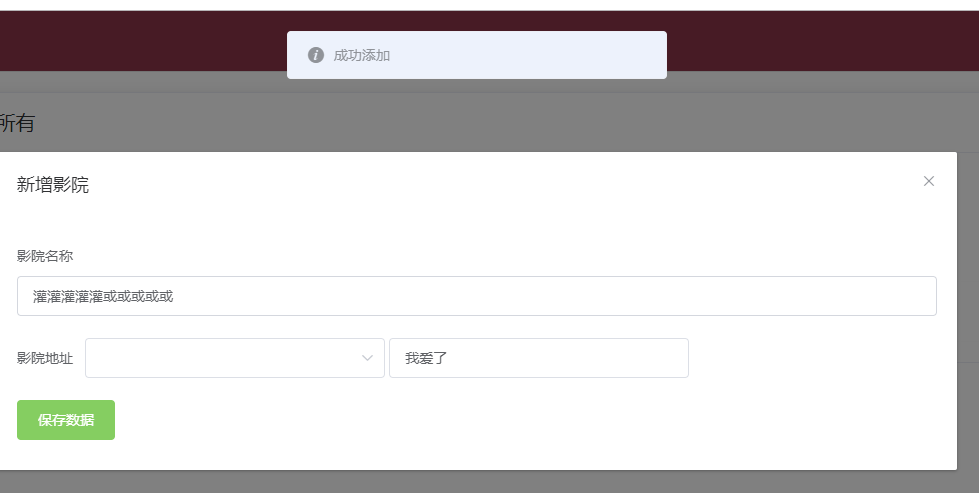


图6-1 用户注册测试图

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。

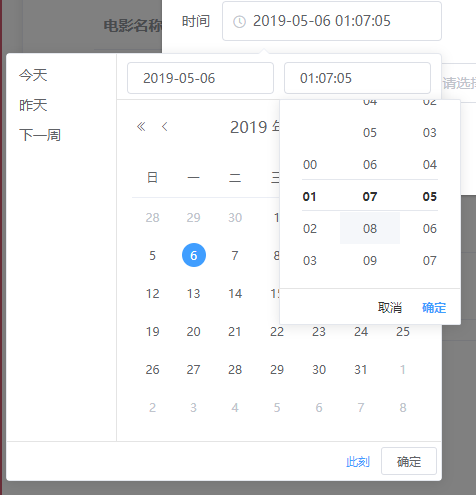




图6-4 时间选择器 图6-5

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

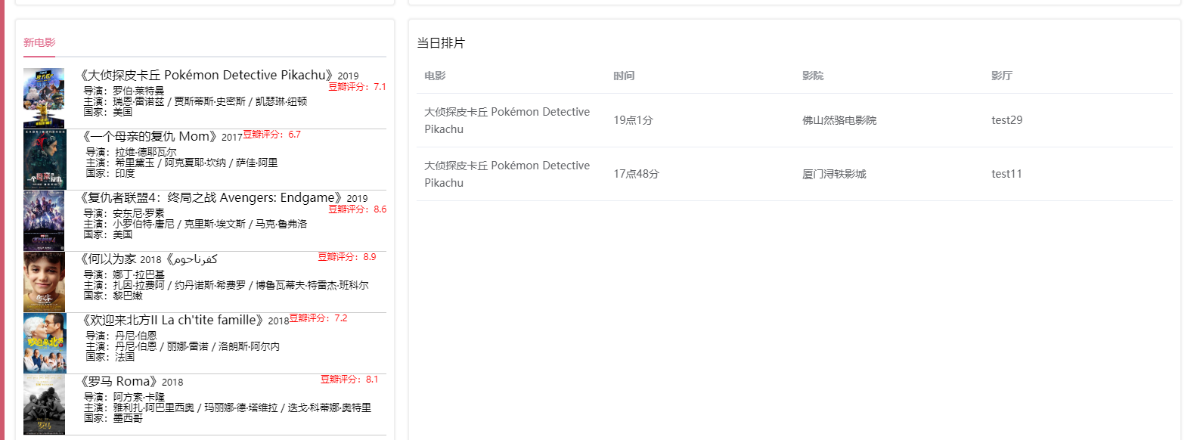


图6-7 添加当天数据后的首页

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

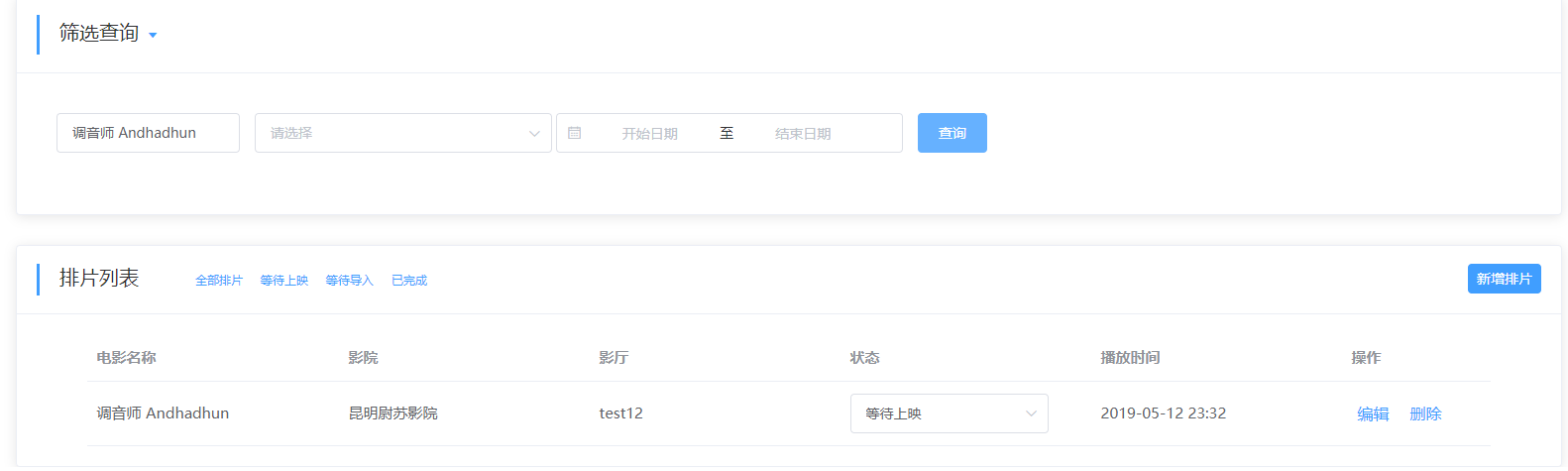


图6-8 筛选数据结果

6.3 导入数据模块

导入数据以电影《大侦探皮卡丘》为例，当添加排片时，统计数据模块中会出现该电影的排场信息，但此时它的销售数据为0，如图6-9所示。

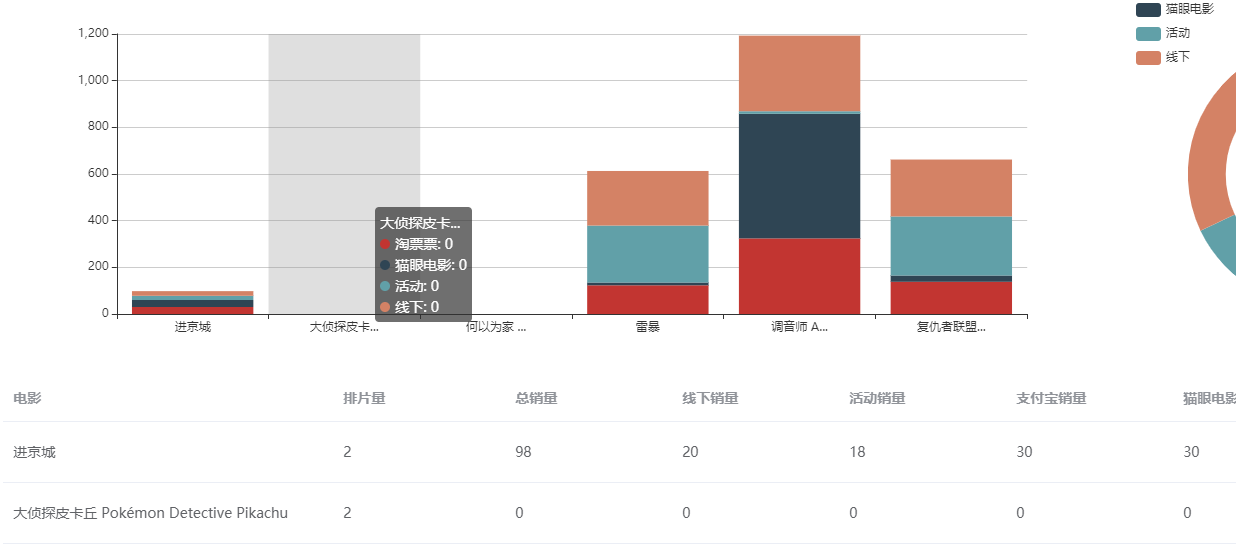


图6-9 未导入销售数据时

在导入数据模块中，如果添加的不是Json格式文件则会弹出错误窗口，如图6-10



图6-10 导入非Json文件时跳出提示框

导入的Json文件中格式应为数组，且数组中每个对象应该包含有type, name, seat等值应该为真的属性，当导入的Json格式不对时，则会提示格式错误信息，如图6-11所示，并且导入数据失败，需要重新导入



图6-11 导入Json文件中属性错误

导入数据成功后，并且后台处理数据结束后，会返回成功提示，并且可以在数据统计界面中看到该影院的销售数据统计，对电影《大侦探皮卡丘》进行销售数据导入后，所产生的数据变动如图6-12

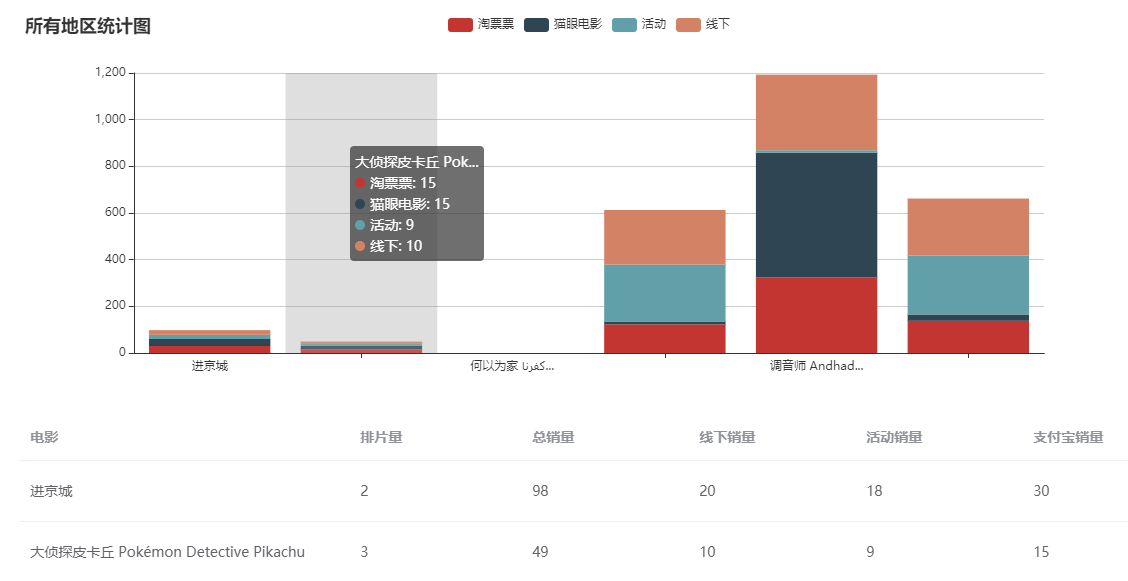


图6-13

# 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。