天津工业大学计算机科学与技术学院

实训总结报告



　项目名称: 高数----您的高级数据信息服务平台

　合作培养企业：天津市融创软通科技股份有限公司

　专业方向：大数据方向

　项目负责人：何晶

　专业班级: 软件Z1802班

姓名：段成源

　实训起止日期：2020.6.15-2020.7.15

## 1 引言

#### 1.1项目概述

###### 1.1.1项目目的

通过本次实习，主要目标是掌握关于基于Python的网络爬虫技术，同时尝试践行前后端分离的开发模式，学习必要的前端数据表现手段，充分理解Ajax、JSON在现代项目开发条环境下的主要应用方法。

另外，经过本次实训的锤炼，能够更加扎实过去学习过的相关知识，提升对于开发技术的熟练程度。

**技术方向：**

* 掌握使用各种开发工具、数据库设计工具、系统设计工具、项目管理工具和缺陷管理工具；熟练使用常用服务器软件安装、配置和开发；
* 养成良好的表达、沟通和团队协作能力、掌握快速学习方法 培养良好的分析问题和解决问题的能力。
* 熟练掌握jQuery 、Layui常用页面框架的使用；
* 熟练掌握MyBatis框架的使用；
* 掌握Scrapy爬虫框架的使用；
* 掌握eCharts图表库的使用；
* 熟练进行MySQL数据的使用以及数据库表设计；
* 掌握前台页面JS的使用；
* 利用Git版本控制软件进行小组内部的分模块开发并进行功能整合实践；
* 培养较强的自学能力、极佳的职业习惯和调试系统Bug的能力，从而熟练软件开发的过程和规范。
* 培养代码编写能力，力争每人达到2k以上的代码量（含页面代码）；

**职业素养方向：**

* 了解企业的文化和制度、熟悉企业的工作流程和工作方式；
* 掌握实际项目分析、设计、开发、测试、提交等完整流程，熟悉项目开发各类文档模板，并按照这些模板撰写项目文档；

###### **1.1.2项目背景**

身处大数据时代，各类网站如豆瓣、哔哩哔哩、当当等作为重要的信息集散地，每天都会发布大量的评论，尤其是针对电影、电视剧、书籍、音乐，均有非常有价值的讨论以及专业的评论，相关信息量较大。

本项目采用了前后端分离的开发方式，前段采用Vue框架进行开发，后端使用SpringBoot框架开发，通过scrapy爬虫机制针对豆瓣电影、音乐、图书和当当图书、B站动漫评论进行数据收集并分析，结果保存与远程mariaDB数据库中。提供简洁明了的数据查询服务，便于查询所需条件的数据，也提供关于数据内容的分析服务，并将其以图表的方式简洁直观的展示出来，方便用户观看。用户可根据分析所获结果，快捷的寻找满意的电影、音乐、书籍等资源，此外团队远程开发利用代码托管平台gitee和代码版本控制器git。

###### **1.1.3项目意义**

把理论和实践相结合的理念得到深化。通过实训，我们更好了解与我们以后工作相对应的工作。全面认知所从事行业、岗位需要具备什么才能，自己做得怎样，是否符合自己的兴趣。如果符合，自己将从哪几方面努力。为自己将来的职业生涯规划作好承前启后的准备。如不符合，分析原因，是否可以通过实践改变，慢慢培养兴趣，如实在失之甚远，可以通过此次实训收获其他方面的所得，比如人际关系的处理能力，工作能力的提升，心理承受能力的加强，这些东西对今后的发展有着比较重要的影响。

通过实训，可以全面了解自己解决问题的能力，自己的性格之中有那些需要改进的地方，知道自己适合什么行业发展。怎样培养独立办事和团体协作的能力，做到二者的辩证统一，以便适应今后工作的需要。

###### **1.1.4参考资料**

授课内容：

Python3，idea2019，git等工具包；

python基础及相关ppt参考资料；

scrapy框架及讲解录屏；

spring boot框架及讲解录屏；

mybatis框架（数据接口）及讲解录屏；

layui基础及讲解录屏；

echarts讲解录屏

提供demo：

豆瓣电影top250scrapy框架python码参考；

数据接口java码参考；

Layui和echarts码参考；

#### **1.2 需求分析**

###### **1.2.1功能需求**

随着信息技术的发展，各类娱乐、学习等网站层出不穷，豆瓣、哔哩哔哩、当当等各类网站呈现给人们纷繁的数据信息。为了给用户呈现直观的网站数据，本项目旨在通过建立用户管理系统，控制相关网站爬取数据的走向（数据量>10000），并将分析图表展示给用户，同时建立搜索查找功能，便于用户搜索、查找想要的数据信息。

###### **1.2.2性能需求**

**数据分析精确度：**考虑到本项目，因此对于数据可视化的部分精确度无较高要求，只需能体现相对占比等数据特征即可。而对于回头客的判断则需要较高的预测准确率，在这里可以拿2019年的数据进行检验来判断预测。

**时间特性：**

响应时间：图片显示时间在0~500ms以内。

更新处理时间：本项目的数据在短期内数据分析可靠度较高，无需实时获取，但需定期更新。同时考虑到用户体验可考虑增加实时更新的功能。

数据转换与传输时间：数据转换主要是涉及到数据库的问题。为了降低一次显示过多的记录会消耗太多的时间这个问题，我们在设计时将让数据库尽量分页显示。传输时间的问题比较复杂，这和服务器的处理能力，电信部门给我们提供的带宽，银行的电子交易网的响应速度有关。

运行时间：本网站如非特殊原因应保持24小时开通。

**适应性：**这里主要是解决用户使用何种设备/系统进行浏览的问题，该项目对于任何设备均适应性良好。

###### **1.2.3非功能性需求**

* 扩展性和可维护性

1） 可扩展性方面，有较强的拓展空间。

① 可考虑将使用平台延伸到终端和手机端。终端安装可为受众群体在公众场合使用提供免费平台；网页手机可以直接打开，适合各场景下、不同人群的观看习惯。

2） 可维护性：

① 维护人员理解网页的结构、接口、功能和内部过程的难度不大，即可理解性强；

② 测试和诊断软件错误的难度不大，即可测试性强

③ 软件需要调整时，修改网页的难度中等，即有较好的可修改性；

④ 网页功能的模块化、信息隐蔽性、高内聚、低耦合等性质，都对于提高网页的可理解性、 可维护性和可修改性有很重要的作用。

* 易用性

直接使用网页链接打开，不需要下载任何插件，轻松简单。

（1）所有内容都以所有人能理解的格式来呈现。

（2）网页里的功能按钮接口组件对任何人来说都可操控。

（3）系统可以满足有经验和无经验的用户，允许用户进行频繁或缓慢的操作。

###### 1.2.4接口需求

硬件接口：可拓展，暂无。

软件接口：错误信息处理接口、登陆注册信息传输接口、数据查询分析传输接口、数据库信息传输接口

###### 1.2.5出错处理需求

错误/异常信息对我们程序的调试 、完善有着重要的指导意义，但是我们不希望用户看到的错误/异常信息，所以我们认为，异常应该在控制/返回消息转发之前尽量处理完毕，同时发生异常的模块记录 log 日志，以便于我们开发者的进一步调试和改进。

**数据层**：数据层部分的异常信息主要是数据类型、数据范围以及约束条件等。该部分的异常信息在一定程度上需要返回给接口调用者，以便于指导用户的数据修改和完善，我们在捕获异常信息之后会对异常类型进行标注处理，做为指导性更正建议。

###### 1.2.6其他需求

**可使用性：**

用户操作简单，体验好，数据展现直观易懂。

**可移植性：**

无需下载通过浏览器访问网页即可使用，可移植性高。

① **PC 端浏览器**：应用可直接在 PC 端浏览器打开。支持浏览器：IE9、Chrome7、360 浏览器、火狐浏览器、UC 浏览器等主流浏览器。

②**移动端浏览器**：应用可直接在移动端端浏览器打开。支持浏览器：safari 浏览器、无线 Mozilla、Chrome、百度浏览器、小米浏览器、UC 浏览器等主流浏览器。

#### **1.3 运行环境**

###### 1.3.1硬件环境

1.笔记本1台

2.内存：4GB以上。

4.局域网畅通

###### 1.3.2软件环境

1. 操作系统：Windows 10。

2. 文档编写工具：Microsoft office 2016（word excel powerpoint visio）。

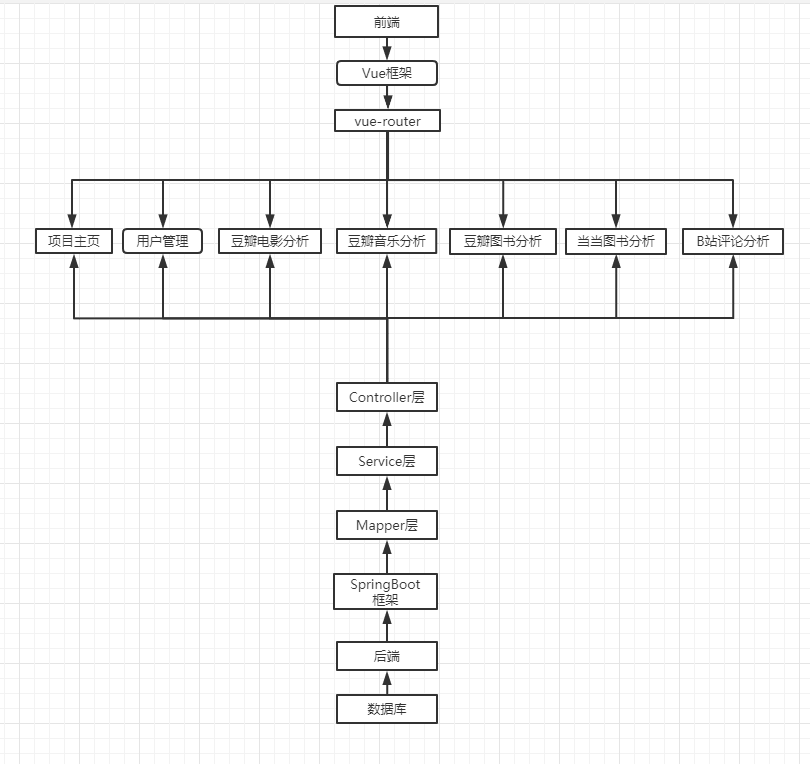
3. 开发工具：IDEA2019，PyCharm2019，MySQL5.7.x，WebStorm2019、Navicat、ECharts。

4. 版本控制工具：Git/码云。

## 2 项目设计

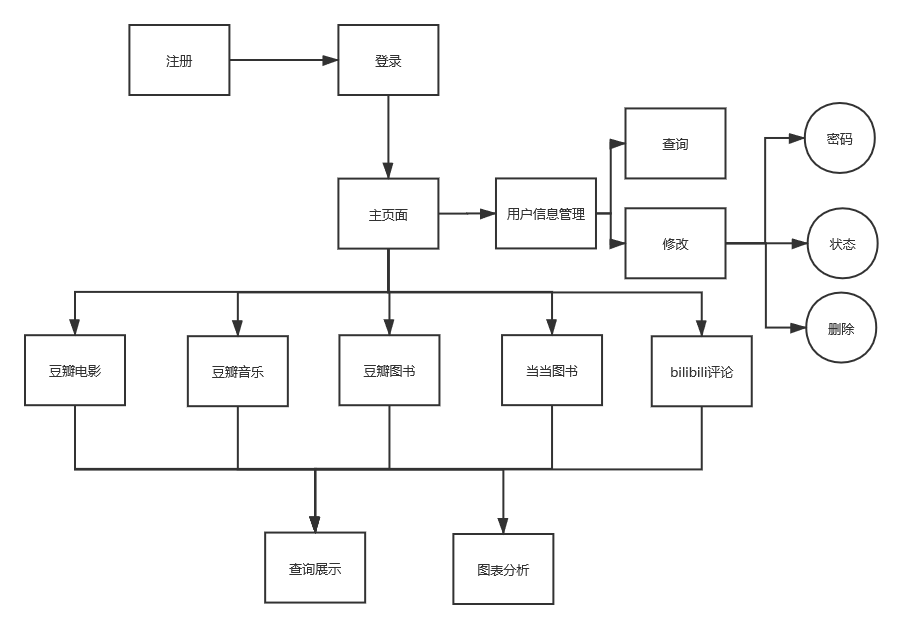
#### **2.1 设计思路**

###### 2.1.1总体设计思路



系统大致可以分为三层：Vue前端、SSM后端、以及MySql数据库三层。Mysql数据库存储通过scrapy爬取到的数据，用以进行后端分析和前端展示；SSM后端主要有三层：Controller层（控制层，负责具体模块的业务流程控制，需要调用service逻辑设计层的接口来控制业务流程）、Mapper层（对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，主要实现一些增删改查操作）、Service层（业务service层，给controller层的类提供接口进行调用。），由此向前端提供接口，完成交互；Vue前端通过接口获得相应数据，完成展示。

###### 2.1.2总体程序流程图



#### **2.2 模块功能介绍**

###### 2.2.1豆瓣电影数据爬取

1) 允许平台用户选取不同的类别后填写需要抓取的关键字，系统会根据关键字进行平台（豆瓣）检索，并将检索后的结果反馈给当前用户。

2) 将相关详情进行入库操作并建立与当前用户的关联，形成该用户的在该分类下的检索历史。

3) 检索历史管理功能，可分类查看检索历史，并删除相关检索历史条目，同时删除该条目对应的一切信息（评论、分析汇总等）

4) 允许平台用户在本平台为抓取的项目评分，编写评论内容，用于生成该平台影片库的评论内容。

###### 2.2.2豆瓣电影数据查询

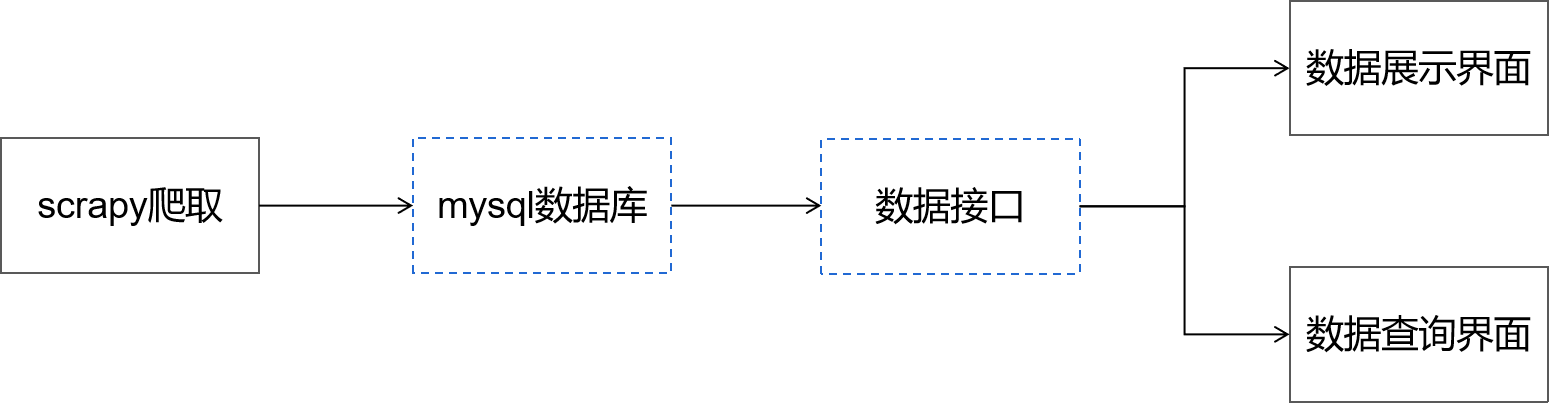
1. 针对豆瓣影片库，爬取电影相关数据字段，同时选取电影名、评分、主演、导演、类型等关键字段展示给用户
2. 允许平台用户在豆瓣影片查询界面通过查询关键字段：电影名、导演、主演根据需求检索相关电影数据。

###### 2.2.3豆瓣电影数据展示

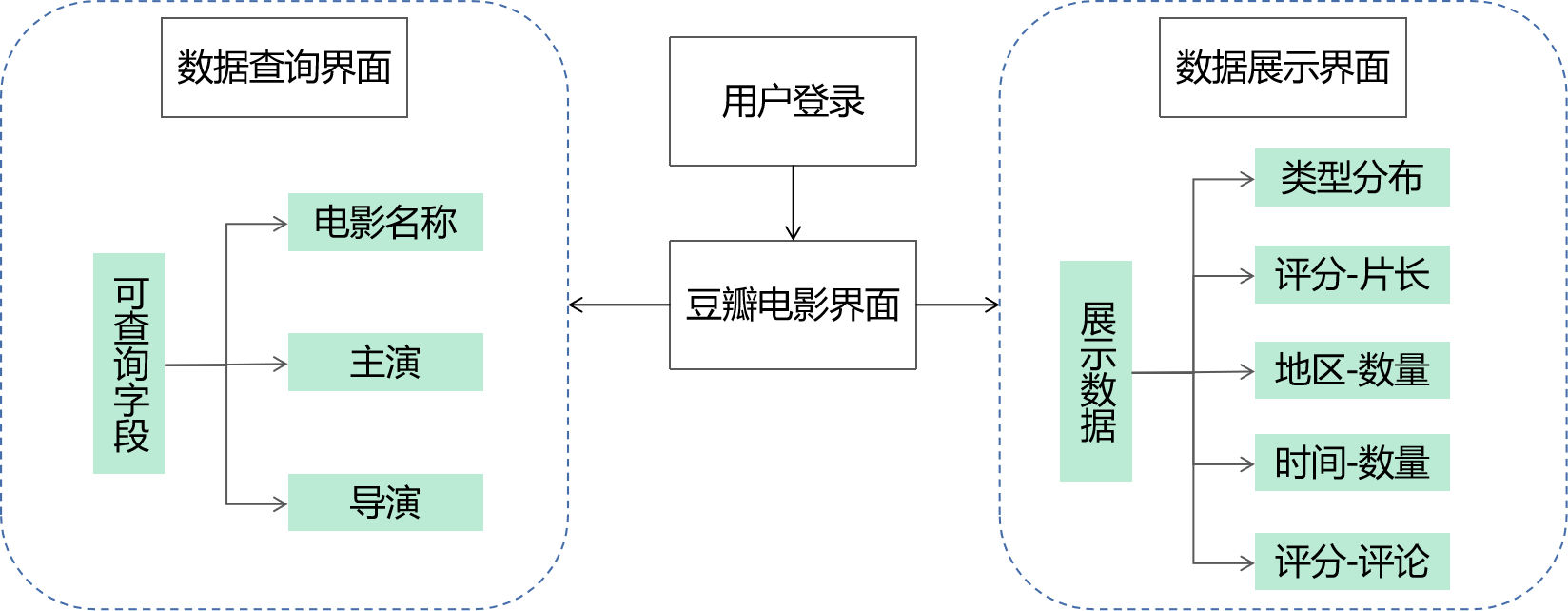
根据爬取来的豆瓣影片数据，主要做如下数据展示：

1. 数据信息点：电影总数、总评论数、平均片长、最高评分、最低评分
2. 各电影类型占比情况-饼图
3. 不同地区产电影数量的比较-柱状图
4. 随时间产生电影数量的变化曲线-折线图
5. 平常与评分的关系-散点图
6. 评论人数与评分的关系-折线图

#### **2.3 模块结构图**



#### **2.4 程序流程图**



#### **2.5 功能设计分工**

|  |  |
| --- | --- |
| **组员** | **分工** |
| 廖嘉伟 | 豆瓣图书数据搜索与展示前端部分与后端部分及后端项目框架搭建 |
| 段成源 | 豆瓣电影数据搜索与展示前端部分与后端部分 |
| 刘博 | 豆瓣音乐数据搜索与展示前端部分与后端部分 |
| 薛彦涵 | 登录、当当图书数据搜索与展示前端部分与后端部分 |
| 李嘉威 | 用户管理、项目主页、注册、B站评论数据搜索与展示前端部分与后端部分 |

**3 详细设计**

#### **3.1系统设计**

**远程Mariadb数据库，由scrapy爬取数据后存入其中，主要负责**：

user表：用户表，包含用户的用户名、密码、昵称、邮箱、地址、年龄、性别、状态

影片数据表（bdcy\_douban\_mivie）：电影名、评分、评论人数、1-5星占比、短评数量、影评数量、电影类型、导演、编剧、主演、国家/地区、语言、上映时间、片长

**scrapy爬虫，包括：**

Scheduler(调度器)：它负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。

Downloader（下载器）：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。

Spider（爬虫）：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。

Item Pipeline(管道)：它负责处理Spider中获取到的Item，并进行后期处理（详细分析、过滤、存储等）的地方。

Downloader Middlewares（下载中间件）：一个可以自定义扩展下载功能的组件。

Spider Middlewares（Spider中间件）：一个可以自定扩展和操作引擎和Spider中间通信的功能组件。

**SSM后端：**

Controller层（控制层，负责具体模块的业务流程控制，需要调用service逻辑设计层的接口来控制业务流程）

Mapper层（对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，主要实现一些增删改查操作）

Service层（业务service层，给controller层的类提供接口进行调用。）

主要包括：用户管理所需接口，包括登录、注册、修改用户信息；豆瓣音乐信息，包括带有关键字（音乐名、歌手、时间、标签、流派）的查询，可视化数据分析

**VUE前端：**

前端框架，主要负责豆瓣音乐关键字查询、可视化图表分析界面

#### **3.2数据库设计**

###### 3.2.1逻辑模型

影片数据表（bdcy\_douban\_mivie）：电影名、评分、评论人数、1-5星占比、短评数量、影评数量、电影类型、导演、编剧、主演、国家/地区、语言、上映时间、片长

###### 3.2.2物理模型

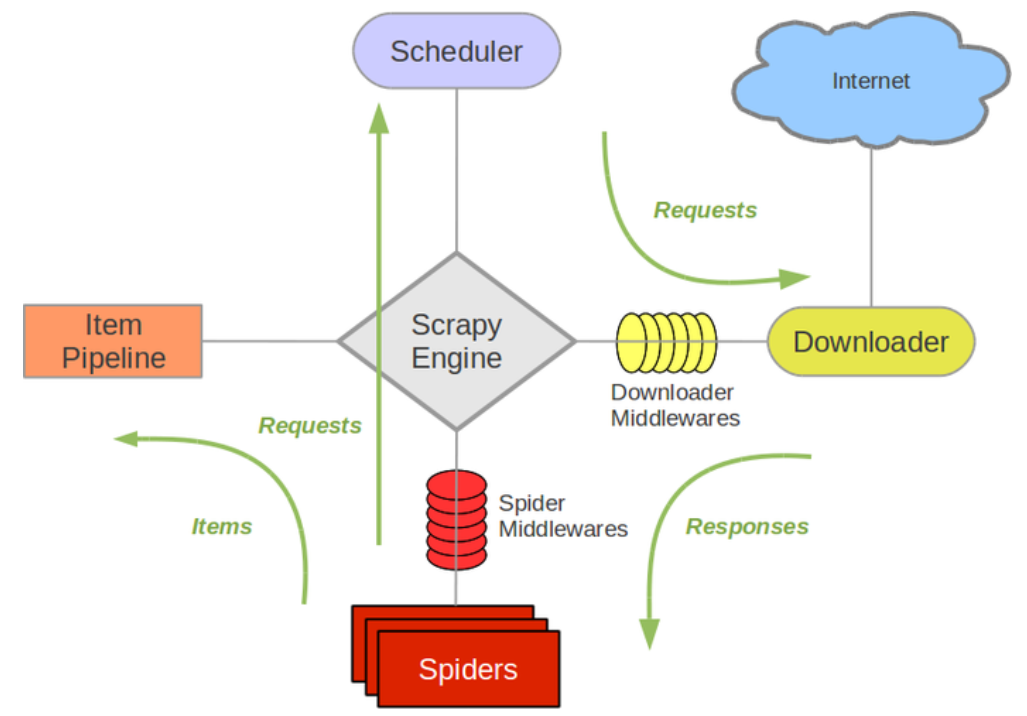
豆瓣电影数据库：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 字段含义 | 类型 | 长度 | 允许为空 | 主键 | 说明 |
| 1 | id | 电影序列 | int | 11 | 否 | 是 |  |
| 2 | movie\_name | 电影名 | varchar | 255 | 否 | 否 |  |
| 3 | score | 电影评分 | float | 255 | 是 | 否 |  |
| 4 | people\_count | 评论人数 | int | 11 | 是 | 否 |  |
| 5 | five\_star | 五星占比 | float | 255 | 是 | 否 |  |
| 6 | four\_star | 四星占比 | float | 255 | 是 | 否 |  |
| 7 | three\_star | 三星占比 | float | 255 | 是 | 否 |  |
| 8 | two\_star | 两星占比 | float | 255 | 是 | 否 |  |
| 9 | one\_star | 一星占比 | float | 255 | 是 | 否 |  |
| 10 | short\_review\_count | 短评数量 | int | 11 | 是 | 否 |  |
| 11 | review\_count | 评论数量 | int | 11 | 是 | 否 |  |
| 12 | movie\_type | 电影类型 | varchar | 255 | 是 | 否 |  |
| 13 | move\_director | 导演 | varchar | 255 | 是 | 否 |  |
| 14 | movie\_script | 编剧 | varchar | 255 | 是 | 否 |  |
| 15 | movie\_role | 主演 | varchar | 255 | 是 | 否 |  |
| 16 | movie\_country | 地区 | varchar | 255 | 是 | 否 |  |
| 17 | movie\_language | 语言 | varchar | 255 | 是 | 否 |  |
| 18 | movie\_start\_time | 上映时间 | varchar | 255 | 是 | 否 |  |
| 19 | movie\_length | 片长 | int | 11 | 是 | 否 |  |
| 20 | movie\_url | 链接 | varchar | 255 | 是 | 否 |  |

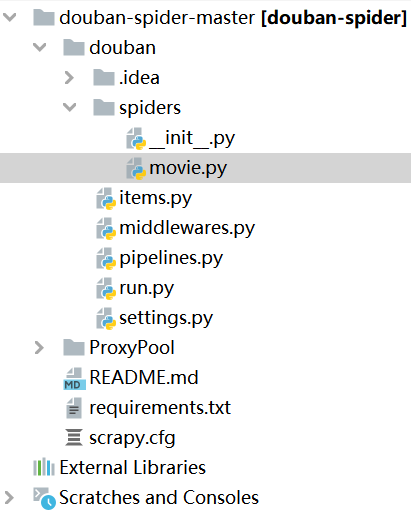
**4 实现方案**

#### **4.1爬取豆瓣电影数据功能实现**

###### 4.1.1 scrapy爬取框架



###### 4.1.2 项目部署目录结构

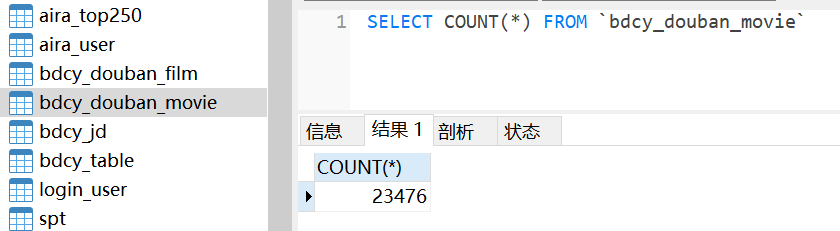


###### 4.1.3 核心代码展示

|  |
| --- |
| **pipelines.py** |
| *# -\*- coding: utf-8 -\*- # Define your item pipelines here # Don't forget to add your pipeline to the ITEM\_PIPELINES setting # See: http://doc.scrapy.org/en/latest/topics/item-pipeline.html* **import** pymongo **from** douban.items **import** \*  **class** DoubanPipeline(object):  **def** process\_item(self, item, spider):  **return** item  **class** MongoPipeline(object):  **def** \_\_init\_\_(self, mongo\_uri, mongo\_db):  self.mongo\_uri = **'mongodb://localhost:27017/'** self.mongo\_db = **'douban'** @classmethod  **def** from\_crawler(cls, crawler):  **return** cls(  mongo\_uri=crawler.settings.get(**'MONGO\_URI'**),  mongo\_db=crawler.settings.get(**'MONGO\_DATABASE'**)  )   **def** open\_spider(self, spider):  self.client = pymongo.MongoClient(self.mongo\_uri)  self.db = self.client[self.mongo\_db]  self.db[DoubanItem.collection].create\_index([(**'id'**, pymongo.ASCENDING)])   **def** close\_spider(self, spider):  self.client.close()   **def** process\_item(self, item, spider):  **if** isinstance(item,DoubanItem):  self.db[item.collection].update({**"id"**: item.get(**'id'**)}, {**'$set'**: item}, **True**)  **return** item |

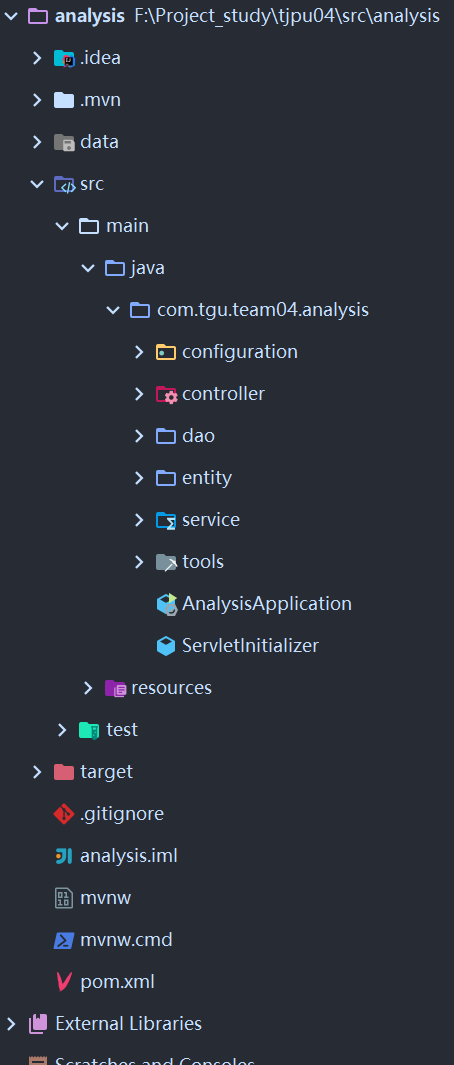
|  |
| --- |
| **movie.py** |
| *# -\*- coding: utf-8 -\*-* **import** scrapy,json **from** scrapy **import** Request, Selector **from** douban.items **import** \* **from** urllib **import** parse **from** pymongo **import** MongoClient **import** logging  *# mongo数据库* client = MongoClient() db = client[**'douban1'**] collection = db[**'film'**]  **class** MovieSpider(scrapy.Spider):  name = **"movie"** allowed\_domains = [**"movie.douban.com"**]  film\_listurl = **'https://movie.douban.com/j/new\_search\_subjects?sort={sorttype}&range=0,10&tags=%E7%94%B5%E5%BD%B1&start={page}&countries={country}'** film\_url = **'https://movie.douban.com/subject/{id}/'**countrylist = [**'中国大陆'**,**'香港'**,**'台湾'**]sortlist = [**'U'**,**'S'**,**'R'**,**'T'**]   **def** start\_requests(self):  **for** sorttype **in** self.sortlist:  **for** country **in** self.countrylist:  country = parse.quote(country)  **for** i **in** range(0,500):  **yield** Request(self.film\_listurl.format(page=i \* 20, country=country, sorttype=sorttype), callback=self.parse)  **def** parse(self, response):  *"""  解析页面，获取电影id，构造电影链接* **:param** *response: Response对象* **:return***:  """* self.logger.debug(response)  result = json.loads(response.text)  **if** result.get(**'data'**):  filmlist = result.get(**'data'**)  **if** filmlist != []:  **for** film **in** filmlist:  *# 获取电影id* id = film.get(**'id'**)  title = film.get(**'title'**)  logging.getLogger(\_\_name\_\_).debug(**"已获取电影id：%s %s"** %(title,id))  *# 已经爬好的id不再爬* **if** collection.find\_one({**"id"**:id}):  logging.getLogger(\_\_name\_\_).debug(**"\*\*\*该电影已经保存！\*\*\*"**)  **else**:  **yield** Request(self.film\_url.format(id=id), callback=self.parse\_film, meta={**'id'**:id})   **def** parse\_film(self, response):  *"""  解析单部电影详情信息* **:param** *response: Response对象* **:return***:  """* selector = Selector(response=response)  *# id* id = response.meta.get(**'id'**)  *# 电影名字* title = selector.xpath(**'//span[@property="v:itemreviewed"]/text()'**).extract()[0]  *# 年份* **try**:  year = selector.xpath(**'//\*[@id="content"]/h1/span[2]/text()'**).extract()[0][1:5]  **except** IndexError:  year = **None** *# 制片国家* region = selector.xpath(**'//\*[@id="info"]'**).re(**'制片国家/地区:</span>\s(.\*)<br>'**)  *# 语言* language = selector.xpath(**'//\*[@id="info"]'**).re(**'语言:</span>\s(.\*)<br>'**)  *# 导演* director = selector.xpath(**'//\*[@rel="v:directedBy"]/text()'**).extract()  *# 类型* type = selector.xpath(**'//\*[@property="v:genre"]/text()'**).extract()  *# 演员* actor = selector.xpath(**'//\*[@rel="v:starring"]/text()'**).extract()  *# 上映日期* date = selector.xpath(**'//span[@property="v:initialReleaseDate"]/text()'**).extract()  *# 片长* runtime = selector.xpath(**'//span[@property="v:runtime"]/text()'**).extract()  *# 评分* **try**:  rate = selector.xpath(**'//strong[@property="v:average"]/text()'**).extract()[0]  **except** IndexError:  rate = **None** *# 评价人数* **try**:  rating\_num = selector.xpath(**'//span[@property="v:votes"]/text()'**).extract()[0]  **except** IndexError:  rating\_num = **None** film\_info\_item = DoubanItem()   field\_map = {  **'id'**: id, **'title'**: title, **'year'**: year, **'region'**: region, **'language'**: language,  **'director'**: director, **'type'**: type, **'actor'**: actor, **'date'**: date,  **'runtime'**: runtime, **'rate'**: rate, **'rating\_num'**: rating\_num  }   **for** field, attr **in** field\_map.items():  film\_info\_item[field] = attr   **yield** film\_info\_item |

###### 4.1.4 成果展示



#### **4.2后端数据接口功能实现**

###### 4.2.1 项目部署目录结构



###### 4.2.2 核心代码

|  |
| --- |
| **controller层：FilmController** |
| *package* com.tgu.team04.analysis.controller;  *import* com.tgu.team04.analysis.entity.\*; *import* com.tgu.team04.analysis.service.*FilmService*; *import* org.springframework.beans.factory.annotation.Autowired; *import* org.springframework.stereotype.Controller; *import* org.springframework.web.bind.annotation.RequestMapping; *import* org.springframework.web.bind.annotation.ResponseBody;  *import* java.util.ArrayList; *import* java.util.HashMap; *import* java.util.*List*; *import* java.util.*Map*;  @Controller @RequestMapping("/douban") *public class* FilmController {  @Autowired  *private FilmService* filmservice;   @RequestMapping("/filmlist")  @ResponseBody  *public* TableData filmList(Integer page, Integer limit){  TableData data = *new* TableData();  System.out.println("page: " + page + ", limit: " + limit);  *List*<FilmComment> result = filmservice.search((*int*)page, (*int*)limit);  *if* (result != *null*){  data.setCode(1000);  data.setMsg("查询成功");  data.setData(result);  }*else* {  data.setCode(2000);  data.setMsg("查询失败");  data.setData(*null*);  }  *return* data;  }   @RequestMapping("/search")  @ResponseBody  *public* TableData search(String movieName, String movieDirector, String movieRole, *int* page, *int* limit){  FilmComment filmComment = *new* FilmComment();  filmComment.setMovieName(movieName);  filmComment.setMovieDirector(movieDirector);  filmComment.setMovieRole(movieRole);   System.out.println(filmComment);  System.out.println("page: " + page + ", limit: " + limit);  TableData data = *new* TableData();  *List*<FilmComment> result = filmservice.searchByOption(filmComment, page, limit);  *if* (result != *null* && result.size() != 0){  data.setCode(1000);  data.setMsg("查询成功");  data.setCount(filmservice.searchCount(filmComment));  data.setData(result);  }*else* {  data.setCode(2000);  data.setMsg("查询失败");  data.setCount(0);  data.setData(*null*);  }  *return* data;  }   @RequestMapping("/query")  @ResponseBody  *public* TableData query(String movieName){   TableData data = *new* TableData();  FilmSummeryData filmSummeryData = filmservice.movieSummery();  *List*<FilmSummeryData> result = *new* ArrayList<>();  result.add(filmSummeryData);  *if* (result != *null* && result.size() != 0){  data.setCode(1000);  data.setMsg("查询成功");  data.setData(result);  }*else* {  data.setCode(2000);  data.setMsg("查询失败");  data.setCount(0);  data.setData(*null*);  }  *return* data;  }   @RequestMapping("/totalnumber")  @ResponseBody  *public int* totalNumber(){  *int* countTotal = filmservice.totalNumber();  *return* countTotal;  }   @RequestMapping("/movielength")  @ResponseBody  *public int* movieLength(){  *int* movielength = filmservice.movieLength();  *return* movielength;  }   @RequestMapping("/peoplenumber")  @ResponseBody  *public int* peopleNumber(){  *int* peoplenumber = filmservice.peopleNumber();  *return* peoplenumber;  }   @RequestMapping("/maxscore")  @ResponseBody  *public* TableData maxScore(){  TableData data = *new* TableData();  System.out.println(data);  *List*<Float> result = filmservice.maxScore();  System.out.println(result);  *if* (result != *null*){  data.setCode(1000);  data.setMsg("查询成功");  data.setData(result);  }*else* {  data.setCode(2000);  data.setMsg("查询失败");  data.setData(*null*);  }  *return* data;  }  @RequestMapping("/simpleAnalysis")  @ResponseBody  *public* TableData simpleAnalysis(String type){  TableData data = *new* TableData();  *List*<FilmComment> filmComments = filmservice.simpleAnalysis(type);   *if* ("score".equalsIgnoreCase(type)) {  *List*<FilmSimpleData> result = *new* ArrayList<>();  *Map*<String,Float> map=*new* HashMap<>();  *boolean* flag=*false*;  String judge=*null*;  *float* num=0;  *for* (FilmComment comment : filmComments) {  String[] arr=comment.getMovieType().split("/");  *if*(map.isEmpty()){  map.put(arr[0],comment.getScore());  }  *for*(String key:map.keySet()){  *if*(arr[0].equals(key)){  judge=key;  num=map.get(key);  flag=*true*;  }  }  *if*(flag){  map.put(judge,comment.getScore()+num);  flag=*false*;  }*else*{  map.put(arr[0],comment.getScore());  }  }  *double* count=0;  *for*(String key:map.keySet()){  FilmSimpleData filmSimpleData = *new* FilmSimpleData();  filmSimpleData.setMovieType(key);  filmSimpleData.setScore(map.get(key));  count+=map.get(key);  result.add(filmSimpleData);  }  System.out.println(count+"++++++++++++++++++++++++++++++++++++++");  *// 返回table data* data = generateTableData(result);  }  *if* ("commentRise".equalsIgnoreCase(type)) {  *List*<FilmCommentData> result = *new* ArrayList<FilmCommentData>();  *for* (FilmComment comment : filmComments) {  FilmCommentData filmCommentData = *new* FilmCommentData();  filmCommentData.setPeopleCount(comment.getPeopleCount());  *float* score = comment.getScore();  filmCommentData.setScore(score);  result.add(filmCommentData);  }  *// 返回table data* data = generateTableData(result);  }  *if* ("wordFrequency".equalsIgnoreCase(type)) {  *List*<FilmAreaData> result = *new* ArrayList<FilmAreaData>();  *for* (FilmComment comment : filmComments) {  FilmAreaData filmAreaData = *new* FilmAreaData();  filmAreaData.setArea(comment.getMovieCountry());  filmAreaData.setCount(comment.getPeopleCount());  result.add(filmAreaData);  }  *// 返回table data* data = generateTableData(result);  }   *if* ("fileCount".equalsIgnoreCase(type)) {  *List*<FilmCountData> result = *new* ArrayList<FilmCountData>();  *for* (FilmComment comment : filmComments) {  String time = comment.getMovieStartTime();  *if* (time == *null*) {  *continue*;  }  FilmCountData filmCountData = *new* FilmCountData();  filmCountData.setFilmCount(comment.getPeopleCount());  filmCountData.setTime(time);  result.add(filmCountData);  }  *// 返回table data* data = generateTableData(result);  }   *if* ("timeScore".equalsIgnoreCase(type)) {  *List*<FilmTimeScoreData> result = *new* ArrayList<FilmTimeScoreData>();  *for* (FilmComment comment : filmComments) {  *float* score = comment.getScore();  FilmTimeScoreData filmTimeScoreData = *new* FilmTimeScoreData();  filmTimeScoreData.setTimeLength(comment.getMovieLength());  filmTimeScoreData.setScore(score);  result.add(filmTimeScoreData);  }  *// 返回table data* data = generateTableData(result);  }  *return* data;  }   *private* <T> TableData generateTableData(*List*<T> result) {  TableData data = *new* TableData();  *if* (result != *null*){  data.setCode(1000);  data.setMsg("查询成功");  data.setData(result);  }*else* {  data.setCode(2000);  data.setMsg("查询失败");  data.setData(*null*);  }  *return* data;  } } |

|  |
| --- |
| dao层：FilmMapper |
| *package* com.tgu.team04.analysis.dao;  *import* com.tgu.team04.analysis.entity.FilmComment; *import* org.apache.ibatis.annotations.Mapper; *import* org.apache.ibatis.annotations.Param; *import* org.apache.ibatis.annotations.Select; *import* org.springframework.stereotype.Repository;  *import* java.util.*List*;  @Repository @Mapper *public interface FilmMapper* {  *// @Select("select movie\_name, score, movie\_type, movie\_director, movie\_script, movie\_role, movie\_country, movie\_language, movie\_start\_time, movie\_length," + // "from bdcy\_douban\_movie " + // "order by score " + // "limit #{start}, #{limit};")* @Select("select \* " +  "from bdcy\_douban\_movie " +  "order by score " +  "limit #{start}, #{limit};")  *List*<FilmComment> selectCommentList(@Param("start") *int* start, @Param("limit") *int* limit);   @Select("select count(0)" +  "from bdcy\_douban\_movie ")  *int* getTotalNumber();   @Select("select AVG(DISTINCT movie\_length) " +  "from bdcy\_douban\_movie " )  *int* getMovieLength();   @Select("select sum(people\_count)" +  "from bdcy\_douban\_movie")  *int* getPeopleNumber();   @Select("select sum(review\_count)" +  "from bdcy\_douban\_movie")  *int* getCommentCount();   @Select("select max(score)" +  "from bdcy\_douban\_movie")  *float* getMaxScore();   @Select("select min(score)" +  "from bdcy\_douban\_movie")  *float* getMinScore();   @Select("select max(score),min(score)" +  "from bdcy\_douban\_movie")  *List*<Float> getScore();   @Select("select count(movie\_type)" +  "from bdcy\_douban\_movie" +  "where movie\_type=#{type}")  *List*<FilmComment> getTypeSort(@Param("type") *List*<String> type);   @Select("<script>" +  "select a.id,a.movie\_name,a.score, a.people\_count, a.five\_star, a.four\_star,a.three\_star," +  "a.two\_star,a.one\_star,a.short\_review\_count,a.review\_count,a.movie\_type,a.movie\_director,a.movie\_script," +  "a.movie\_role,a.movie\_country,a.movie\_language,a.movie\_start\_time,a.movie\_length,a.movie\_url " +  "from bdcy\_douban\_movie a\n" +  "<where>" +  " <if test='comment.movieName != null and comment.movieName.length > 0'>" +  " AND a.movie\_name like #{comment.movieName}" +  " </if>" +  " <if test='comment.movieDirector != null and comment.movieDirector.length > 0'>" +  " AND a.movie\_director like #{comment.movieDirector}" +  " </if>" +  " <if test='comment.movieRole != null and comment.movieRole.length > 0'>" +  " AND a.movie\_role like #{comment.movieRole}" +  " </if>" +  "</where>" +  "<if test='start != null and limit != null'>" +  " limit #{start}, #{limit}" +  "</if>" +  "</script>")  *List*<FilmComment> selectByWhere( @Param("comment")FilmComment comment, @Param("start")Integer start, @Param("limit")Integer limit);    @Select("<script>" +  "select count(1)" +  " from bdcy\_douban\_movie a" +  "<where>" +  " <if test='comment.movieName != null and comment.movieName.length > 0'>" +  " AND a.movie\_name like #{comment.movieName}" +  " </if>" +  " <if test='comment.movieDirector != null and comment.movieDirector.length > 0'>" +  " AND a.movie\_director like #{comment.movieDirector}" +  " </if>" +  " <if test='comment.movieRole != null and comment.movieRole.length > 0'>" +  " AND a.movie\_role like #{comment.movieRole}" +  " </if>" +   "</where>" +   "</script>")  *int* countSelectByWhere(@Param("comment")FilmComment filmComment);   @Select("select movie\_type,avg(score) score " +  "from bdcy\_douban\_movie " +  "group by movie\_type " +  "ORDER BY score DESC LIMIT 20;")  *List*<FilmComment> simpleAnalysis();   @Select("SELECT score, sum(people\_count) people\_count FROM `bdcy\_douban\_movie`\n" +  "group by score ORDER BY score ASC;")  *List*<FilmComment> commentAnalysis();  @Select("SELECT movie\_country, count(movie\_country) people\_count FROM `bdcy\_douban\_movie`\n" +  "GROUP BY movie\_country ORDER BY people\_count DESC limit 20;")  *List*<FilmComment> areaAnalysis();   @Select("SELECT year(movie\_start\_time) movie\_start\_time, count(id) people\_count FROM bdcy\_douban\_movie\n" +  " GROUP BY year(movie\_start\_time)\n" +  "ORDER BY movie\_start\_time;")  *List*<FilmComment> filmCountAnalysis();   @Select("SELECT max(score) score ,movie\_length FROM `bdcy\_douban\_movie`\n" +  "group by score\n" +  "ORDER BY score")  *List*<FilmComment> timeScoreAnalysis(); } |

|  |
| --- |
| entity层：FilmAreaData |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  @Data *public class* FilmAreaData {   *private* String area;   *private float* count;   *public* String getArea() {  *return* area;  }   *public void* setArea(String area) {  *this*.area = area;  }   *public float* getCount() {  *return* count;  }   *public void* setCount(*float* count) {  *this*.count = count;  } } |

|  |
| --- |
| entity层：FilmComment |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  @Data *public class* FilmComment {  *private int* id;   *private* String movieName;   *private float* score;   *private int* peopleCount;   *private float* fiveStar;   *private float* fourStar;   *private float* threeStar;   *private float* twoStar;   *private float* oneStar;   *private int* shortReviewCount;   *private int* reviewCount;   *private* String movieType;   *private* String movieDirector;   *private* String movieScript;   *private* String movieRole;   *private* String movieCountry;   *private* String movieLanguage;   *private* String movieStartTime;   *private int* movieLength;   *private* String movieUrl;   *public int* getId() {  *return* id;  }   *public void* setId(*int* id) {  *this*.id = id;  }   *public* String getMovieName() {  *return* movieName;  }   *public void* setMovieName(String movieName) {  *this*.movieName = movieName;  }   *public float* getScore() {  *return* score;  }   *public void* setScore(*float* score) {  *this*.score = score;  }   *public int* getPeopleCount() {  *return* peopleCount;  }   *public void* setPeopleCount(*int* peopleCount) {  *this*.peopleCount = peopleCount;  }   *public float* getFiveStar() {  *return* fiveStar;  }   *public void* setFiveStar(*float* fiveStar) {  *this*.fiveStar = fiveStar;  }   *public float* getFourStar() {  *return* fourStar;  }   *public void* setFourStar(*float* fourStar) {  *this*.fourStar = fourStar;  }   *public float* getThreeStar() {  *return* threeStar;  }   *public void* setThreeStar(*float* threeStar) {  *this*.threeStar = threeStar;  }   *public float* getTwoStar() {  *return* twoStar;  }   *public void* setTwoStar(*float* twoStar) {  *this*.twoStar = twoStar;  }   *public float* getOneStar() {  *return* oneStar;  }   *public void* setOneStar(*float* oneStar) {  *this*.oneStar = oneStar;  }   *public int* getShortReviewCount() {  *return* shortReviewCount;  }   *public void* setShortReviewCount(*int* shortReviewCount) {  *this*.shortReviewCount = shortReviewCount;  }   *public int* getReviewCount() {  *return* reviewCount;  }   *public void* setReviewCount(*int* reviewCount) {  *this*.reviewCount = reviewCount;  }   *public* String getMovieType() {  *return* movieType;  }   *public void* setMovieType(String movieType) {  *this*.movieType = movieType;  }   *public* String getMovieDirector() {  *return* movieDirector;  }   *public void* setMovieDirector(String movieDirector) {  *this*.movieDirector = movieDirector;  }   *public* String getMovieScript() {  *return* movieScript;  }   *public void* setMovieScript(String movieScript) {  *this*.movieScript = movieScript;  }   *public* String getMovieRole() {  *return* movieRole;  }   *public void* setMovieRole(String movieRole) {  *this*.movieRole = movieRole;  }   *public* String getMovieCountry() {  *return* movieCountry;  }   *public void* setMovieCountry(String movieCountry) {  *this*.movieCountry = movieCountry;  }   *public* String getMovieLanguage() {  *return* movieLanguage;  }   *public void* setMovieLanguage(String movieLanguage) {  *this*.movieLanguage = movieLanguage;  }   *public* String getMovieStartTime() {  *return* movieStartTime;  }   *public void* setMovieStartTime(String movieStartTime) {  *this*.movieStartTime = movieStartTime;  }   *public int* getMovieLength() {  *return* movieLength;  }   *public void* setMovieLength(*int* movieLength) {  *this*.movieLength = movieLength;  }   *public* String getMovieUrl() {  *return* movieUrl;  }   *public void* setMovieUrl(String movieUrl) {  *this*.movieUrl = movieUrl;  } } |

|  |
| --- |
| entity层：FilmCommentData |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  @Data *public class* FilmCommentData {   *private int* peopleCount;   *private float* score;   *public int* getPeopleCount() {  *return* peopleCount;  }   *public void* setPeopleCount(*int* peopleCount) {  *this*.peopleCount = peopleCount;  }   *public float* getScore() {  *return* score;  }   *public void* setScore(*float* score) {  *this*.score = score;  } } |

|  |
| --- |
| entity层：FilmCountData |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  *import* java.util.Date;  @Data *public class* FilmCountData {   *private int* filmCount;   *private* String time;   *public int* getFilmCount() {  *return* filmCount;  }   *public void* setFilmCount(*int* filmCount) {  *this*.filmCount = filmCount;  }   *public* String getTime() {  *return* time;  }   *public void* setTime(String time) {  *this*.time = time;  } } |

|  |
| --- |
| entity层：FilmSimpleData |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  @Data *public class* FilmSimpleData {   *private* String movieType;   *private float* score;   *public* String getMovieType() {  *return* movieType;  }   *public void* setMovieType(String movieType) {  *this*.movieType = movieType;  }   *public float* getScore() {  *return* score;  }   *public void* setScore(*float* score) {  *this*.score = score;  }   @Override  *public* String toString() {  *return* "FilmSimpleData{" +  "movieType='" + movieType + '\'' +  ", score=" + score +  '}';  } } |

|  |
| --- |
| entity层：FilmSummeryData |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  @Data *public class* FilmSummeryData {   *private int* movieLength;   *private int* movieCount;   *private int* movieCommentCount;   *private float* movieMaxScore;   *private float* movieMinScore;   *public int* getMovieLength() {  *return* movieLength;  }   *public void* setMovieLength(*int* movieLength) {  *this*.movieLength = movieLength;  }   *public int* getMovieCount() {  *return* movieCount;  }   *public void* setMovieCount(*int* movieCount) {  *this*.movieCount = movieCount;  }   *public int* getMovieCommentCount() {  *return* movieCommentCount;  }   *public void* setMovieCommentCount(*int* movieCommentCount) {  *this*.movieCommentCount = movieCommentCount;  }   *public float* getMovieMaxScore() {  *return* movieMaxScore;  }   *public void* setMovieMaxScore(*float* movieMaxScore) {  *this*.movieMaxScore = movieMaxScore;  }   *public float* getMovieMinScore() {  *return* movieMinScore;  }   *public void* setMovieMinScore(*float* movieMinScore) {  *this*.movieMinScore = movieMinScore;  } } |

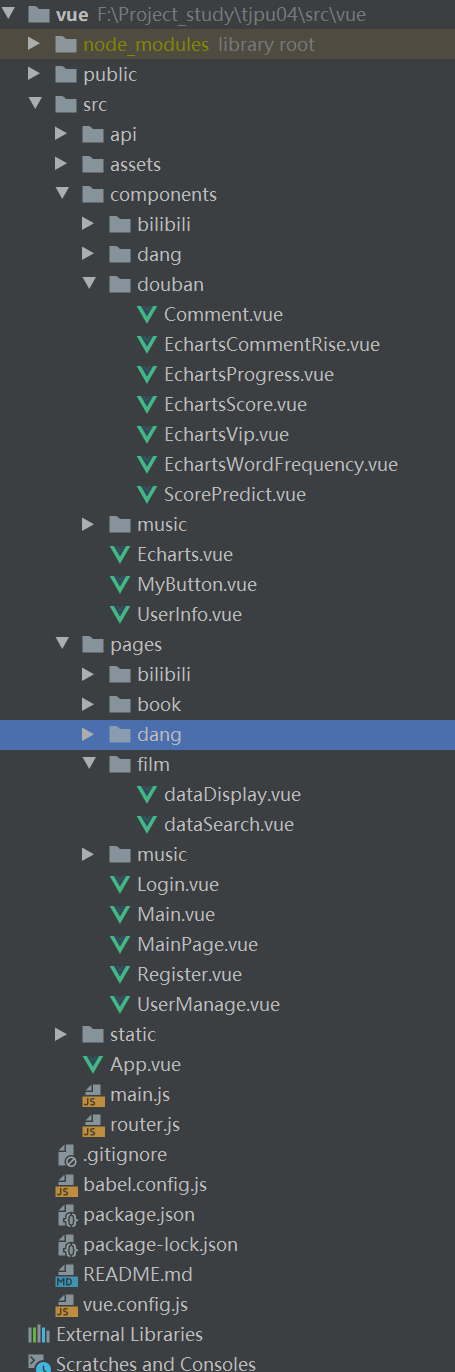
|  |
| --- |
| entity层：FilmTimeScoreData |
| *package* com.tgu.team04.analysis.entity;  *import* lombok.Data;  @Data *public class* FilmTimeScoreData {   *private int* timeLength;   *private float* score;   *public int* getTimeLength() {  *return* timeLength;  }   *public void* setTimeLength(*int* timeLength) {  *this*.timeLength = timeLength;  }   *public float* getScore() {  *return* score;  }   *public void* setScore(*float* score) {  *this*.score = score;  } } |

|  |
| --- |
| service层：FilmServiceImpl |
| *package* com.tgu.team04.analysis.service.impl;  *import* com.tgu.team04.analysis.dao.*FilmMapper*; *import* com.tgu.team04.analysis.entity.AiraSimpleData; *import* com.tgu.team04.analysis.entity.FilmComment; *import* com.tgu.team04.analysis.entity.FilmSummeryData; *import* com.tgu.team04.analysis.service.*FilmService*; *import* org.springframework.beans.factory.annotation.Autowired; *import* org.springframework.stereotype.Service;  *import* java.util.ArrayList; *import* java.util.*List*;  @Service *public class* FilmServiceImpl *implements FilmService* {  @Autowired  *private FilmMapper* mapper;   @Override  *public List*<FilmComment> search(*int* page, *int* limit) {  System.out.println("page: " + page + ", limit: " + limit);  *if*(page > 0 && limit > 0){  System.out.println("查询结果" + mapper.selectCommentList((page - 1)\*limit, limit));  *return* mapper.selectCommentList((page - 1)\*limit, limit);  }  *return null*;  }   @Override  *public int* totalNumber() {  *int* countTotal=mapper.getTotalNumber();   *return* countTotal;  }   @Override  *public int* movieLength() {  *int* movielength=mapper.getMovieLength();   *return* movielength;  }   @Override  *public int* peopleNumber() {  *int* peoplenumber=mapper.getPeopleNumber();   *return* peoplenumber;  }   @Override  *public List*<Float> maxScore() { *// List<Float> maxmin = mapper.getMaxScore(); // List<Float> list = new ArrayList<>(); // int maxscore=mapper.getPeopleNumber(); // int minscore=mapper.getTotalNumber(); // list.add(1,maxscore); // list.add(2,minscore); // Float maxscore = mapper. // return maxmin;  return null*;  }   @Override  *public List*<FilmComment> searchByOption(FilmComment filmComment, *int* page, *int* limit) {  *if* (filmComment.getMovieName() != *null* && !"".equals(filmComment.getMovieName().trim())){  filmComment.setMovieName("%" + filmComment.getMovieName() + "%");  }  *if*(filmComment.getMovieDirector() != *null* && !"".equals(filmComment.getMovieDirector().trim())){  filmComment.setMovieDirector("%" + filmComment.getMovieDirector() + "%");  }  *if*(filmComment.getMovieRole() != *null* && !"".equals(filmComment.getMovieRole().trim())){  filmComment.setMovieRole("%" + filmComment.getMovieRole() + "%");  }  *if* (page > 0 && limit > 0){  *return* mapper.selectByWhere(filmComment, (page - 1)\*limit, limit);  }  *return* mapper.selectByWhere(filmComment, *null*, *null*);  }   @Override  *public int* searchCount(FilmComment filmComment) {  *return* mapper.countSelectByWhere(filmComment);  }   @Override  *public List*<FilmComment> simpleAnalysis(String type) {  *List*<FilmComment> comments = *new* ArrayList<>();  *switch* (type) {  *case* "score" :  comments = mapper.simpleAnalysis();  *break*;  *case* "commentRise" :  comments = mapper.commentAnalysis();  *break*;  *case* "wordFrequency" :  comments = mapper.areaAnalysis();  *break*;  *case* "fileCount" :  comments = mapper.filmCountAnalysis();  *break*;  *case* "timeScore" :  comments = mapper.timeScoreAnalysis();  *break*;   }   *return* comments;  }   @Override  *public* FilmSummeryData movieSummery() {  FilmSummeryData filmSummeryData = *new* FilmSummeryData();  *int* movieCount = mapper.getTotalNumber();  *int* movieLengh = mapper.getMovieLength();  *float* maxScore = mapper.getMaxScore();  *float* minScore = mapper.getMinScore();   *int* commentCount = mapper.getCommentCount();  filmSummeryData.setMovieMaxScore(maxScore);  filmSummeryData.setMovieMinScore(minScore);  filmSummeryData.setMovieCount(movieCount);  filmSummeryData.setMovieLength(movieLengh);  filmSummeryData.setMovieCommentCount(commentCount);  *return* filmSummeryData;  } } |

|  |
| --- |
| service层：FilmService |
| *package* com.tgu.team04.analysis.service;  *import* com.tgu.team04.analysis.entity.AiraSimpleData; *import* com.tgu.team04.analysis.entity.FilmComment; *import* com.tgu.team04.analysis.entity.FilmSummeryData;  *import* java.util.*List*;  *public interface FilmService* {  *List*<FilmComment> search(*int* page, *int* limit);   *int* totalNumber();   *int* movieLength();   *int* peopleNumber();   *List*<Float> maxScore();   *List*<FilmComment> searchByOption(FilmComment filmComment, *int* page, *int* limit);   *int* searchCount(FilmComment filmComment);   *List*<FilmComment> simpleAnalysis(String type);   FilmSummeryData movieSummery(); } |

#### **4.3前端界面功能实现**

###### 4.3.1 项目部署目录结构



###### 4.3.2 数据展示界面-核心代码

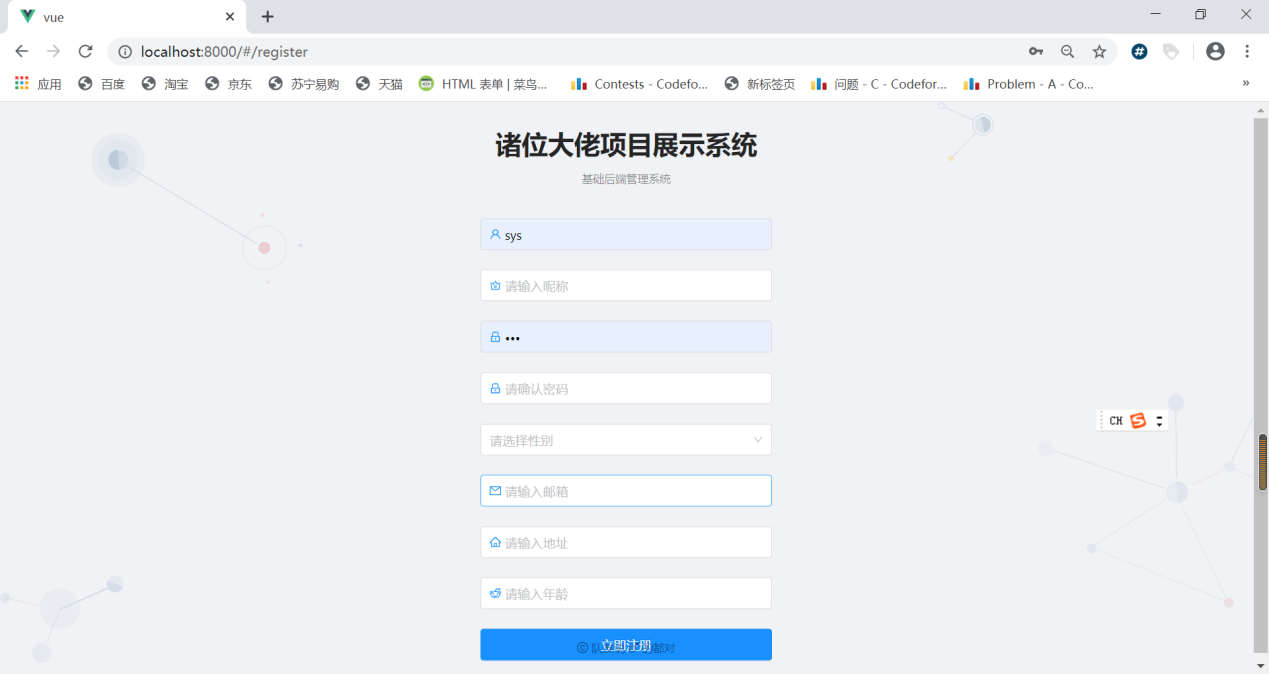
|  |
| --- |
| **dataDisplay.vue** |
| <template>  <div>  <a-row :gutter="[16,16]">  <a-col :span="6" style="width: 20%">  <a-card hoverable>  <a-statistic  title="电影总数"  :value="23476"  :value-style="{ color: '#000080' }"  style="margin-right: 10px"  >  <template #prefix>  <a-icon type="tag" />  </template>  <template #suffix>  <span></span>  </template>  </a-statistic>  </a-card>  </a-col>  <a-col :span="6" style="width: 20%">  <a-card hoverable>  <a-statistic  title="总评论数"  :value="3756"  class="demo-class"  :value-style="{ color: '#0000CD' }"  >  <template #prefix>  <a-icon type="tags" />  </template>  <template #suffix>  <span>w+</span>  </template>  </a-statistic>  </a-card>  </a-col>  <a-col :span="6" style="width: 20%">  <a-card hoverable>  <a-statistic  title="平均片长"  :value="134"  :precision="0"  suffix="min"  class="demo-class"  :value-style="{ color: '#3f8600' }"  >  <template #prefix>  <a-icon type="rocket" :rotate="180" />  </template>  <template #suffix>  <span>min</span>  </template>  </a-statistic>  </a-card>  </a-col>  <a-col :span="6" style="width: 20%">  <a-card hoverable>  <a-statistic  title="最高评分"  :value="9.7"  :precision="1"  suffix=""  class="demo-class"  :value-style="{ color: '#cf1322' }"  >  <template #prefix>  <a-icon type="arrow-up" />  </template>  </a-statistic>  </a-card>  </a-col>  <a-col :span="6" style="width: 20%">  <a-card hoverable>  <a-statistic  title="最低评分"  :value="2.1"  :precision="1"  suffix=""  class="demo-class"  :value-style="{ color: '#cf1322' }"  >  <template #prefix>  <a-icon type="arrow-down" />  </template>  </a-statistic>  </a-card>  </a-col>  </a-row>  <table style="width: 100%">  <tr style="width: 100%">  <div style="background-color: cadetblue; padding: 10px" >  <a-row :gutter="10">  <a-col :span="6" style="width: 40%;">  <a-card title="电影类型分布" :bordered="false" :head-style="{color:'#008b8b'}" :bodyStyle="{padding:'0'}" hoverable>  <div class="easyBox">  <div style="width: 100%" >  <EchartsScore />  </div>  </div>  </a-card>  <template #prefix>  <a-icon type="arrow-up" />  </template>  </a-col>  <a-col :span="6" style="width: 60%">  <a-card title="时间~上映数量" :bordered="false" :headStyle="{color:'#008b8b'}" :bodyStyle="{padding:'0'}" hoverable>  <div class="easyBox">  <div style="width: 100%">  <EchartsVip />  </div>  </div>  <template #prefix>  <a-icon type="arrow-up" />  </template>  </a-card>  </a-col>  </a-row>  </div>  </tr>  <tr style="width: 100%">  <div style="background-color: cadetblue; padding: 10px" >  <a-row :gutter="10">  <a-col :span="6" style="width: 65%;">  <a-card title="地区上映数量" :bordered="false" :head-style="{color:'#008b8b'}" :bodyStyle="{padding:'0'}" hoverable>  <div class="easyBox">  <div style="width: 100%">  <EchartsWordFrequency />  </div>  </div>  <template #prefix>  <a-icon type="arrow-up" />  </template>  </a-card>  </a-col>  <a-col :span="6" style="width: 35%">  <a-card title="评分~片长" :bordered="false" :headStyle="{color:'#008b8b'}" :bodyStyle="{padding:'0'}" hoverable>  <div class="easyBox">  <div style="width: 100%">  <EchartsProgress />  </div>  </div>  <template #prefix>  <a-icon type="arrow-up" />  </template>  </a-card>  </a-col>  </a-row>  </div>  </tr>  <tr>  <div style="background:cadetblue; padding:10px; width: 100%">  <a-card title="评分VS评论人数" :bordered="false" :headStyle="{color:'#008b8b'}" :bodyStyle="{padding:'0'}" hoverable>  <div class="easyItem" >  <div style="width: 100%">  <EchartsCommentRise />  </div>  </div>  <template #prefix>  <a-icon type="arrow-up" />  </template>  </a-card>  </div>  </tr>  </table>  </div> </template>  <script>  import EchartsScore from "../../components/douban/EchartsScore";  import EchartsProgress from "../../components/douban/EchartsProgress";  import EchartsVip from "../../components/douban/EchartsVip";   import EchartsCommentRise from "../../components/douban/EchartsCommentRise";  import EchartsWordFrequency from "../../components/douban/EchartsWordFrequency";  import qs from "qs";  import ***axios*** from "axios"  export default {  name: "dataDisplay",  components:{  EchartsScore,  EchartsProgress,  EchartsCommentRise,  EchartsWordFrequency,  EchartsVip   },  data(){  return{  // 查询数据 我就查四个 前两个和input绑定（双向绑定） 后两个和select绑定（通过回调函数进行修改）  movieCount: null,  movieLength: null,  movieMaxScore: null, // 起初保存默认值  movieMinScore: null,  movieCommentCount: null, // 都是全部查询  // 表格数据  data: [], // 请求结果我会放进去   loading: false // 是否显示加载中图标， 不用管。   }  },  computed:{ // computed是组件的计算属性 这里面不用管 antd不需要动  formItemLayout() {  const { formLayout } = this;  return formLayout === 'horizontal'  ? {  labelCol: { span: 4 },  wrapperCol: { span: 14 },  }  : {};  },  buttonItemLayout() {  const { formLayout } = this;  return formLayout === 'horizontal'  ? {  wrapperCol: { span: 14, offset: 4 },  }  : {};  },  },  //生命周期钩子  mounted() {  this.loading = true; // 和loading有关的可以先忽略   // 数据格式处理，必须做，不用问为什么  // 记得在里面传入后端需要的参数  const data = qs.stringify({  movieName: this.movieName,  })   let array = []; // 保存处理后的查询结果  ***axios***.post("http://localhost:8080/douban/query", data, {headers:{'Content-Type':'application/x-www-form-urlencoded'}})  .then(response => {  // 下面是处理过程  let data = response.data.data; //我想要的评论数据在这里面   for (var index in data){ // 因为是对象类型，所以我遍历它的索引index（想知道数据长啥样，自己console.log）  let comment = data[index] // 然后 通过index获取每一条评论  array = array.concat({ // 将每一条处理后的结果追加到 array数组中  key: comment.id, // 这里面是我需要的数据格式 ！！！！！！！！！！ 自己也搞明白自己需要的！！！ very impotent！  movieLength: comment.movieLength,  movieCount: comment.movieCount,  movieCommentCount: comment.movieCommentCount,  movieMaxScore: comment.movieMaxScore,  movieMinScore: comment.movieMinScore,  })  }  // 总条数和每条评论不在一起， 在 response.data.count里面， 而且也不需要循环， 我们赋给total   //this.data = array // 把处理结果赋值给 data （这才是和表格绑定的数据！）  let result = array[0];  this.movieCommentCount = result.movieCommentCount;  this.movieCount = result.movieCount;  this.movieLength = result.movieLength;  this.movieMaxScore = result.movieMaxScore;  this.movieMinScore = result.movieMinScore;  ***console***.log(result);  this.loading = false  }).catch(error => {  ***console***.log(error)  this.loading = false  })  }  } </script>  <style scoped>  .easyBox{  margin: 10px 0;  display: flex;  justify-content: space-between;  flex-wrap: wrap;  width: 100%;  }  .easyItem{  margin: 10px 0;  display: flex;  justify-content: space-between;  flex-wrap: wrap;  width: 100%;  }  .cu{  display: inline-block;  font-weight:bold;  font-size : 20px;  } </style> |

###### 4.3.3 数据查询界面-核心代码

|  |
| --- |
| **dataSearch.vue** |
| <!--页面结构--> <template>  <div>  <a-page-header  style="border-bottom: 1px solid rgb(235, 237, 240)"  title="豆瓣电影查询"  sub-title="23476"  />  <!-- <MyButton /> -->  <!-- 查询按钮-->  <a-form layout="inline">  <a-form-item  label="电影名称"  :label-col="formItemLayout.labelCol"  :wrapper-col="formItemLayout.wrapperCol"  >  <!--  v-model里 是和 data 绑定的变量名  -->  <a-input placeholder="请输入电影名称" v-model="movieName"/>  </a-form-item>   <a-form-item  label="导演"  :label-col="formItemLayout.labelCol"  :wrapper-col="formItemLayout.wrapperCol"  >  <a-input placeholder="请输入导演名称" v-model="movieDirector"/>  </a-form-item>   <a-form-item  label="主演"  :label-col="formItemLayout.labelCol"  :wrapper-col="formItemLayout.wrapperCol"  >  <a-input placeholder="请输入主演名称" v-model="movieRole"/>  </a-form-item>   <!-- v-on 是绑定事件用的 我们这里绑定了click事件 为click绑定了一个函数  点击时触发  -->  <a-form-item :wrapper-col="buttonItemLayout.wrapperCol">  <a-button type="primary" v-on:click="handleSubmit">  立即查询  </a-button>  </a-form-item>  </a-form>   <!-- 数据列表-->  <!-- 这里没啥改的  第一行 我绑定了一个change事件， 这个是antdv自己写的，原生vue应该没有这个事件，翻页时触发， 可以获得当前页数  :pagination 是为当前组件的属性设置值（动态！） 和data的变量 pagination绑定  -->  <a-table :columns="***columns***" :data-source="data" :loading="loading" v-on:change="nextPage" :pagination = "pagination">  <a slot="name" slot-scope="text">{{ text }}</a>  <span slot="customTitle"><a-icon type="smile-o" /> Name</span>  <span slot="tags" slot-scope="tags">  <a-tag :color=" tags === '普通用户'? '#00a1d6' : tags === '大会员' ? '#fb7299' : '#ff5c7c'"> <!-- 也就在这里和根据标签不同赋予了不同的颜色，别的地方没动-->  {{ tags }}  </a-tag>  </span>  </a-table>  </div> </template>  <!--一些属性和方法--> <script>  //import MyButton from '../../components/MyButton'  import qs from "qs";  import ***axios*** from "axios"   // 这是表格的字段, 按照格式来~  const ***columns*** = [  {  title: '电影名称',  dataIndex: 'movieName',  key: 'movieName',  },  {  title: '评分',  dataIndex: 'score',  key: 'score',  },   {  title: '导演',  dataIndex: 'movieDirector',  key: 'movieDirector',  },  {  title: '主演',  key: 'movieRole',  dataIndex: 'movieRole',   },  {  title: '语言',  dataIndex: 'movieLanguage',  key: 'movieLanguage',  },  {  title: '上映时间',  dataIndex: 'movieStartTime',  key: 'movieStartTime',  },   ];   export default {  name: "index",  // components:{  // MyButton  // },  data(){  return{  // 查询数据 我就查四个 前两个和input绑定（双向绑定） 后两个和select绑定（通过回调函数进行修改）  movieName: null,  movieDirector: null,  movieRole: null, // 起初保存默认值  vipStatus: -1, // 都是全部查询  // 表格数据  data: [], // 请求结果我会放进去  ***columns***, // 刚才说到的 表的字段  loading: false, // 是否显示加载中图标， 不用管。  page: 1, // 当前页数  limit: 10, // 每页多少条  pagination:{ // 分页的设置参数  total: 0, // 把count 传给 total！  }  }  },  computed:{ // computed是组件的计算属性 这里面不用管 antd写好了的，我们也不需要动  formItemLayout() {  const { formLayout } = this;  return formLayout === 'horizontal'  ? {  labelCol: { span: 4 },  wrapperCol: { span: 14 },  }  : {};  },  buttonItemLayout() {  const { formLayout } = this;  return formLayout === 'horizontal'  ? {  wrapperCol: { span: 14, offset: 4 },  }  : {};  },  },  // 生命周期钩子  mounted() {  this.loading = true; // 和loading有关的可以先忽略   // 数据格式处理，必须做，不用问为什么  // 记得在里面传入后端需要的参数  const data = qs.stringify({  movieName: this.movieName,  movieDirector: this.movieDirector,  movieRole: this.movieRole,  page: this.page,  limit: this.limit  })  let array = []; // 保存处理后的查询结果  ***axios***.post("http://localhost:8080/douban/search", data, {headers:{'Content-Type':'application/x-www-form-urlencoded'}})  .then(response => {  // 下面是处理过程  let data = response.data.data; //我想要的评论数据在这里面  for (var index in data){ // 因为是对象类型，所以我遍历它的索引index（想知道数据长啥样，自己console.log）  let comment = data[index] // 然后 通过index获取每一条评论  array = array.concat({ // 将每一条处理后的结果追加到 array数组中  key: comment.id, // 这里面是我需要的数据格式 ！！！！！！！！！！ 自己也搞明白自己需要的！！！ very impotent！  movieName: comment.movieName,  score: comment.score,  peopleCount: comment.peopleCount,  fiveStar: comment.fiveStar,  movieDirector: comment.movieDirector,  movieRole: comment.movieRole,  movieLanguage : comment.movieLanguage,  movieStartTime: comment.movieStartTime,  movieUrl: comment.movieUrl  })  }  // 总条数和每条评论不在一起， 在 response.data.count里面， 而且也不需要循环， 我们赋给total  this.pagination.total = response.data.count  this.data = array // 把处理结果赋值给 data （和表格绑定的数据！）  this.loading = false  }).catch(error => {  ***console***.log(error)  this.loading = false  })  },  methods:{  handleSubmit(){ // 点击立即查询后触发  this.loading = true; // 和loading有关的可以先忽略  // 数据格式处理，必须做，不用问为什么  // 记得在里面传入后端需要的参数  const data = qs.stringify({  movieName: this.movieName,  movieDirector: this.movieDirector,  movieRole: this.movieRole,  page: this.page,  limit: this.limit  })  let array = []; // 保存处理后的查询结果  ***axios***.post("http://localhost:8080/douban/search", data, {headers:{'Content-Type':'application/x-www-form-urlencoded'}})  .then(response => {  // 下面是处理过程  let data = response.data.data; //我想要的评论数据在这里面  for (var index in data){ // 因为是对象类型，所以我遍历它的索引index（想知道数据长啥样，自己console.log）  let comment = data[index] // 然后 通过index获取每一条评论  array = array.concat({ // 将每一条处理后的结果追加到 array数组中  key: comment.id, // 这里面是我需要的数据格式 ！！！！！！！！！！ 自己也搞明白自己需要的！！！ very impotent！  movieName: comment.movieName,  score: comment.score,  peopleCount: comment.peopleCount,  fiveStar: comment.fiveStar,  movieDirector: comment.movieDirector,  movieRole: comment.movieRole,  movieLanguage : comment.movieLanguage,  movieStartTime: comment.movieStartTime,  movieUrl: comment.movieUrl   })  }  // 总条数和每条评论不在一起， 在 response.data.count里面， 而且也不需要循环， 我们赋给total  this.pagination.total = response.data.count  this.data = array // 把处理结果赋值给 data （这才是和表格绑定的数据！）  this.loading = false  }).catch(error => {  ***console***.log(error)  this.loading = false  })  },  nextPage(pagination){  this.page = pagination.current; // 获取当前页码并进行更新  // 和上面的函数一样（没一点不一样）  this.loading = true;  const data = qs.stringify({  movieName: this.movieName,  movieDirector: this.movieDirector,  movieRole: this.movieRole,  page: this.page,  limit: this.limit  })  let array = []; // 保存处理后的查询结果  ***axios***.post("http://localhost:8080/douban/search", data, {headers:{'Content-Type':'application/x-www-form-urlencoded'}})  .then(response => {  let data = response.data.data;  for (var index in data){  let comment = data[index]  array = array.concat({  key: comment.id,  movieName: comment.movieName,  score: comment.score,  peopleCount: comment.peopleCount,  fiveStar: comment.fiveStar,  movieDirector: comment.movieDirector,  movieRole: comment.movieRole,  movieLanguage : comment.movieLanguage,  movieStartTime: comment.movieStartTime,  movieUrl: comment.movieUrl  })  }  this.pagination.total = response.data.count  this.data = array  this.loading = false  }).catch(error => {  ***console***.log(error)  this.loading = false  })  },  test(props){  ***console***.log(props)  }  }  } </script> <!--页面的样式--> <style scoped> </style> |

###### 4.3.4成果展示

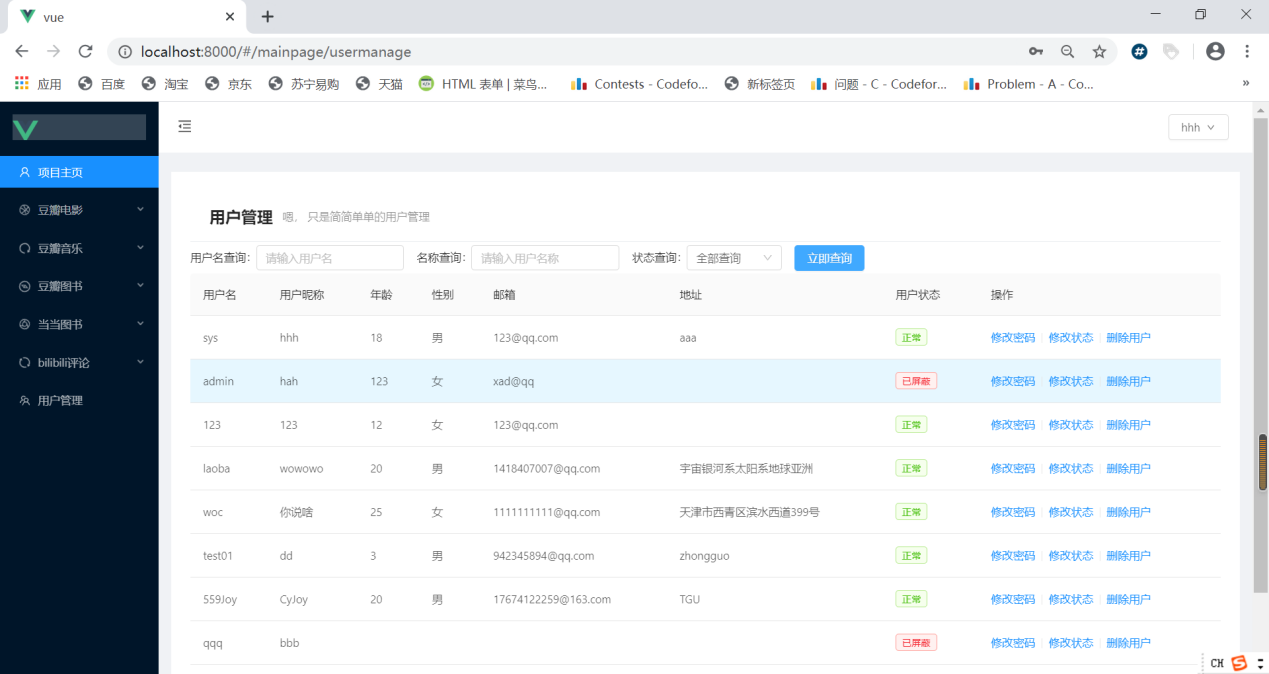
用户注册：系统自动判断条件，若符合规则则将用户添加如数据库中



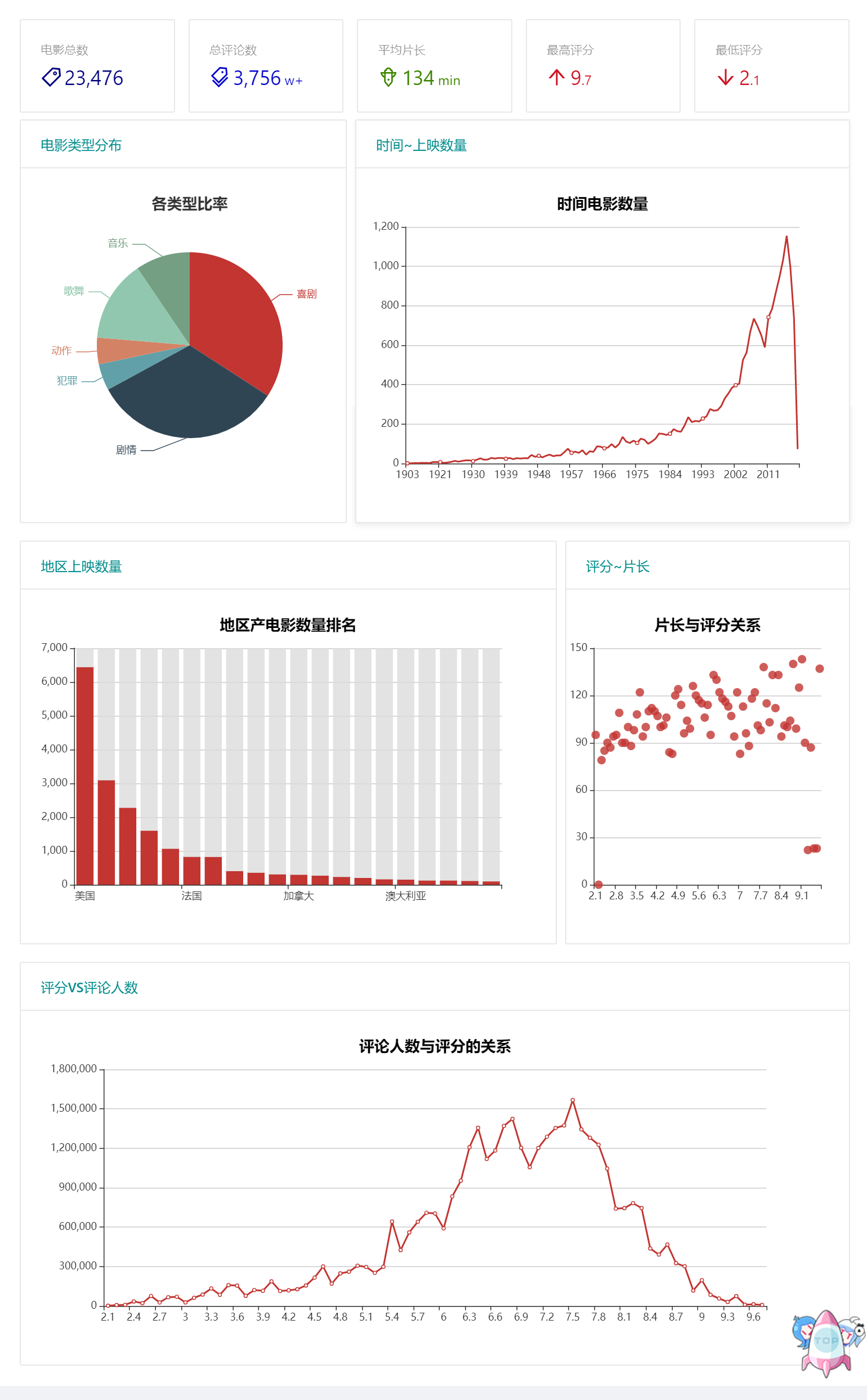
用户登录：传给给后端用户名，判断用户名密码存在时则登录成功，否则失败



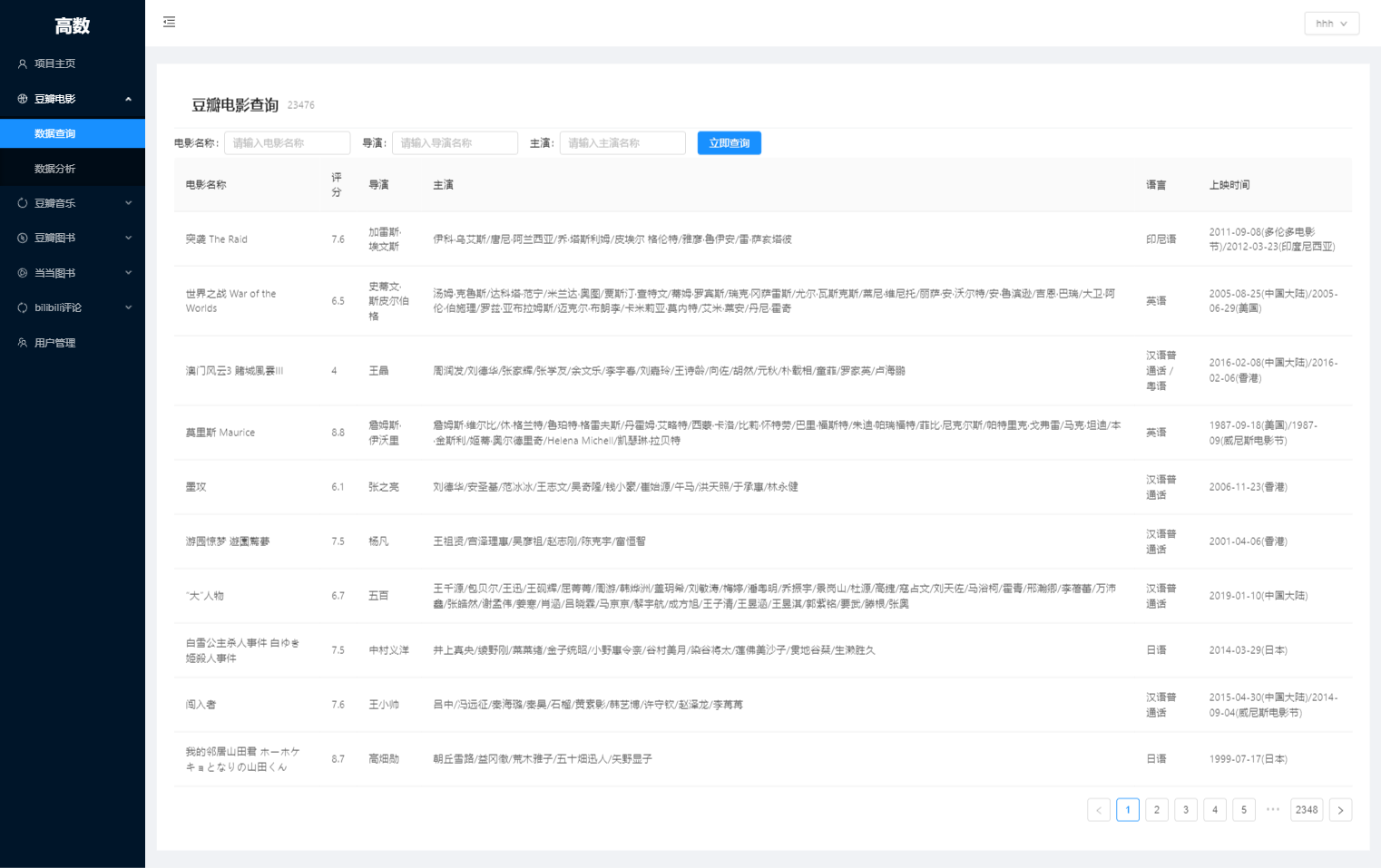
用户管理：可以在界面中查询所有用户，还可以修改用户状态、密码、删除用户



豆瓣电影展示数据界面



豆瓣电影查询界面：可以根据该关键字查询所需的电影

****

**5 系统测试**

测试方法：通过Postman测试接口返回值是否正常，前端登录页面测试，修改室主要依据idea、webstorm自带报错机制。