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

姓名：薛彦涵

　实训起止日期：2020.6.15-2020.7.15

**1 引言**

1.1项目概述

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。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。

为便于完成项目，本次实训中企业为我们提供了scrapy爬虫基础教学、SSM服务器短开发框架，LayUI前端技术以及所需要的模版、软件

1.2 需求分析

随着社会发展，人们的生活质量越来越高，伴随着各种娱乐产业的新增，每天都有海量数据生成。单单只看其中一些数据，是毫无意义的，担当将当量数据集中金雄分析，就能从中取得许多信息，这也是大数据的意义所在。

该项目将爬取豆瓣电影、音乐、图书和当当图书、B站动漫评论的数据，然后所获得的信息进行分析，得到诸如最畅销的书籍、最受欢迎的导演等等数据，之后以表格的形式将分析直观的展示出来，以此为客户挑选电影、音乐、动漫、书籍时提供便捷。

###### 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。

1.3 运行环境

1. 操作系统：Windows 10。

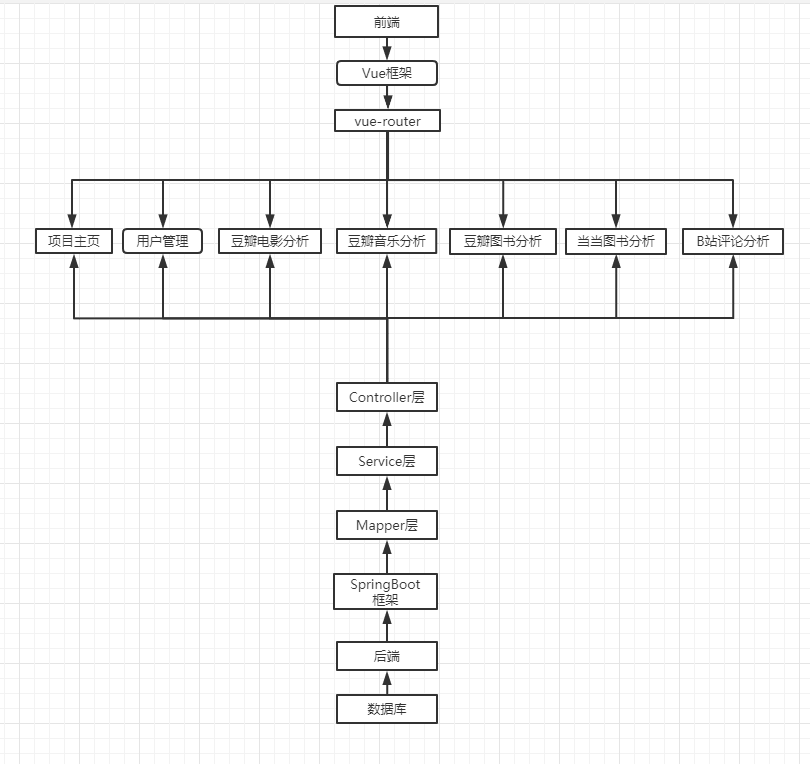
2. 文档编写工具：Microsoft office 2016（word excel powerpoint visio）。

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



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

2.2 模块功能介绍

###### 2.2.当当数据爬取

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（当当）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书籍价格，折扣力度。用于进行价格分析

###### 2.2.2豆瓣电影数据查询

1. 针对当当TOP图书，爬取图书相关数据字段，同时选取书名、推荐、作者、折扣、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：书名、推荐、作者、折扣、出版日期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：书本总数、总推荐比、平均评论数、最高售价、最低售价
2. 各书目类型占比情况-饼图
3. 各分类均价-柱状图
4. 图书出版年份-折线图
5. 评论与推荐的关系-散点图
6. 各个年份上榜书本数目关系-玫瑰图

2.3 模块结构图

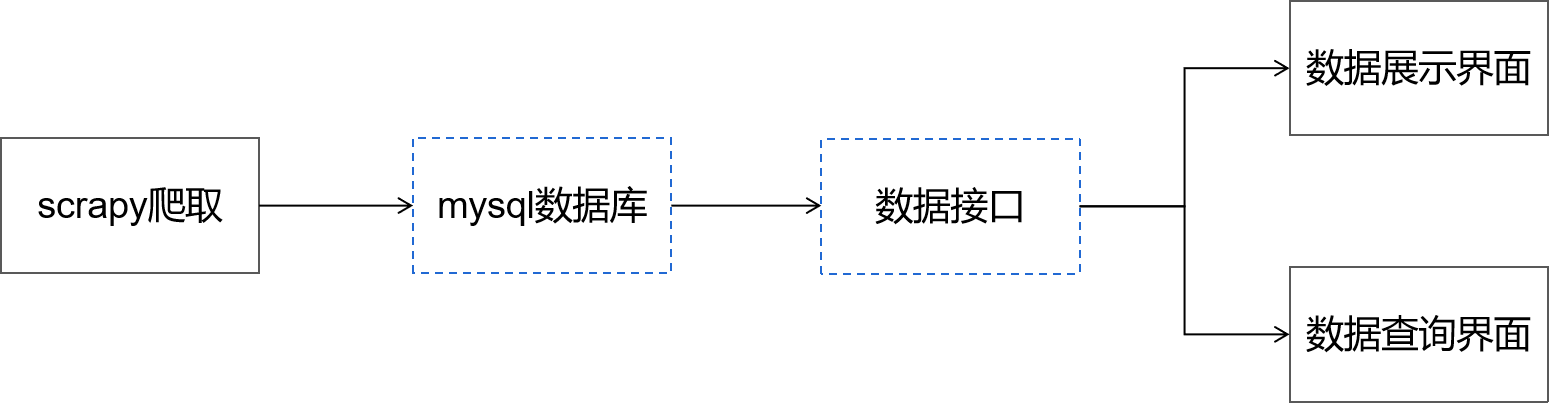


Figure 2模块结构图

项目大致可分为七个模块，包含两个主模块，五个小模块。主模块包括平台外部用户个人信息管理，主要负责用户注册，修改信息；系统基础管理模块负责平台页面框架的搭建，其中包含着当当图书、豆瓣图书、豆瓣电影、豆瓣音乐、B站评论五个模块的内容

2.4 程序流程图

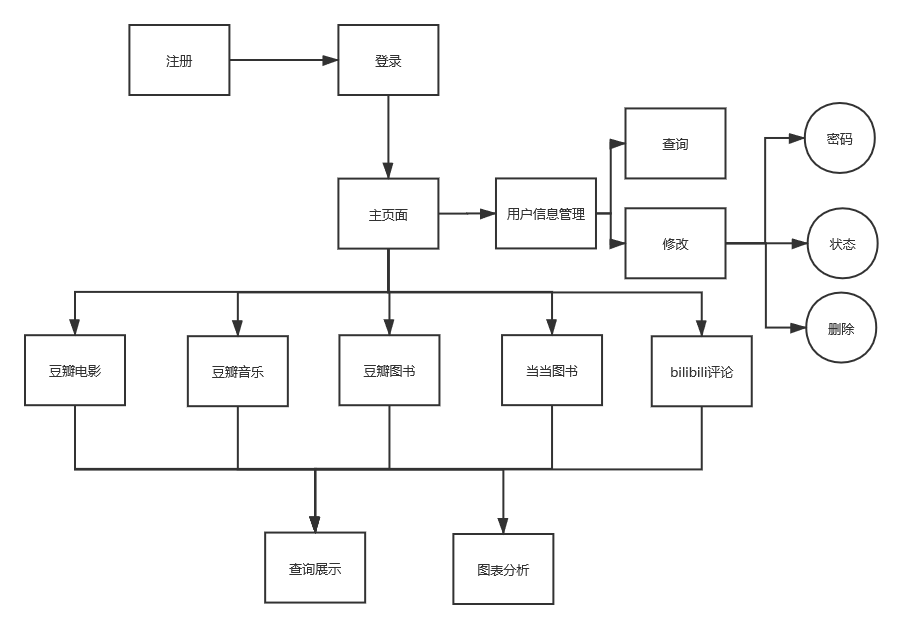


Figure 3程序流程图

2.5 功能设计分工

需详细标注每部分的内容，具体任务及任务安排等。

数据爬取：

豆瓣电影：段成源

豆瓣音乐：刘博

豆瓣图书：廖嘉伟

当当图书：薛彦涵

B站评论：李嘉威

前端框架：

主界面搭建：李嘉威

登录：薛彦涵

注册：李嘉威

用户管理：李嘉威

豆瓣电影：段成源

豆瓣音乐：刘博

豆瓣图书：廖嘉伟

当当图书：薛彦涵

后端

登录：李嘉威

注册： 刘博，薛彦涵

豆瓣电影：段成源

豆瓣音乐：刘博

豆瓣图书：廖嘉伟

当当图书：薛彦涵

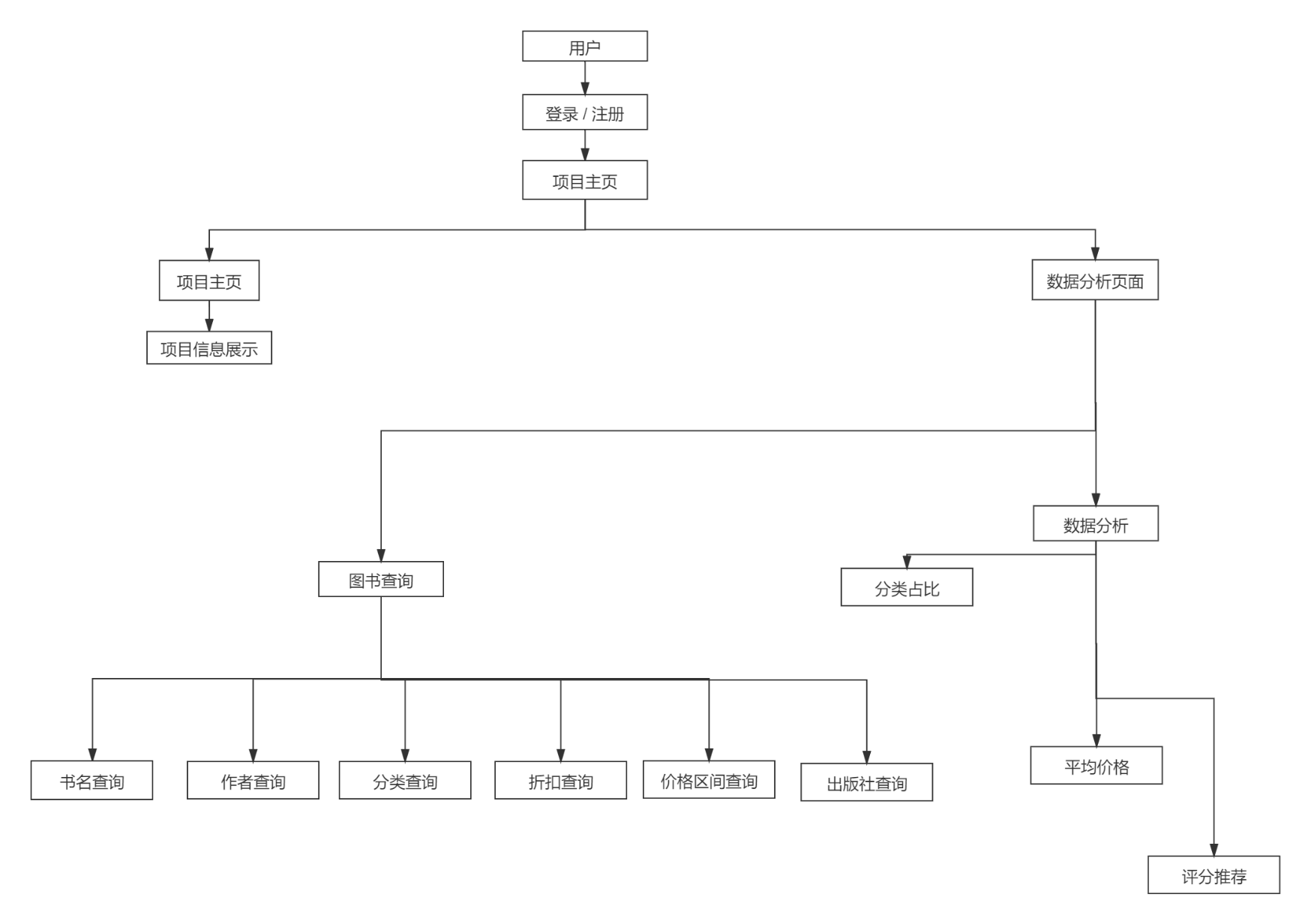
**3 详细设计**（小三黑体，居中）

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。

#### 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



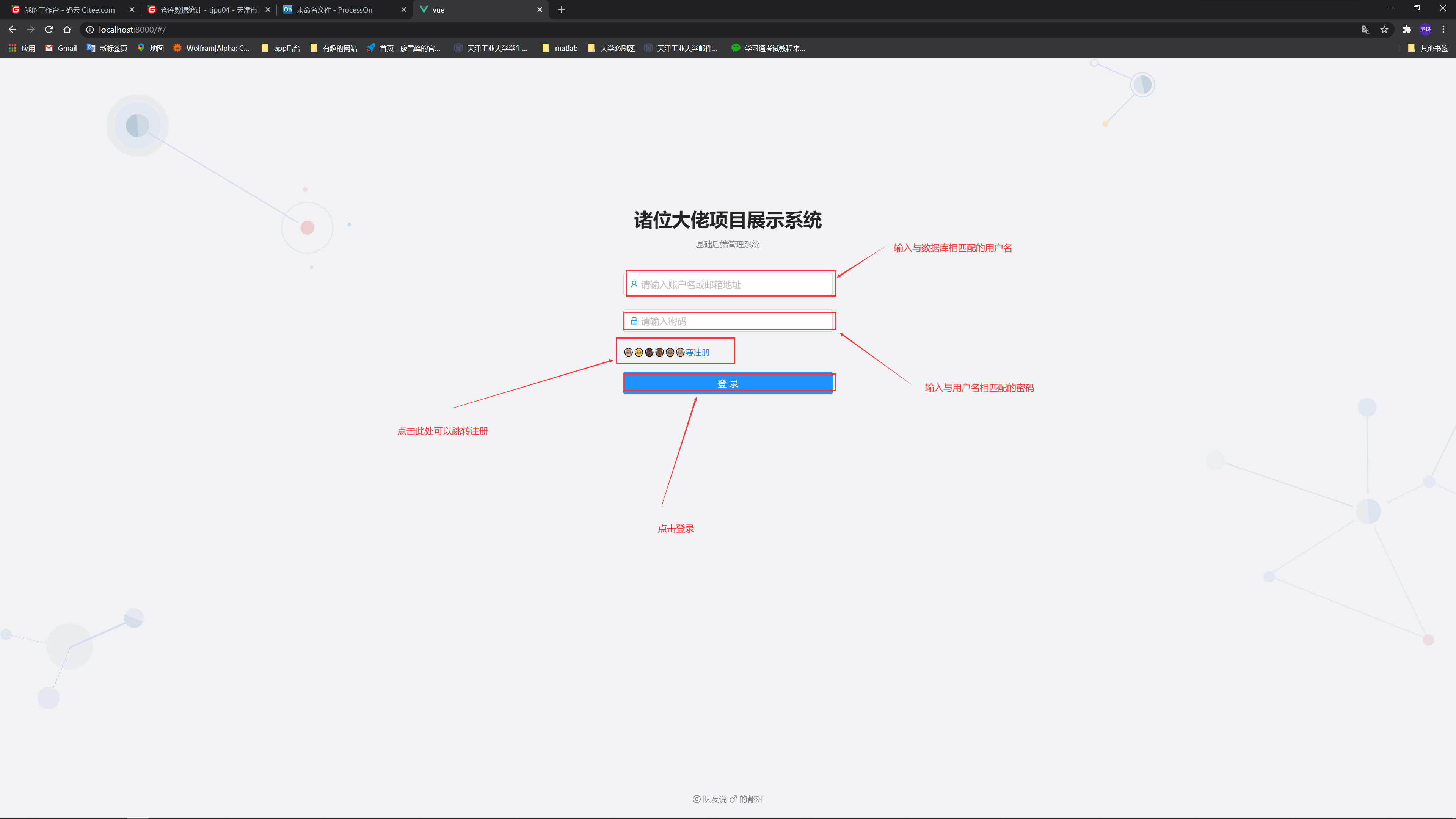
**4 实现方案**（小三黑体，居中）

详细写明项目功能实现描述、核心代码及实现结果等；

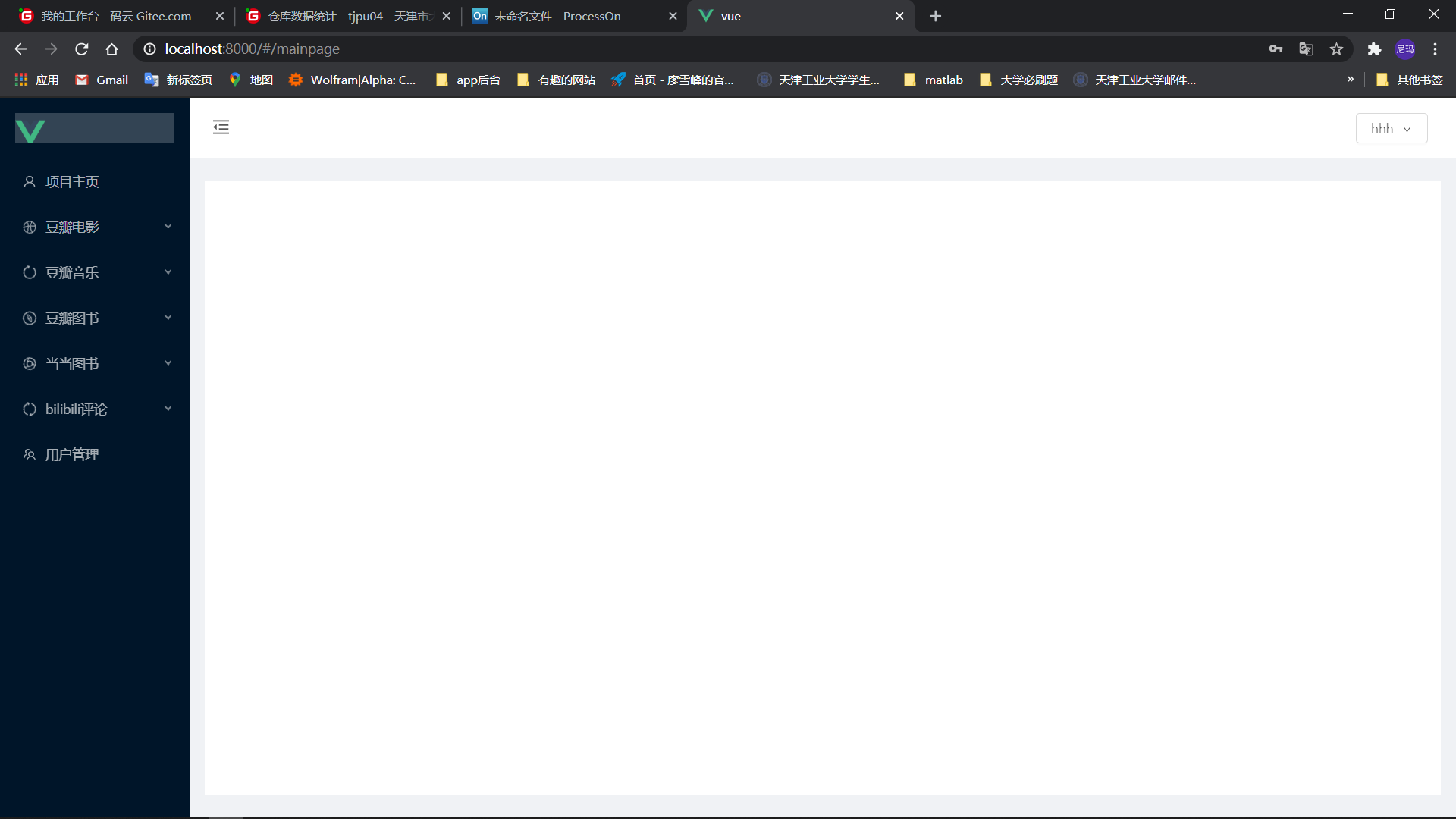
并对调试好的程序，从执行程序弹出界面开始，每一步操作截一个图，并附加说明（图片格式为JPG格式）。

本人负责的项目功能有：部分前端框架（已被废弃），前端登录功能，当当图书的查询功能和基于数据库关于当当网的数据经行数据分析

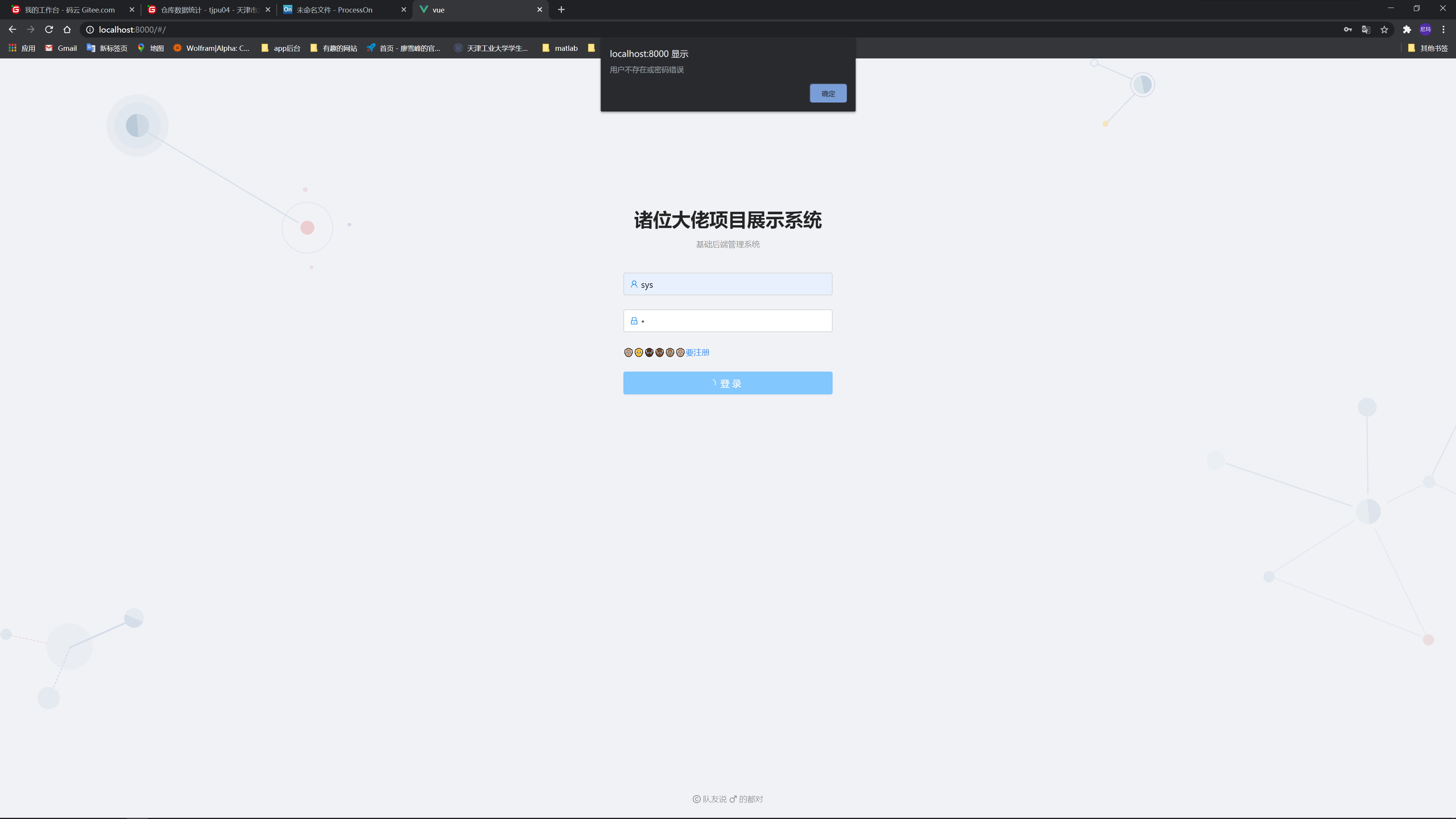
实现描述：当位于登录界面时：在本地运行登录界面



输入成功后，便可以跳转至主界面：



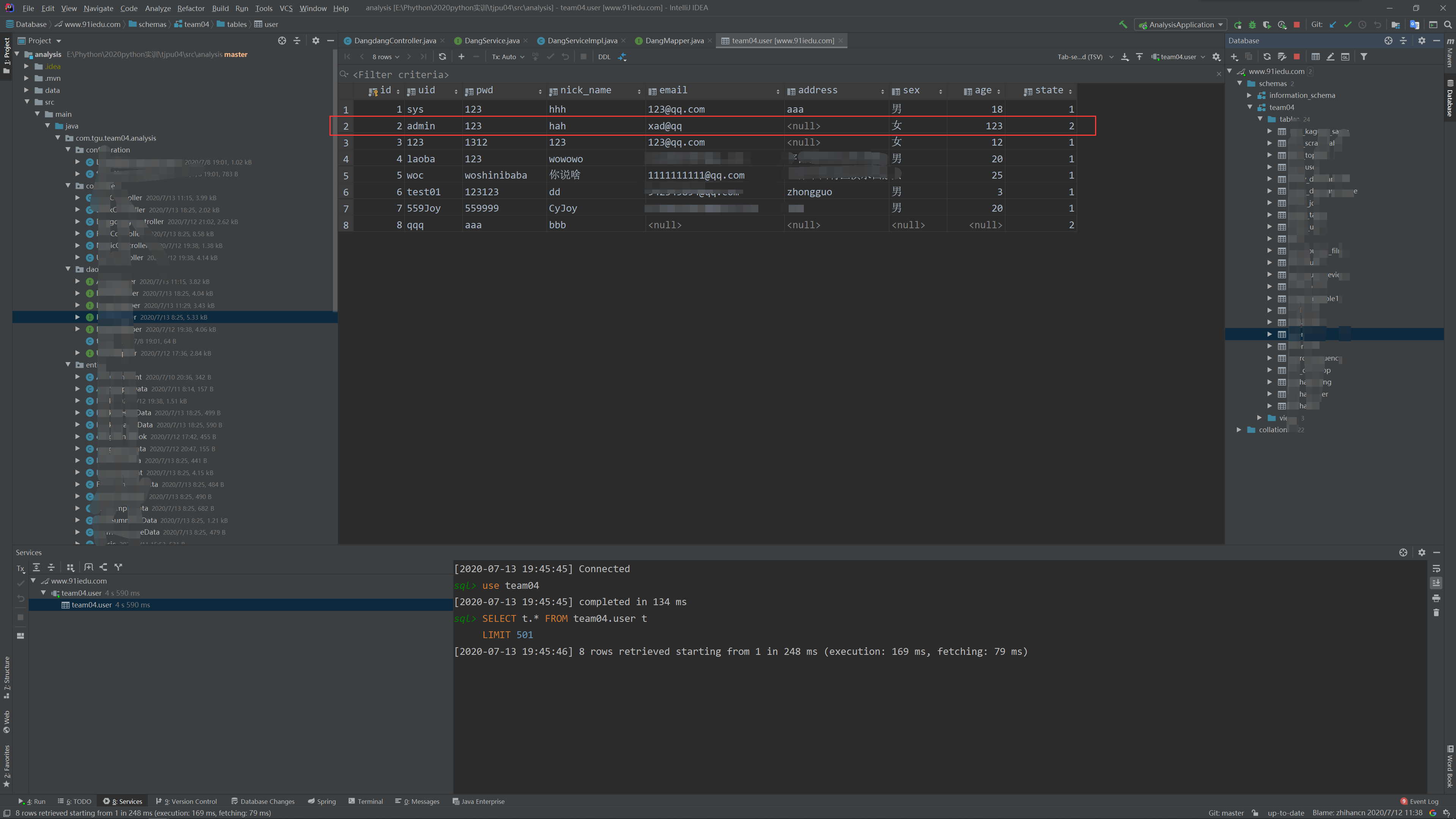
如果输入不正确的用户名或密码，就会弹出警示：

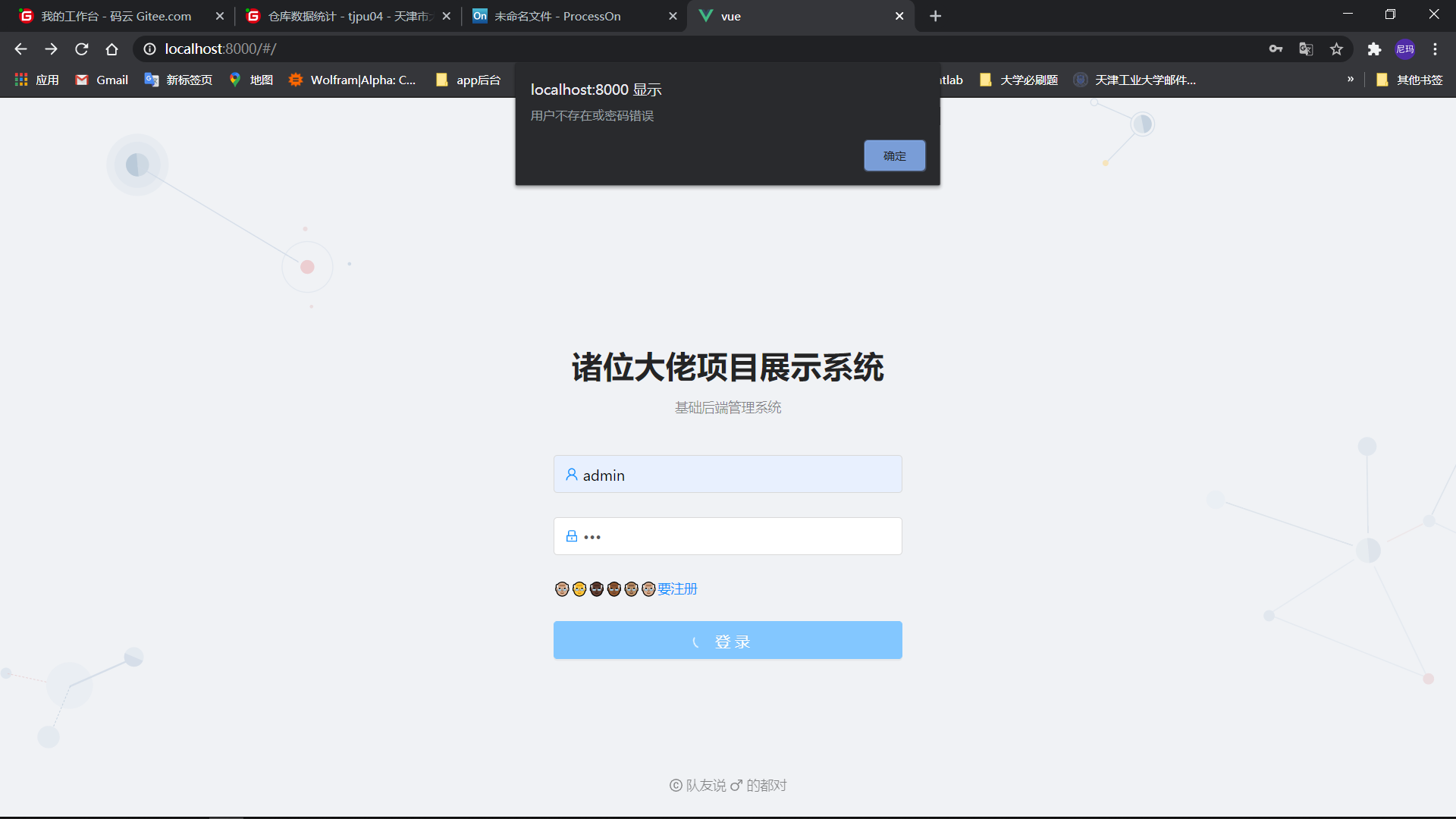


并会自动刷新界面。

如果与数据库用户名密码相匹配，但是用户状态是注销或者未激活，登录界面也不会给予通过

此用户状态不予通过





核心代码：

位于注册界面的前端代码：

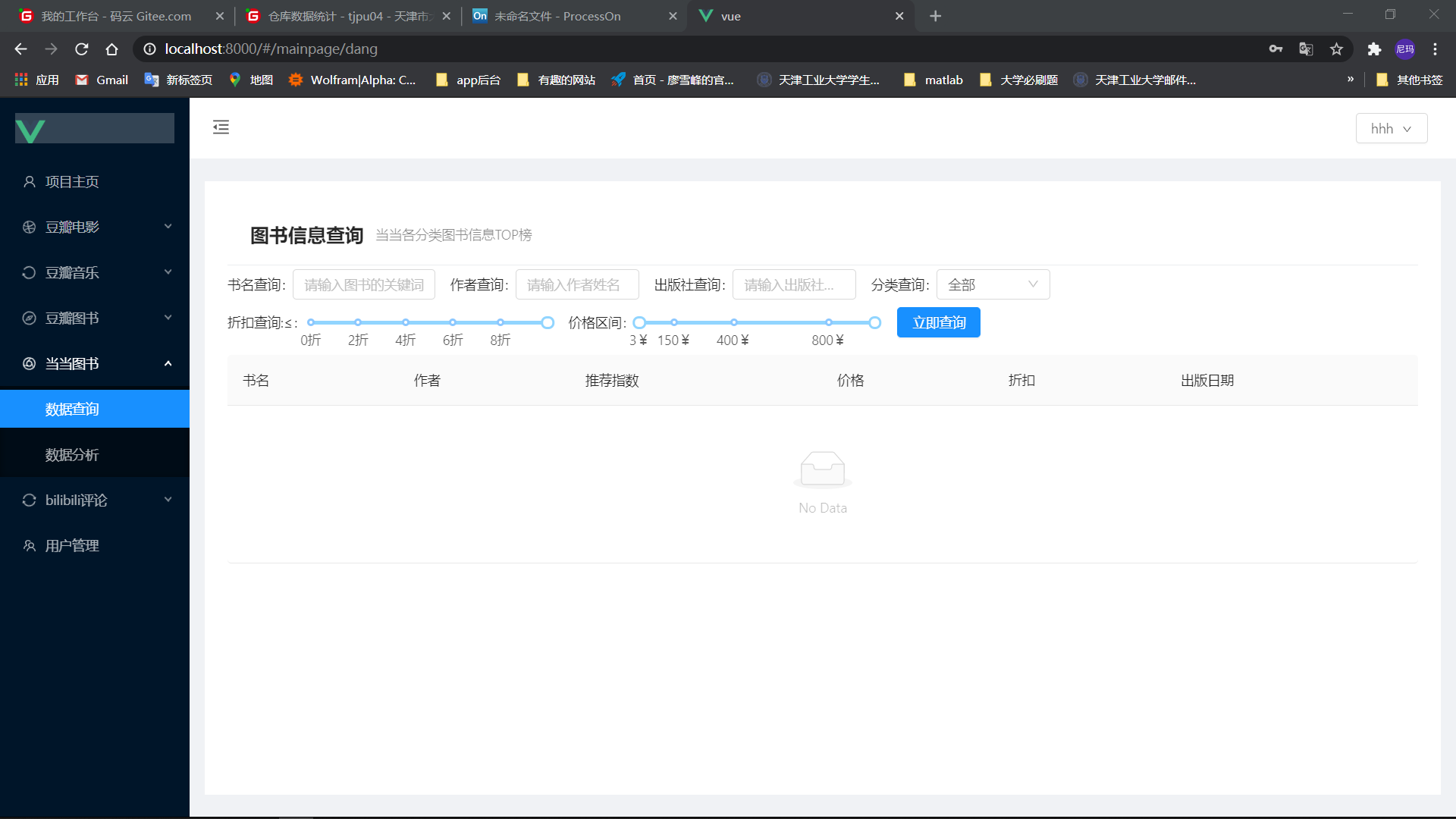
<script>  
 import axios from 'axios'  
 // import router from "../router";  
 import qs from 'qs';  
 // import router from "../router";  
  
 export default {  
 name: 'login',  
 data() {  
 return {  
 form: this.$form.createForm(this),  
 isLoginError: false,  
 userName: null,  
 userPassword: null,  
 iconLoading: false  
 }  
 },  
 methods: {  
  
  
 login: function () {  
 this.iconLoading = {delay: 10};  
 let data = qs.stringify({  
 uid: this.userName,

pwd: this.userPassword  
 })

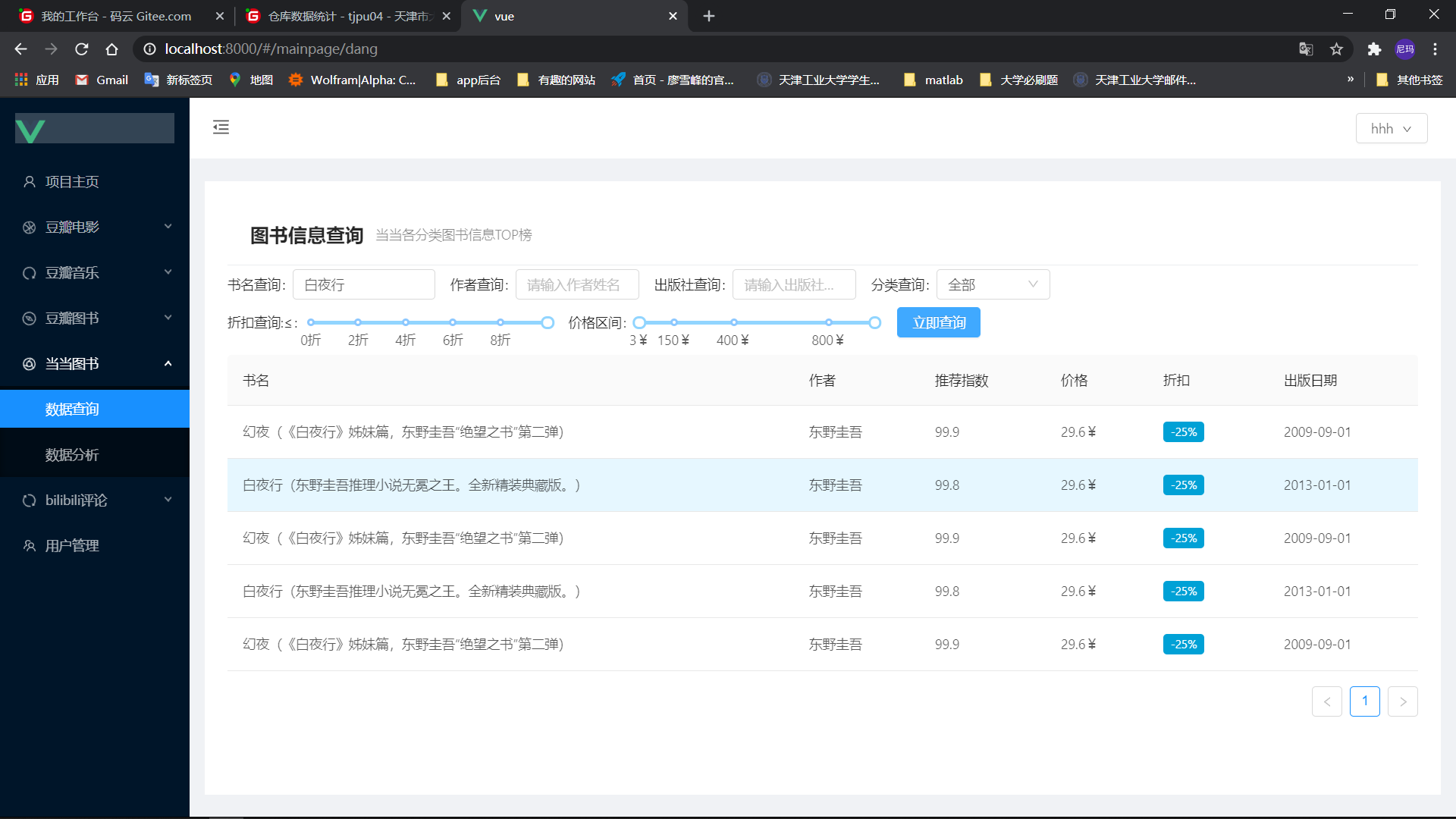
// this.$router.push("/mainpage") 不知道哪位大神在这边加了一个跳转 BY 薛某  
 axios.post('http://localhost:8080/logic/user/login', data,  
 { headers: {'Content-Type': 'application/x-www-form-urlencoded'}  
 }).then(response => {  
 data = response.data;  
 console.log(data);  
 // console.log(typeof (data.code));  
 if (data.code === 1000) {  
 console.log("登录成功");  
 this.$router.push({  
 name: "MainPage",  
 params: {  
 uid: this.userName  
 }  
 });  
 } else if (data.code === 2000) {  
 alert("用户不存在或密码错误");  
 ***window***.location.reload();  
 }  
 }).catch(error => {  
 ***console***.log(error);  
 })  
 }  
 }  
 }

当当图书查询功能描述：允许查询任意书的书名，允许查询作者任意字，允许

查询出版社的任意字，允许级联查询分类，允许通过滑动条经行折扣范围查询，允许通过双端滑动条查询价格区间，以上条件均允许一个或多个同时查询

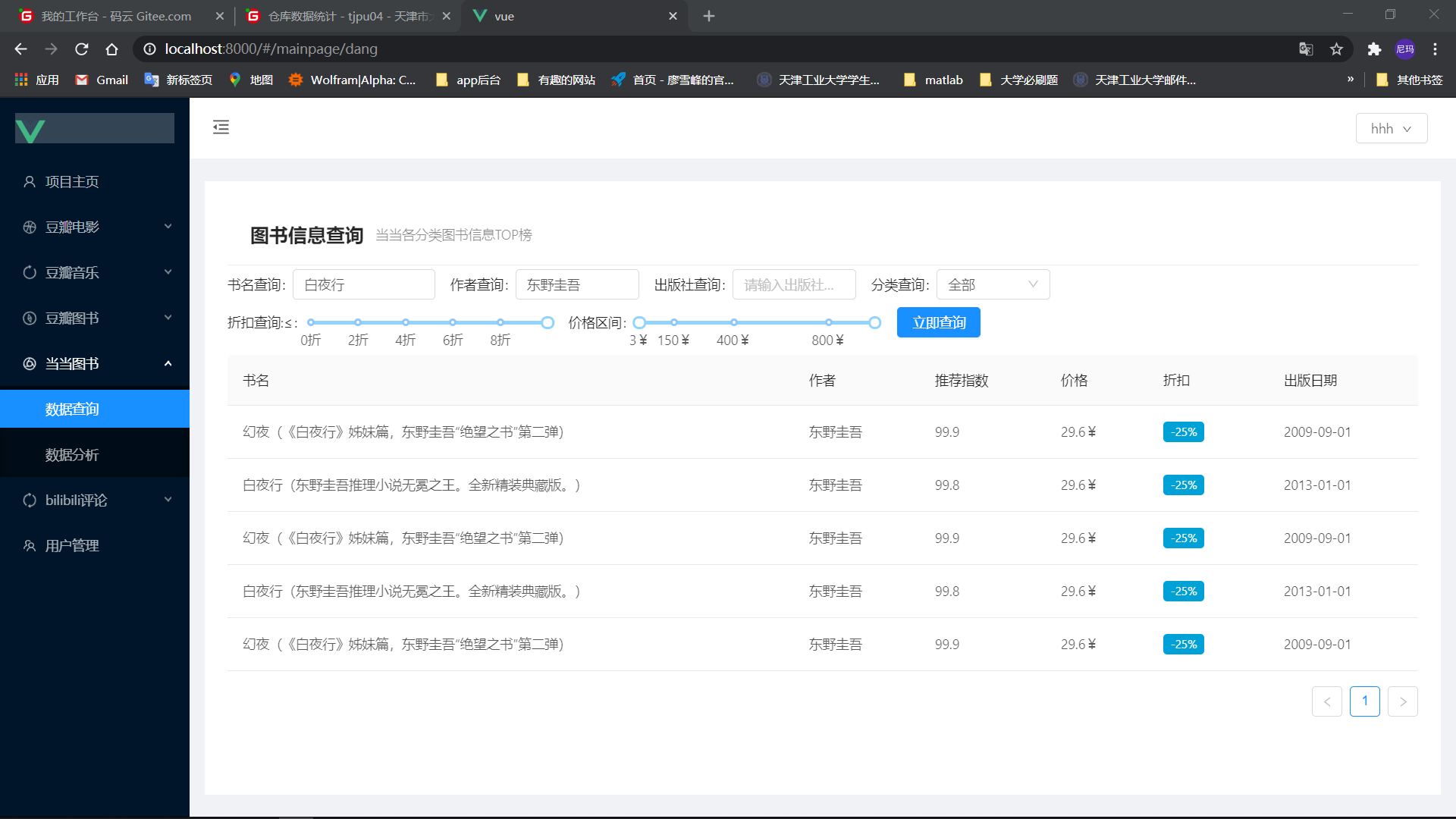


例如：查询《白夜行》：

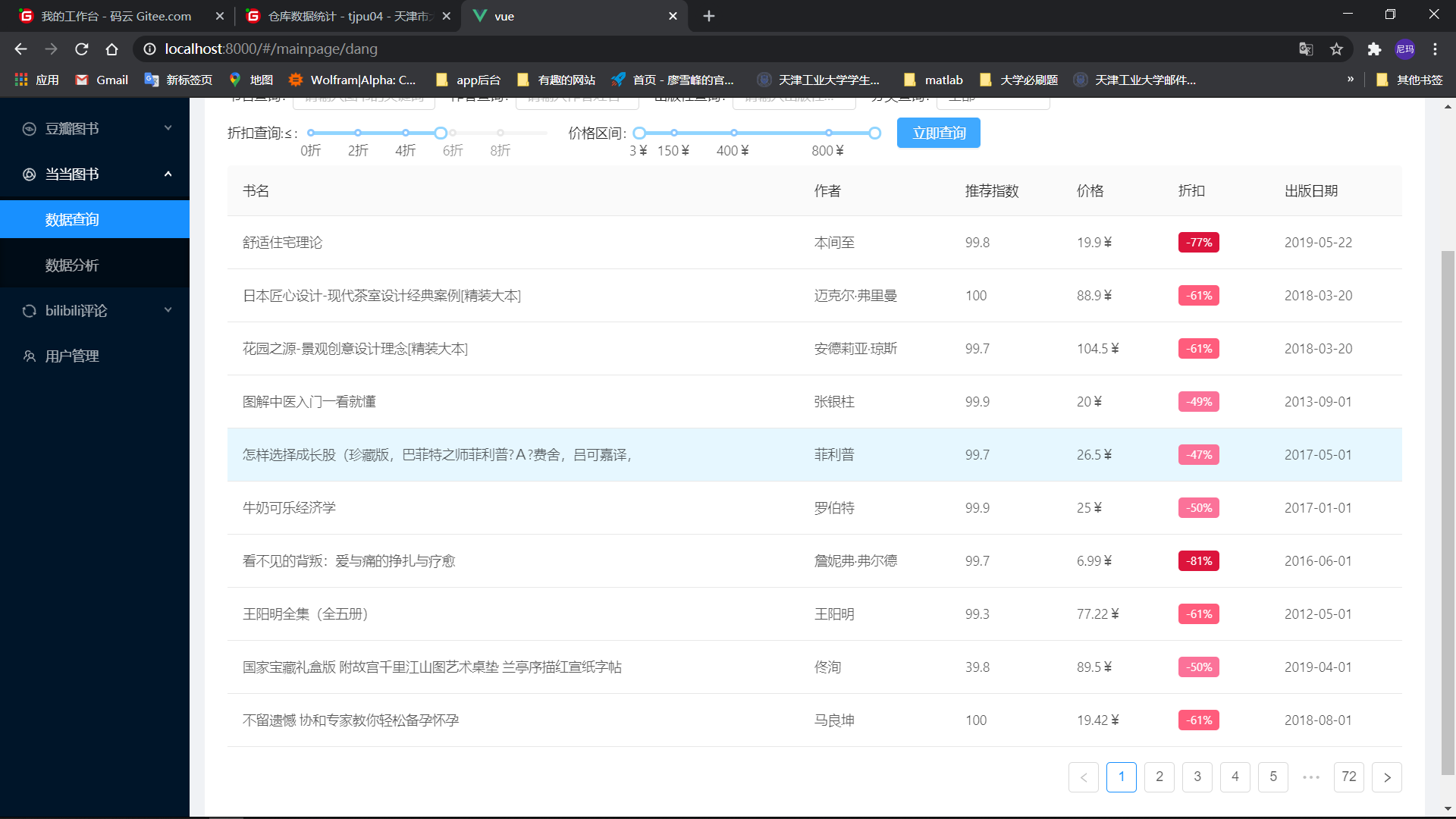


折扣颜色可表示折扣力度

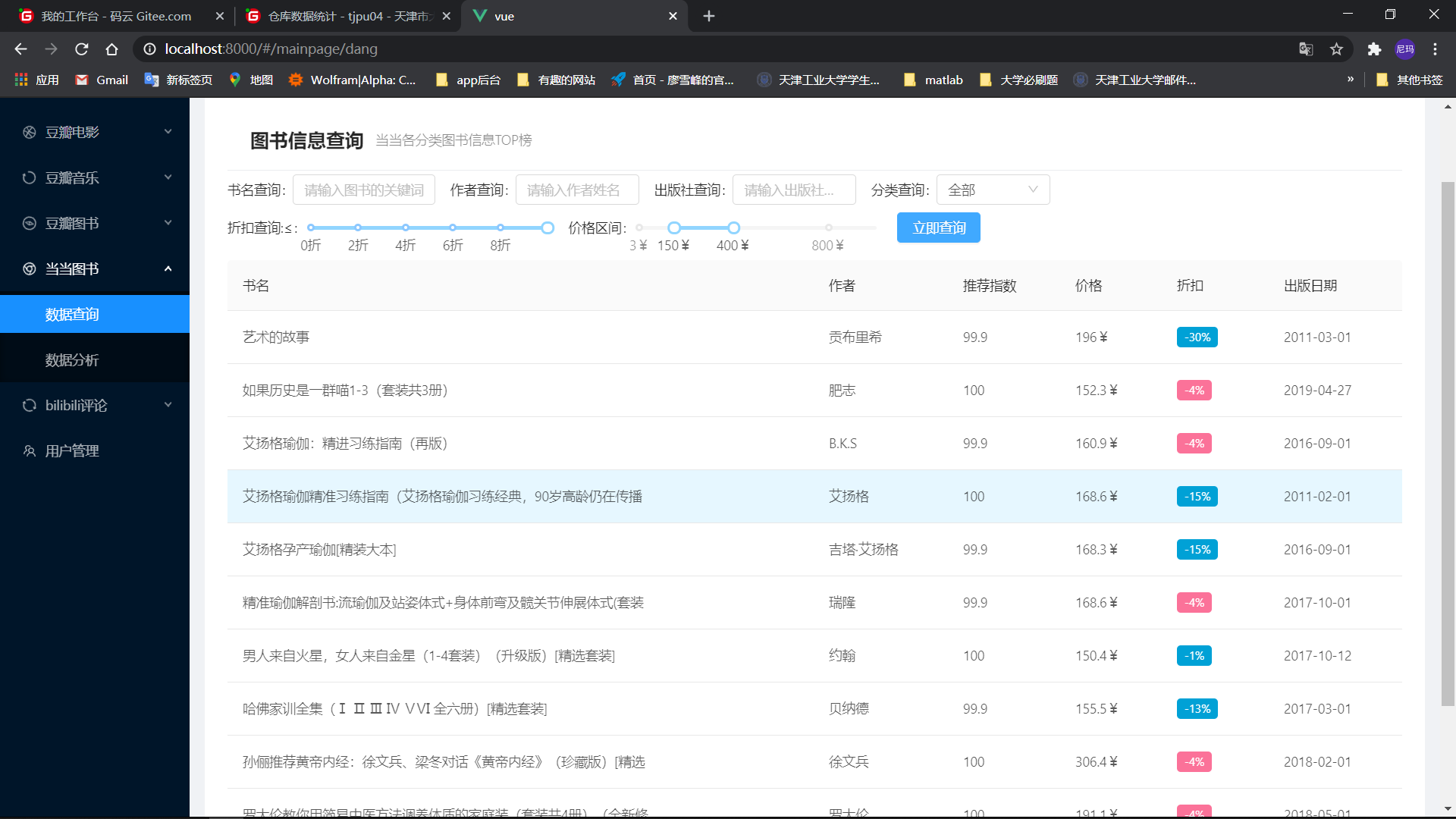
同时查询白夜行 作者 东野圭吾



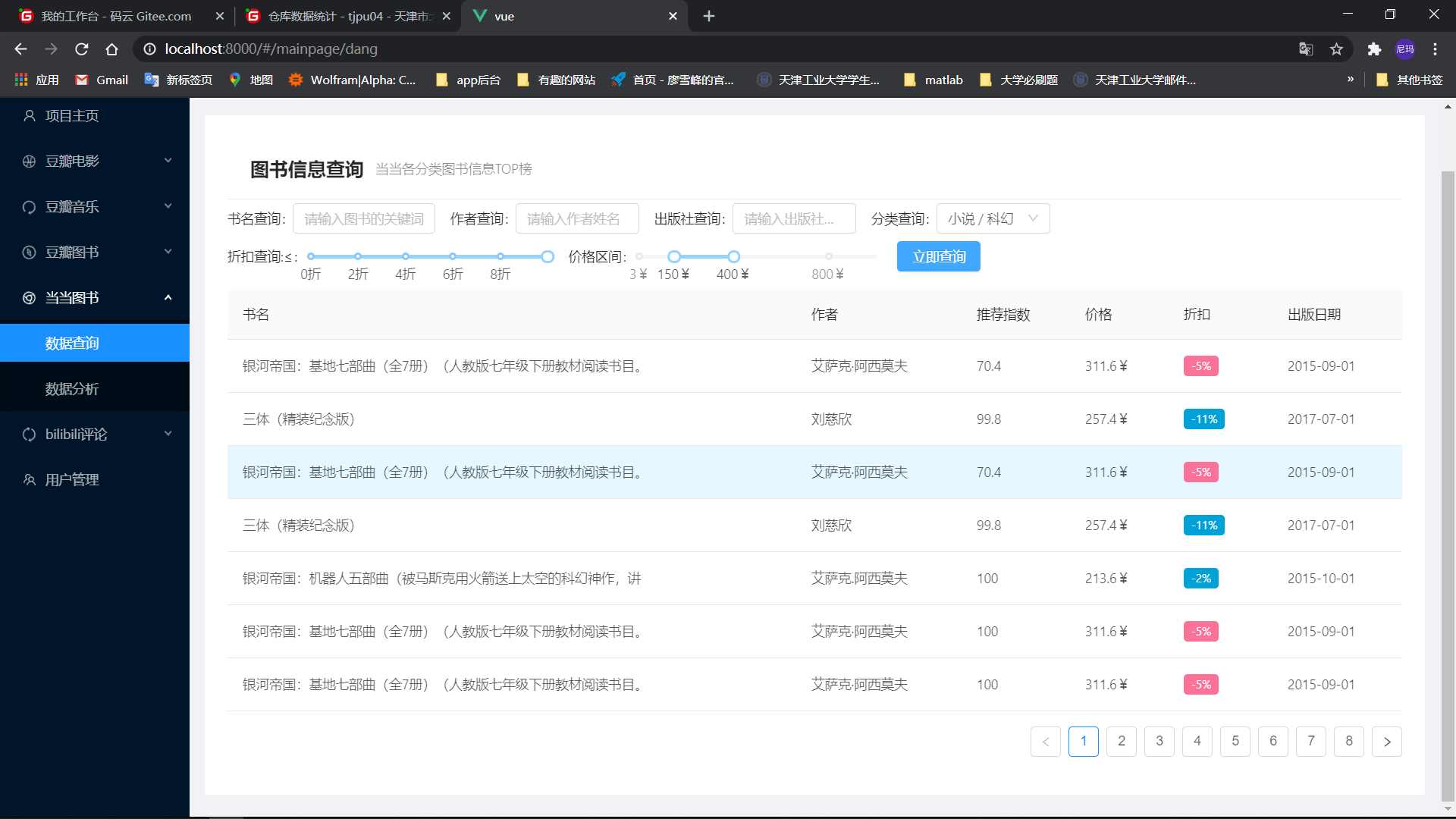
查询折扣力度：6折以内



查询价格区间 150~400元



查询任意一类图书：



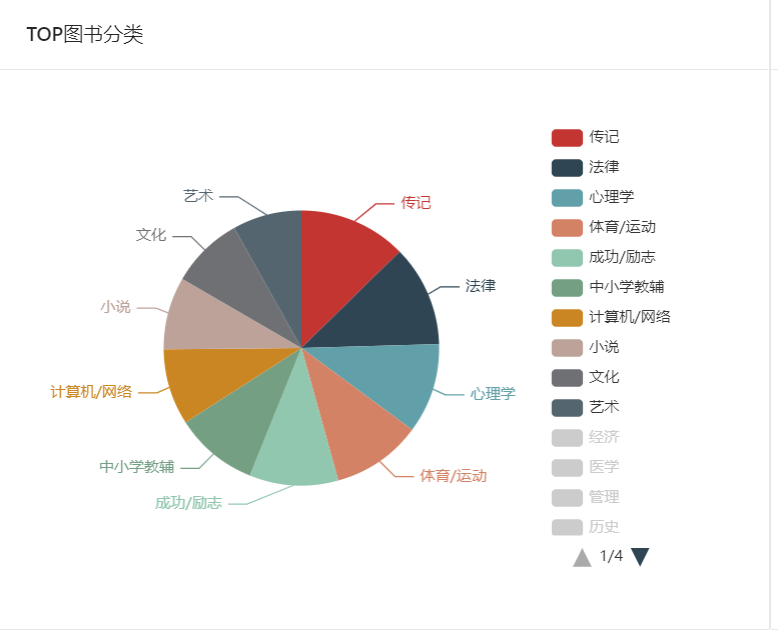
查询核心前端代码：

基础传入后台：

data() {

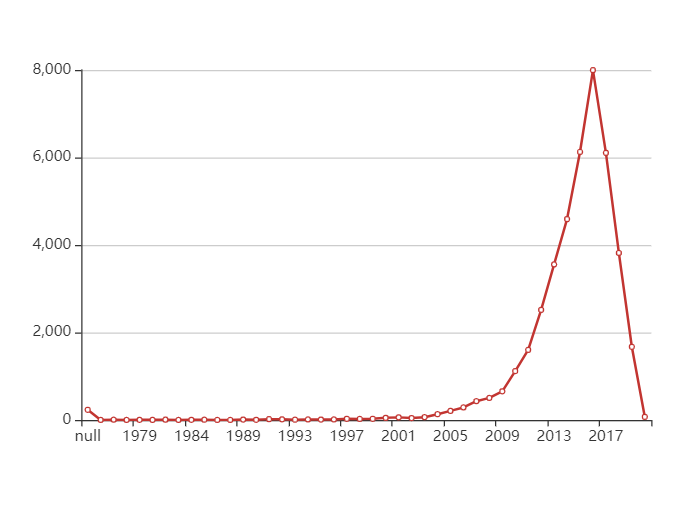
return {  
   
  
 // 查询数据 我就查四个 前两个和input绑定（双向绑定） 后两个和select绑定（通过回调函数进行修改）  
 bookname: null,  
 author: null,  
 presshouse: null,  
 fenlei: 'all',  
 discount: 80,  
 pricebetween:null,  
  
  
  
 // 表格数据  
 data: [], // 请求结果我会放进去  
 data0,  
 columns, // 刚才说到的 表的字段  
 loading: false, // 是否显示加载中图标， 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},  
 }  
 : {};  
 }  
},  
  
methods: {  
 onChangeFenlei(e) { // 分类类型改变时 修改data  
 this.fenlei = e;  
 },  
 onChangeDiscount(e) { //折扣类型改变时 修改data  
 this.discount = e;  
 },  
 onChangeBetween(e){  
 this.pricebetween = e;  
 ***console***.log(e[0]);  
 },  
 handleSubmit() { // 点击立即查询后触发  
 this.loading = true; // 和loading有关的可以先忽略  
 ***console***.log(this.fenlei);  
 ***console***.log(this.fenlei[1]);  
 // 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({  
 bookName: this.bookname,  
 author: this.author,  
 presshouse: this.presshouse,  
 smalllei: this.fenlei[1],  
 discount: this.discount,  
 min: this.pricebetween[0],  
 max: this.pricebetween[1],  
 page: this.page,  
 limit: this.limit  
 })  
  
 /\*  
 \* 这里面我用了一个 array 存储结果，  
 \* 是因为 拿到的数据不符合我的要求， 不是我想要的格式  
 \* 所以我 遍历每一条评论，并对其进行修改  
 \*  
 \* table需要的 数据格式请先用`死数据`做测试，充分了解需要的数据格式之后再进行代码编写  
 \*  
 \* \*/  
 let array = []; // 保存处理后的查询结果  
 ***axios***.post("http://localhost:8080/dangdang/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 Book = data[index] ; // 然后 通过index获取每一条评论  
 ***console***.log(Book);  
 array = array.concat({ // 将每一条处理后的结果追加到 array数组中  
 key: Book.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！  
 bookname: Book.name,  
 author: Book.author,  
 // tags: comment.vipStatus === 0 ? "普通用户" : comment.vipStatus === 1 ? "大会员" : "神秘大会员",  
  
 progress: Book.tuijian,  
 samlllei: Book.samlllei,  
 discount: '-' + (100-Book.ps\*10) + "%",  
 price: Book.pn + "￥",  
 date: Book.ptimes,  
 })  
 }  
 // 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  
 })  
  
 },

当当图书数据统计描述：



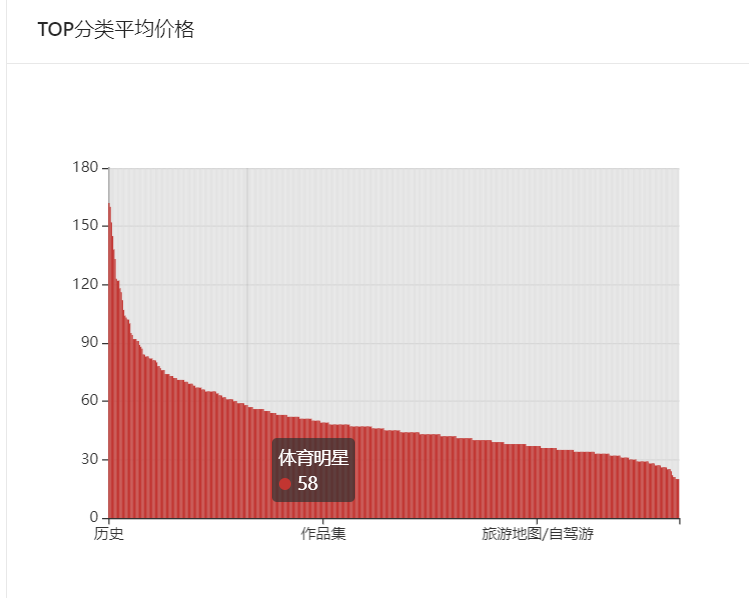
可详细查看数据库中的各个分类图书的数目和所占比

图书出版年份统计

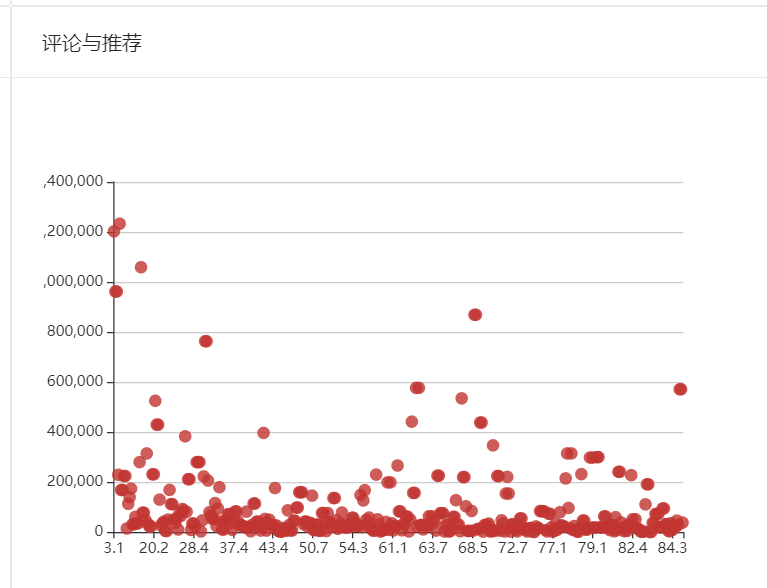


可详细查看上榜图书的出版年份统计

图书分类价格



可详细查看每一门分类图书的平均价格



可详细查看每点评分对应的评论



可详细查看年份收录的Top图书数量

数据分析后端核心代码：

@Select("select biglei as type, count(\*) as count from zhihan\_dang

group by biglei order by count desc limit 0,40")

List<dangdangData> classes();  
  
@Select("select DATE\_FORMAT(ptimes,'%Y') as type ,count(\*) as count from zhihan\_dang \n"+  
 "group by DATE\_FORMAT(ptimes,'%Y') order by DATE\_FORMAT(ptimes,'%Y')")  
List<dangdangData> prosstime();  
  
@Select("select samlllei as type , avg(pn) as count from zhihan\_dang \n"+  
 "group by samlllei order by count desc")  
List<dangdangData> avgclass();  
  
@Select("select tuijian as type ,commentNum as count from zhihan\_dang order by tuijian limit 0,400" )  
List<dangdangData> commentpr();  
  
@Select("select timehot as type, count(\*) as count from zhihan\_dang group by timehot")  
List<dangdangData> yearBook();

@Override  
public List<dangdangData> dangAnalysis(String type) {  
 if (type == null) return null;  
  
 switch (type){  
 case "classes":  
 return mapper.classes();  
 case "prosstime":  
 return mapper.prosstime();  
 case "AvgClass":  
 return mapper.avgclass();  
 case "commentpr":  
 return mapper.commentpr();  
 case "yearBook":  
 return mapper.yearBook();  
  
 }

**5 系统测试**（小三黑体，居中）

详细写明测试方法、测试描述（bug与修正方法等）、预期结果等；

查询大批量数据库缓慢，前后端引入limit 达到分页效果，预先加载首页内容，

增加新功能时，需要在首页mapping 后页mapping 和前端都进行同步修改，否则会导致数据库加载缓慢，或无法向后端发送数据，或前端无法接受数据