

# 浙江大学

## 本科实验报告

课程名称：B/S体系软件设计

姓名：姜雨童

学院：计算机科学与技术

系：计算机科学与技术

专业：计算机科学与技术

学号：3220103450

指导教师：胡晓军

2024年 12月 25日

## 浙江大学实验报告

课程名称: B/S体系软件设计 实验类型: /

实验项目名称: 商品比价网站

学生姓名: 姜雨童 专业: 计算机科学与技术 学号: 3220103450

同组学生姓名: 无 指导老师: 胡晓军

实验地点: / 实验日期: 2024年12月25日

# 开发心得

在本次商品价格比较平台的课程项目中，我从零开始学习B/S相关知识和前后端的架构、简单的爬虫知识等，独立承担了整个开发过程。诚然，这对我来说是一个不小的挑战，同时也是一个十分宝贵的学习机会。以下是我在项目过程中的一些开发体会（含反思总结）、问题及解决办法等。

## 1 开发体会

### 1.1 系统设计

本次项目，我使用前后端分离的方法，这是一种现代主流的Web开发模式，提高了系统的灵活性和可维护性。

在技术栈选择方面，我使用Vue3作为前端框架、Node.js开发后端、mysql作为使用的数据库。因为Vue3提供了响应式和组件化的开发方案，在构建用户界面的时候较为高效便捷，同时Node.js和Vue3联合开发的生态较为成熟，有活跃的社区可供交流学习。

在数据库的设计方面，最终项目使用了三张表（建库建表的脚本文件如下）：`users`、`products`和`stars`，分别记录用户信息、商品信息和用户收藏商品的信息。但是在后续开发的过程中，随着项目功能不断完善（例如呈现历史价格走势，商品降价提醒等），我逐渐发现了当前数据表的不足之处：设计的表过于简单，导致多商品进行多次价格查询时会在`products`表中插入多条不同`id`的信息。这一方面造成了信息的冗余——插入的信息只有价格和时间是有效的，另一方面也造成了开发的困难——同一商品的`id`不唯一，检索商品时需要通过`id`查找商品名称或链接，再以此为依据找到所有属于该商品的条目，这也大大增加了搜索等功能的耗时。

由于发现该问题时，系统开发进程已经过半，出于时间和精力考虑，我并没有更改数据表设计，而是沿用了这一设计，这也导致系统在功能上存在缺陷（如不适合建立包含巨量商品的商品库）。但是本次项目的经历也给了我极大的教训，成为以后学习/工作的宝贵经验。

```
1 CREATE DATABASE price_comparison_db;
2
3 USE price_comparison_db;
4
5 CREATE TABLE users (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     username VARCHAR(255) NOT NULL UNIQUE,
8     password VARCHAR(255) NOT NULL,
9     email VARCHAR(255) NOT NULL UNIQUE,
10    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
11 );
12
13 CREATE TABLE products (
14     id INT AUTO_INCREMENT PRIMARY KEY,
15     name TEXT,
16     category VARCHAR(255),
17     price DECIMAL(10, 2),
18     image_url VARCHAR(255),
19     link TEXT,
20     created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
21 );
22
23 CREATE TABLE stars (
24     id INT AUTO_INCREMENT PRIMARY KEY,
25     user_id INT,
26     product_id INT,
27     price DECIMAL(10, 2),
```

```
28     date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
29     FOREIGN KEY (product_id) REFERENCES products(id),  
30     FOREIGN KEY (user_id) REFERENCES users(id)  
31 );
```

## 1.2 开发阶段

### 1.2.1 前端

由于本学期另一门课程也需要利用Vue3构建前端，因此我对前端的开发包括CSS的使用相对熟练。此外，从零开始设计页面到实现的过程能给予很强烈的正反馈，Vue3响应式设计的特点使得每次修改更新都能立刻看到界面的变化，也让我更有动力去开发简洁美观、用户交互友好的前端界面。

项目中组件页面的切换使用了vue-router库，而项目的状态管理和不同组件间参数的传递则是使用了Vuex库。在小型项目时，可以通过直接传递参数的形式在不同组件间传递信息（如用户名/邮箱等），但是在较为大型的项目中，这样做较为繁琐，也难以保证信息的一致性，而Vuex极好地解决了这一问题，方便了项目开发。

### 1.2.2 后端

本项目我使用了Node.js作为后端技术栈，支持async/await实现异步代码，增强了代码的稳定性和可读性（由于我数据库设计的不足之处，本项目中很多操作都需要不止一次在数据库中进行查询，因此需要等所有查询操作结束后再返回结果，因此异步操作格外重要）。

在实现爬虫获取购物平台商品信息时，我使用了puppeteer库来实现无头浏览器模拟用户行为，cheerio库解析相应的HTML内容。另外在实现降价商品邮件提醒功能时，我用了nodemailer库。可以说Node.js活跃的生态以及丰富的各种库为本项目的开发提供了极大的便利。

另外，为了项目评测的方便，本项目使用了推荐的mysql数据库，但是由于我自己上数据库系统这门课的时候，使用的是sql server，因此我没有接触过mysql相关的图形化界面，所以本项目相关测试文档等内容里所有需要数据库的地方，使用的都是MySQL 8.0 Command Line Client客户端窗口截图。

## 1.3 测试阶段

测试计划与用例设计：我首先制定了详细的测试计划，确定测试的目标和内容。测试要求覆盖所有功能点，如用户注册、用户登录、搜索商品、查看收藏等。然后我为每个功能点编写了详细的测试用例，包括正常场景和边缘情况。

错误处理和边缘情况：测试过程中，我特别关注了错误处理和边缘情况，如未登录直接搜索的行为、密码/邮箱格式检查等，这帮助我发现并修复了可能导致系统崩溃或异常行为的问题。通过这些测试，我增强了本项目系统的鲁棒性和用户交互体验。

## 2 问题及解决办法

项目的开发过程中我遇到了很多的问题，其中一个难点就是爬虫的实现。在用浏览器的开发者工具观察淘宝的页面HTML结构时，我发现所有结构的类名后都带有随机的乱码，查找资料后发现该内容几个小时就会自动更换，没法使用简单的爬虫获取商品信息。其他几大购物网站，如京东，也存在类似的问题，不太方便爬，因此最终选择爬取1688电商平台和苏宁易购。

在最开始，我考虑使用axios库发送HTTP请求（下图1），在解析爬取到的内容时发现网页是动态加载的，而该方法爬取到的内容只有网页加载完全前的一部分内容，并不含项目所需的商品信息（下图2，div id="app"内才是需要的商品信息）。

```

try {
  // console.log('1688 URI:', 'https://p4psearch.1688.com/page.html?hpageId=old-sem-pc-list&keywords=${encodeURIComponent(name)}'); // 测试OK
  // const response = await axios.get('https://p4psearch.1688.com/page.html?hpageId=old-sem-pc-list&keywords=${encodeURIComponent(name)}');
  // const html = response.data;
  // console.log('html:', html); // 测试OK
  // const $ = cheerio.load(html); // failed: 无法获得动态内容

  const products = [];
  const name1688 = '1688' + name;
  // 数据库内有要查询商品
  db.query("SELECT p1.* FROM products p1 WHERE p1.category = ? AND p1.created_at = (SELECT MAX(p2.created_at) FROM products p2 WHERE p2.name = ?)");
  if (err) {
    console.error('查询数据库时出错:', err);
  }
}

```

```

</script>
<div id="app"></div>
<!-- 引用页面 JS -->
p/pc-sem/1.0.159/pages/home/index.js" crossorigin /></script>
<script src="//g.alicdn.com/assets-group/jplus/0.0.13/index.js" crossorigin /></script>
<script src="//g.alicdn.com/assets-group/cbu-splus/0.1.0/index.js" crossorigin></script>
</body>
</html>

```

因此接下来我使用了无头浏览器的方式，等待要求内容到达后再获取网页内容，最终成功实现了1688爬虫。

```

try {
  const browser = await puppeteer.launch(); // 无头浏览器模拟用户行为
  const page = await browser.newPage();
  await page.goto('https://p4psearch.1688.com/page.html?hpageId=old-sem-pc-list&keywords=${encodeURIComponent(name)}', { waitUntil: 'networkidle0' });
  await page.waitForSelector('#app'); // 等待 #app 元素出现
  const appContent = await page.$eval('#app', el => el.innerHTML); // 获取 #app 元素的 HTML 内容
  // console.log('appContent:', appContent); // 测试OK
  await browser.close();
  const $ = cheerio.load(appContent);
}

```


而在爬取苏宁易购平台的时候，我发现它的网页内容是直接呈现的，并不像1688这样包在很多层容器里，导致利用 **axios** 发送请求时获取不了商品信息，因此我还是优先考虑了该方法，最终发现除了价格都能爬到（下图，价格所在的那个容器没有展开，因此获取不了价格）。而更改方法，使用无头浏览器模拟用户行为时，爬到的 **HTML** 内容变成了扫码登录界面的内容，无法直接获取商品信息。

由于实现登录后爬取对我来说还是有点困难，最终我选择了更换爬取的网站（但是后端的各种接口还是写的苏宁易购的名字，只更改了前端的呈现）。

```

</div>
</div>
<div class="res-info">
<div class="price-box">
<span class="def-price" datasku="12438005362||||0000000000" brand_id="000066138" mdmGroupId="R1901001">
</span>
</div>
<div class="title-selling-point">
<a

```



Product Image	Product Name	Price	Action
	AJAZZ 黑爵 12月30号上年 10点 AK680Max 三模磁轴键盘	99元	查看产品
	AULA 狼蛛 S99Pro无线/蓝牙/有线三模 键盘机械手感 RGB背光拼...	98.46元	查看产品
	25日20点: MCHOSE 迈从 G87 三模 机械键盘 TTC 快金轴V2...	269元	
	AULA 狼蛛 S3012 71-80 键游戏背光电竞办公 有线键盘 宝石...	44.85元	

除了爬虫外，还需到了很多其他的问题，譬如在利用 **echart** 库绘制历史价格走势时，容器的宽高必须用 **vw/vh** 才能正确渲染（下图），否则图表显示很窄。

```
<div class="price-history" ref="priceHistoryChart"></div>
```

```
.price-history {
  width: 90vh;
  height: 40vh;
  /* width: 100%;
  height: 330px; */ /* 必须要用vh才能自适应，否则图表显示很窄 */
  margin-top: 20px;
  position: relative;
  border: 1px solid #d87804; /* 测试边框 */
}
```

在部署到Docker上时问题更是层出不穷，报错记录比我命都长，单独拉出来也是这份心得的两倍不止。最后由于后端连接数据库一直有问题，选择了部署到服务器上，网址：[goodprice http://124.220.75.7:8080/](http://124.220.75.7:8080/)。然而部署的服务器似乎不支持 **puppeteer** 库（无头浏览器模拟用户行为，用来爬其他网站的商品库），因此部署到服务器上的是阉割爬虫版的，只能搜索商品库内有的商品（例如手机、茉莉花、郁金香、耳环、水杯、摩托车等）。

（真的搞docker和部署服务器都花了很长时间，不要因为这个大扣分555 > <）

文件

大纲

Markdwn

Mermaid

Docker

```

1 mysql> show variables like 'datadir';
2
3 | Variable_name | Value |
4 |-----|-----|
5 | datadir       | C:\ProgramData\MySQL\MySQL Server 8.0\Data\ |
6 |-----|-----|
7 1 row in set, 1 warning (0.06 sec)

```

授权问题:

```

1 D:\PointMe\HW\BS\24fallBS_Lab> docker-compose up -d
2 [*] Running 1/1
3 ✓ Network 24fallbs_lab-default Created 0.1s
4 - Container 24fallbs_lab-db-1 Creating 0.0s
5 Error response from daemon: Access is denied.

```

修改:

```

1 PS C:\WINDOWS\system32> icacls "C:\ProgramData\MySQL\MySQL Server 8.0\Data" /grant
  Everyone:F
2 >>
3 已处理的文件: C:\ProgramData\MySQL\MySQL Server 8.0\Data
4 已成功处理 1 个文件; 处理 0 个文件时失败

```

容器数据库还是报错:

```

1 D:\PointMe\HW\BS\24fallBS_Lab> docker logs 24fallbs_lab-db-1
2 2024-12-28 08:38:22+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server
  9.1.0-1.el9 started.
3 find: '/var/lib/mysql/#innodb_redo': Permission denied
4 find: '/var/lib/mysql/#innodb_temp': Permission denied
5 find: '/var/lib/mysql/mysql': Permission denied
6 find: '/var/lib/mysql/performance_schema': Permission denied
7 find: '/var/lib/mysql/price_comparison_db': Permission denied
8 ...

```

[Docker加载/var/lib/mysql出现Permission Denied - Mr\\_伍先生 - 博客园](#)

[chown: changing ownership of '/var/lib/mysql/': Permission denied\\_chown: changing ownership of permission denied-CSDN 博客](#)

在 Windows 系统中，不会安装或启用 SELinux (Security-Enhanced Linux)。SELinux 是专为 Linux 系  
统设计的安全模块，因此无论是在 Windows、WSL (Windows Subsystem for Linux) 还是其他  
Windows 环境中，您都不需要关闭 SELinux。

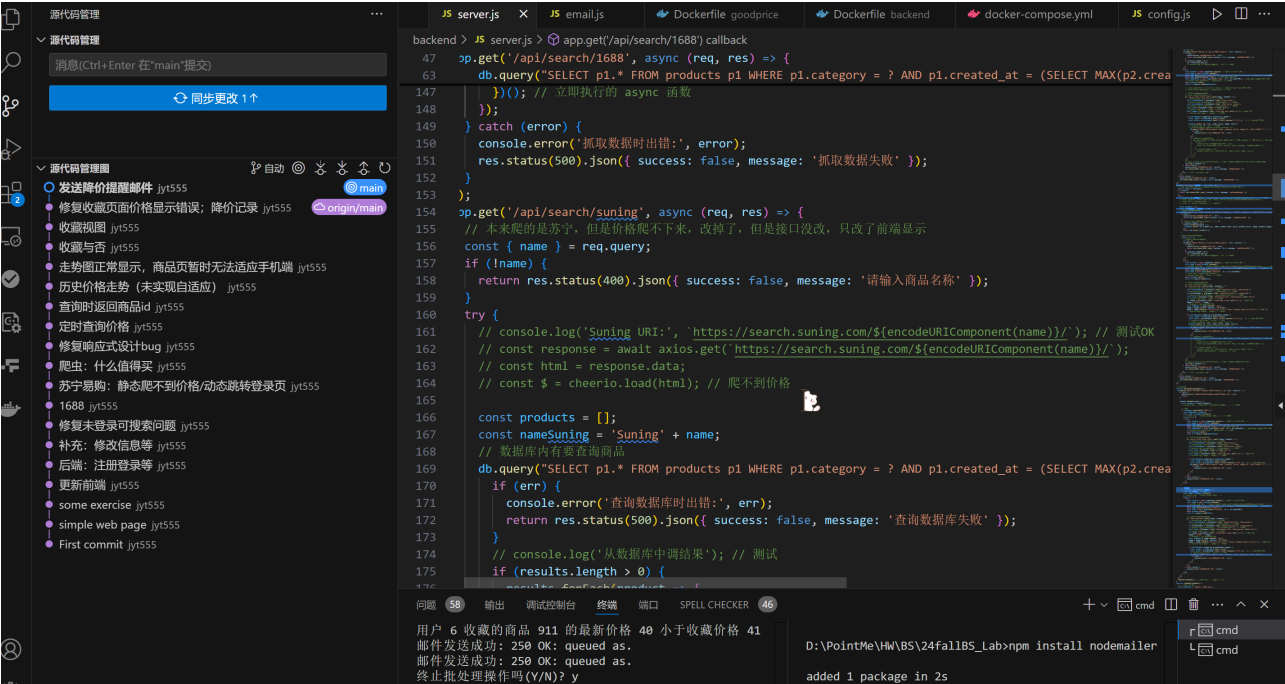
5651 词

Search		Only show running containers						
	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions	
	24fallbs_l		Running (3/3)		0.59%	13 hours ago		
	db-1	mysql:latest	Running	3306:3306	0.58%	13 hours ago		
	backenc	24fallbs_lab-9c2a68dc	Running	5000:5000	0%	13 hours ago		
	frontenc	24fallbs_lab-668d8af3	Running	8080:8080	0.01%	13 hours ago		



受限于篇幅问题，这里不对其他问题做赘述。

另外，项目内包含了 .git 提交信息，文字部分呈现如下（少量bug修复、docker部署前的截图）：



### 3 小结

不论是设计或是代码实现，本项目都存在着一定的问题，但是通过这次完整的项目开发经历，我巩固了数据库方面的知识，学习了B/S架构、前后端开发相关技术栈（Vue3、Node.js、mysql、docker-compose等）的知识。本次项目不仅提高了我的代码水平，也加强了我对前后端分离的B/S设计模式的了解，为以后的学习/工作生涯积攒了宝贵的经验。