

本科实验报告

课程名称： B/S 体系软件设计

姓 名： 张逸航

学 院： 计算机科学与技术学院

系： 计算机系

专 业： 计算机科学与技术

学 号： 3220103921

指导教师： 胡晓军

2024 年 11 月 1 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： B/S体系设计

实验项目名称： 商品比价网站

学生姓名： 张逸航 专业： 计算机科学与技术 学号： 3220103921

同组学生姓名： 无 指导老师： 胡晓军

实验地点： 曹光彪西-503 / 线上 实验日期： 2024 年 11 月 1 日

【商品比价网站】

**设计文档**

负责人：张逸航

日期：2024 年 11 月 1 日

1. **项目概述**
   1. **项目目的**

本项目的主要目的是利用Web开发技术实现一个商品比价网站。该平台将允许用户注册、登录，并提供线上商品的数据统计功能。具体要求包括实现用户认证，数据查询、统计和整理等功能。

* 1. **项目背景**

随着电子商务的快速发展，消费者面临着大量商品和价格的选择。消费者希望能够快速找到同一商品在不同商家中的价格差异，以节省时间和金钱。而网络技术和数据抓取技术的进步使得比价网站能够实时获取和更新商品价格信息，并提醒用户实时关注价格变化。

这样一个背景下，我们需要开发一个商品比价平台，该平台将允许用户轻松地注册、查看商品比价。它还将提供数据收集可视化工具，以便用户可以实时跟踪商品价格变化，还可以帮助用户实现以下目标

1、节省金钱和时间

用户可以比较不同商家的价格，找到最低的购买选项，从而节省开支。通过集中展示多个商家的价格和产品信息，用户可以迅速找到所需商品，避免逐一访问不同网站。

2、获取全面信息做出明智决策

比价网站通常提供产品的详细信息、用户评价和评分，帮助用户更好地了解商品的性能和质量。用户可以比较多个产品的特点和价格，帮助他们在购买时作出更符合自身需求的选择。

3、了解市场趋势

用户可以通过比价网站了解某一产品的市场价格变化，从而判断其购买时机。

总之，商品比价平台的开发将有助于实现更智能、高效的商品比价，为用户提供更好的购物体验。

* 1. **设计要求**
     1. **基本功能**

1. 实现用户注册、登录功能，用户注册时需要填写必要的信息并验证，如用户名、密码要求在6字节以上，email的格式验证，并保证用户名和email在系统中唯一，用户登录后可以进行以下操作。
2. 通过商品名称在主流电商平台上查询该商品实时价格
3. 商品名称建议分词处理优化查询；
4. 查询多个结果的处理
5. 很多平台需要平台用户登录验证后才可以进行查询
6. 支持至少2个以上平台查询价格进行比较（淘宝、京东等）。
7. 建立商品库，将商品信息和商品价格保存在数据库中。商品信息包含名称、多级品类、规格、条码、图片等，方便后续查询。
8. 提供商品查询界面能显示商品信息，把历史价格用图表形式显示（如价格走势图）。
9. 支持设置降价提醒，针对指定商品定时查询最新价格，如有降价发送提醒，可以通过邮件，App推送等方式实现。
10. 样式适配手机，开发手机App或能够在手机浏览器/微信等应用内置的浏览器中友好显示。
    * 1. **增强功能：**

如开发手机端，可以用相机拍摄商品图片或扫码商品条码进行商品查询。

* + 1. **安全需求**

1. **保密性**

* 用于身份验证的用户名和密码应防止未经授权的用户访问系统。
* 某些敏感数据（如用户名，密码）在交换时应加密。密码在存储之前应加密。
* 在用户登录期间，应该防止 SQL 注入，密码强制破解和伪造会话入侵。
* 应构建访问控制以防止合法用户非法使用系统资源。

1. **完整性**

* 防止非法用户对数据进行无意或恶意的修改、插入、删除，防止数据丢失。

1. **约束性**

* 为数据库加上一定的约束，对关键性操作如删除、修改进行限制，并对用户进行警示。
  + 1. **性能需求**
* 能保证在主流的浏览器能够正常运行
* 在高访问量时仍能保证并发运行（至少50人次的并发访问不会致使崩溃）
* 系统应保证运行稳定，避免出现崩溃
* 保证网页的加载时间不会过长

1. **技术选型**
   1. **技术栈**

**·前端开发**

**· 前端基础技术**：HTML/CSS/JavaScript

· **框架：**总体上采用Vue前端框架来构建前端项目

· **辅助技术：**Element UI（CSS样式库） Axios（HTTP 库） Echarts（数据可视化图表库）

· **移动端适配：**使用响应式设计或CSS框架实现手机端适配

·**后端开发**

**· 编程语言：**python

· **Web框架：**django

· **数据库：**Mysql

1. **架构设计**
   1. **总体结构图**
   2. **前端架构**
      1. **前端技术总览**

**·Vue：**作为前端主要框架，用于构建用户界面。

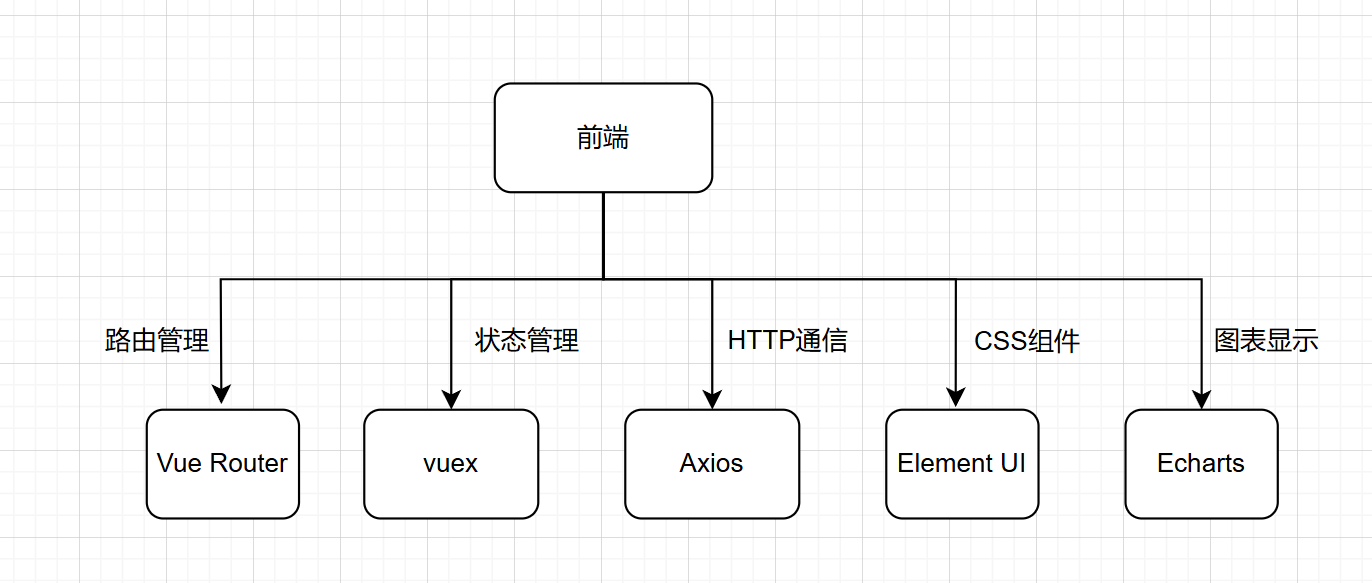
·**Vue Router：**用于处理前端路由。

·**Vuex：**用于状态管理（存储全局信息）。

·**Axios：**用于与后端API进行通信。

·**Element UI：**用于UI组件和响应式设计（CSS样式）。

·**Echarts：**用于绘制统计图表。



* + 1. **前端模块**

1. **用户认证模块**

**· 用户注册：**提供注册界面，要求用户输入用户名、密码、邮箱等信息。前端通过API发送注册请求到后端。

**· 用户登录：**提供登录界面，用户输入用户名和密码进行登录。前端通过API发送登录请求到后端，并在成功登录后接收和存储令牌。

1. **商品管理模块**

**· 商品列表：**展示用户所需要查询的商品列表，包括商品名称、对应平台及价格等信息。

**· 创建商品：**提供界面以创建用户所需要查询的商品，前端通过API发送创建请求到后端。

**· 商品编辑：**用户可以编辑商品名称，前端通过API发送编辑请求到后端。

**· 删除商品：**用户可以删除商品，前端通过API发送删除请求到后端。

1. **数据查询和可视化模块**

**· 数据查询：**用户可以选择平台并查询所选择商品的数据。前端发送查询请求到后端，后端查询后返回相应的数据。

**· 数据可视化：**使用数据可视化库（Echart库），将查询到的数据绘制成图表，以帮助用户分析和理解数据。

1. **统计信息模块**

**· 统计信息：**在首页上展示统计信息，包括商品信息，历史价格数据等。使用数据可视化库来绘制图表。

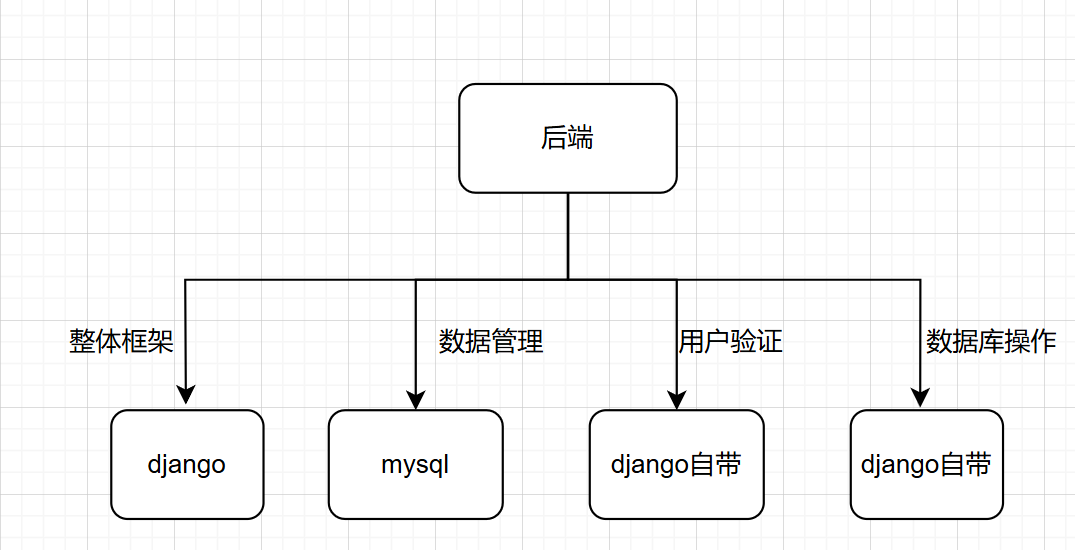
* 1. **后端架构**
     1. **后端技术总览**

**· Spring Boot：**作为后端的主要框架，用于响应和处理服务请求。

**·** **django自带的用户验证：**用于用户认证和授权。

**·** **django自带的ORM：**用于与数据库交互。

**· MySQL数据库：**用于存储用户信息、商品信息和数据记录。



* + 1. **后端模块**

1. **用户认证和授权模块**

**· 用户认证：**使用django自带的用户验证一同进行用户认证和授权。处理用户注册、登录和颁发JWT令牌。

1. **商品管理模块**

**· 商品管理：**使用django管理商品信息的创建、编辑和删除。

· **商品信息API：**提供django以支持前端商品管理操作。

1. **数据查询和统计模块**

**· 数据查询API：**使用python爬取主流电商平台商品信息，提供django以支持前端数据查询操作。

**· 统计信息API：**提供django以获取商品统计数据。

**（4）降价提醒模块**

**· 提醒信息：**当指定商品降价一定幅度时，会通过发送邮件的方式提醒用户商品降价。

1. **数据库设计**

基于项目要求，需要设计数据库表来存储用户信息、商品信息和历史价格记录。

* 1. **数据表设计**
     1. **User表**

**·**该表用来保存用户的相关信息

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| id | INT | 用户的id | 主键、非空、自增 |
| username | VARCHAR(255) | 用户名 | 非空，且唯一 |
| password | VARCHAR(255) | 用户的密码 | 非空，加密存储 |
| email | VARCHAR(255) | 用户的邮箱 | 非空，且唯一 |
| phone | VARCHAR(15) | 用户的手机号 | 可以为空 |

* + 1. **Goods表**

**·**该表用来保存商品的相关信息

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| good\_id | VARCHAR(15) | 商品的id | 主键、非空、自增 |
| good\_name | VARCHAR(255) | 商品名称 | 非空 |
| good\_description | VARCHAR(255) | 商品描述 | 非空 |
| good\_scale | VARCHAR(255) | 商品规格 | 非空 |
| good\_type | VARCHAR(255) | 商品品类 | 非空 |
| good\_code | BLOB | 商品条码图 | 非空 |
| good\_pic | BLOB | 商品图片 | 非空 |

* + 1. **Log表**

**·**该表用来保存和商品相关的信息（此表后期可能会有所改动，仅供参考）

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| message\_id | INT | 信息记录的唯一标识符 | 非空，主键，自增 |
| good\_id | VARCHAR(15) | 关联的商品ID | 非空，外键（参考商品数据表） |
| timestamp | DATETIME | 信息记录的时间戳 | 非空 |
| Price | FLOAT | 商品的最新价格 | 非空 |

* + 1. **User-Good表**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **类型** | **描述** | **备注** |
| message\_good-id | INT | 信息记录的唯一标识符 | 非空，主键，自增 |
| good\_id | VARCHAR(15) | 用户查询的商品ID | 非空，外键（参考商品数据表） |
| user\_id | VARCHAR(15) | 查询的用户 | 非空，外键（参考用户数据表） |

1. **接口设计**

在该项目中，前端采用Axios 库与后端数据请求和交互，后端通过设计若干RESTful 的增删查改 api 供前端调用，其中具体的接口设计如下（此部分在后期可能会有修改，这里仅供参考）：

* 1. **用户相关接口**

用户相关的接口主要提供登录、注册、对User表进行增删改查的功能，其的接口 URL 均以 /user 开头。

* + 1. **用户登录**

|  |  |
| --- | --- |
| **请求路径** | /user/login |
| **请求方法** | POST |
| **请求参数** | ·username (String) - 用户名  ·password (String) - 密码 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息  ·user\_id (Int) - 用户的id  ·username (String) - 用户名  ·email (String) - 邮箱  ·phone (String) - 电话号码  ·token (String) - 令牌 |
| **接口简介** | 用户通过提供用户名和密码进行身份验证，成功登录后返回用户信息，并将token存储在客户端 |

* + 1. **用户注册**

|  |  |
| --- | --- |
| **请求路径** | /user/register |
| **请求方法** | POST |
| **请求参数** | ·username (String) - 用户名  ·password (String) - 密码  ·email (String) - 邮箱  ·phone (String) - 电话号码 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息  ·user\_id (Int) - 用户的id  ·username (String) - 用户名  ·email (String) - 邮箱  ·phone (String) - 电话号码 |
| **接口简介** | 用户通过提供用户名、密码、邮箱和电话号码进行注册，成功注册后返回用户信息以及生成的令牌（前端实现自动跳转） |

* + 1. **修改密码**

|  |  |
| --- | --- |
| **请求路径** | /user/updatePassword |
| **请求方法** | POST |
| **请求参数** | **Header：**  "Authorization" : token  **Body:**  ·username (String) - 用户名  ·old\_password (String) - 旧密码  ·new\_password (String) - 新密码 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息 |
| **接口简介** | 用户通过提供令牌以及旧密码和新密码来修改密码。 |

* 1. **商品相关接口**

商品相关的接口主要提供对商品信息的增删查改功能，这一类接口的 URL 都

用/good 开头。

* + 1. **获取某个商品信息**

|  |  |
| --- | --- |
| **请求路径** | /good/{good\_id} |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·good\_id (String) - 目标商品的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·good\_id (String) - 商品的id  ·user\_id (Int) - 用户的id  ·good\_name (String) - 商品名称  ·good\_description (String) - 商品描述  ·good\_scale(String)-商品规模  ·good\_type(String)-商品类别  ·good\_code(ImageField)-商品条码  ·good\_pic(ImageField)-商品图片  ·good\_price(Float) - 商品价格  ·last\_update(Date) - 商品价格上次更新时间 |
| **接口简介** | 获取特定商品的详细信息。 |

* + 1. **查询用户关注商品列表**

|  |  |
| --- | --- |
| **请求路径** | /good/user/{user\_id} |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·goods (Array) - 包含商品信息的数组  ·good\_id (String) - 商品的id  ·good\_name (String) - 商品名称  ·good\_description (String) - 商品描述  ·good\_price(Float) - 商品价格  ·good\_scale(String)-商品规模  ·good\_type(String)-商品类别  ·good\_code(ImageField)-商品条码  ·good\_pic(ImageField)-商品图片  ·last\_update(Date) - 商品价格上次更新时间 |
| **接口简介** | 获取特定用户关注的商品列表。 |

* + 1. **条件查询商品列表**

|  |  |
| --- | --- |
| **请求路径** | /good/search-with-pagination |
| **请求方法** | GET |
| **请求参数** | **Header：**  "Authorization" : token  **Param:**  ·user\_id (Int) - 用户的id（不为空）  ·good\_id (String) - 商品的id (可为空)  ·good\_name (String) - 商品名称(可为空)  ·current (Int) - 当前页  ·size (Int) - 每页的大小 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·goods (Array) - 包含商品信息的数组  ·good\_id (String) - 商品的id  ·good\_name (String) - 商品名称  ·good\_description (String) - 商品描述  ·good\_price(Float) - 商品价格  ·last\_update(Date) - 商品价格上次更新时间  ·good\_scale(String)-商品规模  ·good\_type(String)-商品类别  ·good\_code(ImageField)-商品条码  ·good\_pic(ImageField)-商品图片  ·totalCount (Int) - 符合条件的数据总数  ·currentPage (Int) - 当前页  ·pageSize (Int) - 页的大小 |
| **接口简介** | 根据查询表单的相关字段和用户ID的值搜索符合条件商品列表数据（注意：表单中所有字段均可为空，后端需要作相应处理，为NULL或者空字符串的字段不作为查询条件，要考虑分页处理）。同时，根据商品名称搜索可以搜索到相近的结果（不要求完全匹配而是like） |

* + 1. **新增商品**

|  |  |
| --- | --- |
| **请求路径** | /good/add |
| **请求方法** | POST |
| **请求参数** | **Header：**  "Authorization" : token  **Body:**  ·user\_id (Int) - 用户的id  ·good\_name (String) - 商品名字 |
| **响应数据** | ·status (Int) - 响应码（响应状态）  ·message (String) - 操作结果消息 |
| **接口简介** | 创建一个新商品。 |

* + 1. **删除商品**

|  |  |
| --- | --- |
| **请求路径** | /good/delete/{good\_id} |
| **请求方法** | DELETE |
| **请求参数** | **Header：**  "Authorization" : token |
| **响应数据** | ·status (Int) - 响应码（响应状态）  · message (String) - 操作结果消息 |
| **接口简介** | 删除相应的商品 |