本科实验报告

课程名称: B/S 体系软件设计
姓名: 陈爽
学院：计算机科学与技术学院
专业：计算机科学与技术
学号：3220105511
指导老师：胡晓军

2024年10月15日

商品比价网站

1　引言

1.1　编写目的

本《系统设计计划》是根据实验的具体目的和要求，在深入分析并查看了客户端提供的相关文件后，制定的详细系统设计结构及其说明文件。该计划旨在全面描述系统的设计流程，确保设计目标明确、需求清晰，能够为后续的系统开发提供坚实的基础和指导。

计划的主要内容包括引言、需求功能分析、运行环境、系统结构设计、数据处理以及系统实现的相关说明。通过这些章节，读者可以全面了解本系统的设计背景、预期功能、技术选型及架构设计等重要内容。系统设计部分详细说明了整体架构，包括各个子系统的设计原则、模块划分以及接口定义，确保各部分功能模块之间的有效协同。此外，还特别关注了系统的出错处理机制，详细阐述了错误检测、错误恢复及日志记录等方面的设计，确保系统的稳健性和容错能力。

为了确保计划的可行性和项目的顺利实施，本计划还制定了详细的时间表和里程碑，涵盖各个阶段的开发任务及预期完成时间。这将有助于团队合理分配资源，确保项目按照既定的进度高效推进，并为后续的开发、测试和发布提供重要的依据。通过这一系统设计计划，可以为后续的项目实施提供明确的路线图和操作指引，确保系统设计和开发工作有序、高效地进行。

1.2　文档范围

本实验文档旨在设计并实现一个功能全面的 Web 商品价格比较网站，涵盖了用户注册与登录、商品价格实时查询、多平台价格对比、历史价格追踪、降价提醒等核心模块。文档内容包括系统的功能需求分析、详细的技术实现方案、数据库设计、前后端交互、以及界面设计等多个方面。通过对商品名称的分词处理、多平台接口的集成查询，网站能够实时获取多个电商平台（如淘宝、京东等）的商品价格，并为用户提供直观的价格比较功能。系统还通过建立商品数据库，记录商品的详细信息及其历史价格变化，并以图表形式展示，帮助用户进行长期价格跟踪和分析。此外，网站支持用户设置个性化的降价提醒功能，系统会定期查询指定商品的最新价格，并通过邮件或推送的方式通知用户，以提升购物的便利性和精准性。考虑到移动端用户的需求，系统界面设计将适配手机浏览器，提供无缝的移动端体验，用户也可以通过扫描商品条码或拍摄商品图片来查询商品信息。

该项目的背景是源于电商购物用户对比价格的需求日益增加，用户往往需要在多个平台之间切换，以找到最具竞争力的价格。通过这一项目的开发，用户可以在同一平台上同时获取多个电商平台的价格信息，简化了价格比较过程，并通过个性化提醒功能，帮助用户在商品降价时及时购买，提升整体购物体验。本文档为开发、测试及维护团队提供了完整的技术规范和设计方案，确保系统在功能上满足用户需求并具有较高的可扩展性和可维护性。

1.3 读者对象

本说明书的预期读者人员包括软件用户，项目经理、软件开发人员、软件测试人员、系统维护人员等

1.4 术语及缩写解释

术语	解释
IPO图	IPO图是输入/处理/输出图的简称，描述输入数据、对数据的处理和输出数据之间的关系。
状态图	状态图描述一个系统或组件可能假设的状态，并显示引起或导致一个状态切换到另一个状态的事件或环境。
类图	类图描述了一个软件系统中的类、接口、关系以及它们之间的静态结构。类图用于表示系统中的对象、类及它们之间的关系，提供了一种直观、可视化的方式来表示系统的结构。
MVC	MVC（Model-View-Controller）是一种设计模式，常用于开发Web应用程序。它将一个应用程序分为三个部分：模型、视图和控制器。其中，模型是应用程序的核心，表示应用程序的数据和业务逻辑；视图负责展示数据给用户；控制器负责接收用户输入并调用相应的模型和视图。

2 系统设计

2.1 项目说明

- 项目名称：商品比价网站
- 任务提出者：浙江大学 B/S 体系软件设计任课老师-胡晓军
- 开发者：浙江大学计算机科学与技术系专业学生陈爽
- 使用平台：淘宝，京东，天猫
- 用户群：购买商品比较价格的群众

2.2 需求分析

2.2.1 实验要求

需要实现的基本功能如下:

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
   (a) 商品名称建议**分词处理优化查询**;
   (b) 查询多个结果的处理
   (c) 很多平台需要**平台用户登录验证后才可以进行查询**
3. 支持至少 2 个以上平台查询价格进行比较（淘宝、京东等）。
4. 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。
5. 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）。
6. 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。
7. 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。

增强功能:

8. 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。

为了提交作业方便，项目使用的数据库，建议使用 mysql，提交作业时同时附带建库建表的脚本文件或数据

2.2.2 功能解析

1. **实现用户注册、登录功能**
   - 用户注册时需填写必要信息，并进行验证，确保用户名和密码符合规定。**密码长度要求至少 6 字节**，同时验证 email 格式，保证用户名和 email 的唯一性。登录后，用户可以访问系统的其他功能。
2. **商品实时价格查询**
   - 用户通过输入商品名称，查询主流电商平台上该商品的实时价格。为了提高查询效率，商品名称建议进行**分词处理**，优化搜索结果。此外，需处理多个平台返回的查询结果，并显示给用户。部分平台可能要求用户**登录电商平台**后才能进行查询，因此系统应支持平台登录验证。
3. **多平台价格比较**
   - 系统至少支持从两个以上的电商平台（如淘宝、京东等）查询价格，并提供价格比较功能，帮助用户轻松找到最具优势的价格。
4. **建立商品数据库**
   - 系统需建立商品库，将商品的名称、多级品类、规格、条码、图片等详细信息保存至数据库中，并实时存储各平台查询到的价格，便于后续查询与分析。
5. **商品信息展示及历史价格查询**

- 用户在商品查询界面中能够查看商品的详细信息，同时系统通过图表形式展示商品的历史价格走势，方便用户跟踪价格变化。

6. **降价提醒功能**

   - 用户可以设置降价提醒，系统将针对指定商品定时查询最新价格，一旦价格下降，系统会通过**邮件或 App 推送**通知用户，以便用户及时作出购买决策。

7. **移动端适配**

   - 系统的界面样式需适配手机端，保证用户在手机浏览器或微信等内置浏览器中能流畅、友好地使用系统功能。同时，支持开发专门的手机 App 以提供更加便捷的用户体验。

8. **商品图片拍摄与条码扫描**

   - 如果开发手机端 App，用户可以通过手机相机拍摄商品图片或扫描商品条码进行商品查询，进一步提高查询的便捷性与用户体验。

2.3 运行环境

2.3.1 *软件层面*

本网站需具备一定的负载能力，确保至少支持 100 人同时在线访问。同时，系统需具备良好的数据存储能力、网络服务吞吐能力以及数据安全特性，以满足高并发访问需求。系统还需兼容 Android 和 iOS 移动终端，并提供对外服务所需的安全保障措施。

在客户端浏览器的支持方面，网站需能够兼容主流浏览器，如 IE、Chrome、Firefox、Opera 等，确保用户在这些浏览器中能够顺畅浏览和使用。此外，网站应兼顾其他浏览器的兼容性，即使在使用不常见的浏览器时，仍能正常体验主要功能。网站的样式设计需适配手机端，确保在手机浏览器或微信等内置浏览器中也能友好显示，为移动端用户提供流畅的使用体验。

- 软件环境：
  - Web 服务器：使用 Nginx 1.10 或以上版本作为 Web 服务器；
  - 前端框架：使用 Vue.js 3.0 或以上版本作为前端框架；
  - 后端框架：使用 django 5.1.2 或以上版本作为后端框架；
  - 数据库：使用 MySQL 5.7 或以上版本作为系统的数据库；mysql, mysqlclient(用来方便python和数据库通信)
  - 版本控制工具：Git。
  - 组件库：magic_UI， Aceternity UI， cult_UI
- 网络环境：使用 HTTP 协议对系统进行传输。

2.3.2 *硬件层面*

- 操作系统：MacOS 15.0.1 (24A348)
- 芯片：MacOS M2 Pro
- 其他硬件能够满足正常需求即可

2.3.3 开发过程

o 遵循一定的代码规范，包括变量命名、函数命名、代码注释等；

o 避免使用过时、不安全或存在漏洞的代码库和组件；

o 严格控制代码质量，避免出现内存泄漏、缓冲区溢出、代码注入等问题；

o 采用版本控制工具，如 Git 等，进行代码管理和协作开发。

2.3.4 系统配置

系统要具有良好的反应速度，课题要求在良好的网络情况下，本系统应该具有如下时间特性要求：单个用户在线时：1. Web 响应用户动作时间小于 1 秒。2. 信息搜索操作响应用户动作时间小于 2 秒。500 个用户同时在线时：1. Web 响应用户动作时间小于 2 秒。2. 信息搜索操作响应用户动作时间小于 5 秒。

2.3.5 安全性需求

1. **保密性**
   1.1 用户名和密码用于身份验证，必须防止未经授权的用户访问系统，尤其是防止未授权用户执行管理员相关操作。
   1.2 系统应建立合理的访问控制机制，防止有权限的管理员跨越权限范围使用系统资源。在敏感数据交换前，数据应加密处理，密码存储时应采用加密方式。
   1.3 在用户登录过程中，系统需防范SQL注入、密码强制破解以及伪造会话入侵等安全威胁。

2. **完整性**

   - 系统应防止非法用户无意或恶意地修改、插入或删除数据，避免数据丢失，确保数据的完整性。

3. **约束性**
   3.1 对数据库关键操作（如删除、修改等）应施加必要的约束，提示用户并限制无权限的操作。
   3.2 不同身份的用户应根据权限范围进行相应操作，确保操作不超越用户权限。

4. **用户信息安全性**
   4.1 设计需特别关注账户信息安全，确保外部人员无法入侵系统。
   4.2 内部操作应留下审计痕迹，系统管理层应定期或不定期维护系统，确保操作安全性。

5. **访问控制**

   - 系统应实现访问控制机制，确保仅授权用户能够访问系统资源。这可以通过身份验证（如用户名和密码）及授权机制（如访问令牌或角色管理）来实现，确保用户只能访问特定的资源。

6. **数据加密**

   - 系统应使用加密技术来保护敏感数据的机密性和完整性。传输过程中应采用SSL/TLS协议加密数据，防止未经授权的用户访问。在存储过程中，敏感数据（如密码）需使用加密算法进行保护，防止数据泄露。

7. **防止攻击**

   - 系统应采取必要的安全措施，防止各种攻击手段，如SQL注入、密码强制破解、会话劫持等。可以通过安全编码实践、防火墙和入侵检测系统等技术手段来实现系统的安全防护。

8. **安全审计**

- 系统应记录所有用户的操作行为，以便在发生安全事件时进行审计和调查。可以使用日志记录和审计工具来跟踪系统内发生的所有事件，帮助管理员发现并分析潜在的安全问题。

2.3.6 可维护性需求

作为一个成熟的系统，在开发初期应充分考虑系统的可维护性。

1. **高内聚、低耦合的系统模块划分**

   - 系统模块设计应做到高内聚、低耦合，确保模块内部的结构紧密且功能一致，同时保证模块间的独立性。这种设计可以简化后续的维护和升级工作，减少模块间的依赖，提高系统的可扩展性和可维护性。

2. **完备、清晰、可读的文档**

   - 文档是软件可维护性的关键因素之一。一个好的文档应具备简明性、统一的书写风格，确保系统的可读性和可修改性。文档应涵盖系统设计、功能说明、操作指南等方面，便于开发、维护和操作人员快速查阅。交付时，文档应齐全、说明详尽，且符合相关标准。

3. **良好的编程风格**

   - 程序代码应保持一致的编程风格，具有详细的注释和清晰的结构。注释应明确说明代码的功能和逻辑，便于调试和测试人员快速定位错误。具体要求包括：避免使用不确定或模糊的代码；采用有意义的变量名和函数名；合理编写注释；使用模块化、结构化的设计方法；确保文档的正确性、一致性和完整性。

4. **严谨的单元测试**

   - 核心模块应编写单元测试，确保模块之间以及系统整体的正常运作。测试要求模块化、结构清晰、易于理解并具有可靠性。测试过程中应显示中间结果，清晰说明系统的输出，并根据需求展示所有输入数据，以确保各模块的正确性。

5. **可扩展性**

   - 系统应具备良好的可扩展性，便于未来添加新功能或模块。通过使用模块化和面向对象的设计方法，可以轻松扩展新类和方法，保证系统能够随业务需求的变化进行灵活扩展。

6. **易于维护的代码**

   - 代码编写应遵循简洁、可读性强的规范，确保后续的修改和更新工作能够快速完成。清晰的代码结构和一致的编写风格能帮助开发人员更容易理解代码，提升维护效率。

7. **自动化测试**

   - 系统应具备自动化测试能力，在修改或更新系统时，能够通过自动化测试工具进行全面测试，确保系统的稳定性和正确性。可使用适当的测试框架，便于开发人员快速验证代码变更。

8. **版本控制**

   - 系统应使用版本控制工具（如Git、SVN）来管理代码和文档，便于跟踪和管理系统的版本和变更历史。这种机制不仅有助于维护代码的完整性，还能方便团队协作开发，确保开发过程中的高效性和一致性。

3 系统功能模块设计

3.1 用户模块

- **用户管理模块**
  - 用户注册
  - 用户登录
  - 修改密码
  - 查看个人信息
  - 修改个人信息
  - 用户登出

3.2 商品管理模块

- **商品库模块**
  - 查看商品信息
  - 添加商品信息
  - 修改、删除商品信息
  - 上传商品图片、条码
- **商品价格查询模块**
  - 输入商品名称查询价格
  - 分词处理优化商品查询
  - 查询商品在多个平台的实时价格
  - 比较各平台的商品价格

3.3 价格跟踪模块

- **历史价格模块**
  - 查看商品历史价格
  - 价格变化趋势图表展示
  - 保存并记录每次查询的价格数据
- **降价提醒模块**
  - 设置商品降价提醒
  - 定时查询商品最新价格
  - 通过邮件、App 推送降价通知

3.4 平台管理模块

- **多平台支持模块**
  - 配置支持的电商平台（淘宝、京东等）
  - 平台用户登录验证
  - 平台 API 数据处理和接入
  - 管理平台账号

3.5 数据统计模块

- **价格数据统计模块**
  - 统计商品价格变化情况
  - 分析不同平台的价格波动
  - 生成商品价格分析报告

4 技术架构

4.1 技术结构汇总

- **前端**:
  - 框架：Vue3
  - 路由管理：Vue Router
  - 组件库：Element Plus
  - 数据可视化：ECharts
  - 加密：jsencrypt
  - 状态管理：Vuex 或 Pinia
- **后端**:
  - 框架：Django
  - 数据库：MySQL
  - 缓存：Redis
  - 密码加密：Django 自带加密机制
  - 数据传输：RESTful API + JWT
- **其他**:
  - 对象存储：阿里云 OSS / AWS S3
  - 安全保障：SSL / TLS

4.2 前端技术栈

1. **框架：Vue3**

- Vue3 被选为前端框架，主要原因在于其现代化的响应式语法简洁且易于上手，社区支持良好，拥有丰富的中文文档，且具有很强的生态系统。Vue3 的 Composition API 提供了更灵活的组件逻辑组织方式，适合开发复杂的商品价格比较网站。

- 配套工具

  - **Vue Router**：用于实现前端的路由管理，便于用户在页面之间无刷新跳转，提升用户体验。

  - **Vuex 或 Pinia**：用于全局状态管理，方便管理用户登录状态、商品信息等共享数据。

2. **组件库：Element Plus**

- Element Plus 是 Vue3 的官方组件库之一，提供了丰富、成熟的 UI 组件，涵盖了表单、按钮、导航、弹框等各种常用的功能模块。使用该组件库可以快速搭建出美观且功能齐全的用户界面，提升开发效率。

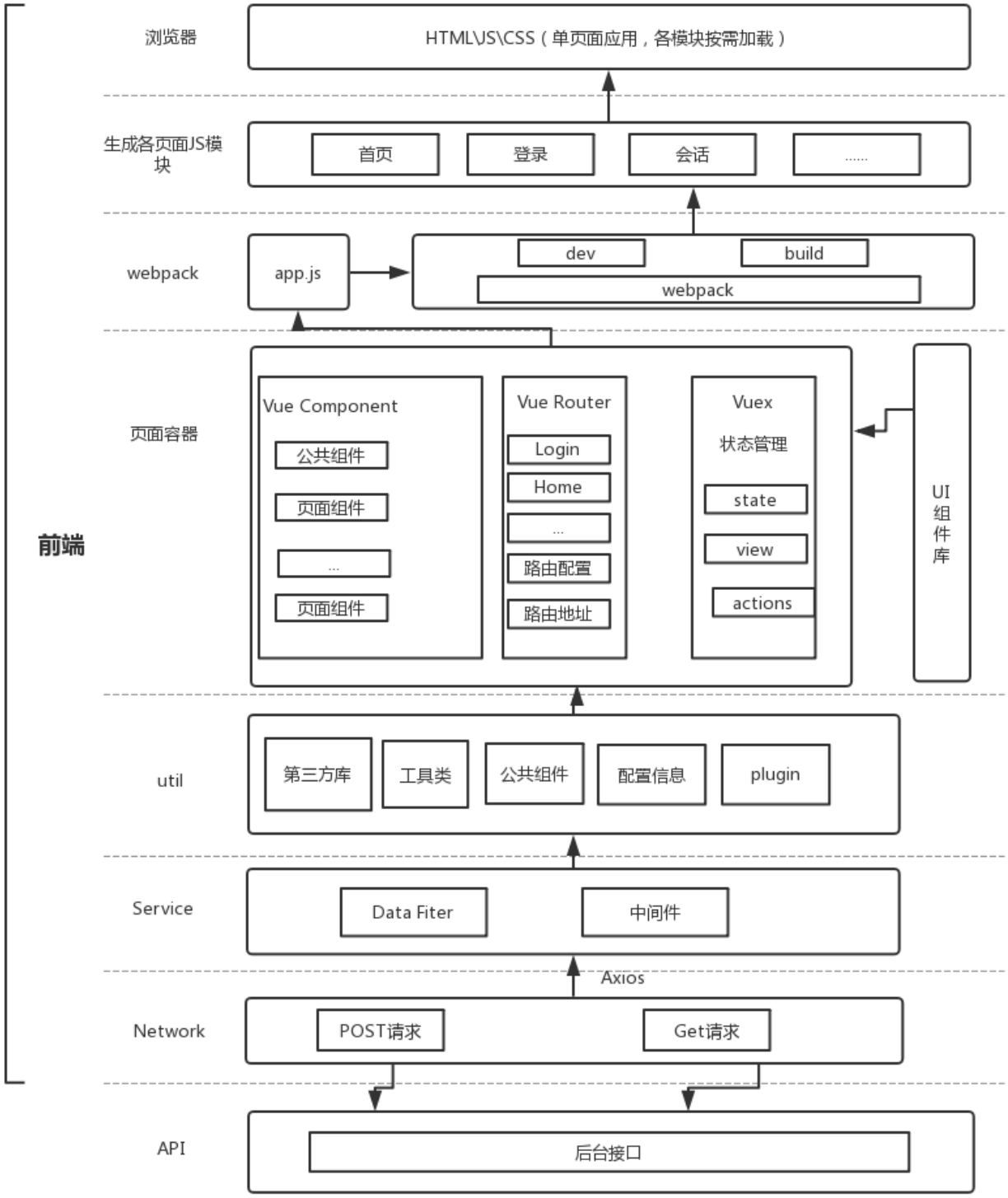
3. **前端加密：jsencrypt**

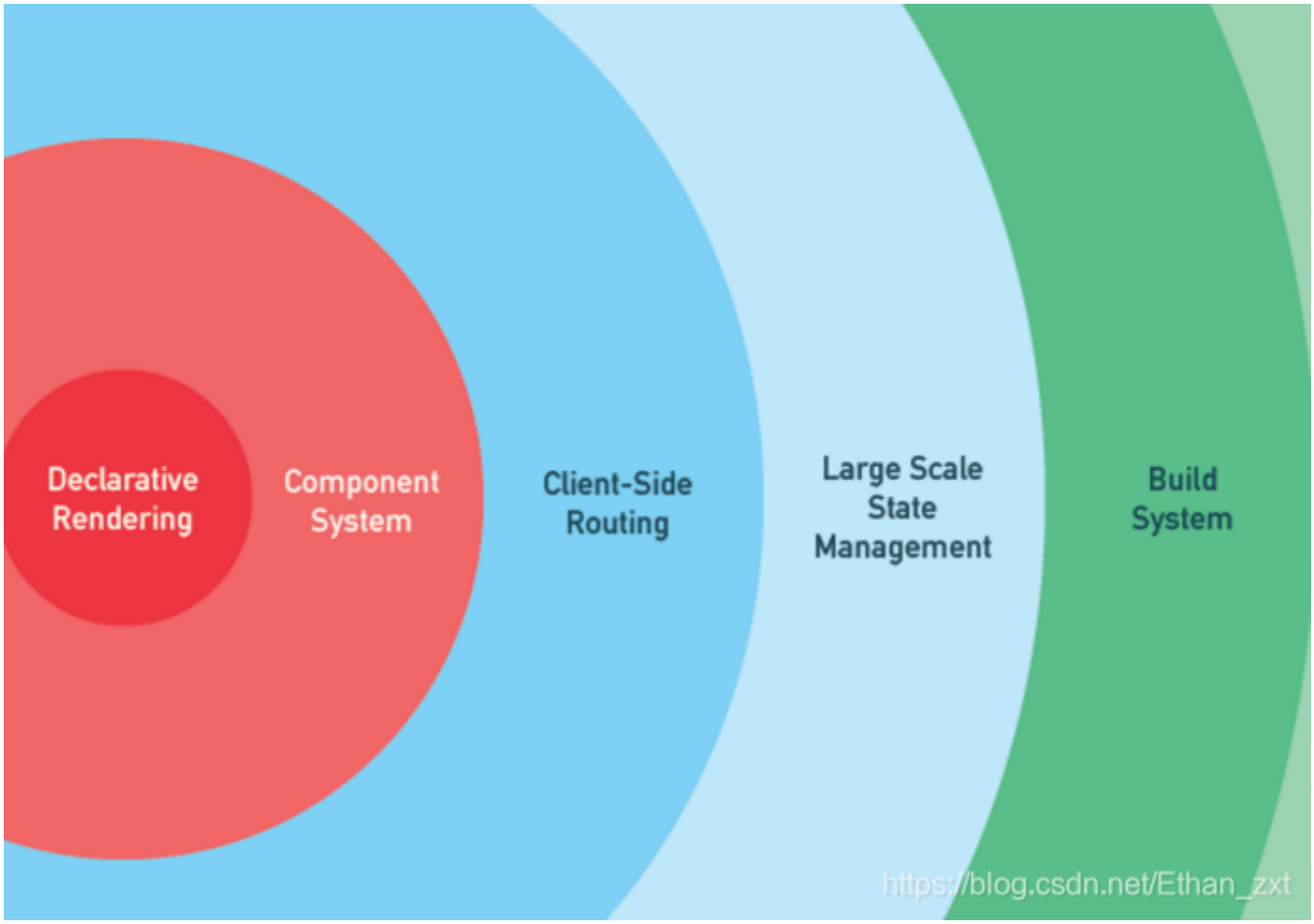
- jsencrypt 是一个 RSA 加密库，前端使用它对用户敏感信息（如密码）进行加密处理，确保在信息传输到后端时的安全性，防止密码在传输过程中被窃取。

4. **数据可视化：ECharts**

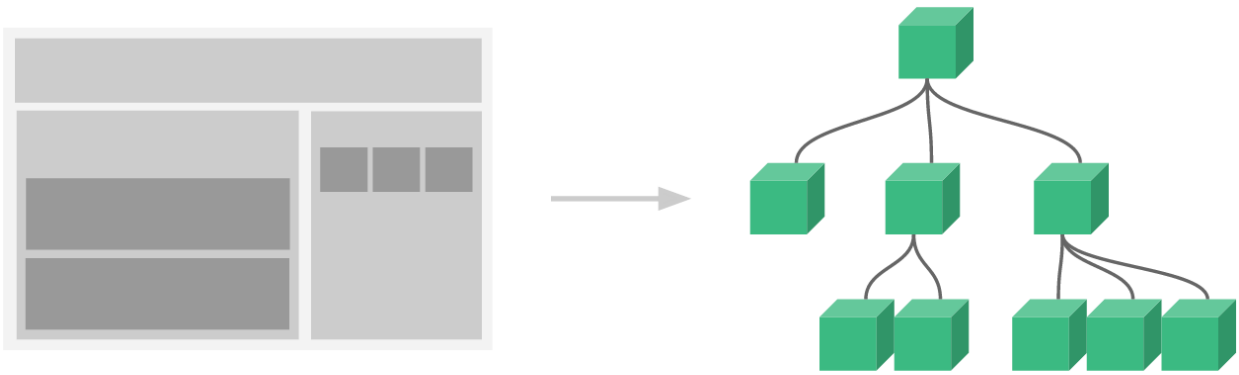
- 为了实现商品价格的历史趋势图，使用 ECharts 作为图表库。ECharts 是一个灵活且强大的图表库，支持各种图表样式，能够高效地展示商品价格变化。

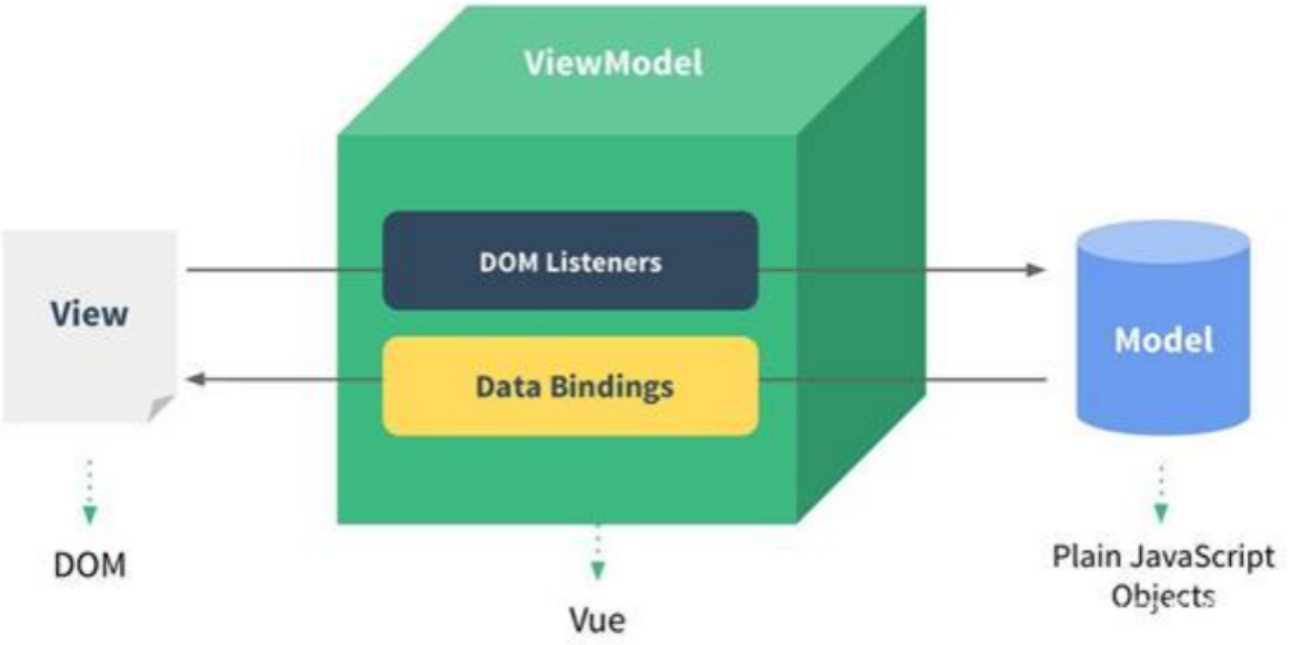
下面是Vue框架的详细讲解

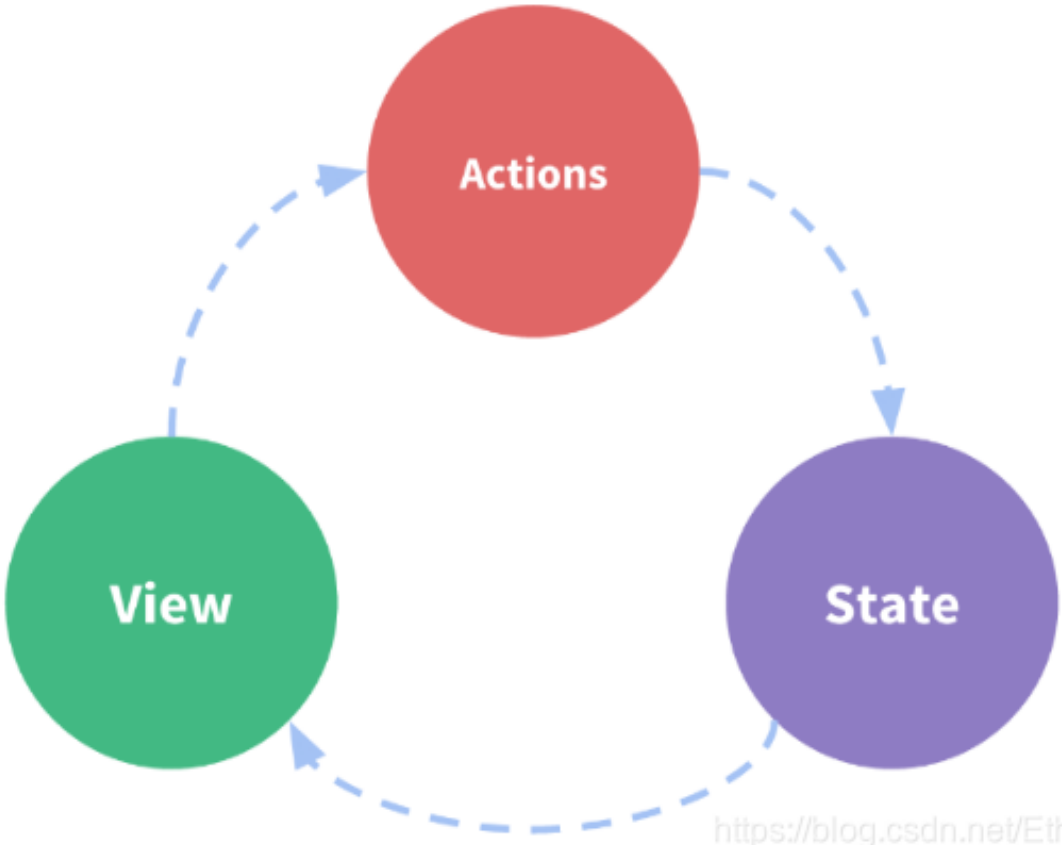

前端
浏览器
HTML\JS\CSS（单页面应用，各模块按需加载）
生成各页面JS模块
首页
登录
会话
.....
webpack
app.js
dev
build
webpack
页面容器
Vue Component
公共组件
页面组件
...
页面组件
Vue Router
Login
Home
...
路由配置
路由地址
Vuex
状态管理
state
view
actions
UI组件库
util
第三方库
工具类
公共组件
配置信息
plugin
Service
Data Fiter
中间件
Axios
Network
POST请求
Get请求
API
后台接口



同时Vue还特别适合组件化和双向绑定







4.3 后端技术栈选择

1. 框架：Django

- Django 是 Python 最流行的全栈 Web 框架之一，具备完备的 ORM 系统、URL 路由管理和表单处理机制，能够快速搭建稳健的后端 API 服务。此外，Django 自带的用户认证系统能够简化用户注册、登录等功能的开发，确保用户信息的安全管理。

2. **数据库：MySQL**

- MySQL 被选为数据库管理系统。它拥有成熟的社区支持和广泛的应用场景，且与 Django 的 ORM 集成度高，能够快速进行数据库操作。商品信息、历史价格数据等将存储于 MySQL 中，便于后续查询和分析。

3. **缓存：Redis**

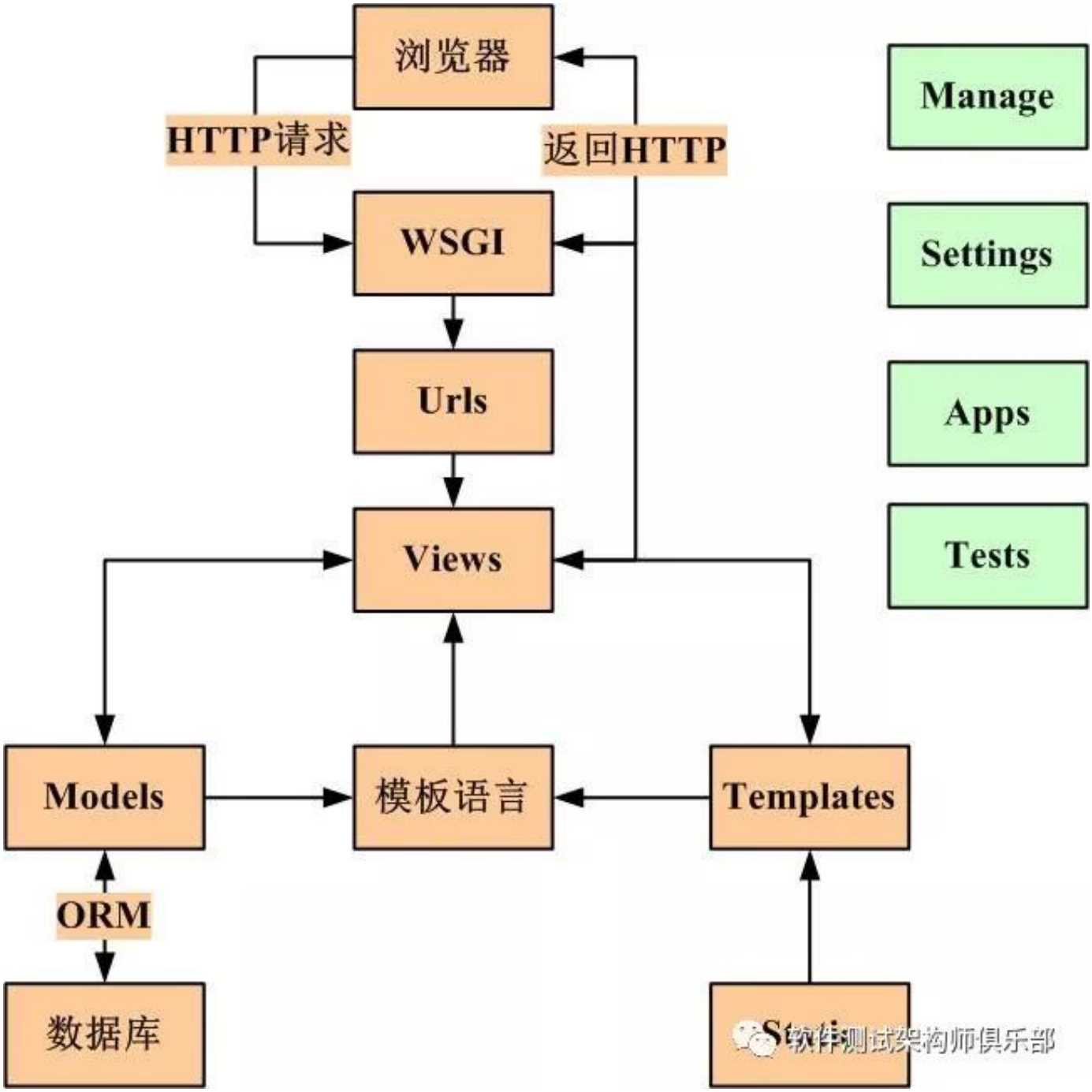
- Redis 用于缓存查询频繁的商品价格数据，减少对数据库的直接访问压力，并提升系统响应速度。特别是针对多次查询同一商品的情况，Redis 能够大幅减少后端的压力，提升用户体验。

4. **密码加密：Django 自带加密功能**

- Django 框架内置了密码加密机制，使用 PBKDF2 等算法对用户密码进行加密存储，确保数据的安全性。

5. **API 服务：第三方电商平台 API**

- 为了获取实时商品价格，系统需要调用多个电商平台（如淘宝、京东等）的 API，获取相关商品的最新价格。部分电商平台可能要求用户登录后才能查询价格，因此需要处理平台的用户身份验证。



4.4 前后端交互及其他相关技术

1. **前后端通信：RESTful API + JWT**

前后端通信采用 RESTful API 设计，通过 HTTP 请求完成数据的交互。使用 JSON Web Token (JWT) 进行用户认证和授权，确保只有登录用户才能进行价格查询等操作。JWT 方便管理前后端的用户会话状态，且适用于 SPA（单页应用）架构。

2. **文件存储：阿里云 OSS 或 AWS S3**

用户上传的商品图片、商品条码等静态资源将存储在对象存储服务中（如阿里云 OSS 或 AWS S3），以便前端能够快速访问这些资源。

3. **安全保障：SSL / TLS**

为了确保前后端通信的安全性，使用 SSL / TLS 协议对数据传输进行加密，避免数据在传输过程中被窃听或篡改。

4. **前端适配：响应式设计**

网站需要适配手机端，因此前端采用响应式设计，确保页面在不同尺寸的设备（如手机、平板和桌面浏览器）上均能良好显示。可使用媒体查询和 flexbox 等 CSS 技术实现样式自适应。

5 数据设计

5.1 数据库设计及建表

5.1.1 用户信息建表

```
CREATE TABLE USER (
  userId INT(11) unsigned NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(64) NOT NULL,
  password VARCHAR(64) NOT NULL,
  email VARCHAR(64) NOT NULL,
  phone VARCHAR(20),
  address VARCHAR(255),
  createTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

用户信息表（`USER`）：存储用户的注册信息，包括用户名、密码、邮箱、电话和地址等。

5.1.2 商品信息表

```
CREATE TABLE PRODUCT (
  productId INT(11) unsigned NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  category VARCHAR(255),
  specifications VARCHAR(255),
  barcode VARCHAR(64),
  image VARCHAR(255),
  description TEXT,
  createTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

商品信息表（`PRODUCT`）：存储商品的基本信息，如商品名称、类别、规格、条码、图片路径等。

5.1.3 商品价格历史表

```
CREATE TABLE PRICE_HISTORY (
  priceId INT(11) unsigned NOT NULL AUTO_INCREMENT PRIMARY KEY,
  productId INT(11) unsigned NOT NULL,
  platform VARCHAR(64) NOT NULL,  -- 电商平台名称，如淘宝、京东等
  price DECIMAL(10, 2) NOT NULL,
  timestamp DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (productId) REFERENCES PRODUCT(productId) ON DELETE CASCADE
);
```

商品价格历史表（`PRICE_HISTORY`）：存储商品在各个平台上的价格记录，包括商品ID、平台名称、价格和记录时间。通过外键与商品表进行关联。

5.1.4 降价提醒表

```
CREATE TABLE PRICE_ALERT (
  alertId INT(11) unsigned NOT NULL AUTO_INCREMENT PRIMARY KEY,
  userId INT(11) unsigned NOT NULL,
  productId INT(11) unsigned NOT NULL,
  targetPrice DECIMAL(10, 2) NOT NULL,   -- 用户设置的目标价格
  email VARCHAR(64) NOT NULL,            -- 用户设置接收提醒的邮箱
  createTime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (userId) REFERENCES USER(userId) ON DELETE CASCADE,
  FOREIGN KEY (productId) REFERENCES PRODUCT(productId) ON DELETE CASCADE
);
```

降价提醒表 (`PRICE_ALERT`): 存储用户设置的降价提醒信息，包括用户ID、商品ID、目标价格和通知邮箱。通过外键与用户表和商品表进行关联。

5.2 用户模块接口

5.2.1 用户登录

- **URL**: `/loginSubmit`
- **请求参数**:
  - `body.username`: 用户名
  - `body.password`: 密码
- **返回值**:
  1. 成功
     - `success: true`: 登录成功
     - `user`: 用户信息（包括用户名、电子邮件等）
  2. 失败
     - `success: false`: 登录失败
     - `message`: 错误消息（如用户名或密码错误）
- 此接口用于用户登录，接受用户名和密码，验证用户身份。如果验证通过，返回成功状态和用户信息；否则，返回失败状态和相应的错误消息。

5.2.2 用户注册

- **URL**: `/registerSubmit`
- **请求参数**:
  - `body.username`: 用户名
  - `body.password`: 密码

- `body.email`：电子邮件
- `body.phone`：电话号码
- `body.gender`：性别
- `body.address`：地址

- **返回值**：

1. 成功
   - `success: true`：注册成功
   - `user`：用户信息
2. 失败
   - `success: false`：注册失败
   - `message`：错误消息（如用户名已存在）

- 此接口用于用户注册，接受必要的用户信息，如用户名、密码、邮箱等。验证信息无误后，成功创建用户账号并返回用户信息，否则返回失败消息。

5.3 商品模块接口

5.3.1 商品查询

- **URL**：`/searchProduct`
- **请求参数**：
  - `query.name`：商品名称
- **返回值**：

1. 成功
   - `success: true`：查询成功
   - `products`：商品信息列表（包括名称、价格、平台等）
2. 失败
   - `success: false`：查询失败
   - `message`：错误消息

- 此接口用于查询商品，通过输入商品名称，系统将返回从多个电商平台获取的商品价格信息列表。

5.3.2 商品添加

- **URL**：`/addProduct`

- **请求参数**:
  - `body.name`: 商品名称
  - `body.category`: 商品分类
  - `body.specifications`: 商品规格
  - `body.barcode`: 商品条码
  - `body.image`: 商品图片
- **返回值**:
  1. 成功
     - `success: true`: 添加成功
     - `product`: 商品信息
  2. 失败
     - `success: false`: 添加失败
     - `message`: 错误消息
- 此接口用于添加新的商品信息到商品库，包括名称、分类、规格等详细信息。

5.4 价格跟踪模块接口

5.4.1 历史价格查询

- **URL**: `/priceHistory`
- **请求参数**:
  - `query.productId`: 商品ID
- **返回值**:
  1. 成功
     - `success: true`: 查询成功
     - `priceHistory`: 历史价格数据（日期、价格等）
  2. 失败
     - `success: false`: 查询失败
     - `message`: 错误消息
- 此接口用于查询指定商品的历史价格信息，返回该商品在多个时间点的价格记录。

5.4.2 降价提醒设置

- **URL**: `/setPriceAlert`
- **请求参数**:
  - `body.productId`: 商品ID
  - `body.targetPrice`: 目标价格
  - `body.email`: 用户的通知邮箱
- **返回值**:
  1. 成功
     - `success: true`: 设置成功
  2. 失败
     - `success: false`: 设置失败
     - `message`: 错误消息
- 此接口用于设置指定商品的降价提醒，当商品价格低于目标价格时，系统将通过电子邮件通知用户。

5.5 平台管理模块接口

5.5.1 平台价格查询

- **URL**: `/platformPriceCheck`
- **请求参数**:
  - `body.productId`: 商品ID
  - `body.platform`: 电商平台名称（如淘宝、京东）
- **返回值**:
  1. 成功
     - `success: true`: 查询成功
     - `price`: 平台上的商品价格
  2. 失败
     - `success: false`: 查询失败
     - `message`: 错误消息
- 此接口用于从指定电商平台查询商品的实时价格，支持多个平台的价格比对。

6 系统出错设计

为了保证系统的稳定性与安全性，系统应具备完善的出错处理机制，能够快速定位并修复各类问题，避免严重故障对用户和业务造成影响。以下是系统常见的错误类型、原因及相应的处理方案。

6.1 出错信息及处理方案

系统输出信息	出错原因	处理方法
数据库无法连接	数据库配置出错、数据库连接数超过上限	修改数据库配置，确保连接配置正确；限制并发访问量，确保连接数在合理范围内。
服务器无法访问	服务器正在维护中或短时间内有大量流量导致服务器瘫痪	联系系统管理员进行紧急维护，检查服务器状态，增加负载均衡机制，提升服务器承载能力。
无法读取磁盘内容	磁盘受损，导致数据无法读取	定期对磁盘和数据库进行周期性备份，确保数据安全。出现问题时尽快恢复备份数据，并检修硬件设备。
非法访问	用户试图访问管理界面或后台程序，可能意图窃取敏感数据	限制普通用户越权访问，采用权限控制，保护后台数据；同时对异常访问进行监控并记录日志，及时报警。
数据库执行出错	用户恶意实施 SQL 注入攻击	通过使用参数化查询构建 SQL 语句，对用户输入进行严格过滤和验证，防止 SQL 注入攻击。
页面无法加载	前端静态资源未能正确加载，或 JavaScript 执行错误	检查前端资源路径，确保所有静态资源正确加载，调试 JavaScript 代码，处理运行时异常。
超时错误	请求处理时间过长，可能是由于服务器响应缓慢或网络延迟	优化服务器性能，减少响应时间；同时可设置合理的请求超时时间，并提供友好的用户提示。
权限不足	用户试图进行超出权限范围的操作，如修改管理员设置	对用户权限进行严格校验，提示用户权限不足，并记录用户操作以便审计。

6.2 补救措施

为确保系统在出现问题时能够迅速恢复和修复，制定以下补救措施：

1. **数据备份与恢复**
   系统应定期对数据库和文件进行备份，采用多级备份方案（如每日备份、每周备份、每月备份）确保数据在出现问题时能够及时恢复。特别是对于关键性数据，必须保障其具备快速恢复能力。发生硬件损坏时，优先从最近的备份数据中进行恢复，并检查系统是否正常运行。

2. **日志监控与告警**
   系统需具备实时日志记录与监控功能，详细记录每一次异常情况及用户操作，以便问题发生时能够快速定位原因。设置告警机制，当系统出现重大错误（如数据库连接失败、非法访问等）时，及时通知管理员并触发自动化应对方案。

3. **异常处理与自动化恢复**
   在出现服务器宕机或网络中断等严重错误时，系统应具备一定的自动化恢复能力。例如，通过设置冗余服务器和负载均衡机制，确保在一台服务器失效的情况下，其他服务器能够立即接管工作，减少服务中断的时间。

4. **SQL 注入防护**
   针对 SQL 注入等攻击风险，系统应在输入数据处理环节严格使用参数绑定的方法构建 SQL 语句，并对用户输入内容进行全面过滤。除此之外，需加强对用户输入数据的安全检查，防止使用特殊字符或构造恶意语句，最大限度避免数据库被攻击。

5. **负载均衡和扩展**
   在流量高峰时，服务器负载可能会迅速升高，导致部分服务不可用。应部署负载均衡器，在流量高峰期将负载分配到不同服务器。同时，可以通过横向扩展（如增加服务器节点）提升系统的可扩展性，确保高流量时系统依然能够正常运作。

通过以上措施，系统能够在出现故障时及时修复问题，减少宕机时间，提高系统的可用性和可靠性，确保用户体验的稳定性。

6.3 系统维护设计

(1) 用户在该系统执行操作时应该留下痕迹，以方便检查系统是否被恶意篡改。同时系统管理员定时查看系统日志，统计非法攻击来源和次数，并针对相应攻击加强安全防范措施。

(2) 系统维护人员及时更新技术漏洞，通过各种手段防止各种对系统的攻击，增强代码的可靠性。

(3) 定期维护数据库，涉及到检查数据库表、检查日志文件等，确保数据库内数据的正确性

7 附录

7.1 项目进度安排

时间段	计划进度
2024.10	完成系统设计与初步框架搭建
2024.11上旬	完成前后端代码逻辑的书写
2024.11 中下旬	系统集成与测试及相关文档编写
2024.12	用户手册等文档编写以及相应组件美化

7.2 备注

本设计文档内容仅供参考，最终效果以实现代码为准