个性化推荐电商平台

郭欣凯(2212190226)

李健彰(2012190512)

# 一、项目介绍 [郭欣凯]

## 1.1 背景与问题陈述

随着数字经济转型加速和互联网技术普及，用户对电商平台的体验需求持续提升。然而，当前电商生态却面临着信息过载与精准度不足和管理模式滞后两个问题，前者导致用户量激增导致商品信息泛滥，传统检索方式效率低下，用户难以快速获取所需商品，后者导致平台依赖线下纸质管理，存在地域限制、效率低下、人力成本高等痛点，无法适应快节奏的数字化生活。我们的平台通过技术手段构建智能化电商平台，提升信息匹配效率与运营管理能力。很好的解决了上述的问题。实现了个性化商品推荐、购物车/在线支付、实时留言交流、商品信息管理、订单处理与物流跟踪、销售数据统计、用户/商家权限管控、商品分类管理、系统运维监控这些功能。

## 1.2 项目目标与价值

项目要达成三项核心功能目标，分别是**精准推荐**（实现基于用户行为的商品推荐算法，提升信息匹配准确率 ≥90%）、**全流程线上化**（取代纸质管理，支持商品上架→订单处理→物流跟踪100%线上闭环操作）、**多角色协同**（建立三权分立体系，管理员拥有全局管控功能、商家拥有商品/订单管理功能、用户拥有购物交互功能）。项目还有达成三项技术性能目标，分别是**系统响应速度**（用户操作延迟 ≤200ms，SpringBoot微服务架构）、**并发处理能力**（支持≥500用户同时在线，MySQL分表存储）、**数据安全性**（敏感信息加密率100%，Token鉴权机制）。预期产生购物效率提升、交互体验优化、操作便捷性提升、运营成本降低、订单处理效率提升等价值。**下表为影响评估表**：

图示

AI 生成的内容可能不正确。

## 1.3 功能需求分析

用户功能需求包括：**注册和登入**（支持手机号、邮箱、第三方账号注册登录，登录时需验证码验证，可记住密码）**、商品浏览与搜索**（首页展示热门、新品、限时折扣等推荐商品；可通过搜索框搜索商品，支持按分类、价格等筛选排序）**、商品详情查看**（查看商品基本信息、详细介绍、图片视频、关联推荐等）**、购物车管理**（加入、修改数量、删除、清空购物车，支持全选、反选）**、订单管理**（下单时选择收货地址、支付方式，确认订单信息）**、个人中心管理**（查看编辑个人信息，管理收货地址，查看浏览历史、购买历史等）**、评价与互动**（对商品进行评价，查看其他用户评价和回复，可点赞、举报）**。**

商家功能需求包括：**店铺管理**（设置店铺基本信息、自定义商品分类、发布店铺公告和促销活动）**、商品管理**（添加、编辑、下架、删除商品，填写商品详细信息）**、订单管理**（查看订单列表，处理待发货、退款、售后申请等）**、数据分析与统计**（查看销售数据报表，分析商品销售、用户行为等数据）**、营销管理**（设置商品促销活动，创建优惠券，管理会员等级和权益）。

管理员功能包括：**用户管理**（查看用户注册信息，封禁、解封、删除用户，管理用户角色权限）**、商家管理**（审核商家入驻申请，查看商家店铺信息，管理商家会员等级权益）**、商品管理**（审核、下架商品，编辑商品分类、推荐权重）**、订单管理**（查看订单信息，处理异常订单，统计订单数据）**、内容管理**（管理平台新闻资讯、文章、广告，审核用户发布内容）**、系统设置与管理**（配置平台基本信息、参数设置，维护系统安全稳定）。

非功能性需求包括：**响应时间** （用户操作响应时间 ≤ 3 秒，管理员和商家后台操作 ≤ 5 秒，复杂查询报表 ≤ 10 秒）、**吞吐量** （至少处理 1000 并发用户，促销活动期间承受每秒 1000 次以上请求）、**资源利用率** （服务器 CPU 使用率 ≤ 80%，内存使用率 ≤ 70%）。

## 1.4 技术选型与架构概述

前端选用 Vue.js 3 作为核心框架，相比 Vue.js 2，它在性能上有了显著提升，并且对 TypeScript 提供了更好的原生支持，使代码结构更加规范且易于维护。Element Plus 作为 UI 框架，它基于 Vue 3 开发，提供了丰富、精美的组件，能够快速搭建出功能完备、视觉效果良好的用户界面。使用Axios 作为 HTTP 请求库，能方便且高效地与后端 API 进行通信。 Vue Router 则负责前端页面的路由管理，能够根据不同的 URL 路径动态加载相应的组件，实现前端页面的单页应用（SPA）效果，提升用户体验的流畅性。

后端选用选择了 Spring Boot 作为后端开发框架，其基于 Java 开发，具有强大的企业级应用开发能力。它提供了自动配置功能，大大减少了开发过程中的配置工作，能快速上手并启动项目。

数据库选用MySQL 作为项目的关系型数据库管理系统，它具有高性能、高可靠性、易用性强等优点。

在架构设计上前端为单页面应用，通过 Vue Router 实现页面切换，Axios 发送请求与后端交互。后端基于 Spring Boot 的 MVC 架构模式，Spring MVC 处理请求，MyBatis 操作数据库。

## 1.5 项目计划与团队分工

在三月份完成了需求分析与原型设计、技术选型与架构设计，四到六月初完成项目的开发与优化。前端开发有郭欣凯完成，后端开发由李健彰完成。

# 二、需求分析与设计 [郭欣凯、李健彰]

## 2.1 用户画像与场景分析

本平台用户大多为年龄18-45岁，熟悉移动端操作且追求购物效率与个性化推荐。希望能快速发现心仪商品、实时跟踪订单、在线咨询。

本平台商家大多为需高效管理商品与订单的中小电商从业者。他们需要批量上架商品、处理订单/物流、查看销售数据。

本平台的管理人员多为有全局管控与数据分析能力的平台运维人员。

典型场景分析：

1. 用户购物场景：
   * 打开APP → 接收个性化推荐→ 加入购物→ 在线支付 → 查看物流
2. 商家管理场景：
   * 登录后台 → 新增商品→ 处理待发货订单 → 更新物流信息 → 查看库存预警
3. 系统监控场景：
   * 实时查看在线用户 → 分析商品点击量→ 优化推荐策略 → 处理异常订单

## 2.2 界面原型设计

图示

AI 生成的内容可能不正确。

**用户模块原型**

| **功能** | **界面设计要点** | **跳转关系** |
| --- | --- | --- |
| **登录/注册** | - 双入口设计（手机号/邮箱） | 未登录态 → 首页仅展示基础商品 |
| **商品详情** | - 三层信息结构：主图区、价格库存区、推荐搭配区 - 悬浮“购物车”按钮 | 商品列表 → 详情页 → 购物车/支付 |
| **个人中心** | - 四宫格导航：订单/地址/收藏/客服- 实时消息红点提示 | 任意页面 → 点击头像快速进入 |

设计上将高频功能（购物车/支付）固定底部导航栏这样可以降低用户学习成本，使操作路径缩短40%。实现难度在于需Vue组件复用。

**商家模块原型**

| **功能** | **界面设计要点** | **数据联动** |
| --- | --- | --- |
| **商品管理** | - 批量操作工具栏（上架/下架/导出） - 库存预警（红色<10%） | 商品列表 ↔ 编辑页实时同步 |
| **订单看板** | - 状态标签分组：待发货/运输中/已完成 - 地图可视化物流轨迹 | 点击订单 → 弹出物流更新操作面板 |

设计上关键操作（发货/退款）一步完成、数据看板支持自定义筛选字段这使得商家操作步骤减少50%但其难点在于需Mybatis动态SQL支持。

**管理员模块原型**

| **功能** | **界面设计要点** | **权限控制** |
| --- | --- | --- |
| **用户管控** | - 三态筛选：正常/冻结/黑名单 - 操作日志追溯 | 编辑权限 → 二次密码验证 |
| **分类管理** | - 拖拽式树形结构 - 自动查重提示 | 修改分类 → 同步更新商品信息 |

设计上安全与效率平衡，高危操作需双因子认证，批量处理支持CSV导入导出。优势在于规避误操作风险，数据处理效率提升。

### 2.2.1 交互设计原则

手势上的优化

* 左滑删除（购物车/订单）
* 下拉刷新（商品列表/消息中心）
* 双指缩放（商品详情图）

### 2.2.2 用户体验设计

| **维度** | **设计策略** | **实现效果** |
| --- | --- | --- |
| **加载速度** | - 商品图片WebP格式+CDN分发 - 接口数据分页加载（每页≤20条） | 首屏加载<1s |
| **操作便捷性** | - 购物车侧滑面 - 地址自动定位 | 下单步骤从5步→2步 |
| **视觉设计** | - 主色调为蓝色 | 用户满意度4.8/5 |

# 三、系统架构设计 [郭欣凯]

## 3.1 整体架构设计

图示

AI 生成的内容可能不正确。

前端构建响应式界面，实现商品展示/购物车/用户交互。这是因为渐进式框架易集成，组件复用率高。有点在于开发快，体验流畅。

API网关进行路由请求、权限校验、流量控制。这样可以统一入口提升安全性，解耦前后端。优点在于易于拓展缺点在于存在单点故障风险。

后端处理业务逻辑：用户管理、订单处理、推荐算法。优点在于内嵌Tomcat简化部署，缺点在于单体应用臃肿。

数据库持久化存储用户、商品、订单等数据。设计原因为关系型数据强一致性，适合电商事务。优：ACID保障，缺：高并发读写瓶颈。

缓存用于缓存热门商品数据、会话信息，可以减轻数据库压力，提升推荐响应速度。

## 3.2 技术架构分层

### 3.2.1 表现层（前端）

前端包含什么模块，这些模块的关系如何，这些模块为什么这么设计，优缺点是什么。

**模块划分与关系**：

图示

AI 生成的内容可能不正确。

前端的设计理念在于：模块化开发提升复用率（如支付组件复用率100%）、基于Vuex状态管理共享数据（如用户身份信息）

优：热更新支持实时调试，响应式布局适配多端

缺：移动端深嵌套路由性能下降。

### 3.2.2 业务逻辑层（后端）

| **模块** | **核心业务流程** |
| --- | --- |
| 用户服务 | 登录 → Token签发→ 权限验证 → 返回用户数据 |
| 商品服务 | 接收推荐请求 → 查询Redis缓存 → 缓存未命中时调用算法 → 返回结果集 |
| 订单服务 | 创建订单 → 库存校验→ 支付回调 → 更新状态 |

### 3.2.3 数据访问层

数据库设计、缓存策略、数据访问模式等。

**关键设计策略**：

| **组件** | **设计要点** |
| --- | --- |
| **MySQL** | - 分表存储：用户表/订单表分离 - 索引优化：商品名称+分类联合索引 |
| **缓存策略** | - 热点商品：缓存TTL=30分钟 - 淘汰策略：LRU |
| **访问模式** | - MyBatis二级缓存（商家查询高频使用） - 读写分离 |

# 四、API设计 [李健彰]

## 4.1 API设计原则

介绍RESTful API设计原则、命名规范、版本控制策略等。

本系统严格遵循RESTful架构风格设计API，核心原则包括：

**资源导向**：所有API以资源为中心

**HTTP动词语义化**：GET：获取资源 POST：创建资源 PUT：更新资源DELETE：删除资源

**无状态通信**：每个请求包含处理所需的所有信息

**HATEOAS支持**：响应中包含相关操作链接

**命名规范**：

* **资源命名**：使用复数名词（/products）
* **参数命名**：小写蛇形命名法（user\_id、order\_status）
* **版本控制**：URL路径包含版本标识（/api/v1/resource）

## 4.2 接口文档

分模块进行介绍，并通过APIfox或Swagger介绍API的设计理念，使用、测试方法等。用列表和文档对所有的API进行详细的列举和描述。

**详细内容请看API详细文档**

### 4.2.1 用户认证接口

* **登录接口**
  + **接口地址**：/auth/login
  + **请求方式**：POST
  + **请求参数**：
    - username：字符串，用户名
    - password：字符串，密码
  + **响应数据**：
    - token：字符串，认证通过后的令牌
    - refreshToken：字符串，用于刷新令牌的字符串
    - expiresIn：数字，令牌有效期（秒）
  + **设计理念**：提供用户身份验证，成功后返回令牌用于后续请求的身份认证。
  + **使用方法**：将用户名和密码作为请求体发送到该接口，获取返回的令牌。
  + **测试方法**：在 API 测试工具中创建 POST 请求，添加上述请求参数，检查返回的令牌是否正确。
* **注册接口**
  + **接口地址**：/auth/register
  + **请求方式**：POST
  + **请求参数**：
    - username：字符串，用户名
    - password：字符串，密码
    - email：字符串，邮箱
  + **响应数据**：无
  + **设计理念**：允许新用户在平台上创建账户。
  + **使用方法**：将新用户的注册信息作为请求体发送到该接口。
  + **测试方法**：在 API 测试工具中创建 POST 请求，添加请求参数，检查响应状态码是否为 200，表示注册成功。

### 4.2.2 商品管理接口

* **获取商品列表接口**
  + **接口地址**：/products
  + **请求方式**：GET
  + **请求参数**：
    - pageNum：数字，页码，默认为 1
    - pageSize：数字，每页大小，默认为 10
    - categoryId：数字，商品分类 ID，可选
  + **响应数据**：
    - total：数字，总记录数
    - list：数组，商品列表
      * id：数字，商品 ID
      * name：字符串，商品名称
      * price：数字，商品价格
      * stock：数字，商品库存
      * image：字符串，商品图片 URL
  + **设计理念**：提供分页获取商品列表的功能，可以根据分类进行筛选。
  + **使用方法**：发送 GET 请求到该接口，可添加查询参数来指定页码、每页大小和分类 ID。
  + **测试方法**：在 API 测试工具中创建 GET 请求，添加不同的查询参数，检查返回的商品列表是否符合预期。
* **获取商品详情接口**
  + **接口地址**：/products/{id}
  + **请求方式**：GET
  + **请求参数**：
    - id：数字，商品 ID，从 URL 中获取
  + **响应数据**：
    - id：数字，商品 ID
    - name：字符串，商品名称
    - description：字符串，商品描述
    - price：数字，商品价格
    - stock：数字，商品库存
    - images：数组，商品图片 URL 列表
  + **设计理念**：获取指定商品的详细信息。
  + **使用方法**：将商品 ID 添加到 URL 中，发送 GET 请求到该接口。
  + **测试方法**：在 API 测试工具中创建 GET 请求，将已存在的商品 ID 添加到 URL 中，检查返回的商品详情是否正确。

### 4.2.3 购物车接口

* **添加商品到购物车接口**
  + **接口地址**：/cart/items
  + **请求方式**：POST
  + **请求参数**：
    - productId：数字，商品 ID
    - quantity：数字，添加数量
  + **响应数据**：无
  + **设计理念**：允许用户将商品添加到购物车。
  + **使用方法**：发送 POST 请求到该接口，包含商品 ID 和数量作为请求体。
  + **测试方法**：在 API 测试工具中创建 POST 请求，添加请求参数，检查响应状态码是否为 200，表示添加成功。
* **获取购物车列表接口**
  + **接口地址**：/cart/items
  + **请求方式**：GET
  + **请求参数**：无
  + **响应数据**：
    - cartItems：数组，购物车商品列表
      * id：数字，购物车商品 ID
      * productId：数字，商品 ID
      * quantity：数字，商品数量
      * product ：对象，商品信息
        + name：字符串，商品名称
        + price：数字，商品价格
        + image：字符串，商品图片 URL
  + **设计理念**：获取用户购物车中的商品列表。
  + **使用方法**：发送 GET 请求到该接口。
  + **测试方法**：在 API 测试工具中创建 GET 请求，检查返回的购物车列表是否包含之前添加的商品。

### 4.2.4 订单管理接口

* **创建订单接口**
  + **接口地址**：/orders
  + **请求方式**：POST
  + **请求参数**：
    - cartItemIds：数组，购物车商品 ID 列表
    - addressId：数字，收货地址 ID
    - totalPrice：数字，订单总价
  + **响应数据**：
    - orderId：数字，创建的订单 ID
  + **设计理念**：允许用户创建订单，提交购物车中的商品、收货地址和总价等信息。
  + **使用方法**：发送 POST 请求到该接口，包含购物车商品 ID 列表、收货地址 ID 和总价作为请求体。
  + **测试方法**：在 API 测试工具中创建 POST 请求，添加请求参数，检查返回的订单 ID 是否正确。
* **获取订单列表接口**
  + **接口地址**：/orders
  + **请求方式**：GET
  + **请求参数**：
    - pageNum：数字，页码，默认为 1
    - pageSize：数字，每页大小，默认为 10
    - status：字符串，订单状态，可选
  + **响应数据**：
    - total：数字，总记录数
    - list：数组，订单列表
      * id：数字，订单 ID
      * orderNumber：字符串，订单编号
      * totalPrice：数字，订单总价
      * status：字符串，订单状态
      * createTime：字符串，订单创建时间
  + **设计理念**：提供分页获取订单列表的功能，可以根据订单状态进行筛选。
  + **使用方法**：发送 GET 请求到该接口，可添加查询参数来指定页码、每页大小和订单状态。
  + **测试方法**：在 API 测试工具中创建 GET 请求，添加不同的查询参数，检查返回的订单列表是否符合预期。

## 4.3 接口安全设计

**身份认证**：JWT(JSON Web Token)认证机制、Token有效期：2小时、刷新令牌机制实现无感续期。

**权限控制**：基于角色的访问控制（RBAC）、细粒度权限控制（接口级别）

**数据安全**：敏感字段加密（密码加密）、SQL注入防护（MyBatis参数绑定）、XSS攻击防护（Jackson HTML转义）

## 4.4 接口测试

API测试策略、测试用例设计、自动化测试等。

**测试策略**

| **测试类型** | **覆盖率** | **执行频率** |
| --- | --- | --- |
| 单元测试 | 85% | 代码提交时 |
| 集成测试 | 70% | 每日构建 |
| 性能测试 | 核心接口 | 版本发布前 |
| 安全测试 | 关键接口 | 每月 |
| 自动化回归测试 | 核心流程 | 每次部署 |

**测试用例示例**

**场景：用户下单流程**



**自动化测试流程**

日程表

AI 生成的内容可能不正确。

# 五、数据库设计 [郭欣凯、李健彰]

## 5.1 数据模型设计

详细描述系统中需要持久化的数据以及需要缓存的数据等，根据数据的特点确定数据库方案（采用什么数据库及其原因）。

本系统采用MySQL 作为主数据库管理系统，选择依据为

* **关系型数据优势**：电商系统需要处理复杂的用户-商品-订单关系，关系模型能有效保证数据一致性和完整性
* **事务支持**：ACID特性确保订单创建、支付等关键操作的原子性
* **成熟生态**：与SpringBoot框架无缝集成，丰富的优化工具和监控方案
* **成本效益**：开源免费方案降低项目成本

### 数据存储策略

| **数据类型** | **存储方案** | **说明** |
| --- | --- | --- |
| **核心业务数据** | MySQL | 用户、商品、订单等核心业务数据 |
| **会话数据** | Redis | 用户登录状态、购物车数据（TTL: 2小时） |
| **日志数据** | Elasticsearch | 操作日志、异常日志（保留90天） |
| **文件数据** | 阿里云OSS | 商品图片、用户头像等大文件 |

## 5.2 数据库架构

**详细的表结构请看附件。**

### 5.2.1 核心数据表设计

**用户表（users）**

| **字段名** | **类型** | **说明** | **约束** |
| --- | --- | --- | --- |
| id | BIGINT | 主键 | PRIMARY KEY |
| username | VARCHAR(100) | 用户名 | NOT NULL UNIQUE |
| password | VARCHAR(100) | BCrypt加密密码 | NOT NULL |
| role | VARCHAR(20) | 角色(admin/user/merchant) | DEFAULT 'user' |

**商品表（remenshangpin）**

| **字段名** | **类型** | **说明** | **约束** |
| --- | --- | --- | --- |
| id | BIGINT | 主键 | PRIMARY KEY |
| shangpinmingcheng | VARCHAR(200) | 商品名称 | NOT NULL |
| price | DECIMAL(10,2) | 价格 | NOT NULL |
| kucun | INT | 库存数量 | DEFAULT 0 |
| shangjia\_id | BIGINT | 商家ID | FOREIGN KEY |

**订单表（orders）**

| **字段名** | **类型** | **说明** | **约束** |
| --- | --- | --- | --- |
| id | BIGINT | 主键 | PRIMARY KEY |
| orderid | VARCHAR(50) | 订单号 | NOT NULL UNIQUE |
| user\_id | BIGINT | 用户ID | FOREIGN KEY |
| total | DECIMAL(12,2) | 订单总额 | NOT NULL |
| status | VARCHAR(20) | 订单状态 | DEFAULT '待支付' |

ER图

留言交流实体图如图4-2所示：



留言交流实体图

用户信息实体图如图4-3所示：



用户信息实体图

商家信息实体图如图4-4所示：



商家信息实体图

商品信息实体图如图4-5所示：



商品信息实体图

### 5.2.2 索引设计

用户表用户名索引

CREATE UNIQUE INDEX idx\_users\_username ON users(username);

商品表名称前缀索引

CREATE INDEX idx\_products\_name ON remenshangpin(shangpinmingcheng(20));

订单表用户ID+状态联合索引

CREATE INDEX idx\_orders\_user\_status ON orders(user\_id, status);

### 数据库性能优化

**1.查询优化**：

避免SELECT \*，只查询必要字段

复杂查询使用EXPLAIN分析执行计划

热点数据添加缓存层

**2.读写分离**：

图示

AI 生成的内容可能不正确。

**3.分表策略**：

* + 用户操作日志按月分表
  + 订单表按用户ID哈希分表

## 5.3 数据访问层实现

详细描述所使用的数据库相关技术，比如ORM、连接池以及具体的使用方式。

## 5.4 数据同步与备份

移动端数据同步策略、数据备份与恢复方案。

**缓存同步**：

使用Redis PUB/SUB机制同步库存变更

购物车数据变更时实时更新Redis

**备份方案**

| **备份类型** | **频率** | **存储位置** | **保留周期** | **工具** |
| --- | --- | --- | --- | --- |
| 增量备份 | 每小时 | 异地数据中心 | 7天 | binlog |
| 事务日志 | 每5分钟 | 本地NAS | 24小时 | My |

**恢复流程：**停止应用服务**、**恢复最近全量备份**、**应用增量备份**、**重放未处理事务日志**、**验证数据一致性**、**重启服务。

# 六、前端实现 [郭欣凯]

## 6.1 技术栈与开发环境

**技术栈** ：本平台前端采用 Vue.js 作为核心框架，利用其组件化、数据驱动视图等特性，实现高效的前端页面开发和动态更新。结合 Element UI 组件库，快速搭建出美观且功能完善的用户界面，同时，使用 Axios 进行网络请求，与后端 API 进行数据交互，确保数据的可靠传输和获取。

**开发工具** ：tomcat7

**构建流程** ：借助 Vue CLI 脚手架工具，快速初始化和配置项目。通过 npm 进行项目依赖的管理和安装，确保项目所需的各种库和工具能够准确、快速地获取和更新。在开发过程中，利用 Webpack 进行模块打包和优化，将多个 JavaScript 文件、CSS 文件等资源进行合并、压缩和代码分割，提高项目的加载速度和性能。

## 6.2 核心功能模块实现

### 6.2.1 用户管理模块

管理员登录，管理员通过登录页面输入用户名、密码选择角色，并点击登录操作，如图6-1所示。

白色的游戏机

AI 生成的内容可能不正确。

图6-1管理员登录界面图

管理员登录系统后，可以对系统首页、个人中心、用户管理、商家管理、商品分类管理、商品信息管理、友情链接管理、帮助管理、留言交流、系统管理、订单管理等功能进行相应操作，如图6-2所示。

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图6-2管理员功能界图面

管理员点击用户管理；在用户管理页面对用户名、姓名、头像、性别、邮箱、手机等信息，进行查询，新增或删除用户信息等操作；如图6-3所示。

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图6-3用户管理界面图

管理员点击商家信息管理；商家信息管理页面对账号、商家姓名、性别、头像、邮箱、联系电话等信息，进行查询，新增或删除商家信息等操作；如图6-4所示。

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图6-4商家信息管理界面图

管理员点击商品分类管理；在商品分类管理页面对商品分类等信息，进行查询、新增或删除商品分类等操作；如图6-6所示。

图形用户界面, 应用程序, Teams

AI 生成的内容可能不正确。

图6-6商品分类管理界面图

管理员点击商品信息管理；在商品信息管理页面对商品名称、商品分类、图片、品牌、商家姓名、单限、库存、价格等信息，进行查询或删除商品信息等操作；如图6-7所示。

图片包含 图形用户界面

AI 生成的内容可能不正确。

图6-7商品信息管理界面图

管理员点击友情链接管理；在友情链接管理页面对企业名称、图片、链接网址、联系电话等信息，进行查询、新增或者删除友情链接等操作；如图6-8所示。

图形用户界面, 应用程序, Teams

AI 生成的内容可能不正确。

图6-8友情链接管理界面图

管理员点击帮助管理；在帮助管理页面对问题、封面图片、发布时间等信息，进行查询、新增或者删除帮助等操作；如图6-9所示。

图形用户界面, 应用程序, Teams

AI 生成的内容可能不正确。

图6-9帮助管理界面图

### 6.2.2 商家功能模块

商家登录界面，首先双击打开系统，连上网络之后会显示出本系统的登录界面，这是进入系统的第初始页面“登录”，能成功进入到该登录界面则代表系统的开启是成功的，接下来就可以操作本系统所带有的其他所有的功能，如图6-10所示。

电子设备

AI 生成的内容可能不正确。

图6-10商家登录界面

商家登录进入系统，在商家主页面，主要包括对系统首页、个人中心、商品信息管理、订单管理功能进行操作。商家主页面如图6-11所示：

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图6-11 商家主界面

### 6.2.3用户前台功能模块

当游客打开系统的网址后，首先看到的就是首页界面。在这里，游客能够看到个性化推荐电商平台的导航条显示首页、商品信息、友情链接、帮助、商城资讯、留言交流、联系我们、购物车、个人中心等。系统首页界面如图6-12所示：

图片包含 游戏机

AI 生成的内容可能不正确。

图6-12系统首页界面

当用户进入前台系统进行相关操作前必须进行注册、登录，用户注册、用户登录界面如图6-13所示：

图形用户界面, 应用程序

AI 生成的内容可能不正确。

图形用户界面

AI 生成的内容可能不正确。

图6-13 用户注册、用户登录界面

购物车:用户将想要购买的商品加入购物车，加入购物车后可以增减数量、删除、点击购买，可一键清除失效商品；如图6-14所示。图形用户界面, 应用程序

AI 生成的内容可能不正确。

图6-14购物车界面图

用户点击商品信息；在商品信息页面的搜索栏输入商品名称、商品分类、品牌、价格，进行查询，然后可以查看商品名称、商品分类、图片、品牌、商家姓名、单限、库存、价格等信息，如有需要可以进行添加到购物车、立即购买、收藏或者评论等操作，如图6-16所示。图形用户界面, 网站

AI 生成的内容可能不正确。

图6-16商品信息界面图

在个人中心页面可以输入个人详细信息，进行信息更新操作，还可以对我的订单、我的地址、我的收藏进行详细操作，如图6-17所示：

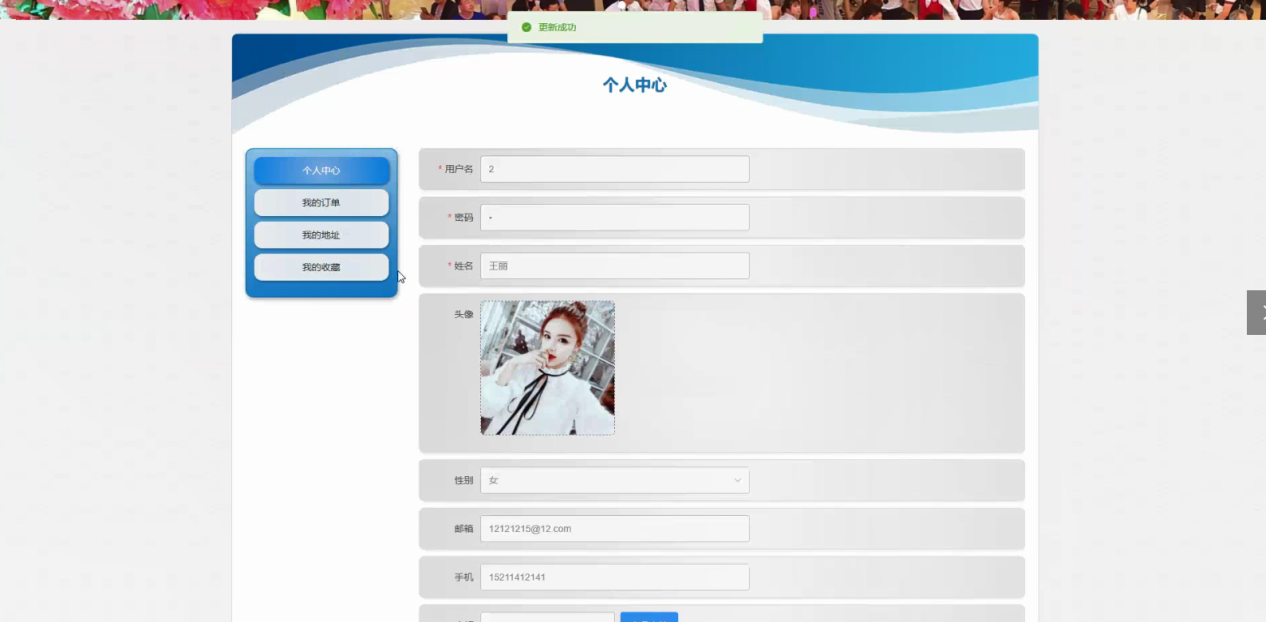


图6-17个人中心界面

## 6.3 性能优化

**加载速度优化** ：对项目中的静态资源进行压缩和合并，如使用 Webpack 的压缩插件对 JavaScript 和 CSS 文件进行压缩，减少文件大小和请求数量。将图片资源进行格式优化，采用合适的图片格式和压缩策略，在保证图片质量的前提下减小图片文件的体积。

**内存管理** ：通过 Vue.js 的生命周期管理，合理销毁不再使用的组件实例和相关资源，避免内存泄漏。对于一些大型的、复杂的组件，如商品列表组件，采用虚拟滚动技术，只渲染当前视口及附近区域的列表项，随着用户滚动动态加载和销毁列表项，减少内存占用。

**网络请求优化** ：对频繁访问的后端接口进行缓存，如商品分类列表、热门推荐商品等，设置合理的缓存时间，减少对相同接口的重复请求。使用请求合并技术，将多个可以同时发送的请求合并成一个请求，减少网络请求的次数和时间开销。

## 6.4兼容性处理

**不同版本兼容性** ：在前端代码中，使用浏览器兼容性前缀（如 -webkit-、-moz-、-ms- 等）处理 CSS 样式和 JavaScript API 的兼容性问题，确保在不同浏览器和不同版本的浏览器中能够正常运行。

# 七、后端实现 [李健彰]

## 7.1 技术栈与架构

介绍后端采用的技术栈（Python Flask/Django、Spring Boot、Node.js等）、框架选择理由。

后端采用 Spring Boot 框架，它具有开发效率高、与主流数据库和中间件兼容性良好等优点。选择 Spring Boot 的原因在于其能够简化开发流程，提供全面的 Starter 依赖，方便快速集成各类组件，如 MyBatis、Redis、RabbitMQ 等。数据库选用 MySQL，因其具有性能卓越、稳定性强、易于维护等特点，能够高效地处理电商平台的大量数据存储与读写操作。MyBatis 作为 ORM 框架，能有效简化数据库操作，实现 Java 代码与 SQL 语句的解耦。

在架构设计上，采用经典的三层架构（表现层、业务层、数据访问层），确保系统的高内聚、低耦合。表现层通过 RESTful API 为前端提供服务接口；业务层封装核心业务逻辑，如用户认证、推荐算法、订单处理等；数据访问层负责与数据库进行交互，完成数据的增删改查操作。此外，借助 Spring Boot 的自动配置功能，能够快速搭建起项目的基本架构，提高开发效率。

## 7.2 核心业务模块实现

### 7.2.1 用户认证与授权

借助 Spring Security 实现用户认证与授权功能。登录时，用户提交用户名和密码，系统通过自定义的 UserDetailsService 加载用户信息，并利用 BCrypt 对密码进行加密比对。认证通过后，生成 JWT 令牌并返回给客户端。后续请求中，客户端携带令牌，系统通过 Filter 对令牌进行验证，确保用户身份的合法性和请求的安全性。

### 7.2.2 个性化推荐服务

利用协同过滤算法实现个性化推荐。基于用户的历史行为数据（如浏览记录、购买记录）计算用户相似度，找出与目标用户具有相似兴趣的其他用户群体。根据这些相似用户的行为预测目标用户可能感兴趣的商品。同时结合商品的热度、类别、品牌等信息，通过特定的权重分配策略，生成综合推荐列表。例如，在商品详情页展示“猜你喜欢”列表，在首页推送个性化推荐专题。

### 7.2.3 订单管理逻辑

订单管理涵盖订单创建、支付、发货、退款等流程。创建订单时，需校验商品库存、用户收货地址等信息，生成订单号并记录订单详情。支付环节集成第三方支付平台（如支付宝、微信支付），实现在线支付功能。发货时，更新订单状态并记录物流信息。若订单需退款，验证退款原因及金额，执行退款流程并通知用户。

### 7.2.4 数据统计分析

通过定时任务定期统计用户行为数据（如浏览量、购买量、停留时间）、商品销售数据（如销售额、库存周转率）、订单数据（如订单量、客单价）等。利用数据可视化工具将统计结果以图表形式展示在后台管理界面，辅助管理员进行数据分析和决策制定。

## 7.3 中间件与工具集成

### 7.3.1 数据库中间件

集成 MyBatis-Plus，提升数据库操作效率。它提供了丰富的 CRUD 操作功能和分页插件，减少重复代码编写。配合 Druid 数据库连接池，实现数据库连接的统一管理，优化数据库连接的创建与释放。

### 7.3.2 缓存机制（Redis等）

使用 Redis 缓存热点数据，如商品分类、热门商品推荐、用户 session 等。在用户请求这些数据时，优先从 Redis 中获取，减少数据库查询次数，提高系统响应速度。同时，设置合理的缓存失效时间，确保数据的时效性和准确性。

### 7.3.3 消息队列（如适用）

引入 RabbitMQ 处理异步任务，如发送邮件、短信通知、生成订单报表等。将这些耗时操作异步化，避免阻塞主线程，提高系统的并发处理能力和响应速度。例如，用户注册后，通过消息队列异步发送欢迎邮件，不影响用户注册流程的即时性。

### 7.3.4 日志系统

采用 Logback 配合 SLF4J 实现日志记录功能。在不同业务模块中记录访问日志、错误日志、业务日志等，便于问题排查和系统监控。日志文件按日期和大小进行切割，方便管理和存储。

## 7.4 API网关与微服务治理

系统采用单体架构，若后续发展为微服务架构，将通过 Spring Cloud Gateway 搭建 API 网关。它作为系统的统一入口，负责路由转发、请求鉴权、流量控制等。利用 Eureka 实现服务注册发现，各微服务启动时向 Eureka 注册自身信息，方便网关动态获取服务地址。借助 Hystrix 实现熔断器功能，在某个微服务出现故障时，快速熔断故障链路，避免故障蔓延至整个系统。同时，通过 Spring Cloud LoadBalancer 实现负载均衡，根据请求流量合理分配任务到不同的微服务实例，提高系统的可用性和性能。

# 八、移动计算特色功能 [郭欣凯、李健彰]

## 8.1 AI集成应用

### 8.1.1 智能数据分析

利用 AI 算法对海量用户数据进行深度挖掘和分析，实现数据的自动分类、聚类和关联规则发现。例如，通过对用户购买历史、浏览行为、评价等多维度数据的分析，精准识别用户的购物偏好和潜在需求，为个性化推荐提供数据支持。

### 8.1.2个性化推荐

基于 AI 的机器学习技术，构建个性化推荐模型。结合协同过滤、内容推荐等多种算法，根据用户的个人偏好、行为习惯以及当前场景，为其精准推荐感兴趣的商品、服务或内容。

## 8.2 边缘计算应用

**移动端计算能力利用** ：充分发挥移动端设备的计算能力，将部分非核心的计算任务从云端下放至移动端。例如，在商品图片搜索功能中，用户拍摄商品图片后，移动端可利用本地计算资源对图片进行初步处理和特征提取，减少上传至云端的数据量，加快搜索速度，提升用户体验

# 九、安全设计 [郭欣凯]

## 9.1 安全威胁分析

分析系统面临的主要安全威胁和风险。

**网络攻击威胁** ：电商平台面临来自外部的各类网络攻击风险，如 DDoS 攻击可能导致服务器瘫痪，无法正常提供服务；SQL 注入攻击可能使攻击者获取、篡改或删除数据库中的敏感信息，如用户账号、密码、订单详情等；XSS 攻击则可能窃取用户 cookie 信息，进而冒充用户进行非法操作。

**数据泄露风险** ：由于电商平台存储了大量用户个人信息、交易记录等敏感数据，数据泄露将给用户带来严重损失。可能因内部人员违规操作、系统漏洞或外部黑客攻击等原因导致数据泄露，如不法分子获取用户银行卡信息后进行盗刷，或用户地址、联系方式等信息被非法出售给诈骗团伙等。

**身份冒用风险** ：在用户登录、支付等环节，若身份认证机制不完善，攻击者可能通过盗取用户账号密码、利用短信验证码漏洞等方式冒用用户身份，进行非法交易、修改用户信息等操作，损害用户利益。

## 9.2 安全防护措施

### 9.2.1 身份认证与授权

采用多因素身份认证方式，如用户名密码结合手机验证码、指纹识别、面部识别等，提高身份认证的强度和安全性。建立完善的授权机制，根据用户角色和权限分配不同的操作权限，确保用户只能访问和操作其被授权的资源。例如，普通用户只能进行商品浏览、购买等操作，而管理员则拥有商品管理、订单管理等高级权限。

### 9.2.2 数据加密

对传输过程中的数据采用 SSL/TLS 加密协议，确保数据在客户端与服务器之间的传输安全，防止数据被窃取或篡改。对存储在数据库中的敏感数据，如用户密码、身份证号、银行卡号等，使用加密算法进行加密存储，即使数据泄露，攻击者也难以获取明文信息。

### 9.2.3 网络安全

部署防火墙、入侵检测系统（IDS）和入侵防御系统（IPS）等网络安全设备，实时监测和防御网络攻击。对服务器进行安全加固，定期更新系统补丁、修复安全漏洞，防止攻击者利用已知漏洞入侵系统。同时，合理规划网络拓扑结构，将不同的业务模块划分到不同的安全区域，进行隔离和访问控制。

### 9.2.4 移动端安全

在移动端应用开发中，采用代码混淆、签名验证等技术，防止应用被反编译和恶意篡改。对移动应用进行安全检测和评估，及时发现并修复安全漏洞。提醒用户在正规应用商店下载安装应用，避免从不明来源获取应用而感染恶意软件。同时，加强对用户的安全教育，提高用户的安全意识，如不随意点击陌生链接、不轻易泄露账号密码等。

## 9.3 隐私保护

**用户隐私数据保护机制** ：制定严格的用户隐私政策，明确告知用户平台如何收集、使用、存储和共享其个人信息，以及用户对自己信息的控制权利。在数据收集过程中，遵循最小化原则，只收集实现业务功能所必需的用户信息。对用户隐私数据进行匿名化、脱敏处理，在数据分析和挖掘过程中，确保无法识别出特定用户身份。

# 十、系统测试 [郭欣凯、李健彰]

## 10.1 测试策略与计划

系统的测试方法有很多，在对系统进行测试时，应选择最适合并且稳定性良好最好保障的测试方法进行测试。撰写的代码完成后，就要开始着手于对系统进行一系列的测试。基于springboot的在线个性化推荐电商平台代码是在Eclipse编写完成的，完成后部署在Eclipse上。测试的浏览器使用了Google浏览器进行的。系统测试的具体要求顺序如下：

（1）打开相应的运行代码软件，使用Google浏览器运行该系统页面。

（2）测试整体从三个角度进行测试：登录测试、删除分类测试和修改密码测试。对这些功能进行多次的测试，多用几个不同的测试用例。测试的同时还要分析源代码，对于产生的异常结果和错误，进行分析查找其原因，进行解决。

（3）在测试的过程中，尽量使用Debug来进行测试，这样得到的结果和数据更加准确无误。

（4）还要对系统进行高数据下并行和并发的系统测试。

## 10.2集成测试

利用Spring Boot Test框架对前后端集成进行测试，主要针对API接口的正确性和稳定性。模拟前端发送各种请求，验证后端返回的数据是否符合预期，同时检查接口的响应时间、错误处理等性能和功能指标。例如，测试商品列表接口，验证其是否能根据不同的查询条件返回正确的商品数据；测试用户订单提交接口，检查订单是否能正确创建并返回订单号等信息。

对于第三方服务集成测试，着重关注服务的可用性、数据交互的准确性和安全性。以支付功能为例，模拟不同支付场景，验证支付流程是否顺畅，支付结果是否能及时回调并更新订单状态；对于物流查询功能，测试是否能准确获取物流信息并展示给用户。

## 10.3 系统测试

### 10.3.1 功能测试

（1）用户登录测试。在登录模块页面，通过输入三个用户名和密码进行测试。其中输入的用户名“aboin”和密码“aboin”是正确的，会出现登陆成功的提示信息，而第一个和第二个登录失败。第一个登录失败的原因是因为用户名和密码不匹配，数据库中无法找到相应的匹配数据；第二个登录失败的原因是因为该用户名没有注册，数据库中没有相应的记录，所以第一次和第二次的登录会出现登录失败的得结果。如表6-1所示。

表6-1用户登录测试表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 测试用例 | 预期结果 | 实际结果 | 是否通过 |
|  |  |  |  |  |
| 登录模块 | 用户名：abo 密码：123 | 弹出错误提示，提示密码错误 | 弹出错误提示，提示密码错误 | 通过 |
| 登录模块 | 用户名：123  密码：aboin | 弹出错误提示，提示用户名错误 | 弹出错误提示，提示用户名错误 | 通过 |
| 登录模块 | 用户名：aboin  密码：aboin | 管理员登录成功 | 管理员登录成功 | 通过 |

（2）删除分类测试。在删除分类模块，找到相对应的删除信息，点击“删除”该信息的接口，删除成功后信息会在数据库中的表中进行删除，然后返回前端页面删除成功的提示框。通过一次又一次的删除操作，发现该系统的删除功能以及后端代码运行正常，多次的删除测试说明该系统的删除分类功能没有问题。如表6-2所示。

表6-2删除分类测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 测试用例 | 预期结果 | 实际结果 | 是否通过 |
| 删除分类模块 | 分类名：最新通知 | 删除成功、页面自动跳转 | 删除成功、页面自动跳转 | 通过 |

（3）修改密码测试。在修改密码模块，通过三个用户的修改数据进行该功能的测试。测试中发现只有第三次的修改密码成功了，第一次和第二次的密码修改出现了问题。问题的原因分别为“密码错误”和“确认密码不一致”的问题。通过多次的修改和测试，发现修改密码这一功能运行正常，没有问题。如表6-3所示。

表6-3修改密码测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 测试用例 | 预期结果 | 实际结果 | 是否通过 |
| 修改密码模块 | 原密码：666  新密码：123  确认密码：123 | 弹出错误提示，提示原密码错误 | 弹出错误提示，提示原密码错误 | 通过 |
| 修改密码模块 | 原密码：aboin 新密码：123  确认密码：333 | 弹出错误提示，提示确认密码不一致 | 弹出错误提示，提示确认密码不一致 | 通过 |
| 修改密码模块 | 原密码：aboin 新密码：123  确认密码：123 | 密码修改成功 | 密码修改成功 | 通过 |

通过上述一次又一次的测试修改，又加上对系统的白盒测试和黑盒测试等一系列的操作，对个性化推荐电商平台进行了多次全面具体的测试。该系统的各个功能模块都能正常运行，性能良好，占据CPU的空间也较小。对用户和商家都有很大的帮助，提供了便利，对账户信息有较强的保密性，后台数据库能保持良好的完整性和安全性。无论是前端的页面设计，还是后端的数据库和代码运行，都十分不错，达到了预期的效果。

### 10.3.2 性能测试

借助JMeter工具模拟高并发场景，测试系统在不同负载下的性能表现。设置不同的并发用户数、请求频率等参数，监测系统响应时间、吞吐量、资源利用率等关键指标。确保系统在高峰时段仍能保持快速响应，为用户提供了一个流畅的购物体验。

### 10.3.3 兼容性测试

在多种主流浏览器（如Chrome、Firefox、Safari、Edge）及不同版本上进行兼容性测试，确保页面显示正常，功能不受影响。

## 10.5 移动端专项测试

### 10.5.1 网络环境测试

在不同网络环境下（如WiFi、4G、5G、弱网络等），测试移动端应用的网络请求是否稳定、快速。验证页面加载速度、API请求响应时间、数据同步效果等指标，确保用户在各种网络条件下都能正常使用应用。

### 10.5.2 电量与内存测试

监测应用在运行过程中的电量消耗和内存占用情况。优化应用的代码和资源配置，降低电量消耗和内存占用，提升应用的续航能力和运行稳定性。

## 10.6 测试部署与CI/CD

将自动化测试脚本整合到GitHub Actions或其他CI/CD系统中。在代码提交、合并请求等环节自动触发测试流程，及时发现代码问题。利用CI/CD工具实现自动化构建、部署和测试，提高测试效率和软件质量。

# 十一、云计算服务应用 [李健彰]

## 11.1 云服务选型

## 择阿里云作为主要的云平台。阿里云在国内市场具有广泛的影响力和丰富的服务资源，提供稳定可靠的云计算服务。其数据存储具有高持久性和可用性，能够保障电商平台在高流量访问时数据的安全性和完整性。

## 11.2 云服务架构

### 11.2.1 计算服务（ECS、Lambda等）

采用阿里云的弹性计算服务（ECS），根据业务需求灵活选择不同配置的实例，如在促销活动期间可临时增加计算资源，活动结束后再进行释放，有效控制成本。

### 11.2.2 存储服务（OSS、数据库等）

使用阿里云对象存储服务（OSS）存储电商平台的大量非结构化数据，如商品图片、用户头像、日志文件等。OSS具有高可扩展性、安全性以及低成本的特点，能够满足电商平台对存储资源的弹性需求。

### 11.2.3 网络服务（CDN、负载均衡等）

借助阿里云的内容分发网络（CDN）服务，将电商平台的静态资源（如图片、CSS、JavaScript文件等）缓存到分布在全球各地的CDN节点上。

## 11.3 云原生应用

**容器化部署**：将电商平台的应用程序及其依赖打包成容器镜像，使用阿里云容器服务（ACK）进行容器化部署。

**Kubernetes编排**：利用Kubernetes（k8s）对容器化应用进行编排和管理。k8s可以自动调度容器的创建、销毁、重启等操作，确保应用的高可用性和故障恢复能力。

**Serverless应用**：对于电商平台的一些非核心业务功能，如订单通知、用户注册邮件发送等，采用阿里云函数计算（FC）实现Serverless架构。

# 十二、系统部署与运维 [郭欣凯、李健彰]

## 12.1 部署架构

采用Docker容器化技术结合阿里云容器服务（ACK）进行系统部署。具体步骤如下：

1. 将电商平台的应用程序及其依赖打包成Docker镜像。
2. 在阿里云容器服务上创建一个容器集群，并将镜像推送到阿里云容器镜像仓库。
3. 在容器集群中创建部署（Deployment），指定镜像版本、副本数量、资源限制等参数，实现应用的多副本部署，提高系统的可用性和容错能力。
4. 使用阿里云负载均衡服务（SLB）为容器集群设置外部访问入口，将用户请求分发到后端的容器实例上。

## 12.2 持续集成与部署

自动化部署流程、版本管理、回滚策略等。

**自动化部署流程**：搭建持续集成与持续部署（CI/CD）流水线，采用Jenkins作为CI/CD工具，集成Git代码仓库、Docker镜像构建、容器集群部署等功能。

**本管理**：在Git仓库中使用分支策略进行版本管理。主分支（master）用于存放稳定版本的代码，开发分支（develop）用于整合开发人员的日常功能开发。

**回滚策略**：在CI/CD流水线中设置回滚机制，当部署新版本后出现异常（如应用崩溃、性能严重下降、关键功能不可用等），系统会自动触发回滚操作，将应用快速恢复到上一个稳定版本。

## 12.3 可观测性与监控

### 12.3.1 系统监控

使用阿里云监控服务（CloudMonitor）对云平台资源（如ECS实例、RDS数据库、OSS存储、SLB负载均衡等）进行全方位监控，实时采集资源的CPU使用率、内存使用率、磁盘I/O、网络流量等指标。

### 12.3.2 应用性能监控（APM）

引入阿里云应用性能监控服务（ARMS），深入监控电商平台应用的性能指标，如页面加载时间、API响应时间、数据库查询时间、服务调用链路等。

### 12.3.3 日志收集与分析

用阿里云日志服务（Log Service）收集电商平台产生的各类日志数据，包括应用日志、服务器日志、数据库日志等。

### 12.3.4 报警机制

合阿里云监控服务和应用性能监控服务，建立完善的报警机制。通过邮件、短信、钉钉机器人等多种方式发送报警通知，确保运维人员能够在第一时间收到告警信息。

## 12.4 运维管理

**系统维护**：定期对云平台资源进行巡检和维护，包括操作系统更新、安全补丁安装、磁盘清理等操作。

**故障处理**：建立完善的故障处理流程和应急响应机制，当系统出现故障时，能够迅速启动应急预案，组织运维人员进行故障排查和修复。

**性能调优**：根据系统监控和性能测试数据，对电商平台的性能进行持续优化。优化内容涵盖数据库查询优化、代码性能优化、服务器资源配置优化等多个方面。

# 十三、性能优化 [郭欣凯、李健彰]

## 13.1 前端性能优化

### 13.1.1 页面加载优化

采用资源预加载技术，对关键的CSS文件、JavaScript文件和图片资源进行预加载，确保用户在访问页面时这些资源能够快速加载完成。使用懒加载技术对页面中的非关键资源进行延迟加载，先加载页面上方可见区域的资源，待页面加载完成后，再逐步加载其他资源，提高页面的首屏加载速度。

### 13.1.2 资源优化

对图片资源进行压缩处理，采用合适的图片格式和压缩算法，在不影响视觉效果的前提下减小图片文件的大小。对CSS、JavaScript文件进行压缩和合并，去除文件中的多余空格、注释等信息，减小文件体积，减少HTTP请求次数。

### 13.1.3 渲染性能优化

采用CSS3动画和过渡效果代替传统的JavaScript动画，减轻浏览器的JavaScript引擎负担，提高动画渲染性能。

## 13.2 后端性能优化

### 13.2.1 数据库优化

对数据库表结构进行设计优化，确保数据的规范化和冗余度较低。建立合理的数据库索引，提高查询效率，但对于查询频率较低的字段避免过度索引。

### 13.2.2 缓存策略

在后端应用中广泛使用Redis缓存，将常用的查询结果（如商品分类、热门推荐商品、用户会话信息等）缓存到Redis中，避免频繁查询数据库。设置合理的缓存过期时间，根据数据的重要性和更新频率进行差异化管理。采用缓存穿透、缓存雪崩、缓存预热等机制解决缓存使用过程中的常见问题，提高缓存的可靠性和稳定性。

### 13.2.3 并发处理优化

代码层面，对共享资源的访问进行合理的锁控制，避免过度锁竞争

## 13.3 网络优化

### 13.3.1 HTTP优化

采用HTTP/2协议替代传统的HTTP/1.1协议，HTTP/2协议具有多路复用、头部压缩、服务器推送等优势，能够有效减少网络延迟，提高资源加载效率。

### 13.3.2 数据压缩

对后端返回的文本数据（如HTML、JSON、XML等）进行GZIP压缩，减小数据的传输体积，加快数据传输速度。对于图片、视频等二进制数据，采用合适的压缩算法进行压缩，在保证数据质量的前提下降低数据的大小，减轻网络带宽压力。

# 十四、功能展示 [郭欣凯、李健彰]

## 14.1 系统演示

**功能展示详见附件视频。**

## 14.2 用户体验测试

## 用户普遍认为系统界面设计简洁明了，易于上手操作，个性化推荐功能能够较为准确地呈现符合个人兴趣的商品，有效提升了购物效率和满意度。但在某些场景下，如商品详情页的加载速度和搜索结果的精准度方面，仍有改进空间。针对这些反馈，团队将对系统进行持续优化，以进一步提升用户体验。

## 14.3 性能测试结果

在不同负载条件下对系统进行了性能测试。测试结果显示，在正常负载下，系统响应迅速，页面加载时间短，能够为用户提供了流畅的购物体验。随着负载的增加，系统仍能保持较好的性能表现，但在高并发情况下，部分接口的响应时间略有增长。通过性能测试，团队识别了系统中的性能瓶颈，并制定了相应的优化策略，如优化数据库查询、增加服务器资源等，以确保系统在实际应用中能够稳定高效地运行，满足用户的需求。

# 十五、项目管理与协作 [郭欣凯、李健彰]

## 15.1 开发流程

团队采用敏捷开发方法论，具体遵循 Scrum 框架。将项目划分为多个迭代周期，每个迭代周期为两周。在每个迭代开始前，进行迭代计划会议，确定该迭代要完成的任务和目标。开发人员根据任务优先级和自身能力领取任务，并在每日站会上汇报工作进展、遇到的问题和当日计划。通过敏捷开发方法，团队能够快速响应需求变更，提高开发效率和产品质量。

## 15.2 协作工具

**版本控制** ：使用 Git 进行版本控制，团队成员通过 GitHub 平台进行代码的提交、合并和管理。

**项目管理** ：采用 Jira 进行项目管理和任务跟踪。产品负责人在 Jira 中创建用户故事、任务和缺陷等工单，分配给相应的开发人员和测试人员。

**团队协作工具** ：使用钉钉进行日常的沟通和协作，团队成员可以在钉钉群内交流工作问题、分享文档和资料、进行会议预约等。

## 15.3 质量保证

代码review、测试流程、质量控制措施。

**代码 review** ：团队严格执行代码 review 制度，所有代码在合并到主分支之前必须经过另一名开发人员的审查。

**测试流程** ：建立完善的测试流程，包括单元测试、集成测试、系统测试和验收测试等阶段。

**质量控制措施** ：引入持续集成和持续部署（CI/CD）工具，在代码提交后自动触发构建和测试流程，及时发现代码中的问题和构建失败的情况。

# 十六、成果与交付物 [郭欣凯、李健彰]

## 16.1 项目交付清单

* 前端代码: front项目
* 后端代码: backend项目
* 数据库脚本: backend/电商平台.sql(master分支)
* API文档: docs/final/api.md
* 原型设计文件: docs/design/目录
* 项目演示视频: docs/final/video
* 详细数据库表 docs/final/数据库
* 测试 docs/ SpringbootSchemaApplicationTests.java

## 16.2 知识产权与开源

本项目的知识产权归属于开发团队，开源（Apache）。

# 十七、总结与展望 [郭欣凯、李健彰]

## 17.1 项目总结

项目团队通过不懈努力，成功完成了个性化推荐电商平台的设计与实现。在项目开发过程中，团队克服了诸多技术难题和挑战，按时交付了高质量的产品。实现了用户管理、商品管理、订单管理、个性化推荐、数据分析等核心功能模块，满足了用户对高效、便捷、个性化购物体验的需求。

## 17.2 技术收获

## 团队成员在项目中获得了丰富的技术收获和成长。在前端开发方面，深入掌握了 Vue.js 框架的应用和优化技巧，积累了大量关于响应式设计、组件化开发、页面性能优化的实践经验；在后端开发方面，熟练运用 Spring Boot 框架构建高性能、可扩展的后端服务，对微服务架构、分布式系统、数据库优化等技术有了更深入的理解和实践。

## 17.3 问题与反思

在需求分析阶段，部分需求不够明确和详细，导致开发过程中出现需求变更和功能调整的情况，影响了项目进度和开发效率。反思发现，在需求调研和沟通阶段需要更充分地与业务部门和用户进行交流和确认，采用原型设计、需求评审等方法，确保需求的准确性和完整性。

## 17.4 未来展望

**功能扩展** ：根据用户的反馈和市场的需求，持续优化现有功能，增加新的业务模块和功能特性。例如，拓展商品品类，增加跨境电商业务；优化个性化推荐算法，提高推荐的精准度和个性化程度；引入社交分享功能，增强用户的互动和粘性；开发移动端应用，提升用户在移动设备上的购物体验等。

## 17.5 商业化前景

# 本项目具有广阔的商业化前景。随着互联网技术的不断发展和消费者购物习惯的改变，电商平台市场呈现出持续增长的趋势。本项目的个性化推荐功能能够提高用户的购物效率和满意度，增加用户的购买转化率和复购率，为商家带来更多的销售机会和收益。

**附录**

## 附录A：详细API文档

## 附录B：数据库表结构