

吉林大学 计算机科学与技术学院

**2024软件工程课程设计文档**

**组号：**

**小组成员：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **学号** | **姓名** | **贡献**  **百分比** | **手写/电子签名** |
| 1 |  |  | 20 |  |
| 2 |  |  | 20 |  |
| 3 |  |  | 15 |  |
| 4 |  |  | 15 |  |
| 5 |  |  | 15 |  |
| 6 |  |  | 15 |  |

**注：贡献百分比必填，手写/电子签名必填**

**目录**

[1 开发计划 4](#_Toc177628671)

[1.1 项目开发背景 4](#_Toc177628672)

[1.2 项目开发设计 4](#_Toc177628673)

[1.2.1 项目开发方法：面向对象开发方法 4](#_Toc177628674)

[1.2.2 项目开发过程：敏捷开发过程 4](#_Toc177628675)

[1.2.3 编程语言和框架：SpringBoot + Vue.js 5](#_Toc177628676)

[1.2.4 版本管理工具：Git 5](#_Toc177628677)

[1.2.5 DevOps 6](#_Toc177628678)

[1.3 工作内容 7](#_Toc177628679)

[1.3.1 工作内容概述 7](#_Toc177628680)

[1.4.1 任务分解和分工 8](#_Toc177628681)

[1.4.2 进度计划 8](#_Toc177628682)

[2 需求分析 10](#_Toc177628683)

[2.1 登录需求 10](#_Toc177628684)

[2.1.1 用例描述 10](#_Toc177628685)

[2.1.2 概念建模 10](#_Toc177628686)

[2.2 目录浏览需求 12](#_Toc177628687)

[2.2.1 用例描述 12](#_Toc177628688)

[2.2.2 概念建模 13](#_Toc177628689)

[2.3 购物车 15](#_Toc177628690)

[2.3.1 用例描述 15](#_Toc177628691)

[2.3.2 概念建模 16](#_Toc177628692)

[2.4 结账 18](#_Toc177628693)

[2.4.1 用例描述 18](#_Toc177628694)

[2.4.2 概念建模 18](#_Toc177628695)

[2.5 订单查询 20](#_Toc177628696)

[2.5.1 用例描述 20](#_Toc177628697)

[2.5.2 概念建模 21](#_Toc177628698)

[2.6 评论 23](#_Toc177628699)

[2.6.1 用例描述 23](#_Toc177628700)

[2.6.2 概念建模 24](#_Toc177628701)

[3 系统设计 26](#_Toc177628702)

[3.1 总体设计 26](#_Toc177628703)

[3.1.1 需求规定 26](#_Toc177628704)

[3.1.2 运行环境 26](#_Toc177628705)

[3.1.3 架构设计 26](#_Toc177628706)

[3.2 模块设计 27](#_Toc177628707)

[3.2.1 用户模块 27](#_Toc177628708)

[3.2.2 产品模块 32](#_Toc177628709)

[3.2.3 购物车模块 35](#_Toc177628710)

[3.2.4 订单模块 38](#_Toc177628711)

[3.2.5 评论模块 41](#_Toc177628712)

[4 测试计划 45](#_Toc177628713)

[4.1 简介 45](#_Toc177628714)

[4.2 测试范围 45](#_Toc177628715)

[4.3 测试目的 45](#_Toc177628716)

[4.4 测试环境 45](#_Toc177628717)

[4.5 角色定义 45](#_Toc177628718)

[4.6 用例描述 46](#_Toc177628719)

[4.6.1 登录 46](#_Toc177628720)

[4.6.2 浏览目录 46](#_Toc177628721)

[4.6.3 添加商品 46](#_Toc177628722)

[4.6.4 删除商品 47](#_Toc177628723)

[4.6.5 清空购物篮 47](#_Toc177628724)

[4.6.6 提交订单 47](#_Toc177628725)

[4.6.7 查看订单历史 48](#_Toc177628726)

[4.6.8 评估商品 48](#_Toc177628727)

[4.6.9 查看评论 49](#_Toc177628728)

[4.6.10 登出 49](#_Toc177628729)

[5 测试报告 50](#_Toc177628730)

[5.1 登录模块测试报告 50](#_Toc177628731)

[5.1.1 网络脱机 50](#_Toc177628732)

[5.1.2 输入错误用户名 51](#_Toc177628733)

[5.1.3 输入错误密码 52](#_Toc177628734)

[5.1.4 正确登录页面初始情况 52](#_Toc177628735)

[5.2 购物车和结算模块 53](#_Toc177628736)

[5.2.1 第一部分：添加商品的测试 53](#_Toc177628737)

[5.2.2 第二部分：删除购物车的商品 54](#_Toc177628738)

[5.2.3 第三部分：清空购物车 56](#_Toc177628739)

[5.2.4 第四部分：提交订单，进行结算 57](#_Toc177628740)

[5.3 订单管理和展示模块 62](#_Toc177628741)

[5.3.1 第一部分：用户浏览的测试 62](#_Toc177628742)

[5.3.2 第二部分：订单管理的测试 65](#_Toc177628743)

[5.4 评价模块 66](#_Toc177628744)

[5.4.1 打开我的订单 66](#_Toc177628745)

[5.4.2 选择一个订单 66](#_Toc177628746)

[5.4.3 选择一个未评价的订单进行评价 67](#_Toc177628747)

[5.4.4 若不打星，只进行文字评价，显示评论保存失败 67](#_Toc177628748)

[5.4.5 若只打星，不进行文字评价，显示评论保存成功 68](#_Toc177628749)

[5.4.6 查看评论：点进主页的商品，可浏览评论： 69](#_Toc177628750)

[6 工作总结 70](#_Toc177628751)

# 开发计划

## 项目开发背景

系统名称： WebOrder 系统

开发人员： 软件工程项目组

用户： 在线音乐乐器购买者

**项目需求描述：**

WebOrder 系统的主要用户通过在线平台进行以下操作：

* 浏览音乐乐器产品目录
* 查看产品评论或评价
* 购买音乐乐器
* 查看过去订单的历史和详细信息

**次要（外部）系统：**

WebOrder 系统利用外部关系数据库管理系统（RDBMS）存储产品信息，如名称、零售价格、运输重量、评论等。此外，RDBMS 还存储用户账户信息（用户ID和密码）、订单历史记录以及运输数据（例如，当前的空运或陆运费用）。

## 项目开发设计

### 项目开发方法：面向对象开发方法

采用面向对象开发方法，完成体系结构分析、用例分析、子系统设计及类设计各步骤。面向对象编程（Object-Oriented Programming, OOP）通过将问题分解为多个对象来进行设计。OOP的核心概念包括对象、类、继承、封装、抽象、多态性和消息传递。这些概念有助于提高代码的可重用性和可维护性，使开发过程更具模块化和灵活性。

### 项目开发过程：敏捷开发过程

应用敏捷开发过程，使用腾讯TAPD平台来支持整个项目生命周期的管理。敏捷方法论强调迭代开发，各个阶段包括业务建模、需求分析、设计、实现、测试和部署。通过用户故事来定义需求，确保在每次迭代中实现实际的业务价值，提高开发响应速度和对变更需求的适应性。

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

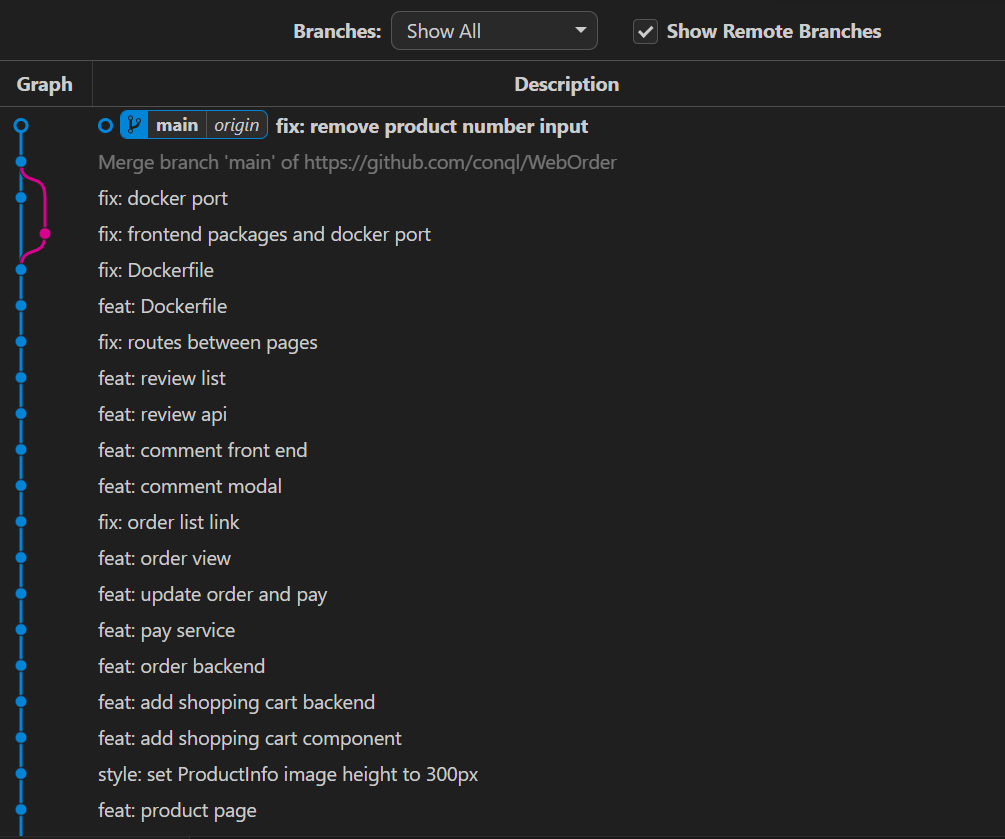
描述已自动生成

### 编程语言和框架：SpringBoot + Vue.js

项目采用BS架构，前端使用Vue.js框架，以提升客户端的交互体验和响应式设计能力。后端使用SpringBoot框架，以确保服务器端的高效开发及稳定性，支持微服务架构，增强系统的扩展性和可维护性。

### 版本管理工具：Git

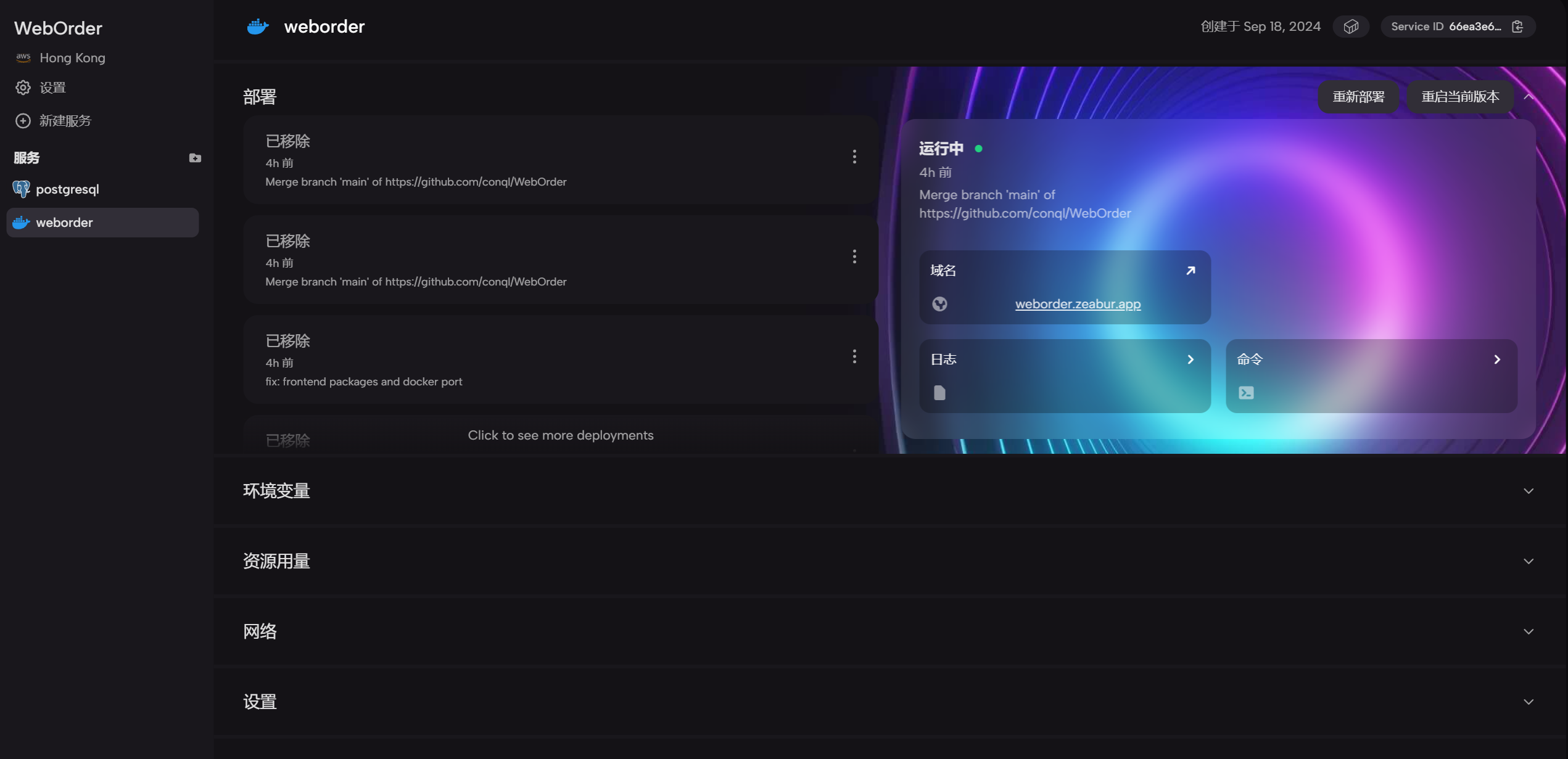
使用Git进行全面的源代码及相关文档管理。Git提供了分布式版本控制功能，便于团队协作，支持分支管理、代码合并和冲突解决，有效保障代码的一致性和开发过程的高效性。

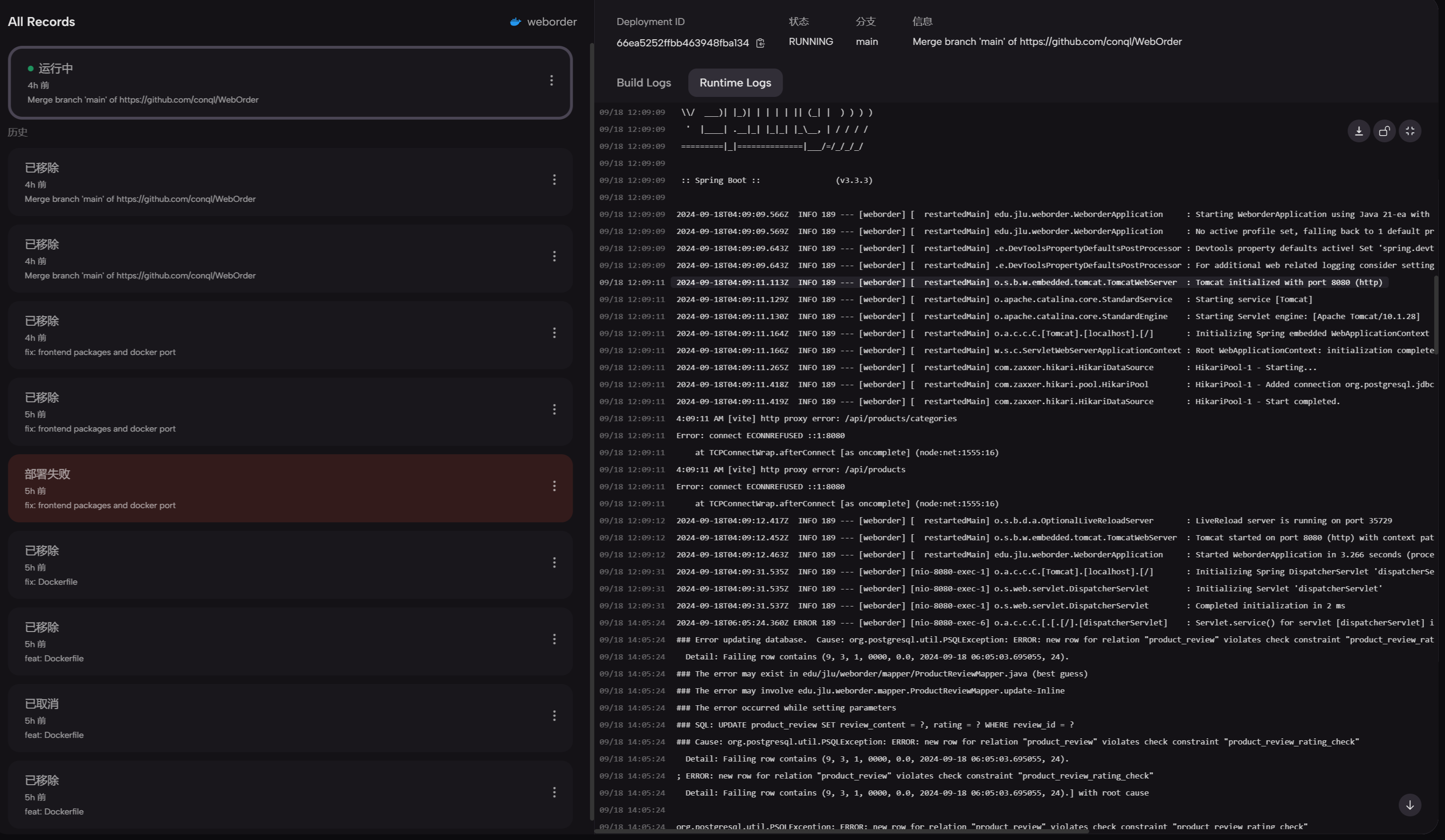


部分Git提交记录

### DevOps

为了便于开发人员部署程序、快速迭代更新，我们实现了DevOps过程，通过Github Hooks和Zeabur平台，自动化部署程序的Docker镜像，上线网站。践行了持续集成 / 持续交付的开发理念。





Zeabur网站部署后台

## 工作内容

### 工作内容概述

#### 项目文档

1. **项目开发计划**（包括人员安排，进度计划）
2. **需求分析文档**
3. **设计文档**
4. **测试报告**
5. **工作总结**：小组成员每人写一份关于自己所做工作的汇报，最后由组长汇总写一份全组工作的总结报告。

#### 迭代过程

1. **需求分析：**

用例描述：生成用例描述、用例图

概念建模：生成初始类列表、初始类图、关键事件序列图

1. **系统设计：**

总体设计：需求规定、运行环境、架构设计

模块设计：用户模块、产品模块、购物车模块、订单模块、评论模块

1. **测试计划：**

测试范围、测试目的、测试环境、角色定义实施计划

### 1.4.1 任务分解和分工

对于项目开发中需要完成的各项工作，从需求分析、设计、实现、测试直到维护，包括文件的编制、审批、打印、分发工作，用户培训工作，软件安装工作等，按层次进行分解，指明每项任务的负责人和参加人员。

|  |  |
| --- | --- |
| 黄光强21210629 | CI/CD流程，面向对象设计方法，软件配置管理，浏览产品的用例建模、登录和登录状态的用例建模、购物车和结算的用例建模 |
| 葛馨木21210620 | 面向对象设计方法应用，浏览不同分类产品的概念建模、浏览不同分类产品的设计建模，浏览不同产品分类的编码，产品评价浏览的用例建模、订单管理和展示的用例建模 |
| 周赛玥21210627 | 登录和登录状态的概念建模、登录和登录状态的设计建模、登录和登录状态的编写、登录和登录状态的测试 |
| 张益豪21210628 | 购物车和结算的概念建模、购物车和结算的设计建模、购物车和结算的编码、购物车和结算的测试 |
| 颜思宇21210630 | 产品评价浏览的概念建模，产品评价浏览的设计建模，评论模块的编码、评论部分的测试 |
| 蒋泽扬21210631 | 订单管理和展示的概念建模，订单管理和展示的设计建模，订单管理和展示的编码、订单管理和展示的测试 |

### 1.4.2 进度计划

整个项目周期为一个月，进度安排包括四周，如下:

第一周：

1. 面向对象方法设计： 面向对象编程（Object-Oriented Programming, OOP）通过将问题分解为多个对象来进行设计。OOP的核心概念包括对象、类、继承、封装、抽象、多态性和消息传递。这些概念有助于提高代码的可重用性和可维护性。
2. 创建CI/CD流程： 配置持续集成与持续交付（CI/CD）流水线以自动化构建、测试与部署流程。使用工具如Jenkins或GitLab CI实现代码变更的自动测试与发布，确保软件持续高质量交付。
3. 用户故事：浏览不同类型的产品列表： 允许用户在应用中浏览多个类别的产品，支持分类浏览，以及按价格、评分等条件进行筛选，以提升用户体验。

第二周：

1. 用户故事：登录： 实现用户登录功能，确保用户身份验证的安全性，使用加密技术保护用户信息。提供社交媒体账号登录集成功能以简化用户访问。
2. 用户故事：购物车与结算功能： 开发购物车模块，支持用户将产品添加到购物车、更新产品数量、删除产品并进行结算。集成支付网关以处理在线支付，确保交易的安全与可靠。

第三周：

1. 用户故事：产品评价浏览： 提供用户对产品进行评论与评分的功能，支持用户浏览其他用户的评价信息，以帮助购物决策。实现对不当内容的自动过滤和审核机制。
2. 用户故事：订单管理和展示： 开发用户订单管理界面，支持用户查看订单历史、状态更新和订单详情。添加订单追踪功能以提升用户的购物体验。

第四周：

1. 完成软件配置管理计划报告： 撰写配置管理计划文档，详细描述软件配置项（SCI）的管理与控制策略，确保版本控制的有效性与完整性。
2. 完成测试、进行工作回顾和总结： 使用单元测试、集成测试、系统测试进行全面软件测试，编写测试报告。召开回顾会议，总结项目阶段性成果与经验教训，为下一个迭代做好准备。

# 需求分析

## 登录需求

### 用例描述

用户登录并开始购物会话。

**参与者：**WebOrder 用户

**前置条件**

1. 用户已在系统中注册
2. 用户拥有有效的用户ID和密码。

**后置条件**

1. 用户成功登录
2. 用户看到欢迎信息和产品列表。

**主事件流**

1. WebOrder用户调用WebOrder应用程序并显示WebOrder屏幕。
2. 用户输入用户ID和密码。
3. WebOrder从RDBMS请求与给定用户ID关联的密码。
4. WebOrder验证给定密码是否等于存储的密码。
5. WebOrder初始化一个空的购物篮。
6. WebOrder显示欢迎信息，包括产品列表。

**备选事件流**

1. **未知用户**：与正常情况相同，但在步骤2之后，WebOrder显示适当的错误消息（“登录失败”）并提供用户重试的机会。
2. **密码错误**：与正常情况相同，但在步骤3之后，WebOrder显示适当的错误消息（“登录失败”）并提供用户重试的机会。
3. **无法连接到RDBMS**：与正常情况相同，但在步骤2之后，WebOrder显示适当的错误消息（“系统错误：未建立连接”）并提供用户重试的机会。

### 概念建模

#### 初始类列表

* + - User

属性：userID, username, password

方法：validatePassword()

* + - ShoppingBasket

属性：items

方法：initialize(), addItem(), removeItem(), empty()

* + - WebOrder

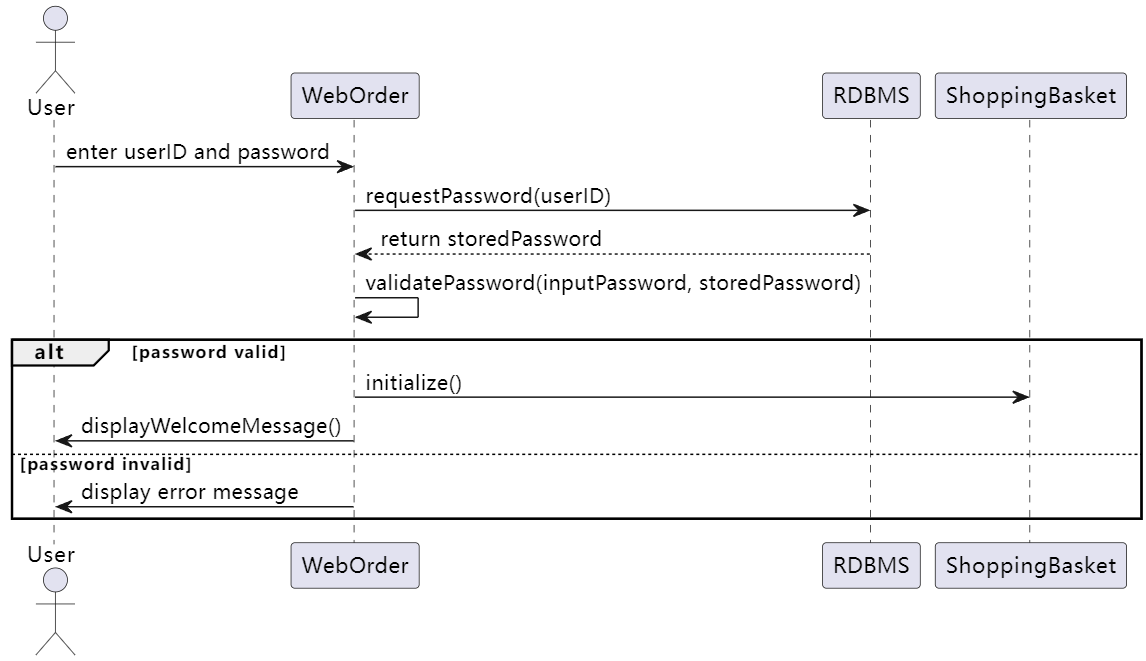
属性：N/A

方法：displayWelcomeMessage(), requestPassword(), validatePassword(), initializeShoppingBasket()

#### 初始类图



#### 序列图



## 目录浏览需求

### 用例描述

用户可以浏览系统中的产品分类，并查看每个产品的详细信息。

**参与者：**WebOrder 用户

**前置条件**

1. 用户已经通过身份验证并成功登录系统。
2. 系统中已经存在至少一个产品分类和产品。

**后置条件**

1. 用户能够查看产品的详细信息，包括名称、价格、描述、重量等。

**主事件流**

1. 用户登录成功后，系统展示产品分类的列表。
2. 用户从列表中选择一个产品分类。
3. 系统向数据库请求该分类的所有产品信息。
4. 系统展示选定分类下的所有产品。
5. 用户从产品列表中选择一个产品。
6. 系统查询并展示用户选择的产品信息，包括描述、价格、重量、产品图片等。

**备选事件流**

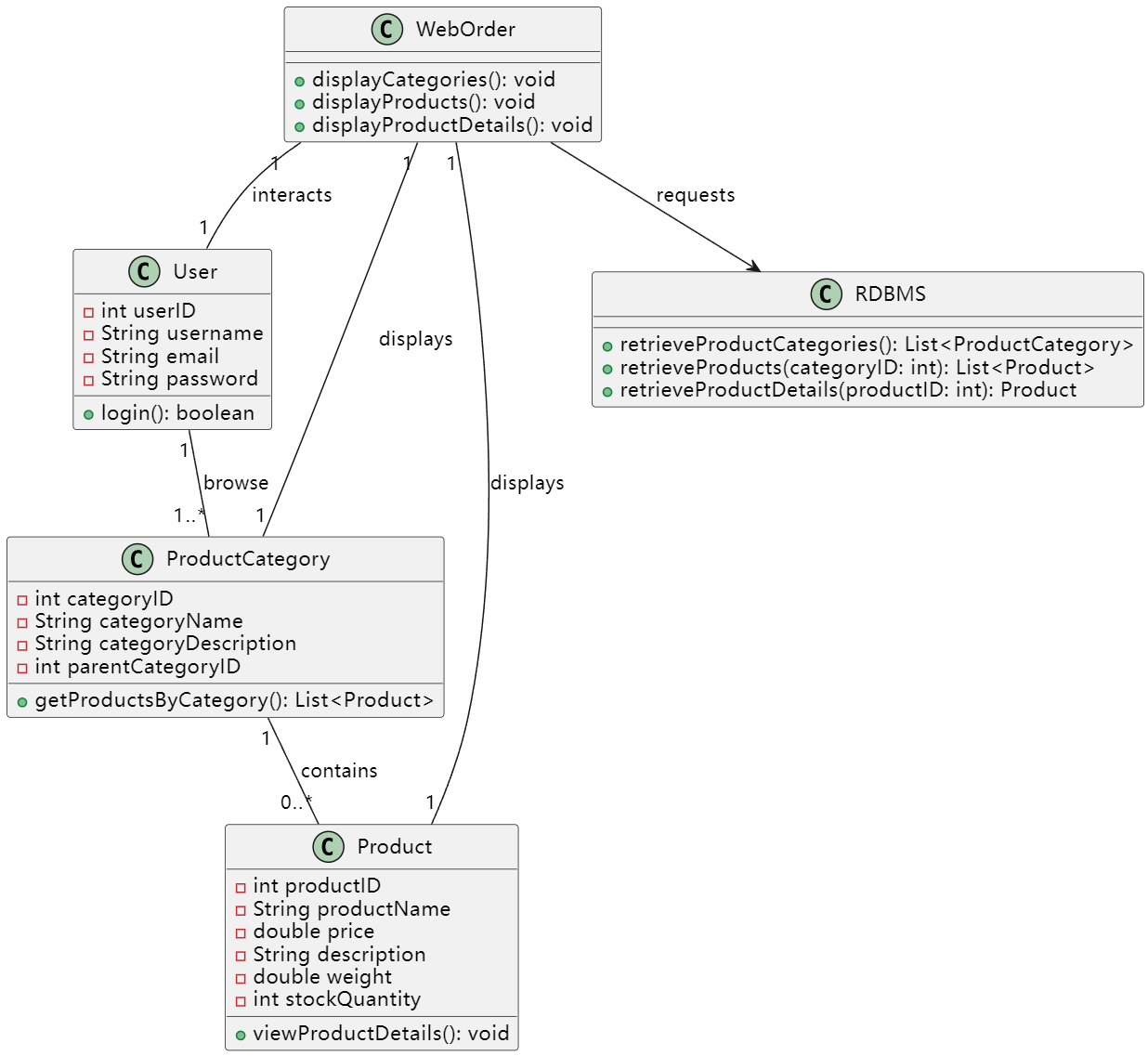
1. **分类为空**：如果某个产品分类下没有产品，系统将在步骤 4 显示“空列表”并返回主流程的第 1 步。
2. **未知产品**：如果产品在 WebOrder 实例运行时被删除，系统将在步骤 6 显示错误消息“404未找到该页面”，返回主流程的第 5 步。
3. **无法连接到 RDBMS**：WebOrder 显示适当的错误消息“系统错误：无法连接”并提供用户重试选项。

### 概念建模

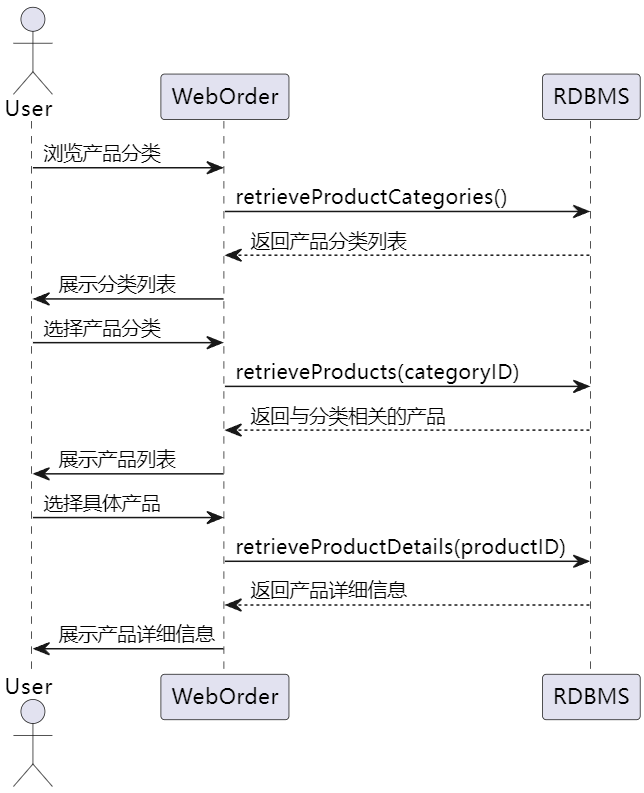
#### 初始类列表

1. User
   * 属性：userID, username, email, password
   * 方法：login()
2. Product
   * 属性：productID, productName, price, description, weight, stockQuantity
   * 方法：viewProductDetails()
3. ProductCategory
   * 属性：categoryID, categoryName, categoryDescription, parentCategoryID
   * 方法：getProductsByCategory()
4. WebOrder
   * 属性：N/A
   * 方法：displayCategories(), displayProducts(), displayProductDetails()
5. RDBMS
   * 方法：retrieveProductCategories(), retrieveProducts(), retrieveProductDetails()

#### 初始类图



#### 序列图



## 购物车

### 用例描述

**用例名称：添加商品**

**描述：**将商品添加到购物车。

**参与者：**WebOrder 用户

**前置条件：**

1. 用户已成功登录。
2. 用户正在浏览商品目录。

**后置条件：**

1. 商品成功添加到购物车。

**主事件流：**

1. 用户选择一个商品，并查看商品信息。
2. 用户输入购买数量，并点击“添加商品”按钮。
3. WebOrder请求选定产品的运输重量和当前单价从RDBMS。
4. WebOrder将指定数量的选定产品添加到购物篮：
5. 如果购物篮中已存在选定商品项，WebOrder将指定数量加入现有项。
6. 如果购物篮中不存在选定商品项，WebOrder为选定商品的新条目添加到购物篮。
7. WebOrder更新总运输重量和购买金额。
8. WebOrder更新购物车内容显示。

**备选事件流：**

1. **不合适的数量**：WebOrder在步骤2后将数量自动限制为大于0。
2. **无法连接到RDBMS**：WebOrder在步骤3后显示适当的错误消息（“系统错误：未建立连接”），并继续步骤1。

**用例名称：移除商品**

**描述：**从购物车中移除商品。

**参与者：**WebOrder 用户

**前置条件：**

1. 用户已成功登录。
2. 用户的购物车中存在商品。

**后置条件：**

1. 选定商品成功从购物车中移除。

**主事件流：**

1. 用户查看购物车中的商品。
2. 用户选择要移除的商品，并点击“移除商品”按钮。
3. WebOrder从购物篮中移除相应条目。
4. WebOrder更新总运输重量和购买金额。
5. WebOrder更新购物车内容显示。

**备选事件流：**

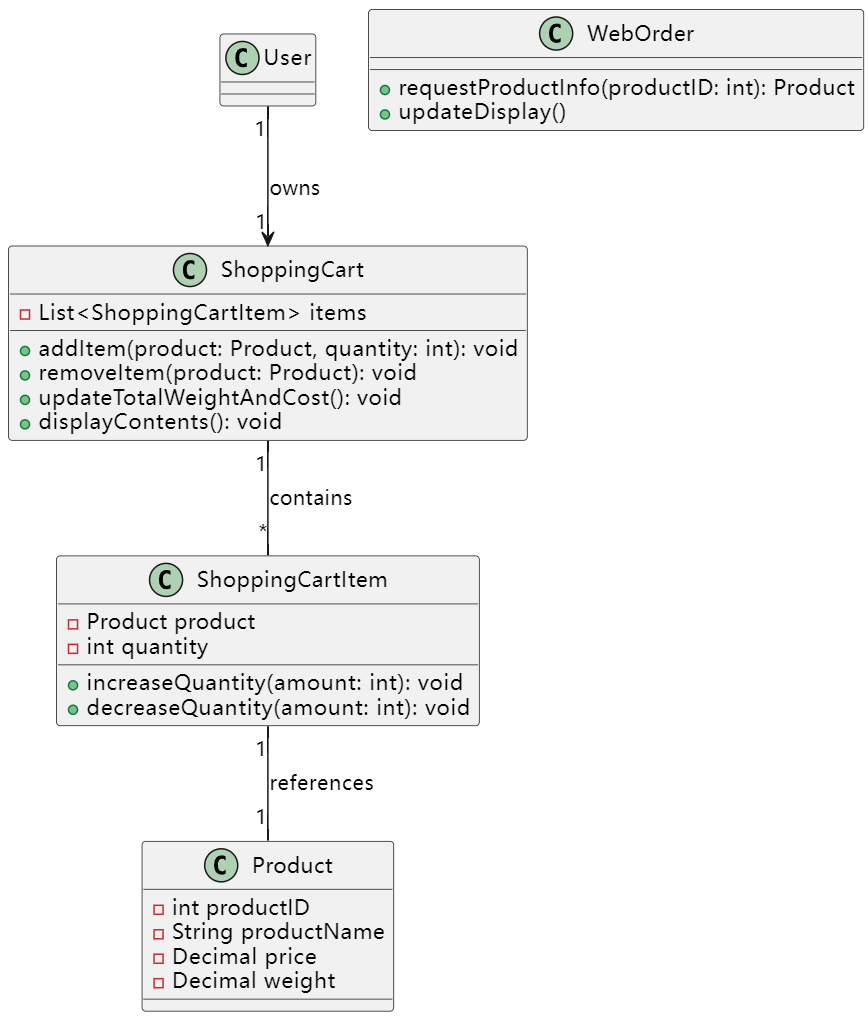
1. **购物车为空**：WebOrder在步骤2后显示适当的错误消息（“购物车中没有商品”），并继续步骤1。

### 概念建模

#### 初始类列表

1. Product
   * 属性：productID, productName, price, weight
   * 方法：N/A
2. ShoppingCartItem
   * 属性：product, quantity
   * 方法：increaseQuantity(), decreaseQuantity()
3. ShoppingCart
   * 属性：items
   * 方法：addItem(), removeItem(), updateTotalWeightAndCost(), displayContents()
4. WebOrder
   * 属性：N/A
   * 方法：requestProductInfo(), updateDisplay()
5. User
   * 属性：userID, username
   * 方法：N/A

#### 初始类图



#### 序列图



## 结账

### 用例描述

**用例名称：结账**

**描述：**用户提交购物车中的订单。

**参与者：**WebOrder 用户

**前置条件**

1. 用户已经登录系统。
2. 用户购物车不为空。

**后置条件**

1. 订单提交成功，用户购物车被清空。
2. 用户看到订单成功提交的确认信息。

**主事件流**

1. 用户在购物车中选择产品。
2. WebOrder更新总成本，包括运费和重量，并显示在屏幕上。
3. 用户选择配送方式（空运或陆运）并点击“提交订单”按钮。
4. WebOrder将订单信息更新到RDBMS中的订单历史记录。
5. 成功后，WebOrder清空购物车中的所有条目。
6. WebOrder将总运费和购买金额重置为零。
7. WebOrder更新购物车内容显示。
8. WebOrder向用户显示成功下单信息。

**备选事件流**

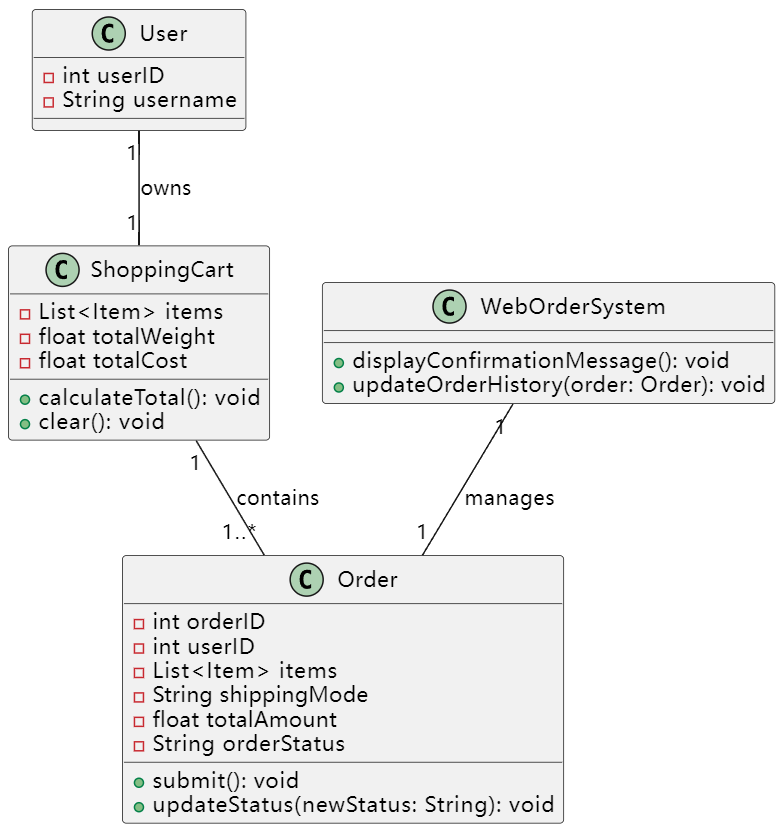
1. **购物车为空**：在步骤3后，WebOrder显示适当的错误屏幕（“购物车为空”）。
2. **无法连接到RDBMS**：在步骤4后，WebOrder显示适当的错误屏幕（“系统错误：未建立连接”），并返回步骤1。

### 概念建模

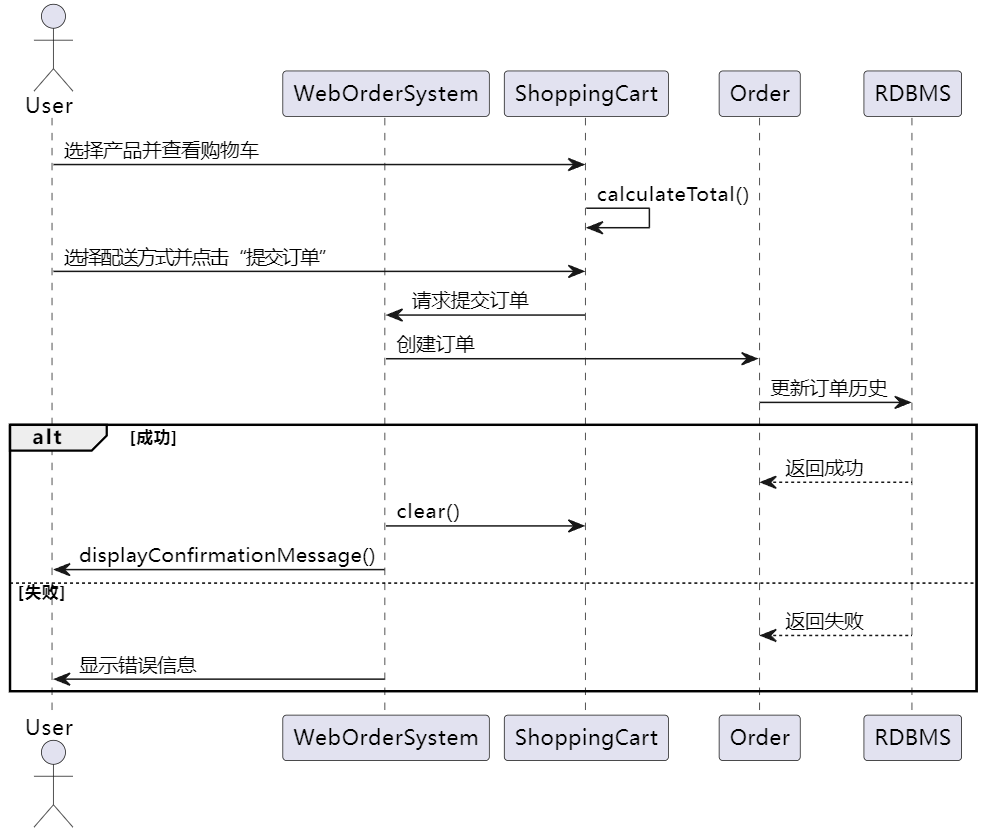
#### 初始类列表

1. User
   * 属性：userID, username
2. ShoppingCart
   * 属性：items, totalWeight, totalCost
   * 方法：calculateTotal(), clear()
3. Order
   * 属性：orderID, userID, items, shippingMode, totalAmount, orderStatus
   * 方法：submit(), updateStatus()
4. WebOrderSystem
   * 属性：N/A
   * 方法：displayConfirmationMessage(), updateOrderHistory()

#### 初始类图



#### 序列图



## 订单查询

### 用例描述

**用例描述：查看订单**

**描述：**用户查看历史订单的列表，以及订单的详细信息。

**参与者：**WebOrder 用户

**前置条件：**用户已成功登录系统。

**后置条件：**用户成功查看订单历史和详细信息。

**主事件流：**

1. 用户点击“订单历史”按钮。
2. WebOrder从RDBMS请求用户的订单历史信息。
3. 用户查看“订单历史”界面。
4. WebOrder显示用户提交的订单列表。

5a. 用户查看订单列表：

* + - 用户点击“完成”按钮。
    - “订单历史”窗口关闭，用户返回主屏幕。

5b. 用户查看订单详细信息：

* + - 用户从订单列表中选择一个订单。
    - WebOrder显示所选订单的详细信息：订单ID、日期、运输方式、总费用、总重量以及订单中的商品名称和数量。
    - 用户点击“完成”按钮。
    - “订单历史”窗口关闭，用户返回主屏幕。

**备选事件流：**

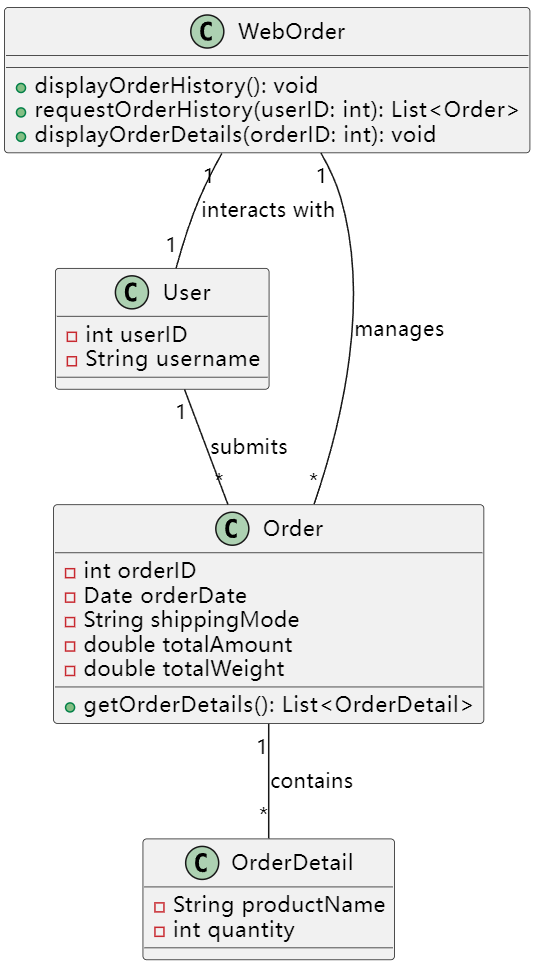
1. **订单历史为空**：在“订单历史”界面，订单列表为空。
2. **无法连接到RDBMS**：在步骤3之后，WebOrder显示适当的错误消息（“系统错误：未建立连接”）。

### 概念建模

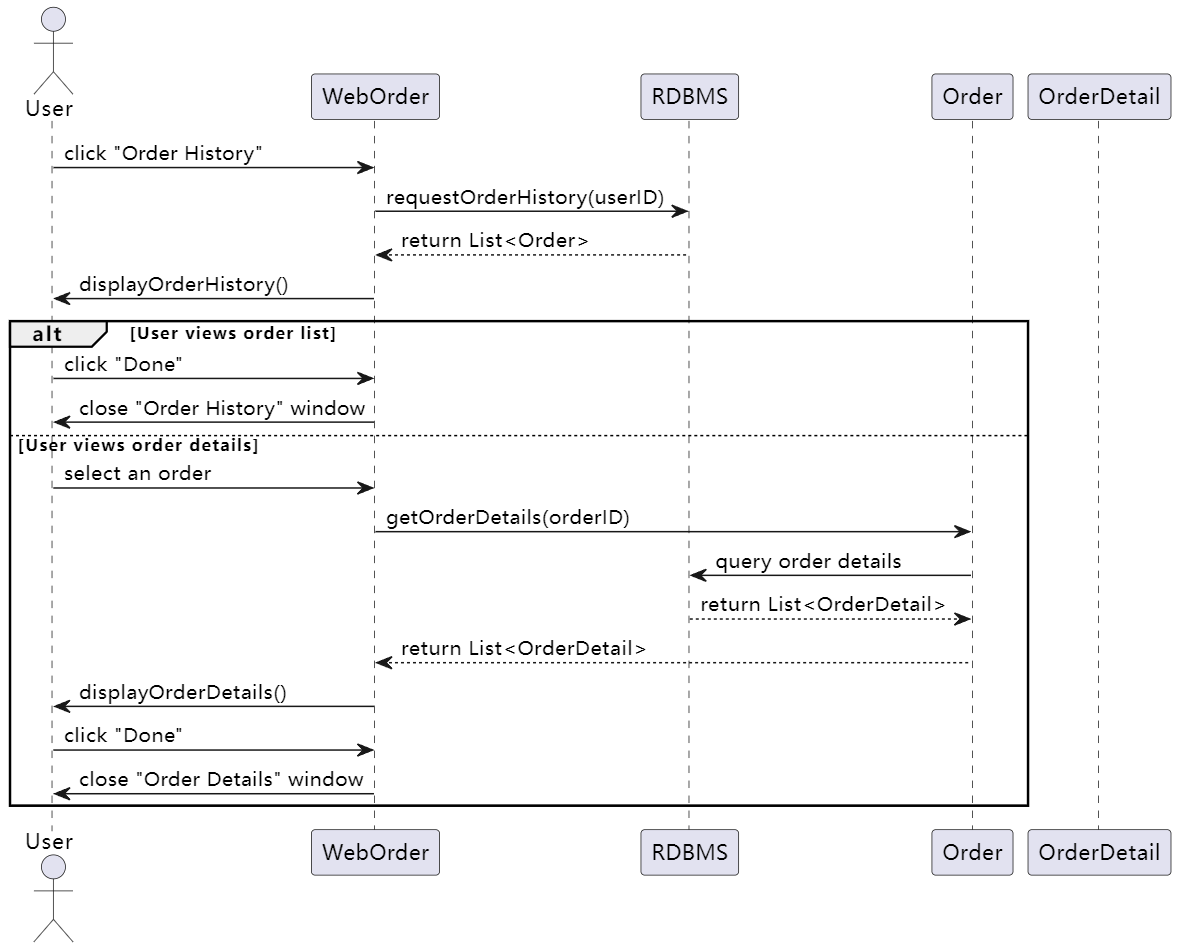
#### 初始类列表

1. User
   * 属性：userID, username
   * 方法：N/A
2. Order
   * 属性：orderID, orderDate, shippingMode, totalAmount, totalWeight
   * 方法：getOrderDetails()
3. OrderDetail
   * 属性：productName, quantity
   * 方法：N/A
4. WebOrder
   * 方法：displayOrderHistory(), requestOrderHistory(), displayOrderDetails()

#### 初始类图



#### 序列图



## 评论

### 用例描述

**用例名称：查看评论**

**描述**

用户查看产品的评论。

**参与者**

WebOrder 用户

**前置条件**

用户已登录系统

用户已选择了一个产品查看信息。

**后置条件**

用户可以查看所选产品的评论。

**主事件流**

* + 1. 用户查看产品信息后，选择“查看评论”按钮。
    2. WebOrder从RDBMS中请求所选产品的评论信息。
    3. 用户被呈现“产品评论”屏幕。
    4. WebOrder显示评论列表。
    5. 用户点击“完成”按钮。
    6. “产品评论”窗口关闭，用户被返回到主屏幕。

**备选事件流**

**产品无评论**：在步骤4，评论列表为空。

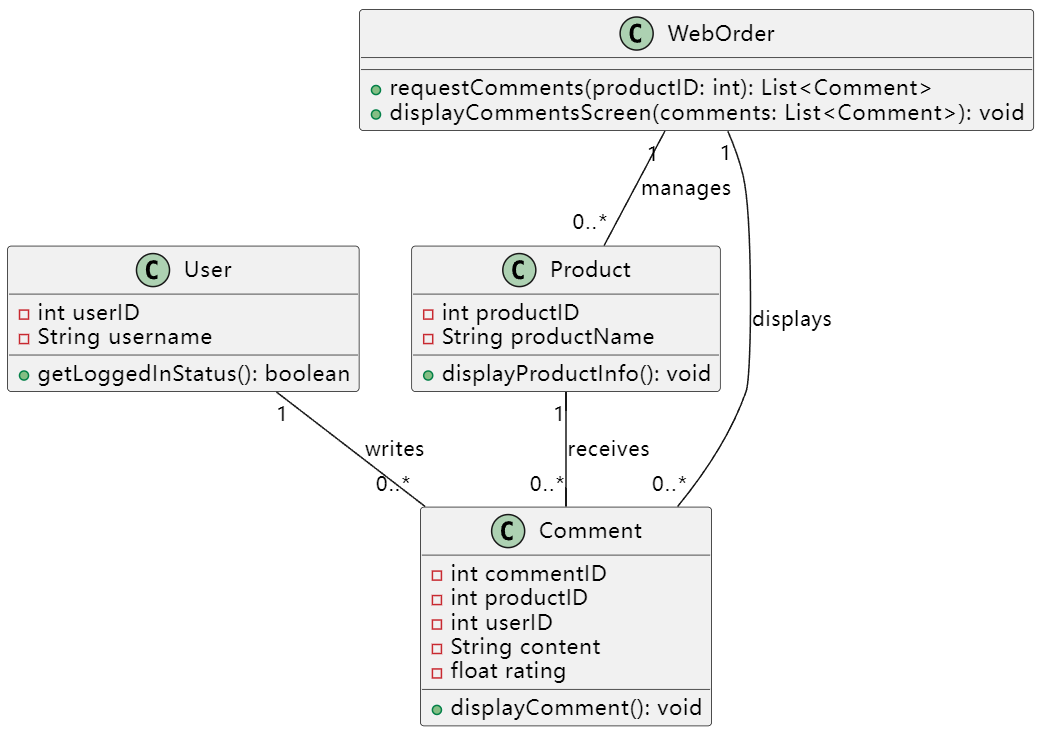
**无法连接到RDBMS**：在步骤2，WebOrder显示“系统错误：未建立连接”消息。

### 概念建模

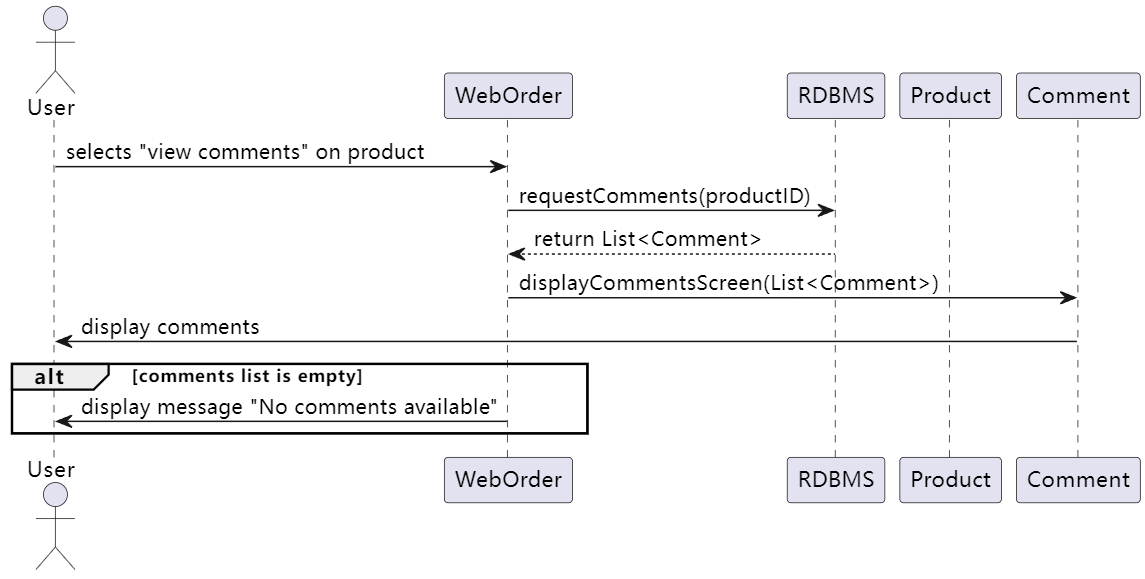
#### 初始类列表

1. User
   * 属性：userID, username
   * 方法：getLoggedInStatus()
2. Product
   * 属性：productID, productName
   * 方法：displayProductInfo()
3. Comment
   * 属性：commentID, productID, userID, content, rating
   * 方法：displayComment()
4. WebOrder
   * 属性：N/A
   * 方法：requestComments(productID: int), displayCommentsScreen()

#### 初始类图



#### 序列图



# 系统设计

## 总体设计

### 需求规定

WebOrder 是一个网上在线乐器购物平台，用户可以登录该网站，浏览不同乐器分类，查看乐器的图片和详细介绍，并下单购物，结算产生订单。系统应当支持订单状态查询、产品评价等功能。

### 运行环境

服务器：Ubuntu 22.04、Docker

客户端：支持 Chrome、Firefox 浏览器的PC或Android、IOS设备

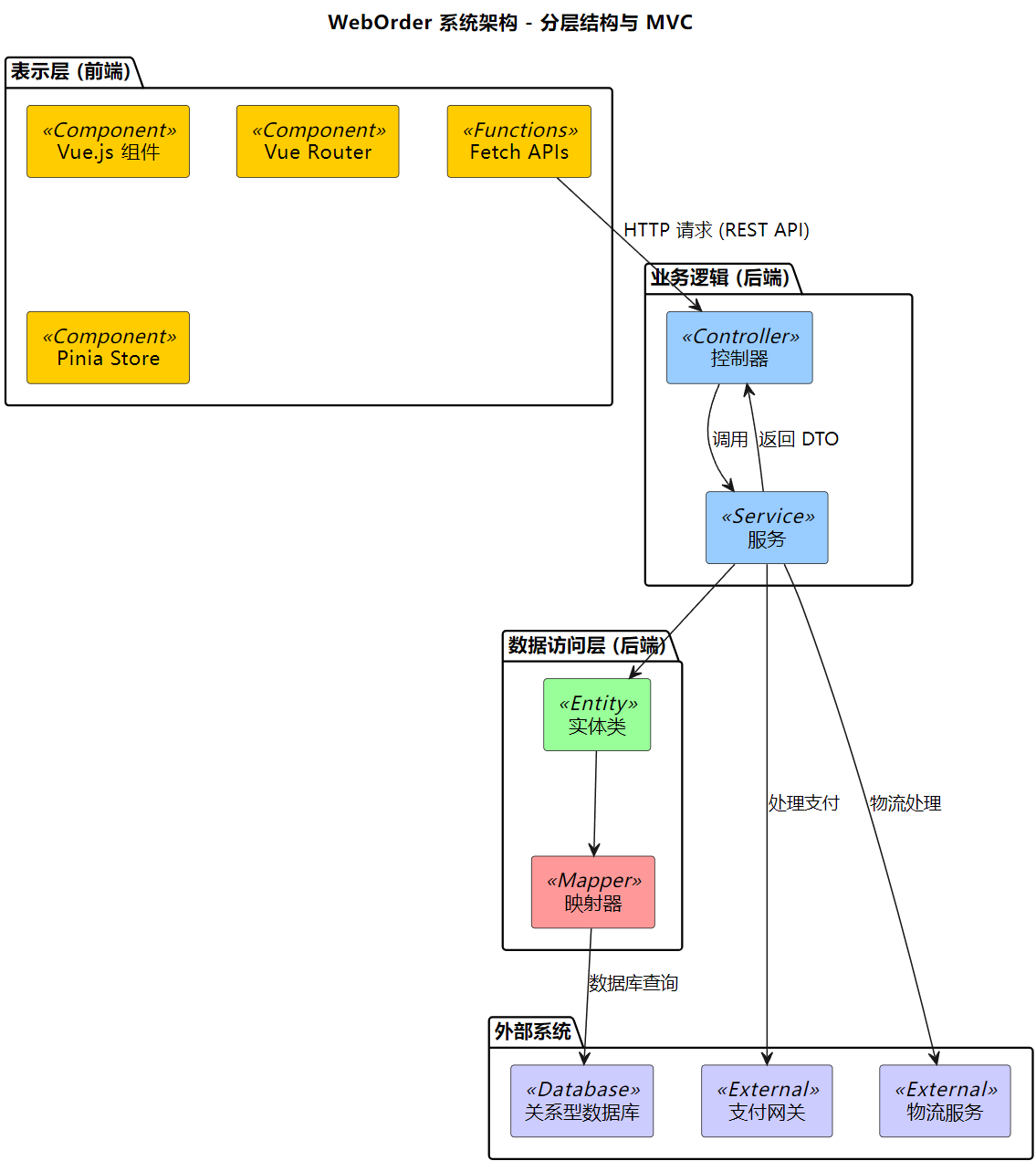


运行流程示意图

### 架构设计

本系统采用 B/S 架构。采用 Ubuntu 为操作系统的工作站，Apache、Tomcat 提供服务器功能，PostgreSQL 为开发软件的数据库服务程序。 前端采用 Vue 3框架，使用Vite、Pinia、Vue Router 以及 Ant Design Vue进行开发，编译打包生成HTML、Javascript、CSS文件。后端采用Java SpringBoot，MyBatis框架。通过 Github Actions 实现 CI/CD 流程，践行 DevOps 理念，自动部署到云服务器中的Docker容器中。

具体而言，该系统基于分层结构和 MVC 模式，分为四个主要部分：表示层（前端）、业务逻辑层（后端）、数据访问层（后端）和外部系统。前端使用 Vue.js 组件与状态管理工具 Pinia，通过 HTTP 请求调用后端控制器（Controller）。控制器协调应用层的服务（Service），处理业务逻辑。服务层同时与持久层进行交互，持久层由实体类（Entity） 和映射器（Mapper） 组成，负责与关系型数据库进行数据操作。整体架构清晰分工，前后端通过 REST API 分离，服务层贯通业务处理和外部系统的集成。



架构设计简图

## 模块设计

根据开发迭代计划，我们按照不同的产品需求和功能将系统分成各个大模块，每个模块都会贯穿不同的系统架构层次。设计的模块包括：

### 用户模块

#### 数据库设计

User表

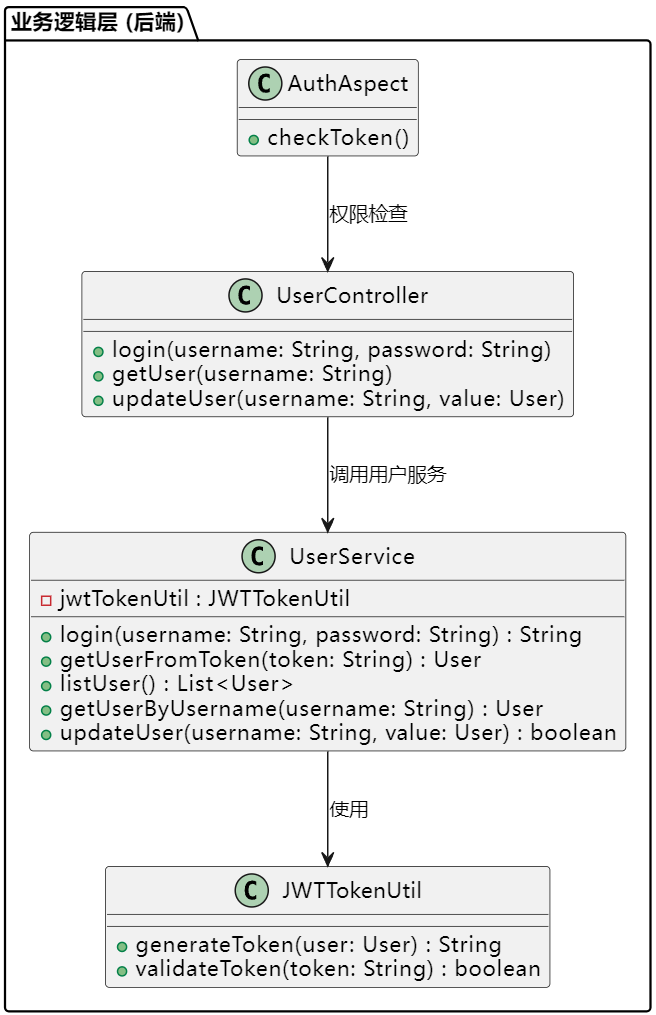
| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| user\_id | SERIAL | 主键，自增 |
| username | VARCHAR(50) | 用户名，不为空 |
| password | VARCHAR(255) | 密码，不为空 |
| phone | VARCHAR(15) | 电话号码 |
| email | VARCHAR(100) | 邮箱 |
| role | VARCHAR(50) | 用户角色，可为admin或user |

#### 详细类图



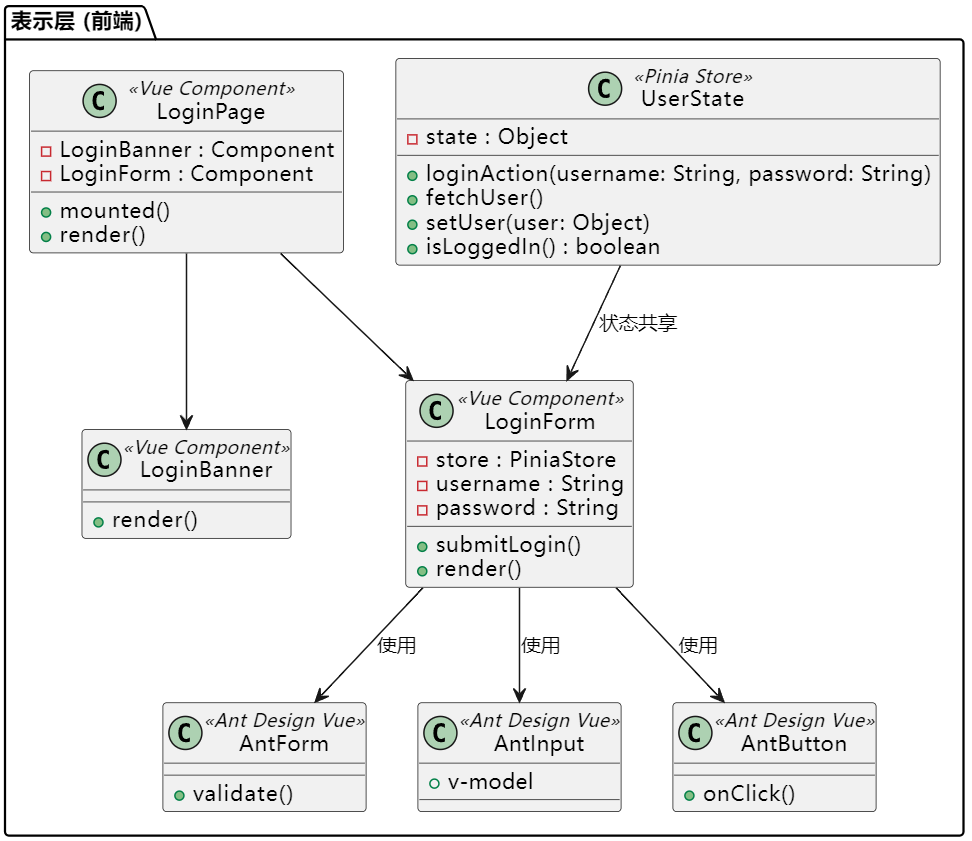
**说明：**

1. User：User实体，映射到数据库中的用户表，存储用户的基本信息。
2. UserMapper：提供用户表的基本读写操作。



**说明：**

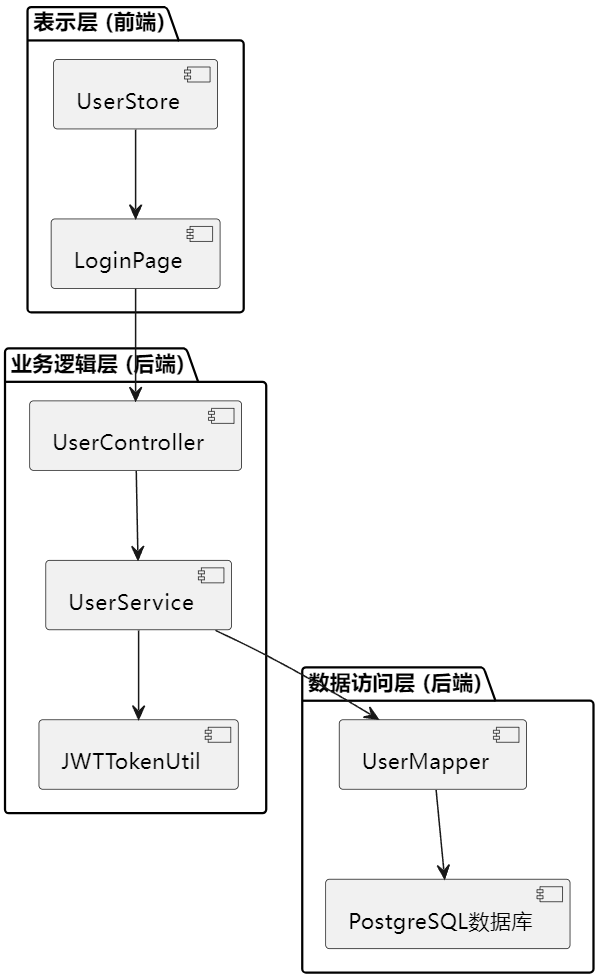
1. AuthAspect：通过 AOP 拦截器，在每个需要鉴权的接口请求之前，检查请求头中的 JWT Token 是否有效。
2. UserController：接受登录请求，调用 UserService 进行登录，并返回 JWT Token。
3. UserService：提供用户相关的业务逻辑，如通过用户名获取用户信息、保存用户信息等。此外，还处理登录请求，验证用户的用户名和密码，生成 JWT Token。在登录成功后，返回一个 Token 给前端。
4. JWTTokenUtil：用于生成和验证 JWT Token，提供必要的工具方法来处理 Token。



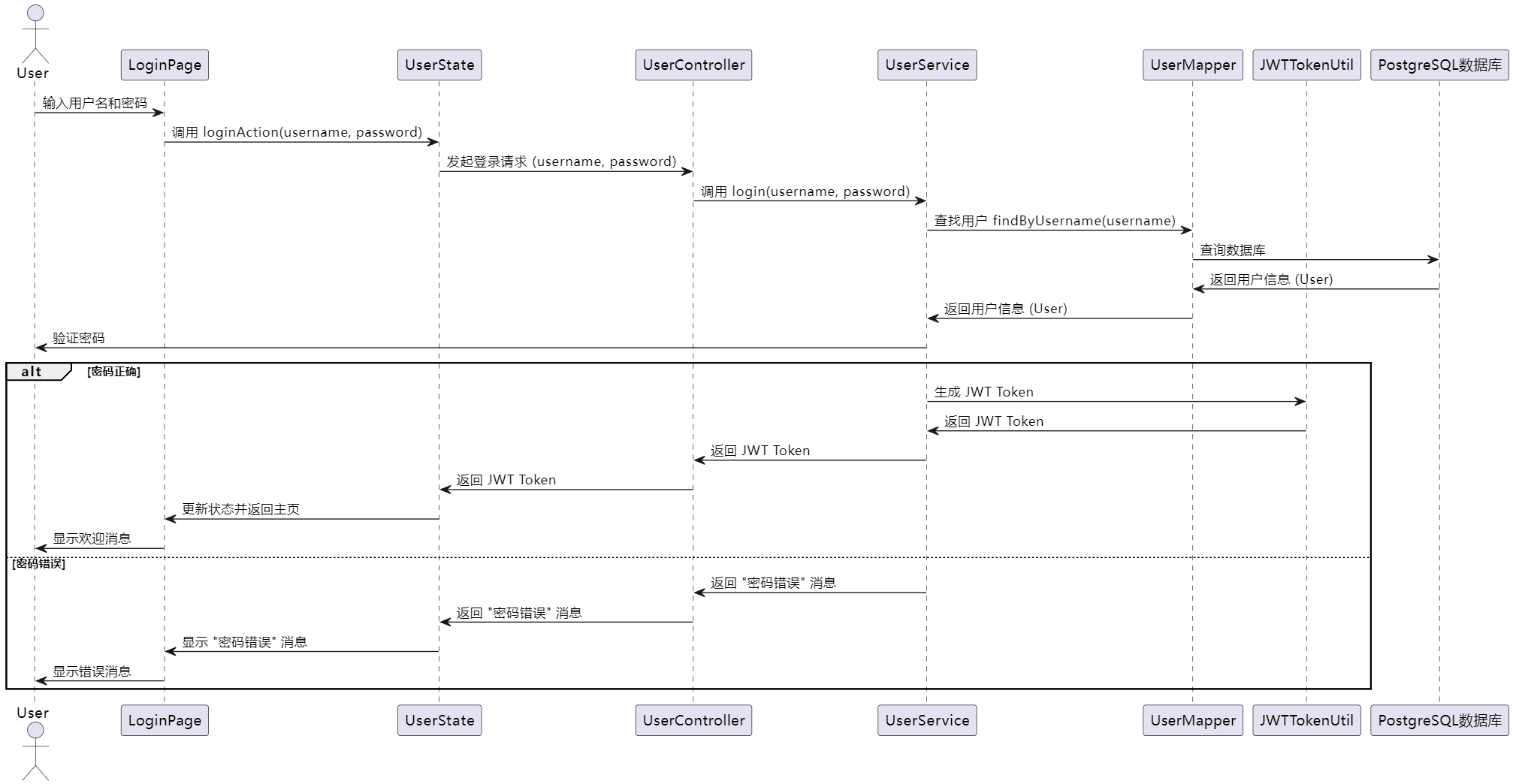
**说明：**

1. LoginPage：Vue 组件，包含 LoginBanner 和 LoginForm 子组件。
2. LoginForm：Vue 组件，负责处理用户输入，使用 Ant Design Vue 提供的 Form、Input 和 Button 组件来构建表单。
3. AntForm、AntInput 和 AntButton：来自 Ant Design Vue 的表单组件，用于构建登录表单。
4. UserState：使用 Pinia 全局状态管理工具，用于登录成功后保存用户状态。包含全局的状态（如用户信息），通过 loginAction 发送登录请求，并在 setUser 中更新用户状态。

#### 整体概览



#### 补充



用户登录过程

### 产品模块

#### 数据库设计

product表

| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| product\_id | SERIAL | 主键，自增 |
| product\_name | VARCHAR(100) | 产品名称，不为空 |
| price | DECIMAL(10,2) | 价格 |
| weight | DECIMAL(10,2) | 重量 |
| description | TEXT | 描述 |
| stock\_quantity | INT | 库存数量 |
| detail | TEXT | 详细信息 |

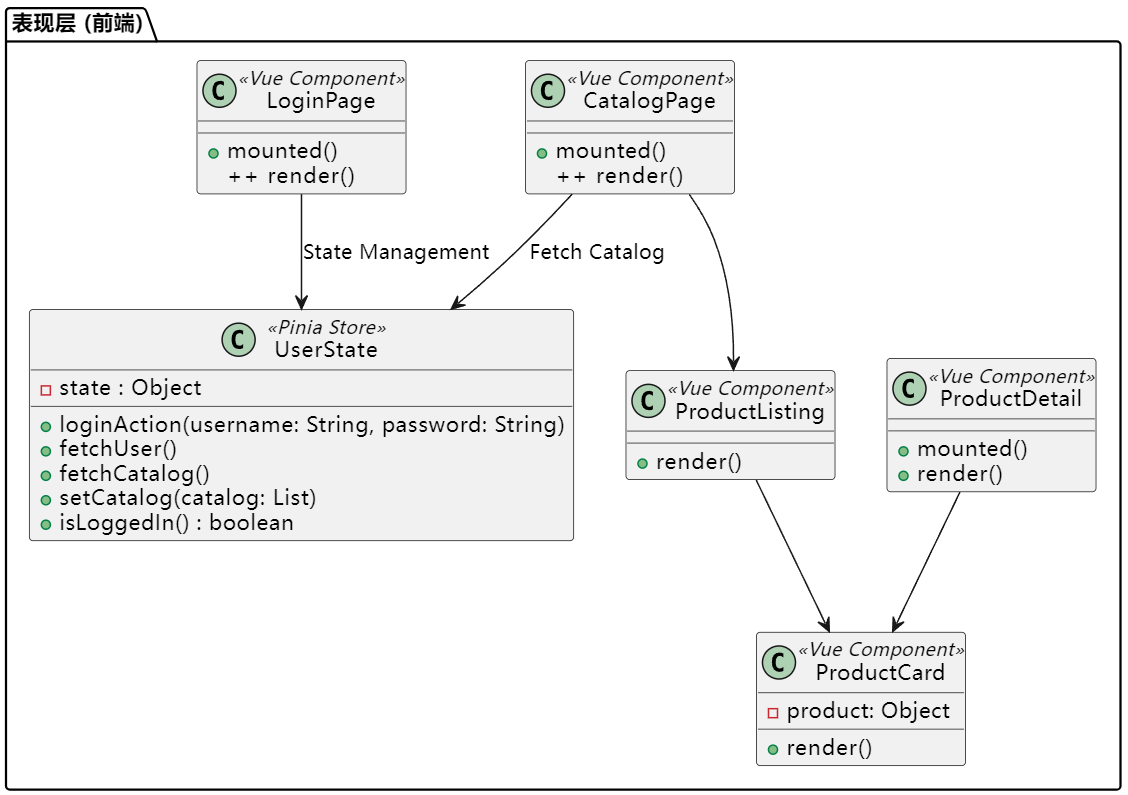
product\_category 表

| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| category\_id | SERIAL | 主键，自增 |
| category\_name | VARCHAR(100) | 分类名称，不为空 |
| category\_description | TEXT | 分类描述 |
| parent\_category\_id | INT | 父分类ID，引用到 self 表的 category\_id 字段，删除时置空 |

product\_category\_link 表

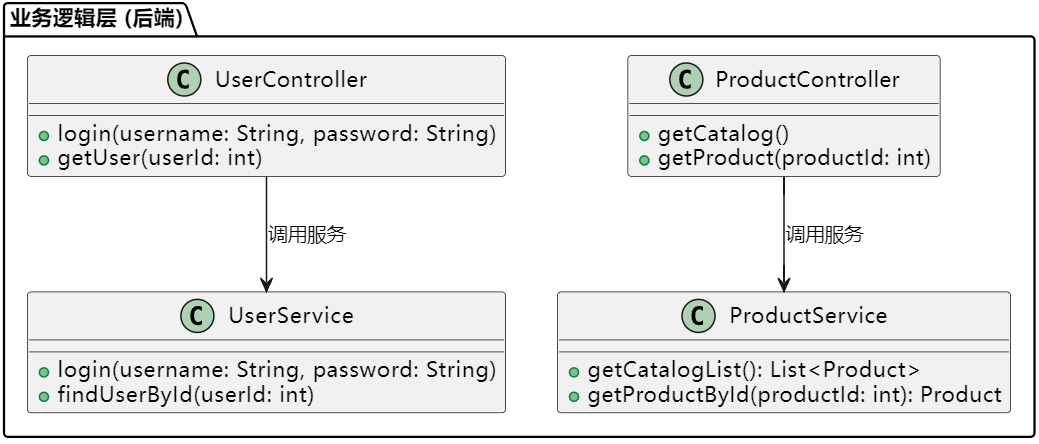
| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| link\_id | SERIAL | 主键，自增 |
| product\_id | INT | 产品ID，引用到 product 表的 product\_id 字段，删除时级联删除 |
| category\_id | INT | 分类ID，引用到 product\_category 表的 category\_id 字段，删除时级联删除 |

#### 详细类图



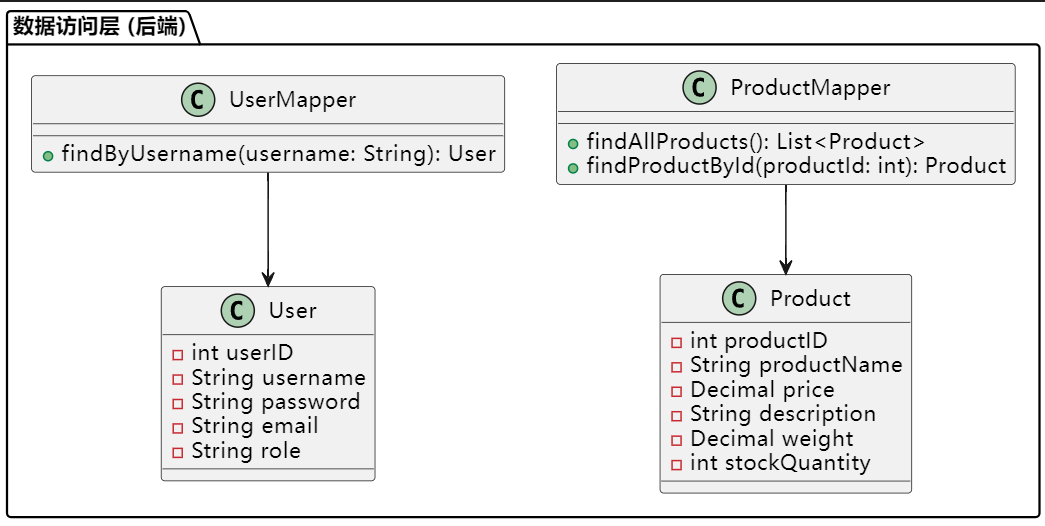
说明：

1. LoginPage：用户输入用户名和密码的页面。
2. CatalogPage：展示产品列表的页面。
3. ProductListing 和 ProductDetail：分别用于展示产品列表和产品详情。
4. UserState：使用 Pinia 实现的全局状态，保存用户数据并提供登录及其他数据操作的功能。



说明：

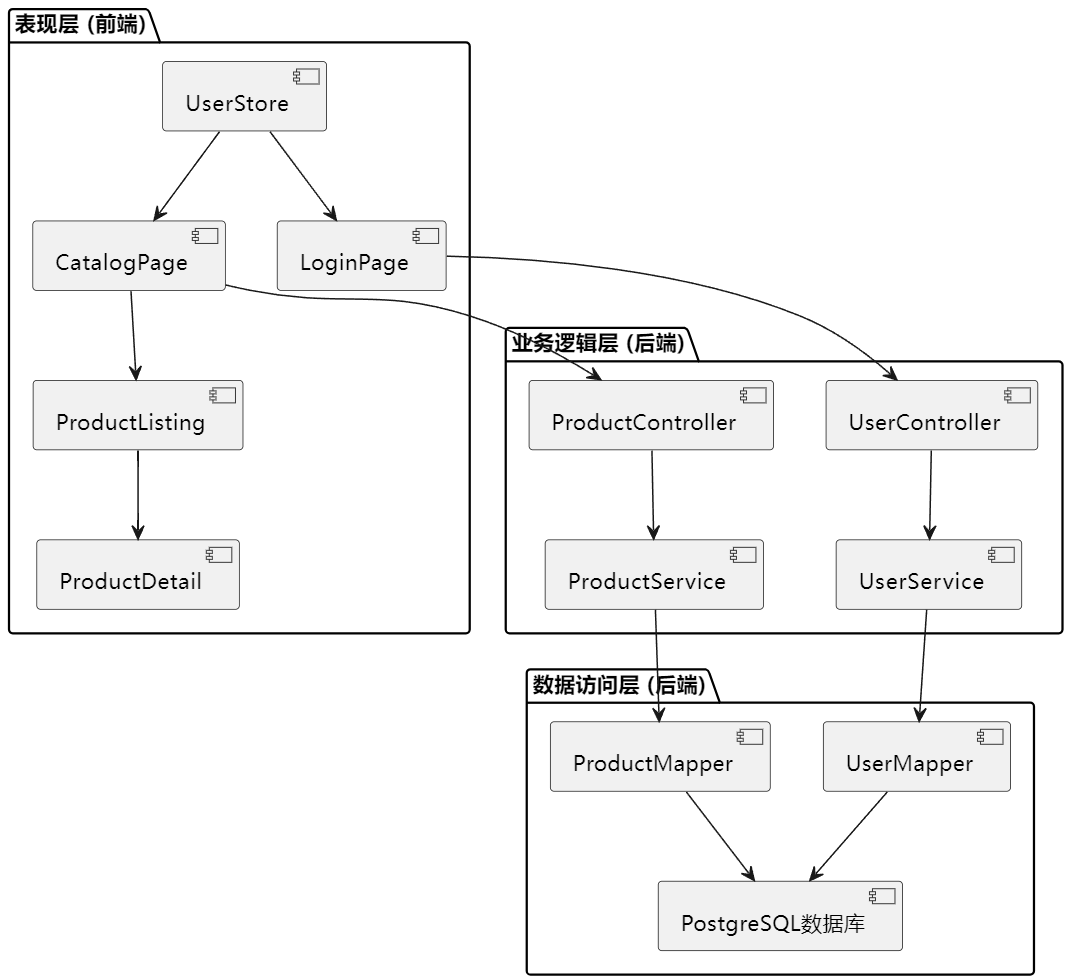
1. UserController：负责管理用户相关的 API，如登录和获取用户信息。
2. ProductController：负责操作产品相关的 API，如获取 产品 列表和产品详情。
3. UserService：管理与用户登录和鉴权逻辑相关的处理。
4. ProductService：处理产品数据的查询、从数据库加载产品等业务逻辑。



说明：

1. UserMapper：从数据库中读取用户相关的信息，例如通过 username 查找用户。
2. ProductMapper：负责从数据库中读取产品信息，根据 productID 获取产品详情。
3. User 和 Product 类分别映射到数据库中的用户表和产品表。

#### 整体概览



#### 补充

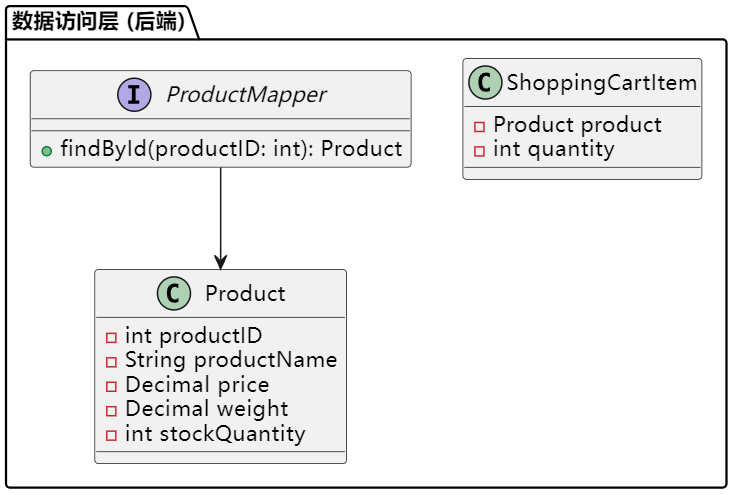
### 购物车模块

#### 数据库设计

shopping\_cart\_item 表

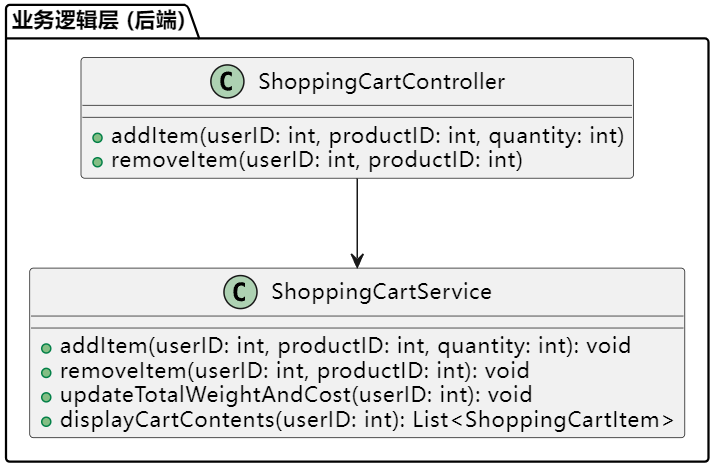
| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| item\_id | SERIAL | 主键，自增 |
| user\_id | INT | 用户ID，引用到 users 表的 user\_id 字段，删除时级联删除 |
| product\_id | INT | 产品ID，引用到 product 表的 product\_id 字段，删除时级联删除 |
| quantity | INT | 数量，不为空 |

#### 详细类图



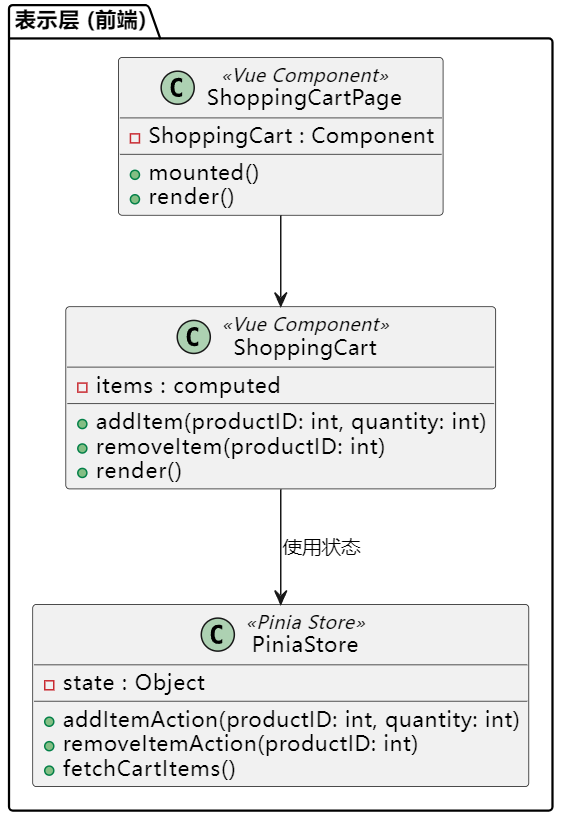
说明：

1. ShoppingCartItem：每个购物车中的产品，映射到数据库
2. Product：产品的详细信息，映射到数据库
3. ProductMapper：负责从数据库中读写产品信息



说明：

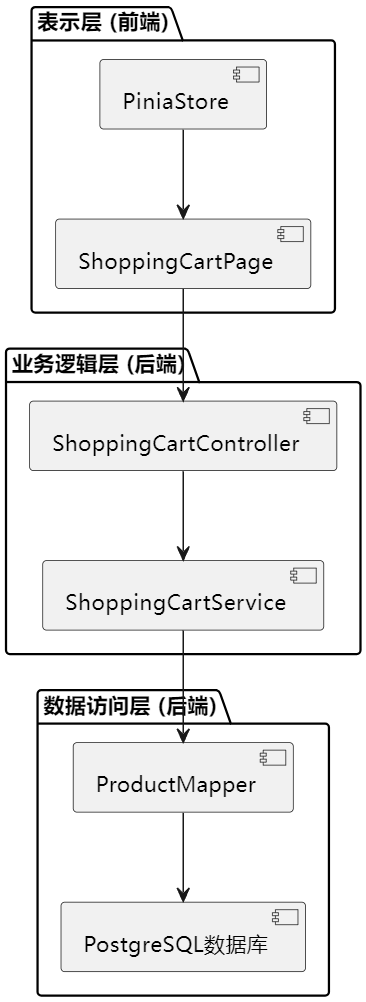
1. ShoppingCartController：负责操作购物车相关API
2. ShoppingCartService：为Contoller提供底层的业务逻辑服务



说明：

1. ShoppingCartPage：购物车的组件
2. ShoppingCartStore：提供操作前端购物车数据的接口

#### 整体概览



#### 补充

### 订单模块

#### 数据库设计

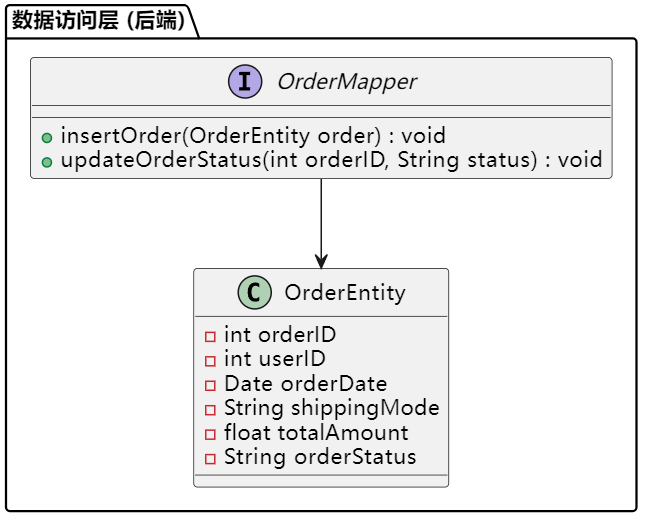
order\_history 表

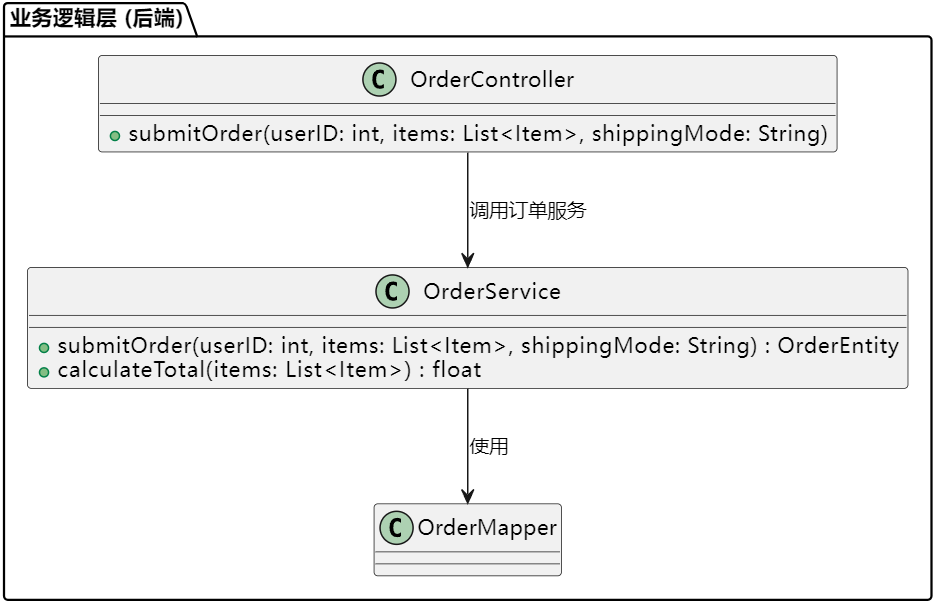
| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| order\_id | SERIAL | 主键，自增 |
| user\_id | INT | 用户ID，引用到 users 表的 user\_id 字段，删除时级联删除 |
| order\_date | TIMESTAMP | 订单日期，默认当前时间 |
| shipping\_mode | VARCHAR(50) | 配送方式 |
| total\_amount | DECIMAL(10,2) | 总金额 |
| order\_status | VARCHAR(20) | 订单状态 |

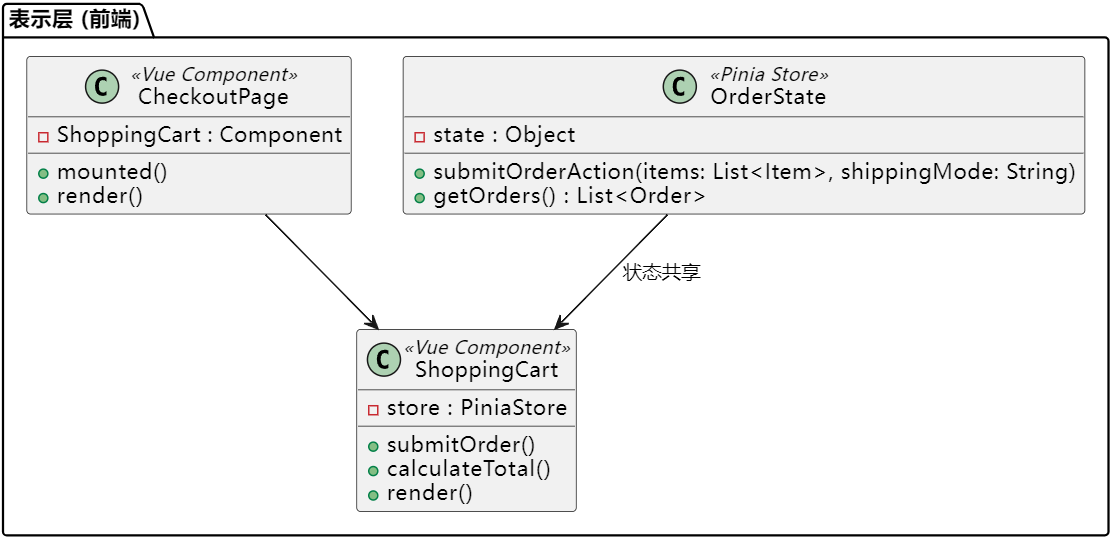
order\_detail 表

| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| detail\_id | SERIAL | 主键，自增 |
| order\_id | INT | 订单ID，引用到 order\_history 表的 order\_id 字段，删除时级联删除 |
| product\_id | INT | 产品ID，引用到 product 表的 product\_id 字段，删除时级联删除 |
| quantity | INT | 数量，不为空 |
| price\_at\_purchase | DECIMAL(10,2) | 购买时的单价 |

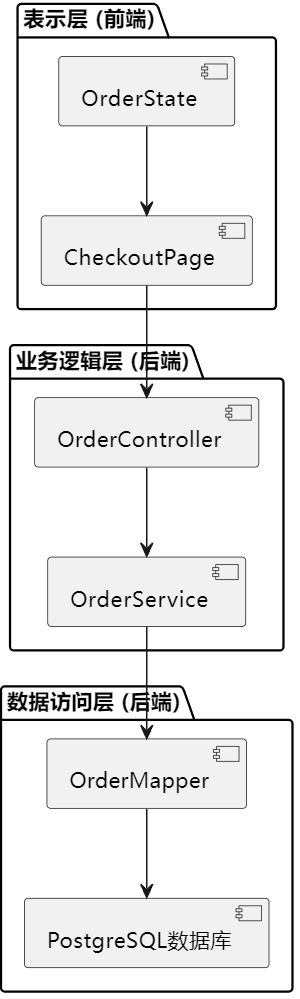
#### 详细类图



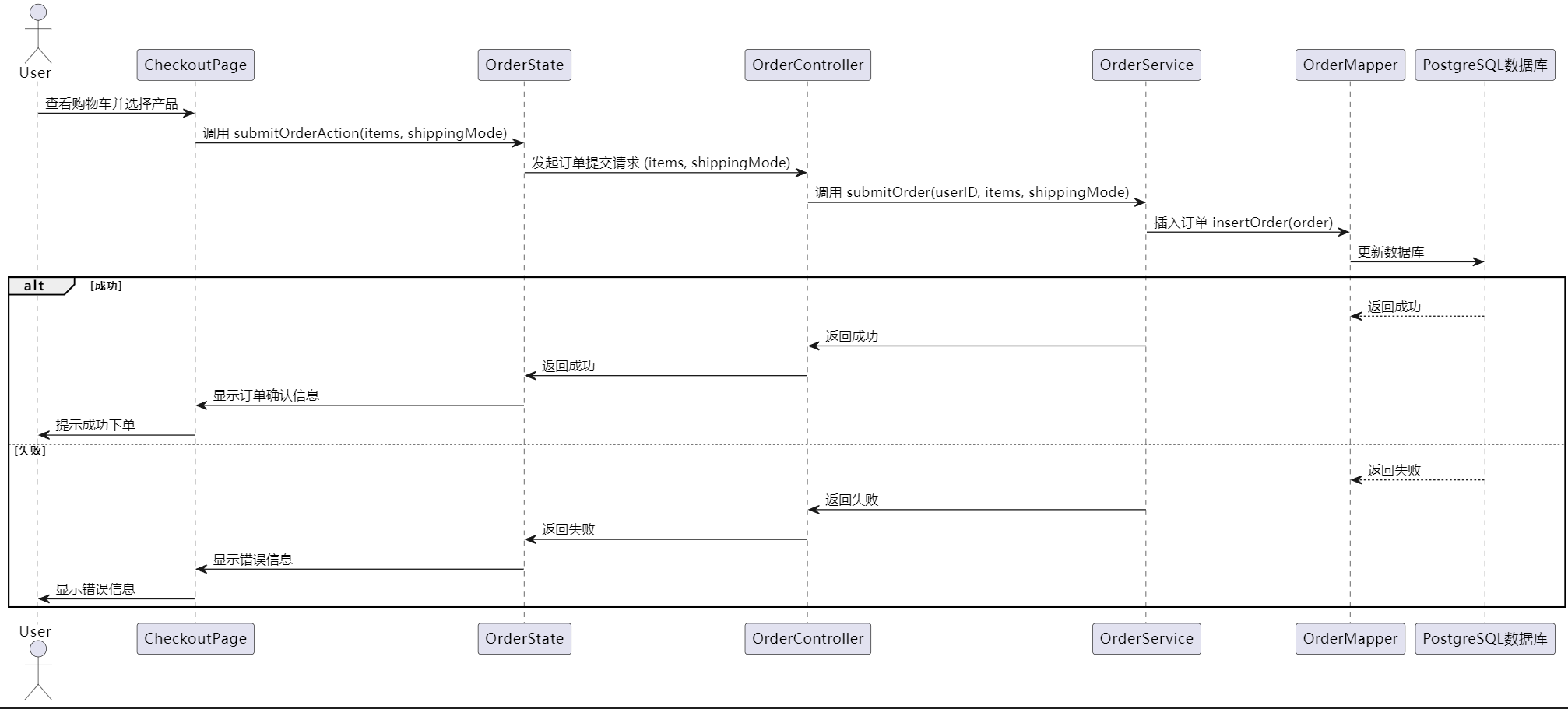




#### 整体概览



#### 补充



### 评论模块

#### 数据库设计

product\_review 表

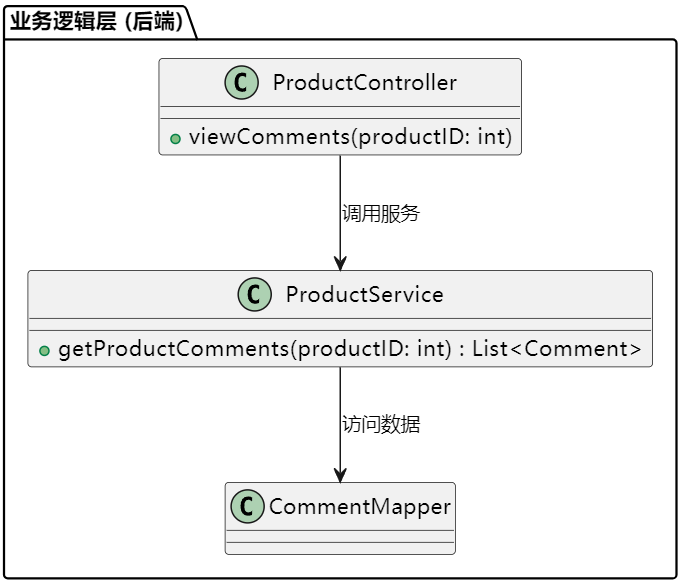
| **字段名** | **类型** | **描述** |
| --- | --- | --- |
| review\_id | SERIAL | 主键，自增 |
| product\_id | INT | 产品ID，引用到 product 表的 product\_id 字段，删除时级联删除 |
| user\_id | INT | 用户ID，引用到 users 表的 user\_id 字段，删除时级联删除 |
| review\_content | TEXT | 评价内容 |
| rating | DECIMAL(2,1) | 评分，范围为1.0到5.0，包含边界 |
| review\_date | TIMESTAMP | 评价日期，默认当前时间 |
| order\_id | INT | 订单ID，引用到 order\_history 表的 order\_id 字段，删除时置空 |

#### 详细类图



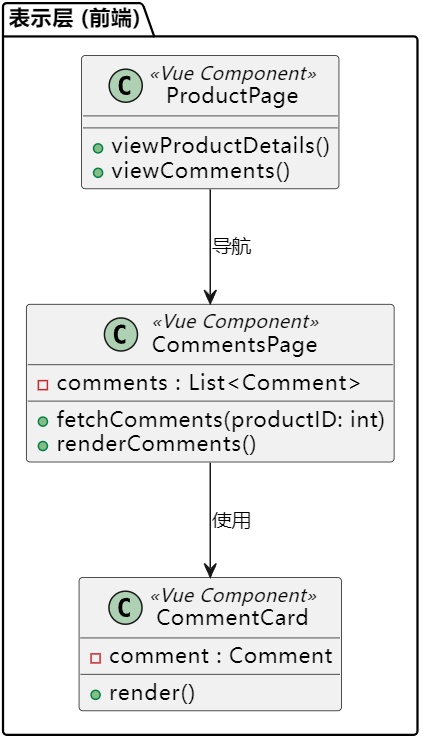
说明：

1. Comment：映射到数据库中的评论表，存储产品评论信息。
2. CommentMapper：提供评论表的基本读写操作。



说明：

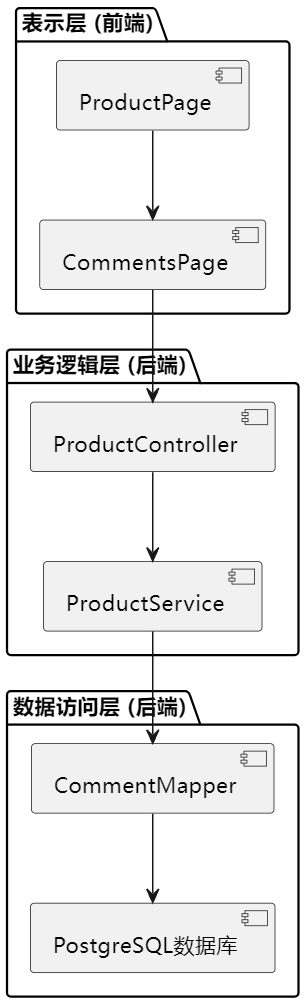
1. ProductService：包含获取产品评论的业务逻辑。
2. ProductController：处理客户端请求并调用ProductService。



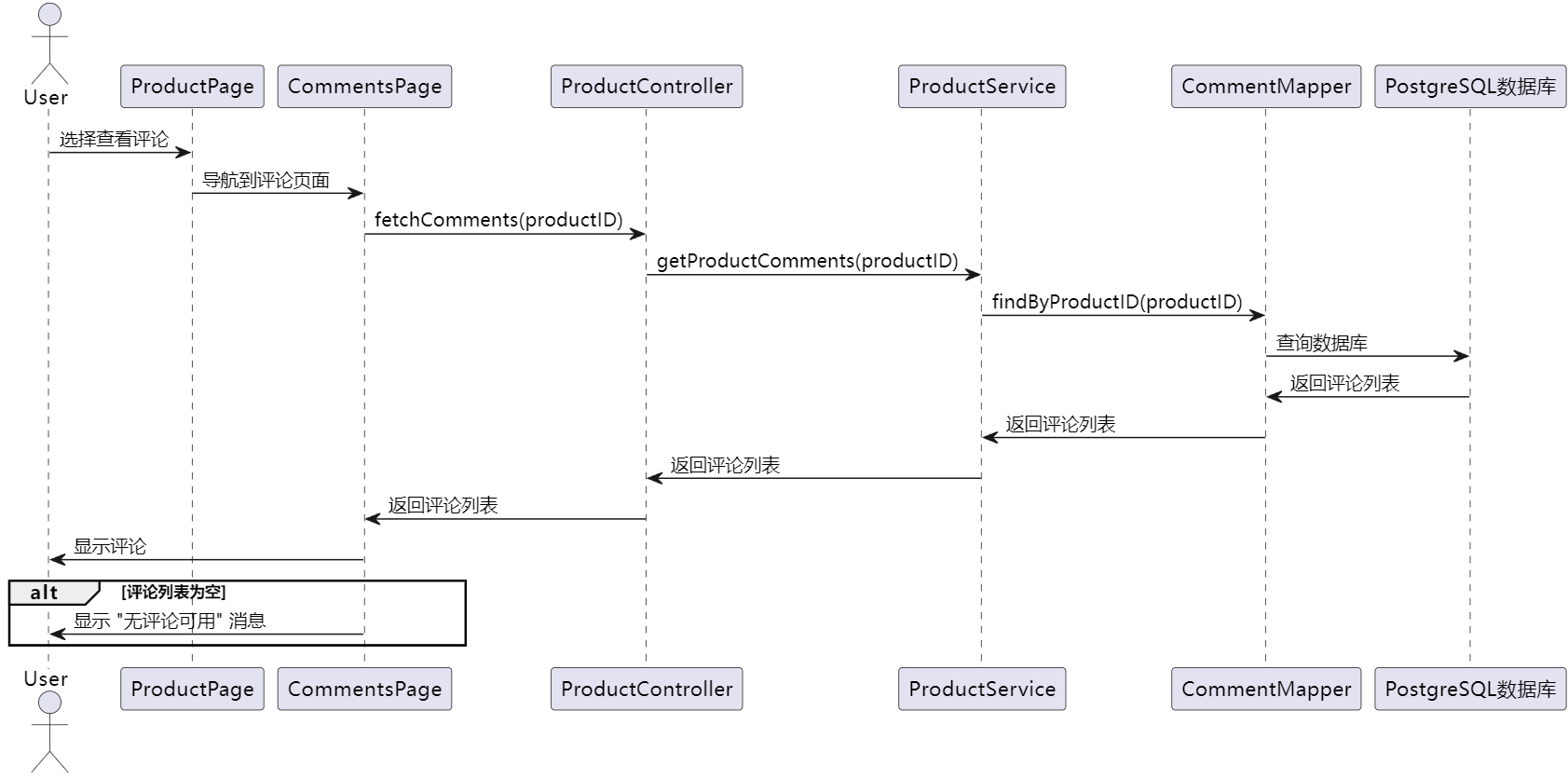
说明：

1. ProductPage：Vue 组件，负责展示产品详情并提供查看评论的入口。
2. CommentsPage：Vue 组件，负责获取和显示评论。
3. CommentCard：单条评论的展示组件。

#### 整体概览



#### 补充



# 测试计划

## 简介

本测试计划的目的是为WebOrder系统制定一个全面的测试方案。WebOrder系统是一种在线购物平台，用户可以浏览并购买乐器，同时查看评论和评估产品。本系统还依赖外部关系数据库（RDBMS）来存储产品信息、用户账户信息、订单历史和运费数据。

## 测试范围

测试范围涵盖WebOrder系统的各个用例和其相应的正常和异常情况，包括登录、浏览目录、添加商品、删除商品、清空购物篮、提交订单、查看订单历史、评价商品、查看评论和登出。

## 测试目的

通过对WebOrder系统的测试，确保以下几个方面：

1. 系统功能的正确性。
2. 用户交互的流畅性和友好性。
3. 数据库连接和数据读取的准确性。
4. 对异常情况的有效处理。

## 测试环境

操作系统：Windows

浏览器：Chrome

数据库：PostgreSQL

网络条件：高速互联网连接

## 角色定义

1. **主要用户（人类用户）**

浏览乐器产品目录 查看评论或评估产品 购买乐器 查看过去订单的历史和详情

1. **次要用户（外部系统）**

数据库系统（RDBMS）：存储产品信息、用户账户信息、订单历史和运费数据

## 用例描述

### 登录

**描述：** 登录并开始购物会话。

**正常场景：**

1. 用户进入WebOrder应用并进入登录界面。
2. 用户输入用户名和密码。
3. WebOrder请求数据库中与该用户名关联的密码。
4. WebOrder验证输入的密码与存储的密码是否一致。
5. WebOrder初始化一个空的购物篮。
6. WebOrder显示欢迎消息及产品列表。

异常场景：

1. 未知用户或密码错误：WebOrder提示“登录失败”。
2. 无法连接数据库：WebOrder提示“登录失败”。

### 浏览目录

**描述：** 浏览可供销售的乐器目录。

**正常场景：**

1. 用户登录后，显示产品列表。
2. 用户从列表中选择一个产品进行查看。
3. WebOrder请求数据库中该产品的信息。
4. WebOrder显示存储的信息（描述、价格、重量、图片等）。

**异常场景：**

1. 未知产品：WebOrder “显示为空”。
2. 无法连接数据库：WebOrder提示 “系统错误”。

### 添加商品

**描述：** 将商品添加到购物篮。

**正常场景：**

1. 用户从产品列表中选择一个商品并查看产品信息。
2. 用户输入购买数量，并点击“添加商品”按钮。
3. WebOrder请求数据库中该产品的运输重量和当前价格。
4. WebOrder将指定数量的商品添加到购物篮：
   * 如果购物篮中已有该商品，则增加数量。
   * 如果购物篮中没有该商品，则新增一个条目。
5. WebOrder更新总重量和总价格。
6. WebOrder更新购物篮内容显示。

**异常场景：**

1. 数量不合适：WebOrder 自动把数量控制在0以上。
2. 无法连接数据库：WebOrder提示“系统错误：无法连接数据库”。

### 删除商品

**描述：** 从购物篮中删除商品。

**正常场景：**

1. 用户选择购物篮中的商品。
2. 用户选中商品并点击“删除商品”按钮。
3. WebOrder从购物篮中删除相应条目。
4. WebOrder更新总重量和购买金额。
5. WebOrder更新购物篮内容显示。

**异常场景：**

1. 购物篮为空：WebOrder不显示“清空”，显示“购物篮中没有商品”。

### 清空购物篮

**描述：** 清空购物篮中的商品。

**正常场景：**

1. 用户选择购物篮中的商品。
2. 用户点击“清空购物篮”按钮。
3. WebOrder清空购物篮中的所有条目。
4. WebOrder将总重量和购买金额重置为零。
5. WebOrder更新购物篮内容显示。

**异常场景：**

1. 购物篮为空：WebOrder提示“购物篮已为空”。

### 提交订单

**描述：** 提交订单。

**正常场景：**

1. 用户选择购物篮中的商品。
2. WebOrder更新总成本和运费。
3. 用户选择运输方式（航空/陆运）并点击“提交订单”按钮。
4. WebOrder将订单历史信息更新到数据库。
5. 成功后，WebOrder清空购物篮。
6. WebOrder重置总重量和购买金额为零。
7. WebOrder更新购物篮内容显示。
8. WebOrder显示成功提交订单的信息。

**异常场景：**

1. 购物篮为空：WebOrder提示“购物篮为空”。
2. 无法连接数据库：WebOrder提示“系统错误：无法连接数据库”。

### 查看订单历史

**描述：** 查看订单历史。

**正常场景：**

1. 用户点击“订单历史”按钮。
2. WebOrder请求用户的订单历史信息。
3. 用户看到“订单历史”界面。
4. WebOrder显示用户提交订单的列表。

**用户查看订单列表：**

1. 用户点击“确定”按钮。
2. “订单历史”窗口关闭，用户返回主界面。

**用户查看订单详情：**

1. 用户从订单列表中选择一笔订单。
2. WebOrder显示该订单的详细信息（订单ID、日期、运输方式、总价格、商品名称及数量）。
3. 用户点击“确定”按钮。
4. “订单历史”窗口关闭，用户返回主界面。

**异常场景：**

1. 订单历史为空：显示空列表。
2. 无法连接数据库：WebOrder提示“系统错误”。

### 评估商品

**描述：** 评估已购买的产品。

**正常场景：**

1. 用户点击“订单历史”按钮。
2. WebOrder请求用户的订单历史信息。
3. 用户看到“订单历史”界面。
4. WebOrder显示用户提交订单的列表。
5. 用户从订单列表中选择一笔订单。
6. WebOrder显示该订单的详细信息。
7. 用户点击“评估”按钮。
8. 用户看到“订单评估”界面。
9. 用户输入评估信息，包括文本、图片和星级（1-5星）。
10. 用户点击“确定”按钮。
11. “订单评估”窗口关闭，用户返回主界面。

**异常场景：**

1. 用户已评估过该商品：提示“该商品已评估过”。 无法连接数据库：WebOrder提示“系统错误：无法连接数据库”。

### 查看评论

**描述：** 查看商品的评论。

正常场景：

1. 用户登录后，显示产品列表。
2. 用户从列表中选择一个产品进行查看。
3. WebOrder请求数据库中该产品的信息。
4. WebOrder显示存储的信息（描述、价格、重量、图片等）。
5. 用户在产品信息页面点击“查看评论”按钮。
6. WebOrder请求该产品的评论信息。
7. 用户看到“产品评论”界面。
8. WebOrder显示评论列表。
9. 用户点击“确定”按钮。
10. “产品评论”窗口关闭，用户返回主界面。

异常场景：

1. 无评论：显示空列表。
2. 无法连接数据库：WebOrder提示“系统错误”。

### 登出

**描述：** 登出并结束购物会话。

**正常场景：**

1. 用户点击“登出”按钮。
2. 如果有未完成的购买，WebOrder会清空购物篮。
3. WebOrder删除该用户的购物篮。
4. WebOrder显示登录界面，并指示用户已登出。

# 测试报告

## 登录模块测试报告

### 网络脱机

图形用户界面, 应用程序

描述已自动生成

### 输入错误用户名

图形用户界面, 应用程序

描述已自动生成

### 输入错误密码

图形用户界面, 应用程序

描述已自动生成

### 正确登录页面初始情况

（正确显示提示信息、产品列表、空的购物栏）

电脑萤幕的截图

描述已自动生成

## 购物车和结算模块

### 第一部分：添加商品的测试

#### 选择一个产品列表并查看产品信息

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

#### 输入要购买的商品数量，点击＋号

图形用户界面, 应用程序

描述已自动生成

#### 添加其他商品，更新购物车

图形用户界面, 应用程序

描述已自动生成

### 第二部分：删除购物车的商品

#### 如果删除单个商品，直接将购买数量改为0

图形用户界面, 文本, 应用程序

描述已自动生成

#### 删除之后，购物车更新

图形用户界面, 文本, 应用程序

描述已自动生成

### 第三部分：清空购物车

#### 点击清空按钮

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

### 第四部分：提交订单，进行结算

#### 点击购物车中的结账键

图形用户界面, 文本, 应用程序

描述已自动生成

#### 可以改变购买数量并且更新总费用

图形用户界面, 应用程序

描述已自动生成

#### 选择配送方式，然后点击确认键

图形用户界面, 应用程序

描述已自动生成

#### 更新购物车

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

#### 成功下单

图形用户界面, 应用程序

描述已自动生成

## 订单管理和展示模块

### 第一部分：用户浏览的测试

#### 用户登录后，显示产品列表

图形用户界面

描述已自动生成

#### 用户从列表中选择一个产品分类

图形用户界面, 网站

描述已自动生成

#### 用户从产品列表中选择一个产品

图形用户界面, 应用程序

描述已自动生成

#### 未知产品的查询

图形用户界面, 应用程序

描述已自动生成

#### 分类列表下的查询(查询处理正确)

图形用户界面, 应用程序, Word

描述已自动生成

图形用户界面, 应用程序, Word

描述已自动生成

### 第二部分：订单管理的测试

#### 用户点击“订单历史”按钮

电脑屏幕截图

描述已自动生成

#### 用户查看订单详情

图形用户界面

中度可信度描述已自动生成

## 评价模块

### 打开我的订单

电脑屏幕截图

描述已自动生成

### 选择一个订单

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

### 选择一个未评价的订单进行评价

图形用户界面, 应用程序

描述已自动生成

### 若不打星，只进行文字评价，显示评论保存失败

图形用户界面, 应用程序

描述已自动生成

文本

描述已自动生成

### 若只打星，不进行文字评价，显示评论保存成功

图形用户界面

描述已自动生成



手机屏幕截图

中度可信度描述已自动生成

### 查看评论：点进主页的商品，可浏览评论：

图形用户界面, 文本, 应用程序, 电子邮件, Teams

描述已自动生成

# 工作总结