

# 文档作者

- 主要编写者：李栋
- 其他编写者：李仁浩

# 文档修改历史

修改人员	日期	修改原因	版本号
李栋	2025-04-25	初稿	v0.1
李仁浩	2025-04-27	补充第五部分接口	v0.2
李栋	2025-05-01	完善开发包图	v0.3
李仁浩	2025-05-05	服务器部署，完善部署部分	v0.4
李栋	2025-05-19	补充自由需求申请商店的部分	v0.5
李栋	2025-05-25	补充自由需求优惠卷部分	v0.6
李栋	2025-06-19	完善部分图	v0.7
李仁浩	2025-06-20	完善物理部署	v0.8

# 目录

1. 引言
  - 1.1.1 编制目的
  - 1.1.2 词汇表
  - 1.1.3 参考资料
2. 产品概述
3. 逻辑视角
4. 组合视角
  - 1.4.1 开发包图
  - 1.4.2 运行时进程
  - 1.4.3 物理部署
5. 接口视角
  - 1.5.1 模块的职责
  - 1.5.2 用户界面层的分解
  - 1.5.3 业务逻辑层的分解
  - 1.5.4 数据层的分解
6. 信息视角

# 1. 引言

## 1.1 编制目的

本报告旨在对番茄商城系统进行概要设计，明确系统的架构与关键模块，为后续的详细设计、开发、测试和部署提供指导依据。通过本报告，开发人员可以了解系统的整体结构和各模块职责，测试人员可以依据模块划分制定测试计划，最终用户及其他相关方也可以通过本报告理解系统功能与结构布局。

通过体系结构设计，本项目期望达到以下目标：

- 规范番茄商城系统的架构设计，确保各模块协同工作，满足功能需求。
- 明确顾客与管理员两大核心角色的系统使用方式与交互流程。
- 支撑实体书本线上销售业务的快速开发与后续可持续扩展。

## 1.2 词汇表

词汇名称	词汇含义
PO (Persistent Object)	持久化对象，直接映射数据库表的实体类。
VO (View Object)	视图对象，封装返回前端的数据结构。
Controller	控制器层，接收HTTP请求并调用Service处理后返回响应。
Service	业务逻辑层，处理具体的业务逻辑。
Repository	数据访问层，封装对数据库的增删改查操作。
Res (Response)	响应对象，封装统一的接口返回格式。
Exception	异常处理模块，定义全局异常、业务异常等。
Enums	枚举类，定义系统中常量或状态。
Configure	配置模块，用于存放系统配置文件或初始化逻辑。

## 1.3 参考资料

- 项目启动文档
- 用例文档
- 项目需求规格说明书 v1.0

# 2. 产品概述

## 2.1 产品定位

番茄商城是番茄读书推出的线上实体书本购买平台，旨在满足广大读者对纸质书籍的购买需求。平台面向顾客和管理员两类用户，提供便捷、高效、安全的购书体验及后台管理支持。

## 2.2 核心功能

- 顾客功能
  - 用户注册与登录
  - 个人信息管理
  - 浏览书籍列表与详情
  - 搜索与筛选书籍
  - 下单购买、在线支付
  - 订单管理与查询
- 管理员功能
  - 书籍信息管理（新增、编辑、删除书籍）
  - 库存管理
  - 订单处理与状态管理
  - 支付记录查询与处理

## 2.3 用户角色

- 顾客：注册后可浏览商城书籍，进行购买、支付、查看订单等操作。
- 管理员：负责商城数据管理，包括书籍信息维护、库存更新、订单处理等。

## 2.4 系统特点

- 线上商城一体化：实现从浏览、购买到支付的完整闭环。
- 支持角色权限管理：区分顾客与管理员功能。
- 良好的用户体验：界面友好，交互流畅。
- 可扩展性设计：支持后续扩展更多品类或营销活动功能。
- 高可靠性：采用分层架构设计，保证系统稳定运行与易于维护。

# 3. 逻辑视角

## 3.1 分层体系结构逻辑视角图

番茄书城管理系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。

层级	名称	功能
展示层（Presentation Layer）	Controller 层	处理 HTTP 请求，接收参数、调用业务逻辑、返回响应
业务逻辑层（Business Logic Layer）	Service 层	编写核心业务逻辑，如注册、登录、计算、流程控制等

层级	名称	功能
数据层 (Data Access Layer)	Repository / DAO 层	与数据库交互，增删改查，持久化数据

## 3.2 分层体系结构逻辑设计方案图

# 4. 组合视角

## 4.1 开发包图

番茄商城系统基于分层架构，按照功能模块进行开发包 (Package) 划分。各开发包之间遵循高内聚、低耦合的设计原则，严格按照层次结构进行依赖。开发包如下。

开发 (物理) 包	依赖的其他开发包
ProductController	ProductService, ProductVO, Response, StockpileVO, Spring Web
AccountController	AccountService, TokenUtil, AccountVO, Response, Spring Web
OrderController	OrderService, ProductService, OrderVO, Response, 支付宝 SDK, Spring Web
CartController	CartService, CartVO, CartListVO, Response, Spring Web
ShopController	ShopService, ShopVO, Response, Spring Web
ReviewController	ReviewService, ReviewVO, Response, Spring Web
MessageController	MessageService, MessageVO, Response, Spring Web
AdvertisementController	AdvertisementService, AdvertisementVO, Response, Spring Web
CouponController	CouponService
ImageController	Spring Web
ProductService	ProductVO, StockpileVO
CouponService	CouponVO, AccountVO
AccountService	AccountVO, MultipartFile
OrderService	OrderVO, 支付宝SDK, HttpServletRequest, HttpServletResponse
CartService	CartVO, CartListVO, CartRequestDTO
ShopService	ShopVO
ReviewService	ReviewVO
MessageService	MessageVO

开发(物理)包	依赖的其他开发包
AdvertisementService	AdvertisementVO
ProductServiceImpl	ProductService, ProductRepository, ProductPO, ProductVO, StockpileVO, StockpileRepository
AccountServiceImpl	AccountService, AccountRepository, AccountPO, AccountVO, TokenUtil, SecurityUtil, OssUtil, ImageUtil
OrderServiceImpl	OrderService, OrderRepository, OrderPO, OrderVO, ProductService, 支付宝SDK
CartServiceImpl	CartService, CartRepository, CartPO, CartVO, ProductService, OrderService
ShopServiceImpl	ShopService, ShopRepository, ShopPO, ShopVO
ReviewServiceImpl	ReviewService, ReviewRepository, ReviewPO, ReviewVO
MessageServiceImpl	MessageService, MessageRepository, MessagePO, MessageVO
AdvertisementServiceImpl	AdvertisementService, AdvertisementRepository, AdvertisementPO, AdvertisementVO
ProductRepository	ProductPO, Spring Data JPA
AccountRepository	AccountPO, RoleEnum, Spring Data JPA
OrderRepository	OrderPO, Spring Data JPA
CartRepository	CartPO, Spring Data JPA
ShopRepository	ShopPO, Spring Data JPA
ReviewRepository	ReviewPO, Spring Data JPA
MessageRepository	MessagePO, Spring Data JPA
AdvertisementRepository	AdvertisementPO, Spring Data JPA
CartOrderRelationRepository	CartOrderRelationPO, Spring Data JPA
SpecificationRepository	SpecificationPO, Spring Data JPA
StockpileRepository	StockpilePO, Spring Data JPA
ProductPO	无
AccountPO	RoleEnum
OrderPO	PaymentStatusEnum
CartPO	无
ShopPO	无
ReviewPO	无

开发(物理)包	依赖的其他开发包
MessagePO	无
AdvertisementPO	无
CartOrderRelationPO	无
SpecificationPO	无
StockpilePO	无
ProductVO	无
AccountVO	无
OrderVO	无
CartVO	无
CartListVO	无
CartRequestDTO	无
ShopVO	无
ReviewVO	无
MessageVO	无
AdvertisementVO	无
CartOrderRelationVO	无
SpecificationVO	无
StockpileVO	无
Response	无
CheckRequestVO	无
OrderInfoVO	无
TokenUtil	JWT, SecurityUtil
SecurityUtil	Spring Security
OssUtil	阿里云OSS SDK
ImageUtil	无
ToolUtil	无
SecurityConfig	Spring Security
MyWebMvcConfig	Spring Web MVC
CorsFilter	Spring Web
LoginInterceptor	Spring Web, TokenUtil

开发(物理)包	依赖的其他开发包
PaymentStatusEnum	无
RoleEnum	无
GlobalExceptionHandler	Spring Web, Response
TomatoMallException	无

#### 外部依赖说明

- **Spring Boot Starter Web** - Web应用框架
- **Spring Boot Starter Data JPA** - 数据持久化框架
- **Spring Boot Starter Security** - 安全框架
- **MySQL Connector** - MySQL数据库驱动
- **Java JWT** - JWT令牌处理库
- **阿里云OSS SDK** - 阿里云对象存储服务
- **支付宝SDK** - 支付宝支付功能
- **Lombok** - 代码简化工具
- **FastJSON** - JSON处理库

## 4.2 运行时进程

番茄书城会有多个客户端进程和一个服务器端进程，其进程图如下结合部署图，客户端进程是在客户端机器上运行，服务器端进程在服务器端机器上运行。

### 4.2.2 进程说明

1. **客户端 (Web浏览器)**：客户端通过浏览器向系统发起请求，通常是通过HTTP协议请求后端服务，客户端可基于REST API与后端交互。
2. **网关/负载均衡层**：负责接收来自客户端的请求并将其路由到合适的后端微服务或者应用程序实例。在分布式环境中，负载均衡层用于确保流量的平衡分配，提升系统的可靠性与可扩展性。
3. **控制层 (Controller)**：控制层处理前端请求，将请求数据解析为相应的视图对象 (VO)，并调用相应的服务层进行业务处理。控制层是系统的入口，负责所有外部请求的接收和响应返回。
4. **业务逻辑层 (Service)**：业务逻辑层负责处理系统中的业务规则，协调各个模块之间的操作。服务层在执行操作时可能会调用多个数据访问层的接口，完成对数据库的增删改查 (CRUD) 操作。

5. **数据访问层（Repository）**：数据访问层直接与数据库进行交互，执行数据存取操作。它提供了一个抽象的接口来处理所有的数据库操作，确保业务层和数据库之间的解耦。
6. **数据库（MySQL）**：数据库用于存储持久化的数据，所有用户数据、书籍信息、订单记录等均保存在数据库中。

## 4.3 物理部署

### 1. 阿里云服务器（后端部署）

- Spring Boot 后端服务（REST API）
- 数据库服务（如 MySQL）
- Nginx（用于反向代理与负载均衡）

### 2. 客户端设备（前端访问）

- 用户通过浏览器或客户端程序访问系统
- 前端通过 HTTP/HTTPS 调用后端 API，与服务器通信

客户端和服务端通过公网网络进行连接，所有数据请求统一通过服务器暴露的 API 接口进行处理。

## 5. 接口视角

### 5.1 模块的职责

客户端各层职责

层	职责
启动模块	负责初始化网络通信机制，启动用户界面
用户界面层	展示客户端窗口网页
业务逻辑层	响应用户界面层的操作并处理业务逻辑
网络通信层	与远程服务器端交互并请求服务

服务器端职责

层	职责
启动模块	负责初始化网络通信机制，启用服务
数据层	负责数据持久化及提供数据访问接口
业务逻辑层	负责业务逻辑处理
控制层	负责接收客户端请求
网络通信层	负责提供服务

层之间调用的接口

接口	服务调用方	服务提供方
account advertisement cart order product	客户端用户界面层	客户端业务逻辑层
AccountController AdvertisementController CartController OrderController ProductController	客户端业务逻辑层	服务端控制层
AccountService AdvertisementService CartService OrderService ProductService	服务端控制层	服务端业务逻辑层
AccountRepository AdvertisementRepository CartRepository OrderRepository ProductRepository	服务端业务逻辑层	服务端数据层

以账号用例来说明层之间的调用，如下所示。层与层之间由接口交互，大大降低了层与层之间的耦合。

## 5.2 用户界面层的分解

根据需求，系统存在如下15个用户界面：商城主页面，登录界面，注册界面，个人主界面，管理员界面，商品详情界面，广告详情界面，购物车界面，管理购物车界面，结账界面，修改信息界面，管理商品界面，管理广告界面，添加商品界面，参与活动界面。

### 5.2.1 职责

模块	职责
MainFrame	界面Frame，负责界面的显示和界面的跳转

### 5.2.2 接口规范

用户界面层模块的接口

模块	语法	前置条件	后置条件
MainFrame	router.push(page:String)	无	显示Frame，跳转到指定页面

用户界面模块需要的服务接口

服务名	服务
account	负责处理账号相关业务逻辑
advertisement	负责处理广告相关业务逻辑
cart	负责处理购物车相关业务逻辑
order	负责处理订单相关业务逻辑
product	负责处理商品相关业务逻辑

## 5.3 业务逻辑层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，Account对象负责处理登录界面、注册界面的业务逻辑；Product对象负责处理商品详情界面的业务逻辑。业务逻辑层的设计如下图所示。

### 5.3.1 职责

模块	职责
accountbl	负责账号登录、账号注册、账号管理所需要的服务
advertisementbl	负责广告详情界面所需要的服务
cartbl	负责购物车界面所需要的服务
orderbl	负责处理订单所需要的服务
productbl	负责商品详情界面所需要的服务

### 5.3.2 接口规范

#### account模块

提供的服务（供接口）	语法	前置条件	后置条件
account.getUserDetails	/GET/account/{username}	存在用户名	返回对应账号信息
account.createAccount	/POST/account/{accountVO}	不存在相同用户名	数据库中增加一个新的账户信息
account.login	/POST/account/login/{username, password}	无	查找数据库中相应的账号，返回登录验证结果及对话TOKEN

提供的服务 (供接口)	语法	前置条件	后置条件
account.updateUser	/PUT/account/{accountVO}	处于登录状态	更新数据库中相应的账户信息

需要的服务 (需接口)	服务
AccountController.getAccountInfo	获取数据库中的账号信息
AccountController.createAccount	在数据库中增加一个新的账户信息
AccountController.login	验证登录信息
AccountController.updateAccount	更新账号信息

## advertisement模块

提供的服务 (供接口)	语法	前置条件	后置条件
advertisement.getAdvertisements	/GET/advertisements	无	返回所有广告信息
advertisement.updateAdvertisement	/put/advertisements/{advertisementVO}	存在id相同的广告	更新id相同的广告信息

提供的服务 (供接口)	语法	前置条件	后置条件
advertisement.addAdvertisement	/POST/advertisements/{advertisementVO}	无	数据库中增加一条新的广告信息
advertisement.deleteAdvertisement	/delete/advertisements/{advertisementId}	验证管理员身份且存在id相同的广告	删除数据库中相应的广告信息

需要的服务 (需接口)	服务
AdvertisementController.getAdvertisements	获取数据库中所有的广告信息
AdvertisementController.createAdvertisement	在数据库中增加一条新的广告信息
AdvertisementController.updateAdvertisement	更新数据库中的广告信息
AdvertisementController.deleteAdvertisement	删除数据库中相应的广告信息

## cart模块

提供的服务 (供接口)	语法	前置条件	后置条件
cart.addCartProduct	/POST/cart/{productId, quantity}	存在相同 productId的商品	数据库中增加一条购物车商品信息

提供的服务 (供接口)	语法	前置条件	后置条件
cart.deleteCartProduct	/DELETE/cart/{cartItemId}	存在相同 cartItemId 的购物车商品	数据库中删除相同 cartItemId 的购物车商品信息
cart.updateCartItem	/patch/cart/{cartItemId, quantity}	存在相同 cartItemId 的购物车商品, quantity 不能超过库存	更新数据库中相同 cartItemId 的购物车商品信息
cart.getCartProducts	/GET/cart	处于登录状态	返回所有的购物车商品信息

需要的服务 (需接口)	服务
CartController.addCart	在数据库中增加一个购物车商品信息
CartController.deleteCart	在数据库中删除相应的购物车商品信息
CartController.updateCart	更新数据库中的购物车商品信息
CartController.getCart	获取所有的购物车信息

## order模块

提供的服务 (供接口)	语法	前置条件	后置条件
order.checkout	/POST/cart/checkout/{cartItemIds, orderInfo, paymentMethod}	选中至少一个购物车商品	数据库中增加一个新的订单信息并返回
order.payOrder	/POST/order/{orderId}/pay	存在相同 orderId 的订单信息	跳转至支付页面

需要的服务 (需接口)	服务
CartController.checkout	在数据库中增加一个订单信息
OrderController.requestPayment	处理订单并返回支付页面
OrderController.handleAlipayNotify	响应支付回调，并更新订单状态

## product模块

提供的服务 (供接口)	语法	前置条件	后置条件
product.getProductsList	/GET/products	无	无
product.getProduct	/GET/products/{productId}	存在相同 productId 的商品	跳转至商品详情界面
product.updateProduct	/PUT/product/{productVO}	验证管理员身份且存在相同 productId 的商品	更新数据库中相同 productId 的商品信息
product.addProduct	/POST/product/{productVO}	验证管理员身份	数据库中增加一个新的商品信息
product.deleteProduct	/DELETE/product/{productId}	验证管理员身份且存在相同 productId 的商品	删除数据库中相同 productId 的商品信息
product.updateStockpile	/PATCH/product/stockpile/{productId}/{amount}	验证管理员身份且存在相同 productId 的商品	更新数据库中相同 productId 的商品库存信息
product.getStockpile	/GET/product/stockpile/{productId}	无	无

需要的服务 (需接口)	服务
ProductController.getAllProduct	获取数据库中所有的商品信息
ProductController.getProductById	获取数据库中指定productId的商品信息
ProductController.addProduct	在数据库中增加一个新的商品信息
ProductController.updateProduct	更新数据库中的商品信息
ProductController.deleteProduct	删除数据库中的商品信息
ProductController.getProductStock	更新数据库中的商品库存信息
ProductController.updateProductStock	获取数据中指定商品的库存信息

## coupon模块

提供的服务 (供接口)	语法	前置条件	后置条件
coupon.getAllCoupons	/GET/coupons/all	无	返回所有优惠券信息

提供的服务 (供接口)	语法	前置条件	后置条件
coupon.getCouponById	/GET/coupons/{id}	存在相同id的优惠券	返回指定优惠券信息
coupon.createCoupon	/POST/coupons/{couponVO}	验证管理员身份	数据库中增加一条新的优惠券信息
coupon.updateCoupon	/PUT/coupons/{id}/{couponVO}	验证管理员身份且存在相同id的优惠券	更新数据库中相同id的优惠券信息
coupon.deleteCoupon	/DELETE/coupons/{id}	验证管理员身份且存在相同id的优惠券	删除数据库中相同id的优惠券信息
coupon.useCoupon	/POST/coupons/use/{accountCouponsRelationVO}	用户拥有该优惠券且未使用	标记优惠券为已使用状态
coupon.getCouponsByAccountId	/GET/coupons/account/{accountId}	存在相同accountId的用户	返回用户拥有的所有优惠券信息
coupon.userReceiveCoupon	/POST/coupons/receive/{accountCouponsRelationVO}	优惠券可领取且用户未拥有	数据库中增加用户优惠券关联信息

需要的服务 (需接口)	服务
CouponController.getAllCoupons	获取数据库中所有的优惠券信息

需要的服务 (需接口)	服务
CouponController.getCouponById	获取数据库中指定id的优惠券信息
CouponController.createCoupon	在数据库中增加一条新的优惠券信息
CouponController.updateCoupon	更新数据库中的优惠券信息
CouponController.deleteCoupon	删除数据库中的优惠券信息
CouponController.useCoupon	更新用户优惠券使用状态
CouponController.getCouponsForAccount	获取用户拥有的优惠券信息
CouponController.userReceiveCoupon	创建用户优惠券关联信息

## review模块

提供的服务 (供接口)	语法	前置条件	后置条件
review.addProductReview	/POST/reviews/product/{reviewVO}	用户已购买该商品	数据库中增加一条商品评论信息
review.addShopReview	/POST/reviews/shop/{reviewVO}	用户已在该商店消费	数据库中增加一条商店评论信息
review.getProductReviews	/GET/reviews/product/{productId}	存在相同 productId 的商品	返回指定商品的所有评论信息
review.getShopReviews	/GET/reviews/shop/{shopId}	存在相同 shopId 的商店	返回指定商店的所有评论信息
review.deleteReview	/DELETE/reviews/{reviewId}	用户是评论作者或管理员身份	删除数据库中相同 reviewId 的评论信息

需要的服务 (需接口)	服务
ReviewController.addProductReview	在数据库中增加一条商品评论信息
ReviewController.addShopReview	在数据库中增加一条商店评论信息
ReviewController.getProductReviews	获取指定商品的所有评论信息
ReviewController.getShopReviews	获取指定商店的所有评论信息
ReviewController.deleteReview	删除数据库中的评论信息

## shop模块

提供的服务（供接口）	语法	前置条件	后置条件
shop.getAllShops	/GET/shop/all	无	返回所有商店信息
shop.createShop	/POST/shop/create/{shopVO}	验证管理员身份	数据库中增加一条新的商店信息
shop.getShopDetail	/GET/shop/detail/{shopId}	存在相同shopId的商店	返回指定商店的详细信息
shop.updateShop	/PUT/shop/update/{shopId}/{shopVO}	验证管理员身份且存在相同shopId的商店	更新数据库中相同shopId的商店信息
shop.deleteShop	/DELETE/shop/delete/{shopId}	验证管理员身份且存在相同shopId的商店	删除数据库中相同shopId的商店信息
shop.getOwnShop	/GET/shop/owner/{ownerId}	存在相同ownerId的用户	返回用户拥有的商店ID

需要的服务（需接口）	服务
ShopController.getAllShops	获取数据库中所有的商店信息
ShopController.createShop	在数据库中增加一条新的商店信息
ShopController.getShopById	获取数据库中指定shopId的商店信息
ShopController.updateShop	更新数据库中的商店信息
ShopController.deleteShop	删除数据库中的商店信息
ShopController.getOwnShopId	获取用户拥有的商店ID

## image模块

提供的服务（供接口）	语法	前置条件	后置条件
image.saveImg	/POST/image/{file}	上传的文件格式正确且大小合理	返回图片存储路径

需要的服务（需接口）	服务
ImageController.saveImg	保存上传的图片文件并返回存储路径

## message模块

提供的服务（供接口）	语法	前置条件	后置条件
message.sendMessage	/POST/message/send/{messageVO}	发送者和接收者都存在	数据库中增加一条新的消息信息
message.getReceivedMessages	/GET/message/received-list/{toUserId}	存在相同 toUserId 的用户	返回用户接收到的所有消息信息
message.getSentMessages	/GET/message/sent-list/{fromUserId}	存在相同 fromUserId 的用户	返回用户发送的所有消息信息
message.markMessageAsRead	/PUT/message/mark-read/{messageld}	存在相同 messageld 的消息	更新消息为已读状态
message.deleteMessage	/DELETE/message/delete/{messageld}	用户是消息发送者或管理员身份	删除数据库中相同 messageld 的消息信息
message.getUnreadMessageCount	/GET/message/unread-count/{userId}	存在相同 userId 的用户	返回用户未读消息数量
message.queryMessageCount	/GET/message/query/{fromUserId, content}	存在相同 fromUserId 的用户	返回指定内容的消息数量

需要的服务（需接口）	服务
MessageController.sendMessage	在数据库中增加一条新的消息信息
MessageController.getMessagesByToUserId	获取用户接收到的消息信息
MessageController.getMessagesByFromUserId	获取用户发送的消息信息
MessageController.markMessageAsRead	更新消息为已读状态
MessageController.deleteMessage	删除数据库中的消息信息
MessageController.getUnreadMessageCount	获取用户未读消息数量
MessageController.queryMessageCount	查询指定内容的消息数量

## 5.4 数据层的分解

数据层主要给业务逻辑层提供数据访问服务和实现数据持久化，包括对数据增、删、改、查的操作。Account业务逻辑需要的服务由AccountService接口提供。数据层模块的描述如下图所示。

## 5.4.1 职责

模块	职责
AccountServiceImpl	实现持久化数据库接口，提供增、删、改、查服务
AccountRepository	基于Mysql数据库，实现对数据的增、删、改、查操作

## 5.4.2 接口规范

提供的服务	语法	前置条件	后置条件
AccountServiceImpl.getAccountByUsername	public AccountVO getAccountByUsername(String username)	数据库中存在相同用户名的数据	在数据库中更新该用户信息
AccountServiceImpl.createAccount	public String createAccount(AccountVO accountVO)	在数据库中不存在相同的用户名	数据库中增加一个用户信息
AccountService.login	public String login(AccountVO accountVO)	无	根据用户名和密码查询数据库并返回登录状态
AccountService.getAccountInfo	public AccountVO getAccountInfo(String username)	处于登录状态	根据username进行查找并返回相应的结果

提供的服务	语法	前置条件	后置条件
AccountService.updateAccount	public String updateAccount(AccountVO accountVO)	处于登录状态	更新数据库中的账户信息
AccountRepository.findByUsername	public Account findByUsername(String username)	数据库中存在相同用户名的数据	查找数据库中相同用户名的账号信息
AccountRepository.save	public void save(AccountPO accountPO)	无	数据库中增、删、改响应的账号账号信息

## 6. 信息视角

### 6.1 描述数据持久化对象 (PO)

#### 1. 用户表 (accounts)

```

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Account {
    @Id
    private Long id;
    private String username;
    private String password;
    private String name;
    private String avatar;
    private String role;
    private String telephone;
    private String email;
    private String location;

    // Getters and Setters
}

```

```
}
```

## 2. 订单表 (orders)

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Order {
    @Id
    private Long orderId;
    @ManyToOne
    private Account account;
    private Double totalAmount;
    private String paymentMethod;
    private String status;
    private String createTime;

    // Getters and Setters
}
```

## 3. 商品表 (products)

```
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Product {
    @Id
    private Long id;
    private String title;
    private Double price;
    private Double rate;
    private String description;
    private String cover;
    private String detail;

    // Getters and Setters
}
```

## 4. 广告表 (advertisements)

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Advertisement {
    @Id
    private Long id;
    private String title;
    private String content;
```

```
    private String imageUrl;
    @ManyToOne
    private Product product;

    // Getters and Setters
}
```

## 5. 购物车商品表 (carts)

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Cart {
    @Id
    private Long cartItemId;
    @ManyToOne
    private Account account;
    @ManyToOne
    private Product product;
    private Integer quantity;

    // Getters and Setters
}
```

## 6. 购物车商品与订单关联表 (carts\_orders\_relation)

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class CartOrderRelation {
    @Id
    private Long cartsOrdersRelationId;
    @ManyToOne
    private Cart cart;
    @ManyToOne
    private Order order;

    // Getters and Setters
}
```

## 7. 商品规格表 (specifications)

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Specification {
    @Id
```

```

private Long specificationId;
private String item;
private String value;
@ManyToOne
private Product product;

// Getters and Setters
}

```

## 8. 商品库存表 (stockpiles)

```

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Stockpile {
    @Id
    private Long stockpileId;
    @ManyToOne
    private Product product;
    private Integer amount;
    private Integer frozen;

    // Getters and Setters
}

```

## 9. 优惠券表 (coupon)

```

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import java.time.LocalDateTime;

@Entity
@Table(name = "coupon")
public class Coupon {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "description")
    private String description;

    @Column(name = "discount_type", nullable = false)
    private Integer discountType; // 1: 百分比折扣, 2: 固定金额折扣

    @Column(name = "discount_value", nullable = false)

```

```

private Double discountValue;

@Column(name = "start_time", nullable = false)
private LocalDateTime startTime;

@Column(name = "end_time", nullable = false)
private LocalDateTime endTime;

@Column(name = "quantity", nullable = false)
private Integer quantity;

@Column(name = "used_quantity", nullable = false)
private Integer usedQuantity = 0;

@Column(name = "is_valid", nullable = false)
private Integer isValid = 1; // 1: 可用, 0: 不可用

// Getters and Setters
}

```

## 10. 用户优惠券关联表 (account\_coupons\_relation)

```

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

@Entity
@Table(name = "account_coupons_relation")
public class AccountCouponsRelation {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "account_id", nullable = false)
    private Integer accountId;

    @Column(name = "coupon_id", nullable = false)
    private Integer couponId;

    @Column(name = "quantity", nullable = false)
    private Integer quantity;

    // Getters and Setters
}

```

## 11. 订单商品表 (order\_items)

```

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;

```

```

import javax.persistence.GenerationType;

@Entity
@Table(name = "order_items")
public class OrderItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "order_item_id", nullable = false)
    private Integer orderItemId;

    @Column(name = "order_id", nullable = false)
    private Integer orderId;

    @Column(name = "product_id", nullable = false)
    private Integer productId;

    @Column(name = "quantity", nullable = false)
    private Integer quantity;

    @Column(name = "price", nullable = false)
    private double price;

    // Getters and Setters
}

```

## 12. 评论表 (reviews)

```

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Enumerated;
import javax.persistence.EnumType;
import java.math.BigDecimal;
import java.time.LocalDateTime;

@Entity
@Table(name = "reviews")
public class Review {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(name = "account_id", nullable = false)
    private Integer accountId;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String content;

    @Column(nullable = false, precision = 3, scale = 2)
    private BigDecimal rate;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)

```

```

private ReviewType type; // PRODUCT, SHOP

@Column(name = "product_id")
private Integer productId;

@Column(name = "shop_id")
private Integer shopId;

@Column(name = "created_at", insertable = false, updatable = false)
private LocalDateTime createdAt;

public enum ReviewType {
    PRODUCT,
    SHOP
}

// Getters and Setters
}

```

## 13. 商店表 (shops)

```

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;

@Entity
@Table(name = "shops")
public class Shop {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;

    @Column(name = "name", nullable = false, length = 100)
    private String name;

    @Column(name = "owner_id", nullable = false)
    private Integer ownerId;

    @Column(name = "icon_url", length = 255)
    private String iconurl;

    @Column(name = "description", length = 255)
    private String description;

    @Column(name = "rate", precision = 3, scale = 2)
    private Double rate;

    @Column(name = "is_valid", nullable = false)
    private Integer isValid = 0; // 1: 有效, 0: 无效

    // Getters and Setters
}

```

```
}
```

## 14. 消息表 (messages)

```
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import java.time.LocalDateTime;

@Entity
@Table(name = "messages")
public class Message {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column(nullable = false)
    private String content;

    @Column(nullable = false)
    private Boolean isRead = false;

    @Column(nullable = false)
    private Integer fromUser;

    @Column(nullable = false)
    private Integer toUser;

    @Column(nullable = false)
    private LocalDateTime createdTime = LocalDateTime.now();

    // Getters and Setters
}
```

\*\*1

## 6.2 数据库表

### 1. 用户表 (accounts)

表名	字段	类型	说明
accounts	id	INT	用户ID
accounts	username	VARCHAR(50)	用户名
accounts	password	VARCHAR(100)	密码
accounts	name	VARCHAR(50)	用户姓名
accounts	avatar	VARCHAR(255)	用户头像链接

表名	字段	类型	说明
accounts	role	VARCHAR(50)	用户身份
accounts	telephone	VARCHAR(11)	用户手机号, 格式需符合1开头的11位数字
accounts	email	VARCHAR(100)	用户邮箱, 格式需符合邮箱规范
accounts	location	VARCHAR(255)	用户所在地

## 2. 订单表 (orders)

表名	字段	类型	说明
orders	order_id	INT	订单ID
orders	account_id	INT	用户ID
orders	total_amount	DECIMAL(10, 2)	订单总金额
orders	payment_method	VARCHAR(50)	支付方式
orders	status	VARCHAR(20)	订单支付状态 (PENDING, SUCCESS, FAILED, TIMEOUT)
orders	create_time	TIMESTAMP	订单创建时间

## 3. 商品表 (products)

表名	字段	类型	说明
products	id	INT	商品ID
products	title	VARCHAR(50)	商品名称
products	price	DECIMAL(10, 2)	商品价格
products	rate	DOUBLE	商品评分
products	description	VARCHAR(255)	商品描述
products	cover	VARCHAR(500)	商品封面URL
products	detail	VARCHAR(500)	商品详细说明

## 4. 广告表 (advertisements)

表名	字段	类型	说明
advertisements	id	INT	广告ID
advertisements	title	VARCHAR(50)	广告标题
advertisements	content	VARCHAR(500)	广告内容

表名	字段	类型	说明
advertisements	image_url	VARCHAR(500)	广告图片URL
advertisements	product_id	INT	所属商品ID

## 5. 购物车商品表 (carts)

表名	字段	类型	说明
carts	cartitem_id	INT	购物车商品ID
carts	account_id	INT	用户ID, 关联用户表
carts	product_id	INT	商品ID, 关联商品表
carts	quantity	INT	商品数量, 默认为1

## 6. 购物车商品与订单关联表 (carts\_orders\_relation)

表名	字段	类型	说明
carts_orders_relation	carts_orders_relation_id	INT	主键ID
carts_orders_relation	cartitem_id	INT	关联购物车商品ID
carts_orders_relation	order_id	INT	关联订单ID

## 7. 商品规格表 (specifications)

表名	字段	类型	说明
specifications	specification_id	INT	规格ID
specifications	item	VARCHAR(50)	规格名称
specifications	value	VARCHAR(255)	规格内容
specifications	product_id	INT	所属商品ID

## 8. 商品库存表 (stockpiles)

表名	字段	类型	说明
stockpiles	stockpile_id	INT	商品库存ID
stockpiles	product_id	INT	所属商品ID
stockpiles	amount	INT	商品库存数, 指可卖的商品数量
stockpiles	frozen	INT	商品库存冻结数, 指不可卖的商品数量

## 9. 优惠券表 (coupon)

表名	字段	类型	说明
coupon	id	INT	优惠券ID
coupon	name	VARCHAR(100)	优惠券名称
coupon	description	VARCHAR(500)	优惠券描述
coupon	discount_type	INT	折扣类型 (1: 百分比折扣, 2: 固定金额折扣)
coupon	discount_value	DECIMAL(10, 2)	折扣值
coupon	start_time	TIMESTAMP	优惠券开始时间
coupon	end_time	TIMESTAMP	优惠券结束时间
coupon	quantity	INT	优惠券总数量
coupon	used_quantity	INT	已使用数量, 默认为0
coupon	is_valid	INT	是否有效 (1: 可用, 0: 不可用), 默认为1

## 10. 用户优惠券关联表 (account\_coupons\_relation)

表名	字段	类型	说明
account_coupons_relation	id	INT	主键ID
account_coupons_relation	account_id	INT	用户ID
account_coupons_relation	coupon_id	INT	优惠券ID
account_coupons_relation	quantity	INT	用户拥有的优惠券数量

## 11. 订单商品表 (order\_items)

表名	字段	类型	说明
order_items	order_item_id	INT	订单商品ID
order_items	order_id	INT	订单ID
order_items	product_id	INT	商品ID
order_items	quantity	INT	商品数量
order_items	price	DECIMAL(10, 2)	商品单价

## 12. 评论表 (reviews)

表名	字段	类型	说明
reviews	id	INT	评论ID
reviews	account_id	INT	评论用户ID
reviews	content	TEXT	评论内容
reviews	rate	DECIMAL(3, 2)	评分 (0.00-5.00)
reviews	type	VARCHAR(10)	评论类型 (PRODUCT: 商品评论, SHOP: 商店评论)
reviews	product_id	INT	商品ID (商品评论时使用)
reviews	shop_id	INT	商店ID (商店评论时使用)
reviews	created_at	TIMESTAMP	评论创建时间

## 13. 商店表 (shops)

表名	字段	类型	说明
shops	id	INT	商店ID
shops	name	VARCHAR(100)	商店名称
shops	owner_id	INT	商店所有者ID
shops	icon_url	VARCHAR(255)	商店图标URL
shops	description	VARCHAR(255)	商店描述
shops	rate	DECIMAL(3, 2)	商店评分 (0.00-5.00)
shops	is_valid	INT	是否有效 (1: 有效, 0: 无效)

## 14. 消息表 (messages)

表名	字段	类型	说明
messages	id	INT	消息ID
messages	content	TEXT	消息内容
messages	is_read	BOOLEAN	是否已读, 默认为false
messages	from_user	INT	发送者用户ID
messages	to_user	INT	接收者用户ID
messages	created_time	TIMESTAMP	消息创建时间

## 15. 商品规格表 (specifications)

表名	字段	类型	说明
specifications	specification_id	INT	规格ID
specifications	item	VARCHAR(50)	规格名称
specifications	value	VARCHAR(255)	规格内容
specifications	product_id	INT	所属商品ID

## 16. 商品库存表 (stockpiles)

表名	字段	类型	说明
stockpiles	stockpile_id	INT	商品库存ID
stockpiles	product_id	INT	所属商品ID
stockpiles	amount	INT	商品库存数，指可卖的商品数量
stockpiles	frozen	INT	商品库存冻结数，指不可卖的商品数量

## 17. 购物车商品与订单关联表 (carts\_orders\_relation)

表名	字段	类型	说明
carts_orders_relation	carts_orders_relation_id	INT	主键ID
carts_orders_relation	cartitem_id	INT	关联购物车商品ID
carts_orders_relation	order_id	INT	关联订单ID