详细设计报告

Version 2.0 ● 2021.2.22





**详细设计报告**

**【融销通**】

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： | QST-22软件工程1班-项目代码-DOC-MD |
| 当前版本： | 1.0 |
| 作 者： |  |
| 完成日期： | 2024-11-5 |
| 批 准 人： |  |
| 批准日期： | 2024-11-5 |
| 签 字： |  |

【扛把子小组】

**变更历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **版本** | **变更日期** | **变更内容** | **变更者** |
| 1 | 1.1 | 2024.11.12 | 目录格式整改 | 崔学童 |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |
| 11 |  |  |  |  |
| 12 |  |  |  |  |

目录

[1. 文档介绍 4](#_Toc182299492)

[1.1 文档目的 4](#_Toc182299493)

[1.1.1指导开发与实施 4](#_Toc182299494)

[1.1.2便于测试与验证 4](#_Toc182299495)

[1.1.3促进沟通与协作 5](#_Toc182299496)

[1.2 文档范围 5](#_Toc182299497)

[1.2.1模块汇总 5](#_Toc182299498)

[1.2.2子功能模块 A 的功能类设计 5](#_Toc182299499)

[1.2.3子功能模块 B 的功能类设计 5](#_Toc182299500)

[1.2.4其他 5](#_Toc182299501)

[1.3 读者对象 6](#_Toc182299502)

[1.4 参考文献 6](#_Toc182299503)

[1.5 术语与缩写解释 6](#_Toc182299504)

[2. 模块命名规则 7](#_Toc182299505)

[3. 模块汇总 7](#_Toc182299506)

[3.1 模块汇总表 7](#_Toc182299507)

[3.2 可复用功能类列表 13](#_Toc182299508)

[3.3 功能类关系图（组件关系图） 13](#_Toc182299509)

[4. 子功能模块A的功能类设计 14](#_Toc182299510)

[3.2 模块A界面设计 57](#_Toc182299511)

[4.子功能模块B的功能类设计 60](#_Toc182299512)

[4.1 模块B功能设计 60](#_Toc182299513)

[5. 其他 62](#_Toc182299514)

# 1. 文档介绍

## 1.1 文档目的

### 1.1.1指导开发与实施

1. 为开发团队提供详细的技术架构说明。采用 B/S 架构以及 SpringBoot+Vue 前后端分离技术开发，有助于开发人员明确技术选型和开发方向，确保系统的稳定性和可扩展性。
2. 详细阐述系统主要功能的设计，包括融资贷款、农产品交易和种植指导等功能模块。对于融资贷款功能，明确贷款申请人的操作流程，无论是选择和指定人一起申请贷款，还是通过系统自动匹配合适人选进行贷款，都要有清晰的业务逻辑和数据流程。
3. 在农产品交易模块中，明确不同用户角色（农民、销售平台、普通买家）的具体操作和交互方式，确保平台能够满足各方的需求，实现高效的农产品交易。
4. 对于种植指导功能，说明农户如何观看种植讲解视频和文章、与其他农户交流以及预约专家进行线下指导的流程，为农户提供切实可行的种植帮助。

### 1.1.2便于测试与验证

1. 详细设计报告为测试人员提供了明确的测试依据。测试人员可以根据报告中的功能描述和业务流程，制定全面的测试计划和测试用例，确保系统的功能正确性和稳定性。
2. 通过对系统各个功能模块的详细描述，测试人员可以更好地理解系统的设计意图，从而更有针对性地进行测试，提高测试效率和质量。

### 1.1.3促进沟通与协作

(1)详细设计报告作为一份重要的文档，能够促进项目团队内部以及与相关各方之间的沟通与协作。开发人员、测试人员、产品经理等可以通过报告更好地理解彼此的工作内容和需求，避免沟通不畅导致的问题。

(2)对于投资方、销售平台等外部合作方，详细设计报告可以让他们更好地了解平台的功能和优势，增强合作信心，促进合作的顺利开展。

总之，详细设计报告的目的是为了全面、系统地描述 “融销通” 平台的设计方案，为开发、测试、实施和维护提供指导，确保平台能够满足用户需求，实现项目目标。

再详细介绍一下种植指导功能的具体操作和交互方式

为确保系统的稳定性和可扩展性，在技术架构上还需要注意哪些方面？

分享一些“融销通”平台的成功案例

## 1.2 文档范围

### 1.2.1模块汇总

对 “融销通” 平台的各个功能模块进行整体梳理和汇总，呈现平台的功能架构体系，让读者对平台的功能全貌有清晰的认识。

### 1.2.2子功能模块的功能类设计

详细描述子功能模块 A 的具体设计，包括功能的实现方式、业务流程、数据交互等方面。可能涉及到功能的操作步骤、输入输出要求、界面设计要点等内容，为开发人员提供具体的实现指导。

### 1.2.3子功能模块的功能类设计

如同子功能模块 A，对子功能模块 B 进行深入剖析和设计阐述。涵盖功能的各个方面，从用户角度出发描述其使用场景和效果，从技术角度说明实现的方法和技术要点。

### 1.2.4其他

可能包括一些辅助性的内容，如平台的性能要求、安全设计、兼容性考虑等。也可能涉及到文档的更新说明、术语解释、参考文献等，为读者提供更全面的信息支持。

总之，本文档围绕 “融销通” 平台的功能设计展开，为平台的开发、测试、维护等提供详细的指导和说明。

## 1.3 读者对象

有以下人员，开发人员，用户，测试人员，项目经理。

## 1.4 参考文献

**SpringBoot 相关书籍或文档**：官方的 Spring Boot 文档，详细介绍了 Spring Boot 的各种特性、配置、开发方法等，对于理解和使用 Spring Boot 框架开发农产品融销一体平台非常有帮助。

**Vue.js 官方文档**：Vue.js 是前端开发的重要框架，其官方文档对于掌握 Vue 的基本概念、组件开发、路由管理、状态管理等方面提供了权威的指导。

**《数据库系统概论》**：农产品融销一体平台需要使用数据库来存储和管理数据，这本书可以帮助你深入理解数据库的基本原理、设计方法和管理技术，为平台的数据库设计和实现提供理论支持。

## 1.5 术语与缩写解释

|  |  |
| --- | --- |
| **缩写、术语** | **解 释** |
| **B/S 架构** | Browser/Server（浏览器 / 服务器）架构，用户通过浏览器访问服务器上的应用程序，无需安装专门的客户端软件 |
| **SpringBoot+vue** | 一种用于快速构建 Java 应用程序的开源框架，简化了 Spring 应用的开发和部署过程，前段使用vue框架。 |
| **模块汇总表** | 相对独立的软件设计单元 |
|  |  |
| … |  |

# 2. 模块命名规则

***提示：****模块设计人员确定本软件的模块命名规则（例如类、函数、变量等），确保模块设计文档的风格与代码的风格保持一致。可以从机构的编程规范中摘取或引用（如果存在的话）。*

# 3. 模块汇总

## 3.1 模块汇总表

|  |
| --- |
| 系统用户信息 |
| *功能类名称* | *功能简述* |
| SysUser*Controller.java* | *系统用户信息控制器层，负责页面跳转，调用Service层实现业务逻辑*  *等功能实现。* |
| SysUser*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如添加系统用户、*  *系统用户信息的可视化、修改系统用户信息等。* |
| SysUser*ServiceImpl.java* | *Service层实现类，实现*SysUser*Service.java接口。* |
| SysUser*Model.java* | *实体类，数据库的sys\_user用户信息表的字段映射。* |
| SysUser*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取系统用户信息等。* |
| SysUser*Mapper.xml* | *Dao层数据访问实现类，实现*SysUser*Mapper.java方法* |

|  |
| --- |
| ***商品和需求管理*** |
| *功能类名称* | *功能简述* |
| FlGoods*Controller.java* | *商品和需求控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlGoods*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如查看商品和商品的需求、编辑商品和需求等。* |
| FlGoods*ServiceImpl.java* | *Service层实现类，实现*FlGoods*Service.java接口。* |
| FlGoods*Model.java* | *实体类，数据库的fl\_goods商品和需求表的字段映射。* |
| FlGoods*Mapper.java* | *Dao层数据访问接口，负责定义方法如获取商品信息和需求信息等。* |
| FlGoods*Mapper.xml* | *Dao层数据访问实现类，实现*FlGoods*Mapper.java方法* |

|  |
| --- |
| ***规格库存*** |
| *功能类名称* | *功能简述* |
| FlSpecification*Controller.java* | *规格库存控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlSpecification*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如查看商品的库存、更新商品库存、下架商品等。* |
| FlSpecification*ServiceImpl.java* | *Service层实现类，实现*FlSpecification*Service.java接口。* |
| FlSpecification*Model.java* | *实体类，数据库的fl\_specification规格库存表的字段映射。* |
| FlSpecification*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取商品的库存等。* |
| FlSpecification*Mapper.xml* | *Dao层数据访问实现类，实现*FlSpecification*Mapper.java方法* |

|  |
| --- |
| ***农业知识页面*** |
| *功能类名称* | *功能简述* |
| FlAgrKnowledge*Controller.java* | *农业知识页面控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlAgrKnowledge*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如获取知识列表、发布知识、编辑知识等。* |
| FlAgrKnowledge*ServiceImpl.java* | *Service层实现类，实现*FlAgrKnowledge*Service.java接口。* |
| FlAgrKnowledge*Model.java* | *实体类，数据库的fl\_agr\_knowledge农业知识表的字段映射。* |
| FlAgrKnowledge*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取农业具体知识等。* |
| FlAgrKnowledge*Mapper.xml* | *Dao层数据访问实现类，实现*FlAgrKnowledge*Mapper.java方法* |

|  |
| --- |
| ***农业知识点赞*** |
| *功能类名称* | *功能简述* |
| FlAgrKnowLike*Controller.java* | *农业知识点赞控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlAgrKnowLike*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如记录点赞信息、查看点赞过的知识、查看其他人的点赞等。* |
| FlAgrKnowLike*ServiceImpl.java* | *Service层实现类，实现*FlAgrKnowLike*Service.java接口。* |
| FlAgrKnowLike*Model.java* | *实体类，数据库的fl\_agr\_know\_like农业知识点赞表的字段映射。* |
| FlAgrKnowLike*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取点赞信息、排序点赞数的多少等。* |
| FlAgrKnowLike*Mapper.xml* | *Dao层数据访问实现类，实现*FlAgrKnowLike*Mapper.java方法* |

|  |
| --- |
| ***农业知识评论信息*** |
| *功能类名称* | *功能简述* |
| FlAgrKnowComment*Controller.java* | *农业知识评论控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlAgrKnowComment*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如评论农业知识、查看评论过的农业知识等。* |
| FlAgrKnowComment*ServiceImpl.java* | *Service层实现类，实现*FlAgrKnowComment*Service.java接口。* |
| FlAgrKnowComment*Model.java* | *实体类，数据库的fl\_agr\_know\_comment农业知识评论表的字段映射。* |
| FlAgrKnowComment*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取农业知识评论表等。* |
| FlAgrKnowComment*Mapper.xml* | *Dao层数据访问实现类，实现*FlAgrKnowComment*Mapper.java方法* |

|  |
| --- |
| ***商品评价*** |
| *功能类名称* | *功能简述* |
| FlGoodsComment*Controller.java* | *商品评价控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlGoodsComment*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如对商品评价、查看其他人的商品评价、查看评价过的商品等。* |
| FlGoodsComment*ServiceImpl.java* | *Service层实现类，实现*FlGoodsComment*Service.java接口。* |
| FlGoodsComment*Model.java* | *实体类，数据库的fl\_goods\_comment商品评价表的字段映射。* |
| FlGoodsComment*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取商品评价表等。* |
| FlGoodsComment*Mapper.xml* | *Dao层数据访问实现类，实现*FlGoodsComment*Mapper.java方法* |

|  |
| --- |
| ***购物车管理*** |
| *功能类名称* | *功能简述* |
| FlShoppingCart*Controller.java* | *购物车管理控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlShoppingCart*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如添加商品到购物车、移除商品、更新商品数量、获取购物车详情等。* |
| FlShoppingCart*ServiceImpl.java* | *Service层实现类，实现*FlShoppingCart*Service.java接口。* |
| FlShoppingCart*Model.java* | *实体类，数据库的fl\_shopping\_cart购物车表的字段映射。* |
| FlShoppingCart*Mapper.java* | *Dao层数据访问接口，负责读取购物车表等。* |
| FlShoppingCart*Mapper.xml* | *Dao层数据访问实现类，实现*FlShoppingCart*Mapper.java方法* |

|  |
| --- |
| ***优惠券管理*** |
| *功能类名称* | *功能简述* |
| FlCoupon*Controller.java* | *优惠券管理控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlCoupon*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如查看优惠券、优惠券的使用等。* |
| FlCoupon*ServiceImpl.java* | *Service层实现类，实现*FlCoupon*Service.java接口。* |
| FlCoupon*Model.java* | *实体类，数据库的fl\_coupon优惠劵表的字段映射。* |
| FlCoupon*Mapper.java* | *Dao层数据访问接口，负责读取优惠券表等。* |
| FlCoupon*Mapper.xml* | *Dao层数据访问实现类，实现*FlCoupon*Mapper.java方法* |

|  |
| --- |
| ***商品订单*** |
| *功能类名称* | *功能简述* |
| FlGoodsOrder*Controller.java* | *商品订单控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlGoodsOrder*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如增加商品订单、订单追踪、更新商品订单等。* |
| FlGoodsOrder*ServiceImpl.java* | *Service层实现类，实现*FlGoodsOrder*Service.java接口。* |
| FlGoodsOrder*Model.java* | *实体类，数据库的fl\_goods\_order商品订单表的字段映射。* |
| FlGoodsOrder*Mapper.java* | *Dao层数据访问接口，负责读取商品订单信息、修改商品订单信息等。* |
| FlGoodsOrder*Mapper.xml* | *Dao层数据访问实现类，实现*FlGoodsOrder*Mapper.java方法* |

|  |
| --- |
| ***用户地址管理*** |
| *功能类名称* | *功能简述* |
| FlAddress*Controller.java* | *用户地址管理控制器层，负责页面跳转，调用Service层实现业务逻辑等功能实现。* |
| FlAddress*Service.java* | *Service层接口，负责业务逻辑处理，负责用户地址信息录入、用户地址的可视化等。* |
| FlAddress*ServiceImpl.java* | *Service层实现类，实现*FlAddress*Service.java接口。* |
| FlAddress*Model.java* | *实体类，数据库的fl\_address用户地址表的字段映射。* |
| FlAddress*Mapper.java* | *Dao层数据访问接口，负责读取用户地址信息、修改用户地址信息等。* |
| FlAddress*Mapper.xml* | *Dao层数据访问实现类，实现*FlAddress*Mapper.java方法* |

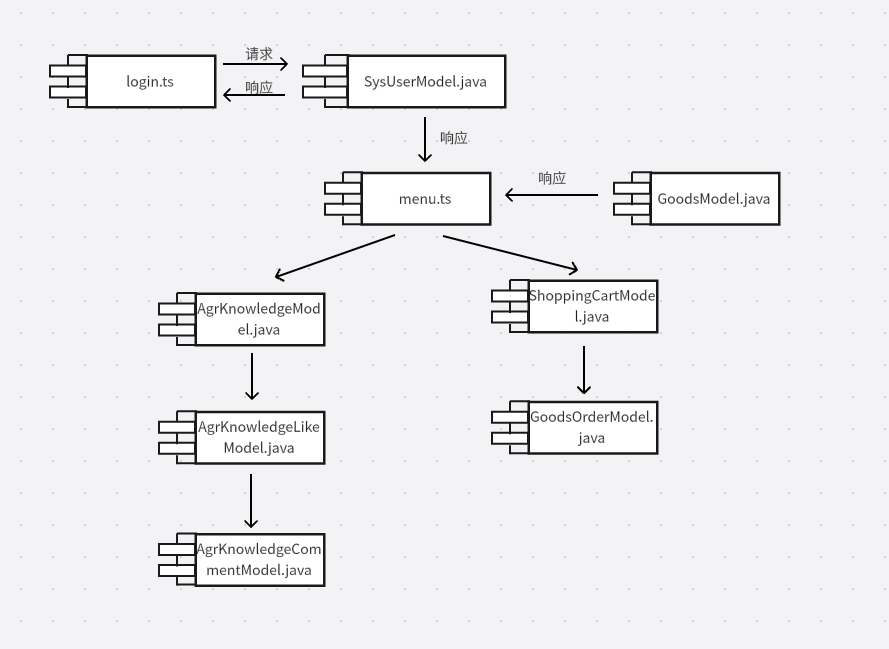
|  |
| --- |
| ***商品种类信息管理*** |
| *功能类名称* | *功能简述* |
| FlVarietys*Controller.java* | *商品种类信息管理控制器层，负责实现打开新增页面，数据新增、打开编辑页面、数据编辑、数据删除。* |
| FlVarietys*Service.java* | *Service层接口，负责业务逻辑处理，负责定义方法如商品信息录入、商品种类分类等。* |
| FlVarietys*ServiceImpl.java* | *Service层实现类，实现*FlVarietys*Service.java接口。* |
| FlVarietys*Model.java* | *实体类，数据库的fl\_varietys种类树表的字段映射。* |
| FlVarietys*Mapper.java* | *Dao层数据访问接口，负责定义方法如读取商品信息、修改商品信息等。* |
| FlVarietys*Mapper.xml* | *Dao层数据访问实现类，实现*FlVarietys*Mapper.java方法* |

## 3.2 可复用功能类列表

***提示：****如果运用了可复用策略，则需要在此处列出符合该策略的模块名称列表，并在后续模块设计中有所考虑。*

* DateUtils 时间工具类
* MapstructUtils Mapstruct 工具类
* MessageUtils 获取i18n资源文件
* ServletUtils 客户端工具类
* SpringUtils spring工具类
* StreamUtils stream 流工具类
* StringUtils 字符串工具类
* Threads 线程相关工具类.
* TreeBuildUtils 扩展 hutool TreeUtil 封装系统树构建
* ValidatorUtils Validator 校验框架工具

## 3.3 功能类关系图（组件关系图）

**

# 4. 子功能模块A的功能类设计

4.1 模块A功能设计

|  |  |
| --- | --- |
| **功能类名称** | **SysUserModel.java** |
| **功能描述** | 数据库的 t\_sys\_user 用户信息表的字段映射 |
| **接口与属性** | 根据 t\_sys\_user 用户信息表的字段，添加对应的属性。<br><br>java<br>public class SysUserModel {<br> private Long userId; // 用户ID<br> private String username; // 用户名<br> private String password; // 密码<br> private String email; // 用户邮箱<br> private String phone; // 用户电话<br> private Date createdAt; // 创建时间<br> private Date updatedAt; // 更新时间<br> // getter & setter 方法<br> public Long getUserId() {<br> return userId;<br> }<br> public void setUserId(Long userId) {<br> this.userId = userId;<br> }<br> public String getUsername() {<br> return username;<br> }<br> public void setUsername(String username) {<br> this.username = username;<br> }<br> public String getPassword() {<br> return password;<br> }<br> public void setPassword(String password) {<br> this.password = password;<br> }<br> public String getEmail() {<br> return email;<br> }<br> public void setEmail(String email) {<br> this.email = email;<br> }<br> public String getPhone() {<br> return phone;<br> }<br> public void setPhone(String phone) {<br> this.phone = phone;<br> }<br> public Date getCreatedAt() {<br> return createdAt;<br> }<br> public void setCreatedAt(Date createdAt) {<br> this.createdAt = createdAt;<br> }<br> public Date getUpdatedAt() {<br> return updatedAt;<br> }<br> public void setUpdatedAt(Date updatedAt) {<br> this.updatedAt = updatedAt;<br> }<br>} |
| **数据结构与算法** | 使用 SysUserModel 类映射数据库中的 t\_sys\_user 表字段。<br>使用 Date 类型存储创建时间和更新时间，便于后续的时间计算和管理。 |
| **补充说明** | SysUserModel 用于与数据库交互，存取用户信息。<br>本类需要配合 Mapper 使用，以实现对用户信息表的CRUD操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **SysUserService.java** |
| **功能描述** | 提供用户信息管理的服务接口，包括用户的注册、登录、修改等功能。 |
| **接口与属性** | java<br>public interface SysUserService {<br> boolean register(SysUserModel user); // 注册<br> SysUserModel login(String username, String password); // 登录<br> boolean updateUserInfo(Long userId, SysUserModel newUserInfo); // 修改用户信息<br>} |
| **数据结构与算法** | 无特殊数据结构与算法，仅作为用户信息管理的业务接口。 |
| **补充说明** | 本接口定义了用户信息相关的基本业务逻辑：注册、登录、信息修改。<br>接口实现将调用 Mapper 层提供的数据库操作方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **SysUserServiceImpl.java** |
| **功能描述** | 实现 SysUserService 接口，提供具体的业务逻辑处理。 |
| **接口与属性** | java<br>@Service<br>public class SysUserServiceImpl implements SysUserService {<br> @Autowired<br> private SysUserMapper sysUserMapper; // 注入Mapper<br><br> @Override<br> public boolean register(SysUserModel user) {<br> return sysUserMapper.insertUser(user) > 0;<br> }<br><br> @Override<br> public SysUserModel login(String username, String password) {<br> return sysUserMapper.selectUserByUsernameAndPassword(username, password);<br> }<br><br> @Override<br> public boolean updateUserInfo(Long userId, SysUserModel newUserInfo) {<br> return sysUserMapper.updateUser(userId, newUserInfo) > 0;<br> }<br>} |
| **数据结构与算法** | 1. **register** 方法：调用 sysUserMapper.insertUser() 将用户信息插入数据库。<br>2. **login** 方法：调用 sysUserMapper.selectUserByUsernameAndPassword() 根据用户名和密码进行查询。<br>3. **updateUserInfo** 方法：调用 sysUserMapper.updateUser() 更新用户信息。 |
| **补充说明** | SysUserServiceImpl 是 SysUserService 接口的实现类，负责用户信息的实际处理逻辑。<br>该类与 SysUserMapper 配合，完成数据存取操作。 |

## 3.1 模块A功能设计

|  |  |
| --- | --- |
| **功能类名称** | **SysUserMapper.java** |
| **功能描述** | 提供与用户信息相关的数据库操作接口。 |
| **接口与属性** | java<br>@Mapper<br>public interface SysUserMapper {<br> int insertUser(SysUserModel user); // 插入用户信息<br> SysUserModel selectUserByUsernameAndPassword(@Param("username") String username, @Param("password") String password); // 根据用户名和密码查询用户<br> int updateUser(@Param("userId") Long userId, @Param("newUserInfo") SysUserModel newUserInfo); // 更新用户信息<br>} |
| **数据结构与算法** | 无复杂的数据结构与算法，主要提供对 SysUserModel 类的数据库操作接口。 |
| **补充说明** | SysUserMapper 是数据访问层接口，定义了基本的数据库操作方法，负责与数据库进行交互。<br>该类与 MyBatis 配合，利用 SQL 映射文件执行 SQL 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **SysUserMapper.xml** |
| **功能描述** | 实现 SysUserMapper 中定义的数据库操作方法。 |
| **接口与属性** | xml<br><mapper namespace="com.example.mapper.SysUserMapper"><br> <insert id="insertUser" parameterType="com.example.model.SysUserModel"><br> INSERT INTO t\_sys\_user (username, password, email, phone)<br> VALUES (#{username}, #{password}, #{email}, #{phone});<br> </insert><br><br> <select id="selectUserByUsernameAndPassword" resultType="com.example.model.SysUserModel"><br> SELECT \* FROM t\_sys\_user WHERE username = #{username} AND password = #{password};<br> </select><br><br> <update id="updateUser"><br> UPDATE t\_sys\_user<br> SET username = #{newUserInfo.username}, email = #{newUserInfo.email}, phone = #{newUserInfo.phone}<br> WHERE user\_id = #{userId};<br> </update><br></mapper> |
| **数据结构与算法** | 1. **insertUser** 方法：向 t\_sys\_user 表插入用户数据。<br>2. **selectUserByUsernameAndPassword** 方法：根据用户名和密码查询用户信息。<br>3. **updateUser** 方法：更新指定用户ID的用户信息。 |
| **补充说明** | SysUserMapper.xml 是 MyBatis 映射文件，负责实现 SQL 操作，确保与数据库的交互顺畅。<br>通过 @Param 注解将方法参数传递给 SQL 语句。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsModel.java** |
| **功能描述** | 数据库 fl\_goods 商品和需求表的字段映射。 |
| **接口与属性** | java<br>public class FlGoodsModel {<br> private Long id; // 商品ID<br> private String name; // 商品名称<br> private String description; // 商品描述<br> private Double price; // 商品价格<br> private Integer stock; // 库存数量<br> private String category; // 商品类别<br> private String status; // 商品状态（例如：上架、下架）<br> private Date createdAt; // 创建时间<br> private Date updatedAt; // 更新时间<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public String getName() { return name; }<br> public void setName(String name) { this.name = name; }<br><br> public String getDescription() { return description; }<br> public void setDescription(String description) { this.description = description; }<br><br> public Double getPrice() { return price; }<br> public void setPrice(Double price) { this.price = price; }<br><br> public Integer getStock() { return stock; }<br> public void setStock(Integer stock) { this.stock = stock; }<br><br> public String getCategory() { return category; }<br> public void setCategory(String category) { this.category = category; }<br><br> public String getStatus() { return status; }<br> public void setStatus(String status) { this.status = status; }<br><br> public Date getCreatedAt() { return createdAt; }<br> public void setCreatedAt(Date createdAt) { this.createdAt = createdAt; }<br><br> public Date getUpdatedAt() { return updatedAt; }<br> public void setUpdatedAt(Date updatedAt) { this.updatedAt = updatedAt; }<br>} |
| **数据结构** | 无复杂的数据结构与算法，主要将 FlGoodsModel 类的字段与数据库表对应。 |
| **构与算法** | 使用 Double 类型存储价格，Integer 类型存储库存，Date 类型存储创建时间和更新时间，便于后续的时间计算和管理。 |
| **补充说明** | FlGoodsModel 用于与数据库交互，存取商品和需求信息。本类需要配合 Mapper 使用，以实现对商品和需求表的 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsServiceImpl.java** |
| **功能描述** | 实现商品和需求的服务接口逻辑，完成具体业务操作。 |
| **接口与方法** | java<br>@Service<br>public class FlGoodsServiceImpl implements FlGoodsService {<br> @Autowired<br> private FlGoodsMapper flGoodsMapper;<br><br> @Override<br> public FlGoodsModel getById(Long id) {<br> return flGoodsMapper.findById(id);<br> }<br><br> @Override<br> public List<FlGoodsModel> getAll() {<br> return flGoodsMapper.findAll();<br> }<br><br> @Override<br> public void addGoods(FlGoodsModel goods) {<br> flGoodsMapper.insert(goods);<br> }<br><br> @Override<br> public void updateGoods(FlGoodsModel goods) {<br> flGoodsMapper.update(goods);<br> }<br><br> @Override<br> public void deleteGoods(Long id) {<br> flGoodsMapper.delete(id);<br> }<br>} |
| **补充说明** | FlGoodsServiceImpl 使用 FlGoodsMapper 进行数据库操作，实现业务逻辑。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsMapper.java** |
| **功能描述** | 商品和需求的数据库映射接口，定义数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlGoodsMapper {<br> FlGoodsModel findById(Long id);<br> List<FlGoodsModel> findAll();<br> void insert(FlGoodsModel goods);<br> void update(FlGoodsModel goods);<br> void delete(Long id);<br>} |
| **补充说明** | 该接口定义数据库访问的 CRUD 方法，并与 Mapper.xml 文件关联。 |

|  |  |
| --- | --- |
| **文件名称** | **FlGoodsMapper.xml** |
| **功能描述** | 定义数据库的 SQL 语句，映射 FlGoodsMapper 接口方法。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br> <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlGoodsMapper"><br> <select id="findById" parameterType="Long" resultType="FlGoodsModel"><br> SELECT \* FROM fl\_goods WHERE id = #{id}<br> </select><br><br> <select id="findAll" resultType="FlGoodsModel"><br> SELECT \* FROM fl\_goods<br> </select><br><br> <insert id="insert" parameterType="FlGoodsModel"><br> INSERT INTO fl\_goods (name, description, price, stock, category, status, created\_at, updated\_at)<br> VALUES (#{name}, #{description}, #{price}, #{stock}, #{category}, #{status}, #{createdAt}, #{updatedAt})<br> </insert><br><br> <update id="update" parameterType="FlGoodsModel"><br> UPDATE fl\_goods SET name=#{name}, description=#{description}, price=#{price}, stock=#{stock}, category=#{category},<br> status=#{status}, created\_at=#{createdAt}, updated\_at=#{updatedAt} WHERE id=#{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_goods WHERE id = #{id}<br> </delete><br></mapper> |
| **补充说明** | FlGoodsMapper.xml 文件中定义了 CRUD 的 SQL 语句，与 FlGoodsMapper 接口方法关联。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlSpecificationModel.java** |
| **功能描述** | 数据库 fl\_specification 表的字段映射类，管理商品规格及库存信息。 |
| **接口与属性** | java<br>public class FlSpecificationModel {<br> private Long id; // 规格ID<br> private Long productId; // 商品ID<br> private String specName; // 规格名称<br> private Integer stock; // 规格库存<br> private Double price; // 规格价格<br> private Date createdAt; // 创建时间<br> private Date updatedAt; // 更新时间<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getProductId() { return productId; }<br> public void setProductId(Long productId) { this.productId = productId; }<br><br> public String getSpecName() { return specName; }<br> public void setSpecName(String specName) { this.specName = specName; }<br><br> public Integer getStock() { return stock; }<br> public void setStock(Integer stock) { this.stock = stock; }<br><br> public Double getPrice() { return price; }<br> public void setPrice(Double price) { this.price = price; }<br><br> public Date getCreatedAt() { return createdAt; }<br> public void setCreatedAt(Date createdAt) { this.createdAt = createdAt; }<br><br> public Date getUpdatedAt() { return updatedAt; }<br> public void setUpdatedAt(Date updatedAt) { this.updatedAt = updatedAt; }<br>} |
| **数据结构** | 无复杂的数据结构，主要将 FlSpecificationModel 的字段与数据库表字段一一对应。 |
| **补充说明** | FlSpecificationModel 提供了 getter 和 setter 方法，便于存取数据库中的规格信息。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlSpecificationService.java** |
| **功能描述** | 提供商品规格的服务接口，用于定义规格库存的业务逻辑。 |
| **接口与方法** | java<br>public interface FlSpecificationService {<br> FlSpecificationModel getById(Long id);<br> List<FlSpecificationModel> getByProductId(Long productId);<br> void addSpecification(FlSpecificationModel spec);<br> void updateSpecification(FlSpecificationModel spec);<br> void deleteSpecification(Long id);<br>} |
| **补充说明** | FlSpecificationService 接口定义了与商品规格相关的业务方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlSpecificationServiceImpl.java** |
| **功能描述** | 实现 FlSpecificationService 接口，执行商品规格的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlSpecificationServiceImpl implements FlSpecificationService {<br> @Autowired<br> private FlSpecificationMapper flSpecificationMapper;<br><br> @Override<br> public FlSpecificationModel getById(Long id) {<br> return flSpecificationMapper.findById(id);<br> }<br><br> @Override<br> public List<FlSpecificationModel> getByProductId(Long productId) {<br> return flSpecificationMapper.findByProductId(productId);<br> }<br><br> @Override<br> public void addSpecification(FlSpecificationModel spec) {<br> flSpecificationMapper.insert(spec);<br> }<br><br> @Override<br> public void updateSpecification(FlSpecificationModel spec) {<br> flSpecificationMapper.update(spec);<br> }<br><br> @Override<br> public void deleteSpecification(Long id) {<br> flSpecificationMapper.delete(id);<br> }<br>} |
| **补充说明** | FlSpecificationServiceImpl 使用 FlSpecificationMapper 进行数据库访问和业务处理。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlSpecificationMapper.java** |
| **功能描述** | 商品规格的数据库映射接口，定义与 fl\_specification 表的操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlSpecificationMapper {<br> FlSpecificationModel findById(Long id);<br> List<FlSpecificationModel> findByProductId(Long productId);<br> void insert(FlSpecificationModel spec);<br> void update(FlSpecificationModel spec);<br> void delete(Long id);<br>} |
| **补充说明** | 接口方法用于映射 SQL 查询，并在 Mapper.xml 中实现。 |

|  |  |
| --- | --- |
| **文件名称** | **FlSpecificationMapper.xml** |
| **功能描述** | 定义与 FlSpecificationMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlSpecificationMapper"><br> <select id="findById" parameterType="Long" resultType="FlSpecificationModel"><br> SELECT \* FROM fl\_specification WHERE id = #{id}<br> </select><br><br> <select id="findByProductId" parameterType="Long" resultType="FlSpecificationModel"><br> SELECT \* FROM fl\_specification WHERE product\_id = #{productId}<br> </select><br><br> <insert id="insert" parameterType="FlSpecificationModel"><br> INSERT INTO fl\_specification (product\_id, spec\_name, stock, price, created\_at, updated\_at)<br> VALUES (#{productId}, #{specName}, #{stock}, #{price}, #{createdAt}, #{updatedAt})<br> </insert><br><br> <update id="update" parameterType="FlSpecificationModel"><br> UPDATE fl\_specification SET spec\_name=#{specName}, stock=#{stock}, price=#{price},<br> created\_at=#{createdAt}, updated\_at=#{updatedAt} WHERE id=#{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_specification WHERE id = #{id}<br> </delete><br></mapper> |
| **补充说明** | FlSpecificationMapper.xml 中定义了数据库操作的详细 SQL 语句。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowledgeModel.java** |
| **功能描述** | 数据库 fl\_agr\_knowledge 表的字段映射类，用于管理农业知识信息。 |
| **接口与属性** | java<br>public class FlAgrKnowledgeModel {<br> private Long id; // 知识ID<br> private String title; // 知识标题<br> private String content; // 知识内容<br> private String category; // 分类（如种植、病虫害等）<br> private String author; // 作者<br> private Date createdAt; // 创建时间<br> private Date updatedAt; // 更新时间<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public String getTitle() { return title; }<br> public void setTitle(String title) { this.title = title; }<br><br> public String getContent() { return content; }<br> public void setContent(String content) { this.content = content; }<br><br> public String getCategory() { return category; }<br> public void setCategory(String category) { this.category = category; }<br><br> public String getAuthor() { return author; }<br> public void setAuthor(String author) { this.author = author; }<br><br> public Date getCreatedAt() { return createdAt; }<br> public void setCreatedAt(Date createdAt) { this.createdAt = createdAt; }<br><br> public Date getUpdatedAt() { return updatedAt; }<br> public void setUpdatedAt(Date updatedAt) { this.updatedAt = updatedAt; }<br>} |
| **数据结构** | FlAgrKnowledgeModel 的字段与数据库表 fl\_agr\_knowledge 字段相对应。 |
| **补充说明** | 该模型类提供了农业知识的存取操作，用于 CRUD 业务逻辑。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowledgeService.java** |
| **功能描述** | 提供农业知识的服务接口，定义农业知识的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlAgrKnowledgeService {<br> FlAgrKnowledgeModel getById(Long id);<br> List<FlAgrKnowledgeModel> getAll();<br> List<FlAgrKnowledgeModel> getByCategory(String category);<br> void addKnowledge(FlAgrKnowledgeModel knowledge);<br> void updateKnowledge(FlAgrKnowledgeModel knowledge);<br> void deleteKnowledge(Long id);<br>} |
| **补充说明** | FlAgrKnowledgeService 定义了农业知识相关的 CRUD 接口方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowledgeServiceImpl.java** |
| **功能描述** | 实现 FlAgrKnowledgeService 接口，处理农业知识的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlAgrKnowledgeServiceImpl implements FlAgrKnowledgeService {<br> @Autowired<br> private FlAgrKnowledgeMapper flAgrKnowledgeMapper;<br><br> @Override<br> public FlAgrKnowledgeModel getById(Long id) {<br> return flAgrKnowledgeMapper.findById(id);<br> }<br><br> @Override<br> public List<FlAgrKnowledgeModel> getAll() {<br> return flAgrKnowledgeMapper.findAll();<br> }<br><br> @Override<br> public List<FlAgrKnowledgeModel> getByCategory(String category) {<br> return flAgrKnowledgeMapper.findByCategory(category);<br> }<br><br> @Override<br> public void addKnowledge(FlAgrKnowledgeModel knowledge) {<br> flAgrKnowledgeMapper.insert(knowledge);<br> }<br><br> @Override<br> public void updateKnowledge(FlAgrKnowledgeModel knowledge) {<br> flAgrKnowledgeMapper.update(knowledge);<br> }<br><br> @Override<br> public void deleteKnowledge(Long id) {<br> flAgrKnowledgeMapper.delete(id);<br> }<br>} |
| **补充说明** | FlAgrKnowledgeServiceImpl 通过 FlAgrKnowledgeMapper 访问数据库，完成业务逻辑实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowledgeMapper.java** |
| **功能描述** | 农业知识的数据库映射接口，定义与 fl\_agr\_knowledge 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlAgrKnowledgeMapper {<br> FlAgrKnowledgeModel findById(Long id);<br> List<FlAgrKnowledgeModel> findAll();<br> List<FlAgrKnowledgeModel> findByCategory(String category);<br> void insert(FlAgrKnowledgeModel knowledge);<br> void update(FlAgrKnowledgeModel knowledge);<br> void delete(Long id);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlAgrKnowledgeMapper.xml** |
| **功能描述** | 定义与 FlAgrKnowledgeMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlAgrKnowledgeMapper"><br> <select id="findById" parameterType="Long" resultType="FlAgrKnowledgeModel"><br> SELECT \* FROM fl\_agr\_knowledge WHERE id = #{id}<br> </select><br><br> <select id="findAll" resultType="FlAgrKnowledgeModel"><br> SELECT \* FROM fl\_agr\_knowledge<br> </select><br><br> <select id="findByCategory" parameterType="String" resultType="FlAgrKnowledgeModel"><br> SELECT \* FROM fl\_agr\_knowledge WHERE category = #{category}<br> </select><br><br> <insert id="insert" parameterType="FlAgrKnowledgeModel"><br> INSERT INTO fl\_agr\_knowledge (title, content, category, author, created\_at, updated\_at)<br> VALUES (#{title}, #{content}, #{category}, #{author}, #{createdAt}, #{updatedAt})<br> </insert><br><br> <update id="update" parameterType="FlAgrKnowledgeModel"><br> UPDATE fl\_agr\_knowledge SET title=#{title}, content=#{content}, category=#{category}, author=#{author},<br> created\_at=#{createdAt}, updated\_at=#{updatedAt} WHERE id=#{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_agr\_knowledge WHERE id = #{id}<br> </delete><br></mapper> |
| **补充说明** | FlAgrKnowledgeMapper.xml 文件中定义了与数据库交互的 SQL 语句。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowLikeModel.java** |
| **功能描述** | 映射数据库 fl\_agr\_know\_like 表，用于记录用户对农业知识的点赞信息。 |
| **接口与属性** | java<br>public class FlAgrKnowLikeModel {<br> private Long id; // 点赞记录ID<br> private Long knowledgeId; // 知识ID<br> private Long userId; // 用户ID<br> private Date likedAt; // 点赞时间<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getKnowledgeId() { return knowledgeId; }<br> public void setKnowledgeId(Long knowledgeId) { this.knowledgeId = knowledgeId; }<br><br> public Long getUserId() { return userId; }<br> public void setUserId(Long userId) { this.userId = userId; }<br><br> public Date getLikedAt() { return likedAt; }<br> public void setLikedAt(Date likedAt) { this.likedAt = likedAt; }<br>} |
| **数据结构** | FlAgrKnowLikeModel 字段与数据库表 fl\_agr\_know\_like 一一对应。 |
| **补充说明** | 该模型类用于点赞信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowLikeService.java** |
| **功能描述** | 提供农业知识点赞的服务接口，定义点赞业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlAgrKnowLikeService {<br> FlAgrKnowLikeModel getById(Long id);<br> List<FlAgrKnowLikeModel> getByKnowledgeId(Long knowledgeId);<br> void addLike(FlAgrKnowLikeModel like);<br> void removeLike(Long id);<br> int getLikeCount(Long knowledgeId);<br>} |
| **补充说明** | FlAgrKnowLikeService 定义了点赞相关的 CRUD 和计数方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowLikeServiceImpl.java** |
| **功能描述** | 实现 FlAgrKnowLikeService 接口，处理点赞信息的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlAgrKnowLikeServiceImpl implements FlAgrKnowLikeService {<br> @Autowired<br> private FlAgrKnowLikeMapper flAgrKnowLikeMapper;<br><br> @Override<br> public FlAgrKnowLikeModel getById(Long id) {<br> return flAgrKnowLikeMapper.findById(id);<br> }<br><br> @Override<br> public List<FlAgrKnowLikeModel> getByKnowledgeId(Long knowledgeId) {<br> return flAgrKnowLikeMapper.findByKnowledgeId(knowledgeId);<br> }<br><br> @Override<br> public void addLike(FlAgrKnowLikeModel like) {<br> flAgrKnowLikeMapper.insert(like);<br> }<br><br> @Override<br> public void removeLike(Long id) {<br> flAgrKnowLikeMapper.delete(id);<br> }<br><br> @Override<br> public int getLikeCount(Long knowledgeId) {<br> return flAgrKnowLikeMapper.countByKnowledgeId(knowledgeId);<br> }<br>} |
| **补充说明** | FlAgrKnowLikeServiceImpl 使用 FlAgrKnowLikeMapper 访问数据库，完成点赞的业务逻辑。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowLikeMapper.java** |
| **功能描述** | 点赞信息的数据库映射接口，定义与 fl\_agr\_know\_like 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlAgrKnowLikeMapper {<br> FlAgrKnowLikeModel findById(Long id);<br> List<FlAgrKnowLikeModel> findByKnowledgeId(Long knowledgeId);<br> void insert(FlAgrKnowLikeModel like);<br> void delete(Long id);<br> int countByKnowledgeId(Long knowledgeId);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlAgrKnowLikeMapper.xml** |
| **功能描述** | 定义与 FlAgrKnowLikeMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlAgrKnowLikeMapper"><br> <select id="findById" parameterType="Long" resultType="FlAgrKnowLikeModel"><br> SELECT \* FROM fl\_agr\_know\_like WHERE id = #{id}<br> </select><br><br> <select id="findByKnowledgeId" parameterType="Long" resultType="FlAgrKnowLikeModel"><br> SELECT \* FROM fl\_agr\_know\_like WHERE knowledge\_id = #{knowledgeId}<br> </select><br><br> <insert id="insert" parameterType="FlAgrKnowLikeModel"><br> INSERT INTO fl\_agr\_know\_like (knowledge\_id, user\_id, liked\_at)<br> VALUES (#{knowledgeId}, #{userId}, #{likedAt})<br> </insert><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_agr\_know\_like WHERE id = #{id}<br> </delete><br><br> <select id="countByKnowledgeId" parameterType="Long" resultType="int"><br> SELECT COUNT(\*) FROM fl\_agr\_know\_like WHERE knowledge\_id = #{knowledgeId}<br> </select><br></mapper> |
| **补充说明** | FlAgrKnowLikeMapper.xml 文件中定义了点赞表的数据库操作 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowCommentModel.java** |
| **功能描述** | 映射数据库 fl\_agr\_know\_comment 表，用于管理用户对农业知识的评论信息。 |
| **接口与属性** | java<br>public class FlAgrKnowCommentModel {<br> private Long id; // 评论ID<br> private Long knowledgeId; // 知识ID<br> private Long userId; // 用户ID<br> private String commentText; // 评论内容<br> private Date commentedAt; // 评论时间<br> private Long parentId; // 父评论ID（用于嵌套评论）<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getKnowledgeId() { return knowledgeId; }<br> public void setKnowledgeId(Long knowledgeId) { this.knowledgeId = knowledgeId; }<br><br> public Long getUserId() { return userId; }<br> public void setUserId(Long userId) { this.userId = userId; }<br><br> public String getCommentText() { return commentText; }<br> public void setCommentText(String commentText) { this.commentText = commentText; }<br><br> public Date getCommentedAt() { return commentedAt; }<br> public void setCommentedAt(Date commentedAt) { this.commentedAt = commentedAt; }<br><br> public Long getParentId() { return parentId; }<br> public void setParentId(Long parentId) { this.parentId = parentId; }<br>} |
| **数据结构** | FlAgrKnowCommentModel 字段与数据库表 fl\_agr\_know\_comment 一一对应。 |
| **补充说明** | 该模型类用于评论信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowCommentService.java** |
| **功能描述** | 提供农业知识评论的服务接口，定义评论的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlAgrKnowCommentService {<br> FlAgrKnowCommentModel getById(Long id);<br> List<FlAgrKnowCommentModel> getByKnowledgeId(Long knowledgeId);<br> void addComment(FlAgrKnowCommentModel comment);<br> void deleteComment(Long id);<br> List<FlAgrKnowCommentModel> getCommentsByParentId(Long parentId);<br>} |
| **补充说明** | FlAgrKnowCommentService 定义了评论的 CRUD 和获取子评论的方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowCommentServiceImpl.java** |
| **功能描述** | 实现 FlAgrKnowCommentService 接口，处理评论的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlAgrKnowCommentServiceImpl implements FlAgrKnowCommentService {<br> @Autowired<br> private FlAgrKnowCommentMapper flAgrKnowCommentMapper;<br><br> @Override<br> public FlAgrKnowCommentModel getById(Long id) {<br> return flAgrKnowCommentMapper.findById(id);<br> }<br><br> @Override<br> public List<FlAgrKnowCommentModel> getByKnowledgeId(Long knowledgeId) {<br> return flAgrKnowCommentMapper.findByKnowledgeId(knowledgeId);<br> }<br><br> @Override<br> public void addComment(FlAgrKnowCommentModel comment) {<br> flAgrKnowCommentMapper.insert(comment);<br> }<br><br> @Override<br> public void deleteComment(Long id) {<br> flAgrKnowCommentMapper.delete(id);<br> }<br><br> @Override<br> public List<FlAgrKnowCommentModel> getCommentsByParentId(Long parentId) {<br> return flAgrKnowCommentMapper.findByParentId(parentId);<br> }<br>} |
| **补充说明** | FlAgrKnowCommentServiceImpl 使用 FlAgrKnowCommentMapper 访问数据库，完成评论业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAgrKnowCommentMapper.java** |
| **功能描述** | 评论信息的数据库映射接口，定义与 fl\_agr\_know\_comment 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlAgrKnowCommentMapper {<br> FlAgrKnowCommentModel findById(Long id);<br> List<FlAgrKnowCommentModel> findByKnowledgeId(Long knowledgeId);<br> void insert(FlAgrKnowCommentModel comment);<br> void delete(Long id);<br> List<FlAgrKnowCommentModel> findByParentId(Long parentId);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlAgrKnowCommentMapper.xml** |
| **功能描述** | 定义与 FlAgrKnowCommentMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlAgrKnowCommentMapper"><br> <select id="findById" parameterType="Long" resultType="FlAgrKnowCommentModel"><br> SELECT \* FROM fl\_agr\_know\_comment WHERE id = #{id}<br> </select><br><br> <select id="findByKnowledgeId" parameterType="Long" resultType="FlAgrKnowCommentModel"><br> SELECT \* FROM fl\_agr\_know\_comment WHERE knowledge\_id = #{knowledgeId}<br> </select><br><br> <insert id="insert" parameterType="FlAgrKnowCommentModel"><br> INSERT INTO fl\_agr\_know\_comment (knowledge\_id, user\_id, comment\_text, commented\_at, parent\_id)<br> VALUES (#{knowledgeId}, #{userId}, #{commentText}, #{commentedAt}, #{parentId})<br> </insert><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_agr\_know\_comment WHERE id = #{id}<br> </delete><br><br> <select id="findByParentId" parameterType="Long" resultType="FlAgrKnowCommentModel"><br> SELECT \* FROM fl\_agr\_know\_comment WHERE parent\_id = #{parentId}<br> </select><br></mapper> |
| **补充说明** | FlAgrKnowCommentMapper.xml 文件中定义了与评论表的数据库交互 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsCommentModel.java** |
| **功能描述** | 映射数据库 fl\_goods\_comment 表，用于管理用户对商品的评价信息。 |
| **接口与属性** | java<br>public class FlGoodsCommentModel {<br> private Long id; // 评价ID<br> private Long goodsId; // 商品ID<br> private Long userId; // 用户ID<br> private String commentText; // 评价内容<br> private int rating; // 评分（例如1-5星）<br> private Date commentedAt; // 评价时间<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getGoodsId() { return goodsId; }<br> public void setGoodsId(Long goodsId) { this.goodsId = goodsId; }<br><br> public Long getUserId() { return userId; }<br> public void setUserId(Long userId) { this.userId = userId; }<br><br> public String getCommentText() { return commentText; }<br> public void setCommentText(String commentText) { this.commentText = commentText; }<br><br> public int getRating() { return rating; }<br> public void setRating(int rating) { this.rating = rating; }<br><br> public Date getCommentedAt() { return commentedAt; }<br> public void setCommentedAt(Date commentedAt) { this.commentedAt = commentedAt; }<br>} |
| **数据结构** | FlGoodsCommentModel 字段与数据库表 fl\_goods\_comment 一一对应。 |
| **补充说明** | 该模型类用于商品评价信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsCommentService.java** |
| **功能描述** | 提供商品评价的服务接口，定义评价的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlGoodsCommentService {<br> FlGoodsCommentModel getById(Long id);<br> List<FlGoodsCommentModel> getByGoodsId(Long goodsId);<br> void addComment(FlGoodsCommentModel comment);<br> void deleteComment(Long id);<br> double getAverageRating(Long goodsId);<br>} |
| **补充说明** | FlGoodsCommentService 定义了评价的 CRUD、按商品ID查询和计算平均评分的方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsCommentServiceImpl.java** |
| **功能描述** | 实现 FlGoodsCommentService 接口，处理商品评价的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlGoodsCommentServiceImpl implements FlGoodsCommentService {<br> @Autowired<br> private FlGoodsCommentMapper flGoodsCommentMapper;<br><br> @Override<br> public FlGoodsCommentModel getById(Long id) {<br> return flGoodsCommentMapper.findById(id);<br> }<br><br> @Override<br> public List<FlGoodsCommentModel> getByGoodsId(Long goodsId) {<br> return flGoodsCommentMapper.findByGoodsId(goodsId);<br> }<br><br> @Override<br> public void addComment(FlGoodsCommentModel comment) {<br> flGoodsCommentMapper.insert(comment);<br> }<br><br> @Override<br> public void deleteComment(Long id) {<br> flGoodsCommentMapper.delete(id);<br> }<br><br> @Override<br> public double getAverageRating(Long goodsId) {<br> return flGoodsCommentMapper.calculateAverageRating(goodsId);<br> }<br>} |
| **补充说明** | FlGoodsCommentServiceImpl 使用 FlGoodsCommentMapper 访问数据库，完成评价业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsCommentMapper.java** |
| **功能描述** | 评价信息的数据库映射接口，定义与 fl\_goods\_comment 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlGoodsCommentMapper {<br> FlGoodsCommentModel findById(Long id);<br> List<FlGoodsCommentModel> findByGoodsId(Long goodsId);<br> void insert(FlGoodsCommentModel comment);<br> void delete(Long id);<br> double calculateAverageRating(Long goodsId);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlGoodsCommentMapper.xml** |
| **功能描述** | 定义与 FlGoodsCommentMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlGoodsCommentMapper"><br> <select id="findById" parameterType="Long" resultType="FlGoodsCommentModel"><br> SELECT \* FROM fl\_goods\_comment WHERE id = #{id}<br> </select><br><br> <select id="findByGoodsId" parameterType="Long" resultType="FlGoodsCommentModel"><br> SELECT \* FROM fl\_goods\_comment WHERE goods\_id = #{goodsId}<br> </select><br><br> <insert id="insert" parameterType="FlGoodsCommentModel"><br> INSERT INTO fl\_goods\_comment (goods\_id, user\_id, comment\_text, rating, commented\_at)<br> VALUES (#{goodsId}, #{userId}, #{commentText}, #{rating}, #{commentedAt})<br> </insert><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_goods\_comment WHERE id = #{id}<br> </delete><br><br> <select id="calculateAverageRating" parameterType="Long" resultType="double"><br> SELECT COALESCE(AVG(rating), 0) FROM fl\_goods\_comment WHERE goods\_id = #{goodsId}<br> </select><br></mapper> |
| **补充说明** | FlGoodsCommentMapper.xml 文件中定义了商品评价表的数据库交互 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlShoppingCartModel.java** |
| **功能描述** | 映射数据库 fl\_shopping\_cart 表，用于管理用户的购物车信息。 |
| **接口与属性** | java<br>public class FlShoppingCartModel {<br> private Long id; // 购物车项ID<br> private Long userId; // 用户ID<br> private Long goodsId; // 商品ID<br> private int quantity; // 商品数量<br> private Date addedAt; // 添加时间<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getUserId() { return userId; }<br> public void setUserId(Long userId) { this.userId = userId; }<br><br> public Long getGoodsId() { return goodsId; }<br> public void setGoodsId(Long goodsId) { this.goodsId = goodsId; }<br><br> public int getQuantity() { return quantity; }<br> public void setQuantity(int quantity) { this.quantity = quantity; }<br><br> public Date getAddedAt() { return addedAt; }<br> public void setAddedAt(Date addedAt) { this.addedAt = addedAt; }<br>} |
| **数据结构** | FlShoppingCartModel 字段与数据库表 fl\_shopping\_cart 一一对应。 |
| **补充说明** | 该模型类用于购物车信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlShoppingCartService.java** |
| **功能描述** | 提供购物车的服务接口，定义购物车的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlShoppingCartService {<br> FlShoppingCartModel getById(Long id);<br> List<FlShoppingCartModel> getByUserId(Long userId);<br> void addItem(FlShoppingCartModel cartItem);<br> void updateItemQuantity(Long id, int quantity);<br> void removeItem(Long id);<br> void clearCart(Long userId);<br>} |
| **补充说明** | FlShoppingCartService 定义了购物车的 CRUD 和清空购物车的方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlShoppingCartServiceImpl.java** |
| **功能描述** | 实现 FlShoppingCartService 接口，处理购物车的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlShoppingCartServiceImpl implements FlShoppingCartService {<br> @Autowired<br> private FlShoppingCartMapper flShoppingCartMapper;<br><br> @Override<br> public FlShoppingCartModel getById(Long id) {<br> return flShoppingCartMapper.findById(id);<br> }<br><br> @Override<br> public List<FlShoppingCartModel> getByUserId(Long userId) {<br> return flShoppingCartMapper.findByUserId(userId);<br> }<br><br> @Override<br> public void addItem(FlShoppingCartModel cartItem) {<br> flShoppingCartMapper.insert(cartItem);<br> }<br><br> @Override<br> public void updateItemQuantity(Long id, int quantity) {<br> flShoppingCartMapper.updateQuantity(id, quantity);<br> }<br><br> @Override<br> public void removeItem(Long id) {<br> flShoppingCartMapper.delete(id);<br> }<br><br> @Override<br> public void clearCart(Long userId) {<br> flShoppingCartMapper.deleteByUserId(userId);<br> }<br>} |
| **补充说明** | FlShoppingCartServiceImpl 使用 FlShoppingCartMapper 访问数据库，完成购物车业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlShoppingCartMapper.java** |
| **功能描述** | 购物车信息的数据库映射接口，定义与 fl\_shopping\_cart 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlShoppingCartMapper {<br> FlShoppingCartModel findById(Long id);<br> List<FlShoppingCartModel> findByUserId(Long userId);<br> void insert(FlShoppingCartModel cartItem);<br> void updateQuantity(Long id, int quantity);<br> void delete(Long id);<br> void deleteByUserId(Long userId);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlShoppingCartMapper.xml** |
| **功能描述** | 定义与 FlShoppingCartMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlShoppingCartMapper"><br> <select id="findById" parameterType="Long" resultType="FlShoppingCartModel"><br> SELECT \* FROM fl\_shopping\_cart WHERE id = #{id}<br> </select><br><br> <select id="findByUserId" parameterType="Long" resultType="FlShoppingCartModel"><br> SELECT \* FROM fl\_shopping\_cart WHERE user\_id = #{userId}<br> </select><br><br> <insert id="insert" parameterType="FlShoppingCartModel"><br> INSERT INTO fl\_shopping\_cart (user\_id, goods\_id, quantity, added\_at)<br> VALUES (#{userId}, #{goodsId}, #{quantity}, #{addedAt})<br> </insert><br><br> <update id="updateQuantity" parameterType="map"><br> UPDATE fl\_shopping\_cart SET quantity = #{quantity} WHERE id = #{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_shopping\_cart WHERE id = #{id}<br> </delete><br><br> <delete id="deleteByUserId" parameterType="Long"><br> DELETE FROM fl\_shopping\_cart WHERE user\_id = #{userId}<br> </delete><br></mapper> |
| **补充说明** | FlShoppingCartMapper.xml 文件中定义了购物车表的数据库交互 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlCouponModel.java** |
| **功能描述** | 映射数据库 fl\_coupon 表，用于管理优惠券信息。 |
| **接口与属性** | java<br>public class FlCouponModel {<br> private Long id; // 优惠券ID<br> private String code; // 优惠券代码<br> private double discountAmount; // 折扣金额<br> private Date validFrom; // 有效期开始<br> private Date validUntil; // 有效期结束<br> private boolean isUsed; // 是否已使用<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public String getCode() { return code; }<br> public void setCode(String code) { this.code = code; }<br><br> public double getDiscountAmount() { return discountAmount; }<br> public void setDiscountAmount(double discountAmount) { this.discountAmount = discountAmount; }<br><br> public Date getValidFrom() { return validFrom; }<br> public void setValidFrom(Date validFrom) { this.validFrom = validFrom; }<br><br> public Date getValidUntil() { return validUntil; }<br> public void setValidUntil(Date validUntil) { this.validUntil = validUntil; }<br><br> public boolean isUsed() { return isUsed; }<br> public void setUsed(boolean used) { isUsed = used; }<br>} |
| **数据结构** | FlCouponModel 字段与数据库表 fl\_coupon 一一对应。 |
| **补充说明** | 该模型类用于优惠券信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlCouponService.java** |
| **功能描述** | 提供优惠券的服务接口，定义优惠券的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlCouponService {<br> FlCouponModel getById(Long id);<br> FlCouponModel getByCode(String code);<br> void addCoupon(FlCouponModel coupon);<br> void markAsUsed(Long id);<br> List<FlCouponModel> getValidCoupons();<br>} |
| **补充说明** | FlCouponService 定义了优惠券的 CRUD、按代码查询和标记已使用的方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlCouponServiceImpl.java** |
| **功能描述** | 实现 FlCouponService 接口，处理优惠券的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlCouponServiceImpl implements FlCouponService {<br> @Autowired<br> private FlCouponMapper flCouponMapper;<br><br> @Override<br> public FlCouponModel getById(Long id) {<br> return flCouponMapper.findById(id);<br> }<br><br> @Override<br> public FlCouponModel getByCode(String code) {<br> return flCouponMapper.findByCode(code);<br> }<br><br> @Override<br> public void addCoupon(FlCouponModel coupon) {<br> flCouponMapper.insert(coupon);<br> }<br><br> @Override<br> public void markAsUsed(Long id) {<br> flCouponMapper.updateUsageStatus(id, true);<br> }<br><br> @Override<br> public List<FlCouponModel> getValidCoupons() {<br> return flCouponMapper.findValidCoupons();<br> }<br>} |
| **补充说明** | FlCouponServiceImpl 使用 FlCouponMapper 访问数据库，完成优惠券业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlCouponMapper.java** |
| **功能描述** | 优惠券信息的数据库映射接口，定义与 fl\_coupon 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlCouponMapper {<br> FlCouponModel findById(Long id);<br> FlCouponModel findByCode(String code);<br> void insert(FlCouponModel coupon);<br> void updateUsageStatus(Long id, boolean isUsed);<br> List<FlCouponModel> findValidCoupons();<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlCouponMapper.xml** |
| **功能描述** | 定义与 FlCouponMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlCouponMapper"><br> <select id="findById" parameterType="Long" resultType="FlCouponModel"><br> SELECT \* FROM fl\_coupon WHERE id = #{id}<br> </select><br><br> <select id="findByCode" parameterType="String" resultType="FlCouponModel"><br> SELECT \* FROM fl\_coupon WHERE code = #{code}<br> </select><br><br> <insert id="insert" parameterType="FlCouponModel"><br> INSERT INTO fl\_coupon (code, discount\_amount, valid\_from, valid\_until, is\_used)<br> VALUES (#{code}, #{discountAmount}, #{validFrom}, #{validUntil}, #{isUsed})<br> </insert><br><br> <update id="updateUsageStatus" parameterType="map"><br> UPDATE fl\_coupon SET is\_used = #{isUsed} WHERE id = #{id}<br> </update><br><br> <select id="findValidCoupons" resultType="FlCouponModel"><br> SELECT \* FROM fl\_coupon WHERE is\_used = false AND valid\_from <= NOW() AND valid\_until >= NOW()<br> </select><br></mapper> |
| **补充说明** | FlCouponMapper.xml 文件中定义了优惠券表的数据库交互 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsOrderModel.java** |
| **功能描述** | 映射数据库 fl\_goods\_order 表，用于管理商品订单信息。 |
| **接口与属性** | java<br>public class FlGoodsOrderModel {<br> private Long id; // 订单ID<br> private Long userId; // 用户ID<br> private Long goodsId; // 商品ID<br> private int quantity; // 商品数量<br> private double totalAmount; // 总金额<br> private String status; // 订单状态<br> private Date orderDate; // 下单日期<br> private Date deliveryDate; // 发货日期<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getUserId() { return userId; }<br> public void setUserId(Long userId) { this.userId = userId; }<br><br> public Long getGoodsId() { return goodsId; }<br> public void setGoodsId(Long goodsId) { this.goodsId = goodsId; }<br><br> public int getQuantity() { return quantity; }<br> public void setQuantity(int quantity) { this.quantity = quantity; }<br><br> public double getTotalAmount() { return totalAmount; }<br> public void setTotalAmount(double totalAmount) { this.totalAmount = totalAmount; }<br><br> public String getStatus() { return status; }<br> public void setStatus(String status) { this.status = status; }<br><br> public Date getOrderDate() { return orderDate; }<br> public void setOrderDate(Date orderDate) { this.orderDate = orderDate; }<br><br> public Date getDeliveryDate() { return deliveryDate; }<br> public void setDeliveryDate(Date deliveryDate) { this.deliveryDate = deliveryDate; }<br>} |
| **数据结构** | FlGoodsOrderModel 字段与数据库表 fl\_goods\_order 一一对应。 |
| **补充说明** | 该模型类用于订单信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsOrderService.java** |
| **功能描述** | 提供商品订单的服务接口，定义商品订单的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlGoodsOrderService {<br> FlGoodsOrderModel getById(Long id);<br> List<FlGoodsOrderModel> getByUserId(Long userId);<br> void createOrder(FlGoodsOrderModel order);<br> void updateOrderStatus(Long id, String status);<br> void deleteOrder(Long id);<br> List<FlGoodsOrderModel> getOrdersByStatus(String status);<br>} |
| **补充说明** | FlGoodsOrderService 定义了订单的 CRUD、按状态查询和更新状态的方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsOrderServiceImpl.java** |
| **功能描述** | 实现 FlGoodsOrderService 接口，处理商品订单的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlGoodsOrderServiceImpl implements FlGoodsOrderService {<br> @Autowired<br> private FlGoodsOrderMapper flGoodsOrderMapper;<br><br> @Override<br> public FlGoodsOrderModel getById(Long id) {<br> return flGoodsOrderMapper.findById(id);<br> }<br><br> @Override<br> public List<FlGoodsOrderModel> getByUserId(Long userId) {<br> return flGoodsOrderMapper.findByUserId(userId);<br> }<br><br> @Override<br> public void createOrder(FlGoodsOrderModel order) {<br> flGoodsOrderMapper.insert(order);<br> }<br><br> @Override<br> public void updateOrderStatus(Long id, String status) {<br> flGoodsOrderMapper.updateStatus(id, status);<br> }<br><br> @Override<br> public void deleteOrder(Long id) {<br> flGoodsOrderMapper.delete(id);<br> }<br><br> @Override<br> public List<FlGoodsOrderModel> getOrdersByStatus(String status) {<br> return flGoodsOrderMapper.findByStatus(status);<br> }<br>} |
| **补充说明** | FlGoodsOrderServiceImpl 使用 FlGoodsOrderMapper 访问数据库，完成订单业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlGoodsOrderMapper.java** |
| **功能描述** | 商品订单信息的数据库映射接口，定义与 fl\_goods\_order 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlGoodsOrderMapper {<br> FlGoodsOrderModel findById(Long id);<br> List<FlGoodsOrderModel> findByUserId(Long userId);<br> void insert(FlGoodsOrderModel order);<br> void updateStatus(Long id, String status);<br> void delete(Long id);<br> List<FlGoodsOrderModel> findByStatus(String status);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlGoodsOrderMapper.xml** |
| **功能描述** | 定义与 FlGoodsOrderMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlGoodsOrderMapper"><br> <select id="findById" parameterType="Long" resultType="FlGoodsOrderModel"><br> SELECT \* FROM fl\_goods\_order WHERE id = #{id}<br> </select><br><br> <select id="findByUserId" parameterType="Long" resultType="FlGoodsOrderModel"><br> SELECT \* FROM fl\_goods\_order WHERE user\_id = #{userId}<br> </select><br><br> <insert id="insert" parameterType="FlGoodsOrderModel"><br> INSERT INTO fl\_goods\_order (user\_id, goods\_id, quantity, total\_amount, status, order\_date, delivery\_date)<br> VALUES (#{userId}, #{goodsId}, #{quantity}, #{totalAmount}, #{status}, #{orderDate}, #{deliveryDate})<br> </insert><br><br> <update id="updateStatus" parameterType="map"><br> UPDATE fl\_goods\_order SET status = #{status} WHERE id = #{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_goods\_order WHERE id = #{id}<br> </delete><br><br> <select id="findByStatus" parameterType="String" resultType="FlGoodsOrderModel"><br> SELECT \* FROM fl\_goods\_order WHERE status = #{status}<br> </select><br></mapper> |
| **补充说明** | FlGoodsOrderMapper.xml 文件中定义了订单表的数据库交互 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAddressModel.java** |
| **功能描述** | 映射数据库 fl\_address 表，用于管理用户地址信息。 |
| **接口与属性** | java<br>public class FlAddressModel {<br> private Long id; // 地址ID<br> private Long userId; // 用户ID<br> private String recipientName; // 收件人姓名<br> private String phone; // 联系电话<br> private String addressLine; // 地址详情<br> private String city; // 城市<br> private String state; // 省份<br> private String postalCode; // 邮政编码<br> private boolean isDefault; // 是否默认地址<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public Long getUserId() { return userId; }<br> public void setUserId(Long userId) { this.userId = userId; }<br><br> public String getRecipientName() { return recipientName; }<br> public void setRecipientName(String recipientName) { this.recipientName = recipientName; }<br><br> public String getPhone() { return phone; }<br> public void setPhone(String phone) { this.phone = phone; }<br><br> public String getAddressLine() { return addressLine; }<br> public void setAddressLine(String addressLine) { this.addressLine = addressLine; }<br><br> public String getCity() { return city; }<br> public void setCity(String city) { this.city = city; }<br><br> public String getState() { return state; }<br> public void setState(String state) { this.state = state; }<br><br> public String getPostalCode() { return postalCode; }<br> public void setPostalCode(String postalCode) { this.postalCode = postalCode; }<br><br> public boolean isDefault() { return isDefault; }<br> public void setDefault(boolean isDefault) { this.isDefault = isDefault; }<br>} |
| **数据结构** | FlAddressModel 字段与数据库表 fl\_address 一一对应。 |
| **补充说明** | 该模型类用于用户地址信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAddressService.java** |
| **功能描述** | 提供用户地址的服务接口，定义地址管理的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlAddressService {<br> FlAddressModel getById(Long id);<br> List<FlAddressModel> getByUserId(Long userId);<br> void addAddress(FlAddressModel address);<br> void updateAddress(FlAddressModel address);<br> void deleteAddress(Long id);<br> FlAddressModel getDefaultAddress(Long userId);<br>} |
| **补充说明** | FlAddressService 定义了用户地址的 CRUD、按用户ID查询、删除及获取默认地址的方法。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAddressServiceImpl.java** |
| **功能描述** | 实现 FlAddressService 接口，处理用户地址的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlAddressServiceImpl implements FlAddressService {<br> @Autowired<br> private FlAddressMapper flAddressMapper;<br><br> @Override<br> public FlAddressModel getById(Long id) {<br> return flAddressMapper.findById(id);<br> }<br><br> @Override<br> public List<FlAddressModel> getByUserId(Long userId) {<br> return flAddressMapper.findByUserId(userId);<br> }<br><br> @Override<br> public void addAddress(FlAddressModel address) {<br> flAddressMapper.insert(address);<br> }<br><br> @Override<br> public void updateAddress(FlAddressModel address) {<br> flAddressMapper.update(address);<br> }<br><br> @Override<br> public void deleteAddress(Long id) {<br> flAddressMapper.delete(id);<br> }<br><br> @Override<br> public FlAddressModel getDefaultAddress(Long userId) {<br> return flAddressMapper.findDefaultByUserId(userId);<br> }<br>} |
| **补充说明** | FlAddressServiceImpl 使用 FlAddressMapper 访问数据库，完成用户地址业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlAddressMapper.java** |
| **功能描述** | 用户地址信息的数据库映射接口，定义与 fl\_address 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlAddressMapper {<br> FlAddressModel findById(Long id);<br> List<FlAddressModel> findByUserId(Long userId);<br> void insert(FlAddressModel address);<br> void update(FlAddressModel address);<br> void delete(Long id);<br> FlAddressModel findDefaultByUserId(Long userId);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlAddressMapper.xml** |
| **功能描述** | 定义与 FlAddressMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlAddressMapper"><br> <select id="findById" parameterType="Long" resultType="FlAddressModel"><br> SELECT \* FROM fl\_address WHERE id = #{id}<br> </select><br><br> <select id="findByUserId" parameterType="Long" resultType="FlAddressModel"><br> SELECT \* FROM fl\_address WHERE user\_id = #{userId}<br> </select><br><br> <insert id="insert" parameterType="FlAddressModel"><br> INSERT INTO fl\_address (user\_id, recipient\_name, phone, address\_line, city, state, postal\_code, is\_default)<br> VALUES (#{userId}, #{recipientName}, #{phone}, #{addressLine}, #{city}, #{state}, #{postalCode}, #{isDefault})<br> </insert><br><br> <update id="update" parameterType="FlAddressModel"><br> UPDATE fl\_address SET recipient\_name = #{recipientName}, phone = #{phone}, address\_line = #{addressLine},<br> city = #{city}, state = #{state}, postal\_code = #{postalCode}, is\_default = #{isDefault}<br> WHERE id = #{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_address WHERE id = #{id}<br> </delete><br><br> <select id="findDefaultByUserId" parameterType="Long" resultType="FlAddressModel"><br> SELECT \* FROM fl\_address WHERE user\_id = #{userId} AND is\_default = true LIMIT 1<br> </select><br></mapper> |
| **补充说明** | FlAddressMapper.xml 文件中定义了地址表的数据库交互 SQL。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlVarietyModel.java** |
| **功能描述** | 映射数据库 fl\_varietys 表，用于管理商品种类树信息。 |
| **接口与属性** | java<br>public class FlVarietyModel {<br> private Long id; // 种类ID<br> private String name; // 种类名称<br> private Long parentId; // 父种类ID<br> private int level; // 种类层级<br> private boolean isLeaf; // 是否叶子节点<br><br> // getter & setter 方法<br> public Long getId() { return id; }<br> public void setId(Long id) { this.id = id; }<br><br> public String getName() { return name; }<br> public void setName(String name) { this.name = name; }<br><br> public Long getParentId() { return parentId; }<br> public void setParentId(Long parentId) { this.parentId = parentId; }<br><br> public int getLevel() { return level; }<br> public void setLevel(int level) { this.level = level; }<br><br> public boolean isLeaf() { return isLeaf; }<br> public void setLeaf(boolean isLeaf) { this.isLeaf = isLeaf; }<br>} |
| **数据结构** | FlVarietyModel 字段与数据库表 fl\_varietys 一一对应。 |
| **补充说明** | 该模型类用于种类树信息的存取操作，以便在业务层进行 CRUD 操作。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlVarietyService.java** |
| **功能描述** | 提供商品种类的服务接口，定义种类树的业务逻辑方法。 |
| **接口与方法** | java<br>public interface FlVarietyService {<br> FlVarietyModel getById(Long id);<br> List<FlVarietyModel> getByParentId(Long parentId);<br> void addVariety(FlVarietyModel variety);<br> void updateVariety(FlVarietyModel variety);<br> void deleteVariety(Long id);<br>} |
| **补充说明** | FlVarietyService 定义了种类树的 CRUD、按父ID查询等方法。 |

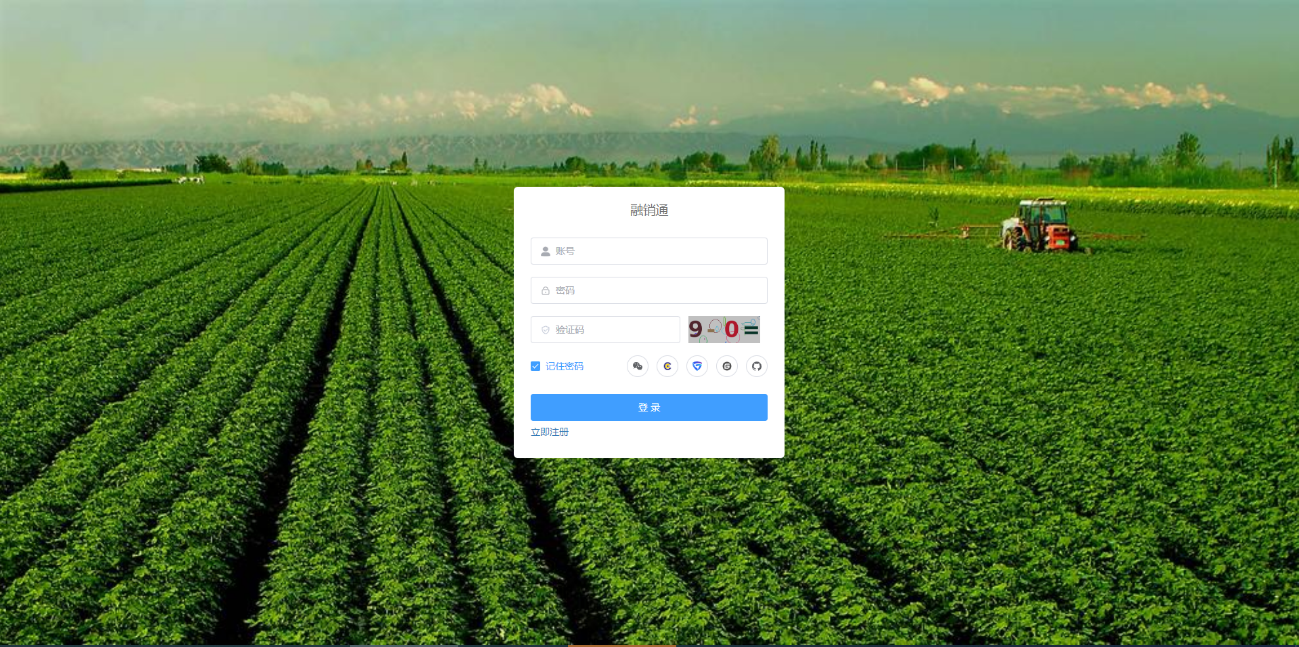
|  |  |
| --- | --- |
| **功能类名称** | **FlVarietyServiceImpl.java** |
| **功能描述** | 实现 FlVarietyService 接口，处理商品种类的业务逻辑。 |
| **接口与方法** | java<br>@Service<br>public class FlVarietyServiceImpl implements FlVarietyService {<br> @Autowired<br> private FlVarietyMapper flVarietyMapper;<br><br> @Override<br> public FlVarietyModel getById(Long id) {<br> return flVarietyMapper.findById(id);<br> }<br><br> @Override<br> public List<FlVarietyModel> getByParentId(Long parentId) {<br> return flVarietyMapper.findByParentId(parentId);<br> }<br><br> @Override<br> public void addVariety(FlVarietyModel variety) {<br> flVarietyMapper.insert(variety);<br> }<br><br> @Override<br> public void updateVariety(FlVarietyModel variety) {<br> flVarietyMapper.update(variety);<br> }<br><br> @Override<br> public void deleteVariety(Long id) {<br> flVarietyMapper.delete(id);<br> }<br>} |
| **补充说明** | FlVarietyServiceImpl 使用 FlVarietyMapper 访问数据库，完成种类树业务逻辑的实现。 |

|  |  |
| --- | --- |
| **功能类名称** | **FlVarietyMapper.java** |
| **功能描述** | 商品种类信息的数据库映射接口，定义与 fl\_varietys 表的数据库操作方法。 |
| **接口与方法** | java<br>@Mapper<br>public interface FlVarietyMapper {<br> FlVarietyModel findById(Long id);<br> List<FlVarietyModel> findByParentId(Long parentId);<br> void insert(FlVarietyModel variety);<br> void update(FlVarietyModel variety);<br> void delete(Long id);<br>} |
| **补充说明** | 接口方法在 Mapper.xml 文件中定义具体 SQL 查询。 |

|  |  |
| --- | --- |
| **文件名称** | **FlVarietyMapper.xml** |
| **功能描述** | 定义与 FlVarietyMapper 接口关联的 SQL 语句。 |
| **SQL 配置** | xml<br><?xml version="1.0" encoding="UTF-8" ?><br><!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd"><br><br><mapper namespace="com.example.mapper.FlVarietyMapper"><br> <select id="findById" parameterType="Long" resultType="FlVarietyModel"><br> SELECT \* FROM fl\_varietys WHERE id = #{id}<br> </select><br><br> <select id="findByParentId" parameterType="Long" resultType="FlVarietyModel"><br> SELECT \* FROM fl\_varietys WHERE parent\_id = #{parentId}<br> </select><br><br> <insert id="insert" parameterType="FlVarietyModel"><br> INSERT INTO fl\_varietys (name, parent\_id, level, is\_leaf)<br> VALUES (#{name}, #{parentId}, #{level}, #{isLeaf})<br> </insert><br><br> <update id="update" parameterType="FlVarietyModel"><br> UPDATE fl\_varietys SET name = #{name}, parent\_id = #{parentId}, level = #{level}, is\_leaf = #{isLeaf}<br> WHERE id = #{id}<br> </update><br><br> <delete id="delete" parameterType="Long"><br> DELETE FROM fl\_varietys WHERE id = #{id}<br> </delete><br></mapper> |
| **补充说明** | FlVarietyMapper.xml 文件中定义了种类树表的数据库交互 SQL。 |

## 3.2 模块A界面设计

登陆页面如图所示，登录时，用户需要输入用户名、密码、验证码、勾选记住密码（可选操作），并单击“登录”按钮，完成登录过程，也可以通过点击“立即注册”超链接跳转到注册页面进行注册操作。



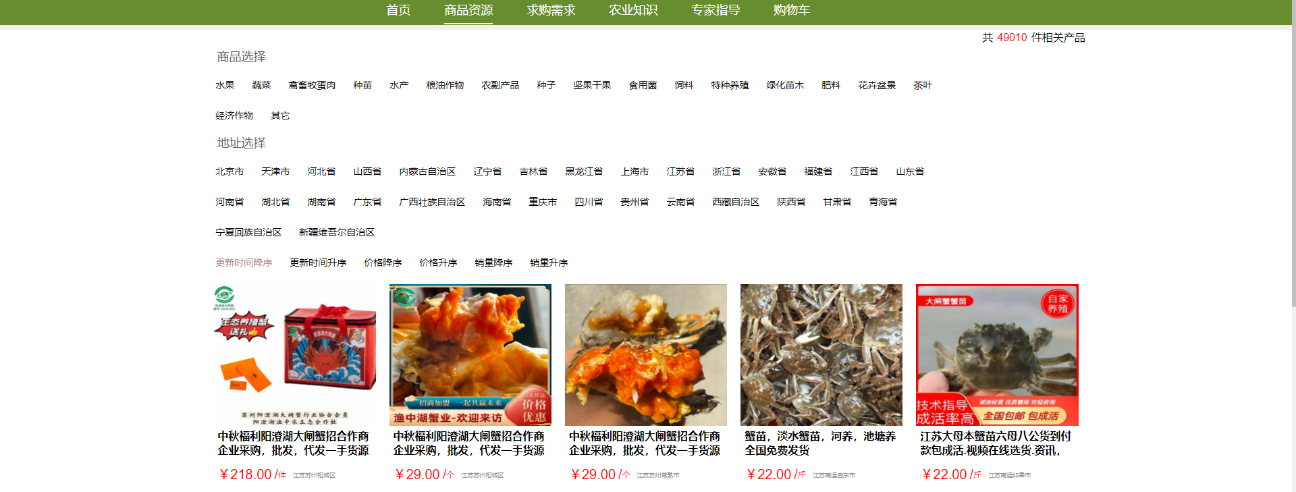
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **页面ID** | **login页面** | | **用例编号** | HR\_register\_001 |
| **编号** | **控件** | **控件类型** | **类型名称** | **控件的功能描述** |
| 1 | 账户 | Text | 文本框 | \*用户在这里输入账户 |
| 2 | 密码 | Password | 密码框 | \*用户在这里输入密码 |
| 3 | 验证码图片 | img | 图片 | 用户获取通过点击验证码图片进行刷新的验证码图案 |
| 4 | 验证码 | Text | 文本框 | \*用户在这里输入验证码 |
| 5 | 记住密码 | CheckBox | 单选框 | \*用户选中后，可以记住密码 |
| 6 | 立即注册 | HyperLink | 超链接 | \*用户点击，进入注册用户模式 |
| 7 | 第三方登录图标 | HyperLink | 超链接 | \*用户点击第三方图标，可以使用第三方账户登录 |
| 8 | 登录 | Submit | 按钮 | \*用户点击登录按钮，完成登录 |

用户页面如图所示，进入用户页面时，用户通过点击菜单栏的不同选项，快速跳转到有需要的页面，在“首页”页面，用户可以查看商品名称，价格以及商品图片，用户也可以通过筛选栏，进行商品选择、地址选择和排序方式进行条件筛选，快速筛选出需要的商品，点击对应商品是，可以查看商品的详情；在“商品资源”页面时也是同样操作

**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **页面ID** | **用户页面** | | **用例编号** | HR\_register\_001 |
| **编号** | **控件** | **控件类型** | **类型名称** | **控件的功能描述** |
| 1 | 首页 | Link（超链接） | 导航栏超链接 | \*用户可以在这里选择对应的页面 |
| 2 | 商品选择 | List | 选择列表 | \*用户在这里商品名称条件 |
| 3 | 地址选择 | List | 选择列表 | \*用户在这里地址条件 |
| 4 | 排序 | Link（超链接） | 超链接 | \*用户在这里选择排序规则 |
| 5 | 商品图标 | FontAwesome | 图标控件 | \*用户可以查看商品的描述、内容以及价格 |
| 6 | 立即注册 | HyperLink | 超链接 | \*用户点击，进入注册用户模式 |
| 7 | 商品资源 | HyperLink | 超链接 | \*用户切换到“商品资源”页面 |
| 8 | 求购需求 | HyperLink | 超链接 | \*用户切换到“求购需求”页面 |
| 9 | 农业知识 | HyperLink | 超链接 | \*用户切换到“农业知识”页面 |
| 10 | 专家指导 | HyperLink | 超链接 | \*用户切换到“专家指导”页面 |
| 11 | 购物车 | HyperLink | 超链接 | \*用户切换到“购物车”页面 |

商品资源页面如图所示，处于商品资源页面时，用户通过点击菜单栏的不同选项，快速跳转到有需要的页面，在“商品资源”页面，用户可以查看商品名称，价格以及商品图片，用户也可以通过筛选栏，进行商品选择、地址选择和排序方式进行条件筛选，快速筛选出需要的商品，点击对应商品是，可以查看商品的详情；在“商品资源”页面时也是同样操作



|  |  |  |  |
| --- | --- | --- | --- |
| **页面ID** | **商品资源页面** | **用例编号** | HR\_register\_001 |
| **编号** | **控件** | **控件类型** | **类型名称** | **控件的功能描述** |
| 1 | 菜单栏 | Link（超链接） | 导航栏超链接 | \*用户可以在这里选择对应的页面 |
| 2 | 商品选择 | List | 选择列表 | \*用户在这里商品名称条件 |
| 3 | 地址选择 | List | 选择列表 | \*用户在这里地址条件 |
| 4 | 排序 | Link（超链接） | 超链接 | \*用户在这里选择排序规则 |
| 5 | 商品图标 | FontAwesome | 图标控件 | \*用户可以查看商品的描述、内容以及价格 |
| 6 | 立即注册 | HyperLink | 超链接 | \*用户点击，进入注册用户模式 |
| 7 | 商品资源 | HyperLink | 超链接 | \*用户切换到“商品资源”页面 |
| 8 | 求购需求 | HyperLink | 超链接 | \*用户切换到“求购需求”页面 |
| 9 | 农业知识 | HyperLink | 超链接 | \*用户切换到“农业知识”页面 |
| 10 | 专家指导 | HyperLink | 超链接 | \*用户切换到“专家指导”页面 |
| 11 | 购物车 | HyperLink | 超链接 | \*用户切换到“购物车”页面 |
| 12 | 首页 | HyperLink | 超链接 | \*用户切换到“首页”页面 |

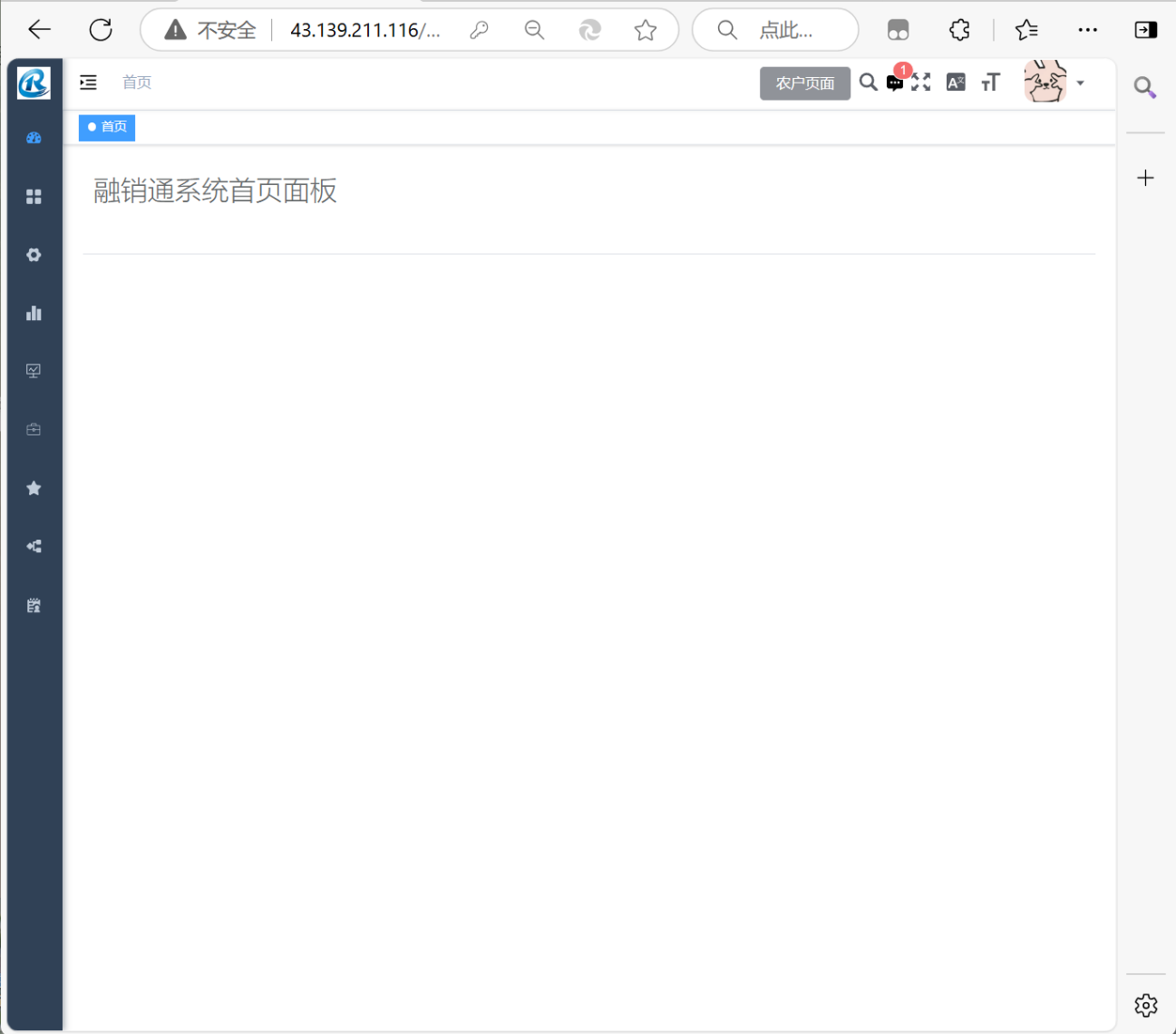
# 4.子功能模块B的功能类设计

## 4.1 模块B功能设计

|  |  |
| --- | --- |
| 功能类名称 |  |
| 功能描述 |  |
| 接口与属性 | **提示：**用专业的设计（开发）工具来设计本模块的接口与属性，说明函数功能、输入参数、输出参数、返回值等。此处粘贴即可。 |
| 数据结构  与算法 | **提示：**不论是采用经典的还是专用的数据结构与算法，都应该作必要的描述。不仅用于指导程序的实现，还可以让人们清楚地了解该对象类是如何设计的。 |
| 补充说明 |  |

## 4.2 模块B功能设计

后台界面如图所示，用户可以点击左边的菜单栏导航到对应的功能，可以点击“农户页面”按钮切换到农户页面，点击搜索按钮可以进行搜索，点击消息按钮可以进行消息查看，点击放大按钮可以对页面进行放大，点击头像图标可以进行个人信息管理



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **页面ID** | **后台页面** | | **用例编号** | HR\_register\_001 |
| **编号** | **控件** | **控件类型** | **类型名称** | **控件的功能描述** |
| 1 | 导航栏 | List | 导航栏列表 | \*用户可以在这里选择对应的列表 |
| 2 | 导航栏收展按钮 | Button | 按钮 | \*用户点击此选项对菜单栏进行展开和收藏切换 |
| 3 | 农户页面按钮 | Button | 按钮 | \*用户点击按钮进行切换到农户页面 |
| 4 | 搜索按钮 | Button | 按钮 | \*用户点击搜索按钮搜索对应的功能 |
| 5 | 消息查看按钮 | Button | 按钮 | \*用户可以点击消息查看按钮对消息进行查看 |
| 6 | 图标控件 | FontAwesome | 图标控件 | \*用户点击图标控件对个人信息进行管理 |

# 5. 其他