# 《数据库系统原理课程设计》

**课题名称： 闲置物品交易平台**

课题负责人名（学号）：

同组成员名单（角色）：

指导教师：

**评阅成绩：**

评阅意见：

提交报告时间：2023年 12 月 27 日

**闲置物品交易平台**

计算机科学与技术

**学生** xx **指导老师** xxx

**[摘要]** 这项项目旨在开发一个大学生闲置物品交易平台，解决学生在离校和毕业时面临的大量无法带走的闲置物品问题。我们的团队将通过该项目运用在课堂上学习的数据库知识，提高对数据库设计和操作的理解和技能。项目主要采用IDEA, MySQL, Git, Maven, Spring Boot和Layui等工具和框架进行开发。我们的技术路线遵循MVC理念，明确划分业务逻辑层、数据访问层和表示层，以实现高度的代码模块化和清晰的项目结构。项目以解决现实问题为出发点，采用合理的技术路线，旨在为用户提供有效的解决方案。

**关键词**：闲置物品交易，Spring Boot，数据库， Mybatis

**一、绪论**

* 1. **开发背景**

**在大学生活中，许多学生在离开学校和毕业时都会遇到有大量物品无法带走的情况，这些物品多是书籍、家电和生活用品等。虽然这些闲置物品对于毕业生来说已无太大用途，但对于其他在校大学生却有着较高的实用价值。然而，目前缺乏一个有效的平台让这些闲置物品得到合适的流转。因此，本项目旨在开发一个大学生闲置物品交易平台，利用网络的广泛性，方便大学生之间的二手物品交易。**

**此外，我们的团队在课堂上学习了数据库的相关知识，通过这次项目，也希望能实践和应用这些理论知识，以增强我们对于数据库设计和操作的理解和技能。**

* 1. **开发环境**

本项目的开发环境如下：

* 集成开发工具：IDEA
* 数据库管理系统：MySQL
* 版本控制：Git
* 构建工具：Maven
* 后端开发框架：Spring Boot
* 前端开发框架： Layui

项目包含实体、控制器、数据访问对象、配置类、实用程序类等主要的典型组成部分。项目还包含有用于映射数据库表的mapper文件、静态文件存储、动态页面生成，以及包含项目配置信息的application.yml文件。

* 1. **技术路线**

本项目的技术路线如下：

使用Spring Boot框架进行后端开发，采用MVC理念，将业务逻辑层、数据访问层和表示层分离，提高了代码的模块化程度，使得项目结构更为清晰。

数据存储方面，选择MySQL作为数据库管理系统，保证数据的结构化存储，并能处理大量数据的存储和检索。

使用Mybatis作为持久层框架，通过配置映射文件，直接操作数据库，简化了数据访问代码的编写，提高了开发效率。

项目采用Maven进行项目管理，包括项目结构的定义、依赖库管理及项目构建等。

整体来看，项目立足于解析现实生活中的问题，采用合理的技术路线，有效解决用户的需求。

**二、概要设计**

**2.1需求概述**

主要输入：

* 用户注册：需要用户名、密码、邮箱、性别等基本信息。

登录的识别信息：用户名和密码。

* 添加商品：输入商品名称，描述，图片，商品种类，商品价格，以及所属用户的用户id。
* 发布留言：输入留言内容，以及对应的用户id和商品id。
* 执行购物车操作：需要用户的id，商品的id。
* 创建订单：需要用户发送收件人姓名，收件人地址，收件人电话，以及所购买商品的id。

主要输出：

各项功能操作中，涉及的数据表内容－比如用户的登录信息，添加的商品信息，新增的留言，购物车的商品，订单状态等等。

处理的性能要求：

由于这是一个校园级别的线上商城，主要用户群体为校园内的学生和教职工，因此，我们的系统需要适应校园网络环境和使用高峰期。虽然用户量相对全国化或全球化的电商网站要小很多，但是我依然希望系统可以24小时运行，并在学生大量使用的高峰期（例如晚上和周末）保持良好的响应速度，为用户提供流畅的购物体验。

系统主要面板：

* 用户面板：用户可以查看和编辑个人信息，也可以进行商品的购买，查看自己的订单，以及和其他用户的留言。
* 商品面板：所有上架商品的信息都能在这里找到，包括商品的基本信息，以及用户的留言。
* 留言面板：方便用户留言和查看留言。

购物车面板：用户可以将选中商品加入购物车，也可以查看购物车内所有商品。

* 订单面板：用户可以在这里查看所有的订单信息，包括订单状态，配送信息等。

系统要求：

系统需要支持多种浏览器，支持主流的操作系统如Windows 7、Windows 10、MacOS以及Linux等。同时，需要服务器端安装有MySQL 8.0用于数据存储。

2.3 基本概念

实体：

用户(User)：参与此系统的人，负责购买商品。

商品(Item): 在平台上出售的产品。

订单(Order)：由用户创建，包含他们所购买商品的信息。

购物车(Cart)：临时存储用户想购买的商品，等待结账。

留言(Message)：用户之间通过留言交流，对商品的评价和询问。

关系：

用户(User)与商品(Item)：用户浏览商品，选择商品加入购物车或直接购买生产订单，也可以对商品进行留言。

用户(User)与购物车(Cart)：用户通过添加或移除购物车中的商品操作购物车。

用户(User)与订单(Order)：用户通过购物车结账生成订单，也可以在订单中查看购买的商品信息，以及支付。

订单(Order)与商品(Item)：订单中包含用户所购买的商品信息。

用户(User)与留言(Message)：用户之间可以通过留言来交流信息，提出关于商品的问题或者分享他们的购物车。

处理流程：

普通用户：

1. 用户注册并登录系统。

2．登录后的用户可以浏览商品，用户可以点击类别选择感兴趣的商品进行浏览并将其添加至购物车。

3. 用户可以在购物车中查看所有已添加的商品，也可以移除其中的商品。

4. 用户通过购物车生成订单并支付，订单创建后将不能更改，在用户支付后订单完成。

5. 用户可以在商品页面或商品对应的留言板中发表留言，询问关于商品的问题或者分享购物心得。

6. 系统会对用户的行为进行跟踪和记录，比如浏览商品，生成订单，发表留言等。

管理员：

1. 管理员登录系统后，可以管理用户信息，包括但不限于查看、修改和删除用户信息。

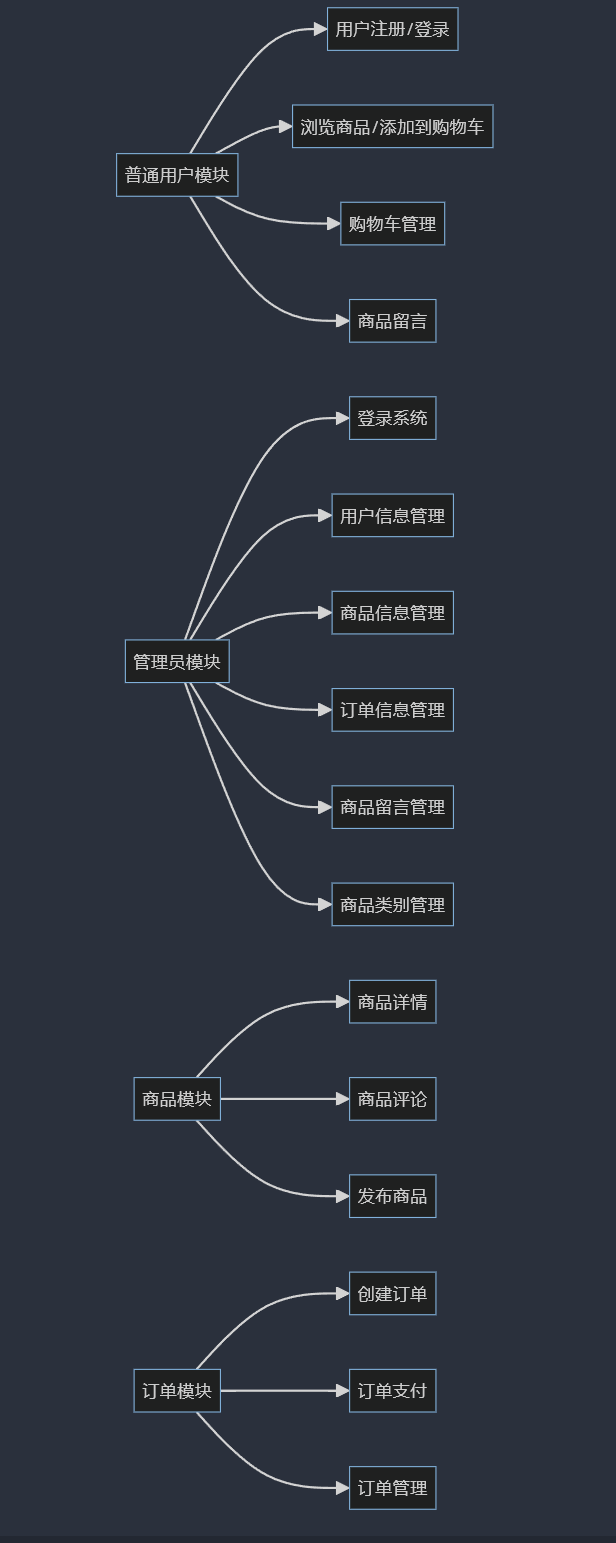
2．管理员可以管理商品信息，包括添加新的商品，修改或删除已有的商品信息。

3. 管理员可以查看所有的订单信息，并对订单进行管理（比如确认订单，取消订单等）。

4. 管理员可以查看、删除用户在商品页面或商品对应的留言板上发表的留言。

5. 管理员可以管理商品的分类，包括添加新的分类，修改或删除已有的分类。

2.2 系统功能模块图



三、详细设计

3.1 UI设计



**表1：登录UI界面**

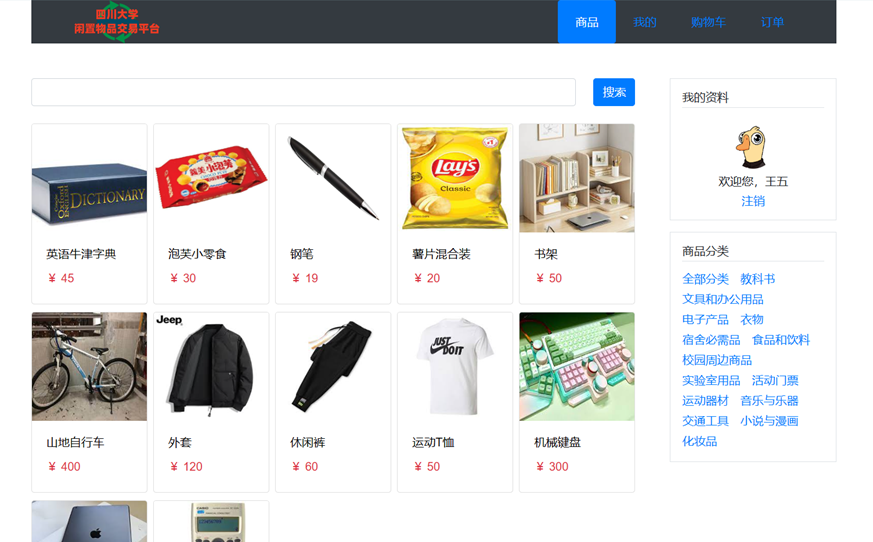
| 界面构件 | 变量 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 文本输入框 | email | login() | ‘/loginAction’ | 接受用户输入的邮箱 |
| 文本输入框 | password | login() | ‘/loginAction’ | 接受用户输入的密码 |
| 登录按钮 | - | login() | ‘/loginAction’ | 提交登录请求，并根据返回信息，页面跳转或弹出警告 |
| 注册按钮 | - | - | ‘/register’ | 跳转至注册页面 |



**表2：注册UI界面**

| 界面构件 | 变量 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 文本输入框 | email | register() | ‘/registerAction’ | 接受用户输入的邮箱 |
| 密码输入框 | password | register() | ‘/registerAction’ | 接受用户输入的密码 |
| 密码确认框 | password2 | register() | ‘/registerAction’ | 验证用户两次输入的密码一致 |
| 文本输入框 | name | register() | ‘/registerAction’ | 接受用户输入的真实姓名 |
| 下拉选择框 | sex | register() | ‘/registerAction’ | 接受用户选择的性别 |
| 注册按钮 | - | register() | ‘/registerAction’ | 提交注册请求，并根据返回信息，进行页面跳转或弹出警告 |

**商城界面预览**



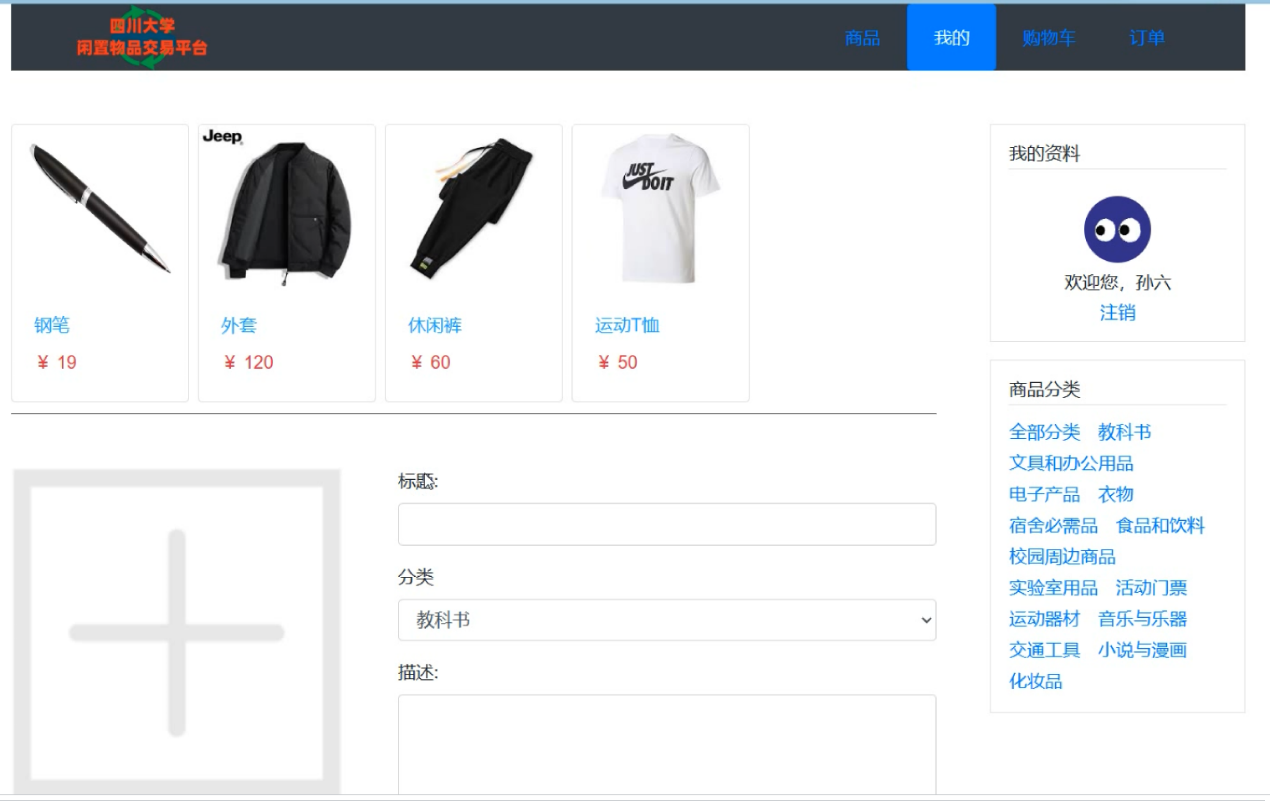
**表3：商品UI界面**

| 界面构件 | 变量 | 函 数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 文本输入框 | search | search() | ‘/searchAction’ | 接受用户输入的关键字进行搜索 |
| 搜索按钮 | - | search() | ‘/searchAction’ | 根据用户输入的关键字，提交搜索请求 |
| 商品项 | good | load() | ‘/good/details?id=’+good.getId() | 显示商品详细信息 ，点击后跳转至商品详情页 |
| 商品图片 | good | load() | good.getImg() | 显示商品图片 |
| 商品标题 | good | load() | good.getTitle() | 显示商品的标题 |
| 商品价格 | good | load() | good.getPrice() | 显示商品的价格 |



**表5：商品详情UI界面**

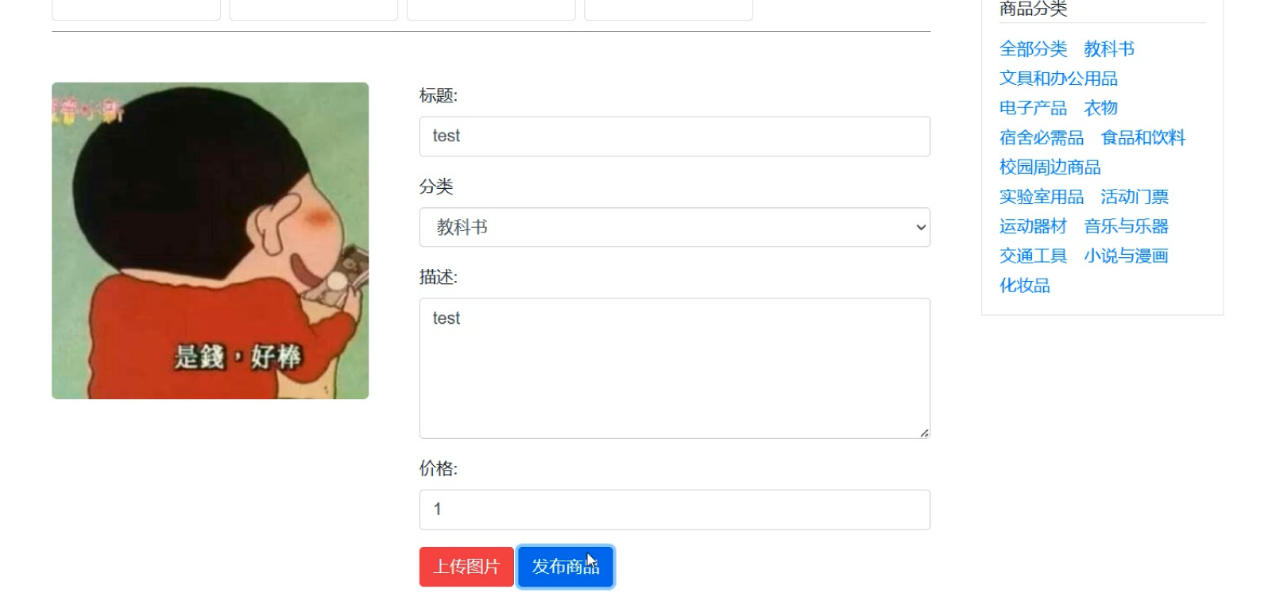
| 界面构件 | 变量 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 文本显示框 | product | - | - | 显示商品名称 |
| 文本显示框 | price | - | - | 显示商品的价格 |
| 按钮 | - | addToCart() | ‘/add-to-cart’ | 将商品添加至购物车 |
| 按钮 | - | removeFromCart() | ‘/remove-from-cart’ | 将商品从购物车中移除 |
| 按钮 | - | buyNow() | ‘/buy-now’ | 跳转至购买页面 |
| 按钮 | - | deleteUserReview() | ‘/delete-user-review’ | 删除用户的评论 |
| 文本输入框 | userReview | leaveReview() | ‘/leave-review’ | 输入用户的评论 |





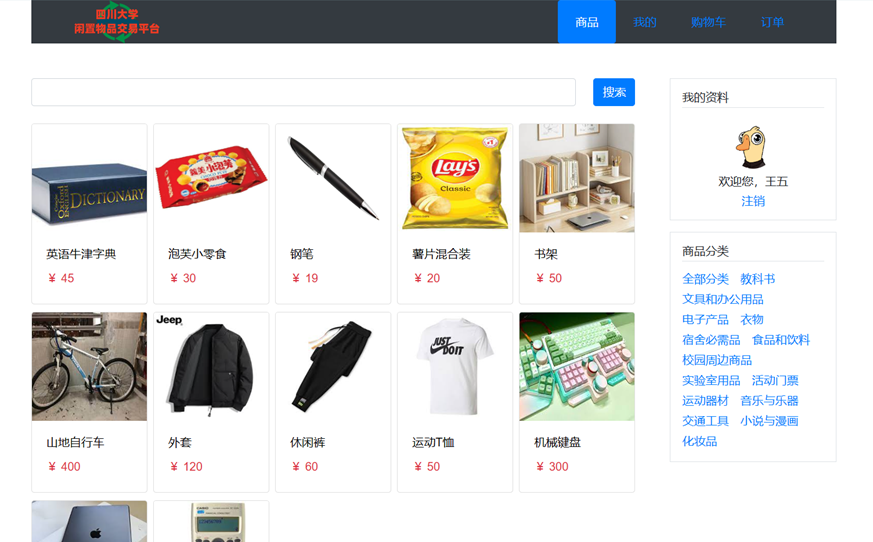
**表8：订单信息UI界面**

| 界面构件 | 变量 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 图片显示框 | order | - | ‘{order.getTitle()}’ | 显示订单中商品的名称 |
| 文本显示框 | order | - | ‘{order.getTakeName()}’ | 显示收件人的姓名 |
| 文本显示框 | order | - | ‘{order.getTakeAddress()}’ | 显示收件人的地址 |
| 文本显示框 | order | - | ‘{order.getExpressNumber()}’ | 显示快递单号 |
| 按钮 | - | - | ‘/order/pay?orderId=’+order.getId() | 确认订单，货到付款 |
| 按钮 | - | - | - | 确认发货 |
| 按钮 | - | - | ‘/order/receipt?orderId=’+order.getId() | 确认收货 |
| 按钮 | - | - | ‘/order/cancel?orderId=’+order.getId() | 取消订单 |



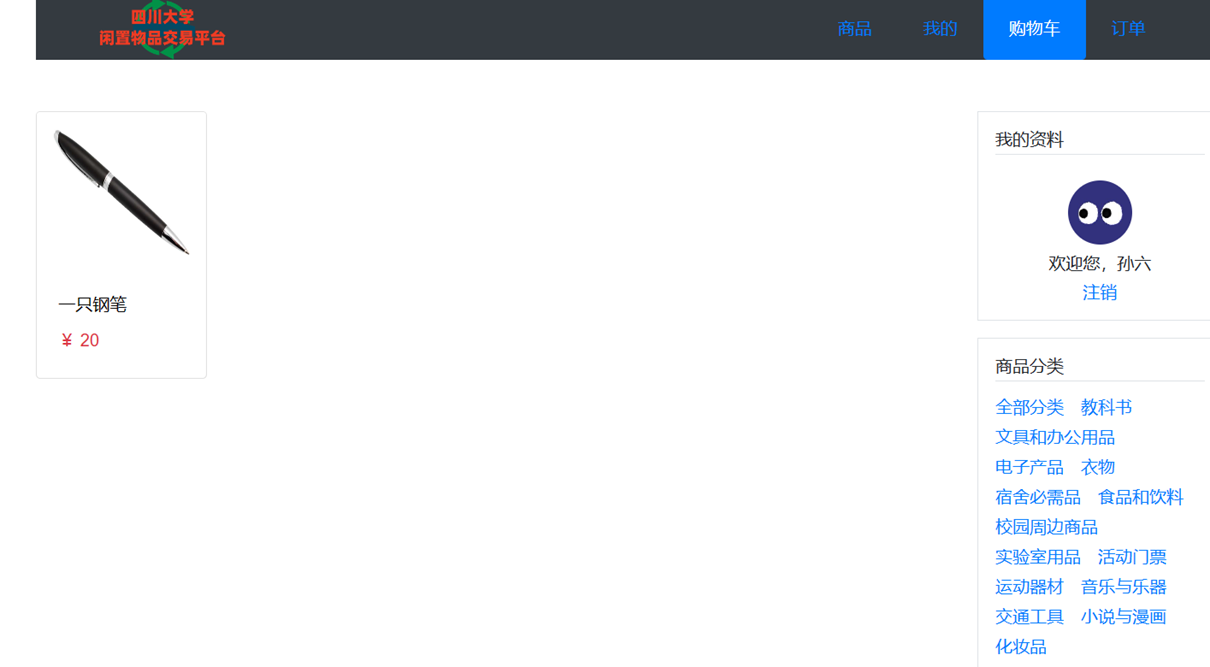
**表7：售卖商品UI界面**

| 界面构件 | 变量 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 图片显示框 | good | update() | ‘{good.getTitle()}’ | 显示商品的标题 |
| 文本显示框 | good | - | ‘{category.getName()}’, ‘${category.getId()}’ | 选择商品的类别 |
| 文本区域输入框 | describe | - | - | 输入商品的描述 |
| 文本输入框 | price | - | - | 输入商品的价格 |
| 按钮 | - | upload() | - | 上传商品的图片 |
| 提交按钮 | - | - | ‘/sale/publish’ | 发布商品信息 |



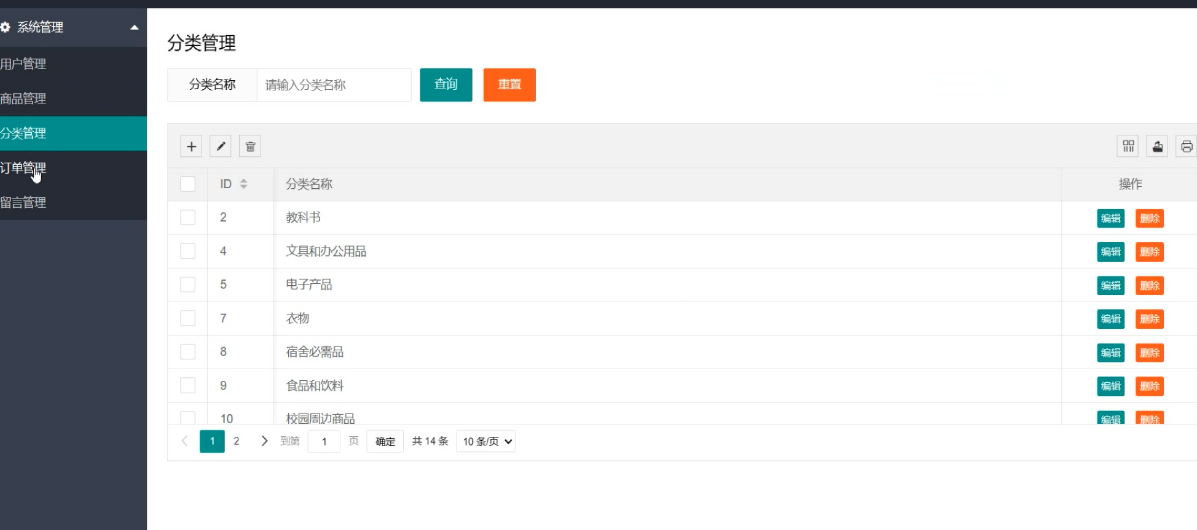
**表9：公共布局页面UI界面**

| 界面构件 | 使用API | 功能描述 |
| --- | --- | --- |
| 图像 | ‘/upload/img/logo.png’ | 显示网站的Logo |
| 链接 | ‘/good/’ | 跳转到商品页面 |
| 链接 | ‘/sale/’ | 跳转到我的页面 |
| 链接 | ‘/car/’ | 跳转到购物车页面 |
| 链接 | ‘/order/’ | 跳转到订单页面 |
| 代码块 | ‘location.pathname’ | 确定当前活动的导航链接 |



**表10：购物车UI界面**

| 界面构件 | 变量 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- | --- |
| 图片显示框 | good | - | ‘{good.getTitle()}’ | 显示商品的名称 |
| 文本显示框 | good | - | ‘${good.getPrice()}’ | 显示商品的价格 |
| 代码块 | - | row()/num() | - | 根据购物车中商品数量动态调整显示格式 |



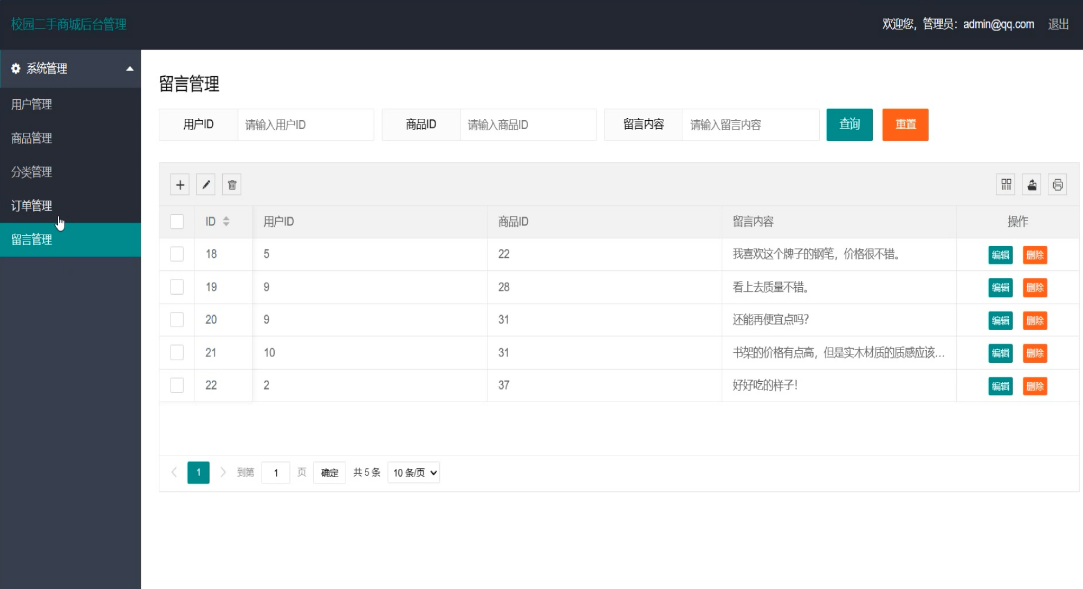
**表11：分类管理UI界面**

| 界面构件 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- |
| 输入框 | - | name | 输入分类名称进行查询 |
| 按钮 | 提交查询 | - | 提交查询请求 |
| 按钮 | 重置 | - | 重置查询输入 |
| 数据表 | - | ‘/category/query’ | 显示分类查询结果 |
| 输入框 | - | name | 输入分类名称进行更新 |
| 按钮 | 提交更新 | ‘/category/update’ | 提交更新请求 |
| 按钮 | 重置 | - | 重置更新输入 |



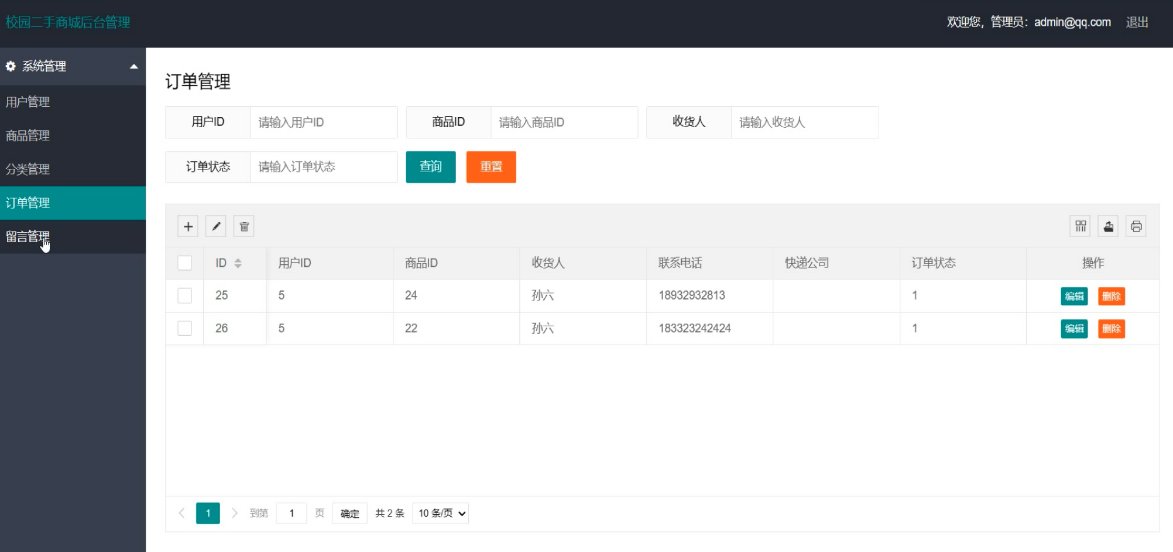
**表12：商品管理UI界面**

| 界面构件 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- |
| 输入框 | - | title | 输入商品名称进行查询 |
| 输入框 | - | userId | 输入用户ID进行查询 |
| 输入框 | - | categoryId | 输入分类ID进行查询 |
| 按钮 | 提交查询 | - | 提交查询请求 |
| 按钮 | 重置 | - | 重置查询输入 |
| 数据表 | - | ‘/good/query’ | 显示商品查询结果 |
| 输入框 | - | title | 输入商品名称进行更新 |
| 输入框 | - | userId | 输入用户ID进行更新 |
| 输入框 | - | categoryId | 输入分类ID进行更新 |
| 输入框 | - | price | 输入商品价格进行更新 |
| 输入框 | - | state | 输入商品状态进行更新 |
| 输入框 | - | describe | 输入商品描述进行更新 |
| 按钮 | 提交更新 | ‘/good/update’ | 提交更新请求 |
| 按钮 | 重置 | - | 重置更新输入 |



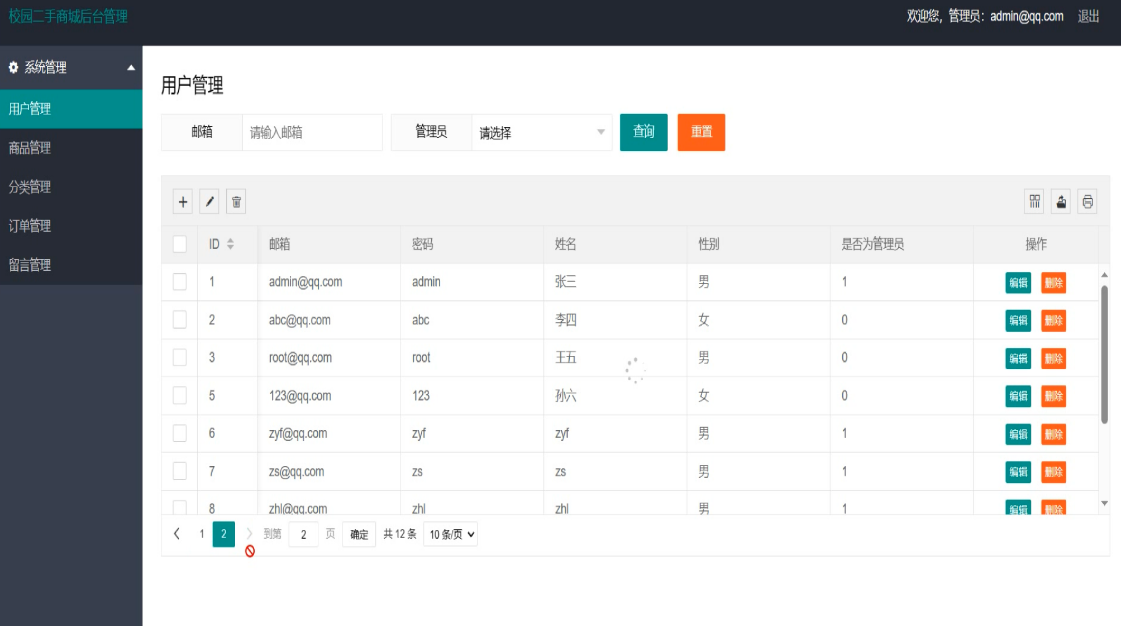
**表13：留言管理UI界面**

| 界面构件 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- |
| 输入框 | - | ‘userId’ | 输入用户ID进行查询 |
| 输入框 | - | ‘goodId’ | 输入商品ID进行查询 |
| 输入框 | - | ‘content’ | 输入留言内容进行查询 |
| 按钮 | 提交查询 | - | 提交查询请求 |
| 按钮 | 重置 | - | 重置查询输入 |
| 数据表 | - | ‘/message/query’ | 显示留言查询结果 |
| 输入框 | - | ‘userId’ | 输入用户ID进行更新 |
| 输入框 | - | ‘goodId’ | 输入商品ID进行更新 |
| 输入框 | - | ‘content’ | 输入留言内容进行更新 |
| 按钮 | 提交更新 | ‘/message/update’ | 提交更新请求 |
| 按钮 | 重置 | - | 重置更新输入 |



**表14：订单管理UI界面**

| 界面构件 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- |
| 输入框 | - | ‘userId’ | 输入用户ID进行查询 |
| 输入框 | - | ‘goodId’ | 输入商品ID进行查询 |
| 输入框 | - | ‘takeName’ | 输入收货人姓名进行查询 |
| 输入框 | - | ‘state’ | 输入订单状态进行查询 |
| 按钮 | 提交查询 | - | 提交查询请求 |
| 按钮 | 重置 | - | 重置查询输入 |
| 数据表 | - | ‘/order/query’ | 显示订单查询结果 |
| 输入框 | - | ‘userId’ | 输入用户ID进行更新 |
| 输入框 | - | ‘goodId’ | 输入商品ID进行更新 |
| 输入框 | - | ‘takeName’ | 输入收货人姓名进行更新 |
| 输入框 | - | ‘takeAddress’ | 输入收货地址进行更新 |
| 输入框 | - | ‘takePhone’ | 输入联系电话进行更新 |
| 输入框 | - | ‘expressName’ | 输入快递公司名称进行更新 |
| 输入框 | - | ‘expressNumber’ | 输入快递单号进行更新 |
| 输入框 | - | ‘state’ | 输入订单状态进行更新 |
| 按钮 | 提交更新 | ‘/order/update’ | 提交更新请求 |
| 按钮 | 重置 | - | 重置更新输入 |

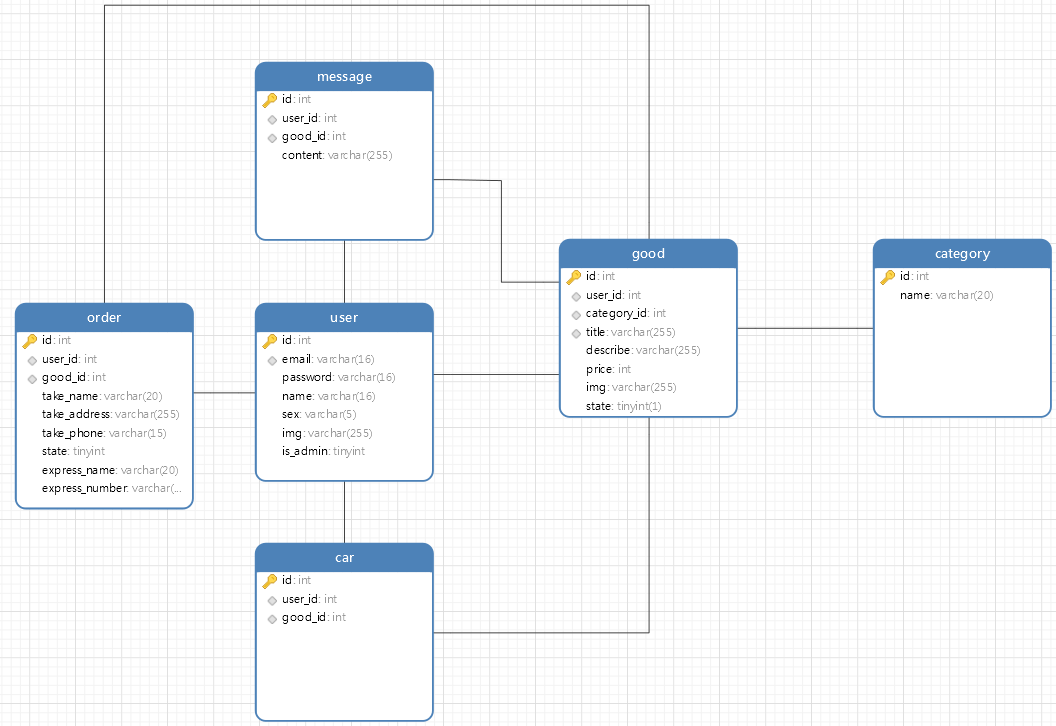


**表15：用户管理UI界面**

| 界面构件 | 函数 | 使用API | 功能描述 |
| --- | --- | --- | --- |
| 输入框 | - | ‘email’ | 输入邮箱进行查询 |
| 下拉选择框 | - | ‘isAdmin’ | 选择用户是否为管理员进行查询 |
| 按钮 | 提交查询 | - | 提交查询请求 |
| 按钮 | 重置 | - | 重置查询输入 |
| 数据表 | - | ‘/query’ | 显示用户查询结果 |
| 输入框 | - | ‘email’ | 输入邮箱进行更新 |
| 输入框 | - | ‘password’ | 输入密码进行更新 |
| 下拉选择框 | - | ‘isAdmin’ | 选择用户是否为管理员进行更新 |
| 按钮 | 提交更新 | ‘/user/update’ | 提交更新请求 |
| 按钮 | 重置 | - | 重置更新输入 |

#### 2 数据库设计

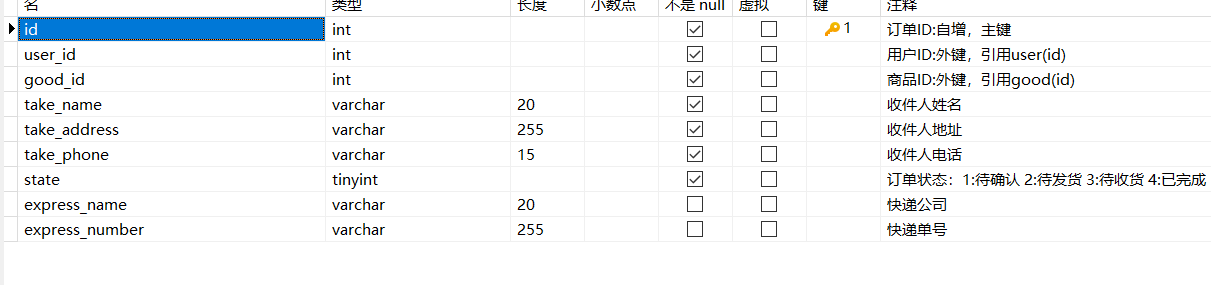
##### ER图



##### 数据库表结构

User表  


Order表



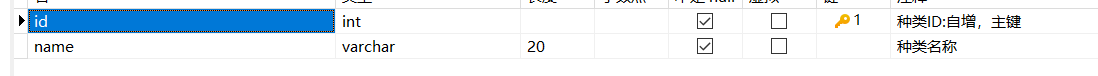
Message表



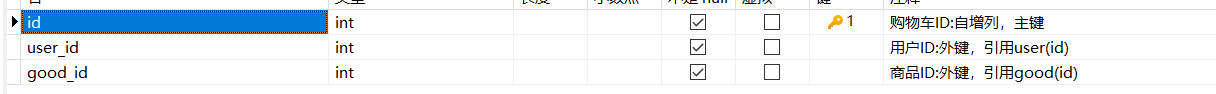
Good表



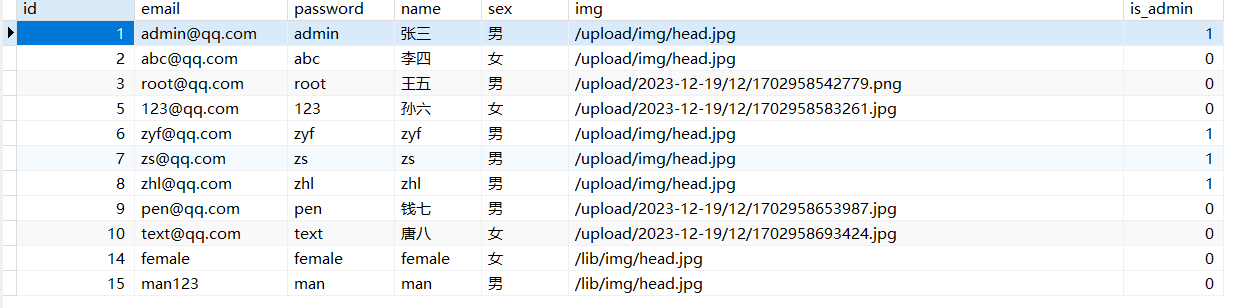
Category表



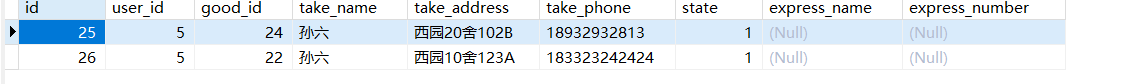
Car表



##### 数据库数据样例

User表  


Order表



Message表



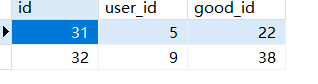
Good表



Category表



Car表



**表1：car表**

| 表项 | 类型及要求 | 备注 |
| --- | --- | --- |
| id | int NOT NULL AUTO\_INCREMENT | 购物车ID:自增列，主键 |
| user\_id | int NOT NULL | 用户ID:外键，引用user(id) |
| good\_id | int NOT NULL | 商品ID:外键，引用good(id) |

**表2：category表**

| 表项 | 类型及要求 | 备注 |
| --- | --- | --- |
| id | int NOT NULL AUTO\_INCREMENT | 种类ID:自增，主键 |
| name | varchar(20) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 种类名称 |

**表3：good表**

| 表项 | 类型及要求 | 备注 |
| --- | --- | --- |
| id | int NOT NULL AUTO\_INCREMENT | 商品ID：自增，主键 |
| user\_id | int NOT NULL | 用户ID:外键，引用user(id) |
| category\_id | int NOT NULL | 种类ID:外键，引用category（id） |
| title | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 商品名称 |
| describe | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 商品描述 |
| price | int NOT NULL | 商品价格 |
| img | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 商品图片 |
| state | tinyint(1) NOT NULL DEFAULT 1 | 商品状态：（1：未出售）（0：已出售） |

**表4：message表**

| 表项 | 类型及要求 | 备注 |
| --- | --- | --- |
| id | int NOT NULL AUTO\_INCREMENT | 留言ID:自增，主键 |
| user\_id | int NOT NULL | 用户ID:外键，引用user(id) |
| good\_id | int NOT NULL | 商品ID:外键，引用good(id) |
| content | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 留言内容 |

**表5：order表**

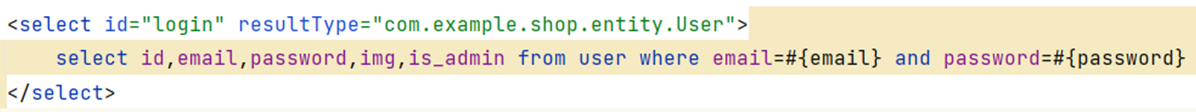
| 表项 | 类型及要求 | 备注 |
| --- | --- | --- |
| id | int NOT NULL AUTO\_INCREMENT | 订单ID:自增，主键 |
| user\_id | int NOT NULL | 用户ID:外键，引用user(id) |
| good\_id | int NOT NULL | 商品ID:外键，引用good(id) |
| take\_name | varchar(20) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 收件人姓名 |
| take\_address | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 收件人地址 |
| take\_phone | varchar(15) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 收件人电话 |
| state | tinyint NOT NULL DEFAULT 1 | 订单状态：1:待确认 2:待发货 3:待收货 4:已完成 |
| express\_name | varchar(20) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT NULL | 快递公司 |
| express\_number | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT NULL | 快递单号 |

**表6：user表**

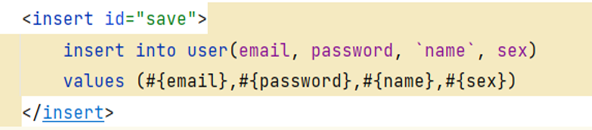
| 表项 | 类型及要求 | 备注 |
| --- | --- | --- |
| id | int NOT NULL AUTO\_INCREMENT | 用户ID:自增，主键 |
| email | varchar(16) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 用户邮箱 |
| password | varchar(16) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NOT NULL | 用户密码 |
| name | varchar(16) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT NULL | 用户姓名 |
| sex | varchar(5) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT NULL | 用户性别 |
| img | varchar(255) CHARACTER SET utf8mb3 COLLATE utf8mb3\_general\_ci NULL DEFAULT ‘/lib/img/head.jpg’ | 用户头像 |
| is\_admin | tinyint NOT NULL DEFAULT 0 | 是否为管理员（1：是，0：不是） |

##### Sql优化

###### 登录注册

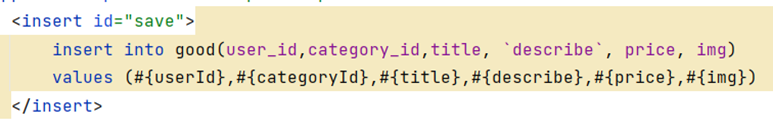


对于登录select语句，where中需要判断两个条件，但是不宜给password设计上索引，并且email不应该重复，所以对email字段添加了唯一的B+Tree索引。

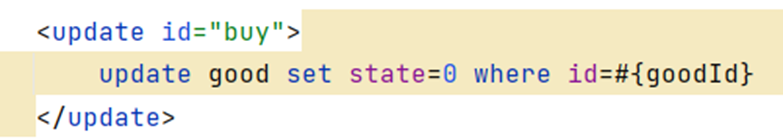


对于注册语句，因为user表中有自增主键，因此插入数据并不会造成底层数据移动，只会添加到mysql页末尾，减少了随机IO。

###### 商品发布与交易



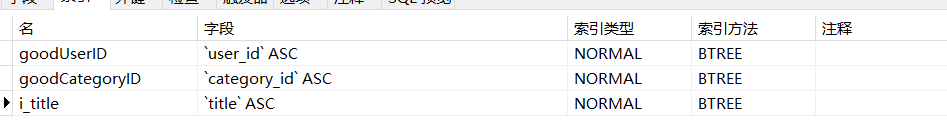
对于发布商品语句，因为good表中有自增主键，因此插入数据并不会造成底层数据移动，只会添加到mysql页末尾，减少了随机IO。



对于交易语句，判断的是商品唯一的id，是good表的主键，因此update可以直接走主键索引。

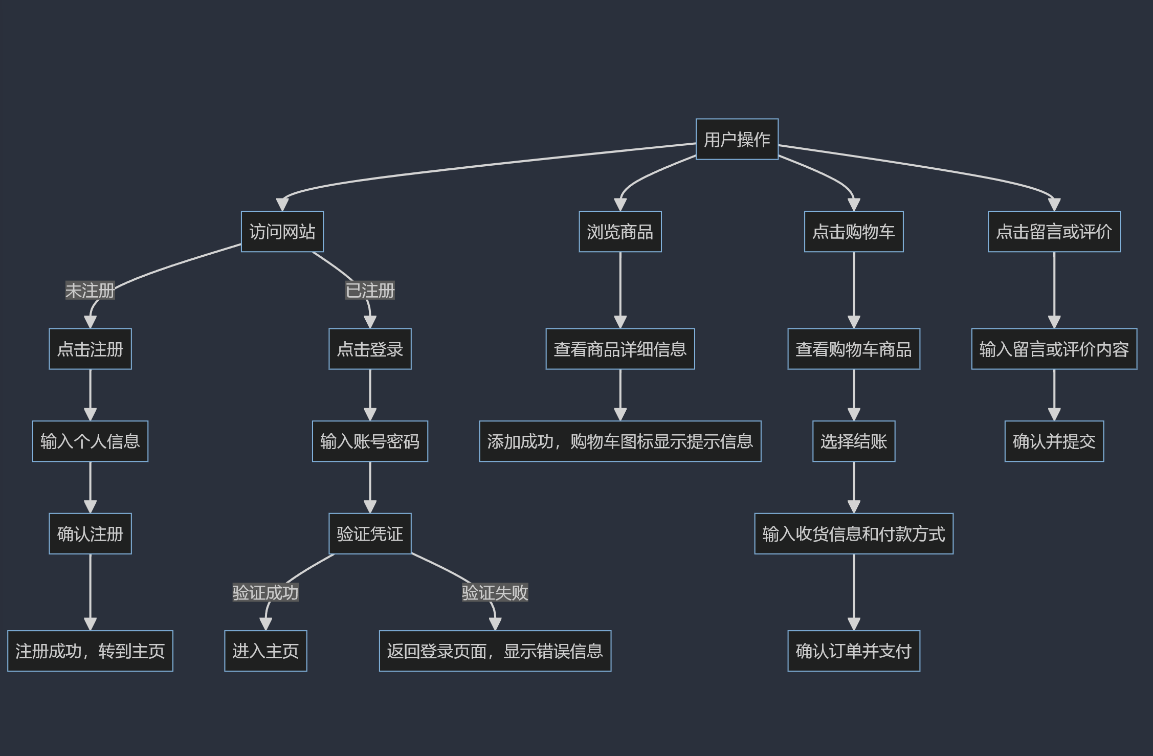
###### 3.4.3 搜索与过滤

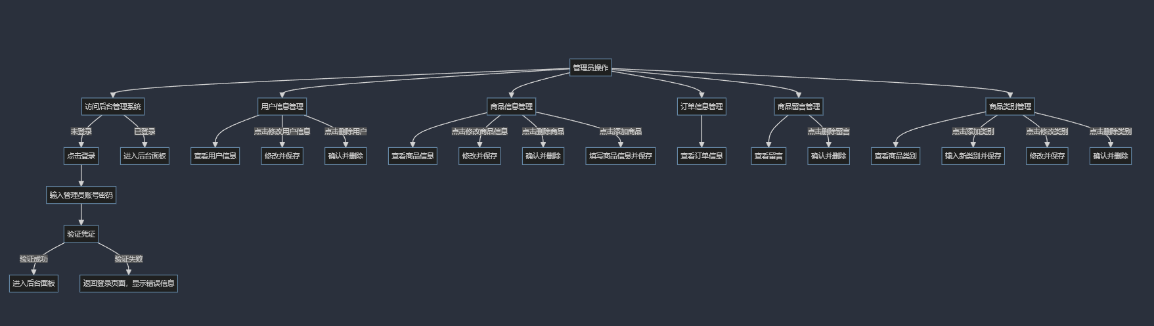
##### 



搜索与过滤语句使用了三个where的条件，但是mybitis框架会在项目运行时自动生成对应的select语句，对于这三个where条件各自都加上了索引。

* 1. 功能模块流程图
     1. 普通用户



* + 1. 管理员：

4、系统功能和测试

|  |  |  |
| --- | --- | --- |
| **功能** | **输入举例** | **输出举例** |
| 添加商品 | 耳机 九成新 价格120 | 提示输入成功 |
| 管理商品 | 将耳机的价格从120变为110 | 提示价格改动成功 |
| 管理商品类型 | 将耳机的商品类型更改 | 提示更改成功 |
| 登陆 | 用户名user密码123456 | 登陆成功 |
| 更改密码 | 原密码123456  新密码123456789  确认密码123456789 | 提示更改成功 |
| 注册用户 | 用户名、密码及确认密码 | 提示注册成功或失败 |
| 购买记录 | 点击 | 购买商品名称、订单号及时间 |
| 购买商品 | 点击 | 弹出确认及支付页面 |

测试用例：

| 用例唯一标识 | SHATRAP\_TEST-1.1 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 分别输入（正确的账号，正确的密码，正确的验证码），（正确的账号，错误的密码，正确的验证码），（正确的账号，正确的密码，错误的验证码）验证登录功能的可用性 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）在浏览器网址栏输入对应的登录网址127.0.0.1:8080/  （2）输入正确的账号，错误的密码，正确的验证码，并点击登录  （3）输入正确的账号，正确的密码，错误的验证码，并点击登录  （4）输入正确的账号，正确的密码，正确的验证码，并点击登录 | 1、显示登录页面  2、显示显示账号或密码错误  3、显示验证码错误，并自动刷新验证码  4、显示登录成功，并跳转到商城主页面 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |
|  |  | | |

| 用例唯一标识 | SHATRAP\_TEST-1.2 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 输入  1.（唯一的账号，重复且相同的密码，输入邮箱，点击获取验证码，输入正确的验证码）  2.（唯一的账号，重复且相同的密码，输入邮箱，点击获取验证码，输入错误的验证码）  3.（唯一的账号，不相同的密码，输入邮箱，点击获取验证码，输入正确的验证码）  4.（不唯一的账号，相同的密码，输入邮箱，点击获取验证码，输入正确的验证码）  点击注册，验证注册功能 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）在浏览器网址栏输入对应的登录网址127.0.0.1:8080/  （2）点击注册账号  （3）唯一的账号，重复且相同的密码，输入邮箱，点击获取验证码，输入错误的验证码  （4）唯一的账号，不相同的密码，输入邮箱，点击获取验证码，输入正确的验证码  （5）不唯一的账号，相同的密码，输入邮箱，点击获取验证码，输入正确的验证码  （6）唯一的账号，重复且相同的密码，输入已经注册过的邮箱，点击获取验证码，输入正确的验证码  （7）唯一的账号，重复且相同的密码，输入邮箱，点击获取验证码，输入正确的验证码 | （1）显示登录页面  （2）显示注册页面  （3）显示验证码错误  （4）显示两次输入密码不相同  （5）显示该账号已经有用户注册过了  （6）显示该邮箱已经被注册过了  （7）显示注册成功，并跳转到登录页面 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

| 用例唯一标识 | SHATRAP\_TEST-1.3 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 验证身份后，输入1.（重复相同密码）2.（不相同密码）点击修改密码 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）输入重复相同密码，点击确认  （2）输入不相同密码，点击确认 | （1）显示修改成功，并跳转到登录页面  （2）显示两次密码不相同。 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

| 用例唯一标识 | SHATRAP\_TEST-2.1 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 添加商品 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）输入商品名称  （2）输入商品价格与商品描述 | （1）提示发布成功 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

| 用例唯一标识 | SHATRAP\_TEST-2.2 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 管理商品信息 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）点击商品  （2）更改商品价格与商品描述 | （1）提示修改成功 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

| 用例唯一标识 | SHATRAP\_TEST-2.3 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 商品加入购物车 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）点击商品  （2）点击添加到购物车 | （1）提示加入购物车成功 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

| 用例唯一标识 | SHATRAP\_TEST-2.4 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 购买商品 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）点击商品  （2）点击购买商品 | （1）跳转到外部网站 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

| 用例唯一标识 | SHATRAP\_TEST-3.1 | | |
| --- | --- | --- | --- |
| 用例描述（功能描述） | 搜索商品信息 | | |
| 环境描述 | 系统正确配置并启动运行 | | |
| 实例编号 | 操作步骤 | 期望输出 | 测试记录 |
| 010 | （1）选择搜索框  （2）输入商品名称 | （1）显示相关商品情况，如失败则显示空白 | 与期望相符 |
| 测试人员：周穗 | | 测试时间：23.12.17 | |
| 备注 |  | | |

5、总结与体会

5.1 总结

我们小组已成功构建了基本的数据库管理系统原型，为整个系统的后续开发和完善奠定了基础。当前，我们已能做到进行基础功能的实施，比如产品展示、订单管理、用户管理和产品评论等。

每位小组成员均承担了一部分独特的任务，使我们一起完成了一个完整的开发周期，包括需求分析、设计、开发和测试。参与小组的合作和互动，使我们不仅提高了领导和团队协作力，还提升了处理问题和决策的能力。

就目前阶段而言，我们的项目已经初见雏形。虽然还存在需要改进和优化的部分，但是总的来说，我们已经实现了一套基本的数据库管理系统，可以进行基本的商品展示，订单管理，用户管理和商品评论等功能。

在项目管理层面，我们通过合理的分工和高效的协作，成功地推动了项目的进展。每个团队成员都有明确的职责和任务，按时完成了各自的分工。

5.2 体会与理解

在数据库应用方面，我们深感其对于存储和管理数据的重要性。我们了解到，良好的数据库结构不仅能帮助我们高效地组织和处理数据，还能为需求不断变更的项目提供支持。

在设计数据库结构时，我们从课本理论走向了真实的应用场景，锻炼了实际技能，学会了如何根据实际需求去构建和优化数据模型。我们体验到了，一个好的数据库设计不仅能为整个系统提供稳定的支持，而且能为后期的维护减轻压力。

我们也从中了解到了数据库查询优化的重要性，并学会了如何进行索引的设计和查询语句的优化，意识到这些对系统性能和用户体验的关键性。

对于数据库设计，我理解了它的重要性，以及如何根据实际需求来设计表结构和关联。我也体验到了前后端分离的好处，可以让开发更加专注和高效。同时，我也学习到了如何利用Git进行版本控制，避免了很多可能的问题。

这次项目使我们对数据库有了更深的理解，使我们更加重视这个技能。我们认识到，数据库不仅在数据存储方面发挥着关键作用，而且在理解和处理业务流程上也是非常重要的。

以上就是我们小组在这个项目中的总结和体会。通过这个项目，我们体验到了知识理论向实际操作转化的过程，也意识到了团队协作对项目成功的重要性。我们会在今后的学习和工作中，运用和发扬在此项目中学到的宝贵经验和教训。

六、小组分工

# 参考文献

[1] Spring Boot 开发文档. <https://spring.io/projects/spring-boot/>, 2023

[2] Mybatis Spring 文档. <https://mybatis.org/spring/zh/>, 2023

[3] LayUI 使用手册. <https://layui.dev/docs/2/>, 2023

[4] mermaid 使用手册. <https://mermaid.js.org/intro/syntax-reference.html> , 2023