**【校园跳蚤市场系统】**

**概要设计说明书**

**[版本号V\_1.0]**

**2022年11月29日**

**组长：杜康睿**

**副组长：李佳骏**

**组员：吴羽腾 卢晓雅 湛馥榕**

评 审 日 期：2022 年 12 月 5 日

目录

[**【校园跳蚤市场系统】** 1](#_Toc121857521)

[**1.导言** 3](#_Toc121857522)

[**1.1目的** 3](#_Toc121857523)

[**1.2范围** 3](#_Toc121857524)

[**1.3缩写说明** 3](#_Toc121857525)

[**1.4引用标准** 4](#_Toc121857526)

[**1.5参考资料** 4](#_Toc121857527)

[**1.6版本更新信息** 4](#_Toc121857528)

[**2.系统分析** 5](#_Toc121857529)

[**3.界面设计** 5](#_Toc121857530)

[**4.体系结构** 9](#_Toc121857531)

[**4.1 Django体系结构** 9](#_Toc121857532)

[**4.2系统体系结构** 11](#_Toc121857533)

[**5.数据模型** 12](#_Toc121857534)

[**5.1数据库的概念结构模型设计** 12](#_Toc121857535)

[**5.2数据库的逻辑结构模型设计** 13](#_Toc121857536)

[**6.模块设计** 17](#_Toc121857537)

**1.导言**

**1.1目的**

该文档的目的是描述校园跳蚤市场的概要设计，其主要内容包括：

* 系统功能简介
* 系统结构设计
* 系统接口设计
* 数据设计
* 模块设计
* 界面设计

本文档的预期的读者是：

* 开发人员
* 项目管理人员
* 测试人员

**1.2范围**

该文档定义了系统的结构和单元接口，但未确定单元的实现方法，这部分内容将在详细设计/实现中确定。

**1.3缩写说明**

UML

Unified Modeling Language（统一建模语言）的缩写，是一个标准的建模语言。

MVC

Model-View-Controller（模型－视图－控制器），M是指业务模型（Model），V是指用户界面（View），C则是控制器（Controller），使用MVC的目的是将M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式。

MTV

Django 框架主要关注的是模型（Model）、模板（Template）和视图（Views），称为MTV模式。

ORM

Object Relational Mapping，简称ORM（对象关系映射技术），用于实现面向对象编程语言里不同类型系统的数据之间的转换。

**1.4引用标准**

[1] 《企业文档格式标准》

北京长江软件有限公司

[2] 《软件概要设计报告格式标准》

北京长江软件有限公司软件工程过程化组织

**1.5参考资料**

[1] 《实战ｓｔｒｕｃｔ》 〔美〕Ｔｅｄ　Ｈｕｓｔｅｄ

机械工业出版社

[2] 《软件重构》

清华大学出版社

**1.6版本更新信息**

本文档的更新记录如表所示。

**表Ｂ-1 版本更新记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 000 | 2022.11.13 | 0.1 | 全部 | 初始发布版本 |
| 001 | 2022.12.01 | 1.0 | 6章节 | 修改模块设计部分 |

**2.系统分析**

校园跳蚤市场应该包含的系统的功能

￮ 用户登录：通过邮箱和密码实现用户登录

￮ 商品搜索：用户在搜索框中输入商品名称，查找相应商品。并返回相应商品。

￮ 发布商品：用户在商品管理界面填写商品信息，提交表单发布商品。

￮ 购买商品：浏览商品并可以选择需要的商品进行购买。

￮ 查看订单：查看购买的订单。

￮ 商品评价：对订单的商品进行评价。

**3.界面设计**

* 登录界面

分为登录、注册和忘记密码三个功能块：登录块通过用户名和密码以及验证码实现用户登录，并判断用户的权限，如图3-1所示；注册块需要用户设置用户名和密码，以及填入邮箱来注册账号，如图3-2所示；忘记密码块通过邮箱重新设置密码的操作，如图3-3所示。

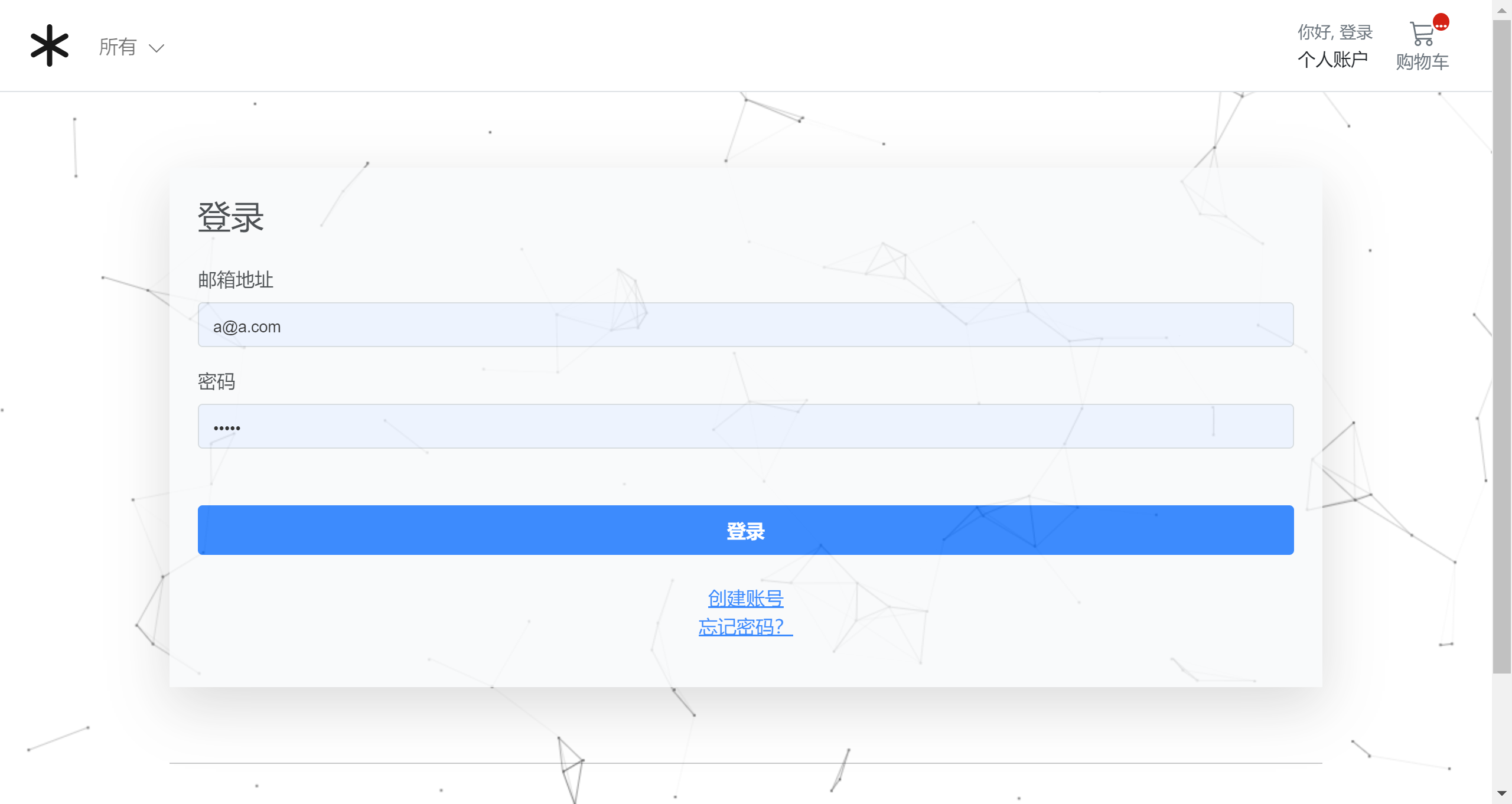


图3-1 登录页面设计

图形用户界面, 应用程序

描述已自动生成

图3-2 忘记密码页面设计

图形用户界面, 应用程序

描述已自动生成

图3-3 注册账号页面设计

* 个人信息页面

可以查看个人信息并进行修改。设计页面如下图所示：

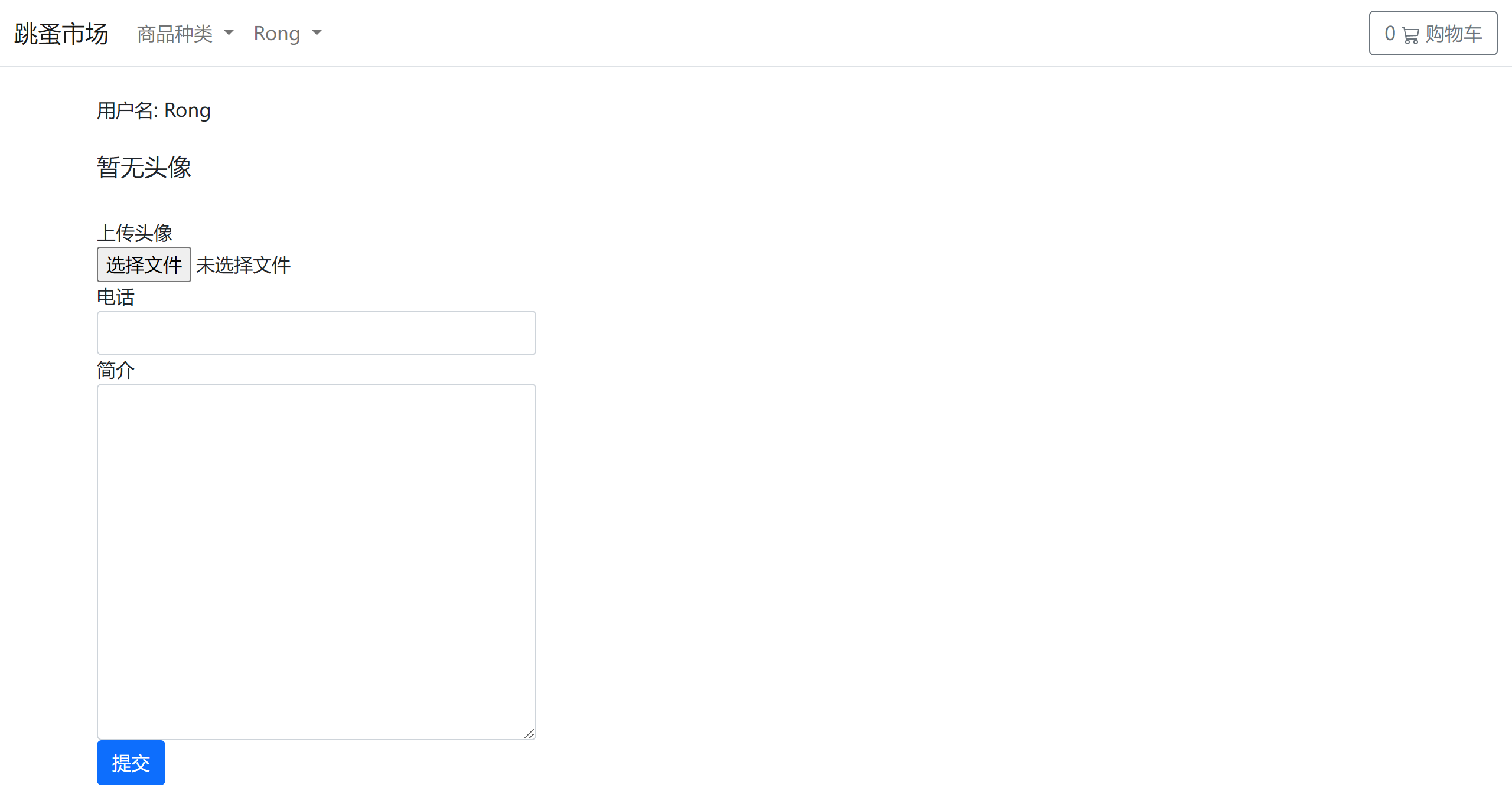


图3-4 个人信息页面设计

* 系统首页

根据用户的权限，进入首页，在首页中展示此用户相应可以访问的界面并且用户可以在首页打开个人信息以及修改密码的界面进行相关操作。点击商品，即可查看详细信息。设计页面如下图所示：

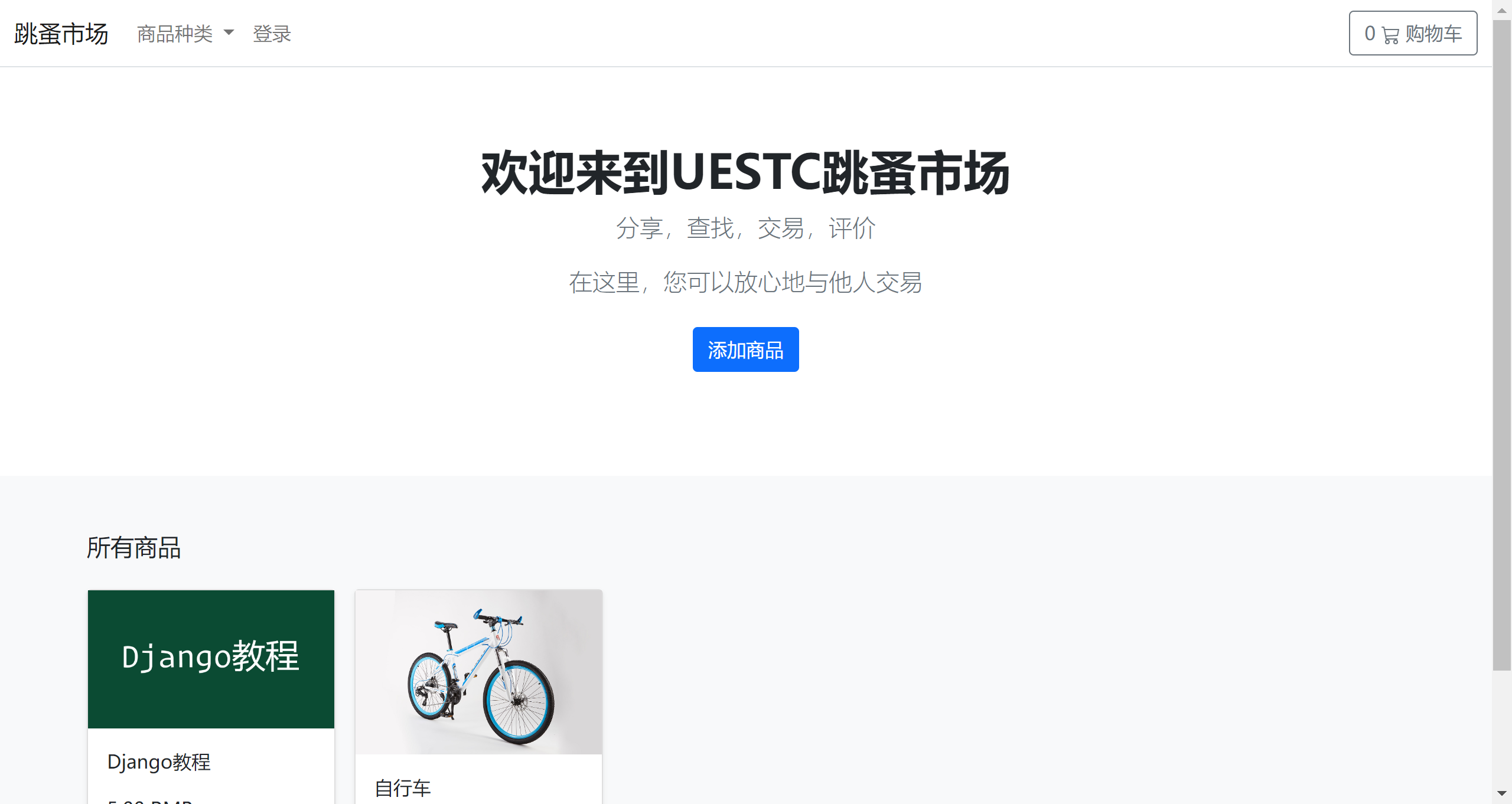


图3-5 系统首页页面设计

* 商品详情页面

进入商品详情页面，查看商品详细信息。可将商品加入购物车。设计页面如下图所示：



图3-6 商品详情页面设计

* 购物车页面

进入购物车页面，查看已经加入购物车的商品，在购物车页面可以提交订单。设计页面如下图所示：

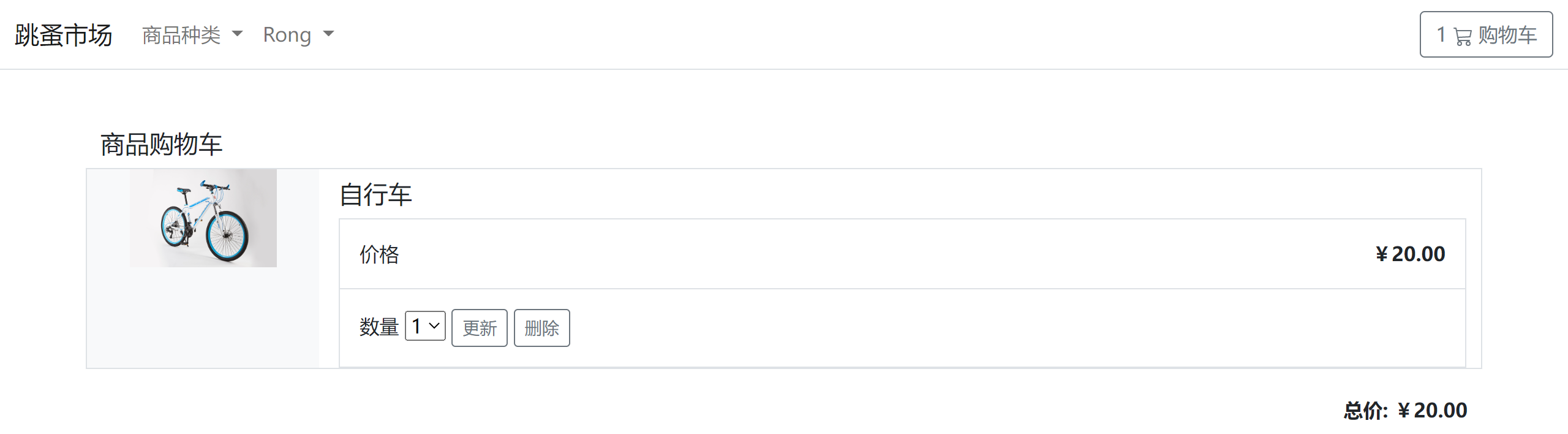


图3-7 购物车页面设计

* 订单页面

可以查看过往的订单记录。设计页面如下图所示：



图3-8 订单页面设计

* 发布商品

在发布商品页面，添加需要发布的商品。设计页面如下图：



图3-9 发布商品页面设计

**4.体系结构**

系统的总体结构设计遵循如下原则：

1. 系统应具有良好的适应性：能适应用户对系统的软件环境、管理内容、模式和界面的要求；
2. 系统应具有可靠性：采用成熟的技术方法和软件开发平台，以保证在以后的实际应用中安全、可靠；
3. 系统应具有较好的安全性：应提高完善的安全机制和用户权限限制机制，确保数据的受限访问；
4. 系统应具有良好的可维护性：系统应易于维护、安装；
5. 系统应具有良好的可扩展性：系统应适应未来信息化建设的要求，能方便得进行功能扩展，以建立完善的信息集成管理体系。

目前软件项目中有很多体系结构，其中Django是比较流行的一种。本系统采用Django体系结构，Django 是一个基于模型 (Model) －模板（Template）- 视图 (View) (MTV) 模式的应用架构的开源框架。

**4.1 Django体系结构**

对于开发Web应用，要从头设计并开发出一个可靠、稳定的框架不是一件容易的事情，随着Web开发技术的日趋成熟，在web开发领域出现了一些现成的优秀的框架，开发者可以直接使用它们，Django就是一个很好的框架结构。

Django继承并简化了MVC架构。MVC中的Controller部分基本全由Django完成。View部分被分割成两部分，即：负责HTML渲染的模板和负责显示逻辑的视图。Django 体系结构总览图如下：



图4-1 Django的MVC架构

**Django 框架主要关注的是模型（Model）、模板（Template）和视图（View），称为MTV模式。**它们各自的职责如下：

**表4-1 Django的MTV模式**

|  |  |
| --- | --- |
| 层次 | 职责 |
| 模型（Model），即数据存取层 | 处理与数据相关的所有事务： 如何存取、如何验证有效性、包含哪些行为以及数据之间的关系等。 |
| 模板（Template），即业务逻辑层 | 处理与表现相关的决定： 如何在页面或其他类型文档中进行显示。 |
| 视图（View），即表现层 | 存取模型及调取恰当模板的相关逻辑。模型与模板的桥梁。 |

**简单来说就是Model存取数据，View决定需要调取哪些数据，而Template则负责将调取出的数据以合理的方式展现出来。**

在 Django 中，并不需要直接去操作数据库，使用***对象关系映射****（****Object Relational Mapping****，简称****ORM****）技术，*只用使用Python语法定义好**模型**，底层代码不用关心，**模型**中会自动生成操作数据库所必要的命令。

Django架构处理请求的流程可以概括如下：

1. Django中间件收到一个用户请求；
2. Django通过URLconf查找对应的视图后进行URL路由分发；
3. 视图接收到请求，查询到对应的模型，调用模板生成HTML；
4. 视图返回一个处理后的HTML内容；
5. WEB服务器将响应内容发给客户端。

可以用下图表示：

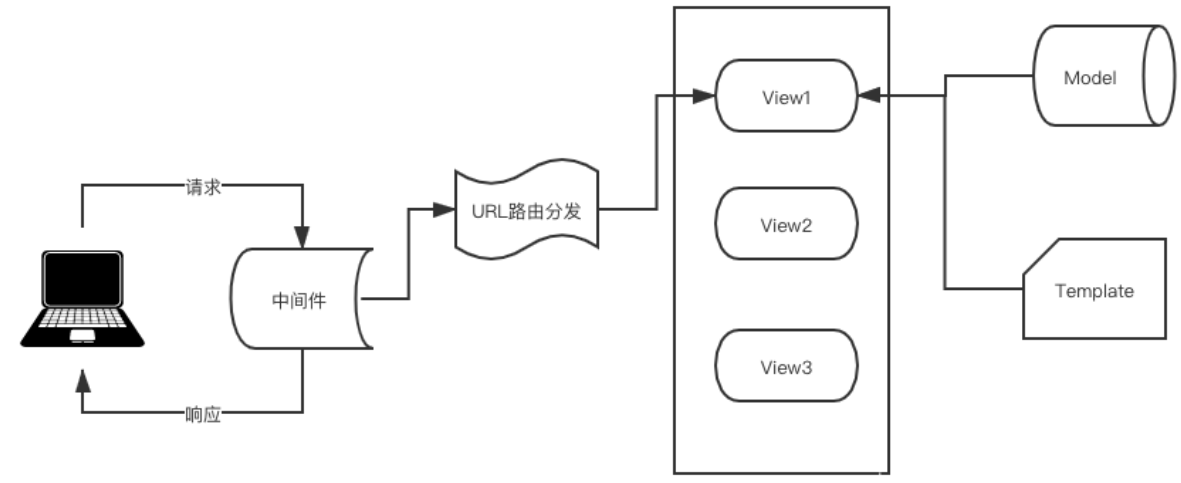


图4-2 Django架构处理请求的流程

**4.2系统体系结构**

根据系统分析结果，该系统从结构上应满足：

* 基于浏览器进行显示以方便用户使用；
* 采用MTV的三层体系结构，分化各个功能组件；

用下图可以将体系结构形象地展示出来。

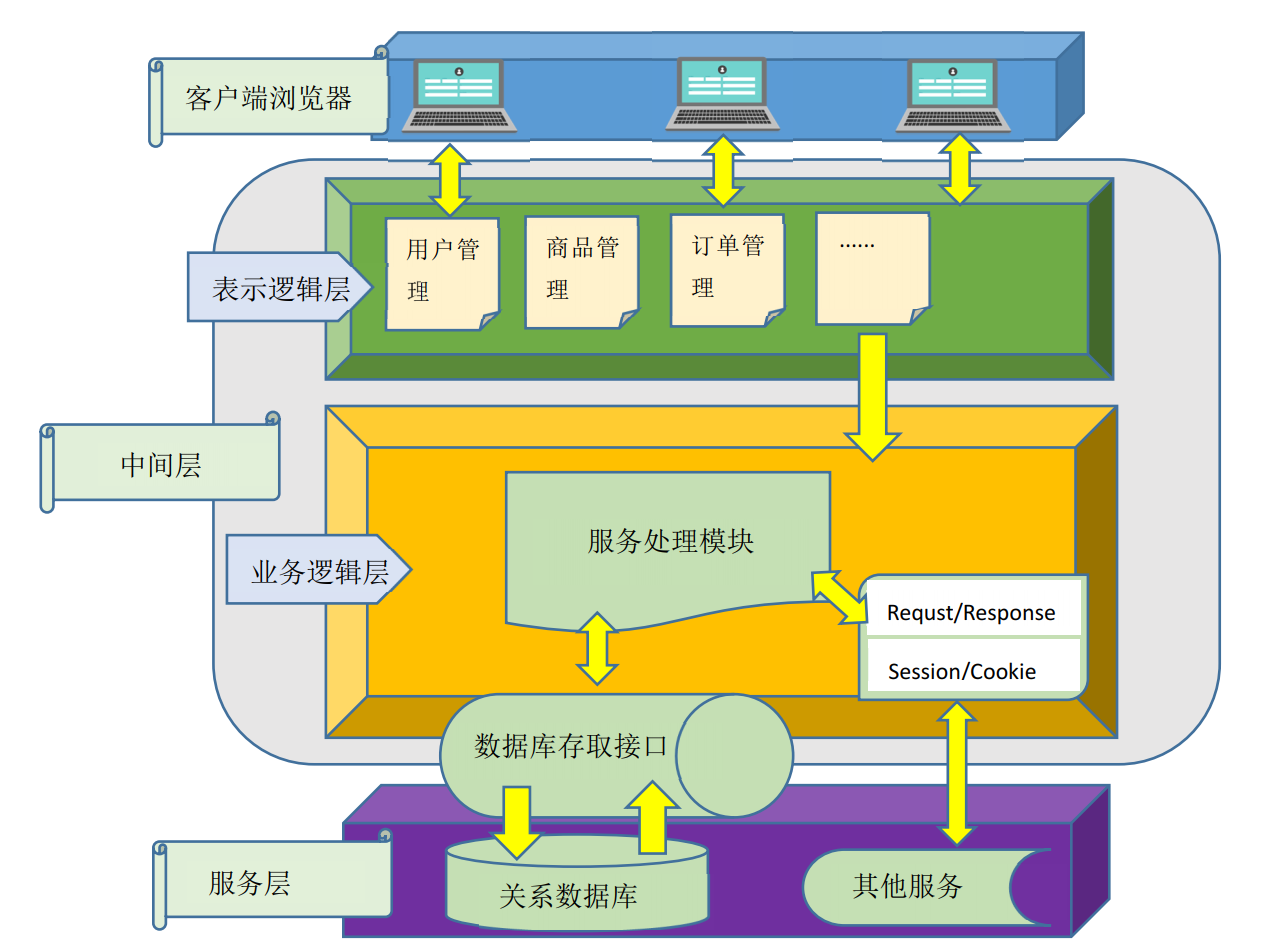


图4-3 校园跳蚤市场的体系结构

客户层主要是指用户登录的Web浏览器；中间层负责平台的业务逻辑处理和表示逻辑生成；服务层提供底层的信息数据库服务器。

1.客户层：用于与企业信息系统的用户进行交互以及显示根据特定业务规则进行计算后的结果。本系统将完全采用基于WEB的（B/S架构）客户端，即用户可以直接通过浏览器来访问和使用本系统。

2.中间层：这相当于三层标准架构中的Web应用服务层，支持诸如响应客户请求以及查询等功能。并且由中间层进行逻辑处理，再处理的结果反馈给客户或者发送到数据库中。

3.服务层：主要是数据库系统，这里的数据库系统主要是关系数据库系统(RDMS)，Django中自带轻量级数据库sqlite3，创建工程时自动在工程目录中生成一个数据库文件db.sqlite3作为数据库文件。这个单文件数据库可以满足项目数据库的基本功能要求。

**5.数据模型**

本系统的数据模型主要是进行数据库的设计。

**5.1数据库的概念结构模型设计**

概念设计以反映现实世界中的实体、属性和它们之间的关系等的原始数据形式，建立数据库的每一幅用户视图。图5-1是系统E-R图。跳蚤市场系统中用户可上传自己想要出售的商品，买家购买商品前需要完善自己的地址信息，购买某商品后则会生成订单，订单将会记录交易的相关信息，买家收到商品后即可点击确认收货完成此次交易。

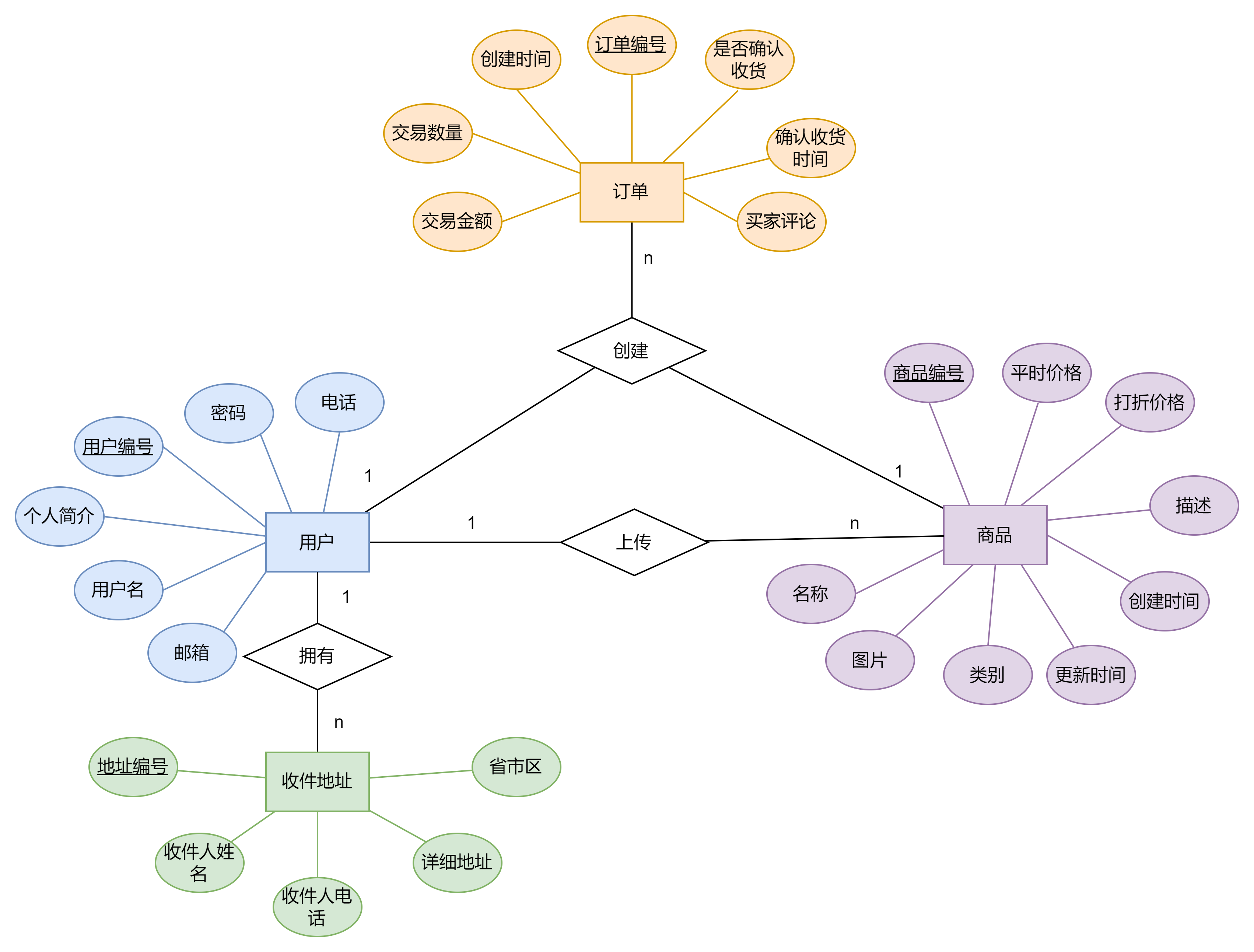


图5-1 系统E-R图

**5.2数据库的逻辑结构模型设计**

数据库的逻辑设计是将各局部的E-R图进行分解、合并后重新组织起来形成数据库全局逻辑结构，包括所确定的关键字和属性、重新确定的记录结构、所建立的各个数据之间的相互关系。根据本系统需求分析，系统的数据库包括了商品管理、用户管理、订单管理等部分。

商品管理包括的库表：

* Product - 商品信息表

用户管理包括的库表：

* User－用户信息表
* Address - 收件地址表

订单管理包括的库表：

* Ordering－订单信息表

下列表项是对各个库表的设计：

**表5－1：Product（商品）表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 字段类型 | 长度 | 是否允许为空 | 默认值 | 备注 |
| id | int | int |  | NO |  | 商品ID（主键） |
| user\_id | int | int |  | NO |  | 上传人ID（外键） |
| name | varchar(255) | varchar | 255 | NO |  | 商品名称 |
| regular\_price | int | int |  | NO |  | 平时价格 |
| discount\_price | int | int |  | NO |  | 打折价格 |
| category | varchar(255) | varchar | 255 | YES |  | 类别 |
| image | image | image |  | YES |  | 商品图片 |
| description | varchar(255) | varchar | 255 | YES |  | 描述 |
| updated\_date | dateTime | dateTime |  | YES |  | 更新日期 |
| created\_date | dateTime | dateTime |  | YES |  | 创建日期 |

**表5－2：User（用户）表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 字段类型 | 长度 | 是否允许为空 | 备注 |
| id | int | int |  | NO | 用户ID |
| username | varchar(255) | varchar | 255 | NO | 用户名 |
| email | varchar(255) | varchar | 255 | NO | 邮箱 |
| password | varchar(255) | varchar | 255 | NO | 密码 |
| mobile | varchar | varchar | 11 | YES | 电话 |
| resume | varchar(255) | varchar | 255 | YES | 个人简介 |

**表5－3：Address（地址）表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 字段类型 | 长度 | 是否允许为空 | 备注 |
| id | int | int |  | NO | 地址ID（主键） |
| user\_id | int | int |  | NO | 用户ID（外键） |
| name | varchar(255) | varchar | 255 | NO | 收件人姓名 |
| mobile | varchar | varchar | 11 | NO | 收件人电话 |
| province | varchar(255) | varchar | 255 | YES | 省市区 |
| mobile | varchar | varchar | 11 | YES | 详细地址 |
| resume | varchar(255) | varchar | 255 | YES | 个人简介 |

**表5－4：Ordering（订单）表**

**注：订单的外键是商品和买家，卖家作为商品的外键和订单产生间接的关联**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 字段类型 | 长度 | 是否允许为空 | 备注 |
| id | int | int |  | NO | 订单ID（主键） |
| seller\_id | int | int |  | NO | 卖家ID（外键） |
| buyer\_id | int | int |  | NO | 买家ID（外键） |
| product\_id | int | int |  | NO | 商品ID（外键） |
| ordering\_amount | int | int |  | NO | 交易金额 |
| quantity | int | int |  | NO | 交易数量 |
| comment | varchar(255) | varchar | 255 | YES | 卖家评论 |
| created\_date | dateTime | dateTime |  | YES | 创建日期 |
| billing\_status | Boolean | Boolean |  | NO | 是否确认收货 |
| finish\_date | dateTime | dateTime |  | YES | 确认收货日期 |

在确定了各个表主键字段的基础上,依据表与表相关字段之间的联系建立了各表之间的关系，如图5-2所示。

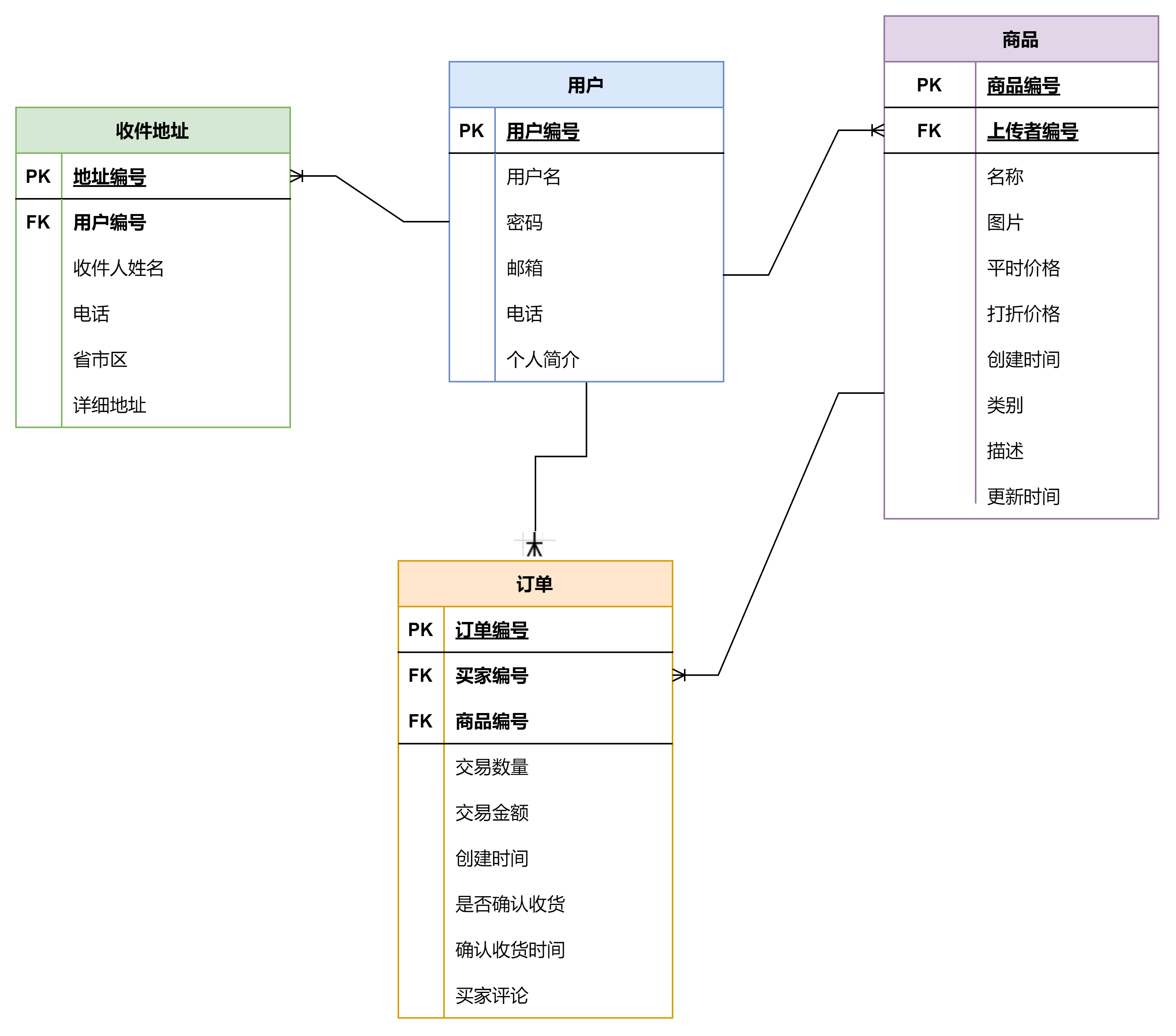


图5-2 各表之间的关系

**6.模块设计**

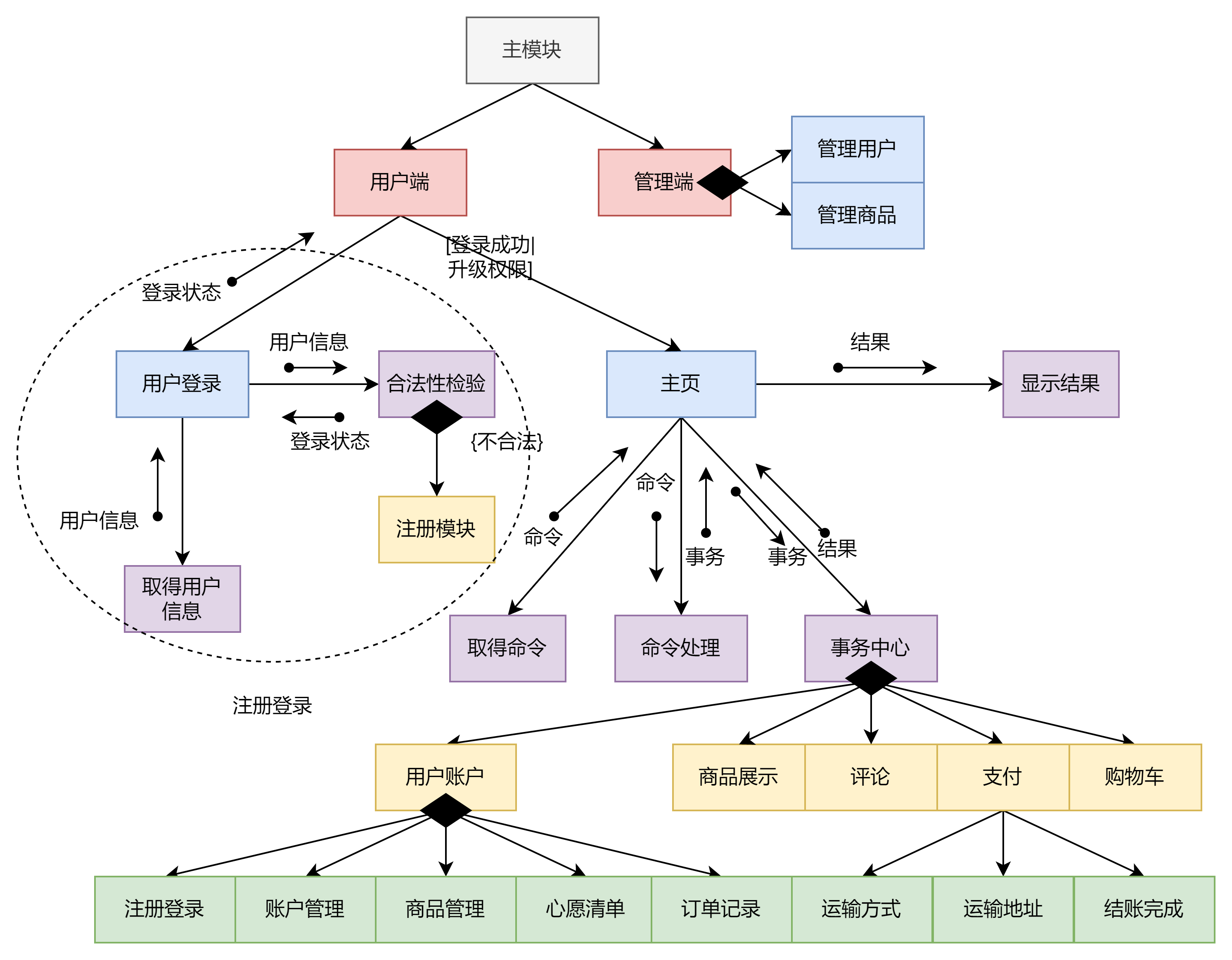


图6-1 结构图

采用基于数据流的设计方法进行体系结构设计，画出如上图所示的结构图。虚线圈出的部分是用户账户下注册登录模块的内部结构。

系统分为用户端与管理端，管理端有管理用户与管理商品两个模块。用户端如果不登录，提交网址后进入主页后只有游客权限，可以查看商品，加入商品至购物车和心愿清单。但没有账户，无法进行支付与评论。

游客进入用户账户下的注册登录界面，调用“取得用户信息”模块，获得用户信息传回登录模块。登录模块把信息传给合法性检验模块，若合法性检验模块发现用户id不存在（不合法），会调用注册模块，重定向至注册界面。若用户id存在（合法），合法性检验模块检测密码是否正确，从而判定是否登录成功，并将登录状态传递给用户登录模块。用户登录模块将登录状态传给主模块，若登录成功，用户权限升级。

主页模块如果获得命令，将命令传给命令处理模块，命令处理模块返回结果（事务）给主页模块，主页模块将此结果传给事务中心。事务中心获得事务后依据事务的不同传给不同的模块，这些模块都将直接或间接的调用数据库管理模块以获得相应的数据或存入相应的数据。

事务中心根据信息选择性调用以下5个模块，分别是用户账户、商品展示、评论、支付以及购物车。在用户账户下，有4个小模块，注册登录、账户管理、商品管理、心愿清单以及订单记录。支付模块包含3个小模块，运输方式、运输地址以及结账完成。运输方式用于选择快递方式。在运输地址模块中，可以添加、修改地址，也可以设置默认地址。方式、地址均选择完成后，点击结账，会进入结账完成界面，表示当前订单已完成。