

本科生毕业设计(论文)

中文题目　闲置物品交易系统的设计与实现

外文题目　Design and Implementation of

Idle Goods Trading System

学　　号　　　201626705010

姓　　名　　　 陈德斌

学　　院　　　软件学院

专　　业　　　软件工程

指导教师　　　　徐虹

完成时间

江西师范大学教务处制

声　　明

本人郑重声明：

所呈交的毕业设计（论文）是本人在指导教师指导下进行的研究工作及取得的研究成果。其中除加以标注和致谢的地方，以及法律规定允许的之外，不包含其他人已经发表或撰写完成并以某种方式公开过的研究成果，也不包含为获得其他教育机构的学位或证书而作的材料。其他同志对本研究所做的任何贡献均已在文中作了明确的说明并表示谢意。

本毕业设计（论文）成果是本人在江西师范大学读书期间在指导教师指导下取得的，成果归江西师范大学所有。

特此声明。

声明人（毕业设计（论文）作者）学号：

声明人（毕业设计（论文）作者）签名：

签名日期：　　　　年　　月　　日

# 

摘 要

随着大学生经济消费能力的日益增强，一些东西被买回去后，由于用处不大而闲置在寝室，但对其他同学来说，这些物品也许有相当大的使用价值，甚至，一些还有使用价值的物品被同学们随意丢弃，不仅造成了资源浪费，还加剧了环境污染。我校大批毕业生离校之际有许多拿不走、带不回去的物品如用过的考研、计算机二级等的资料、水壶衣架、书桌、台灯等等生活用品，就这些物品如何处置，开发一个校园闲置物品交易系统，可以说是众望所归了。

本系统基于.net core Api、Entityframeworkcore、前后端分离等技术开发，网站功能分为用户管理模块、选购商品模块、商品管理模块和订单管理模块等，可以帮助同学们将闲置物品再利用。

论文首先介绍了物品闲置物品交易系统的现状和意义，然后对网站进行了需求分析，在对网站进行详细分析后，依次阐述了各个模块的实现，通过测试分析，该系统运行稳定可靠，最后对系统进行了总结和展望。

**关键词**：闲置物品；.net core；VueJs；交易

Abstract

With the increasing economic consumption ability of college students, some things are bought back and idle in the dorm because they are not very useful, but for other students, these items may have considerable use value, and even some have use value The items discarded by the students are not only wasteful of resources, but also aggravate environmental pollution. When a large number of graduates from our school leave the school, there are many items that cannot be taken or brought back, such as used postgraduate examinations, computer secondary information, kettle hangers, desks, table lamps and other daily necessities. How to dispose of these items, It can be said that the development of a trading system for idle items on campus is well received.

This system is developed based on technologies such as .net core Api, Entityframeworkcore, and separation of front and back ends. The website functions are divided into user management module, product purchase module, product management module and order management module, etc., which can help students to reuse idle items.

The paper first introduces the status and significance of the idle item trading system, and then analyzes the needs of the website. After detailed analysis of the website, the implementation of each module is explained in turn. Through test analysis, the system runs stable and reliable. The system is summarized and prospected.

**Key words**: Idle items; .net core; VueJs; transactions

目录

目 录

[**第 1 章 绪论 1**](#_bookmark0)

[1.1 系统开发的背景 1](#_bookmark1)

[1.2 系统开发的目标、意义 1](#_bookmark2)

[1.3 本文的主要工作 2](#_bookmark3)

[1.4 论文结构 2](#_bookmark4)

[**第 2 章 需求分析 3**](#_bookmark5)

[2.1 系统功能需求 3](#_bookmark6)

[2.2 数据需求 11](#_bookmark7)

[2.3 系统非功能性需求 12](#_bookmark8)

[**第 3 章 系统设计 14**](#_bookmark9)

[3.1 功能模块设计 14](#_bookmark10)

[3.2 数据库设计 28](#_bookmark11)

[**第 4 章 模块实现 32**](#_bookmark12)

[4.1 系统软件架构 32](#_bookmark13)

[4.2 用户管理实现 36](#_bookmark14)

[4.3 商品选购实现 42](#_bookmark15)

[4.4 商品管理实现 45](#_bookmark16)

[4.5 订单管理实现 47](#_bookmark17)

[**第 5 章 总结和展望 51**](#_bookmark19)

[5.1 总结 51](#_bookmark20)

[5.2 展望 51](#_bookmark21)

[**参考文献 53**](#_bookmark22)

[**致谢 54**](#_bookmark22)

**第 1 章** **绪论**

1.1 系统开发的背景

随着大学生经济消费能力的日益增强以及互联网电商的各种营销刺激下，大学生们非常容易形成“冲动”消费，一些东西被买回去后，由于相对用处不大而闲置在寝室，因而产生了大量的闲置物品，但对其他同学来说，这些物品也许有相当大的使用价值，甚至，一些还有使用价值的物品被同学们随意丢弃，不仅造成了资源浪费，还加剧了环境污染。

目前，保护生态环境是我国面临的一项十分艰巨的任务，大学生是社会的中坚力量，树立正确的生态环境道德和环境保护意识是社会发展的必然要求，也是公民道德建设的重要内容。在环境保护的践行者之中，大学生是最有积极性和执行力的群体，有效地利用好这个群体，将有力推动环境保护的发展，倡导节约新风尚。

我校大批毕业生离校之际有许多拿不走、带不回去的物品如用过的考研、计算机二级等的资料、水壶衣架、书桌、台灯等等生活用品，然而这些闲置物品都是闲散的，不能得到很好的二次利用。而且通常大学生们消费水平都不太高，希望能够处理自己的这些闲置物品，亦或者想要得到一些物美价廉的商品，已经成为大学生们需要解决的一个难题。就这些物品如何处置，开发一个校园闲置物品交易平台，可以说是众望所归了。

1.2 系统开发的目标、意义

本系统致力于闲置物品交易，通过本系统，我们希望可以为烦恼于如何处理闲置物品以及想要以较低的价格获得一些自己想要的物品的大学生用户提供帮助。本系统可以为闲置物品的处理提供渠道，用户可以在本系统发布自己的闲置物品，通过交易可以以物换钱，实现闲置物品的处理。用户亦可以在系统查询自己需要的东西，通过交易能以较低的价格获取自己想要的物品。

随着互联网的大力发展，互联网对人们的影响日渐增强，基于互联网的影响力，若系统得到大力推广，既能满足大学生们日益增长的生活需要，又可促进可利用资源的循环使用，减少资源的浪费。在注重商品交易的同时，更多的考虑到环境保护因素，贴合时下主流文化，让每个人在交易的同时，能够参与到环境保护中。随着时间的迁移，当代大学生的消费观会潜移默化的收到影响，从而为其建立良好的消费观提供帮助，将有力推动环境保护的发展，倡导节约新风尚。

1.3 本文的主要工作

该项目为本人个人项目，本人完成了用户管理模块、选购商品模块、商品管理模块和订单管理模块的设计与实现。

1.4 论文结构

本论文是对闲置物品交易系统的设计与实现过程的总结性描述，主要分为五个部分，分五个章节来详细描述闲置物品交易系统。

第一章，绪论。包括系统开发的背景、目标、意义、主要工作和论文结构。

第二章，需求分析。包括系统功能需求、数据需求和系统非功能性需求。

第三章，系统设计。包括功能模块设计和数据库设计。

第四章，模块实现。包括系统软件架构和具体模块的实现。

第五章，总结和展望。

**第 2 章 需求分析**

2.1 系统功能需求

2.1.1 角色分析

从闲置物品交易系统的实际需求分析，系统涉及到以下角色如表 2.1所示：

表 2.1 系统角色分析

|  |  |
| --- | --- |
| **角色** | **职责** |
| 游客 | 可浏览网站、注册成为会员 |
| 会员 | 拥有用户管理、选购商品、商品管理和订单管理功能 |

2.1.2 总体用例分析

闲置物品交易系统的参与者主要包括游客和会员两个，经过分析系统包括浏览网站、注册、用户管理、商品选购、商品管理和订单管理六个用例，如下图 2.1 所示。



图 2.1 系统总体用例图

在本闲置物品交易系统的开发中，本人完成用户管理、商品选购、商品管理、订单管理四个功能模块（以下称为闲置物品交易系统）的分析、设计与实现。下面对其相关功能进行分析。

2.1.3 用户管理分析

用户管理用例包括用户登陆和用户个人资料修改2个子用例。其用例图如图2.2所示。



图 2.2 用户管理用例图

下面以登陆子用例和个人资料修改子用例进行分析，它们的用例描述分别如表2.2、2.3所示。

表2.2 登陆子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 登陆 |
| **用例描述** | 描述了游客使用本系统进行登陆变为会员的过程 |
| **参与者表** | 游客 |
| **前置条件** | 游客已经在浏览闲置物品交易系统网站 |
| **后置条件** | 游客登陆成功 |
| **基本操作流** | 1．游客鼠标点击“登陆”按钮，发出登陆请求；  2．游客填写登陆信息；  3．游客点击提交按钮； |
| **可选操作流** | 在按“提交”按钮之前，游客随时可以按“返回”按钮，登陆状态不改变。 |

表2.3 修改个人信息子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 修改个人信息 |
| **用例描述** | 描述了会员使用本系统进行修改个人信息的过程 |
| **参与者表** | 会员 |
| **前置条件** | 用户已登录 |
| **后置条件** | 信息修改成功 |
| **基本操作流** | 1．用户鼠标点击“修改个人信息”按钮；  2．用户编辑要修改的信息，按“修改”按钮，完成修改； |
| **可选操作流** | 返回个人中心 |

2.1.4 商品选购分析

商品选购用例中包括检索商品、查看商品详情、将商品放入购物车和将商品移出购物车等 4 个子用例，其用例图如图 2.3 所示。



图 2.3 商品选购用例图

下面以检索商品子用例、查看商品详情子用例进行分析，它们的用例描述分别如表 2.4、2.5 所示。

表2.4 检索商品子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 检索商品 |
| **用例描述** | 描述了会员使用本系统进行检索商品的过程 |
| **参与者表** | 会员 |
| **前置条件** | 无 |
| **后置条件** | 系统显示所检索的商品列表 |
| **基本操作流** | 1.会员提供关键字；  2.查看关键字相关商品列表。 |
| **可选操作流** | 无 |

表2.5 查看商品详情子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 查看商品详情 |
| **用例描述** | 描述了会员使用本系统查看商品详情的过程 |
| **参与者表** | 会员 |
| **前置条件** | 系统已提供商品列表 |
| **后置条件** | 系统显示所选中的商品详情信息 |
| **基本操作流** | 1.会员选中待查看的商品记录；  2.查看相关商品的详情信息。 |
| **可选操作流** | 无 |

2.1.5 商品管理分析

商品管理用例中包括发布商品、修改商品和下架商品等3个子用例，其用例图如图2.4所示。



图 2.4 商品选购用例图

下面以发布商品子用例、修改商品子用例和下架商品子用例进行分析，它们的用例描述分别如表 2.6、2.7 和2.8所示。

表2.6 发布商品子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 发布商品 |
| **用例描述** | 描述了会员使用本系统进行发布商品的过程 |
| **参与者表** | 会员 |
| **前置条件** | 会员已登录系统 |
| **后置条件** | 商品发布成功 |
| **基本操作流** | 1．用户鼠标点击“发布商品”按钮；  2．用户编辑商品，点击“提交”按钮； |
| **可选操作流** | 返回 |

表2.7 修改商品子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 修改商品 |
| **用例描述** | 描述了会员使用本系统进行修改商品的过程 |
| **参与者表** | 会员 |
| **前置条件** | 会员已登录系统 |
| **后置条件** | 商品修改成功 |
| **基本操作流** | 1．用户鼠标点击“编辑商品”按钮；  2．用户编辑商品，点击“提交”按钮； |
| **可选操作流** | 返回 |

表2.8 删除商品子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 删除商品 |
| **用例描述** | 描述了会员使用本系统进行删除商品的过程 |
| **参与者表** | 会员 |
| **前置条件** | 会员已登录系统 |
| **后置条件** | 商品下架成功 |
| **基本操作流** | 1．用户选中列表某商品；  2．用户鼠标点击“删除”按钮； |
| **可选操作流** | 返回 |

2.1.6 订单管理分析

订单管理用例中包括提交订单、查看订单、查看订单明细和删除订单等4个用例，其用例图如图2.5所示。



图 2.5 订单管理用例图

下面以提交订单子用例、查看订单子用例、查看订单明细子用例和删除订单子用例进行分析，它们的用例描述分别如表 2.9、2.10、2.11和2.12所示。

表2.9 提交订单子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 提交订单 |
| **用例描述** | 描述了会员使用本系统进行提交订单的过程 |
| **参与者表** | 会员 |
| **前置条件** | 会员已登录系统 |
| **后置条件** | 订单提交成功 |
| **基本操作流** | 1.会员在商品界面选择商品点击购买；  2.会员在购物车选择商品点击购买 |
| **可选操作流** | 返回 |

表2.10 查看订单子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 查看订单 |
| **用例描述** | 描述了会员使用本系统进行查看订单的过程 |
| **参与者表** | 会员 |
| **前置条件** | 会员已登录系统 |
| **后置条件** | 展示订单列表 |
| **基本操作流** | 1. 用户进入用户中心 2. 用户点击“我的订单” |
| **可选操作流** | 查看相关订单明细 |

表2.11 查看订单明细子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 查看订单明细 |
| **用例描述** | 描述了会员使用本系统进行查看订单明细的过程 |
| **参与者表** | 会员 |
| **前置条件** | 1.会员已登录系统  2.会员已有提交的订单 |
| **后置条件** | 展示订单详情 |
| **基本操作流** | 用户点击订单项，查看订单相关明细 |
| **可选操作流** | 返回 |

表2.12 删除订单子用例描述

|  |  |
| --- | --- |
| **描述项** | **说明** |
| **用例名称** | 删除订单 |
| **用例描述** | 描述了会员使用本系统进行删除订单的过程 |
| **参与者表** | 会员 |
| **前置条件** | 1. 会员已登录系统 2. 会员已有提交订单 |
| **后置条件** | 订单删除成功 |
| **基本操作流** | 1．用户选中订单  2．用户鼠标点击“删除订单”按钮； |
| **可选操作流** | 返回 |

2.2 数据需求

经过分析后得知系统中所需的类和各类之间的关系，利用概念类图进行描述。图中闲置物品交易系统界面列提供用户管理、商品选购、商品管理和订单管理等操作，如图2.6所示。



图 2.6 系统概念类图

根据以上概念类图的内容，闲置物品交易系统包括地址、订单详情、订单、商品类型、商品信息、购物车记录和用户等实体。其中，用户与地址为一对多关系，用户与订单为一对多关系，订单与订单详情为一对多关系，用户与购物车为一对一关系，商品信息与商品类型为多对一关系，商品信息与购物车为多对一关系。闲置物品交易系统的 ER 图如图2.7 所示。



图2.7 闲置物品交易系统 E-R 图

2.3系统非功能性需求

2.3.1 可靠性和可用性需求

**安全措施需求：**

注册用户可能会忘记密码或者账号信息，但是注册用户可以通过密码找回功能将密码以及账号信息找回，并使用。

**安全性需求：**

为了防止没有修改权限的用户无意间修改了数据，不同的用户，有不同的权限，注册用户有注册用户的登录界面。当用户登录时，系统会根据用户ID和用户密码，系统会自动进行核实，鉴定此用户是否合法，如果合法，用户才能进一步操作。

**软件质量属性：**

本系统易用性良好，使用方便，具有较好的可靠性，有效性可维护性可移植性。

2.3.2 出错及处理需求

1.**出错信息**

出现错误时，捕捉到网站错误，存入错误记录文件中，供相关人员查看。用户不会看到具体的错误而是跳转到错误页面。

2**.补救措施**

网站出现错误后可采取的措施：

1、根据错误记录文件，做出相应错误处理。

2、检测是否启动服务。

3、备份数据，重新部署网站。

4、重新启动服务。

3.**网站维护设计**

为方便网站的维护，在代码设计中，以对象设计为主、结构设计为辅，对于难以理解的步骤给出相应的注释，方便维护人员查看代码和阅读代码。还有会写出相应的文档来记录编码时的问题，为以后网站升级做好基础。

2.3.3 网站未来可能需求

1、未来用户需求：

如果“闲置物品交易网站”有幸得到大家喜爱，随着用户的增多可能会增加更多好玩的活动、更多的用户体验，从而使得本网站更加丰富，更能满足用户的需求。

2、未来系统需求：

分析未来的用户量的剧增，那就对网站的内部要求更苛刻了，我们会分地区建立多个服务器，然后由一个总的服务器进行管理下面的服务器，把服务器的硬件加大，对数据库进行数量增加。

**第 3 章 系统设计**

3.1 功能模块设计

根据需求阶段的功能分析可知，闲置物品交易系统可以划分为用户管理、商品选购、商品管理和订单管理4个模块，其功能包图如图 3.1 所示。[1][2]



图3.1 闲置物品交易系统功能包图

本节中，将对闲置物品交易系统功能包图中的各个功能模块进行具体设计，具体如下。

3.1.1 用户管理设计

**（1）功能结构设计**

“用户管理”功能分为注册、登录和修改个人信息三个部分，其包图如图3.2所示。



图 3.2 用户管理包图

**（2）类图设计**



图 3.3 用户管理类图

如上图 3.3 所示是用户管理类图中，含有一个用户类，用户类主要负责检测用户是否存在，获取用户、展示用户列表、添加用户、删除用户和修改用户信息等功能。

**（3）顺序图设计**

注册功能的顺序图如图 3.4 所示。



图 3.4 注册功能顺序图

登录功能的顺序图如图 3.5 所示。



图 3.5登录功能顺序图

修改用户信息功能的顺序图如图 3.6 所示。



图 3.6修改用户信息功能顺序图

**（4）核心处理流程设计**

针对上述顺序图，图 3.5 登录功能和图 3.6 修改个人信息功能这两个功能还可以进一步细化，如图 3.7 和图 3.8 所示。

图 3.7 登录活动图



图 3.8 修改个人信息活动

3.1.2 商品选购设计

**（1）功能结构设计**

“商品选购”功能主要目的是为了提供用户购买商品的功能。此外，该功能还负责将商品加入购物车，对购物车进行管理。商品选购模块主要包括商品信息的查询获取以及购物车管理两个子模块。其中商品信息获取模块主要分为检索商品和查看商品详细信息两个组成部分。购物车管理又分为将商品放入购物车和将商品从购物车移除两个部分，其包图如图 3.9所示。



图 3.9 商品选购包图

**（2）类图设计**

图 3.10 商品选购类图

如上图3.10 所示是商品选购类图中,一共有三个类，商品类、商品类别类和购物车类。其中：商品类和商品类别类负责获取商品信息、商品列表的展示、检索商品和商品详情的展示等功能，商品类与商品类别类为多对一的对应关系；购物车列负责将商品加入购物车、将商品移出购物车和展示购物车等功能。

**（3）顺序图设计**

商品信息获取功能的顺序图如图 3.11 所示。



图 3.11 商品信息获取顺序图

购物车管理功能的顺序图如图 3.12 所示。



图 3.12 购物车管理顺序图

**（4）核心处理流程设计**

针对上述顺序图，图 3.11 中“根据关键字搜索商品”和图 3.12 中“添加购物车记录”这两个方法还可以进一步细化，如图 3.13 和图 3.14 所示。

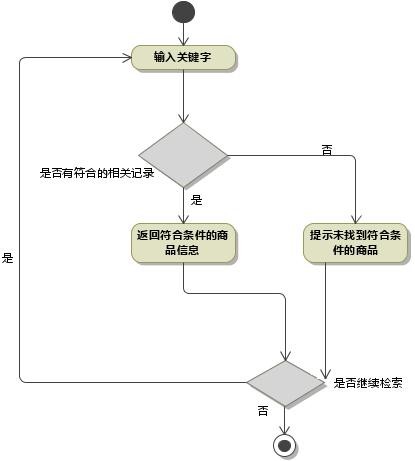


图 3.13 根据关键字搜索商品活动图

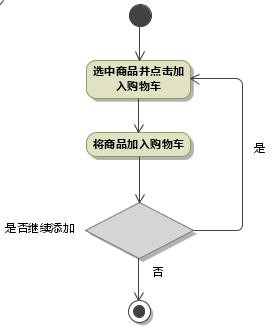


图 3.14 添加购物车记录活动图

3.1.3 商品管理设计

**（1）功能结构设计**

“商品管理”功能的主要目的是为了用户能上传自己的闲置物品，以提供其他用户购买的功能。此外还负责对自己上传的商品信息进行修改和下架商品。因此商品管理模块主要是商品信息的获取后对商品的增删改模块。其中商品获取模块主要分为展示用户商品列表和对商品的操作模块。其包图如图3.15所示。



图 3.15 商品管理包图

**（2）类图设计**



图 3.16 商品管理类图

如上图3.16所示的商品管理类图中，一共有2个类，其中：商品类负责获取商品信息、添加商品、修改商品和删除商品功能；商品类型类负责获取商品类型、添加类型、修改类型和删除类型功能；商品类与商品类型类为多对一的对应关系。

**（3）顺序图设计**

添加商品功能的顺序图如图3.17所示。

图 3.17 添加商品顺序图

查询用户商品列表功能的顺序图如图3.18所示



图 3.18 获取商品列表顺序图

**（4）核心处理流程设计**

针对上述顺序图，对图进行进一步细化。如图3.19和3.20所示。

图 3.19 添加商品活动图



图 3.20 商品获取活动图

3.1.4 订单管理设计

**（1）功能结构设计**

“订单管理”功能主要分为提交订单、取消订单和查询订单三部分。其中查询订单分为查看所有订单和查看单个订单信息两个部分。其包图如图3.21所示。

图 3.21 订单管理功能包图

**（2）类图设计**



图 3.22 订单管理类图

如上图3.22所示是订单管理类中，一共有两个类，其中订单类负责获取用户订单等功能；订单明细表负责提交订单和获取订单明细信息等功能。其中订单类与订单明细类为一对一的对应关系。

**（3）顺序图设计**

订单列表获取功能的顺序图如图3.23所示。



图 3.23 获取订单列表顺序图



订单明细获取功能的顺序图如图3.24所示。

图 3.24 获取订单明细顺序图

**（4）核心处理流程设计**

针对上述顺序图，对其进一步细化，如图3.25所示。



图 3.25 订单管理活动图

3.2 数据库设计

闲置物品交易系统的主要表的结构如下所示。[3]

（1）商品类别

保存商品类别的基本信息。如表 3.1 所示。

表3.1商品类别表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| TypeID | Int | Not | 主键 | 商品类型ID |
| TypeName | String | Not |  | 商品类型名 |

（2）商品信息

保存商品的基本信息。如表3.2 所示。

表3.2 商品表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| ProductID | Int | Not | 主键 | 商品类型ID |
| OwnerID | String | Not | 外键 | 卖家ID |
| ProductName | String | Not |  | 商品名 |
| ProductTypeID | Int | Not |  | 商品类型ID |
| Price | Double | Not |  | 单价 |
| Stock | Int | Not |  | 库存 |
| Description | String | Not |  | 描述 |
| Status | String | Not |  | 状态 |
| Image | String | Not |  | 图片 |
| Photo1 | String | Not |  | 细节图片1 |
| Photo2 | String | Not |  | 细节图片2 |
| Photo3 | String | Not |  | 细节图片3 |

（3）购物车记录

保存购物车记录的基本信息。如表 3.3所示。[4][5]

表3.3 购物车表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| ShoppingCartID | Int | Not | 主键 | 购物车ID |
| UserID | Int | Not | 外键 | 用户ID |
| ProductName | String | Not |  | 商品名 |
| ProductImg | String | Not |  | 商品图片 |
| ProductID | Int | Not |  | 商品ID |
| Number | Int | Not |  | 数量 |
| UnitPrice | Double | Not |  | 单价 |
| TotalPrice | Double | Not |  | 总额 |

（4）订单

保存订单项的基本信息。如表 3.4 所示。

表3.4 订单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| OrderID | Int | Not | 主键 | 订单ID |
| UserID | Int | Not | 外键 | 用户ID |
| ProductID | Int | Not | 外键 | 商品ID |
| TotalPrice | Double | Not |  | 总额 |
| Status | String | Not |  | 订单状态 |

（5）订单明细

保存订单的详细信息。如表 3.5 所示。

表3.5 订单明细表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| OrderID | Int | Not | 主键 | 购物车ID |
| UserID | Int | Not | 外键 | 用户ID |
| ProductID | Int | Not | 外键 | 商品ID |
| ProductName | String | Not |  | 商品名 |
| ProductImg | String | Not |  | 商品图片 |
| Number | Int | Not |  | 数量 |
| UnitPrice | Double | Not |  | 单价 |
| TotalPrice | Double | Not |  | 总额 |
| ReceivePeople | String | Not |  | 收件人 |
| Phone | String | Not |  | 手机号 |
| Address | String | Not |  | 地址 |
| OrderCreateTime | DateTime | Not |  | 创建时间 |

（6）用户表

保存用户的基本信息。如表 3.6 所示

表 3.6 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| UserID | Int | Not | 主键 | 用户ID |
| UserName | String | Not |  | 用户名 |
| Name | String | Not |  | 姓名 |
| Avatar | String | Not |  | 头像 |
| Email | String | Not |  | 邮箱 |
| Sex | String | Not |  | 性别 |
| Password | Strign | Not |  | 密码 |

（7）地址

保存用户的地址信息。如表 3.7 所示。

表 3.7 地址表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段名** | **数据类型** | **允许（null）值** | **约束** | **备注** |
| AddressID | Int | Not | 主键 | 地址ID |
| UserID | Int | Not | 外键 | 用户ID |
| Province | String | Not |  | 省份 |
| City | String | Not |  | 城市 |
| AddressDetail | String | Not |  | 详细地址 |
| IsDefault | String | Not |  | 是否默认 |

给出数据表之间的关系图，如图3.26所示。[6]



图3.26数据表关系图

**第 4 章 模块实现**

4.1 系统软件架构

4.1.1 系统开发架构

本系统遵循DDD（领域驱动设计）的原则，基于asp.net core的开发框架，将工程分为四个层，如图4.1所示。[7]



图4.1 系统架构图

如图4.1所示的系统架构中，具体的层次职责如下：

（1）展现层（Presentation）：提供用户界面，实现用户交互操作。

（2）Web层：提供应用程序编程接口（Web Api）。

（3）应用层（Application）：进行展现层与领域层之间的协调，协调业务对象来执行应用程序的任务。

（4）领域层（Domain）：领域层就是业务层，是一个项目的核心，所有的业务规则都应该在领域层实现。实体（Entity）代表业务领域的数据和操作，在实践中，通过用来映射成数据库表；仓储接口（IRepository）：仓储用来操作数据库进行数据存取。仓储接口在领域层定义。

（5）基础设施层（Infrastructure）：提供通用技术来支持更高的层。ORM工具使用EntityFrameWorkCore，仓储（Repository）：对领域层仓储接口的实现。

DDD（领域驱动设计）架构中，Presentation（展示层）、Web层(Api)、Application（应用层）、Domain（领域层）和Infrastructure（基础设施层）各司其职，意在职责分离。Infrastructure层的Repository实现Domain层的IRepository接口，相当于IDAL和DAL，通过Infrastructure层ORM来实现与数据库的交互，Web层Api来接收这些数据，Presentation层的HTML通过请求API资源来渲染页面。

闲置物品交易系统实现的工程文件结构图如图4.2所示。

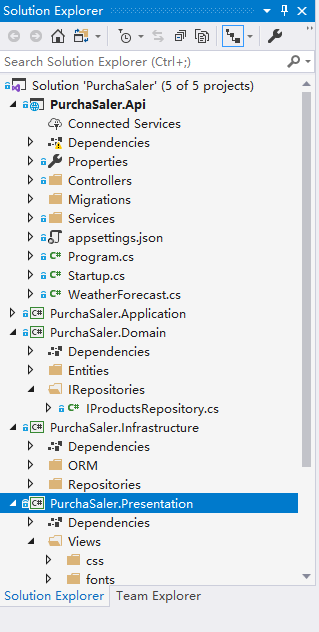


图4.2工程文件结构图

4.1.2 .net core框架全局配置appsetting.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"ConnectionStrings": {

"constr": "Data Source=localhost;Database=PurchaSalerV1;User id=sa;Password=Pass@word"

},

"JwtSetting": {

"issuer": "PurchaSaler.com",

"audience": "PurchaSaler.com",

"secretkey": "rtryuiqnjkxnjjkslakjbjnzxhhfulzxl"

},

"AllowedHosts": "\*"

}

4.1.3 StartUp配置

**StartUp配置：**连接字符串、注册JWT服务、注册跨域服务、注册Swagger Api文档服务和一些默认配置。[8]

namespace PurchaSaler.Api

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddTransient<IProductsRepository, ProductsRepository>();

services.AddTransient<IOrdersRepository, OrdersRepository>();

services.AddTransient<IOrderItemsRepository, OrderItemsRepository>();

services.AddTransient<IShoppingCartsRepository, ShoppingCartsRepository>();

services.AddTransient<IUsersRepository, UsersRepository>();

services.AddControllers();

//jwt验证

var key = Encoding.UTF8.GetBytes(Configuration["JwtSetting:secretkey"]);

services.AddAuthentication(option =>

{

option.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;

option.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;

option.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;

}).AddJwtBearer(option =>

{

option.RequireHttpsMetadata = false;

option.SaveToken = false;

option.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key),

ValidateIssuer = false,

ValidateAudience = false,

ClockSkew = TimeSpan.Zero

};

});

//注册Cors服务

services.AddCors(option =>

{

option.AddPolicy("any", buider =>

{

buider.AllowAnyOrigin()

.AllowAnyMethod()

.AllowAnyHeader();

});

});

services.AddDbContext<PurchaSalerDbContext>(option => option.UseSqlServer(

Configuration.GetConnectionString("constr"),b=>b.MigrationsAssembly("PurchaSaler.Api")));

services.AddSwaggerGen(option =>

{

option.SwaggerDoc("v1.1", new OpenApiInfo

{

Title="PurchaSaler",

Version="v1.1"

});

});

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseRouting();

app.UseCors();

app.UseAuthentication();

app.UseAuthorization();

app.UseStaticFiles();

app.UseSwagger();

app.UseSwaggerUI(option =>

{

option.SwaggerEndpoint("/swagger/v1.1/swagger.json", "PurchaSaler");

});

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

4.2 用户管理实现

4.2.1 设计思想

前后端分离，会员通过注册可以拥有自己的账号，通过登录账号可以进入个人中心进行个人资料修改。登录时服务端产生token，客户端保存token。在访问相应资源时，请求头带上token，实现权限访问。

4.2.2 操作界面

注册界面清新简洁，没有相当复杂的步骤，对于新入网站的游客朋友非常友好，简约而不简单。如图4.3注册界面所示。

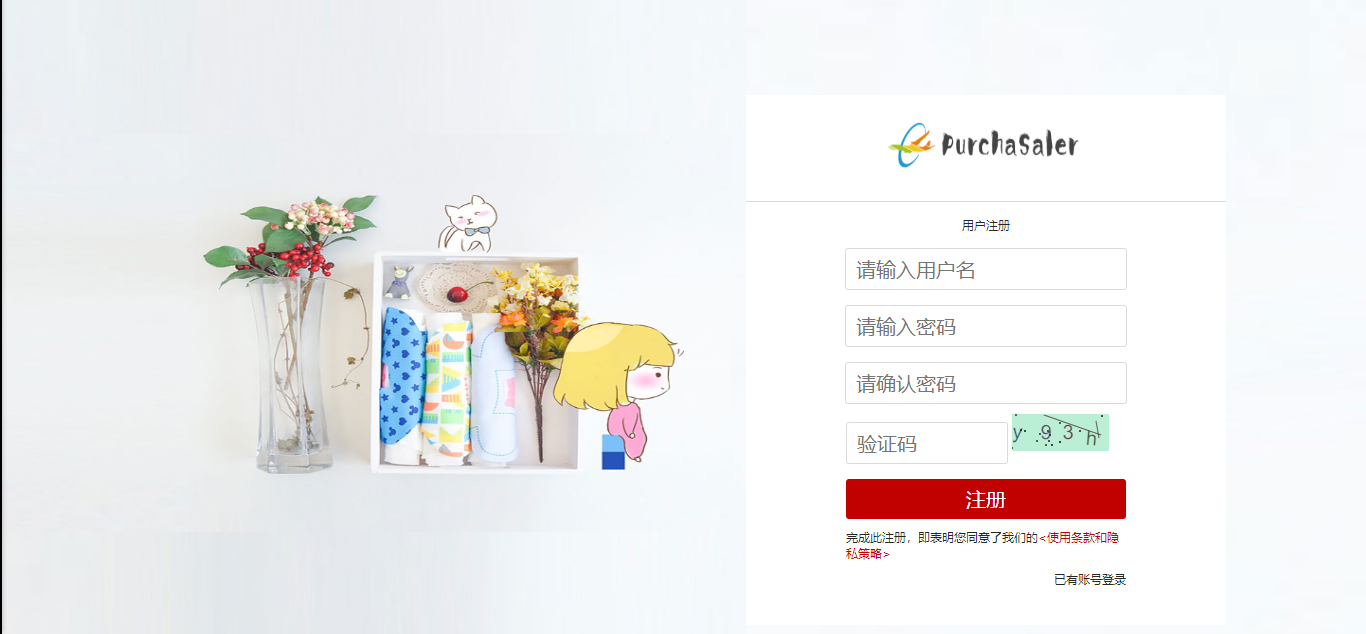


图4.3 注册界面

登录界面简单自然，一目了然。如图4.4登录界面所示。

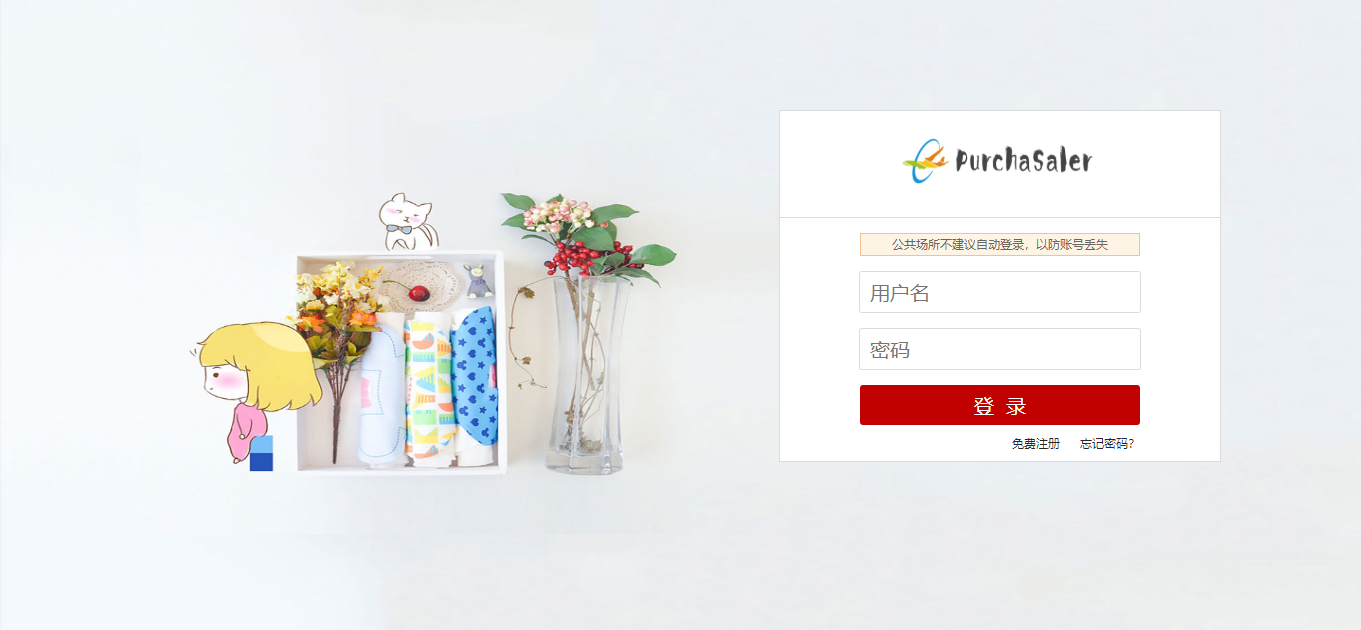


图4.4登录界面

个人资料修改界面层次分明。如图4.5个人资料修改界面所示。

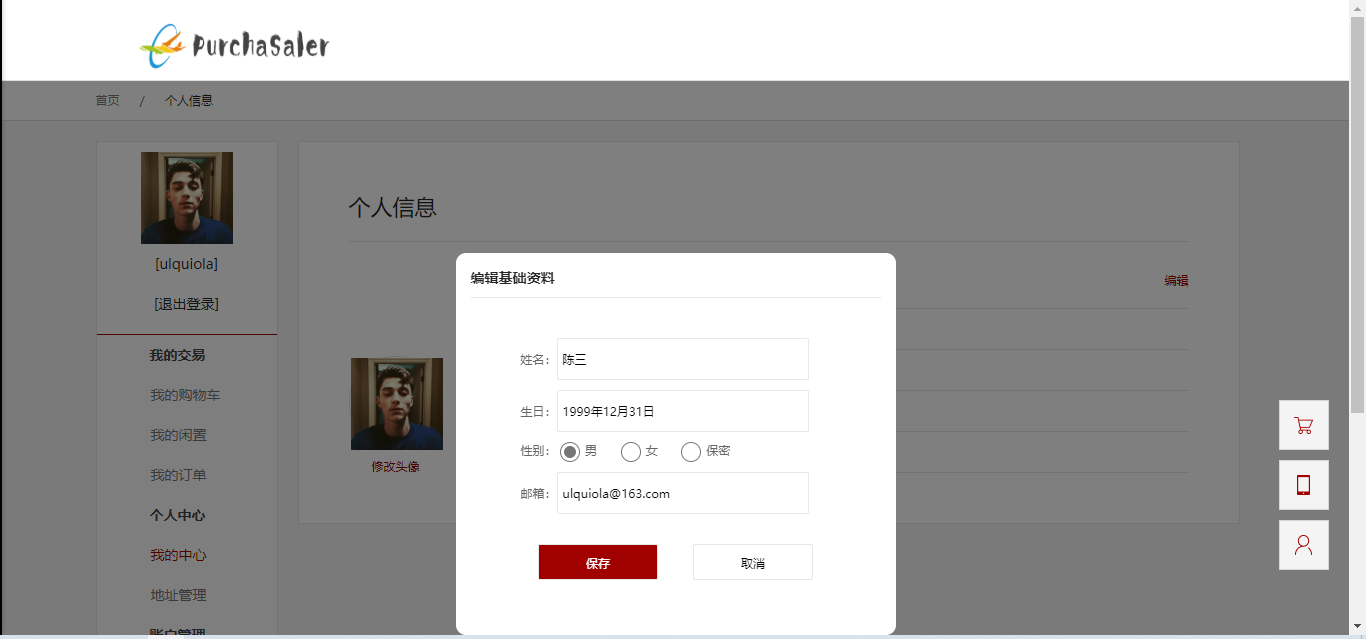


图4.5 个人资料修改界面

4.2.3 核心代码

**跨域处理**：由于一个域下的资源试图访问另一个域下的资源的跨域请求会被拒绝，所以需要建立一个跨域类CorsController继承ControllerBase,作为基类，加入开启跨域策略特性，其他Api继承跨域类CorsController，实现全局跨域。

namespace PurchaSaler.Api.Controllers

{

    [EnableCors("any")]

    [Route("api/[controller]")]

    [ApiController]

    public class CorsController : ControllerBase

    {

    }

}

services.AddCors(option =>

            {

                option.AddPolicy("any", buider =>

                 {

                     buider.AllowAnyOrigin()

                     .AllowAnyMethod()

                     .AllowAnyHeader();

                 });

            });

app.UseCors();

**JsonWenToken处理：**用户使用用户名密码来请求服务器，服务器进行验证用户的信息，服务器通过验证发送给用户一个token，客户端存储token，并在每次请求时附送上这个token值，服务端验证token值，并返回数据。[9]

namespace PurchaSaler.Api.Services

{

    public class JsonWebToken

    {

        private readonly IConfiguration \_config;

        public JsonWebToken(IConfiguration config)

        {

            \_config = config;

        }

        public string GenerateToken(Users userinfo)

        {

            var tokenDescriptor = new SecurityTokenDescriptor

            {

                Subject = new ClaimsIdentity(new Claim[]

                {

                    //使用Name存储用户id,以便取出

                    new Claim(ClaimTypes.Name, userinfo.UserID.ToString())

                }),

                Expires = DateTime. UtcNow.AddDays(1),

                SigningCredentials = new SigningCredentials(

                    new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["JwtSetting:secretkey"])),

                    SecurityAlgorithms.HmacSha256Signature)

            };

            var tokenHandler = new JwtSecurityTokenHandler();

            var securityToken = tokenHandler.CreateToken(tokenDescriptor);

            var token = tokenHandler.WriteToken(securityToken);

            return token;

        }

    }

}

**用户注册：**服务端接收用户客户端传递过来的用户名和密码，验证通过后，将密码通过Md5加密存入数据库，实现用户注册。[10]

[HttpPost("Register")]

public IActionResult Register(Users user)

{

bool IsExisted = \_usersRepository.IsExisted(user.UserName);

if(ModelState.IsValid && !IsExisted)

{

//哈希密码

user.Password = EncryptProvider.Md5(user.Password);

var man = new Users()

{

//UserId 主键自增

UserName = user.UserName,

Name = "",

Avatar = "img/defaultAvt.png",

Birthday = DateTime.Now.ToString("D"),

Sex = "保密",

Email = "",

Password = user.Password

};

\_usersRepository.AddUser(man);

return Ok("注册成功");

}

else

{

return StatusCode(422,"用户已存在");

}

}

**用户登录：**接收客户端传递过来的用户信息，服务端将用户输入的密码通过Md5算法加密，然后取出用户数据库密码与之匹配，匹配密码成功后，服务端通过JsonWebToken类生成token，客户端存储服务端返回的token。

[HttpPost("Login")]

public IActionResult Login(LoginVM user)

{

if (ModelState.IsValid)

{

bool IsExisted = \_usersRepository.IsExisted(user.UserName);

//用户是否存在

if(IsExisted)

{

//取出这个人的信息

var man = \_usersRepository.GetUserByName(user.UserName);

//验证密码

if (EncryptProvider.Md5(user.Password) == man.Password)

{

//生成token

var jwt = new JsonWebToken(\_config);

var token = jwt.GenerateToken(man);

return new JsonResult(token);

}

else

{

return StatusCode(422, "密码错误");

}

}

else

{

return StatusCode(422, "用户不存在");

}

}

else

{

return StatusCode(422, "实体验证失败");

}

}

**用户信息修改：**使用Authorize特性，服务端接收客户端的用户信息修改请求，若请求头中token无效则拒绝访问。

[Authorize]

[HttpPost("UpdateInfo")]

public IActionResult UpdateInfo(UpdateInfoVM infoVM)

{

var claimsIdentity = User.Identity as ClaimsIdentity;

var userid = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

var man = \_usersRepository.GetUserByID(userid);

man.Name = infoVM.Name;

man.Birthday = infoVM.Birthday;

man.Email = infoVM.Email;

man.Sex = infoVM.Sex;

\_usersRepository.ModifyUser(man);

return Ok();

}

**图片上传：**头像的上传和修改，使用Authorize特性，服务端接收客户端的请求，若请求头中token无效则拒绝访问。

[Authorize]

[HttpPost("UpdateAvatar")]

public IActionResult UpdateAvatar(IFormFile file)

{

var claimsIdentity = User.Identity as ClaimsIdentity;

var userid = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

var man = \_usersRepository.GetUserByID(userid);

var upload = new UploadImages(\_environment);

string avatar = upload.Upload(file);

man.Avatar = avatar;

\_usersRepository.ModifyUser(man);

return Ok();

}

**图片文件上传处理：**图片文件上传到服务端，将图片文件copy到服务端静态文件目录，并返回静态图片地址。

public class UploadImages

{

private readonly IWebHostEnvironment \_environment;

public UploadImages(IWebHostEnvironment environment)

{

\_environment = environment;

}

public string Upload(IFormFile file)

{

try

{

if (file.Length > 0)

{

if (!Directory.Exists(\_environment.WebRootPath + "\\images\\"))

{

Directory.CreateDirectory(\_environment.WebRootPath + "\\images\\");

}

using (FileStream fileStream = File.Create

(\_environment.WebRootPath + "\\images\\" + file.FileName))

{

file.CopyTo(fileStream);

fileStream.Flush();

return "http:\\\\localhost:5000"+"\\images\\" + file.FileName;

}

}

else

{

return "Failed";

}

}

catch (Exception ex)

{

return ex.Message.ToString();

}

}

4.3 商品选购实现

4.3.1 设计思想

商品展示，用户通过点击可查看商品的详情，登录用户在商品详情页可将商品加入购物车和直接购买结算。

4.3.2 操作界面

购物车界面清新简洁，布局良好。将ajax请求的用户购物车列表数据通过vuejs渲染出来，如图4.6购物车界面所示。

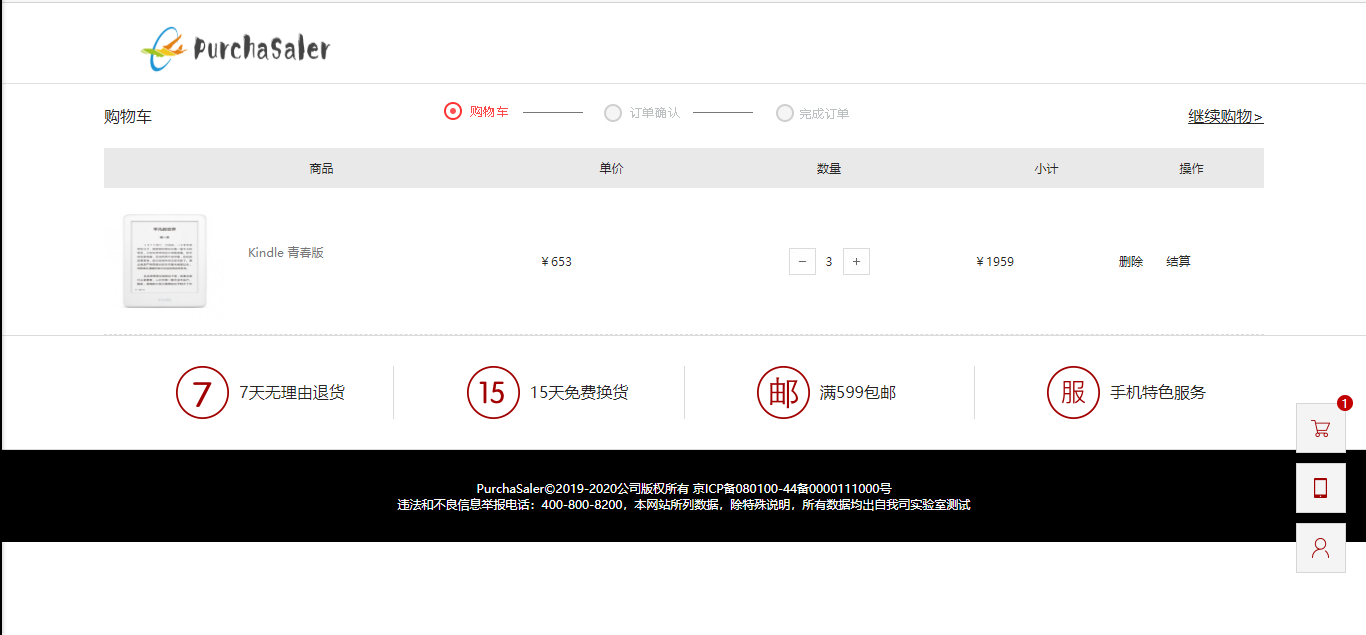


图4.6 购物车界面

商品详情界面通过ajax请求服务端商品详情数据，通过vuejs将数据渲染出来。如图4.7购物车界面所示。



图4.7商品详情界面

4.3.3 核心代码

**商品详情展示：**客户端向服务请求商品详情，服务端通过商品id，在数据库查询商品，返回商品详情信息。

[HttpGet("ProductDetail")]

public IActionResult GetProductDetail(int productid)

{

var product = \_productsRepository.GetProductByID(productid);

return new JsonResult(product);

}

**获取当前登录用户的购物车列表：**服务端接收客户端的获取当前用户购物车列表请求，若请求头中token无效则拒绝访问，客户端解析token，取出当前登录用户的id，在数据库查询用户购物车信息，返回购物车列表。

[Authorize]

[HttpGet("GetCarts")]

public IActionResult GetCarts()

{

var claimsIdentity = User.Identity as ClaimsIdentity;

int userid = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

var shoppingcart = \_shoppingCartsRepository.GetSomeOneAllShoppingCarts(userid);

return new JsonResult(shoppingcart);

}

**添加购物车：**服务端接收客户端的添加购物车请求，若请求头中token无效则拒绝访问，将服务端传递的购物车信息添加进数据库表中，实现添加购物车操作。

[Authorize]

[HttpPost("AddShoppingCarts")]

public IActionResult AddShoppingCarts([FromBody]ShoppingCarts shoppingCart)

{

var claimsIdentity = User.Identity as ClaimsIdentity;

int userid = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

int productid = shoppingCart.ProductID;

var cart = new ShoppingCarts()

{

//ShoppingCartID 自增

UserID = userid,

ProductID = productid,

ProductName = shoppingCart.ProductName,

ProductImg = shoppingCart.ProductImg,

Number = shoppingCart.Number,

UnitPrice = shoppingCart.UnitPrice,

TotalPrice = shoppingCart.TotalPrice,

};

int result = \_shoppingCartsRepository.GetShoppingCartsCount(userid,productid);

if (result > 0)

{

return StatusCode(403,"商品已存在与购物车");

}

else

{

\_shoppingCartsRepository.AddShoppingCarts(cart);

return Ok("添加成功");

}

}

**移除购物车：**服务端接收客户端的移除购物车请求，若请求头中token无效则拒绝访问，通过对象的方式传递参数，解决参数为空的问题。服务端接收客户端传递过来的购物车id，实现移除购物车操作。

[HttpPost("Remove")]

public ActionResult Remove([FromBody]RequestId id)

{

int cartid = id.Id;

var delobj = \_shoppingCartsRepository.GetOneShoppingCart(cartid);

if (delobj != null)

{

\_shoppingCartsRepository.RemoveShoppingCarts(delobj);

return Ok("删除成功");

}

return StatusCode(404);

}

4.4 商品管理实现

4.4.1 设计思想

用户登录后可以进入个人中心，发布自己的闲置物品，并对自己发布的商品进行编辑修改。

4.4.2 操作界面

商品发布界面样式自然清新，方便用户理解。图片上传通过ajax上传到服务端，服务端返回图片地址，将用户输入的商品信息以及图片地址传递到服务端请求发布商品。如图4.8商品发布界面所示。

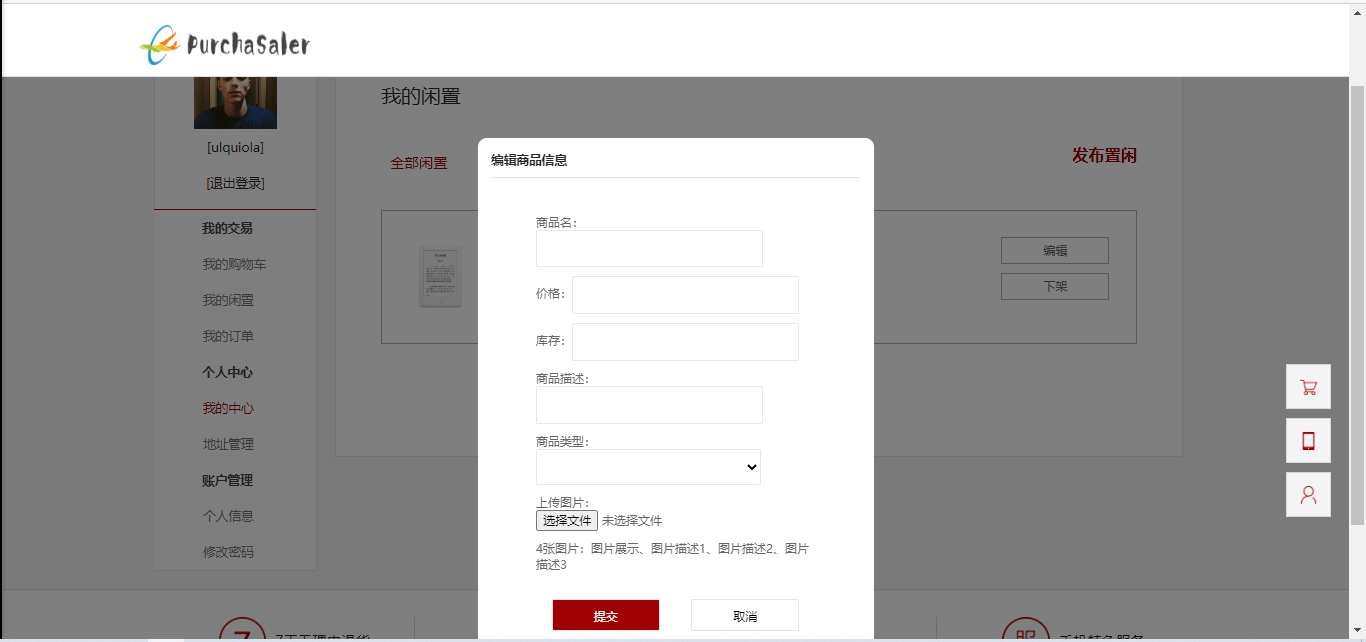


图4.8商品发布界面

4.4.3 核心代码

**商品发布：**服务端接收客户端的发布商品请求，若请求头中token无效则拒绝访问，服务端将客户端传递过来的商品信息添加到数据库，并返回结果。

[Authorize]

[HttpPost("AddProduct")]

//[FormBody]要加，不然报400错误

public IActionResult AddProduct([FromBody]AddProductVM productVM)

{

//all=12

var claimsIdentity = User.Identity as ClaimsIdentity;

var product = new Products();

//主键自增

product.OwnerID = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

product.ProductName = productVM.ProductName;

product.ProductTypeID = Convert.ToInt32(productVM.ProductTypeID);

product.Description = productVM.Description;

product.Price = Convert.ToDouble(productVM.Price);

product.Status = "";

product.Stock = Convert.ToInt32(productVM.Stock);

product.image = productVM.image;

product.photo1 = productVM.photo1;

product.photo2 = productVM.photo2;

product.photo3 = productVM.photo3;

\_productsRepository.AddProduct(product);

return Ok();

}

**商品修改：**服务端接收客户端的商品修改请求，若请求头中token无效则拒绝访问，将修改的商品信息对比数据库，如果商品信息状态被改变，则保存到数据库，实现商品修改操作。

[Authorize]

[HttpPost("UpdateProduct")]

public IActionResult UpdateProduct([FromBody]UpdateProductVM productVM)

{

var product = new Products();

product.ProductID = Convert.ToInt32(productVM.ProductID);

product.OwnerID = Convert.ToInt32(productVM.OwnerID);

product.ProductTypeID = Convert.ToInt32(productVM.ProductTypeID);

product.ProductName = productVM.ProductName;

product.Description = productVM.Description;

product.Price = Convert.ToDouble(productVM.Price);

product.Status = "";

product.Stock = Convert.ToInt32(productVM.Stock);

product.image = productVM.image;

product.photo1 = productVM.photo1;

product.photo2 = productVM.photo2;

product.photo3 = productVM.photo3;

\_productsRepository.UpdateProduct(product);

return Ok("修改成功");

}

**商品下架：**服务端接收客户端的商品下架请求，若请求头中token无效则拒绝访问，接收客户端传递过来的商品id，数据库移除商品，实现商品下架操作。

[Authorize]

[HttpPost("DelProduct")]

public IActionResult DelProduct([FromBody]RequestId id)

{

var pid = id.Id;

\_productsRepository.DelProducts(pid);

return Ok("删除成功");

}

4.5 订单管理实现

4.5.1 设计思想

登录用户在选购商品详情页点击购买商品或对购物车商品点击购买，实现生成订单功能。登录用户在个人中心订单项可以查看自己的订单列表。[11]

4.5.2 操作界面

通过订单结算界面，用户可以选择已有收件信息或添加新收件信息，点击支付，验证通过后，实现订单结算。如图4.9订单界面所示。

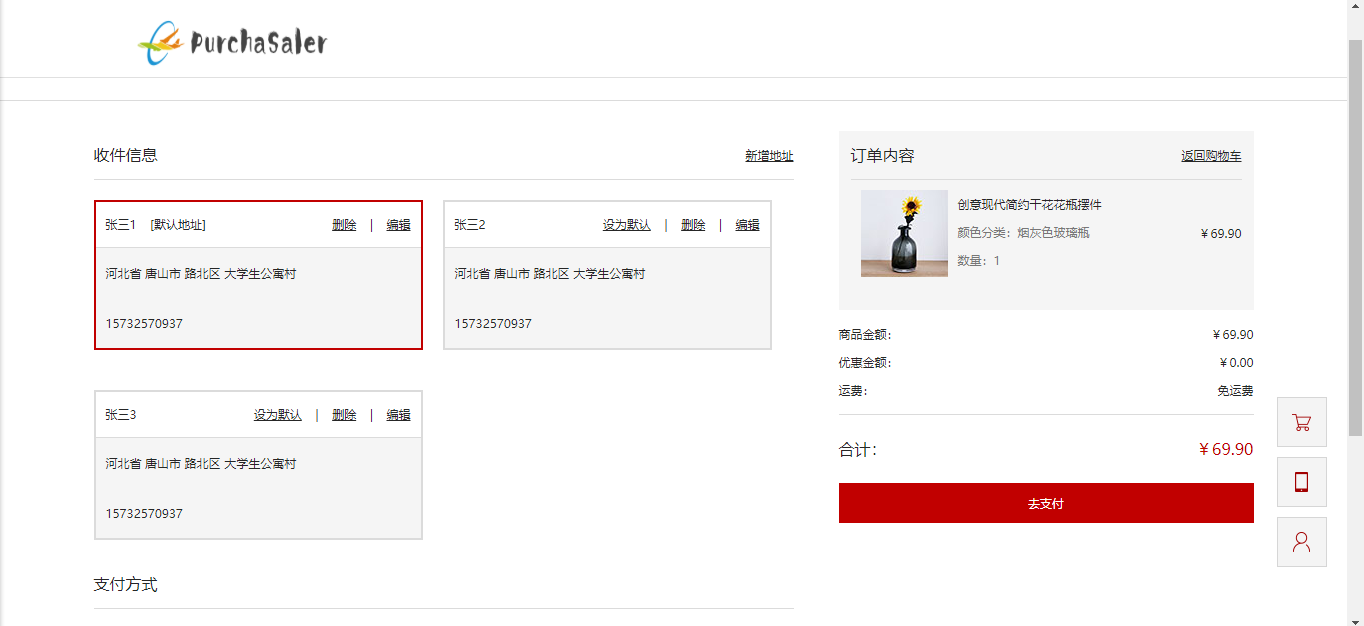


图4.9订单结算界面

订单列表界面一目了然，方便用户查看自己的订单项。获取本地客户端存储的localStorage['token'],将token加入的订单列表请求头中，服务端返回对应订单列表数据，界面渲染数据。如图4.10订单列表界面所示。



图4.10订单列表界面

订单详情界面采用统一样式渲染，观感良好。将服务端返回的订单数据通过渲染生成订单详情界面。如图4.11订单详情界面所示。



图4.11订单详情界面

4.5.3 核心代码

**获取订单列表：**接收客户端的获取用户订单列表请求，加入Authorize特性，若请求头中token无效则拒绝访问，客户端解析token，取出当前登录用户的id，在数据库查询用户订单信息，返回订单列表信息。

[Authorize]

[HttpPost("GetOrders")]

public IActionResult GerOrders()

{

var claimsIdentity = User.Identity as ClaimsIdentity;

var userid = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

var orders = \_ordersRepository.GetSomeOneOrders(userid);

return new JsonResult(orders);

}

**生成订单：**接收客户端的生成订单请求，若请求头中token无效则拒绝访问，服务端接收客户端传递的订单数据，通过验证后，将生成订单所需数据添加入数据库中，实现生成订单操作。

[Authorize]

[HttpPost("GenerateOrders")]

public IActionResult GenerateOrders([FromBody]Orders orders)

{

//all=10

var claimsIdentity = User.Identity as ClaimsIdentity;

var obj = new Orders();

//主键自增

obj.UserID = Convert.ToInt32(claimsIdentity.FindFirst(ClaimTypes.Name)?.Value);

obj.ProductID = orders.ProductID;

obj.ProductName = orders.ProductName;

obj.ProductImg = orders.ProductImg;

obj.Number = orders.Number;

obj.UnitPrice = orders.UnitPrice;

obj.TotalPrice = orders.TotalPrice;

obj.OrderCreateTime = DateTime.Now;

obj.Address = orders.Address;

\_ordersRepository.AddOrders(obj);

return Ok();

}

**删除订单：**接收客户端删除订单列表请求，加若请求头中token无效则拒绝访问，前端以json格式传递参数，后端以对象接收，解决参数传递不过来和为空的问题。将用户选中的订单，通过订单号在数据库中删除相应数据，实现订单删除操作。

[Authorize]

[HttpPost("DelOrder")]

public IActionResult DelOrder([FromBody]RequestId id)

{

var orderid = id.Id;

\_ordersRepository.DelOrder(orderid);

return Ok("删除成功");

}

**第 5 章 总结和展望**

5.1 总结

本闲置物品交易系统基本实现了用户管理、商品选购、商品管理和订单管理等基本功能。本系统可以为闲置物品的处理提供渠道，用户可以在本系统发布自己的闲置物品，通过交易可以以物换钱，实现闲置物品的处理。用户亦可以在系统查询自己需要的东西，通过交易能以较低的价格获取自己想要的物品。

通过这段时间的对闲置物品交易系统的设计与实现，从中获益良多。

1、研究了前后端分离技术、http网络请求、.net core Api技术、Entityframeworkcore 技术、JsonWebToken技术、依赖关系注入等技术，从而为系统原型的实现提供了解决方案。

2.在研究设计的基础之上对系统进行实现，该系统大体上实现了闲置物品交易系统设计方案中所提到的用户管理、商品选购、商品管理和订单管理4个模块的功能。

3.对系统进行测试，尽管在测试过程中还有一些尚未解决的问题，但是这些都是预期可以解决的。

回顾这段时间的系统开发工作，总结起来那就是，软件的开发是相当辛苦的，由于是个人项目，所有的内容都自己包揽，但成功以后的喜悦也是非常美妙的，而且我发现，其中你投入的心血越多，成功以后你所获得的快乐与充实感也更多更强。

5.2 展望

本闲置物品交易系统的大部分模块已经实现，但是，一个完整的系统仅仅靠近几个月的分析、设计来开发是远远不够的，它需要更长的时间来测试并完善。本系统的设计与实现设计多方面的理论、方法和技术，需要在实际应用中不断积累和完善，因此，本系统仍然存在着一些需要改进和不足的问题，需要在以下几个方面还需做进一步的研究和开发。

1、界面的清晰，简单设计。界面的清晰简单可以使用户一目了然，本系统的学习成本很低，轻松的找到自己想要的操作。不然则会浪费用户时间，甚至打消用户的积极性，对于网站的好感度大大降低。所以系统的整体设计需要进一步的美观，网页交互界面也需要进一步完善。

2、系统的拓展性。社会是不断进步与发展的，对于用户来说，系统的拓展性是很重要的，随着时代的进步，用户对于系统的要求也会越来越高。因此系统要随着用户的需求而不断发生变化，不断更新系统的内容来满足用户更多的需求。未来将增加管理员模块，系统还需要进一步的改善和加强，系统的内容也需要更加丰富和多样化。

3、系统的可移植性。系统需要考虑能够运行在不同的平台上，本系统采用前后端分离，前端通过浏览器都能访问，后端采用 dotnet core 技术，具有跨平台的特点，现阶段是在适配 Windows 平台，后续会将系统迁移至Linux系统平台。

4、技术的更新换代。.NET5 即将发布，未来会考虑Asp.net core3.1 技术迁至.NET5。

时光荏苒，大学生活即将接入尾声，希望本系统能将我大学中学到的知识得到展现。阴霾终将散去，春天必定到来，未来可期，新的阶段，意味着新的起点，新的机遇，新的挑战。希望用我所学，在新的阶段中让自己更上一层楼。

**参考文献**

[1] 庞敏.基于UML的电子商务系统设计[J].电子设计工程，2017-05-05.

[2] 程杰.大话设计模式 [M].北京：清华大学出版社，2019.

[3] 牛小宝.基于MySQL的云数据库设计与实现[D].南京:南京邮电大学，2016.

[4] 胡强. Web服务流程的结构范式及其判定算法[J].计算机学报，2015-01-15.

[5] 崔旭冉.基于SQLServer的标准化考试系统设计与实现[J].电脑知识与技术，2019-02-15.

[6] 王珊，萨师煊. 数据库系统概论(第5版) [J].中国大学教学，2018-05-15.

[7] 陈鹏.基于.Net Web Api的物流订单软件的设计与实现[J].福建电脑，2016.04-25

[8] 孙继红.基于.NET三层架构的网上商城设计与实现[J].电脑编程技巧与维护，2008-11-18.

[9] 李文清,杨庆红.数据结构（C语言版） [M].北京：人民邮电出版社，2016.

[10] 王晓东.计算机算法设计与分析 [M].北京：电子工业出版社，2018.

[11] Design and Implementation of Photographic Community System Based on ASP.NET MVC [J]. Jiangxi Normal University.2019.

**致谢**

首先，我要诚挚的感谢我的导师徐虹老师，在闲置物品交易系统的设计与实现过程中，得到了徐虹老师的悉心指导，在论文的写作过程中，多次得到她的督促，并且她为我的论文指出了许多宝贵的修改意见。徐虹老师严谨治学的态度与求实的工作作风及丰富的学识给我留下深刻的印象，使我受益匪浅。

其次，我要感谢在这次系统的设计与开发过程中给予我帮助的同学、朋友，在他们的帮助下，我终于圆满的完成了系统的设计与实现工作，再次向他们表示深深的感谢。

最后，我要感谢几年来给我们上课的老师以及学院的各位工作人员，正是有他们的辛勤劳动，才使我有足够的能力完成整个项目，再次向他们表示真诚的谢意。