廣東工業大學

本科毕业设计（论文）

基于 MEAN 的校园二手交易系统

学　　院　计算机学院

专　　业　计算机科学与技术

年级班别　13 级（2）班

学　　号　3213005859

学生姓名　沈晖莹

指导教师　饶东宁

2017 年 5 月

基于 MEAN 的校园二手交易系统　　沈晔莹　　计算机学院

摘　　要

随着科技进步和生活水平的提高，人们拥有的生活物资也越来越多。与此同时，人们理性购物的思想也在提高，二手物品以其低廉的价格越来越受到人们的欢迎。近年来，很多大学校园兴起了二手物品交易的热潮，学生们通过跳蚤市场或网络来买卖二手物品。随着网络技术的迅猛发展，许多人在学校贴吧和微信交流群等网络交流方式发布信息变卖物品，但是这种方法有着受众范围小、不便于查找信息等问题。校园二手交易系统正是基于上述的问题而开发的。

本系统是采用 MEAN 技术栈开发的基于 B/S 架构的校园二手交易平台。本系统包括了用户模块、物品模块、交易模块这三个功能模块。游客通过注册成为本站用户后，既可以发布二手物品信息，也可以发布物品的求购信息，满足了用户的多层需求。物品模块实现了发布、管理、关键字查找等功能，交易模块实现了发出交易请求和处理交易请求的功能。整个系统构造了一个校园内网上交流信息的平台，利用网络的优势更好地实现了资源整合。

关键词：校园二手交易，MEAN，Express，Node.js，MongoDB，AngularJS

Abstract

With the progress of science and technology and the improvement of living standards, people have more and more living materials. At the same time, people's thinking of rational shopping is also improving, while second-hand items with its low price are getting more popular. In recent years, many university campuses have risen a second-hand goods trading boom. The students through the flea market or network to buy second-hand items. With the rapid development of network technology, many people in the school through the way of Post Bar and WeChat exchange group and other network communication way to release information and sell items, but this method has a series of problems, such as a small audience range, not easy to find information and so on. Campus second-hand trading system is based on the above problems and development.

This system of the campus second-hand trading platform which is developed by MEAN technology stack is based on the B/S architecture. The system consists of the user module, the item module and the transaction module. After becoming users of this site through registration, visitors can both publish second-hand goods information and publish items of purchase information to meet the user's multiple needs. Item module to achieve the release, management, keyword search and other functions, the transaction module to achieve the transaction request and deal with the transaction request function. The whole system to build a campus online exchange of information platform, the use of the advantages of the network to better achieve the integration of resources.

Key words: Campus second-hand transactions，MEAN，Express，Node.js，MongoDB，AngularJS

目　　录

1 绪论

1.1 课题背景

当前，随着人们生活水平的提高，生活物资也不断增多，同时由于各种原因，闲置物品也会不断增加。尤其在大学校园中，伴随着学生购买能力的提高和每年的升学毕业，各种类型的二手物品也越来越多。但是由于缺乏管理二手物品的时间和精力，二手交易市场的举办次数少之又少。并且二手交易市场存在着规模小、信息滞后、持续时间短等种种限制[1]，导致未能实现二手物品的有效流转。与此同时，随着信息科技社会的发展，人们越来越热衷并依赖于通过网络渠道获取信息。在网络时代下，如果将传统的二手交易市场与现代的网络科技结合在一起，将会更便于用户进行二手交易。加上学生群体具有地域集中、需求物品相似等有利条件，创建一个校园二手交易系统是一个可行的方案。

1.2 目的和意义

为卖家更好的解决二手物品的堆积问题和管理问题，给买家提供一个发布购买需求的平台，为需要的买家提供廉价、有用的二手物品，使用户足不出户就可以了解到最新的二手物品信息，促进校园二手物品交易的有效和高效进行。

在大学校园中，存在着很多的二手物品，但是由于信息不流通以及传统二手物品信息交流方式的笨拙，导致了很多仍然具有一定价值或非常具有价值的二手物品的囤积，乃至被当作废弃物处理。现在通过进入系统，可以方便快捷的发布和浏览任何二手物品的信息。由于网络的普及使用，因此，只要宣传有效，每个人都可以让他所发布的信息让全校了解，争取信息资源的最大化利用[2]。不仅可以将一些同学们平时闲置的物品变废为宝，而且也促进了人与人之间的交流，拉近了同学们之间的距离[3]，同时还为卖东西的用户带来了直接的经济收入，为买东西的用户提供了具有一定质量保障但价格低廉的物品。

1.3 主要研究内容及工作任务

针对校园二手交易进行调研分析，确定当前要解决的问题，根据用户的网络使用习惯及应用平台，通过具体的需求分析建立校园二手网络交易系统的基础架构模型，最终

确认采用基于 B/S 架构的设计方案，使用 Web 开发的相关技术体系设计一个相对完善的校园二手交易系统。

“基于 MEAN 的校园二手交易系统”主要使用了 MEAN 技术栈进行开发，包括 Node.js 服务器、Express 框架、MongoDB 数据库以及 AngularJS 框架。研究的主要内容如下：

（1）校园二手交易平台的用户特征和功能需求

（2）数据库设计及系统框架的搭建

（3）针对需求，分析设计系统的功能模块，实现校园二手交易的主要功能

本文通过实际调查走访以及网络调查等多种方法来获取需求，并通过快速模型建立的方法进一步细化需求，通过用例图、数据流图及 E-R 图等方法对系统进行详细设计。对各功能模块实现中的细节和难点进行了详细讨论与分析，得出以下主要工作任务：

1、校园二手交易系统的设计与实现开发背景，分析高校二手交易现状的存在问题及进一步发展的必要性，从各方面对系统进行可行性分析和研究

2、系统需求的总体描述。通过使用用例图、数据流图等方式描述系统的功能需求，分析系统的各个功能模块，选择开发技术及部署开发和运行环境。

3、系统详细设计与实现。从数据库设计、服务器搭建、客户端页面实现这三方面逐步实现系统的主要功能。

4、系统测试。针对系统的主要功能进行实例测试，并对结果进行分析和总结。

1.4 本文的结构

本文档第 1，2 章介绍课题背景和相关技术，第 3 章进行可行性分析及需求分析，并介绍 MEAN 技术栈的优势。第 4，5 章分别是系统详细设计和实现。最后第 6 章展示测试示例和结果分析。

2 相关技术简介

2.1 MEAN 技术栈

MEAN 是一个强大的 JavaScript 全栈解决方案，它由四大组件组成：数据库 MongoDB、Web 服务器框架 Express、Web 客户端框架 AngularJS，以及服务器平台 Node.js。其理念是仅使用 JavaScript 一种语言来驱动整个应用[4]。其最鲜明的特点有以下几个：

（1）整个应用只使用一种语言；

（2）整个应用的所有部分都支持 MVC 架构，而且都必须使用 MVC 架构；

（3）不再需要对数据结构进行串行化和并行化操作，只需要使用 JSON 对象来进行数据封装即可。

2.2 Node.js

Node.js 是 MEAN 中的 N，它是一个让 JavaScript 访问各种本地 API 和网络 API 的运行环境。实际上它是对 Google V8 引擎进行了封装。V8 引擎执行 JavaScript 的速度非常快，性能非常好。Node.js 对一些特殊用例进行了优化，提供了替代的 API，使得 V8 在非浏览器环境下运行得更好。

Node.js 是一个基于 Chrome JavaScript 运行时建立的平台，用于方便地搭建响应速度快、易于扩展的网络应用。Node.js 使用事件驱动、非阻塞 I/O 模型而得以轻量和高效，非常适合在分布式设备上运行数据密集型的实时应用[5]。

2.3 Express

使用于 Web 应用：Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，它提供一系列强大的特性，帮助创建各种 Web 和移动设备应用。

API：丰富的 HTTP 快捷方法和任意排列组合的 Connect 中间件，让创建健壮、友好的 API 变得既快速又简单。

性能：Express 不对 Node.js 已有的特性进行二次抽象，只是在它之上扩展了 Web 应用所需的基本功能。

2.4 MongoDB

存储和使用数据的能力对于大多数应用程序至关重要。在 MEAN 堆栈中，选择的数据库是 MongoDB，MEAN 中的 M。MongoDB 非常适合堆栈。像 Node.js 一样，它是快速和可扩展的。

MongoDB 是一个基于分布式文件存储的数据库，旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。[6]

MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似 json 的 bson 格式，因此可以存储比较复杂的数据类型。Mongo 最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

MongoDB 已经在多个站点部署，其主要场景如下：

1）网站实时数据处理。它非常适合实时的插入、更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。

2）缓存。由于性能很高，它适合作为信息基础设施的缓存层。在系统重启之后，由它搭建的持久化缓存层可以避免下层的数据源过载。

3）高伸缩性的场景。非常适合由数十或数百台服务器组成的数据库，它的路线图中已经包含对 MapReduce 引擎的内置支持。

不适用的场景如下：

1）要求高度事务性的系统。

2）传统的商业智能应用。

3）复杂的跨文档（表）级联查询。

2.5 AngularJS

AngularJS 是 MEAN 中的 A。简单来说，AngularJS 是一个 JavaScript 框架，用于直接在前端处理数据[7]。传统的做事方式是在服务器上拥有所有的数据处理和应用逻辑，然后将 HTML 传递给浏览器。AngularJS 使您可以将部分或全部此处理和逻辑移出浏览

器，有时使服务器只是从数据库传递数据。

如果您熟悉 jQuery，您可能会想知道 AngularJS 是否以同样的方式工作。简短的答案是否定的，不是真的。在 HTML 被发送到浏览器并且文档对象模型（DOM）已经完全加载之后，jQuery 通常被添加到页面以提供交互性。AngularJS 提前一步，并根据提供的数据帮助组合 HTML。

如前所述，AngularJS 可以根据所提供的数据将 HTML 放在一起，但它不仅仅是这样。如果数据发生变化，它也会立即更新 HTML，如果 HTML 更改，还可以更新数据。这被称为双向数据绑定。

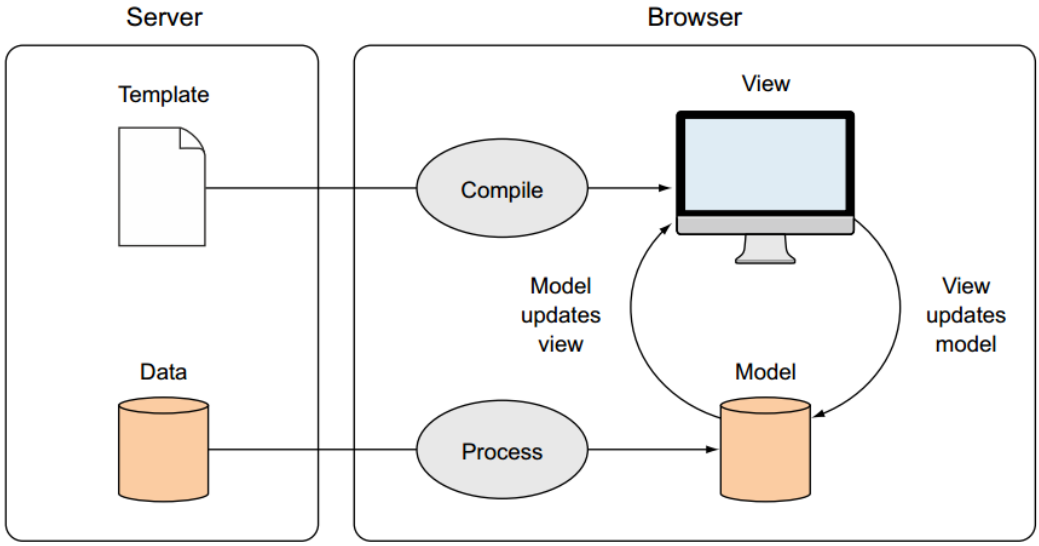

图 2.1 双向数据绑定——模型和视图在浏览器中处理并绑定在一起，其中一个更新时另一个也会立即更新

3 需求分析

3.1 可行性分析

3.1.1 社会可行性

在二手市场经济中，交易买卖是权益者和消费者之间的链接。

每年都有大量的毕业生，他们不可能将几年配置的东西都带走，面临权衡，面对搬运物品所耗费的高额成本，保留这些东西的效用如果低于售出获利的效用，作为一个理性的决策者，自然会选择低价出售，资源的重复利用方面可大大降低成本，价格低廉的物品对于消费者而言几乎没有什么替用差异。

当在购买物品过后的一段时间拥有了再次购买的能力，或者是想购买更新更高好产品的欲望，都愿意卖旧买新；可见，学生的消费习惯和消费层次也为二手市场提供了发展空间[8]。

但是跳蚤市场的时间短暂，并且受到场地和时间的限制，无法满足学生的需求。基于以上问题，本课题设计开发了校园二手交易系统，从根本上解决了传统的校园二手信息发布交流方式的弊端，使得校园各种二手物品信息得到了有效的整合，方便了广大学生。

3.1.2 技术可行性

从技术上来说，由于是基于 Web 的，可以相对比较容易面对用户的实际需求而开发。在现今，各种网络应用的开发已经相当成熟，出现了几大主流的开发语言和工具，都可以非常有效的支持开发这样一个系统。在今天有很多架构可以选择用以建立一个 Web 应用，MEAN 是一个全面的 JavaScript 解决方案，可以帮助我们快速构建可靠和可维护的生产 Web 应用程序。[9]

MEAN 是一个 JavaScript 平台的现代 Web 开发框架总称，它是 MongoDB + Express +AngularJS + Node.js 四个框架的第一个字母组合。它与传统 LAMP 一样是一种全套开发工具的简称。

MongoDB 是一个使用 JSON 风格存储的数据库，非常适合 JavaScript。

Express 是一个 Web 应用框架，提供有帮助的组件和模块帮助建立一个网站应用。

AngularJS 是一个前端 MVC 框架。

Node.js 是一个并发异步事件驱动的 JavaScript 服务器后端开发平台。

在 MongoDB 中我们可以直接存储 JSON 格式的数据，然后在 Express 和 Node.js 服务器编写一个基于 JSON 的查询，并无缝地(无需像其他语言需要在 JSON 和语言数据模型之间转换)传递 JSON 到 AngularJS 前端。

同时，数据库调试和管理也变得轻松了许多，存储在数据库中的对象基本上等同于我们在客户端看到的对象。更妙的是，前端工作人员也能够轻松了解后端代码和数据库查询，使用的是相同的语法和对象，不必考虑多套语言的最佳实践，降低了构建 Web 应用的入门门槛。

3.1.3 经济可行性

经济可行性主要是指网站投入与产出之间的关系。投入主要包括硬件设施和软件系统，开发费用，后期运营与维护等内容。网站的效益主要包括提高网站访问量，改善服务质量，增加网站订单或者其他方面的经济效益。经济可行性分析主要包括一下三个方面：

（1） 网站投入成本

在网站开发时期，投入成本主要有软件开发费用，硬件和软件设备费用，宣传成本，运营管理成本等，因此在初期可能投入相对较大。但是在很大程度上，这些成本属于一次性投资成本，对后续的资金流不会造成太大的影响；在人力资源投入上，因为该校园二手交易系统本身是为广大校内学生服务，因此我们可以采取招募志愿者和喜爱互联网，有志于创业的学生，参与网站信息的更新与管理，节省人力支出费用。

（2） 网站投资回报

本系统是专门针对广大学生而研发的，在学生中间容易引起关注，从而带来巨大的网站流量和商业潜力。网站可以引入企业参与建设，如广告位出租，广告信息发布等，产生投资回报。此外，随着网站访问量的增加，网站更容易形成口碑[10]。与外界的合作将会进一步加强，投资回报也相对客观。

（3） 社会效益

二手交易平台是专门针对校园二手物品的买卖，提倡绿色低碳的消费观，实现了资源的循环再利用。相对于原有的摆地摊、跳骚市场等，二手交易网站不受时间和空间的限制，更具方便性和经济性。

3.1.4 操作可行性

（1）二手物品种类繁多、方便学生生活

新型大学校园二手物品交易平台交易物品实用，方便学生生活。二手交易平台提供的多为方便学生学习与生活物品的信息，如二手工具书在大学生中非常抢手，上大学后面临各种各样的专业考试，这些书考前必须用，考试之后却不存在多大价值，通过二手物品交易平台可以实现有效的流通，节约学生的考试成本。自行车在大学校园也是抢手货，大学校园一般面积较大，学生对自行车的需求大。买新的，价格高，还容易丢，用了大学四年还不知怎样处理，如果通过二手物品交易平台，既能买到价钱合理的二手自行车，又能在毕业后确保能够再转手，方便了大学四年的学习生活。二手的电子产品也备受欢迎，大学生对于电子产品的更新速度快，新鲜感强，通过二手交易平台能够做到资源的有效配置。生活必需品也是二手交易平台的主要角色，暖水瓶、衣架、电子秤、台灯、毛绒玩具、健身器材……看似零碎的生活物品如果都能通过二手交易平台进行交易，则实现了资源的可重复利用，也实现了勤俭节约。[11]

（2）交易物品质量能够得以保障

新型大学校园二手物品交易平台交易物品安全，保障学生交易。由于在平台上进行交易的基本为本校的学生，他们在发布供求信息时都提供了真实的联系方式，物品价格及质量可以直接当面确定，而且作为交易的双方都为学生，不是单纯为了利益而进行的交易买卖，二手物品的质量可以得到保证。作为中介机构的二手交易平台在交易前会对交易物品进行审核，交易后对其承担一定的担保责任，保证交易的顺利进行。

（3）交易平台丰富学生交易渠道

新型大学校园二手物品交易平台实行网络交易与实体买卖同时进行，丰富学生交易渠道。对活动进行大面积推广工作，具体可通过海报、发放名片等形式。继而在此基础上，组织第一次大型二手卖场。推出固定的活动周期（即固定活动日），并同步开通网上

店铺，实时更新卖家信息。新型大学校园二手物品交易平台采用先进的现代企业管理制度，有完善的机构宗旨，积极地机构文化。制定相关管理条例细则，严格按其标准执行。网页管理过程中，实施专人设计管理，网页架构贴合大学生爱好趋向。利用良好的摄影技术，最大化的展现卖家风采。校园网的全面覆盖为新型大学校园二手物品交易平网络交易提供了便捷的条件。在校大学生 80%以上的同学使用校园网，与校园网的绑定实现了交易信息资源传输与共享、提高交易效率。

3.2 需求描述

3.2.1 用户类和特征

校园二手交易系统的用户以在校大学生为主，其用户群体具有以下特征：（1）网络使用频率高，习惯通过网络渠道获取各种信息；（2）区域集中，以校园及校园周边人群为主，见面交易安全方便；（3）收入来源单一，经济拮据，倾向于购买高质量二价格低廉的物品，也有通过售卖闲置物品换取资金、或以物换物的需求；（4）需求物品种类集中，如宿舍里的生活用品和书籍单车等，都是大多数学生生活中的必需品，也是容易被闲置的物品。

3.2.2 用例图

用例图是系统建模的起点，可以使用用例图对将要开发系统的实际工作流程进行业务姜末，从业务模型的基础上过渡到系统建模的开始，可以通过用例图来搜集用户的需求，明确和系统相关的用户和其他系统，同时确定系统将会提供什么功能，以及各个功能之间的关系。[12]

图 4.1 是校园二手交易系统的用例图。参与者只包含用户和系统本身，但用户可以细分为未注册用户、信息发布方及信息接收方。不同角色具有不同的权利，如未注册用户只能浏览物品和注册，信息接收方可发起交易请求，信息发布方可修改属于本人的物品信息。用例图的构建可以更好地描述用户需求，同时也有助于分析得出系统的概要功能和行为。

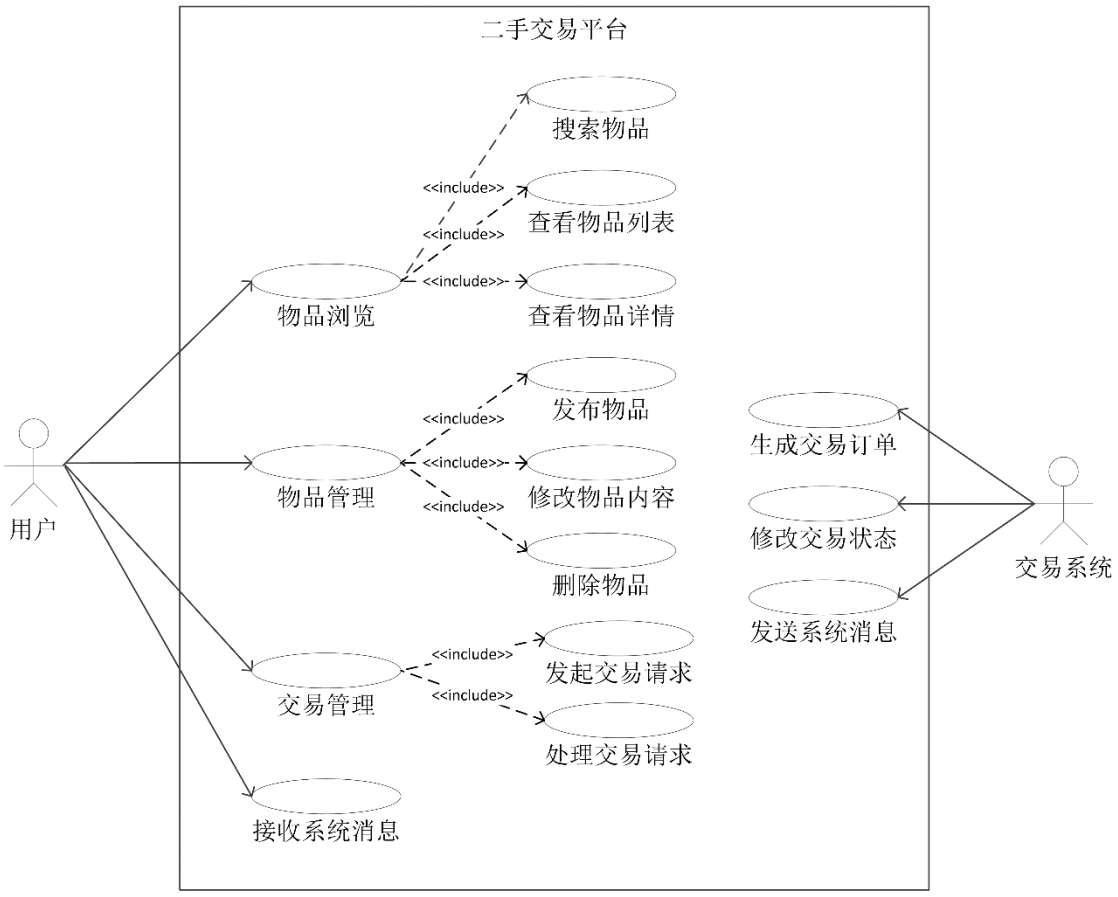

图 4.1　校园二手交易系统的用例图

用例图从整体上概括地描述了系统的功能，具体到每一个用例的详细工作过程，则需要单独的文档来描述，这些文档称为用例描述文档。下面根据校园二手交易系统的“交易管理”用例，给出该用例的详细用例方案如图 4.2 所示。

校园二手交易系统——交易管理用例

1. 简要说明

本用例描述用户选择物品后发起交易及另一方处理交易请求的过程。

2. 事件流

（1） 基本流

1.用户 A 选择需要的物品，发起交易请求

2.系统生成交易订单并通知用户 B

3.用户 B 收到系统消息，并“同意交易”

4.系统修改交易状态并通知用户 A

5.用户 A 收到系统消息，并确认“完成交易”

6.系统修改交易状态和产品状态，交易结束

（2） 备选流

3.a 如果用户 B“取消交易”，则系统修改交易状态，交易结束

3. 特殊需求

无

4. 前置条件

用户必须是系统注册用户，并执行用户登录用例。

5. 后置条件

无

6. 相关的数据

图 4.2 交易管理用例描述

3.3 功能需求

本系统功能主要包括以下几点：

3.3.1 用户注册

在本系统中，未注册用户只能浏览物品信息，不能在网站上发布出售物品或求购物品信息。因此，要想通过本系统进行 C2C 电子商务，要先注册成为本网站用户。在注册时需要填写这些信息：用户名、密码、手机号码。此外，注册成功之后可以在个人中心处修改宿舍地址、微信号和 QQ 号等个人信息。本系统主要的用户群体为本校学生，因此，用户信息重点突出学生特色。

3.3.2 信息发布功能

注册用户可以在网站上发布闲置物品信息或求购信息，在发布物品信息时，发布者需要填写物品名称，物品类别，物息类别（闲置/求购），详情描述等。

3.3.3 物品分类信息检索

物品分类信息检索主要分两大部分，一是物品类别检索，在发布信息时发布者需要注明物品类别，根据物品的自动分类进行物品搜索；另一个是关键字检索，搜索者输入物品信息关键字，与已发布的物品信息进行匹配，搜索到符合要求的物品。

3.3.4 出售/求购物品信息浏览

用户在搜索到适合的物品时，可进入物品详细信息页面查看物品信息，进一步判断物品是否符合自己的需求，并决定是否进行交易。

3.3.5 物品信息修改

信息发布者可以在个人中心修改已发布物品的详细信息；如果闲置物品已出售或者是已购买到求购物品，在交易结束后，发布者可以在个人中心修改交易状态，避免物品的再次交易。

3.3.6 用户个人物品

会员可以个人中心查看自己已经发布闲置物品和求购物品信息。

综上所述，系统功能结构图如图 4.3 所示。

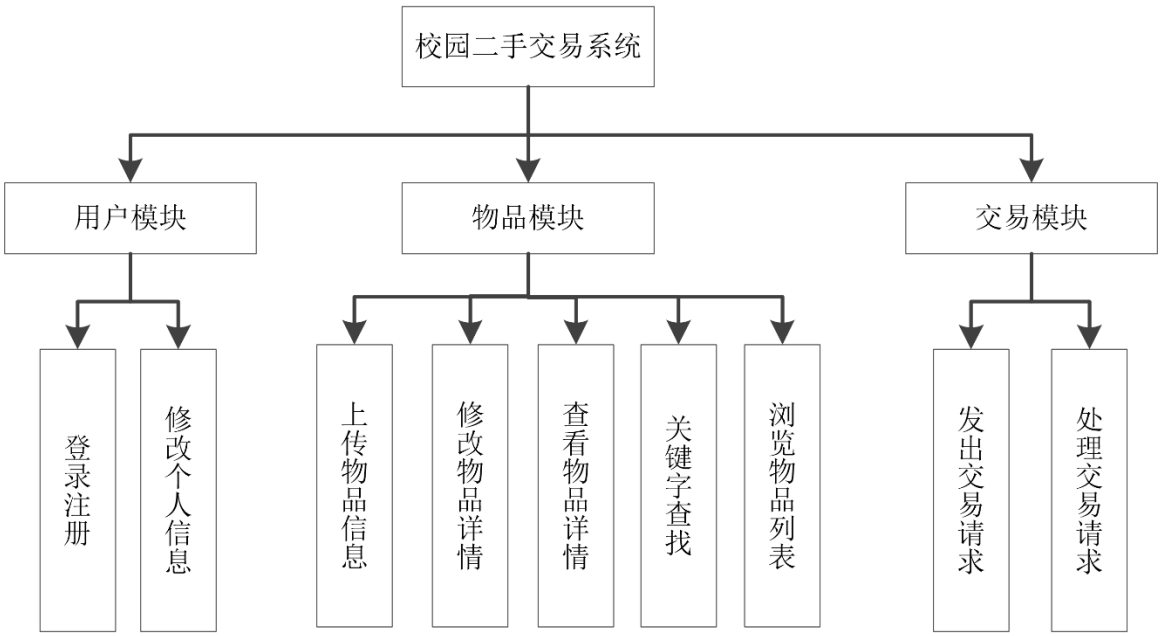

图 4.3　校园二手交易系统的功能模块图

4 详细设计

4.1 数据流图

数据流图是结构化分析的最基本的工具。它以图形的方式描绘数据在系统中流动和处理的过程[13]。图 5.1 是本课题系统的数据流图，它描述了系统的大部分功能，也可以通过数据的流动和处理过程得出数据库的大体设计。

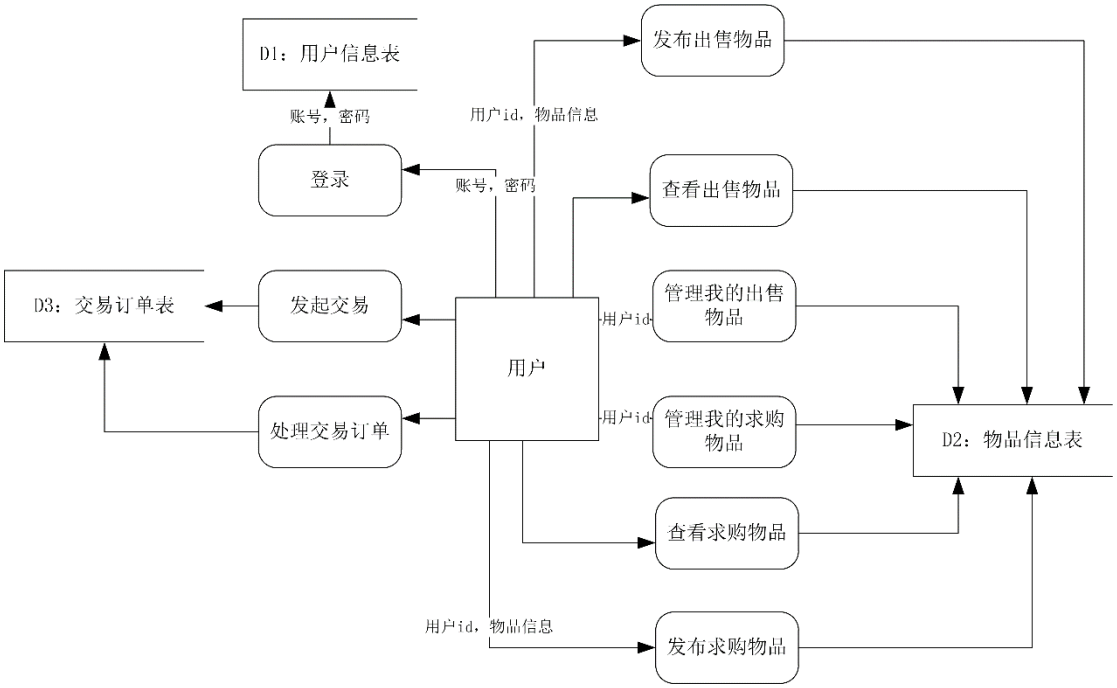

图 5.1 校园二手交易系统的数据流图

4.2 功能结构设计

按照页面结构将系统设计为一下几个部分。

4.2.1 物品浏览

物品浏览是所有用户（包括未注册用户）的基本功能 ，分为物品列表、物品详情和关键字搜索三种浏览方式。

（1）物品列表

物品列表又分为出售物品列表和求购物品列表两类，以同样的结构向用户展示物品的基本信息，包括图片、物品名称、交易类型、交易价格等信息。

（2）物品详情

物品详情页面便于用户进一步了解物品，除展示物品基本信息之外，还需要查看放大的物品图片，并根据用户身份选择显示可操作功能。对于非当前物品的创建用户显示“交易”按钮，对当前物品的创建用户显示“修改”和“删除”按钮。

（3）关键字搜索

该功能需要实现对类别和关键字进行搜索。

4.2.2 用户登录注册

游客可通过最快捷的方式进行注册，注册成功后直接跳转到登录后的首页。已注册用户登录成功后一样是跳转到首页。

4.2.3 发布物品信息

用于给注册用户发布出售物品和求购物品信息。发布信息时需要提交物品类型、名称、类别、价格、详情描述等具体信息，同时提供上传物品图片的功能。

4.2.4 发起交易请求

已登录用户可以发起交易请求，但是交易对象必须是他人的物品。发起交易请求后，系统应当自动生成交易订单，并通知物品的创建者。

4.2.5 个人中心

所有属于个人的管理操作都放在个人中心，包括物品管理、个人信息管理、交易订单管理，以及登出功能等。

（1）物品管理

物品管理分为“我的出售物品”和“我的求购物品”。以表格形式展示物品列表，并可对物品进行查看、修改、删除操作。

（2）个人信息管理

包括对个人信息的查看和修改操作。

（3）交易订单管理

使用列表展示所有的交易订单，可以对订单进行查看和修改状态的操作。修改状态

之后，系统自动通知对方。如果状态修改为“同意交易”且进行线下的实际交易操作之后，需要自行将物品的状态修改为“下架”。

通过查看交易订单详情可以获知交易双方的个人基本信息，然后再通过其他联系方式进行现场交易。

5 系统实现

5.1 数据库设计与实现

5.1.1 采用 Mongoose 技术

Mongoose 是一个稳健的 Node.js ODM 模块，可让 Express 应用支持 MongoDB。Mongoose 使用模型来模型化实体，Mongoose 的设计目标在于将 MongoDB 的无模式方法与实际应用开发的需求衔接起来。

5.1.2 数据库设计

本系统的 E-R 图设计如图 6.1 所示，描述了各实体间的联系及各实体的属性。

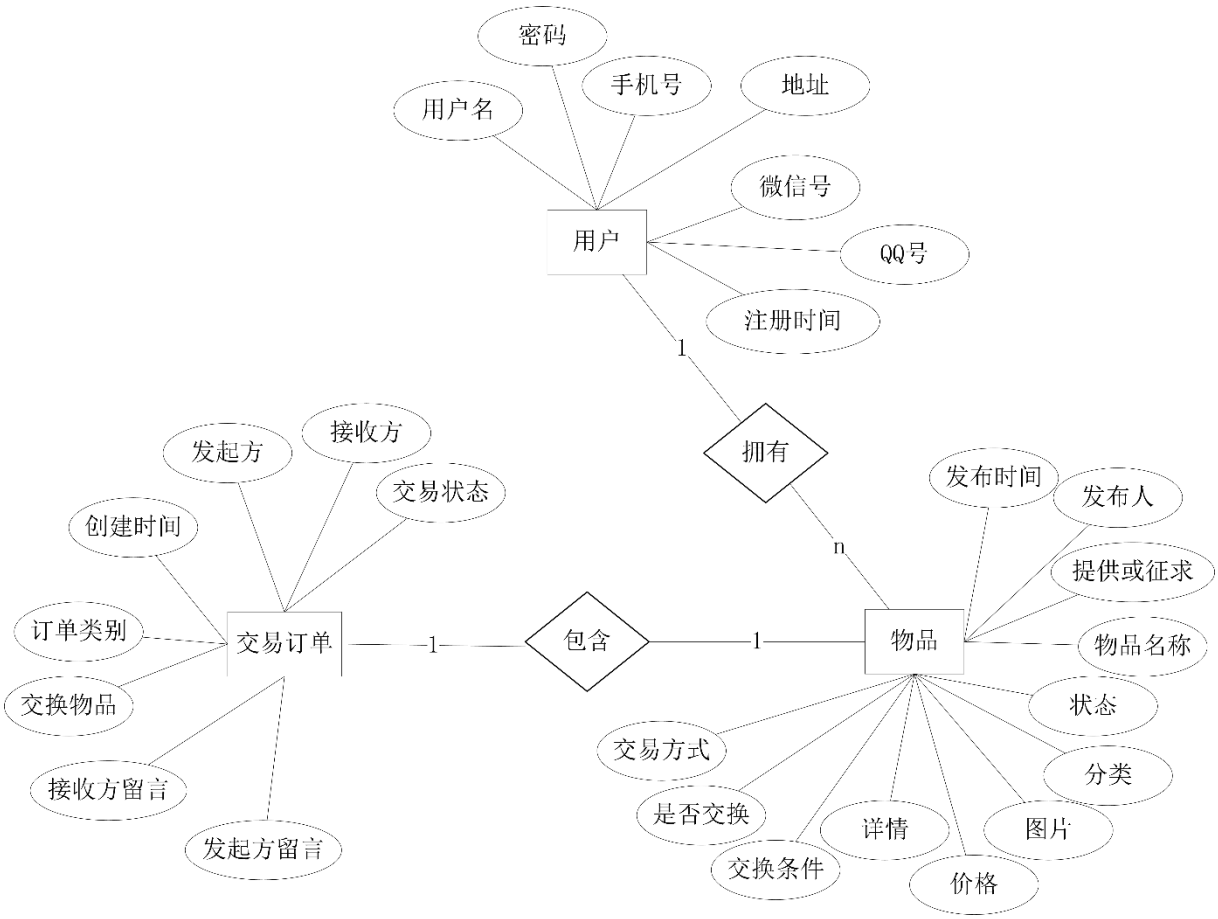

图 6.1 校园二手交易系统的 E-R 图

5.1.3 数据库实现

1. 安装 Mongoose

在项目根目录下打开命令行窗口，运行命令：

```
$ npm install mongoose –save
```

2. 连接 MongoDB

连接 MongoDB 实例，需要用到 MongoDB 连接字符串。MongoDB 连接字符串是一个 URL，用于为 MongoDB 驱动指定需要连接的数据库实例，其构造如下：

mongodb://username:password@hostname:port/database

但是在实际应用开发过程中，我们可以将应用程序变量存在环境配置文件中。

3. 以 TradeOrder 模型为例

MongoDB 使用集合存储多个文档时，并不要求所有文档的结构一致。但在实际应用中，我们需要相似的文档结构。Mongoose 可以通过模式对象定义文档的各个属性，各属性都有相应的类型和约束，以便于控制文档结构。

我们需要先创建一个模式。在 app/models 文件夹下创建 tradeOrder.server.model.js 文件，代码如下：

```
var mongoose = require('mongoose'),
    Schema = mongoose.Schema;

var TradeOrderSchema = new Schema({
    created: {
        type: Date,
        default: Date.now,
    },
    // 其它字段
});
mongoose.model('TradeOrder', TradeOrderSchema);
```

以上代码首先使用模式构造器定义了 TradeOrderSchema 对象，然后通过模式实例定义了 TradeOrder 模型。

此外，我们还需要注册一下 TradeOrder 模型，在 config 目录下新建文件 mongoose.js，

代码如下：

```
var config = require('./config'),
    mongoose = require('mongoose');
module.exports = function () {
    var db = mongoose.connect(config.db);
    require('../app/models/tradeOrder.server.model');
    return db;
}
```

5.2 用户模块

5.2.1 使用 Passport 模块管理用户权限

处理用户登录和注册的鉴权，对于大多数的 Web 应用来讲都是至关重要的一环。本系统通过 Passport 模块的本地策略实现对用户权限的管理。

这部分的核心代码如下：

```
var passport = require('passport'),
    mongoose = require('mongoose');
module.exports = function() {
    var User = mongoose.model('User');
    passport.serializeUser(function(user, done) {
        done(null, user.id);
    });
    passport.deserializeUser(function(id, done) {
        User.findOne({
            _id: id
        }, '-password -salt', function(err, user) {
            done(err, user);
```

```
        });
    });
    require('./strategies/local.js')();
};
```

上述代码中，passport.serializeUser()和 passport.deserializeUser()是用于定义 Passport 处理用户信息的方法。当用户身份验证完成之后，Passport 会将用户的_id 属性存到会话中。当需要使用 user 对象的时候，Passport 便使用_id 属性从数据库中获取用户信息。另外，代码中还包含了本地策略配置文件，这样，server.js 就可以完成 Passport 本地策略的加载。

5.2.2 对密码进行哈希加密

如果在实际应用中直接存储用户的密码明文，一旦泄露后果将会不堪设想。因此，在存储用户对象到数据库之前，预处理中间件将会做以下处理：首先，使用伪随机方法生成一个 salt；然后，使用实例方法 hashPassword()对原密码执行哈希操作。在 User 模型中的主要代码如下：

```
// 预存储处理中间件，用以执行对用户密码的哈希操作
UserSchema.pre('save', function(next) {
    if (this.password) {
        // 使用伪随机方法生成一个 salt
        this.salt = new Buffer(crypto.randomBytes(16).toString('base64'), 'base64');
        this.password = this.hashPassword(this.password); // 对原密码执行哈希操作
    }
    next();
});
// 实例方法，通过使用 Node.js 的 crypto 模块来执行用户密码的哈希操作
UserSchema.methods.hashPassword = function(password) {
    return crypto.pbkdf2Sync(password, this.salt, 10000, 64).toString('base64');
```

```
};
// 实例方法，将接收的参数字符串的哈希结果与数据库中存储的用户密码哈希值进行对比
UserSchema.methods.authenticate = function(password) {
    return this.password === this.hashPassword(password);
};
// 静态方法，用于为新用户确定一个唯一可用的用户名
UserSchema.statics.findUniqueUsername = function(username, suffix, callback) {
    var _this = this;
    var possibleUsername = username + (suffix || '');
    _this.findOne({
        username: possibleUsername
    }, function(err, user) {
        if (!err) {
            if (!user) {
                callback(possibleUsername);
            } else {
                return _this.findUniqueUsername(username, (suffix || 0) +
                    1, callback);
            }
        } else {
            callback(null);
        }
    });
};
```

此外，实例方法 authenticate 用于将接收的参数字符串的哈希结果与数据库中存储的用

户密码哈希值进行对比；静态方法 findUniqueUsername 用于为新用户确定一个唯一可用的用户名。

5.2.3 用户的登录注册

通过修改用户控制器来增加登录注册的功能，主要代码如下：

```
exports.signup = function(req, res, next) {
    if (!req.user) {
        var user = new User(req.body);
        user.provider = 'local';

        user.save(function(err) {
            if (err) {
                var message = getErrorMessage(err);
                req.flash('error', message);
                return res.redirect('/signup');
            }
            req.login(user, function(err) {
                if (err) return next(err);
                return res.redirect('/');
            });
        });
    } else {
        return res.redirect('/');
    }
};
exports.signout = function(req, res) {
    req.logout();
    res.redirect('/');
```

```
};
```

此外，还需要一个用于处理 Mongoose 错误对象并返回统一格式的错误消息的方法，代码如下：

```
// error handling function
var getErrorMessage = function(err) {
    var message = '';
    if (err.code) { // MongoDB 索引错误的错误代码
        switch (err.code) {
            case 11000:
            case 11001:
                message = '用户名已存在';
                break;
            default:
                message = 'Something went wrong';
        }
    } else if(err.errors) { // Mongoose 检验错误的 err.errors 对象
        for (var errName in err.errors) {
            if (err.errors[errName].message) message = err.errors[errName].
                message;
        }
    }else {
        message = 'Unknown server error';
    }
    return message;
};
```

5.2.4 客户端的身份验证

本系统采用的方式为：直接由 Express 应用在 EJS 视图中填充 user 对象，然后以

AngularJS 服务的方式来进行封装。

5.3 其他模块的前后端实现

5.3.1 创建 MEAN 的 CURD 模块

CURD 模块（增删查改模块）是 MEAN 应用的基本构件。一个 CURD 模块包含了两个 MVC 结构，功能分布在 AngularJS 和 Express 两部分。

1、配置 Express 组件

要建立 CURD 模块的 Express 部分，首先需要创建一个 Mongoose 模型，然后是创建用于处理模块业务逻辑的 Express 控制器，最后是为控制器方法创建 REST 风格 API 的路由。

下面以发起新的交易请求的 create()方法为例：

```
exports.create = function (req, res) {
    var tradeOrder = new TradeOrder(req.body);
    tradeOrder.creator = req.user;

    tradeOrder.save(function (err) {
        if(err) {
            return res.status(400).send({
                message: getErrorMessage(err)
            });
        }else {
            res.json(tradeOrder);
        }
    })
};
```

该方法先从 HTTP 请求对象中获取 JSON 对象，用这个 JSON 对象来创建相应的文档，再调用 Mongoose 模型的 save()方法保存到 MongoDB。

此外，本系统采用了 REST 风格 API 的体系结构设计。REST 风格的架构实现通常基于这几点：（1）每个资源一个基本 URL；（2）使用 JSON 作为数据结构格式；（3）使用标准的 HTTP 方法。基于以上几点，本系统设计 product 的 Express 路由的基本代码如下：

```
var users = require('../controllers/user.server.controller'),
    product = require('../controllers/product.server.controller');

module.exports = function (app) {
    app.route('/api/product')
        .post(users.requiresLogin, product.create)
        .get(product.query);

    app.route('/api/product/:productId')
        .get(product.read)
        .put(users.requiresLogin, product.update)
        .delete(users.requiresLogin, product.hasAuthentication, product.delete);

    app.param('productId', product.productByID);
}
```

上述代码创建了如下五个相关的 API：

- GET http://localhost:3000/product：用于返回物品列表
- POST http://localhost:3000/product：用于创建并返回新物品
- GET http://localhost:3000/product: productid：用于请求特定单个物品
- PUT http://localhost:3000/product: productid：用于更新并返回特定物品
- DELETE http://localhost:3000/product: productid：用于删除并返回特定物品

2、实现 AngularJS 的 MVC 模块

CURD 模块的第二大部分是 AngularJS MVC 模块。该模块包括一个使用$resource 工厂方法与 Express API 进行通信的 AngularJS 服务，一个包含客户端模块逻辑的 AngularJS 控制器，以及多个提供给用户进行增删改查操作的界面视图。

可以实现 AngularJS 应用与后端 API 的通信可以通过$http 服务来实现，但$http 服务提供的是 HTTP 请求较为低级的接口，因此本系统使用了 ngResource 模块，这是一个提供了更为简便的与 REST 风格数据通信的方法。使用 ngResource 模块的主要代码如下：

```
angular.module('tradeOrder').factory('TradeOrder', ['$resource', function ($resource) {
    return $resource('api/tradeOrder/:tradeOrderId', {
        tradeOrderId: '@_id'
    },{
        update: {method: 'PUT'}
    });
}]);
```

因为模块的逻辑全部都集中在 AngularJS 控制器里，所以这里的控制器提供了执行增删改查操作所需要的所有方法。此处不做详述。

5.3.2 文件上传功能的实现

普通的表单提交可以直接使用 GET 方法，但是当提交的内容包含文件时就需要使用 POST 方法来传数据到服务端。这里采用的方法是，提交表单时将文件存放到服务端的指定文件夹下，然后在数据库中存储响应的路径。下面以上传图片为例说明文件上传功能的实现。

HTML 部分需要特别设置 type 属性为 file，同时 accept 类型设为“image/*”：

```
<input type="file" accept="image/*" name="image" id="upImage">
```

同时，在使用$resource 服务时，需要另外扩展资源方法，其中 post 方法就是用于上传文件的，做好相应配置：

```
angular.module('product').factory('Product', ['$resource', function ($resource) {
    return $resource('api/product/:productId',
```

```
        {
            productId: '@_id'
        },
        {
            update: {method: 'PUT'},
            post: {
                method:'POST',
                headers: {'Content-Type':undefined},
                transformRequest: angular.identity
            }
        }
    );
}])
```

做好以上准备之后就可以在客户端使用 post 方法来上传图片了，在控制器中添加如下代码：

```
$scope.create = function () {
    var data = new FormData($('#create_product')[0]);
    if(!$('[name="image"]').val()){ //  如果没有上传图片
        data.delete('image');
    }

    Product.post(
        {},
        data,
        function (res) {
            $location.path('products/' + res.offer_or_get + '/' + res._id);
        },
```

```
            function (errRes) {
                console.log(errRes)
                console.log('err!!')
            }
        )
};
```

上述代码先将表单内容封装成 FormData 格式，然后再使用 post 方法将数据传到服务端。服务端接收到表单数据之后，需要通过额外的 formidable 模块来处理表单数据，主要代码如下：

```
exports.create = function (req, res) {
    var upImageDir = path.resolve(__dirname, '../../public/img/products/') + '/';
    fs.existsSync(upImageDir) || fs.mkdirSync(upImageDir);
    var form = new formidable.IncomingForm();
    form.encoding = 'utf-8';
    form.uploadDir = upImageDir;
    form.keepExtensions = true;
    form.maxFieldsSize = 3 * 1024 * 1024;
    var product = new Product();
    form.parse(req, function (err, fields, files) {
        if(files) {
            product.creator = req.user;
            //……其它数据段的赋值
        }
        product.image_url = 'img/bg.jpg';
        if(err) {// 这里进行错误处理}
        }
```

此外，数据库中存放的只是图片的路径，因此为避免图片命名重复，还需要对上传

图片的文件名进行处理。在上述函数内部的最后添加以下这段代码：

```
for (var key in files) {
    var extName = ''; //后缀名
    switch (key.type) {
        case 'image/pjpeg':
        case 'image/jpeg':
            extName = 'jpg';
            break;
        case 'image/png':
        case 'image/x-png':
        default:
            extName = 'png';
            break;
    }
    var productName = (new Date()).getTime() + '.' + extName;
    var newPath = form.uploadDir + productName;
    fs.renameSync(files[key].path, newPath); //  重命名
    product.image_url = "/img/products/" + productName;
```

通过以上处理，就可以实现图片上传的功能了。

6 系统测试

6.1 测试案例

6.1.1 物品浏览

系统将物品分为出售物品和求购物品两大类，下面均以出售物品为测试案例。注意此块对所有用户开放，包括未注册用户。

（1）物品列表

出售物品列表如图 7.1 所示。

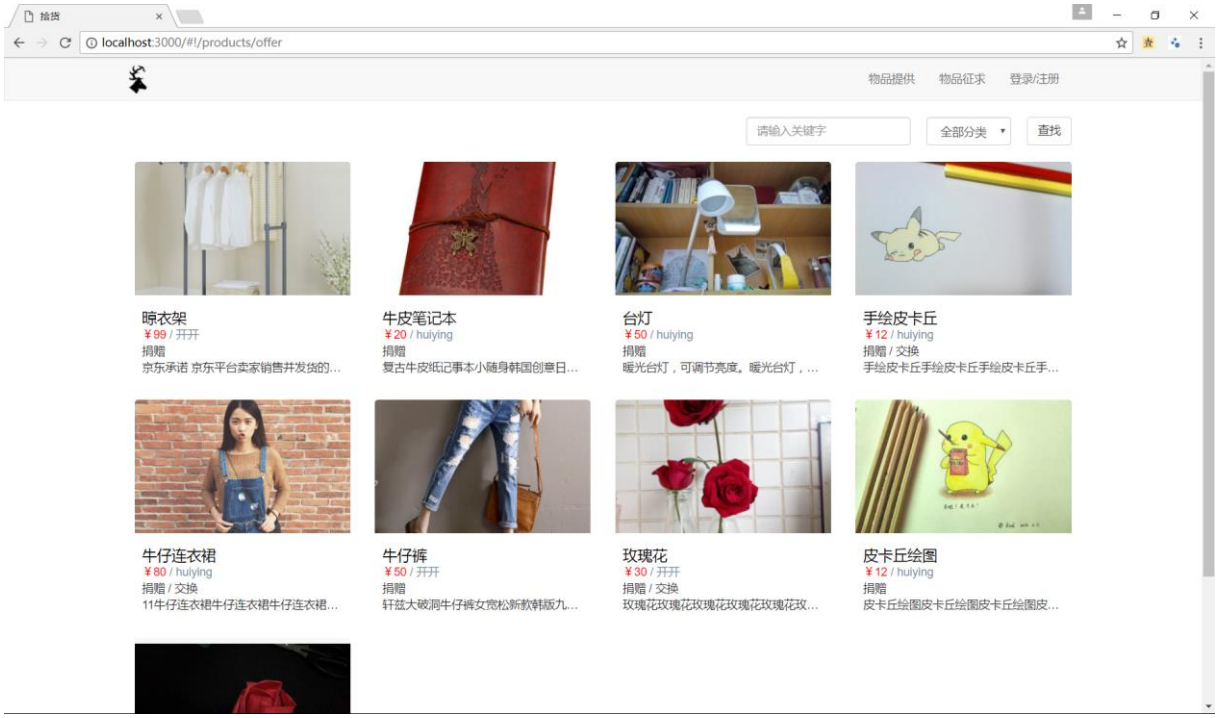

图 7. 1 物品提供列表界面

（2）关键字搜索

使用“牛仔”关键字和“衣物”类别的搜索结果如图 7.2 所示。

图 7.2 使用“牛仔”关键字和“衣物”类别的搜索结果

（3）物品详情

物品详情如图 7.3 所示，点击图片可放大。

图 7.3 物品详情页面

6.1.2 用户注册登录

（1）用户注册

输入：用户名，密码，手机号（如图 7.4 所示）

输出：注册成功后使用注册账号登录，并跳转到网站首页（如图 7.5 所示）

图 7.4 使用“晖莹”为用户名的用户注册界面

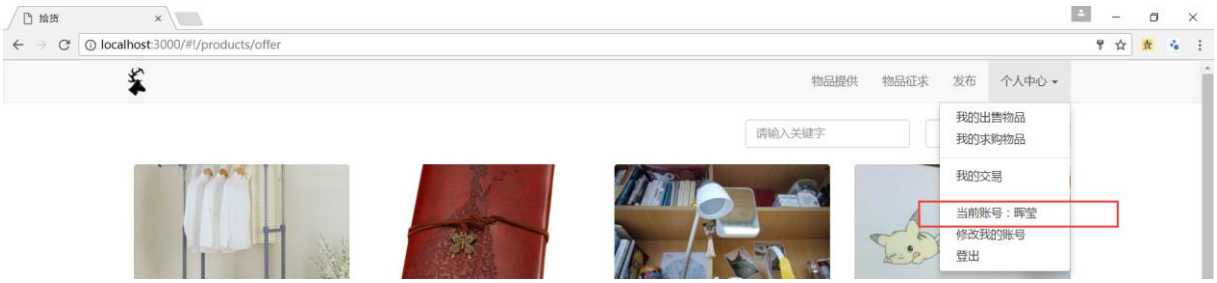

图 7.5 注册成功后自动跳转到登陆后的网站首页

（2）用户登录

输入：用户名，密码（如图 7.6 所示）

输出：登录成功后直接跳转到网站首页（效果与图 7.5 一致）

图 7.6 已注册用户的登录界面

6.1.3 发布物品信息

由于出售物品和求购物品的文档都属于 product 这个集合，因而发布界面使用同一个界面，用户需要根据物品的实际信息进行填写并提交，如图 7.7 所示。

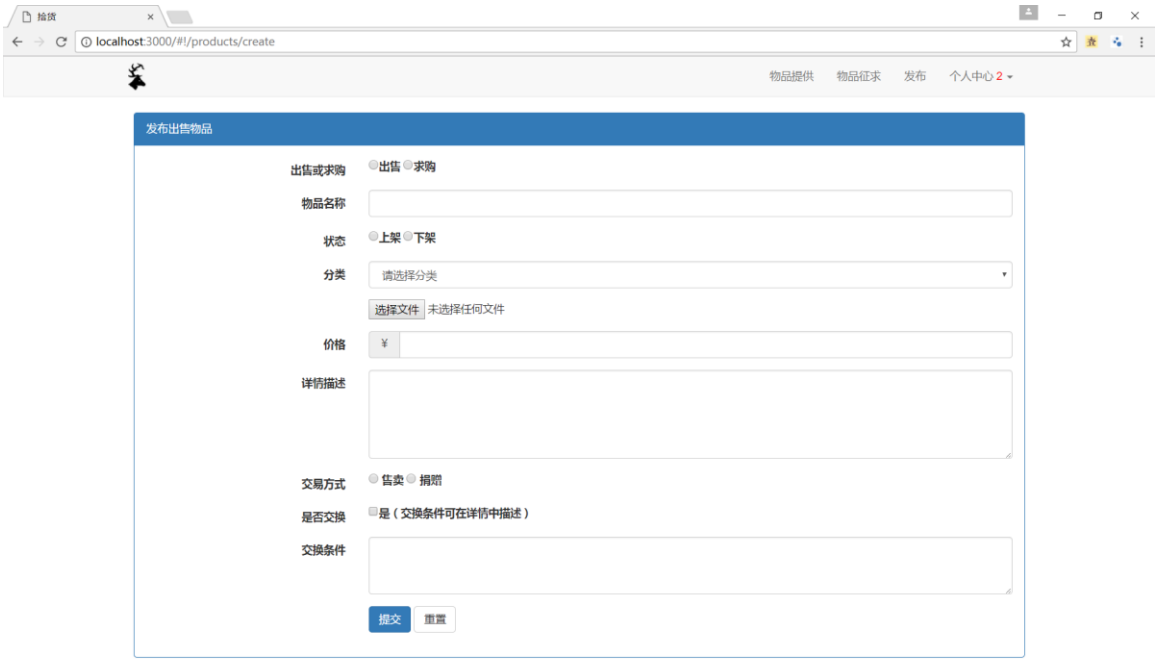

图 7.7 发布物品信息的界面

6.1.4 发起交易请求

在物品详情页面点击“交易”按钮即可发起交易请求，系统生成交易订单并通知交易请求的接收方。注意交易请求只可向他人发布的物品发起。图 7.8 展现了成功发起请求的界面。

图 7.8 点击交易后系统生成交易订单并返回提示

6.1.5 个人中心

个人中心主要包括了个人物品管理、个人资料管理和交易请求管理等功能，如图 7.9 所示。

图 7.9 个人中心的所有功能

（1）物品管理

物品管理包括了对物品的查看、修改、删除的处理，主要的管理页面如图 7.10 所示。

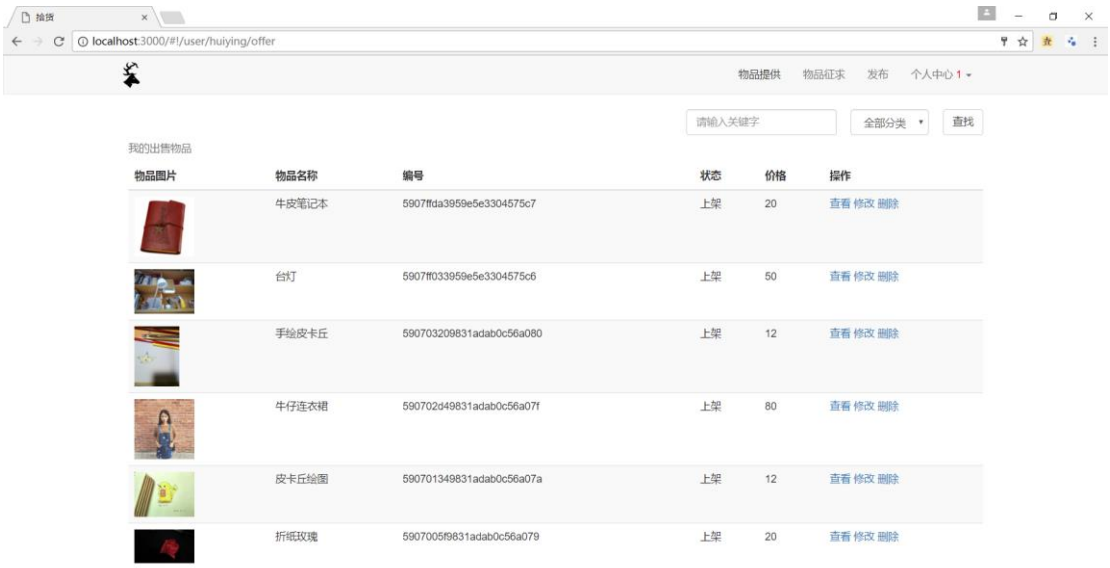

图 7.10 “我的出售物品”管理页面

（2）账号管理

账号管理包括了个人资料的查看和修改，修改页面如图 7.11 所示。

图 7.11 个人资料的修改页面

（3）交易订单管理

如果当前有待处理的交易请求，“个人中心”旁会高亮显示待处理请求的个数，打开“我的交易”页面可以看到所有关于本人的交易请求信息，包括“我发起的交易请求”和“我收到的交易请求”，如图 7.12 所示。

图 7.12 账号“huiying”的交易请求

通过查看交易请求的详情可以得知交易双方的信息，从而继续通过其他途径来交流和交易，如图 7.13 所示。

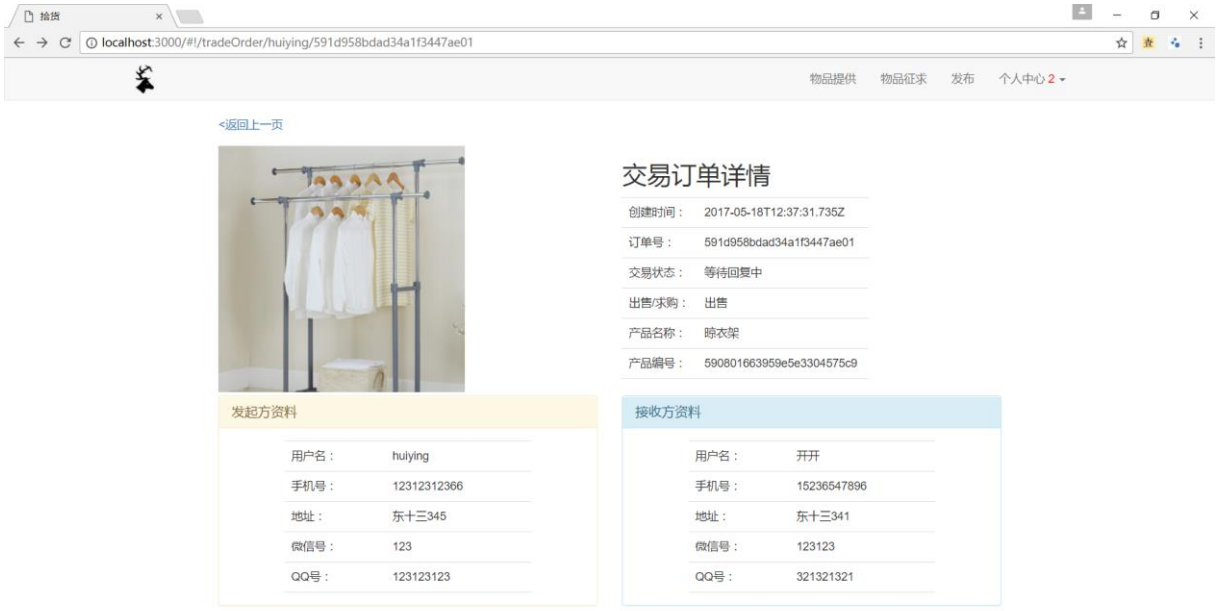

图 7.13 交易订单详情的界面

6.2 测试结果及分析

测试结果完成了交易平台的发布、搜索信息以及其它前文提及的功能需求，同时界面采用响应式开发，外观整洁大方，易于操作。

结　论

1、总结

本次开发的校园二手交易系统基本完成了提供发布物品信息和搜索物品信息的功能，为校园里的二手交易提供了参考价值。本次毕业设计按照严格的开发流程，对该系统进行了需求分析，概要设计，详细设计及技术调研，加强了我的开发能力和文档编写能力。

2、存在的问题和进一步开发的建议

虽然为完成该系统，我已阅读大量资料并进行了细致的研究分析，但是由于时间仓促、水平有限，本系统还有很多需要进一步改进的地方。比如，可以进一步实现留言和在线支付的功能。鉴于校园内的交易距离较近，方便线下的直接交易，以及可以借助微信 QQ 等其它方式进行交流，目前本系统只能采用通过其它方式进一步交流确定，然后货到付款的方式。

参 考 文 献

[1] 宋媛, 张大成, 孙理曹, 海兵. 基于微信二手交易平台的搭建与运作[J]. 合肥学院学报(社会科学版), 2015, 32(04): 105-126

[2] 范炎峰. 浅析校园二手市场[J]. 科技展望, 2015, (10): 251-252

[3] 张亮亮, 孟庆国, 李瑞英, 陈鲁丰, 杨洪勇. 基于校园网上二手交易平台的研究[J]. 电脑知识与技术, 2015, (16): 252-253

[4] Amos Q.Haviv. MEAN Web 开发[M]. 北京:人民邮电出版社, 2015. 3-4

[5] 朴灵. 深入浅出 Node.js[M]. 北京:人民邮电出版社, 2013. 8-10

[6] Praveen Garg, Ms Monika Sharma. "BIG DATA" Analysis Using Hadoop & MongoDB[J]. International Journal of Modern Computer Science, 2016, 4(03): 153-156

[7] Simon Holmes. Getting MEAN with MongoDB,Express,Angular and Node[M]. the United States of America:Manning Publications, 2015. 13-16

[8] 董英茹. 简谈 AngularJS 在下一代 Web 开发中的应用[J]. 软件工程师, 2015, (05): 30-31

[9] 张鸿轩. 大学二手交易的经济市场分析[J]. 企业导报, 2015, (02): 96-97

[10] 林豪杰，基于 MVC 三层架构的校园电子商务平台的设计与实现：[硕士学位论文]. 成都：电子科技大学，2011

[11] 何凯, 辛怡俐. 构建高校二手交易平台的可行性分析[J]. 中国商界(下半月), 2009, (06): 160

[12] 李磊, 王养廷, 杜启军. 面向对象技术及 UML 教程[M]. 北京:人民邮电出版社, 2010. 61-76

[13] 李代平. 软件工程[M]. 北京:清华大学出版社, 2011. 67-71

致　谢

本论文是在我的指导教师饶东宁的亲切关怀和悉心指导下完成的。感谢饶东宁老师对我论文期间的耐心指导，饶东宁老师的温和与平易近人感动着我，严谨的治学态度，认真的学习精神影响着我，使我在学习上有所步，编程习惯上有所改善，学习能力有所长进。没有老师的辛苦的指导，也没有我今天的顺利毕业。在此，谨向饶东宁老师表示诚挚的谢意!

在实际的设计中，也感谢我的舍友，同学，大家一起奋斗，一起努力，相互帮助。能将自己学到的知识，技能运用到毕业设计，实际工作中，我非常满足，能完成这次设计也是一段难忘的经历。

在这毕业之际，感谢学校四年的教育，提供的资源，让我学到一技之长。最后感谢家人的关怀和支持，在我遇到困难与挫折时能给我力量。感谢和我一起走过大学时光的舍友，同学，与你们一起的日子，非常开心。大学即将结束，我会继续努力，实现自己的梦想。