武汉交通职业学院

2018 ～ 2019 学年第 三 学期

**毕业设计（论文）**

**课题** 基于Node.js的网上图书商城的搭建

**姓名 赵思**

**系部 电子信息工程系 专业 计算机应用技术**

**班级**  16移动（2）班 **学号** 1606271741

**指导教师** 蔡晓庆

目录

[摘 要 I](#_Toc4109484)

[第一章 引 言 1](#_Toc4109485)

[1.1 课题背景 1](#_Toc4109486)

[1.2 课题分析 2](#_Toc4109487)

[1.3 研究目的 2](#_Toc4109488)

[第二章 网 站 需 求 3](#_Toc4109489)

[2.1 前台需求 3](#_Toc4109490)

[2.1.1 有关商品的需求 3](#_Toc4109494)

[2.1.2 有关用户的需求 3](#_Toc4109495)

[2.2 后台需求 4](#_Toc4109496)

[2.2.1 有关管理员账号的管理 4](#_Toc4109497)

[2.2.2 有关统计页需求 5](#_Toc4109498)

[2.2.3 有关账号管理的需求 5](#_Toc4109499)

[2.2.4 有关订单管理的需求 5](#_Toc4109500)

[2.2.5 有关书籍管理的需求 5](#_Toc4109501)

[第三章 网站数据库设计 6](#_Toc4109502)

[3.1 数据库背景 6](#_Toc4109505)

[3.2 数据库常识 6](#_Toc4109506)

[3.3 数据库开发技术 7](#_Toc4109507)

[3.4 数据库构建 7](#_Toc4109508)

[3.4.1 构建书籍表 7](#_Toc4109509)

[3.4.2 构建书籍类型表 10](#_Toc4109510)

[3.4.3 构建用户表 11](#_Toc4109511)

[3.4.4 构建管理员表 12](#_Toc4109512)

[3.4.5 构建订单表 14](#_Toc4109513)

[3.4.6 构建购物车表 16](#_Toc4109514)

[第四章 搭建项目前后端框架 17](#_Toc4109515)

[4.1 搭建项目前端框架 17](#_Toc4109516)

[4.1.1 技术栈 17](#_Toc4109517)

[4.1.2 搭建过程 18](#_Toc4109518)

[4.1.3 项目目录 19](#_Toc4109519)

[4.2 搭建项目后端框架 21](#_Toc4109520)

[4.2.1 技术栈 21](#_Toc4109521)

[4.2.2 搭建过程 21](#_Toc4109522)

[4.2.3 项目目录 27](#_Toc4109523)

[第五章 网上书店后台管理功能 28](#_Toc4109524)

[5.1 管理员功能 28](#_Toc4109525)

[5.1.1 管理员登录 28](#_Toc4109526)

[5.1.2 编辑管理员个人信息 29](#_Toc4109527)

[5.1.3 修改管理员个人密码 30](#_Toc4109528)

[5.1.4 注销登录 30](#_Toc4109529)

[5.1.5 管理员权限 31](#_Toc4109530)

[5.2 统计功能 33](#_Toc4109531)

[5.3 账号管理 33](#_Toc4109532)

[5.3.1 用户账号管理 34](#_Toc4109533)

[5.3.2 管理员账号管理 36](#_Toc4109534)

[5.4 订单管理 38](#_Toc4109535)

[5.4.1 查看未付款的订单列表 38](#_Toc4109536)

[5.4.2 查看未发货订单并进行发货 38](#_Toc4109537)

[5.4.3 查看已发货的订单 39](#_Toc4109538)

[5.4.4 查看已收货的订单 39](#_Toc4109539)

[5.5 书籍管理 39](#_Toc4109540)

[5.5.1 分页显示书籍列表 40](#_Toc4109541)

[5.5.2 按书籍类别显示 40](#_Toc4109542)

[5.5.3 搜索书籍 41](#_Toc4109543)

[5.5.4 添加书籍 41](#_Toc4109544)

[5.5.5 书籍编辑 42](#_Toc4109545)

[5.5.6 下架书籍 44](#_Toc4109546)

[第六章 网上书店前台功能 44](#_Toc4109547)

[6.1 未登录时商城基本功能 44](#_Toc4109548)

[6.1.1 分类浏览 45](#_Toc4109549)

[6.1.2 书籍名称关键字搜索 46](#_Toc4109550)

[6.1.3 书籍详情的查看 46](#_Toc4109551)

[6.2 用户登录注册 47](#_Toc4109552)

[6.2.1 用户登录 47](#_Toc4109553)

[6.2.2 账号注册 48](#_Toc4109554)

[6.3 用户登陆商城后的功能 49](#_Toc4109555)

[6.3.1 书籍的购买和添加购物车 49](#_Toc4109556)

[6.3.2 查看购物车列表 51](#_Toc4109557)

[6.3.3 查看订单列表 51](#_Toc4109558)

[6.3.4 用户个人信息管理 52](#_Toc4109559)

[第七章 结论总结 55](#_Toc4109560)

[7.1 步骤归纳及课题分析问题结论 55](#_Toc4109561)

[7.2 研究目的有关新老技术问题结论 56](#_Toc4109562)

[7.2.1 新技术的优点 56](#_Toc4109563)

[7.2.2 新技术的缺点 56](#_Toc4109564)

[第八章 致谢 58](#_Toc4109565)

[第九章 参考资料 59](#_Toc4109566)

摘 要

现代化的高速网络催生出电子商务这个巨大的平台，各类商品的交易可以在电子商务网站上快捷的实现，本课题依据全后端分离的思想，将全栈性的搭建一个服务器基于Node.js，数据库基于MySQL，前端基于Vue 2.0框架，主要售卖图书的“网上图书商城”这一电子商务平台。

本图书商城网站的构建分为三大部分。

第一是数据库的设计，第二是Node.js服务端代码的编写。第三是前端页面的搭建。

数据库设计方面通过具体的调研参考深入思考，设计了图书表，图书类型表，客户表，订单表等等关系表。

服务端方面通过pm2来进行进程管理，Async 来优化异步的SQL操作，充分运用Node 单线程，事件驱动和非阻塞IO的特点，快速的响应。

前端以Vue 数据驱动和单页面富应用的思想，并通过WebPack项目打包工具，构建工程化的前端项目，以此提供友好的用户体验性和浏览器兼容性。

**关键词** 网上图书商城 全栈 数据驱动 单页面富应用 Node.js MySQL Vue

目录

1. 引 言
   1. 课题背景

网站是互联网的基础单元，是互联网上承载信息的载体，网民于网站上进行信息的浏览并相应的对信息进行传播，因此，网站在互联网上的组成地位极其重要。

正是因为网站的重要性，开发网站的技术在不断的进行提升，这种提升，不管是在前端方面还是在后端方面都有充分的体现。

后端方面。

在之前，一直采用C++、Java、PHP等编程语言来开发后端服务器，作为服务器的语言开发，它们的确是各有优势，能提供其他语言提供不了的服务器性能，处理其他语言处理不了的高并发量，但是，也正是因为他们性能强大，而导致十分臃肿，创建服务还需要具体的服务器来搭载，虽然在大中型服务器上，他们是不二之选，可是在小型或者自用服务器上，却并不是一个很好的选择。

而现在，新出现了Go语言和Node.js这些新型的服务器开发语言，其中Node.js独树一帜，它是一个JavaScript语言的运行环境，它以Chrome V8引擎进行代码解释，他有三个独特而强大的特性：事件驱动、非阻塞I/O、单进程。用Node.js编写并开启一个服务器也只需要不到15行代码，它的三个特点足以说明Node,js的方便快捷，短小精悍，其现已成为中小型服务器的首选开发语言。

前端方面。

在之前，前端只不过写一写HTML、CSS和JavaScript 代码，来构建页面，提供一些交互，做一些适应，并且在与后端的开发模式上，采用前后端混合开发的模式，前端代码与后端代码混合，前端程序写好页面后，由后端程序员来混入后端代码中，进行模板字符串的拼接，从而拼凑出前端页面，这种前后端混合的开发方式，给开发和维护带来极大的不便，列如，就算页面代码简单的修改，也需要在后端服务器上进行繁杂修改。而现在，前端也不再只是做页面，前端的发展是飞速的，涌现了各种各样的概念：前后端分离，渐进式页面，单页面富应用。如今的前端变得越来越工程化，通过WebPack等打包工具，让前端项目的构建和最终打包变的方面，Vue等前端框架的出现，对单页面富应用给出了很好的解释，其让网站不再是一张张HTML的页面，在一个HTML页面上构建一整个网站，已是一个轻松简单的事情。

前端和后端的种种变化，透露着网站技术的新颖性和复杂性。

* 1. 课题分析

本文课题为《基于Node.js的网上图书商城的构建》。

在本文中，我们将研究讨论如何基于Node.js来全栈性的构建一个网上图书商城的这一电子商务网站。因为是全栈性的构建，不单单仅限于Node.js的后端服务器，还涉及到前端界面和数据库的设计两大类，所以，本文将研究讨论以下三个问题。

1. 如何使用Node.js及相关框架来构建结构完备的后端项目，用以提供API接口的给前端调用，正确的接受前端请求提取数据，通过连接及操作数据库，来返回正确的数据于前端
2. 如何使用WebPack项目工具来配置前端项目，搭建Vue前端框架的开发环境，使用状态管理和前端路由，来构建单页面富应用的前端网站
3. 如何使用MySQL数据库，来创建适合图书商城的数据库，并完备的设计各类表结构，做到高效的读写数据。

本文将具体讨论以上三个问题，并对每个问题具体执行，最终实际的构建一个“基于Node.js的网上图书商城”。

* 1. 研究目的

如上课题背景的分析，当下前端和后端技术的发展和变化十分快速和复杂，新概念不断的被提出，从而不断发展成为新的技术。

本文“基于Node.js的网上图书商城”项目，在前端和后端上，对这些新技术进行实践。

在前端上，运用到了WebPack打包工具，来使前端项目工程化，基于 Vue 2.0框架构建出单页面富应用的网站。

在后端上，基于Node.js和Express框架来快速方便的搭建小型服务器。

本文将对项目中的新技术点一一分析研究，并放在项目中进行综合性的实践，有以下目的。

1. 构建图书商城网站，学习如何进行全栈性的项目开发
2. 学习前后端新技术，并在实际项目中具体应用
3. 观察并对比老技术，发现新技术的优点
4. 找出新技术的一些缺点，阐述其不足及应该如何改进

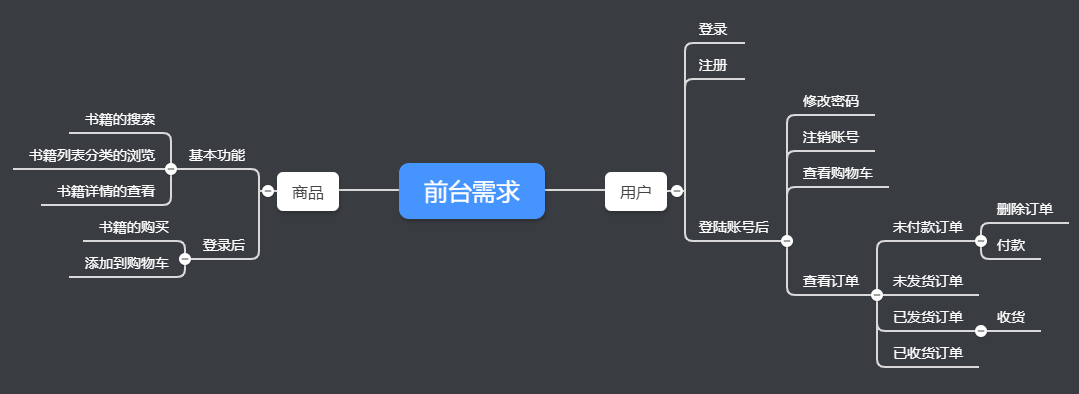
本文，将以“课题背景”为驱动，“课题问题”为主线，“研究目的”为结论，来具体分析研究此课题。

1. 网 站 需 求

本网站主要以线上售卖图书为核心的电子商务网站，网站总共包含两大部分。

* 1. 前台需求

前台页面主要提供给用户浏览，做为客户端使用，其中包含两大需求点。

* 商品
* 用户

前台需求（图2-1）

2. 1. 1. 有关商品的需求

商品需求点上，用户在没有登录时，具有浏览书籍，搜索书籍，查看书籍的详情，用户在登陆后，就可以购买书籍，添加书籍到购物车中等功能

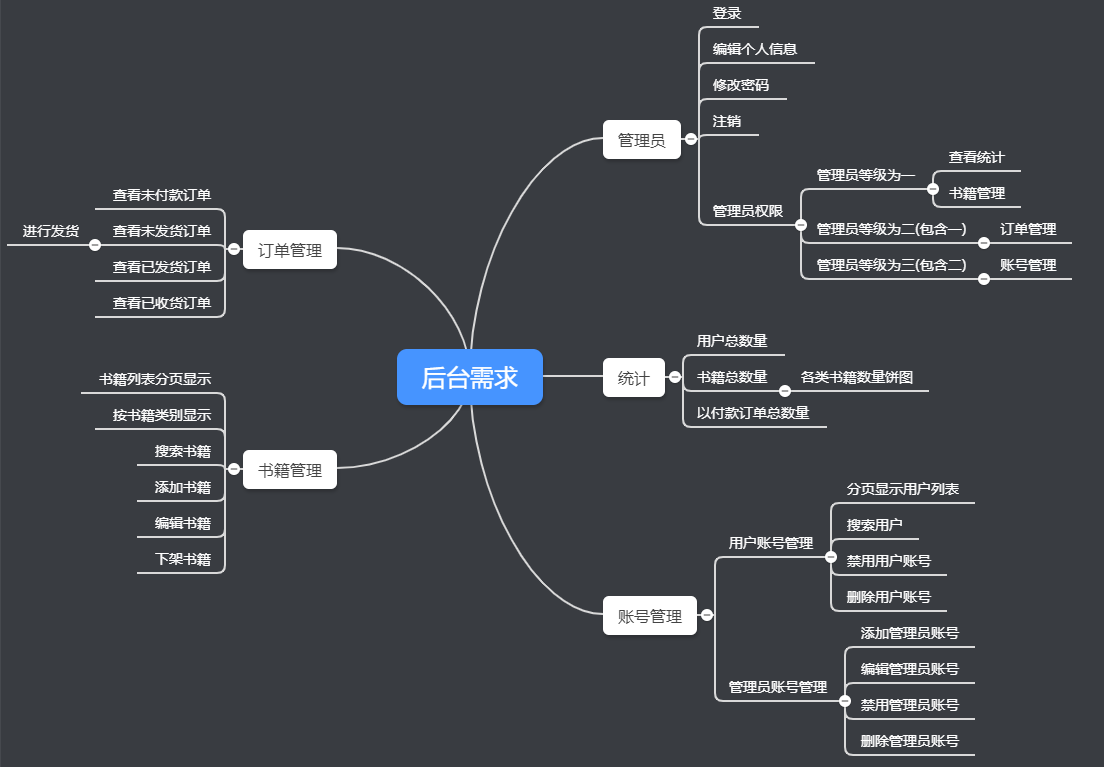
* + 1. 有关用户的需求

用户需求点上，用户可以登录网站，没有账号可以进行注册。在登录账号后，可以修改密码、注销账号、查看自己的购物车、查看自己的订单。订单页可以查看未付款的订单、未发货的订单、已发货的订单、已收货的订单。在未付款的订单中可以删除订单，或者对订单进行付款。在已发货的订单中，可以对订单进行收货。

* 1. 后台需求

网站的后台，主要提供管理员使用，通过后台可视化的管理网站，其中五大需求点。

* 管理员
* 统计
* 账号管理
* 订单管理
* 书籍管理



后台需求（图2-2）

* + 1. 有关管理员账号的管理

网站管理员在登录页面登录，管理员是具有等级的，不同等级具有不同的权限，具体权限可以参考下表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **权限**  **等级** | 查看统计 | 账号管理 | 订单管理 | 书籍管理 |
| 第一等级 | ✓ |  |  | ✓ |
| 第二等级 | ✓ |  | ✓ | ✓ |
| 第三等级 | ✓ | ✓ | ✓ | ✓ |

管理员权限表（表2-2-1）

管理员在登录后可以编辑个人信息，修改密码，注销等操作

* + 1. 有关统计页需求

在统计页中，可以看到当前注册用户的总数量，已付款订单的总数量，在统计书籍数量时，因为书籍具有不同的分类，所以通过饼图的形式，显示不同类别下书籍的数量

* + 1. 有关账号管理的需求

账号管理分为管理员账号的管理和用户账号的管理。

管理员账号的管理下有添加不同等级的管理员账号、编辑管理员账号、禁用管理员账号、删除管理员账号。

用户账号的管理下有分页显示用户的列表、搜索用户、禁用用户账号、删除用户的账号

* + 1. 有关订单管理的需求

订单管理下，可以分别查看未付款的订单、未发货的订单、并可以在未发货的订单下，进行发货。还可以查看已发货的订单，已收货的订单列表。

* + 1. 有关书籍管理的需求

在书籍管理页，可以分页的显示书籍列表，还可以通过选择书籍类别来显示书籍，还可以搜索书籍，添加书籍，编辑书籍的信息，并且下架书籍。

1. 网站数据库设计
3. 1. 数据库背景

网上图书商城是具有电子商务性质的网站，他对数据的要求较为复杂，设计到用户的数据存储和书籍书籍存储两大类。

在书籍数据中，书籍具有类别之分，不同书籍的类别应该分开存储，并且，书籍的数量庞大，需要有效的数据存储方式

在用户数据中，要处理用户的订单和购物车及用户信息等等数据，这些数据，都要求需快速的数据进行存储和读取，等数据的响应速度要求较高。

以上对数据的存储方式和响应速度，表示需要对数据的存储和响应需要有一个科学的方式来解决此问题。

因为数据库是当下解决此类问题的首选方案，因此，我们选择数据库来进行网站数据的存储。

* 1. 数据库常识

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。

每个数据库都有一个或多个不同的 API 用于创建，访问，管理，搜索和复制所保存的数据。

我们也可以将数据存储在文件中，但是在文件中读写数据速度相对较慢。

所以，现在我们使用关系型数据库管理系统（RDBMS）来存储和管理的大数据量。所谓的关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

RDBMS 即关系数据库管理系统(Relational Database Management System)的特点：

1. 数据以表格的形式出现
2. 每行为各种记录名称
3. 每列为记录名称所对应的数据域
4. 许多的行和列组成一张表单
5. 若干的表单组成database
   1. 数据库开发技术

|  |  |
| --- | --- |
| 所用数据库 | MySQL |
| 数据库版本 | 8.0.14 |
| 数据库语言 | SQL |

数据库开发技术（表3-3）

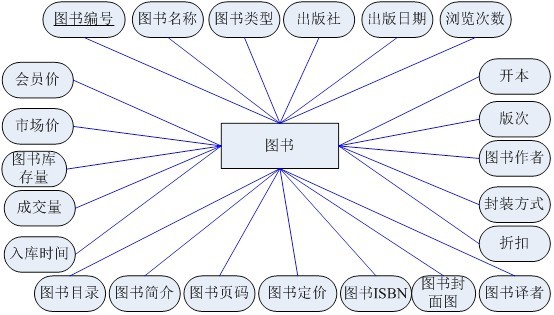
MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

* 1. 数据库构建

根据网站需求，将在MySQL 数据库中构建名称为“milinbook”的数据库，数据库采用“utf8mb4”编码，数据库将设计书籍表、书籍类型表、用户表、管理员表、订单表、购物车表。

* + 1. 构建书籍表
       1. E R 图

书籍表ER图（图3-4-1）

* + - 1. 关系表

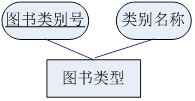
|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| BookId | 图书编号 | int | 主码（自增字段） |
| BookTypeId | 图书类型 | int | not null, 引用tb\_booktypeinfo的外码 |
| BookName | 课程名 | varchar(20) | not null |
| BookPress | 出版社 | varchar(20) | not null |
| BookPubDate | 出版日期 | datetime | not null |
| BookSize | 开本 | varchar(10) | not null |
| BookVersion | 版次 | varchar(10) | not null |
| BookAuthor | 图书作者 | varchar(10) | not null |
| BookTanslor | 图书译者 | varchar(10) |  |
| Bookisbn | 图书ISBN | varchar(20) | not null |
| BookPrice | 图书定价 | money | not null |
| BookPages | 图书页码 | int | not null |
| BookOutline | 图书简介 | varchar(200) | not null |
| BookCatalog | 图书目录 | varchar(200) | not null |
| BookMprice | 市场价 | money | not null |
| BookPrprice | 会员价 | money | not null |
| BookDealmount | 成交量 | int | not null |
| BookLookmount | 浏览次数 | int | not null |
| BookDiscount | 折扣 | decimal | not null |
| BookPic | 图书封面图 | varchar(20) | not null |
| BookStoremount | 图书库存量 | int | not null |
| BookStoretime | 入库时间 | datetime | not null |
| BookPackstyle | 封装方式 | varchar(20) | not null |

书籍关系表（表3-4-1）

* + - 1. 示列代码

1. **DROP** **TABLE** IF EXISTS `tb\_bookinfo`;
2. **CREATE** **TABLE** `tb\_bookinfo`  (
3. `BookId` **int**(11) NOT NULL AUTO\_INCREMENT COMMENT '图书编号',
4. `BookTypeId` **int**(11) NOT NULL COMMENT '图书类型',
5. `BookName` **varchar**(100) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '图书名',
6. `BookPress` **varchar**(255) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NULL **DEFAULT** NULL COMMENT '出版社',
7. `BookPubDate` datetime(0) NOT NULL COMMENT '出版日期',
8. `BookSize` **int**(11) NULL **DEFAULT** NULL COMMENT '开本',
9. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*中间省略\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
10. **PRIMARY** **KEY** (`BookId`) USING BTREE,
11. **UNIQUE** **INDEX** `BookISBN\_F`(`Bookisbn`) USING BTREE COMMENT 'ISBN 书籍编号唯一',
12. **INDEX** `BookTypeId\_F`(`BookTypeId`) USING BTREE,
13. **CONSTRAINT** `BookTypeId\_F` **FOREIGN** **KEY** (`BookTypeId`) **REFERENCES** `tb\_booktypeinfo` (`BookTypeId`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**
14. ) ENGINE = InnoDB AUTO\_INCREMENT = 428 **CHARACTER** **SET** = utf8 **COLLATE** = utf8\_general\_ci ROW\_FORMAT = **Dynamic**;

书籍表 （示列代码3-4-1）

* + 1. 构建书籍类型表
       1. E R 图

书籍类型ER图（图3-4-2）

* + - 1. 关系表

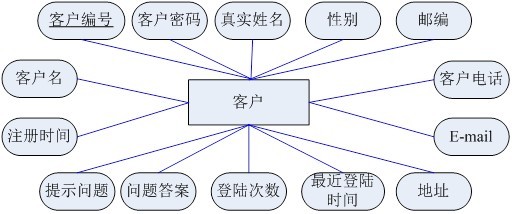
|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| BookTypeId | 图书类型编号 | int | 主码（自增字段） |
| BookTypeName | 类别名称 | varchar(50) | not null |

书籍类型关系表（表3-4-2）

* + - 1. 示列代码

1. **DROP** **TABLE** IF EXISTS `tb\_booktypeinfo`;
2. **CREATE** **TABLE** `tb\_booktypeinfo`  (
3. `BookTypeId` **int**(11) NOT NULL AUTO\_INCREMENT COMMENT '图书类型编号 自增字段',
4. `BookTypeName` **varchar**(50) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '  \r\n类别名称',
5. **PRIMARY** **KEY** (`BookTypeId`) USING BTREE
6. ) ENGINE = InnoDB AUTO\_INCREMENT = 29 **CHARACTER** **SET** = utf8 **COLLATE** = utf8\_general\_ci ROW\_FORMAT = **Dynamic**;

书籍类型表（示列代码3-4-2）

* + 1. 构建用户表
       1. E R 图

用户表ER图（图3-4-3）

* + - 1. 关系表

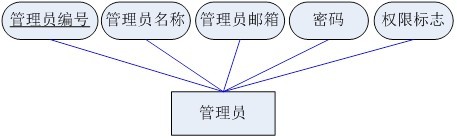
|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| CustomerId | 客户编号 | int | 主码（自增字段） |
| CustomerName | 客户名称 | varchar(20) | 主码not null |
| CustomerPwd | 客户密码 | varchar(20) | not null |
| Customertruename | 真实姓名 | varchar(20) | not null |
| CustomerSex | 性别 | varchar(2) | not null, 取“男”或“女” |
| CustomerTel | 客户电话 | varchar(20) | not null |
| CustomerEmail | E-mail | varchar(20) | not null |
| CustomerAddr | 地址 | varchar(20) | not null |
| CustomerRegTime | 注册时间 | datetime | not null |
| CustomerQues | 提示问题 | varchar(200) |  |
| CustomerAnswer | 问题答案 | varchar(200) |  |
| CustomerLogTime | 登陆次数 | int | not null |
| CustomerLastLogT | 最近登陆时间 | datetime | not null |

用户关系表（表3-4-3）

* + - 1. 示列代码

1. **DROP** **TABLE** IF EXISTS `tb\_customerinfo`;
2. **CREATE** **TABLE** `tb\_customerinfo`  (
3. `CustomerId` **int**(11) NOT NULL AUTO\_INCREMENT COMMENT '客户编号',
4. `CustomerName` **varchar**(100) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '客户昵称',
5. `CustomerPwd` **varchar**(100) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '客户密码',
6. `CustomerTrueName` **varchar**(100) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '真实姓名',
7. **PRIMARY** **KEY** (`CustomerId`) USING BTREE,
8. **UNIQUE** **INDEX** `customerinfoName\_F`(`CustomerName`) USING BTREE COMMENT '客户的昵称唯一性',
9. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*中间省略\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
10. **UNIQUE** **INDEX** `customerinfoEmail\_F`(`CustomerEmail`) USING BTREE COMMENT '邮件即为账号唯一性',
11. **UNIQUE** **INDEX** `customerinfoTel\_F`(`CustomerTel`) USING BTREE COMMENT '客户手机号码唯一性',
12. **INDEX** `CustomerId`(`CustomerId`) USING BTREE
13. ) ENGINE = InnoDB AUTO\_INCREMENT = 12 **CHARACTER** **SET** = utf8 **COLLATE** = utf8\_general\_ci ROW\_FORMAT = **Dynamic**;

用户表（示列代码3-4-3）

* + 1. 构建管理员表
       1. E R 图

管理员ER图（图3-4-4）

* + - 1. 关系表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| AdminId | 管理员编号 | int | 主码（自增字段） |
| AdminName | 管理员名称 | varchar(10) | not null |
| AdminPwd | 密码 | varchar(20) | not null |
| AdminFlag | 权限标志 | int | not null |

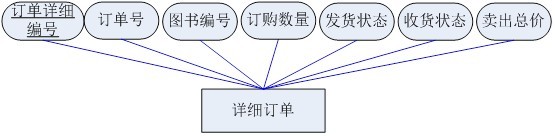
管理员关系表（表3-4-4）

* + - 1. 示列代码

1. **DROP** **TABLE** IF EXISTS `tb\_manager`;
2. **CREATE** **TABLE** `tb\_manager`  (
3. `AdminId` **int**(11) NOT NULL AUTO\_INCREMENT COMMENT '管理员编号',
4. `AdminName` **varchar**(255) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '管理员昵称',
5. `AdminAccount` **varchar**(255) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '管理员账号 邮箱',
6. `AdminPwd` **varchar**(255) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '密码',
7. **UNIQUE** **INDEX** `UN\_Name`(`AdminName`) USING BTREE COMMENT '姓名唯一',
8. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*中间省略\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
9. **UNIQUE** **INDEX** `UN\_Account`(`AdminAccount`) USING BTREE COMMENT '账号唯一'
10. ) ENGINE = InnoDB AUTO\_INCREMENT = 24 **CHARACTER** **SET** = utf8 **COLLATE** = utf8\_general\_ci ROW\_FORMAT = **Dynamic**;

管理员表（示列代码3-4-4）

* + 1. 构建订单表
       1. E R 图

订单表ER图（图3-4-5）

* + - 1. 关系表

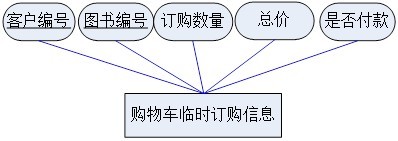
|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| Id | 自动编号 | int | not null（自增字段） |
| OrderId | 订单编号 | varchar(20) | 主码 |
| CustomerId | 客户编号 | int | not null, 引用tb\_customerinfo的外码 |
| Orderdate | 下单日期 | datetime | not nul |
| Ordermount | 总订购数量 | int | not null |
| message | 留言 | varchar(100) |  |
| postmethod | 送货方式 | varchar(100) | not null |
| paymethod | 支付方式 | varchar(100) | not null |
| recevername | 收货人姓名 | varchar(10) | not null |
| receveraddr | 收货地址 | varchar(20) | not null |
| recevertel | 收货人电话号码 | varchar(10) | not null |
| memo | 备注 | varchar(100) |  |
| totalprice | 总卖出价 | money | not null |

订单关系表（表3-4-5）

* + - 1. 示列代码

1. **DROP** **TABLE** IF EXISTS `tb\_order`;
2. **CREATE** **TABLE** `tb\_order`  (
3. `id` **int**(11) NOT NULL AUTO\_INCREMENT COMMENT '订单的自动编号',
4. `OrderId` **varchar**(255) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '订单编号',
5. `CustomerId` **int**(11) NOT NULL COMMENT '客户编号',
6. `BookId` **int**(11) NOT NULL COMMENT '图书编号\r\n\r\n',
7. `ordermount` **int**(11) NOT NULL COMMENT '订购数量',
8. `Orderdate` datetime(0) NOT NULL COMMENT '下单日期',
9. `isPlay` enum('1','0') **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL **DEFAULT** '0' COMMENT '支付状态\r\n1、以支付\r\n2、未支付',
10. `paymethod` enum('1','2','3') **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL **DEFAULT** '1' COMMENT '支付方式\r\n1、支付宝\r\n2、微信支付\r\n3、银行卡支付',
11. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*中间省略\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
12. `address` **varchar**(255) **CHARACTER** **SET** utf8 **COLLATE** utf8\_general\_ci NOT NULL COMMENT '收货地址',
13. **PRIMARY** **KEY** (`id`, `OrderId`) USING BTREE,
14. **UNIQUE** **INDEX** `OrderId\_OF`(`OrderId`) USING BTREE COMMENT '订单编号的唯一性',
15. **INDEX** `CustomerId\_O\_F`(`CustomerId`) USING BTREE,
16. **INDEX** `BookId\_O\_F`(`BookId`) USING BTREE,
17. **CONSTRAINT** `BookId\_O\_F` **FOREIGN** **KEY** (`BookId`) **REFERENCES** `tb\_bookinfo` (`BookId`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,
18. **CONSTRAINT** `CustomerId\_O\_F` **FOREIGN** **KEY** (`CustomerId`) **REFERENCES** `tb\_customerinfo` (`CustomerId`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**
19. ) ENGINE = InnoDB AUTO\_INCREMENT = 18 **CHARACTER** **SET** = utf8 **COLLATE** = utf8\_general\_ci ROW\_FORMAT = **Dynamic**;

订单表（示列代码3-4-5）

* + 1. 构建购物车表
       1. E R 图

购物车ER图（图3-4-6）

* + - 1. 关系表

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 说明 | 数据类型 | 约束 |
| CustomerId | 客户编号 | int | 主码（自增字段） |
| BookId | 图书编号 | int | 主码, 引用tb\_bookinfo的外码 |
| ordermount | 订购数量 | int | not null |
| price | 总价 | money | not null |
| ispay | 是否付款 | varchar(10) | not null，默认为未付款，取“未付款”或“已付款” |

购物车关系表（表3-4-6）

* + - 1. 示列代码

1. **DROP** **TABLE** IF EXISTS `tb\_shopbook`;
2. **CREATE** **TABLE** `tb\_shopbook`  (
3. `shopCarId` **int**(11) NOT NULL AUTO\_INCREMENT COMMENT '购物车表记录的id',
4. `CustomerId` **int**(11) NOT NULL COMMENT '购物车的客户编号',
5. `BookId` **int**(11) NOT NULL COMMENT '图书编号\r\n\r\n',
6. `ordermount` **int**(11) NOT NULL COMMENT '订购数量',
7. **PRIMARY** **KEY** (`shopCarId`) USING BTREE,
8. **INDEX** `CustomerId\_F`(`CustomerId`) USING BTREE,
9. **INDEX** `BookId\_F`(`BookId`) USING BTREE,
10. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*中间省略\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
11. **CONSTRAINT** `BookId\_F` **FOREIGN** **KEY** (`BookId`) **REFERENCES** `tb\_bookinfo` (`BookId`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**,
12. **CONSTRAINT** `CustomerId\_F` **FOREIGN** **KEY** (`CustomerId`) **REFERENCES** `tb\_customerinfo` (`CustomerId`) **ON** **DELETE** **CASCADE** **ON** **UPDATE** **CASCADE**
13. ) ENGINE = InnoDB AUTO\_INCREMENT = 13 **CHARACTER** **SET** = utf8 **COLLATE** = utf8\_general\_ci ROW\_FORMAT = **Dynamic**;

购物车表（示列代码3-4-6）

1. 搭建项目前后端框架
   1. 搭建项目前端框架
      1. 技术栈

|  |  |
| --- | --- |
| 打包工具 | WebPack3.0 |
| 包管理器 | Npm |
| 代码语法检查 | EsLint |
| 前端框架 | Vue 2.0 |
| 状态管理 | vuex |
| 路由管理 | vue-router |
| Ui 框架 | Element Ui |
| 网络请求 | axios |
| CSS预编译 | SCSS |

前端技术栈（表4-1-1）

* + 1. 搭建过程
       1. 安装node.j 使用 npm 包管理工具

工程化的前端项目需要下载各类包，从而搭建完整的项目，npm和yarn 都是常用的包管理工具，在本项目中，使用npm来进行包的管理，而npm是基于Node.js环境来运行的，所以我们需要先安装Node.js，安装好后，也不需要安装npm，因为Node.js集成了npm包管理器的安装。

访问Node.js 官网来下载安装程序进行Node.js的安装

1. https://nodejs.org/zh-cn/

cmd控制台查看npm 包的版本，如果出现版本号，说明安装成功

1. npm -v
   * + 1. 全局安装vue-cli 快速初始化vue项目

在之前，使用vue需要进行大量的配置和繁杂的工作，为了简化程序员的负担，也为了vue的持续健壮的发展，vue社区推出了vue-cli，此插件能快速的构建一个vue项目，省去繁杂的配置工作。

1. npm install -g vue-cli

安装完vue-cli后，创建一个项目文件夹，在文件夹内打开cmd窗口，执行vue-cli 初始化项目的命令

1. vue init webpack

命令执行完后，会生成项目目录，但是相应的项目依赖包文件没有安装，还需要执行

1. npm install

来安装依赖包

* + - 1. 安装技术栈所需要使用的依赖

在本技术栈之中，使用了一些工具来开发，使用哪个工具，就需要安装此工具的依赖包，列如，本项目的UI框架使用了Element Ui，那么就需要安装Element Ui 的依赖包再进行使用。

1. npm i element-ui -S

如上方法，再进行一些相关的配置，就能在项目中使用Element UI 了。

同理，我们就可以用相关的方法来进行其他依赖项的安装，在这里便不加多叙述。

* + - 1. 运行项目，构建项目

WebPack 打包工具在开发模式中，将打包好的文件存放在内存之中，以便快速的开发并进行热加载。但最后项目开发完成，需要放置生产环境中，就先对项目进行打包构建，对项目的各类文件进行压缩打包，大大减少项目体积，最后生成压缩好的项目文件夹，

1. npm run dev  # 通过此命令来启动项目，终端会打印项目的运行端口，在浏览器中进行访问
3. npm run build  # 在开发完成后，通过此命令来构建最终的项目文件，存放于dist目录下
4. # 复制dist目录下的文件，存放于线上服务器，开启线上服务器即可
   * 1. 项目目录

项目搭建完成后，会有以下的项目目录。

本项目名称为milin-book

1. milin-book     # 项目名称 主文件夹
2. │
3. ├─build     # WebPack 配置文件夹
4. │     build.js     # 生产环境构建
5. │     check-versions.js     # 版本检查
6. │     utils.js     # 有关构建的相关工具类
7. │     vue-loader.conf.js     # 加载器配置
8. │     webpack.base.conf.js     # WebPack基础配
9. │     webpack.dev.conf.js     # WebPack开发模式下的配置
10. │     webpack.prod.conf.js     # WebPack 生产环境下的配置
11. ├─config     # 项目配置
12. ├─dist     # 最终构建的文件目录
13. ├─node\_modules     # 项目依赖文件夹
14. ├─src     # 源码目录
15. │  │  App.vue     # vue 根节点
16. │  │  main.js     # vue 入口文件
17. │  ├─api     # 接口调用类文件目录
18. │  ├─assets     # 资源文件目录
19. │  ├─components     # vue公用组件目录
20. │  ├─router     # vue 路由目录
21. │  ├─store     # vuex 状态管理其目录
22. │  ├─style     # 样式文件目录
23. │  ├─utils     # vue开发工具类目录
24. │  └─views     # vue 视图页面目录
25. │
26. └─**static**     # 纯静态资源，不会被WebPack 构建解析
27. │  .babelrc     # babelrc 插件配置文件
28. │  .editorconfig     # 编辑风格文件 来统一编辑风格
29. │  .eslintignore     # ESLint 代码检查忽略文件
30. │  .eslintrc.js     # ESLint 代码检查文件
31. │  .gitignore     # git 体检忽略文件
32. │  .postcssrc.js     # CSS兼容插件配置文件
33. │  favicon.ico     # 网站.ico
34. │  index.html     # 网站html就够
35. │  package-lock.json     # npm版本控制文件
36. │  package.json     # 包管理文件

以上目录结构即为本项目前端项目的目录结构，本项目主要操作src目录，在components目录下新建公用组件，在router中增加路由，在views中编写页面代码，在style文件中编写通用样式，需要调用API则，在API文件中新增接口的调用方法。

* 1. 搭建项目后端框架
     1. 技术栈

|  |  |
| --- | --- |
| 服务器 | Node.js |
| 服务器开发语言 | JavaScript |
| 数据库 | MySQL |
| 服务器框架 | Express |
| 进程管理 | Pm2 服务器进程管理器 |
| Cookie工具 | cookie-parser |
| SQL异步优化工具 | Async |

后端技术栈（表 4-2-1）

* + 1. 搭建过程
       1. 置Express,导出Express 实例

因为本项目的Node.js服务器时基于Express框架的，所以，必须先配置Express框架，导出配置好的Express 实例，再根据实例来开启服务器，访问路由，连接数据库，然后承担起服务器的相应功能。

首先，在项目中新建文件夹config，此文件夹作为存放配置文件的文件夹。在config文件夹下建立express.js文件，此文件用来配置Express

* + - * 1. 引入Express,创建实例App

1. // 初始化Express 框架
2. var express = require('express');
3. module.exports  = function(){
4. console.log('init expesss...');
5. var app = express();
6. }
   * + - 1. 配置bodyParser和CookiePar中间件

要获取客户端通过Post或者Get传入的数据，则需要使用bodyParser工具，要操作Cookie，则需要使用CookiePar工具，这些工具都是Express的中间件，所谓中间件，就在于客户端和服务器通信过程中对请求进行处理过滤的工具。

1. /\*\*\*\*\*\*基于上文配置\*\*\*\*\*\*/
3. **var** bodyParser = require('body-parser');
4. **var** CookIePar = require('cookie-parser');
5. module.exports  = **function**(){
6. // 配置中间件 但还是通过body 来获取
7. app.use(bodyParser.urlencoded({ extended: **false** }));
8. app.use(bodyParser.json());
10. app.use(CookIePar());
11. }

以上配置后，我们就可以在路由处理函数中通过req.body.password来获取POST请求中传来的数据，还可以通过req.query.searchText来获取GET请求传来的数据。

还可以通过req.cookies来获取和操作cookie中的数据。

* + - * 1. 配置跨域和开放资源目录

1. /\*\*\*\*\*\*基于上文配置\*\*\*\*\*\*/
3. module.exports  = **function**(){
4. //2.0 将所有api的请求响应content-type设置为application/json
5. app.all('/api/\*', (req, res, next) => {
6. //设置允许跨域响应报文头
7. //设置跨域
8. // 启用 Node 服务器端的 cors 跨域
9. res.header("Access-Control-Allow-Credentials", "true");
10. res.header("Access-Control-Allow-Origin", "http://localhost:8080")
11. res.header("Access-Control-Allow-Headers", "X-Requested-With, mytoken")
12. res.header("Access-Control-Allow-Headers", "X-Requested-With, Authorization")
13. res.header("Access-Control-Allow-Headers", "Content-Type,Content-Length, Authorization, Accept,X-Requested-With");
14. res.header("Access-Control-Allow-Methods","PUT,POST,GET,DELETE,OPTIONS");
15. res.header("X-Powered-By",' 3.2.1')
16. res.setHeader('Content-Type','application/json;charset=utf-8')
17. **if**(req.method == "OPTIONS") {
18. res.sendStatus(200);
19. }
20. **else** {
21. next()
22. }
23. });
25. // 开放资源 目录
26. app.use('/public', express.**static**('public'));
27. app.use('/node\_modules', express.**static**('node\_modules'));
28. }

因为不同源的域名违反了HTTP的安全处理机制，所以，在开发时，需要进行跨域的配置。

其次，还需要开放服务器的资源目录，让客户端获取到静态资源。

* + - * 1. 配置路由

路由是后端服务器的一个重要环节，配置路由文件，让服务器根据客户端通过不同的方式访问各个地址，来进行不同的处理，从而响应不同的数据。

1. /\*\*\*\*\*\*基于上文配置\*\*\*\*\*\*/
3. module.exports  = **function**(){
4. // 配置router 文件
5. require('../app/routes/milinbook.server.routes')(app);
6. }

文件milinbook.server.routes.js 导出的是一个函数，此函数需要参数Express实例。

因为，在此处导入文件，传入参数，运行此函数，具体关于router文件的配置，在下文的有关router文件配置中。

* + - 1. 创建数据库模型，导出数据库连接对象

服务器需要连接数据库，通过数据库来对数据进行增删改查，在Node.js中，需要先创建数据模型，再连接数据库，最后导出数据库连接对象，才能编写SQL语句，对数据库进行操作。

首先，我们再项目目录中建立app文件夹，在文件夹下建立models文件夹，因为可能存在多个数据库模型，然后在文件夹下建milinbook.server.model.js文件，再此文件中编写代码，建立连接，导出连接。

1. // 数据库连接生成模块
3. // 导入模块
4. **var** mysql = require('mysql');
6. **var** connection = mysql.createConnection({
7. host:'localhost',
8. user:'root',
9. password:'asdzxc456',
10. database:'milinbook'
11. });
13. // 建立连接
14. connection.connect();
16. // 导出连接
17. module.exports = connection

如上代码，我们给出连接的配置，进行MySQL数据库的连接，最终导出连接对象，连接对象再路由处理函数中将会用到，通过数据库连接对象来编写SQL语句，操作数据库。

* + - 1. 连接路由处理文件，导出路由处理函数

当客户端对服务器进行请求时，再路由文件中侦测到请求，再调用路由处理函数，对请求进行处理，然后返回处理后的数据给客户端。

在app文件夹下建立，controllers 文件夹，在其下再建立milinbook.server.controller.js文件，此文件，用来对请求进行处理。

1. // 数据操作模块
2. **var** db = require('../models/milinbook.server.model')
4. module.exports = {
5. // 测试数据
6. getAll: **function**(req, res, next){
7. **var**  sql = 'SELECT \* FROM test';
8. db.query(sql,**function** (err,result) {
9. **if** (err) {
10. **return** res.status(200).json({
11. data:**null**,
12. code:0,
13. message:"server data err"
14. });
15. } **else** {
16. res.status(200).json({
17. data:result,
18. code:0,
19. message:"data ok"
20. });
21. }
22. })
23. }
24. }

如上的测试代码，假入我们在路由文件中侦测到用户的请求，随即调用getAll请求处理函数，传入请求对象和响应对象。

加载上文中配置的数据库连接对象db，通过db,query方法，传入SQL和回调函数，操作数据库，以及向客户端响应处理后的数据。

* + - 1. 创建路由文件，侦测路由的请求

本项目的后端服务器，用于提供路由地址，前端根据路由地址，来进行相应的请求，传入后端所需要的数据，又从后端获取想要得到的数据。

在app文件夹下建立routes文件夹，再在文件夹下建立milinbook.server.routes.js文件，此文件用于侦测路由请求，调用请求处理函数的功能，暴露路由对象的功能。

1. **var** MilinBookController = require('../controllers/milinbook.server.controller');
2. module.exports = **function**(app){
3. // 测试
4. // 监听url 调用数据操作模块中的方法
5. app.route('/api/test')
6. .get(MilinBookController.getAll)
7. }

如上代码，导入milinbook.server.controller.js文件，获取到请求处理函数对象，通过之前在Express配置文件中运行路由文件导出的方法，传入的app(Express实例)参数，再通过app.route来对路由进行监听。

同一个路由地址，可能会有不同的请求方式。

* + - 1. 建立服务器启动文件

通过以上Express框架的配置，路由的监听，数据库的连接，请求的处理各类文件的建立，现在已经能够开启Node.js服务。我们将建立一个开启服务的文件，通过Node.js运行此文件，来开启服务。4

在项目目录中建立bin文件夹，在文件夹下建立[www.js](http://www.js)文件，在此文件中编写开启服务器的代码。

1. **var** app = require('../app');
2. **var** config = require('../config/config');
4. app.listen(config.port, **function**(){
5. console.log('app started, listening on port:', config.port);
6. });

通过

1. node bin/www

运行服务器。再在前端调用API接口，即可获取所需数据。

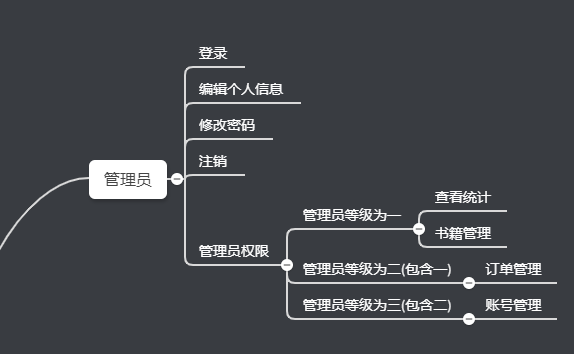
* + 1. 项目目录

配置完成后，可得以下的后端框架的目录。

1. milin-book-api   # 项目名称
2. ├─app
3. │  ├─controllers
4. │  │      milinbook.server.controller.js    # 请求处理文件
5. │  ├─models
6. │  │      milinbook.server.model.js      # 数据库模型，导出连接对象
7. │  ├─routes
8. │  │      milinbook.server.routes.js        # 监听路由
9. │  └─utils
10. │          admin.js   # 有关管理员的公用方法
11. │          user.js       # 有关用户公用方法
12. ├─bin
13. │      www     # 启动服务器的文件
14. ├─config
15. │  │  config.js   # 开发模式及生产模式的配置
16. │  │  express.js   # Express框架的配置
17. │  └─env
18. │          development.js      # 端口配置
19. ├─node\_modules   # 依赖包文件夹
20. └─**public**   # 资源目录
21. └─upload
22. └─img
23. │  .bowerrc
24. │  .gitignore
25. │  app.js   获取Express 实列
26. │  package-lock.json
27. │  package.json   # 依赖包记录
28. │  processes.json

以上就是本项目基本的项目目录，主要开发在于app文件夹中，在 milinbook.server.routes.js路由文件中添加路由连接，暴露API接口，侦测到客户端的访问后，在 milinbook.server.controller.js 编写请求处理函数，在路由文件中调用处理函数进行处理。

1. 网上书店后台管理功能
   1. 管理员功能

依据上文有关后台管理中管理员的需求，通过编写前端和后端代码，实现了如下的功能。

管理员功能（图 5-1）

* + 1. 管理员登录

管理员通过最高管理员在数据库添加账号后，在登录页面进行登录，登录成功后，跳转到管理界面，根据管理员权限的不同，各有不同的功能。

如图在此页面进行管理员的登录，输入账号时，会通过正则表达式，检验字符串的合法性（是否符合邮箱规则），也会通过正则表达式，检验密码的合法性（同时带有字母和数字），如果输入不合法，则点击登录时，不会进行登录的请求。

在输入都合法后，点击登录，会判断用户名时否存在和密码时否正确，如有错误，会进行相关的提示。

在用户名存在，并且密码也正确后，会判断当前登录的此管理员账号时否被禁用，如未被禁用，则成功跳转到管理界面，如账号被禁用，则进行相关的提示，不允许登录。

* + 1. 编辑管理员个人信息

登陆成功之后，即跳转到后台管理界面，在个人头像处，会有编辑个人信息，修改密码，注销等功能。

如上图，点击编辑按钮，就行个人信息的编辑。弹出编辑框。

* + 1. 修改管理员个人密码
    2. 注销登录

点击注销登录的按钮，即跳转到管理员登录页，重新进行登录。

* + 1. 管理员权限

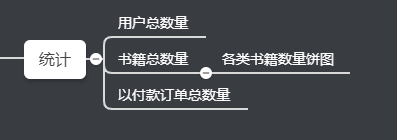
管理员具有权限等级

* 等级一：查看统计、管理书籍
* 等级二：查看统计、管理书籍、管理订单
* 等级三：查看统计、管理书籍、管理订单、账号管理
  + - 1. 管理员等级一

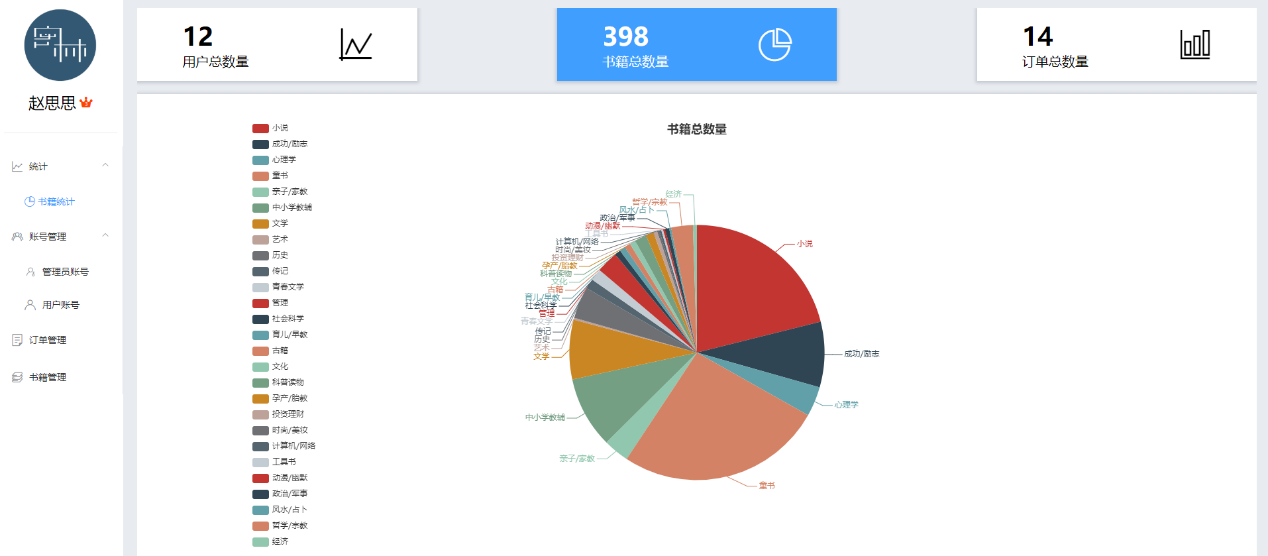
等级为一的管理员，只能查看书籍的统计和书籍的管理。

* + - 1. 管理员等级二

如图，等级为二的管理员，只能查看书籍的统计、订单管理、书籍管理。

* + - 1. 管理员等级三
  1. 统计功能

统计页，可以查看当前注册用户的总数量、书籍总数量、已付款订单总数量、并通过饼图的方式，可视化的显示各类书籍数量。



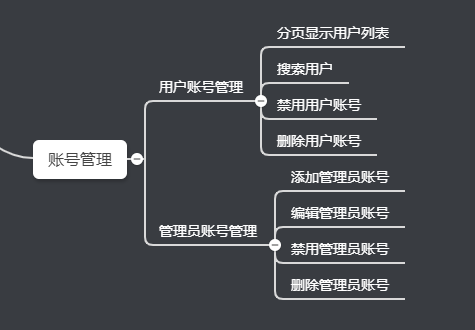
* 1. 账号管理

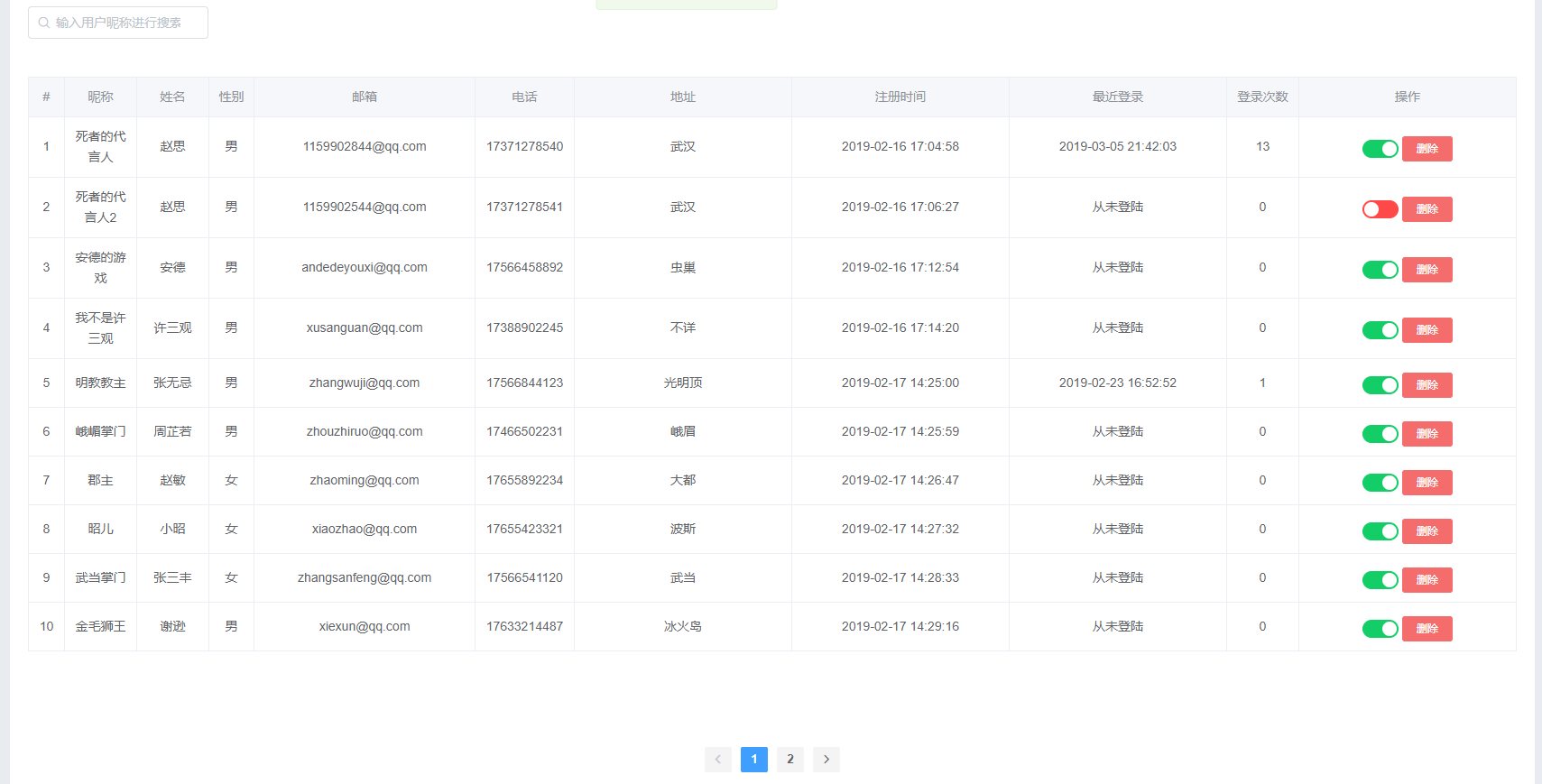
账号管理页（需管理员等级为一才能进入）的功能分为两大部分。

* 用户账号管理
* 管理员账号管理

在用户账号管理中，可以搜索用户账号、禁用用户账号、删除用户账号

在管理员账号管理中，可以添加管理员账号，编辑管理员账号，禁用管理员账号、删除管理员账号。



* + 1. 用户账号管理
       1. 分页显示用户列表

如图，可点击下方分页器进行分页的显示。可以查看到，用户昵称、姓名、性别、邮箱、电话、地址、注册时间、最近登录时间、登陆次数等信息。

* + - 1. 搜索用户账号

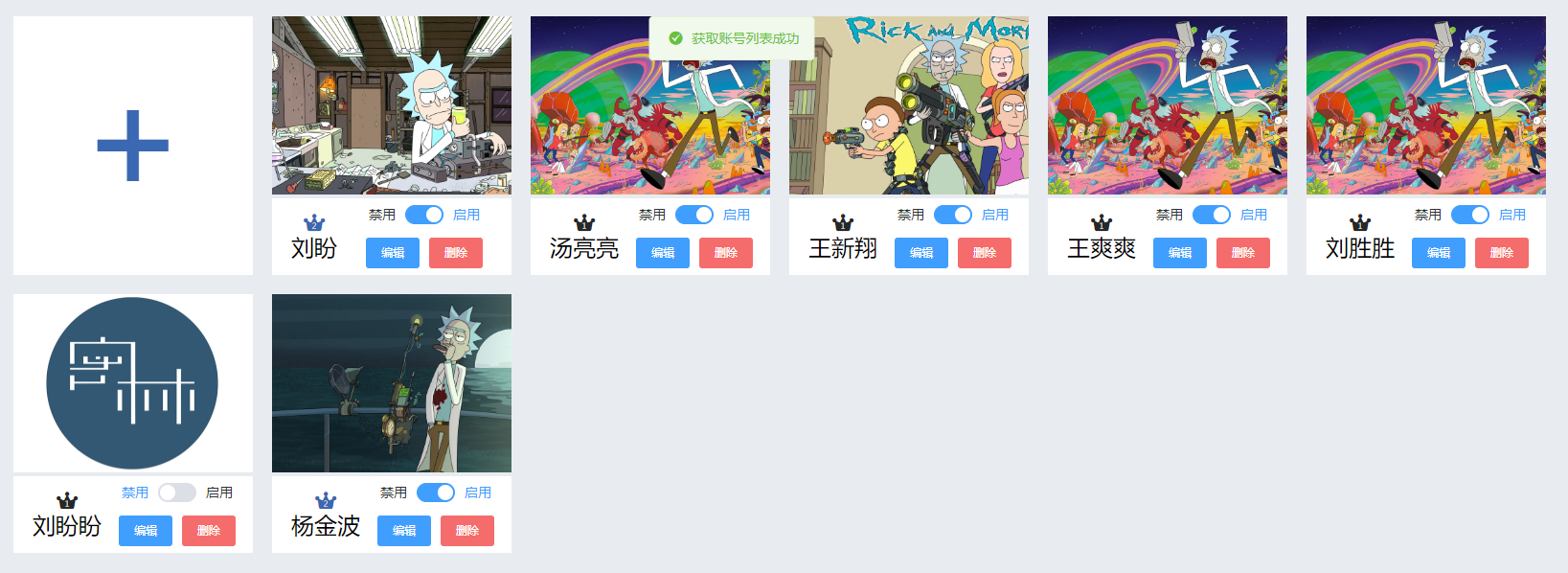
输入用户昵称关键字，进行用户账号的搜索。列表即显示符合关键字搜索的用户账号

* + - 1. 禁用和删除用户账号

在用户列表中。每一行用户都有删除列，在此列中可以对用户的账号进行禁用和删除。

* + 1. 管理员账号管理

进行管理员账号页，可以查看到当前所有的管理员账号。



* + - 1. 添加管理员账号

点击上图中的“+”号，即弹出添加管理员账号的弹窗，在弹窗中输入信息，即可，通过正则合法性的验证，即可添加管理员账号。

新添加的管理员账号有默认的密码和默认的头像，默认的密码是不可更改的，默认的头像可以点击头像选择器，再次上传头像。

* + - 1. 编辑管理员账号

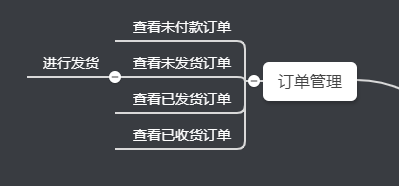
点击管理员面板上的编辑管理员账号按钮，即可编辑管理员账号



* + - 1. 禁用和删除管理员账号

点击面板上的开关，即可启用或者禁用此管理员账号，点击删除，确认删除后，即可删除此管理员账号。

* 1. 订单管理

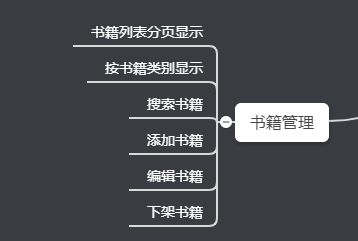
在订单页，即可查看各类订单，并可以对各类订单进行管理，列入，未发货而已付款的订单，则需要进行发货。

* + 1. 查看未付款的订单列表

如上选项卡为“未付款”，即显示未付款的订单列表，可以查看到订单信息有：订单编号、下单日期、书名、书籍原价、折扣、书籍现价、购买数量、总价。

* + 1. 查看未发货订单并进行发货

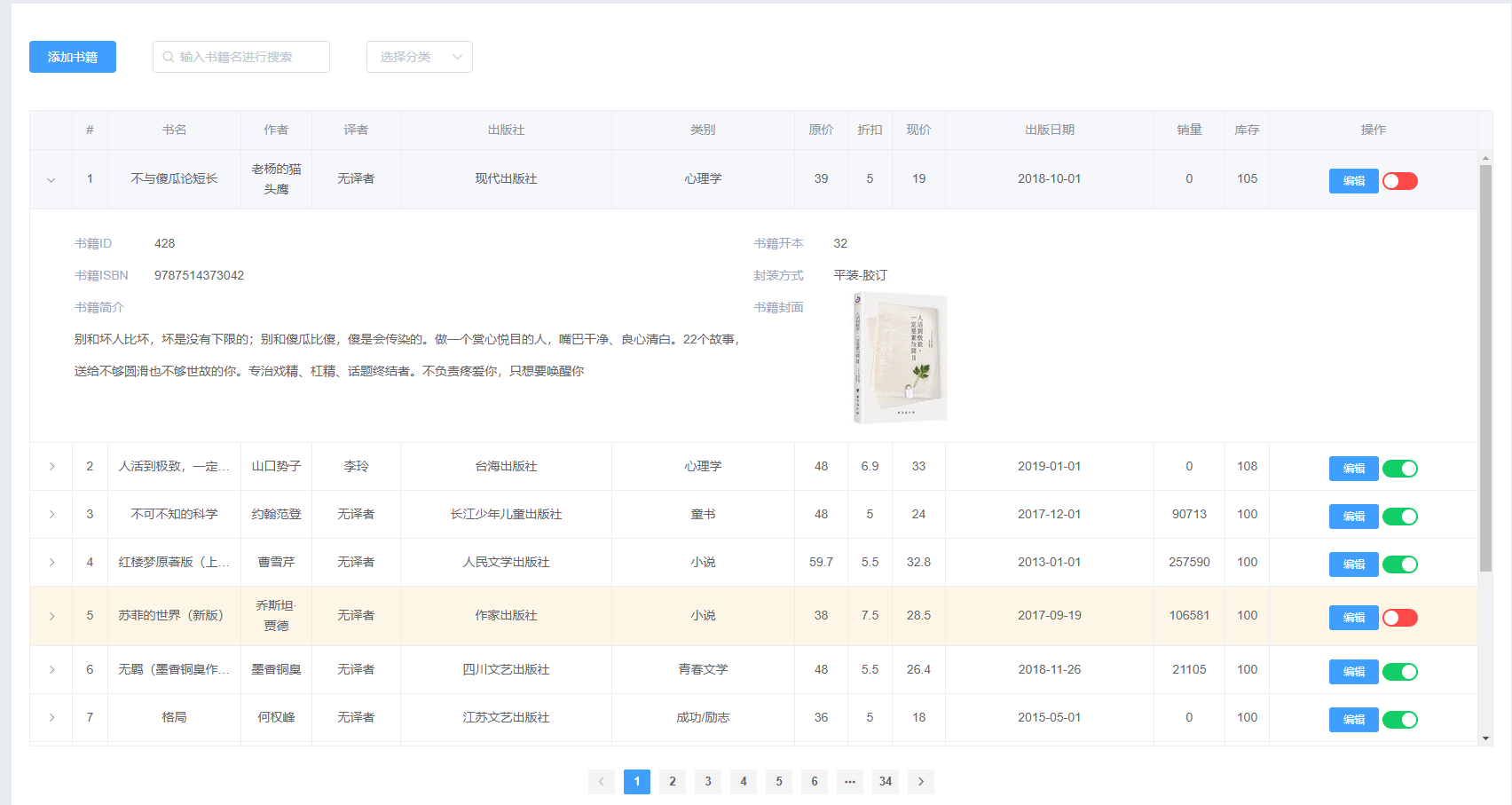
点击选项卡于未发货位置，则即可查看未发货的订单，未发货的订单字段信息于为付款订单信息一致，只是多了一个操作列，可以对已经付款的订单进行发货。

* + 1. 查看已发货的订单
    2. 查看已收货的订单
  1. 书籍管理

书籍管理页，可以对书籍进行管理，分页显示书籍列表，按书籍的类别娴熟书籍列表，这两个条件也可以叠加起来对书籍进行筛选。

还可以输入书籍名称，对书籍进行搜索。

点击添加按钮，可以对书籍进行添加，还可以编辑书籍的信息以及下架书籍，下架后的书籍在前台客户中的书籍列表是不显示的。

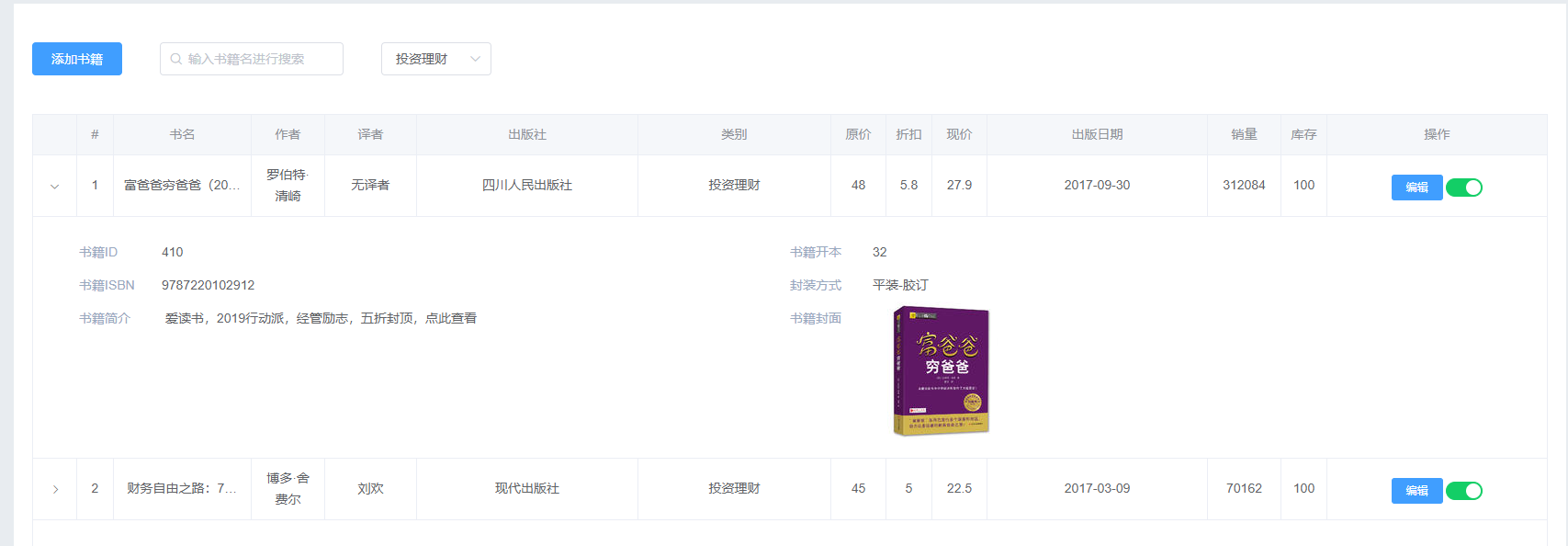
* + 1. 分页显示书籍列表

如图所示，点击分页器，即可分页浏览书籍列表。

* + 1. 按书籍类别显示

书籍默认分类为显示全部分类，下拉书籍分类选择器，即可之查看此分类下的书籍。

如我选择“投资理财”分类，则列表只显示“投资理财”类的书籍



* + 1. 搜索书籍

在搜索输入框中输入书名关键字，即可通过此关键字对书籍进行搜索。

如下图，输入“格局”书名关键字进行搜索，得到搜索结果。

* + 1. 添加书籍

书籍管理中很重要的一部分是对书籍进行添加，点击“添加书籍”按钮，即弹出添加书籍信息的弹窗，可对书籍进行添加。

需添加的字段有：

* 书名 【必填】
* 作者 【必填】
* 译者
* 出版社 【必填】
* 书籍类别 【必填】
* 书籍定价 【必填】
* 折扣
* 书籍现价 【必填】
* 出版日期 【必填】
* 库存 【必填】
* 开本 【必填】
* 书籍ISBN 【必填】
* 封装方式
* 书籍简介
* 书籍封面图



* + 1. 书籍编辑

对书籍信息进行编辑是维护书籍的重要工作，

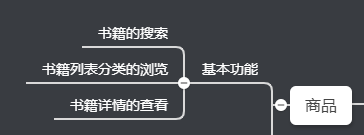
点击书籍行的编辑按钮即弹出编辑书籍信息的弹窗，必填字段如添加书籍信息一样，可以对书籍的封面图进行更换等操作。



* + 1. 下架书籍

点击开关，即可下架或者上架书籍，下架的书籍在客户端页面不会显示

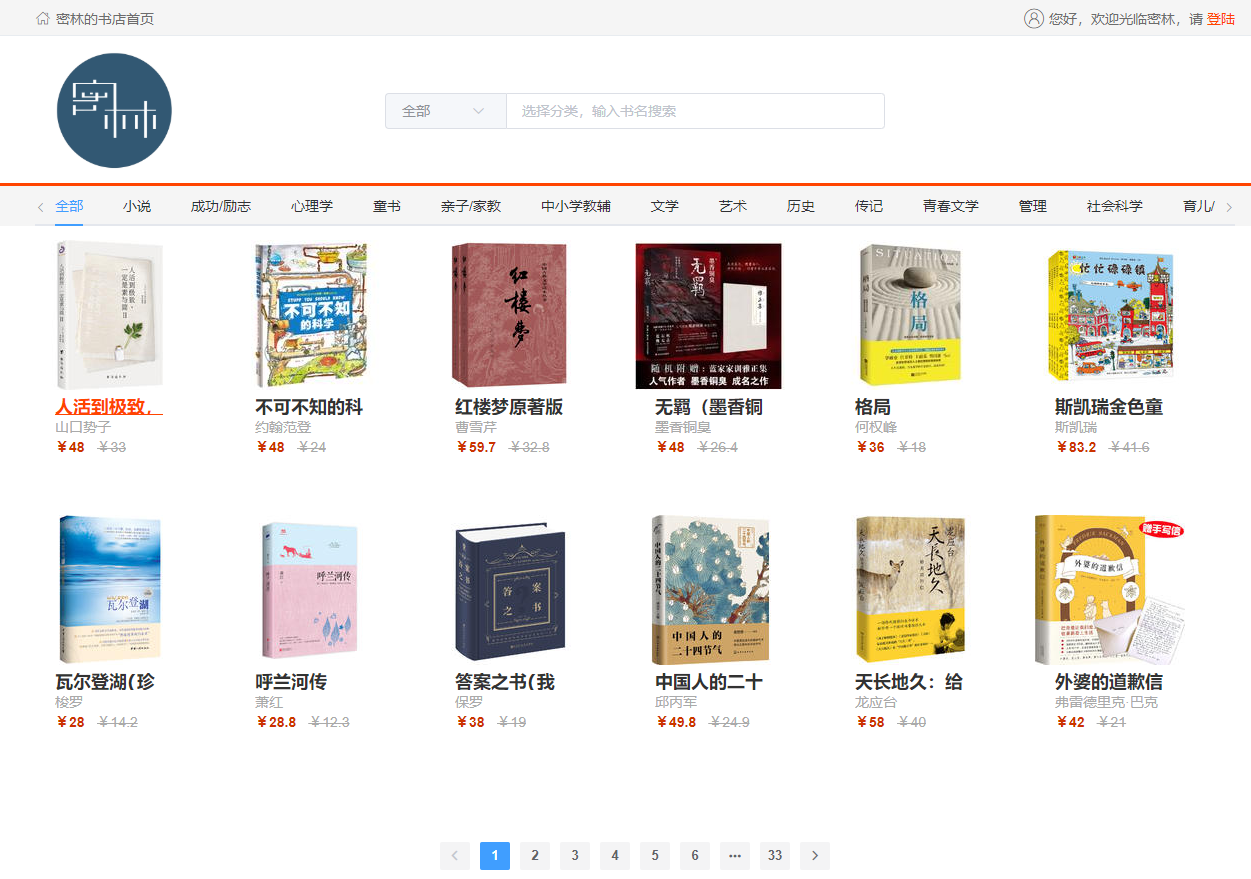
1. 网上书店前台功能
   1. 未登录时商城基本功能



在用户未登录时，进入书城，会有一些基本的功能，可以对书籍进行搜索，也可以对书籍进行分类浏览，还可以对书籍详情的查看，但是不能进行购买和添加到购物车，如果在未登录的情况下，尝试购买和添加书籍到购物车，就会跳转到用户登录界面，提示用户进行登录后才能进行购买书籍。

在为登录时，会有提示登录的入口



在密林网上书城首页，即可分页浏览商品，通过分类或者关键字对书籍进行搜索

* + 1. 分类浏览

如图，分类类别为社会科学类，书籍列表相应的显示社会科学类别的书籍。

* + 1. 书籍名称关键字搜索

如图，在搜索框搜索答案之书，即可在书籍列表显示。

搜索书籍时，可以书籍名称关键字和书籍类别结合搜索，提升搜索精度。

* + 1. 书籍详情的查看

如图，点击《史记（精装新版，全三册）》书籍，弹出显示书籍详情的弹窗，在详情弹窗中，可以看到书籍简介和作者、出版时间、出版社、销量、原价、现价。

如果需要购买，还可以选择送货地址，在点击加入购物车和立即购买时，会判断用户是否登录，如果用户未登录，则会跳转到登录页面，进行登录。



* 1. 用户登录注册
     1. 用户登录

用户想要在书城进行购买，则需要进行登录，跳转到登录页面后进行登录，输入邮箱格式的账号和数字与字母混合的密码，才能通过验证，验证规则由正则表达时进行匹配。当密码和账号相匹配时，将跳转到商城首页，不过被禁用

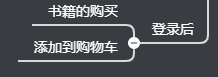
的用户账号，不会进行跳转，会提示账号已被禁用。

* + 1. 账号注册

当用户不存在账号时，可以点击注册账号进行账号的注册，注册账号要求

填写昵称、邮箱账号、姓名、性别、手机号、城市、密码、确认密码，还可以进行头像的上传，这里提供默认的头像。

在以上填写的字段中，邮箱账号作为登录时使用的账号。

* 1. 用户登陆商城后的功能
     1. 书籍的购买和添加购物车

在用户登录之后，点击书籍的详情就可以对书籍进行购买和添加到购物车了。

* + - 1. 书籍的购买

点击书籍详情，弹出详情框，点击购买按钮，购买书籍，在购买之前，需要填写地址，没有填写地址的话，不能进行购买，并有提示。



填写好地址后，点击购买按钮，弹出订单详情。

订单详情信息存有客户昵称、订单编号、下单日期、商品名称、商品原价、折扣、商品现价、购买数量、收货地址、留言和付款方式。

最终，会计算订单的总价格，会提示付款，如果点击取消付款或者点击页面空白并且确认取消付款的话，订单会添加到用户的未付款订单列表中，下次想要付款，可以进行付款。

如果直接点击了付款，订单会添加到用户未发货的订单列表中，等到卖家发货。

* + - 1. 添加到购物车

点击添加购物车后，商品添加到购物车之中，会在用户的购物车列表中显示，并且也能直观的子啊界面显示购物车和订单的数量。

* + 1. 查看购物车列表

点击顶部栏的“我的购物车”即可进入购物车列表，查看购物车中的商品。

如图，在购物车列表中，可以对购物车中的商品进行购买，也可以删除购物车中的商品。

点击删除后，会提示用户是否确认删除商品，用户确认删除，则会删除购物车中的商品。

点击购买后，会弹出订单详情的弹窗，重复商品购买的流程。

* + 1. 查看订单列表

点击顶部栏的“我的订单”后，进行订单列表页，订单列表页通过选项卡和标签页分类显示订单列表。

订单类别有：未付款、未发货、以发货、已收货

* + - 1. 未付款订单列表

在未付款订单列表中，具有删除订单和支付订单两个功能，可以对订单进行删除和支付。

点击删除时，会确认用户是否删除此未付款订单，如果用户确认删除，则进行调用后端接口，进行删除。

点击支付时，会弹出订单详情表的弹窗，显示总价格的信息，重复支付商品的流程，

* + - 1. 以发货订单列表

点击以发货选项卡进行以发货订单列表，显示以发货订单信息。

在操作列，可以对已发货的订单进行收货，确认收货的订单，进入到以收货的订单列表中。

* + 1. 用户个人信息管理

把鼠标放入用户的昵称上时，可以弹出操作用户个人信息的面板，面板带有用户的头像，并且可以修改用户密码和注销当前账号。



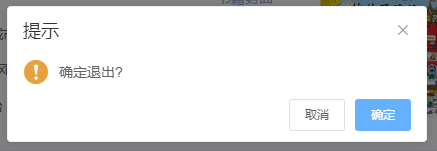
* + - 1. 修改用户密码

在面板上点击修改密码后，弹出修改密码的弹窗，输入合法的密码字符组合，通过正则表达式的验证，即可调用后端接口进行请求修改密码，如果旧密码匹配则进行修改，并在修改成功后，跳转到用户登录页面。

* + - 1. 注销登录

点击注销登录，提示用户是否确认注销当前账号。

点击确认后，立即跳转到用户等略界面。



1. 结论总结
   1. 步骤归纳及课题分析问题结论

制作《网上图书书城》这一全栈性的电子商务网站大约花费了3周，在这三周的时间中，分别完成了数据库的设计和爬虫爬取测试数据、后端服务器的代码编写、前端页面和逻辑代码的编写。

构建全栈性的网站，需要对Web开发大概的全面了解，对SQL技术和后端服务器技术，前端技术都要有一定的实践。步骤如下。

**第一：确立需求文档**

项目确立是以需求为前提的，如果没有明确的需求，那么就没有构建项目的基础，有明确的需求文档，知道项目做出来的具体功能如何，我们就可以按照需求文档去设计数据库，去编写接口，去确定前端页面风格等等。

**第二：设计数据库**

数据库的设计是后端项目的基础，因为后端逻辑代码的编写，无非就是对数据增删查改，而，如何科学的存放数据，如何让数据的增删查改更具有逻辑性，且不冲突，这都是至关重要的，对于此，在没有搭建数据库的经验下，需要查找资料，阅读别人完善的数据库设计文档，熟练的构建ER图和阅读ER图，能对整个数据库系统快速的了解。

**第三：搭建服务器和服务器框架**

可选择的服务器及服务器语言有很多，但选择一种适合自己的才是最好的，开发大中型项目主要用PHP和Java以及C++，但在于小型项目，可以使用Node.js，但是在有自己熟练使用的服务器前提下，应该尽量使用自己所熟悉的服务器来进行服务器的搭建。搭建服务器后，最好选择一种框架，这样能节省我们编写后端服务器代码很多时间，因为框架提供了很多成熟的解决方案，列入数据库的连接，模型的搭建，路由的构建和监听等等。

**第四：编写前端页面和逻辑代码**

如果时间宽裕或者有PS的基础，可以自己来编写页面的PSD原稿，如果没有，则可以选择一种前端UI框架，快速的进行前端页面的开发，并且现在的前端框架都可以自定义主题。

因为后端提供的就是API接口，前端编写完页面后，就可以通过接口，完成具体的功能。

* 1. 研究目的有关新老技术问题结论
     1. 新技术的优点

本项目主要使用新技术有单页面富应用的Vue2.0框架， 和前端项目化打包和构建的WebPack3.0，开发中小型服务器的Node.js，他们各有优点。

**第一：组件化单页面富应用Vue 2.0**

现在的前端页面，已经不流行单独编写HTML代码，一个一个页面的去访问的时候，除非在于服务端渲染完全的静态化项目。编写一个HTML页面，在一个页面里通过路由锚点跳转，理论来说，可以编写如淘宝那样庞大的网站也没有问题，并且组件化的编写前端页面，可以做到很好的复用，不用到处复制粘贴HTML代码和CSS代码。

**第二：WebPack 3.0打包和构建前端项目**

既然现在的前端都是项目化的构建，那么必然最后开发完成后，整个项目的打包，部署到服务器上及发布线上。

前端代码拥有大量的JS代码和CSS代码以及静态文件的处理，这就要求最终的生产模式的版本要求精简规范。而WebPack则主要提供这样的功能，他通过配置文件，把整个项目的逻辑代码，样式代码和静态文件集中起来，分类处理，通过各种解析器去解析，得到合并后的代码， 并进行压缩，做到项目的最精简化，

**第三：使用Node.js搭建服务器**

Node.js是JavaScrip的运行环境，它可以通过JavaScript编写代码，基于谷歌浏览器的V8引擎开启服务，完成各样的服务器功能。Node.js在于它的短小精悍，使用它来开启一个服务所用代码也不过3行，并且由于其服务器语言为JavaScript，是前端开发人员已经熟练掌握的语言，不需要去专门学习新的语言，学习的成本也很低，可以说是前端开发人员开启中小型服务器的首选选了。

* + 1. 新技术的缺点

新技术过些时候也就是老技术，新技术也是会有缺点，这些缺点页促使着前端的进步和发展。

**第一：组件化单页面富应用Vue 2.0**

组件化的富应用虽然非常方便以及友好，但是就如前文所说，其理论上能开发像淘宝这样庞大的Web网站，可真正谁也没有这样做，这就说明，Vue并不是很适合开发大型网站，（当然Vue完成可以，只是相对于其他选择）。

**第二：WebPack 3.0打包和构建前端项目**

WebPack的打包能力是毋庸置疑的，但这些能级是建立在各种各样的解析器和配置上的，列如光一个Vue项目的WebPack配置，多达三个文件，并且还要经过不断的调式，学习WebPack打包工具具有很大的时间成本。

当然，现在WebPack4.0推出了零配置启动的解决方案，但是其隐藏了配置文件又反而带来使用在WebPack3.0时代的插件时，不好配置的问题。

所以，WebPack最终缺点在于配置的繁琐和调式难度。

**第三：使用Node.js搭建服务器**

Node.js是中小型服务器的首选，但也正是因为如此，它始终是比肩不上像C++、Java这样的重量行服务器，虽然Node.js可以通过服务器集群来运行，但是在其他问题的解决方案上，Node.js绝对没法于老牌服务器相比，这样正式Node.js的缺点。

1. 致谢

大学的学习生活即将结束，在此，我要感谢所有曾经教导过我的老师和关心过我的同学，他们在我成长过程中给予了我很大的帮助。

本毕业设计和本毕业论文是我在实习的时候完成的，通过在公司的实习，更让我认识到现实开发的复杂性，但是，公司给了我很多学习的机会，在这里，我也非常的感谢他们，谢谢他们！

必须一提的是，需要感谢一位同事也是同学，它教会使用Node.js来编写爬虫程序，以此，我来爬取的测试所需的图书数据，这帮了我极大的忙，为我省去了很多时间，专心用于代码的编写，在这里说一声谢谢！

但很重要的是感谢指导老师，指导老师给我们确立了课题，确立了目标，督促和评阅我们的实习日志，很是幸苦，在这儿里也谢谢她！

最后，祝愿老师们：身体健康，工作顺利！祝愿同学们：事业有成！祝我既将离开的母校桃李满天下！

1. 参考资料

[1] 《数据库系统概论》 2006.01 清华大学出版社

[2] 《数据库原理与应用》 2008.05 高等教育出版社

[3] 《SQL Server实例教程（2008版）》 2012.1 电子工业出版社

[4] MySQL 菜鸟教程 http://www.runoob.com/mysql/mysql-tutorial.html

[5] 《Vue 实战》 2017.10 清华大学出版社

[6] Vue 官方文档 https://cn.vuejs.org/

[7] 《Node.js 实战》 2014.3 人民邮电出版社发行部

|  |  |  |
| --- | --- | --- |
| 指导教师意见 | 指导教师（签名及时间）： | |
| 答辩意见 | 答辩小组组长（签名及时间）： | |
| 成绩评定 | 毕业设计（论文）成绩 |  |
| 答辩成绩 |  |
| 总评成绩 |  |