《数据库应用系统》设计概要/详细设计

项目名称：数据库应用系统

子系统名称：线上图书商城

当前版本：V.12.07

最后修改时间：2023年12月7日

版权所有◎2023年

# 引言

## 系统标识

项目名称：数据库应用系统——线上图书商城（Database Application System——Online BookStore）

项目简称：图书商城（DBAS——OBS）

项目组成员：

|  |  |  |
| --- | --- | --- |
| 姓名 | 职务 | 联系方法 |
| 刘征昊 | 开发工程师（后端） | 21371445 |
| 刘子晗 | 开发工程师（数据库） | 21371440 |
| 于珈尉 | 开发工程师（前端） | 21371048 |
| 杨舜杰 | 开发工程师（前端） | 21373140 |
| 吴程翔 | 开发工程师（前端） | 21373305 |

## 编写目的

人们现在的生活方式因为网络的普及发生了巨大变化，由于电子商务在人们的视野中出现,人们对电子商务额外的关注。人们可以足不出户买到世界各地的图书,网上商城可以销售各式各样的图书,其中包括虚拟商品、电子商品、日常生活用品等等。我们的目标不只是在网上展示我们的图书，更重要的是，让更多的客户了解图书创造更多的商机。所以我们目前的挑战是前台界面的设计，要把顾客的眼球吸引住，选则比较人性化的界面设计，要更直观的表现，从而上顾客买到喜欢的图书。

本系统的主要意义在于，全力以赴为用户提供一个操作方便，界面简洁，信息直观的网上交易系统。使用该系统的用户，可以先浏览到图书信息、系统公告，并可以注册成为本网站的用户，可以利用购物车选择自己想买的图书，然后向商家提交订单，从而完成网上的交易流程。

## 本文的读者群

从事计算机科学领域的相关人员（如老师、学生、工程师等）。

知识储备：MySQL操作（包括但不限于增删改查）；Django框架；Django操作MySQL数据库；Node.js搭建Web服务器；Vue.js框架；前后端连通

## 专门术语及缩略词定义

无

## 参考文献

1. <https://www.djangoproject.com/>
2. <https://www.django-rest-framework.org/>
3. <https://cn.vuejs.org/>
4. <https://element-plus.org/zh-CN/>
5. <https://www.mysqlzh.com/>

# 概要设计

## 设计注意事项、原则、目标及项目规范

### 设计方法

图书商城系统的设计遵循“低耦合，高内聚”的设计原则，将系统功能分解为前端的信息展示与用户操作、后端的操作请求处理及数据库交互以及数据库管理系统的数据增删改查实现三个较为独立的部分。

具体方法上，故本项目采用前后端分离的方法，将三个部分分别交由Vue框架、Django框架、MySQL数据库来完成。

### 开发框架选择依据

Vue作为一款轻量级前端框架，大小只有18-21KB，工程搭建简单，只需几行命令符，且主体语言为JS/TS，开发者可以灵活地将其他框架的项目迁移到Vue，具有很高的集成能力；Vue框架将组成页面的HTML、CSS和TS/JS合并到一个组件中，可以被其它组件或页面重复利用，这种组件化的特性可以很好地将一个庞大复杂前端工程拆分为一个一个的组件，重复利用的性质也大大提高了开发效率；MVVM为Vue框架提供了数据的双向绑定，减少了DOM操作，更高效地实现了视图和数据的交互，同时MVVM使界面、交互和数据层分离，便于设计人员负责设计界面、后端开发人员提供接口、前端开发人员专注于业务交互逻辑的实现；Vue有许多npm扩展包和开发工具，可以在一个文件夹下统一管理所有外部插件的全局使用。基于上述理由，本项目采用Vue作为前端框架

Django的主体语言为Python，Python语言使用简单，相较于其它编程语言（如C++、Java），可以使用更少的代码来完成更多的功能，同时有很多功能丰富且安装便捷的第三方库；Django支持ORM，可以让开发人员更加轻松地操作数据库，而不用写繁琐的SQL语句；Django提供了完整的MVC架构，让开发人员可以更加清晰地组织和管理代码；Django还提供了大量的内置功能（比如表单处理、用户认证等等）。基于上述理由，本项目采用Django作为后端框架。

MySQL数据库体积小，且优化了SQL查询算法，本项目数据规模并不算大（最大的数据表也不过300条数据），采用MySQL数据库可以满足更高的性能需求；MySQL为多种编程语言提供了API，比如本项目所使用的Python；MySQL提供了多语言支持，常见的编码如中文的GB2312、国际的UTF8等都可以用作数据表名和数据列名。基于上述利用，本项目采用MySQL作为数据库存储。

### 协议/接口要求

遵循RESTful API接口规范。

### 编程语言与代码规范

编程语言上，前端采用Vue+Typescript，后端采用Python。具体而言，前端采用Vue框架，后端采用Django框架。

代码规范上，总体而言：变量命名中前后端统一采用驼峰命名法，数据库内表的属性名统一采用下划线命名法；缩进中，如果地位相等则不需要缩进，如果属于某一个代码的内部代码则需要缩进；代码中添加必要的注释。

对于不同的编程语言，具体如下：

Python语言规则：只对软件包/模块使用import语句，而不对独立的类型、类或函数使用；对每一个模块使用完整路径来导入；避免可变的全局声明；仅对简单情况使用lambda表达式；对支持它们的类型使用默认的迭代器，如列表、字典等；如果可能的话，使用“隐式”False；当有明显的优势时，请明智地使用装饰器；避免和限制地使用staticmethod classmethod；可以根据PEP-484使用类型提示注释Python代码，并在构建时对代码进行类型检查；

Python风格规则：不要用分号终止一行；最大行长度为80个字符；用4个空格缩进代码块；更多地使用f字符串，不要用格式化%format；

Typescript语法规范：一般情况下，标识符不使用$；标识符禁止使用\_作为前缀或后缀；命名应当具有描述性且易于读者理解；省略对于Typescript多余的注释；不要使用@override；调用构造函数时必须使用括号；不要使用#private语法声明私有成员；

Typescript语言特性： 必须使用const或let声明变量，尽可能使用const；实例化异常对象时，必须使用new Error()；对对象迭代时，必须使用if语句对对象的属性进行过滤；

### 模块之间的数据交互

对于前端与后端的数据交互，本系统统一使用JSON格式来编码数据，并将数据封装在HTTP协议的GET/POST请求中，在前后端之间进行传输。

对于后端与数据库的数据交互，则通过pymysql包所提供的SQL编程接口实现。

### 系统设计目标

遵循简洁、易用的设计目标，为用户提供跟为友善的环境。

该系统主要实现以下目标：构造构造数据库、实现数据查询功能、实现数据统计功能、数据添加功能、数据删除功能。以上功能将通过B/S架构来实现。

此外，本系统还应具有身份验证、阻止非法行为等功能，实现较高的安全性。

## 总体结构设计概述

## 构件设计

### 构件1：用户登录

#### 功能描述

用户输入自己的用户名和密码，登录系统

#### 对象模型

#### 接口规范

请求的URL：/bookStore/api/users/custom\_filter

请求方式：GET

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 | 是否必须 |
| uname | string | 用户昵称 | Y |
| passwd | string | 用户密码 | Y |

返回参数：

1. [

2. {

3. “uid”: int,

4. “uname”: string,

5. “pwdhash”: string,

6. “email”: string,

7. “tel”: string

8. }

9. ]

10.

### 构件2：用户注册

#### 功能描述

对于一个未在该系统中注册的用户，可以在该系统中注册一个账号，输入必要的用户名和密码，可以选择输入邮箱、电话

#### 对象模型

#### 请求参数

请求的URL：/bookStore/api/users/

请求方式：POST

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 | 是否必须 |
| uname | string | 用户昵称 | Y |
| passwd | string | 用户密码 | Y |
| email | string | 用户邮箱 | N |
| tel | string | 用户电话 | N |

返回参数：

1. {

2. “uid”: int,

3. “uname”: string,

4. “pwdhash”: string,

5. “email”: string/null,

6. “tel”: string/null

7. }

8.

### 构件3：高级搜索

#### 功能描述

获取全部的书籍信息

#### 对象模型

#### 接口规范

请求的URL：/bookStore/api/books/

请求方式：GET

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 | 是否必须 |
| page | int | 搜寻的页数 | Y |

返回参数：

1. {

2. “count”: int,

3. “next”: string,

4. “previous”: string/null,

5. “result”: [

6. {

7. “book\_id”: int,

8. “authors”: {

9. “author\_id”: int,

10. “aname”: string

11. },

12. “publishers”: {

13. “publisher\_id”: int,

14. “pname”: string

15. },

16. “category”: {

17. “category\_id”: int,

18. “category\_name”: string

19. },

20. “bname”: string,

21. “price”: double,

22. “pub\_year”: int,

23. “url”: string,

24. “isbn”: string,

25. “sales”: int,

26. “rate”: double

27. },

28. ]

29. }

30.

### 构件4：书籍浏览

#### 2.3.4.1 功能描述

#### 2.3.4.2 对象模型

#### 2.3.4.3 接口规范

### 构件5：获取收藏夹

#### 功能描述

根据用户的输入来返回用户期望查询的个人收藏夹内容

#### 2.3.5.2 对象模型

#### 2.3.5.3 接口规范

请求的URL：/bookStore/api/collection/custom\_filter

请求方式：GET

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 | 是否必须 |
| id | int | 自增主键 | N |
| uid | int | 用户的id | Y |
| book\_id | int | 收藏的书的id | N |

返回参数：

### 构件6：删除收藏夹中的内容

#### 功能描述

对于收藏夹中的东西，用户可能随时会想将某一本书移出收藏夹，该功能便可以帮助用户便捷地将待操作的书籍移出收藏夹

#### 对象模型

#### 接口规范

请求的URL：/bookStore/api/collection/{id}/

请求方式：DELETE

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 | 是否必须 |
| id | int | 在收藏夹中的序号 | Y |

返回参数：

### 构件7：个人中心

#### 功能描述

#### 对象模型

#### 接口规范

### 2.3.8 构件8：购物车

#### 2.3.8.1 功能描述

用户可查询自己的购物车中的内容

#### 2.3.8.2 对象模型

#### 2.3.8.3 接口规范

请求的URL：/bookStore/api/shoppingcarts/custom\_filter/

请求方式：POST

请求参数：

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名称 | 参数类型 | 参数描述 | 是否必须 |
| id | int | 该条记录在购物车中的序号 | N |
| user\_id | int | 用户的id | Y |
| book\_id | int | 购买的书籍的id | Y |
| amount | int | 购买的数量 | N |

返回参数：

1. [

2. {

3. “id”: int,

4. “user”: {

5. “uid”: int,

6. “uname”: string,

7. “pwdhash”: string,

8. “email”: string,

9. “tel”: string

10. },

11. “book”: {

12. “book\_id”: int,

13. “author”:{

14. “author\_id”: int,

15. “aname”: string

16. },

17. “publisher”:{

18. “pub\_id”: int,

19. “pname”: string

20. },

21. “category”:{

22. “category\_id”: int,

23. “category\_name”: string

24. },

25. “bname”: string,

26. “price”: double,

27. “pub\_year”: int,

28. “url”: string,

29. “isbn”: string,

30. “sales”: int,

31. “rate”: double

32. },

33. “amount”: int

34. }

35. ]

36.

## 构件合作模型

## 图形用户接口设计

## 数据库设计

### 对象永久存储策略

本项目中，采用MySQL数据库进行对象的永久存储。

### 数据库表设计

books：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| book\_id | INT | N |  | PK | 序号 |  |
| bname | VARCHAR(255) | N |  |  | 书名 |  |
| author\_id | INT | N |  | FK | 作者 |  |
| pub\_id | INT | N |  | FK | 出版社 |  |
| category\_id | INT | N |  | FK | 种类 |  |
| price | DECIMAL(10,2) | N |  |  | 价格 |  |
| pub\_year | INT | N |  |  | 出版年 |  |
| url | VARCHAR(255) | N |  |  | 网址 |  |
| isbn | CHAR(10) | N |  |  | isbn号 |  |
| sales | INT | Y | 0 |  | 销售量 |  |
| rate | DECIMAL(2,1) | N |  |  | 评分 |  |

* book\_id：
* 字段描述：书籍的id，用来方便标识每个书
* 字段格式：整数
* 取值范围：[0，100000]
* bname：
* 字段描述：书籍名称
* 字段格式：字符串
* 取值范围：无
* author\_id：
* 字段描述：作者id，用来标识每个书的作者，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* pub\_id：
* 字段描述：出版社id，用来标识每个书的出版社，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* category\_id：
* 字段描述：种类id，用来标识每个书所属的种类，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* price：
* 字段描述：该书在市面上对应的价格，以人民币为单位
* 字段格式：浮点数，保留两位小数
* 取值范围：[0.01，3000000.99]
* pub\_year：
* 字段描述：出版年份
* 字段格式：合法的4位数年份
* 取值范围：[1900，2023]
* url：
* 字段描述：该书的图片对应的网址
* 字段格式：https://pictures.abebooks.com/xxxxxx.jpg
* 取值范围：无
* isbn：
* 字段描述：该书对应的10位isbn号
* 字段格式：10位长的字符串，将前9位数字依序分别乘以10到2的数目，将其乘积相加，综合用11去除，若无余数则检查号码为0，否则，以11减去余数，所得差数即为检查好，若差数为10则以X表示
* 取值范围：无
* sales：
* 字段描述：该书的销售量
* 字段格式：整数
* 取值范围：[0，1000000]
* rate：

字段描述：该书的全网评分

字段格式：浮点数，保留1位小数

取值范围：[0.0，5.0]

authros：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| author\_id | INT | N |  | PK | 序号 |  |
| aname | VARCHAR(255) | N |  |  | 作者名 |  |

* author\_id：
* 字段描述：作者的id，用来方便标识每本书的作者
* 字段格式：整数
* 取值范围：[0，100000]
* aname：
* 字段描述：作者的姓名
* 字段格式：字符串
* 取值范围：无

publishers：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| publisher\_id | INT | N |  | PK | 序号 |  |
| pname | VARCHAR(255) | N |  |  | 出版社名 |  |

* publisher\_id：
* 字段描述：出版社的id，用来方便标识每本书的出版社
* 字段格式：整数
* 取值范围：[0，100000]
* pname：
* 字段描述：出版社的名称
* 字段格式：字符串
* 取值范围：无

category：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| category\_id | INT | N |  | PK | 序号 |  |
| category\_name | VARCHAR(255) | N |  |  | 类名 |  |

* category\_id：
* 字段描述：种类的id，用来方便标识每本书所属的种类
* 字段格式：整数
* 取值范围：[0，100000]
* category\_name：
* 字段描述：种类的名称
* 字段格式：字符串
* 取值范围：无

users：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| uid | INT | N |  | PK | 序号 |  |
| uname | VARCHAR(100) | N |  |  | 用户名 |  |
| pwdhash | VARCHAR(255) | N |  |  | 口令哈希 |  |
| email | VARCHAR(100) | Y |  |  | 邮箱 |  |
| tel | CHAR(11) | Y |  |  | 电话 |  |

* uid：
* 字段描述：用户id，用来方便标识每个用户
* 字段格式：整数
* 取值范围：[0，100000]
* uname：
* 字段描述：用户昵称
* 字段格式：字符串
* 取值范围：无
* pwdhash：
* 字段描述：用户登录密码的哈希值，哈希算法采用sha256
* 字段格式：长度为64位的16进制串（即字符出只包含0-9的数字和A-F的大写字母）
* 取值范围：无
* email：
* 字段描述：用户的邮箱
* 字段格式：符合邮箱格式的字符串，本系统的合法邮箱后缀包括：@163.com、@qq.com、@gmail.com、@mail.hk.com、@yahoo.co.id
* 取值范围：无
* tel：
* 字段描述：用户的电话
* 字段格式：符合中国电话号码格式的11位字符串
* 取值范围：无

shoppinghistorys：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| id | INT | N |  | PK | 标记 |  |
| uid | INT | N |  | FK | 用户 |  |
| book\_id | INT | N |  | FK | 书 |  |
| date | DATETIME | N |  |  | 日期 |  |
| amount | INT | N |  |  | 购买量 |  |

* id：
* 字段描述：用来标识每一条购物记录的主键，方便前后端的操作
* 字段格式：整数
* 取值范围：[0，100000]
* uid：
* 字段描述：用户id，用来标识该购物历史中买书用户，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* book\_id：
* 字段描述：书id，用来标识该购物历史中用户所买的书，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* date：
* 字段描述：购买日期
* 字段格式：符合mysql的datetime类型，为YYYY-MM-DD HH:MM:SS
* 取值范围：无
* amount：

字段格式：

shoppingcarts：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| uid | INT | N |  | PK | 用户 |  |
| book\_id | INT | N |  | PK | 书 |  |
| amount | INT | N |  |  | 量 | >0 |

* uid：
* 字段描述：用户id，用来标识该购物车所属的用户，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* book\_id：
* 字段描述：书id，用来标识该购物车中用户待购买的书，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* amount：
* 字段描述：购买量
* 字段格式：整数
* 取值范围：>0

collections：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 允许空 | 缺省值 | 主外键 | 描述 | 约束 |
| uid | INT | N |  | PK | 用户 |  |
| book\_id | INT | N |  | PK | 书 |  |

* uid：
* 字段描述：用户id，用来标识该收藏夹所属用户，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]
* book\_id：
* 字段描述：书id，用来表示该收藏夹中用户收藏的书籍，方便建立外键连接
* 字段格式：整数
* 取值范围：[0，100000]

### 存储过程

在数据库建立过程中，通过自行编写爬虫脚本将书籍数据信息存储到本地的csv文件中，并对数据进行处理，将处理好的数据录入MySQL中完成数据初始化；在后续的使用过程中，通过数据库管理员来对数据进行增删操作。

### 触发器设计

设计目的：为了保证用户在删除一条购物车信息后，自动将对应书籍添加到购买历史中。

设计方法：在shoppingcarts表中添加一个after delete的触发器shop，并将删除的数据中的uid和book\_id以及删除时的日期添加到表shoppinghistory中。

## 系统使用方法设计

1. git clone本项目至本地。
2. 启动MySQL服务器，创建数据库bookStore。
3. 进入前端项目bookStore\_vue，运行npm run dev，启动前端服务器。
4. 进入后端项目django，运行python3 manage.py runserver启动后端服务器。

## 公共函数库

### Python

* [Django](https://pypi.org/project/Django/4.2.1/)
* [Django-rest-framework](https://pypi.org/project/djangorestframework/)
* [coreapi](https://pypi.org/project/coreapi/)
* [pymysql](https://pypi.org/project/pymysql/)

### Vue

* axios@1.4.0
* element-plus@2.3.8
* vue@3.3.4

## 非功能性需求

### 安全性

本项目应满足的安全性包括：保密性、防泄露、访问控制

### 易用性

本项目应易于用户使用，即前端页面不可过于复杂

### 资源使用率

本项目的CPU占用率应小于等于50%，内存占用应小于等于50%

### 兼容性需求

系统应支持IOS、Android、Windows、MacOS操作系统；最多只有5%的系统实现需要具体到操作系统；替换关系数据库系统的平均时间不超过2小时，并且保证没有数据丢失。

## 辅助工具

* Navicate
* Drawio
* Github

# 系统详细设计

## 开发环境概述

### 版本控制工具

本项目中采用GIT作为版本控制工具。

### 编译环境

* 硬件平台：前端采用AMD Ryzen 7 5800H with Radeon Graphics及NVIDIA GeForce RTX 3060 Laptop GPU，后端及数据库采用MacBook Pro及Apple M1 Pro。
* 操作系统：前端采用Windows 11 Version 22H2，后端及数据库采用Sonoma 14.1.2。
* 开发工具与环境：
  + 开发工具：PyCharm 2023.2 (Professional Edition)
  + 环境包括：Node.js v21.3.0、npm@10.2.4、Python3.10.8、mysql Ver 8.0.33

### 源程序目录

前端程序位于项目的/forend文件夹中，目录结构如下：

后端程序位于项目的/backend/django文件夹中，目录结构如下：

├── backend\_django

   ├── \_\_init\_\_.py

   ├── \_\_pycache\_\_

   ├── asgi.py

   ├── settings.py

   ├── urls.py

   └── wsgi.py

├── bookStore

   ├── \_\_init\_\_.py

   ├── \_\_pycache\_\_

   ├── admin.py

   ├── apps.py

   ├── migrations

   ├── models.py

   ├── serializers.py

   ├── tests.py

   ├── urls.py

   └── views.py

└── manage.py

### 文档存储目录

各文档存储目录位于/docs文件夹中，目录结构如下：

├── foreEnd\_docs

   ├── foreEnd\_usage.md

├── sql\_docs

   ├── 数据库系统及安全-实验手册-Lab3.docx

   ├── 概念设计.pdf

   ├── 物理设计.xlsx

   ├── 系统设计.docx

   ├── 逻辑设计.pdf

## 构件详细设计

### 构件1：用户登录

#### 源程序列表

#### 设计思想

#### 实现描述

### 构件2：用户注册

#### 源程序列表

#### 设计思想

#### 实现描述

### 构件3：高级搜索

#### 源程序列表

#### 设计思想

#### 实现描述

### 构件4：书籍浏览

#### 源程序列表

#### 设计思想

#### 实现描述

### 构件5：获取收藏夹

#### 源程序列表

#### 设计思想

#### 实现描述

### 构件6：删除收藏夹中的内容

#### 源程序列表

#### 设计思想

#### 实现描述

### 构件7：个人中心

#### 源程序列表

#### 设计思想

#### 3.2.7.3 实现描述

### 构件8：购物车

#### 源程序列表

#### 设计思想

#### 3.2.8.3 实现描述