武汉纺织大学

Web应用开发课程设计

**图书管理系统**

**学 院： 数学与计算机学院**

**班 级： 物联网11802**

**姓 名： 邹正**

**学 号： 1804241014**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月15日**

目录

[1 需求分析 3](#_Toc22334)

[2 系统设计要求 3](#_Toc32254)

[3 系统实现 5](#_Toc8346)

[3.1 项目结构 5](#_Toc927)

[三层架构 5](#_Toc2996)

[基于B/S的三层mvc架构 5](#_Toc27380)

[类与页面的设计 7](#_Toc1515)

[前端设计 8](#_Toc27988)

[后端设计 8](#_Toc9949)

[3.2数据库具体实现 10](#_Toc8420)

[3.2.1 创建数据库 10](#_Toc7449)

[3.2.2 创建数据模型对应数据表 10](#_Toc3852)

[3.3 DAL层设计： 12](#_Toc21809)

[3.3.2 BookMapper：对书籍的增删改查操作 13](#_Toc32693)

[3.3.4 userMapper：拥有用户权限能进行的各种操作 15](#_Toc22149)

[4 系统测试 18](#_Toc10098)

[4.1登录页面 18](#_Toc27455)

[4.2用户界面 18](#_Toc6150)

[用户登录首页 18](#_Toc5465)

[用户借书记录查看 19](#_Toc3545)

[查询书籍页面 20](#_Toc8354)

[个人信息以及修改页面 21](#_Toc19037)

[4.3管理员界面 21](#_Toc25116)

[管理员首页 22](#_Toc27261)

[添加书籍页面 22](#_Toc30512)

[新建书籍类别页面 22](#_Toc24961)

[通过类别查询书籍 23](#_Toc11139)

[查询用户页面 23](#_Toc17420)

[新增用户界面 24](#_Toc8820)

[修改信息页面 24](#_Toc11300)

[5 总结 25](#_Toc2442)

# 1 需求分析

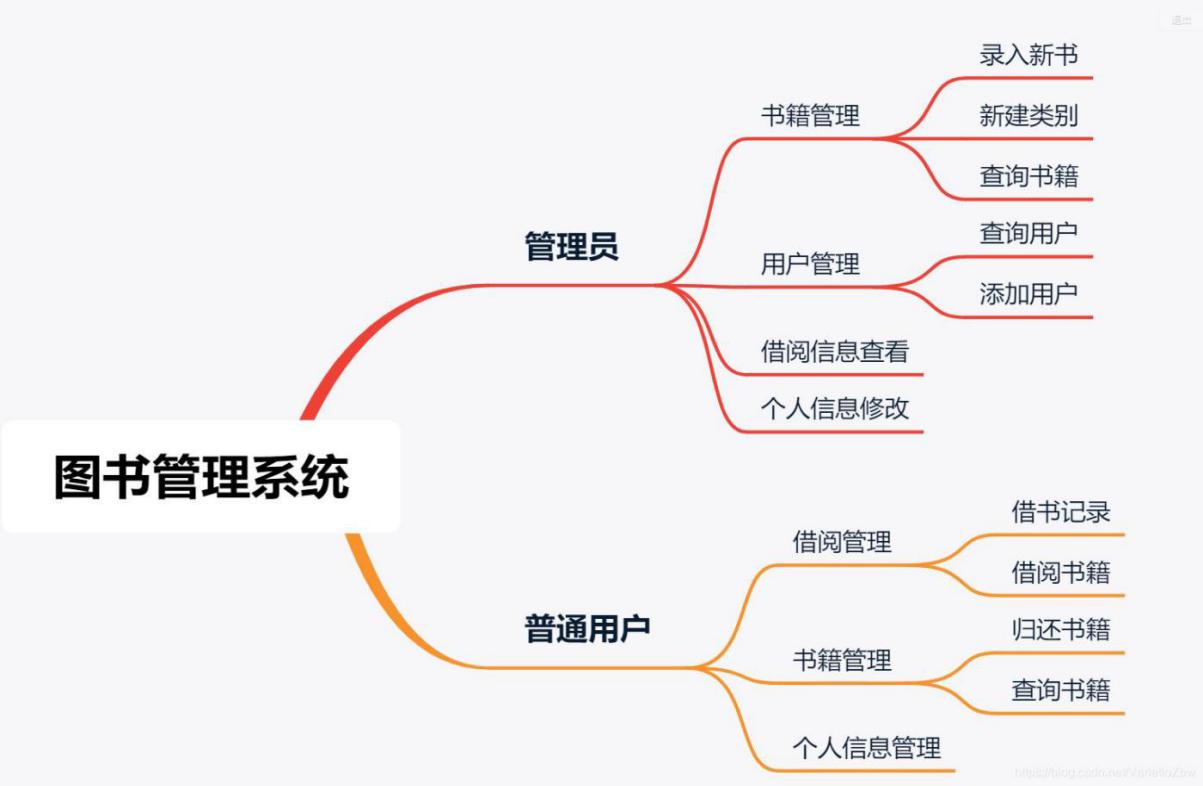
图书馆管理系统用于计算机管理图书馆里的书籍，方便学生进行更方便快捷的结束程序。实现简单的图书馆系统的读者功能：登陆，查询图书，借阅图书，归还图书，查询借阅记录，修改密码。使信息管理更具体、更明朗、更电子化，让人一目了然,给人们带来无穷的快捷。

# **2 系统设计要求**

系统功能需求分析描述如下:

1. 系统主菜单:进入系统和退出系统
2. 功能菜单:显示可以进行的操作

(3）查询主菜单:根据图书编号、书名、作者进行查询，并可显示所有图书信息



# **3 系统实现**

## 3.1 项目结构

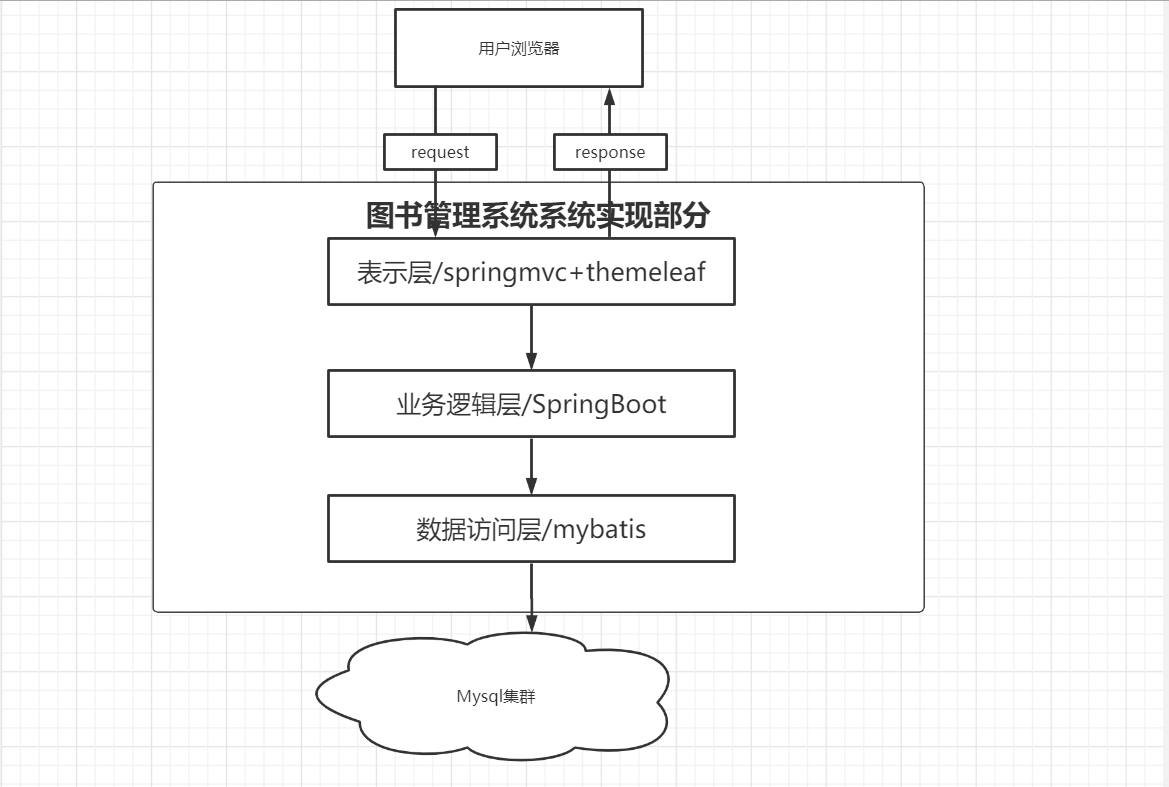
## 三层架构

### 基于B/S的三层mvc架构

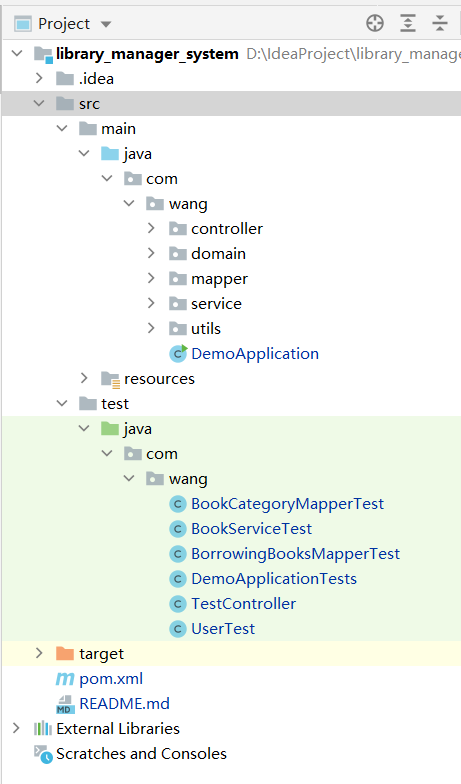
三层架构是指：视图层View、服务层Service、持久层Dao，分别完成不同的功能。

View层：用于接收用户提交请求的代码在这里编写。

Service层：系统的业务逻辑主要在这里编写。

Dao层：直接操作数据库的代码在这里编写。

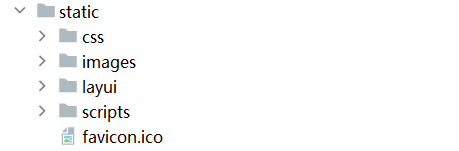
如图，在IDEA中创建maven项目目录结构



## 类与页面的设计

前端设计：

#### 静态资源

静态资源都放在static包下：如图片，样式，图标等

#### 页面资源

Admin：存放管理员页面

User：存放用户页面

Commom：存放公共页面资源



### 后端设计

#### **视图层**

#### **Model层**

Model层设计主要依赖于orm思想

ORM Object Relational Mapping 对象关系映射，是为了解决面向对象的类，与关系数据库的表之间，存在的不匹配的现象，通过使用描述对象和关系之间映射的元数据，在程序中的类对象，与关系数据库的表之间建立持久的关系，用于在程序中描述数据库表。本质上就是将数据从一种形式转换到另外一种形式。

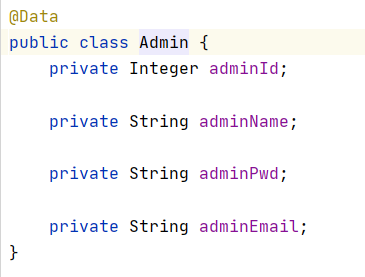
其中包括对象实体类和传输对象类还有条件对象

实体对象：与数据库表先对应的类

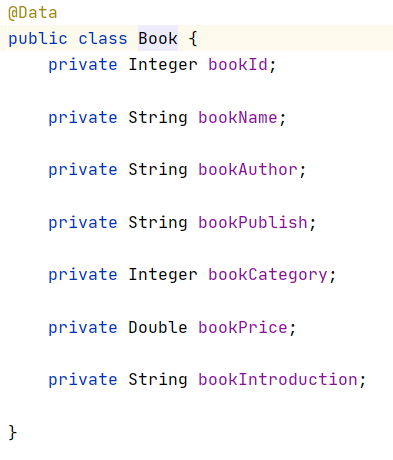
传输对象：用于在mvc三层间数据的传输

条件对象：通过对实体对象进行条件的包装，使得更灵活的进行数据库的增删改查操作，后缀example，如以下各实体类对应表

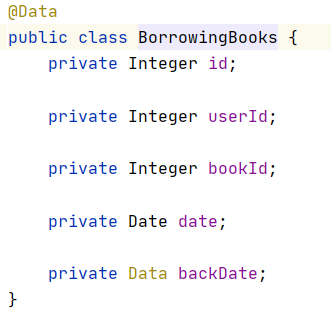
Admin表



Book表



Brrowing表



其核心就是sql字段类型和java字段类型之间的转换

**3.2数据库具体实现**

### 

### 3.2.1 创建数据库

CREATE DATABASE IF NOT EXISTS library-manager-system default charset utf8mb4 COLLATE utf8mb4\_general\_ci;

### 3.2.2 创建数据模型对应数据表

Dept表

CREATE TABLE `dept` (

`dept\_id` int NOT NULL AUTO\_INCREMENT,

`dept\_name` varchar(20) DEFAULT NULL,

PRIMARY KEY (`dept\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=5 DEFAULT CHARSET=utf8;

Admin表

CREATE TABLE `admin` (

`admin\_id` int NOT NULL AUTO\_INCREMENT,

`admin\_name` varchar(20) DEFAULT NULL,

`admin\_pwd` varchar(20) DEFAULT NULL,

`admin\_email` varchar(20) DEFAULT NULL,

PRIMARY KEY (`admin\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=2 DEFAULT CHARSET=utf8;

book\_category表

CREATE TABLE `book\_category` (

`category\_id` int NOT NULL AUTO\_INCREMENT,

`category\_name` varchar(20) DEFAULT NULL,

PRIMARY KEY (`category\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=50 DEFAULT CHARSET=utf8;

User表

CREATE TABLE `user` (

`user\_id` int NOT NULL AUTO\_INCREMENT,

`user\_name` varchar(20) DEFAULT NULL,

`user\_pwd` varchar(20) DEFAULT NULL,

`user\_email` varchar(30) DEFAULT NULL,

`user\_dept` int DEFAULT NULL,

PRIMARY KEY (`user\_id`),

KEY `user\_dept\_key` (`user\_dept`),

CONSTRAINT `user\_dept\_key` FOREIGN KEY (`user\_dept`) REFERENCES `dept` (`dept\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=27 DEFAULT CHARSET=utf8;

Book表

CREATE TABLE `book` (

`book\_id` int NOT NULL AUTO\_INCREMENT,

`book\_name` varchar(20) NOT NULL,

`book\_author` varchar(20) DEFAULT NULL,

`book\_publish` varchar(20) DEFAULT NULL,

`book\_category` int DEFAULT NULL,

`book\_price` double DEFAULT NULL,

`book\_introduction` varchar(100) DEFAULT NULL,

PRIMARY KEY (`book\_id`),

KEY `book\_category` (`book\_category`) USING BTREE,

CONSTRAINT `book\_ibfk\_1` FOREIGN KEY (`book\_category`) REFERENCES `book\_category` (`category\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=74 DEFAULT CHARSET=utf8;

borrowingbooks表

CREATE TABLE `borrowingbooks` (

`id` int NOT NULL AUTO\_INCREMENT,

`user\_id` int DEFAULT NULL,

`book\_id` int DEFAULT NULL,

`date` date DEFAULT NULL,

`back\_data` date DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `user\_id` (`user\_id`) USING BTREE,

KEY `book\_id` (`book\_id`) USING BTREE,

CONSTRAINT `borrowingbooks\_ibfk\_1` FOREIGN KEY (`book\_id`) REFERENCES `book` (`book\_id`),

CONSTRAINT `borrowingbooks\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=67 DEFAULT CHARSET=utf8;

**3.3**DAL层设计：

DAL层的类主要实现对应数据表的插、删、改、查等操作，以及存储过程的调用系

统实现，其中每一个mapper接口都对应一个实现他的mybatis映射文件。

3.3.1 AdminMapper

能过通过条件对象对数据表进行各种参数的增删改查（包括用户和书籍以及借阅情况）

package com.wang.mapper;  
import com.wang.domain.Admin;  
import com.wang.domain.AdminExample;  
import org.apache.ibatis.annotations.Param;  
import java.util.List;  
  
public interface AdminMapper {  
 long countByExample(AdminExample example);  
  
 int deleteByExample(AdminExample example);  
  
 int deleteByPrimaryKey(Integer adminId);  
  
 int insert(Admin record);  
  
 int insertSelective(Admin record);  
  
 List<Admin> selectByExample(AdminExample example);  
  
 Admin selectByPrimaryKey(Integer adminId);  
  
 int updateByExampleSelective(@Param("record") Admin record, @Param("example") AdminExample example);  
 int updateByExample(@Param("record") Admin record, @Param("example") AdminExample example);  
 int updateByPrimaryKeySelective(Admin record);  
 int updateByPrimaryKey(Admin record);  
}

**3.3.2** BookMapper：对书籍的增删改查操作

package com.wang.mapper;  
  
import com.wang.domain.Book;  
import com.wang.domain.BookExample;  
import org.apache.ibatis.annotations.Param;  
  
import java.util.List;  
  
public interface BookMapper {  
 long countByExample(BookExample example);  
  
 int deleteByExample(BookExample example);  
  
 int deleteByPrimaryKey(Integer bookId);  
  
 int insert(Book record);  
  
 int insertSelective(Book record);  
  
 List<Book> selectByExample(BookExample example);  
  
 List<Book> selectBooksByPartInfo(String bookPartInfo);  
  
 Book selectByPrimaryKey(Integer bookId);  
  
 int updateByExampleSelective(@Param("record") Book record, @Param("example") BookExample example);  
  
 int updateByExample(@Param("record") Book record, @Param("example") BookExample example);  
  
 int updateByPrimaryKeySelective(Book record);  
  
 int updateByPrimaryKey(Book record);  
  
 //按书籍种类分页查找  
 List<Book> selectByCategoryId(@Param("categoryId") int categoryId, @Param("currIndex") int currIndex, @Param("pageSize") int PageSize);  
  
 //查找某一类别书籍的总数  
 int selectBookCountByCategoryId(@Param("categoryId") int categoryId);  
}

**3.3.3** BorrowingBooksMapper：对借阅记录的增删改查的各种操作

package com.wang.mapper;  
  
import com.wang.domain.BorrowingBooks;  
import com.wang.domain.BorrowingBooksExample;  
import org.apache.ibatis.annotations.Param;  
  
import java.util.List;  
  
public interface BorrowingBooksMapper {  
 long countByExample(BorrowingBooksExample example);  
  
 int deleteByExample(BorrowingBooksExample example);  
  
 int deleteByPrimaryKey(Integer id);  
  
 int insert(BorrowingBooks record);  
  
 int insertSelective(BorrowingBooks record);  
  
 List<BorrowingBooks> selectByExample(BorrowingBooksExample example);  
  
 BorrowingBooks selectByPrimaryKey(Integer id);  
  
 int updateByExampleSelective(@Param("record") BorrowingBooks record, @Param("example") BorrowingBooksExample example);  
  
 int updateByExample(@Param("record") BorrowingBooks record, @Param("example") BorrowingBooksExample example);  
  
 int updateByPrimaryKeySelective(BorrowingBooks record);  
  
 int updateByPrimaryKey(BorrowingBooks record);  
  
 //查询userId用户的所有借书记录  
 List<BorrowingBooks> selectAllBorrowRecord(@Param("userId") int userId);  
  
 //查询该userId用户的总借书记录数目  
 int selectAllRecordCount(@Param("userId") int userId);  
  
 // 分页查询所有记录  
 List<BorrowingBooks> selectAllByPage(@Param("currIndex") int currIndex, @Param("pageSize") int pageSize);  
  
 //查询所有记录总数  
 int selectAll();  
}

## **3.3.4** userMapper：拥有用户权限能进行的各种操作

package com.wang.mapper;  
  
import com.wang.domain.User;  
import com.wang.domain.UserExample;  
import org.apache.ibatis.annotations.Param;  
  
import java.util.List;  
  
public interface UserMapper {  
 long countByExample(UserExample example);  
  
 int deleteByExample(UserExample example);  
  
 int deleteByPrimaryKey(Integer userId);  
  
 int insert(User record);  
  
 int insertSelective(User record);  
  
 List<User> selectByExample(UserExample example);  
  
 User selectByPrimaryKey(Integer userId);  
  
 int updateByExampleSelective(@Param("record") User record, @Param("example") UserExample example);  
  
 int updateByExample(@Param("record") User record, @Param("example") UserExample example);  
  
 int updateByPrimaryKeySelective(User record);  
  
 int updateByPrimaryKey(User record);  
  
 //分页查询  
 List<User> selectByPageNum(@Param("currIndex") int currIndex, @Param("pageSize") int pageSize);  
  
 //查询总数  
 int selectUserCount();  
}

其他的mapper类（DAO类）也相似。

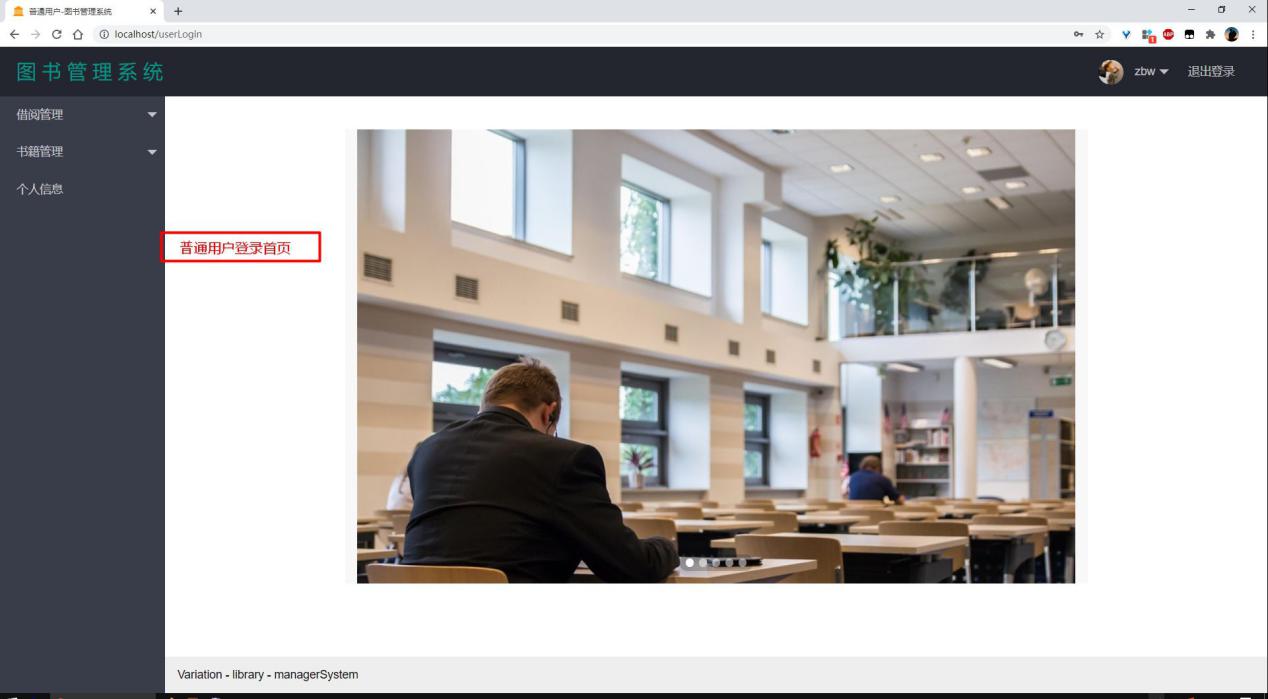
# **4 系统测试**

## 4.1登录页面

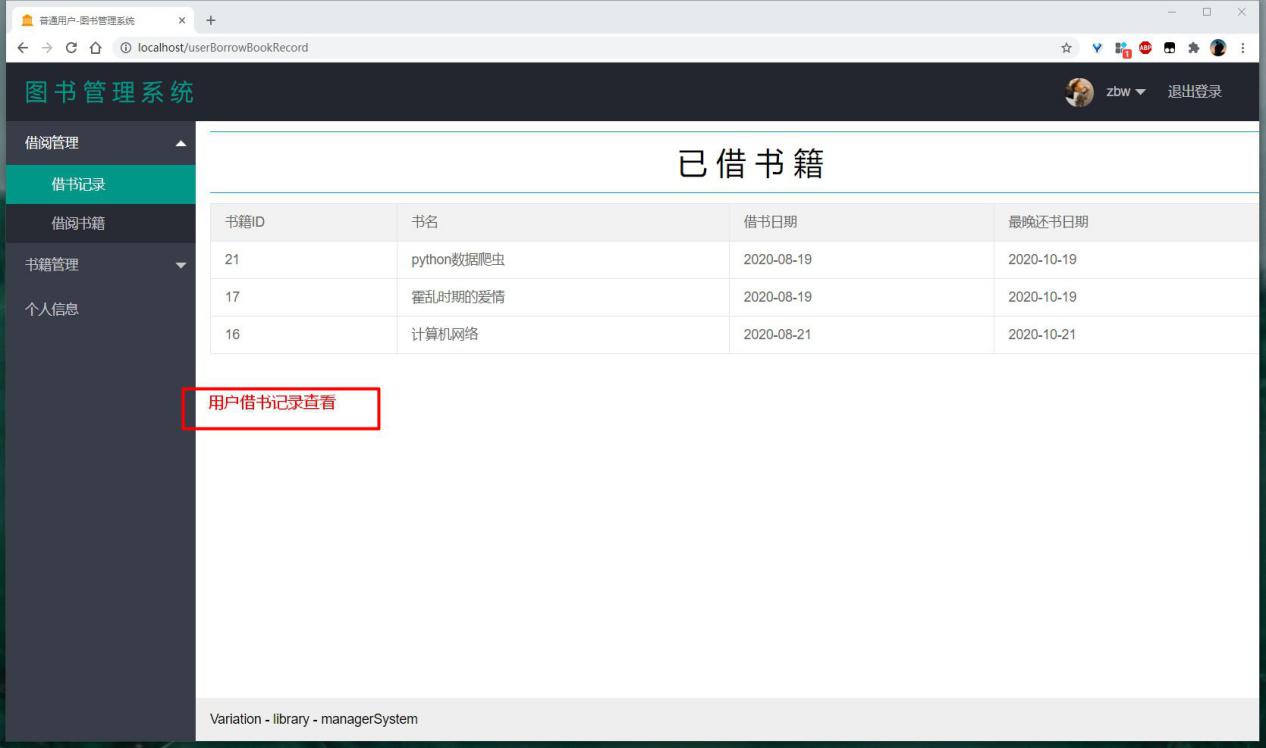


**4.2用户界面**

用户登录首页



用户借书记录查看



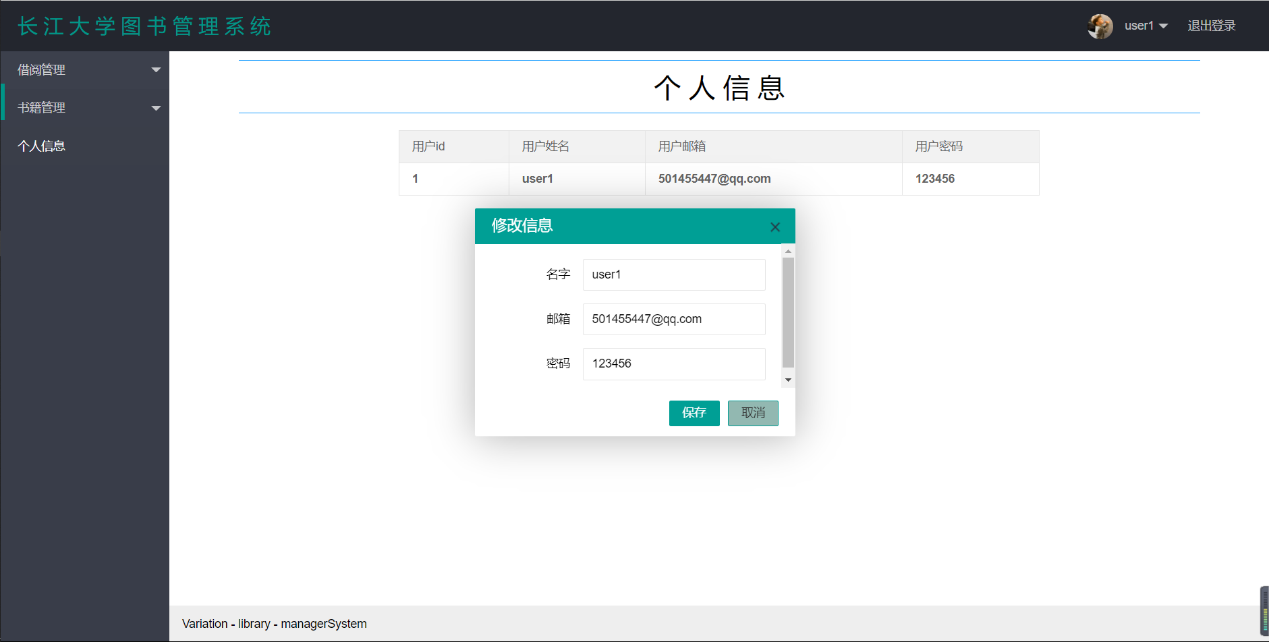
用户借书页面

#### 用户还书页面

查询书籍页面

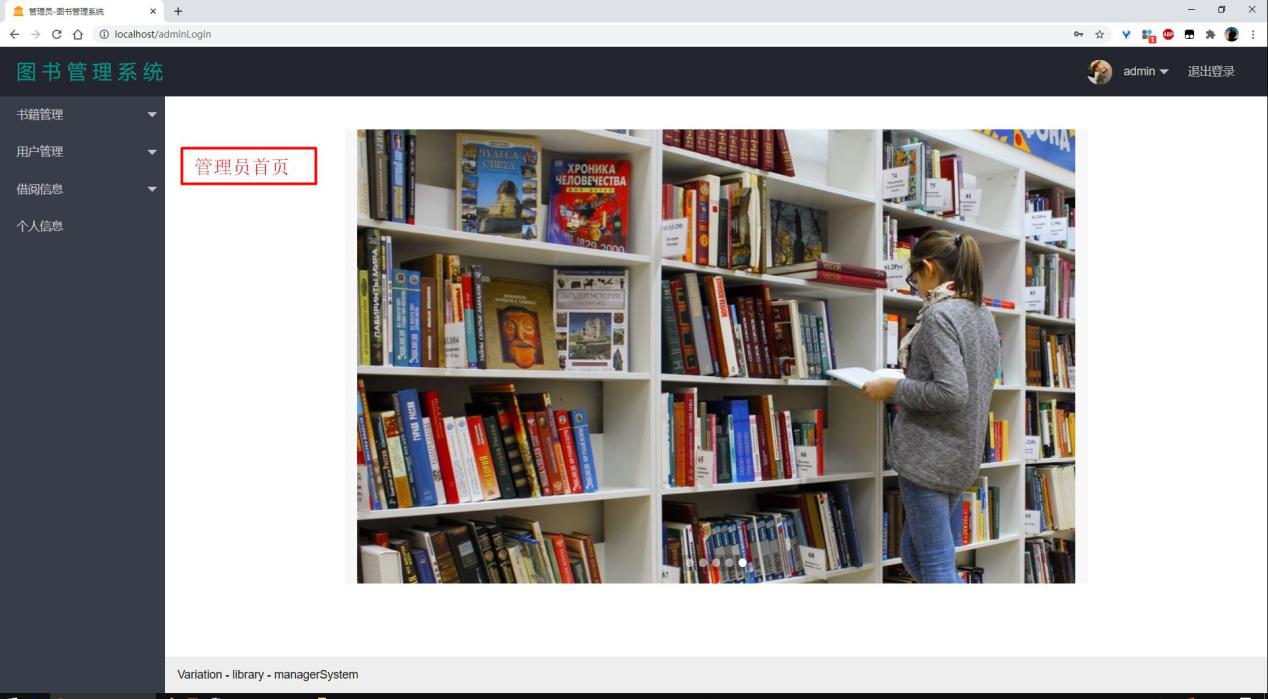


个人信息以及修改页面

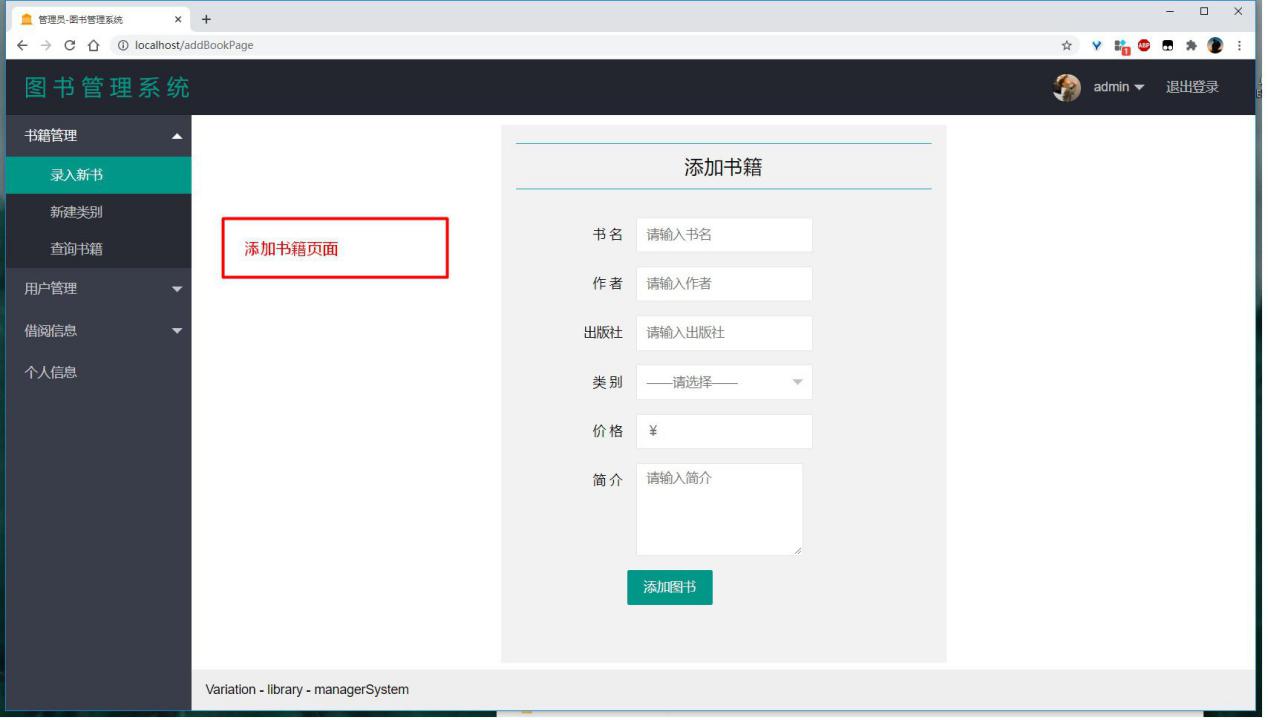


**4.3管理员界面**

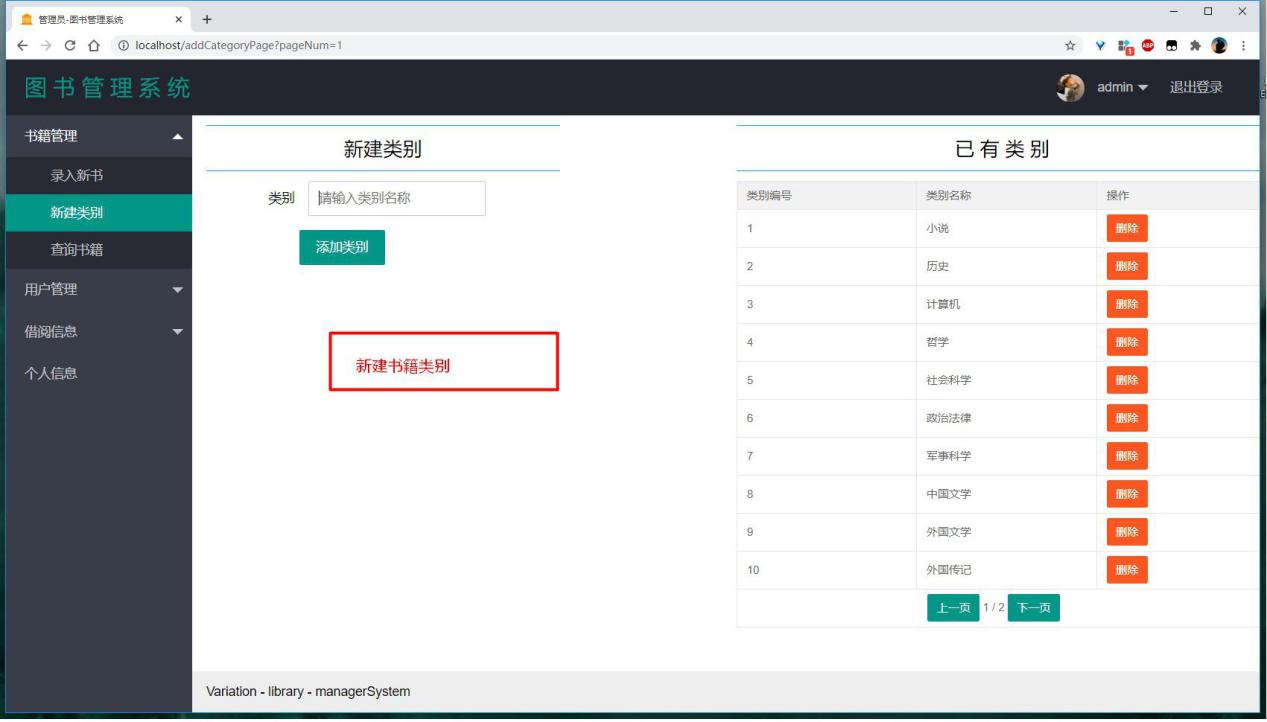
**管理员首页**

****

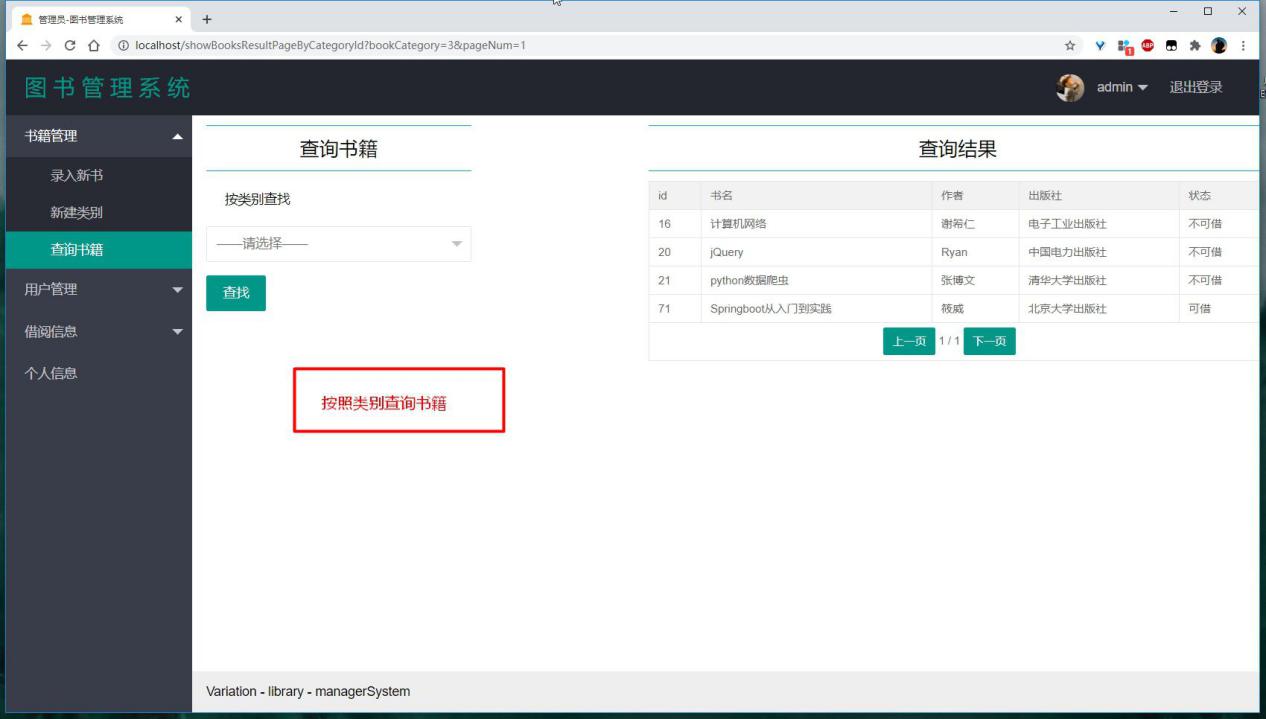
**添加书籍页面**

****

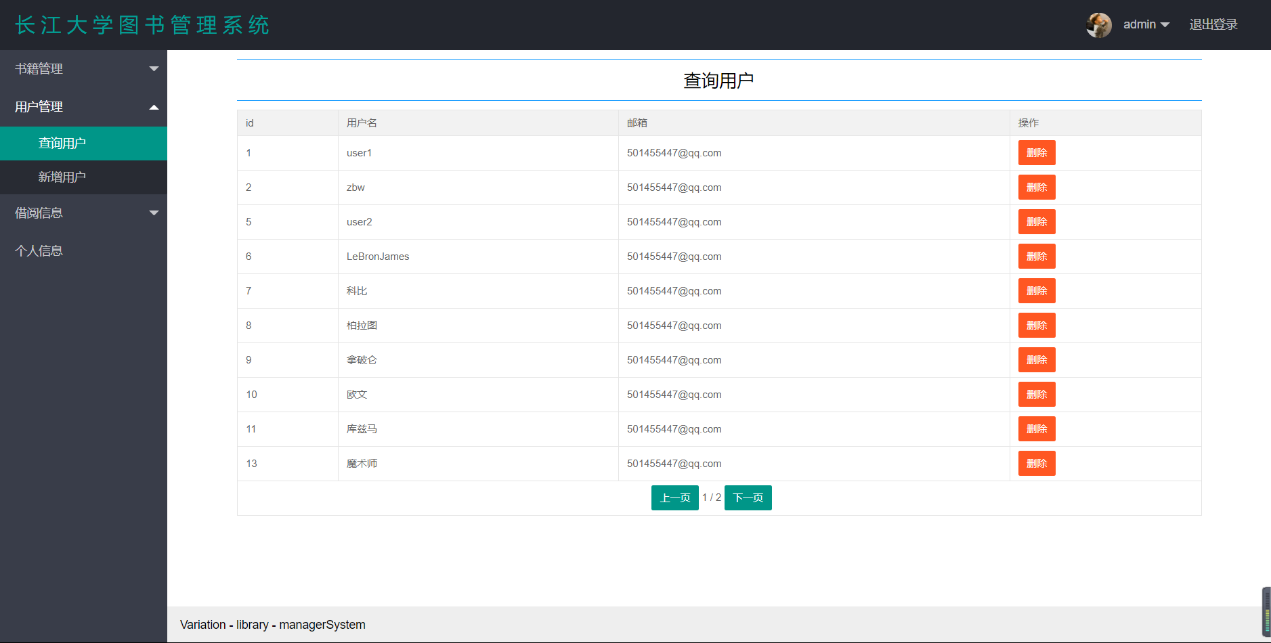
**新建书籍类别页面**

****

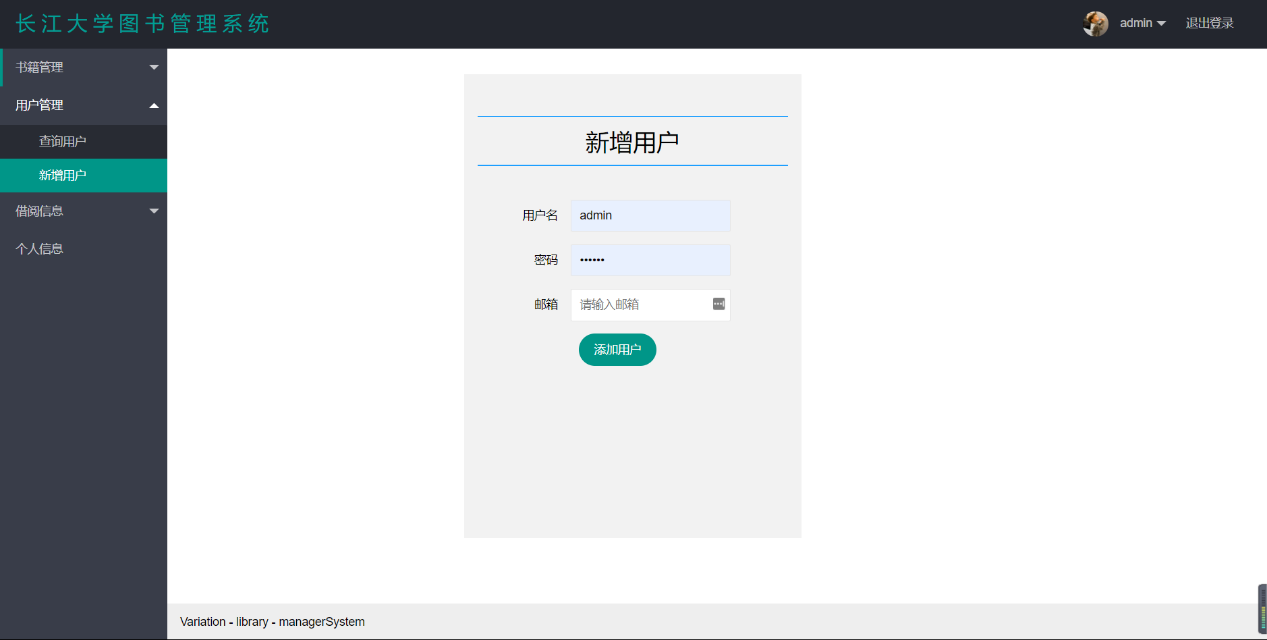
**通过类别查询书籍**

****

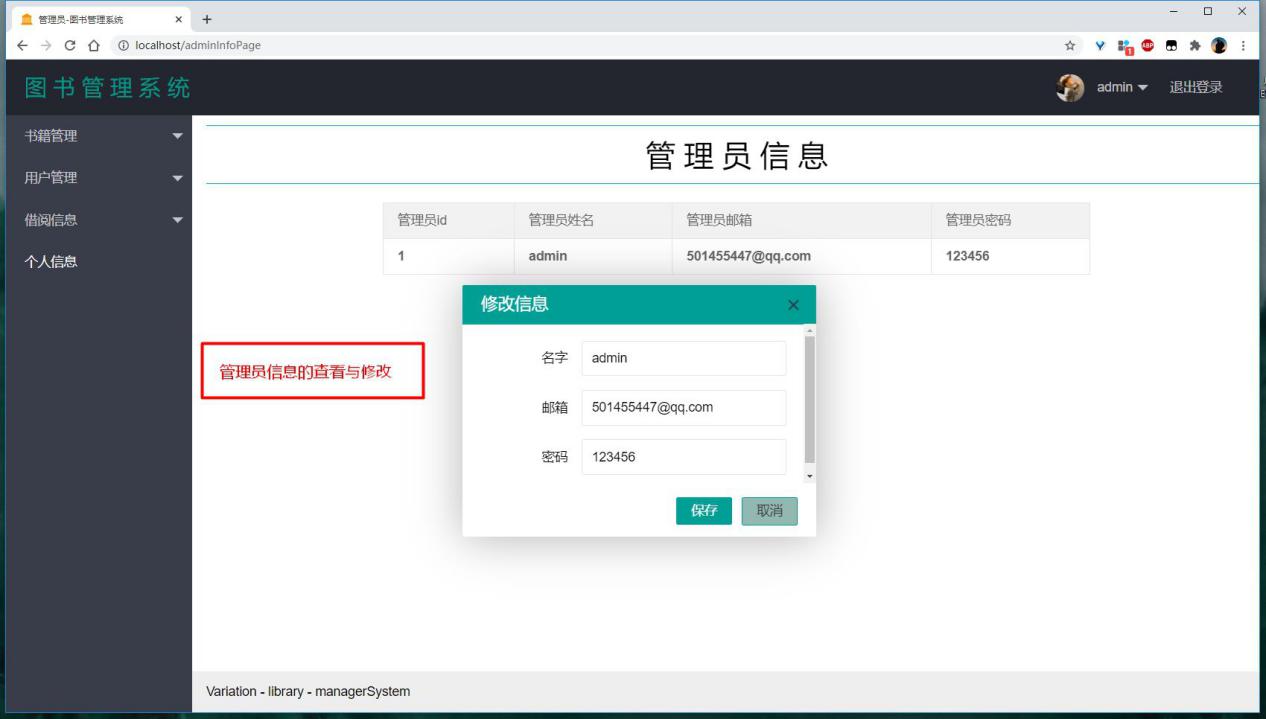
**查询用户页面**



**新增用户界面**



**修改信息页面**

****

# **5 总结**

在本次数据库课程设计中遇到了不少困难，我一般采取以下措施来解决问题：网上查找资料；浏览老师提供的课件；和同学讨论；在图书馆查找相关书籍资料。遇到困难时的心态要平和，冷静地去查找问题并解决问题。

课程设计中程序比较复杂，在调试时应该仔细，在程序调试时，注意指针，将不必要的命令去除。除了课堂外，课程设计是最能学到东西的，最考验人的。在做课程设计的这段时间，时刻都感到自己学的知识有多么的贫瘠。经过这次课程设计，也让我对java这门语言有了更深的认识，操作能力有了提高。要想写出的代码能运行，需要耐心细心，以及相当的毅力。通常都是需要多次的修改才能达到所要的运行结果。只有经过多次编辑，多次编译，再多次运行，才能编写出更好的程序。