



**专项设计报告**

**基于Web的图书管理系统**

**学 院 计算机学院**

**专 业 网络工程**

**年级班别 1604**

**学 号 3216005086**

**学生姓名 张微**

**指导教师 张凡龙**

**2019年 10月**

目录

[1 需求分析](#_Toc29232_WPSOffice_Level1) [5](#_Toc29232_WPSOffice_Level1)

[1.1需求定义](#_Toc29232_WPSOffice_Level2) [5](#_Toc29232_WPSOffice_Level2)

[1.1.1图书管理](#_Toc29232_WPSOffice_Level3) [5](#_Toc29232_WPSOffice_Level3)

[1.1.2 读者管理](#_Toc27987_WPSOffice_Level3) [5](#_Toc27987_WPSOffice_Level3)

[1.1.3管理员管理](#_Toc12218_WPSOffice_Level3) [5](#_Toc12218_WPSOffice_Level3)

[1.1.4借还管理](#_Toc26112_WPSOffice_Level3) [5](#_Toc26112_WPSOffice_Level3)

[1.1.5罚单管理](#_Toc16364_WPSOffice_Level3) [6](#_Toc16364_WPSOffice_Level3)

[1.1.6还书信息](#_Toc7212_WPSOffice_Level3) [6](#_Toc7212_WPSOffice_Level3)

[1.2 功能需求](#_Toc7212_WPSOffice_Level2) [6](#_Toc7212_WPSOffice_Level2)

[2 概念结构设计](#_Toc7212_WPSOffice_Level1) [7](#_Toc7212_WPSOffice_Level1)

[2.1读者信息模块](#_Toc13083_WPSOffice_Level2) [7](#_Toc13083_WPSOffice_Level2)

[2.2 图书信息模块](#_Toc3513_WPSOffice_Level2) [8](#_Toc3513_WPSOffice_Level2)

[2.3 管理员信息模块](#_Toc3659_WPSOffice_Level2) [8](#_Toc3659_WPSOffice_Level2)

[2.4 完整的E-R图](#_Toc20290_WPSOffice_Level2) [8](#_Toc20290_WPSOffice_Level2)

[3 逻辑结构设计](#_Toc13083_WPSOffice_Level1) [9](#_Toc13083_WPSOffice_Level1)

[3.1数据库逻辑设计](#_Toc10378_WPSOffice_Level2) [9](#_Toc10378_WPSOffice_Level2)

[3.1.1 借阅关系](#_Toc13083_WPSOffice_Level3) [9](#_Toc13083_WPSOffice_Level3)

[3.2数据库逻辑结构](#_Toc23449_WPSOffice_Level2) [9](#_Toc23449_WPSOffice_Level2)

[3.2.1 图书信息表](#_Toc3513_WPSOffice_Level3) [10](#_Toc3513_WPSOffice_Level3)

[4 物理设计](#_Toc3513_WPSOffice_Level1) [10](#_Toc3513_WPSOffice_Level1)

[4.1 建立图书馆数据库（library）](#_Toc2921_WPSOffice_Level2) [10](#_Toc2921_WPSOffice_Level2)

[4.2 建立图书表（book）](#_Toc20462_WPSOffice_Level2) [10](#_Toc20462_WPSOffice_Level2)

[4.3 建立读者表（reader）](#_Toc14774_WPSOffice_Level2) [11](#_Toc14774_WPSOffice_Level2)

[4.4 建立借阅表（borrow）](#_Toc31944_WPSOffice_Level2) [11](#_Toc31944_WPSOffice_Level2)

[4.5 建立管理员表（administartors）](#_Toc32511_WPSOffice_Level2) [12](#_Toc32511_WPSOffice_Level2)

[4.6 建立还书信息表（return\_table）](#_Toc28672_WPSOffice_Level2) [12](#_Toc28672_WPSOffice_Level2)

[4.7建立罚单信息表（ticket）](#_Toc26681_WPSOffice_Level2) [12](#_Toc26681_WPSOffice_Level2)

[5 实施过程](#_Toc3659_WPSOffice_Level1) [13](#_Toc3659_WPSOffice_Level1)

[5.1装载数据过程](#_Toc18252_WPSOffice_Level2) [13](#_Toc18252_WPSOffice_Level2)

[5.1.1 用DML语句对book表加入数据](#_Toc3659_WPSOffice_Level3) [13](#_Toc3659_WPSOffice_Level3)

[5.1.2 用DML语句对reader表加入数据](#_Toc20290_WPSOffice_Level3) [13](#_Toc20290_WPSOffice_Level3)

[5.1.3 用DML语句对administartors表加入数据](#_Toc10378_WPSOffice_Level3) [14](#_Toc10378_WPSOffice_Level3)

[5.2设计视图](#_Toc5771_WPSOffice_Level2) [14](#_Toc5771_WPSOffice_Level2)

[5.2.1 在reader,borrow和book上创建借书者的全面信息视图reader\_borrow](#_Toc23449_WPSOffice_Level3) [14](#_Toc23449_WPSOffice_Level3)

[5.2.3 创建个人所有借书归还记录视图 reader\_borrow\_return](#_Toc2921_WPSOffice_Level3) [14](#_Toc2921_WPSOffice_Level3)

[5.3 存储过程设计](#_Toc22615_WPSOffice_Level2) [15](#_Toc22615_WPSOffice_Level2)

[5.3.1 产生罚单proc\_gen\_ticket](#_Toc20462_WPSOffice_Level3) [15](#_Toc20462_WPSOffice_Level3)

[5.3.2 借书过程实现](#_Toc14774_WPSOffice_Level3) [15](#_Toc14774_WPSOffice_Level3)

[5.3.3 还书存储过程proc\_return](#_Toc31944_WPSOffice_Level3) [16](#_Toc31944_WPSOffice_Level3)

[5.3.4 交罚单存储过程proc\_payoff](#_Toc32511_WPSOffice_Level3) [18](#_Toc32511_WPSOffice_Level3)

[5.3.5 删除图书存储过程proc\_deleteBook](#_Toc28672_WPSOffice_Level3) [18](#_Toc28672_WPSOffice_Level3)

[5.4触发器设计](#_Toc25999_WPSOffice_Level2) [19](#_Toc25999_WPSOffice_Level2)

[5.4.1 设计触发器borrow\_after\_insert](#_Toc26681_WPSOffice_Level3) [19](#_Toc26681_WPSOffice_Level3)

[5.4.2 设计触发器return\_after\_insert](#_Toc18252_WPSOffice_Level3) [19](#_Toc18252_WPSOffice_Level3)

[5.4.3 定义定时器（事件）eventJob](#_Toc5771_WPSOffice_Level3) [19](#_Toc5771_WPSOffice_Level3)

[6 系统实现](#_Toc20290_WPSOffice_Level1) [20](#_Toc20290_WPSOffice_Level1)

[6.1 搭建前端框架](#_Toc25216_WPSOffice_Level2) [20](#_Toc25216_WPSOffice_Level2)

[6.1.1 搭建 vue-cli 脚手架](#_Toc22615_WPSOffice_Level3) [20](#_Toc22615_WPSOffice_Level3)

[6.1.2 引入 axious、element库](#_Toc25999_WPSOffice_Level3) [20](#_Toc25999_WPSOffice_Level3)

[6.1.3 上传到github仓库](#_Toc25216_WPSOffice_Level3) [21](#_Toc25216_WPSOffice_Level3)

[6.2 使用 NodeJs 建立本地服务器](#_Toc26765_WPSOffice_Level2) [21](#_Toc26765_WPSOffice_Level2)

[6.3 图书管理-页面](#_Toc24505_WPSOffice_Level2) [22](#_Toc24505_WPSOffice_Level2)

[6.3.1 功能介绍](#_Toc26765_WPSOffice_Level3) [22](#_Toc26765_WPSOffice_Level3)

[6.3.2 接口设计](#_Toc24505_WPSOffice_Level3) [23](#_Toc24505_WPSOffice_Level3)

[6.4 读者管理-页面](#_Toc7790_WPSOffice_Level2) [24](#_Toc7790_WPSOffice_Level2)

[6.4.1 功能介绍](#_Toc7790_WPSOffice_Level3) [24](#_Toc7790_WPSOffice_Level3)

[6.4.2 接口设计](#_Toc29972_WPSOffice_Level3) [26](#_Toc29972_WPSOffice_Level3)

[6.5 管理员管理-页面](#_Toc29972_WPSOffice_Level2) [28](#_Toc29972_WPSOffice_Level2)

[6.5.1 功能介绍](#_Toc31362_WPSOffice_Level3) [28](#_Toc31362_WPSOffice_Level3)

[6.5.2 接口设计](#_Toc18690_WPSOffice_Level3) [29](#_Toc18690_WPSOffice_Level3)

[6.6 借还管理-页面](#_Toc31362_WPSOffice_Level2) [30](#_Toc31362_WPSOffice_Level2)

[6.6.1 功能介绍](#_Toc4181_WPSOffice_Level3) [30](#_Toc4181_WPSOffice_Level3)

[6.6.2 接口设计](#_Toc29781_WPSOffice_Level3) [32](#_Toc29781_WPSOffice_Level3)

[6.7 罚单管理-页面](#_Toc18690_WPSOffice_Level2) [33](#_Toc18690_WPSOffice_Level2)

[6.7.1 功能介绍](#_Toc25366_WPSOffice_Level3) [33](#_Toc25366_WPSOffice_Level3)

[6.7.2 接口设计](#_Toc11393_WPSOffice_Level3) [34](#_Toc11393_WPSOffice_Level3)

[6.8 还书信息-页面](#_Toc4181_WPSOffice_Level2) [34](#_Toc4181_WPSOffice_Level2)

[6.8.1 功能介绍](#_Toc7528_WPSOffice_Level3) [35](#_Toc7528_WPSOffice_Level3)

[6.8.2 接口设计](#_Toc30895_WPSOffice_Level3) [35](#_Toc30895_WPSOffice_Level3)

**1 需求分析**

本图书管理系统主要实现对图书馆的管理：图书、读者、管理员、借阅。由此，结构可分为：图书管理、读者管理、管理员管理、借还管理、罚单管理、还书信息。

**1.1需求定义**

1.1.1图书管理

可对图书信息进行浏览、编辑（包括对图书的修改、入库、删除）

1).浏览图书信息

2).编辑图书信息：修改、入库、删除

1.1.2 读者管理

可对读者信息进行浏览、编辑（包括对读者的修改、入库、删除）

1).浏览读者信息

2).编辑读者信息：修改、添加、删除

1.1.3管理员管理

可对管理员信息进行浏览、编辑（包括对管理员的修改、入库、删除）

1).浏览管理员信息

2).编辑管理员信息：修改、添加、删除

1.1.4借还管理

可对借书信息进行浏览；进行借书操作，每次只能进行一次操作，也可对新增借书进行删除；进行还书操作，如逾期欠费有提示，需要进行缴费操作，才能还书

1).浏览图书借书信息

2).借书（每次只能操作一次）-> 可对新增借书的删除

4).还书（逾期欠费需缴费才能还书）

1.1.5罚单管理

可对罚单信息进行浏览；进行缴费操作；对已缴费的记录进行清理删除

1).浏览罚单信息

2).缴费

3).删除已缴费的记录

1.1.6还书信息

可浏览还书信息

**1.2 功能需求**

1. 图书的查阅、修改、删除、图书入库：编号、书名、作者、出版社、在库数、类别
2. 读者的查询、修改、删除、新增读者：借书证号、姓名、班级、院系
3. 管理员的查询、修改、删除、新增管理员：编号、姓名、负责书库
4. 借还信息的查询、还书（逾期欠费需缴费才能还书）、新增借书：借书证号、读者名、编号、书名、借书日期、预计归还日期，可删除新增记录
5. 罚单的查阅、缴费、删除（已缴费）
6. 还书信息的查阅

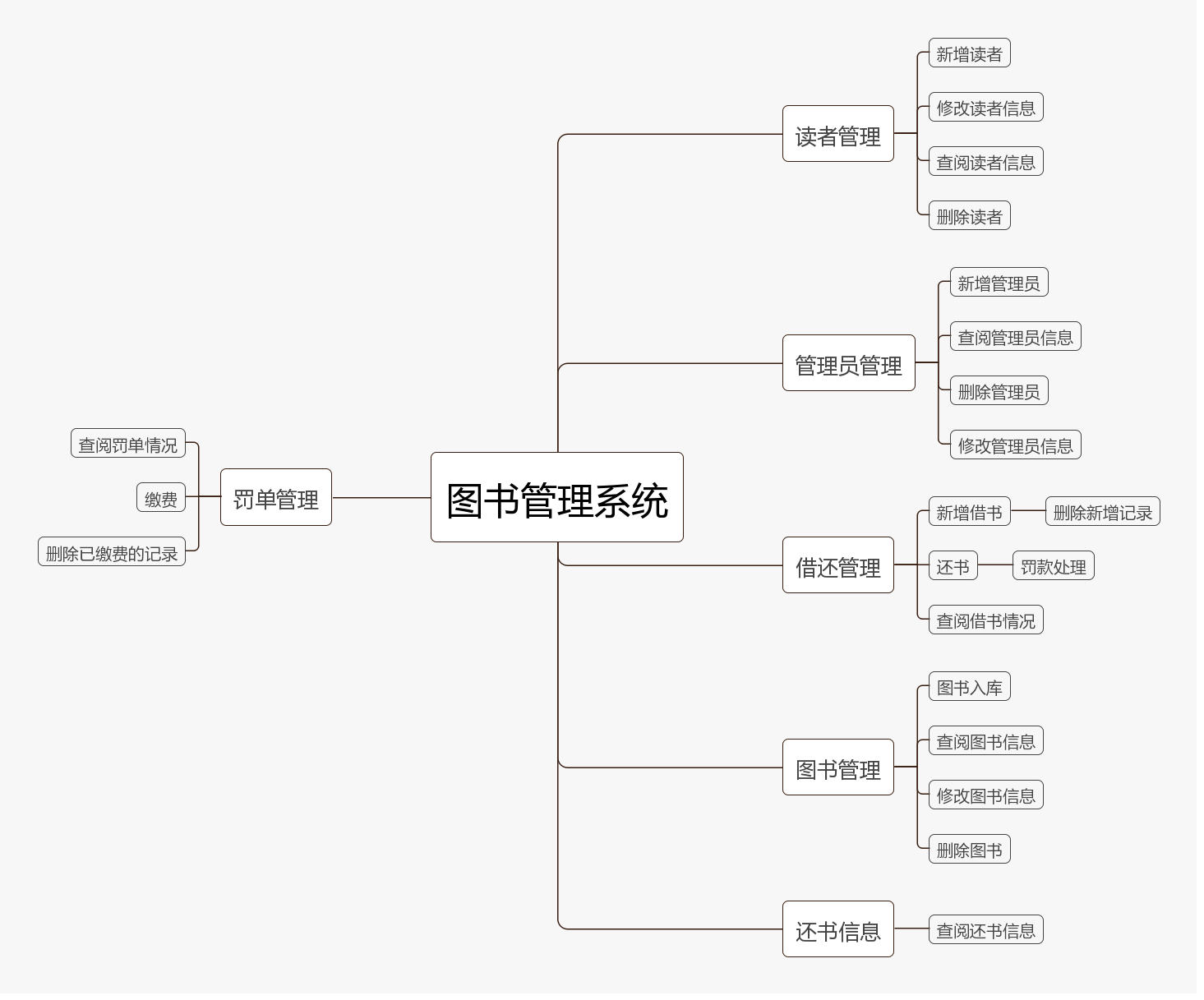


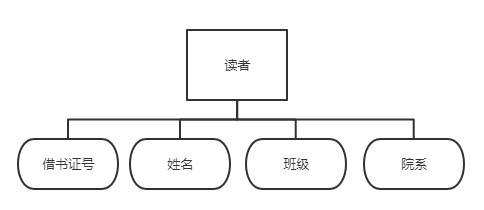
图1 图书管理系统功能图

**2 概念结构设计**

根据需求分析我们规划出实体有：读者信息实体、管理员信息实体、图书信息实体。各个实体具体的描述E-R实体如下。

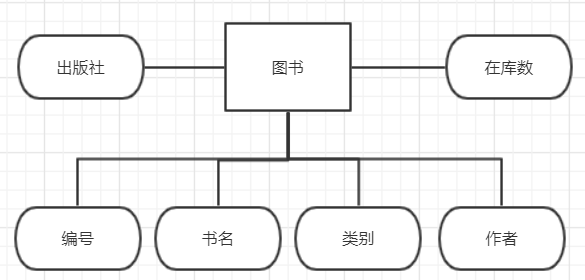
**2.1读者信息模块**

读者属性包括：借书证号、姓名、班级、院系



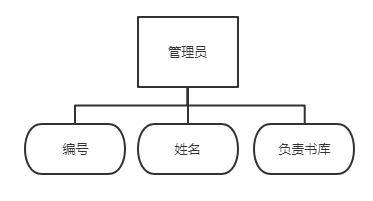
**2.2 图书信息模块**

图书属性包括：编号、书名、作者、出版社、在库数、类别



**2.3 管理员信息模块**

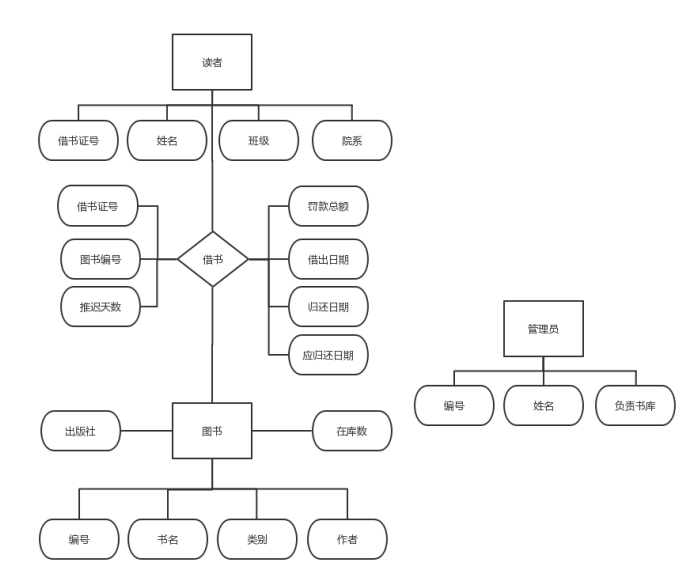
管理员属性包括：编号、姓名、负责书库



**2.4 完整的E-R图**

实体间的联系：

1. 读者借书：拥有借书证号、图书编号、借出日期、应归还日期 属性
2. 读者还书：拥有借书证号、图书编号、借出日期、应归还日期、归还日期、推迟天数、罚款总额 属性



**3 逻辑结构设计**

**3.1数据库逻辑设计**

由E-R模型得到关系模型：借阅关系模型

3.1.1 借阅关系

属性：借书证号、图书编号、借出日期、应归还日期、归还日期、推迟天数、罚款总额

主键：借书证号、图书编号

**3.2数据库逻辑结构**

1. 图书信息表(属性：id(INT(11))、name(VARCHAR(20))、author(VARCHAR(20))、publisher(VARCHAR(20))、num(INT(11))、sort(VARCHAR(20)))
2. 读者信息表(属性：id(INT(11))、name(VARCHAR(20))、class(VARCHAR(20))、department(VARCHAR(20)))
3. 管理员信息表(属性：id(INT(11))、name(VARCHAR(20))、stackroom(VARCHAR(10)))
4. 借书信息表(属性：bookId(INT(11))、readerId(INT(11))、borrow\_date(DATE)、expect\_rerurn\_date(DATE))
5. 罚单信息表(属性：bookId(INT(11))、readerId(INT(11))、over\_date(INT)、ticket\_fee(FLOAT)、payoff(INT))
6. 还书信息表(属性：bookId(INT(11))、readerId(INT(11))、borrow\_date(DATE)、rerurn\_date(DATE))

3.2.1 图书信息表

**4 物理设计**

由上述数据库逻辑结构转换为实际数据模型，本图书管理系统的各个表如下：

**4.1 建立图书馆数据库（library）**

**CREATE** **DATABASE** library;

**4.2 建立图书表（book）**

**CREATE** **TABLE** `book` (

`id` **INT**(11) **NOT** **NULL** **DEFAULT** '0',

`name` **VARCHAR**(20) **NOT** **NULL**,

`author` **VARCHAR**(20) **NOT** **NULL**,

`publisher` **VARCHAR**(20) **NOT** **NULL**,

`num` **INT**(11) **NOT** **NULL**,

`sort` **VARCHAR**(20) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

)

**COLLATE**='utf8\_general\_ci'

**ENGINE**=**InnoDB**

;

desc book;



**4.3 建立读者表（reader）**

**CREATE** **TABLE** `reader` (

`id` **INT**(11) **NOT** **NULL** **DEFAULT** '0',

`name` **VARCHAR**(20) **NOT** **NULL**,

`class` **VARCHAR**(10) **NOT** **NULL**,

`department` **VARCHAR**(10) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

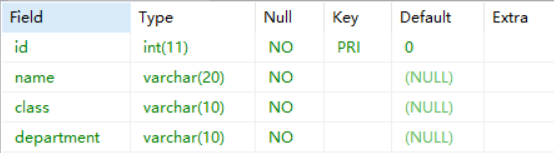
)

**COLLATE**='utf8\_general\_ci'

**ENGINE**=**InnoDB**

;

**desc** reader;



**4.4 建立借阅表（borrow）**

**CREATE** **TABLE** `borrow` (

`bookId` **INT**(11) **NOT** **NULL** **COMMENT** '学生编号',

`readerId` **INT**(11) **NOT** **NULL** **COMMENT** '图书编号',

`borrow\_date` **DATE** **NULL** **DEFAULT** **NULL** **COMMENT** '借书时间',

`expect\_return\_date` **DATE** **NULL** **DEFAULT** **NULL** **COMMENT** '预期归还时间',

**PRIMARY** **KEY** (`bookId`, `readerId`),

**INDEX** `READERID` (`readerId`, `bookId`),

**CONSTRAINT** `BOOKID` **FOREIGN** **KEY** (`bookId`) **REFERENCES** `book` (`id`),

**CONSTRAINT** `READERID` **FOREIGN** **KEY** (`readerId`) **REFERENCES** `reader` (`id`)

)

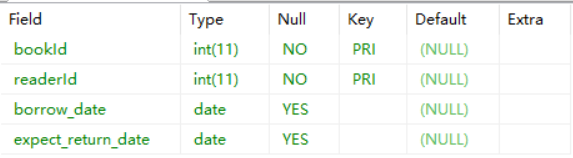
**COMMENT**='存储学生的借书信息'

**COLLATE**='latin1\_swedish\_ci'

**ENGINE**=**InnoDB**

;

**desc** borrow;



**4.5 建立管理员表（administartors）**

**CREATE** **TABLE** `administrators` (

`id` **INT**(11) **NOT** **NULL** **DEFAULT** '0',

`name` **VARCHAR**(20) **NOT** **NULL**,

`stackroom` **VARCHAR**(10) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

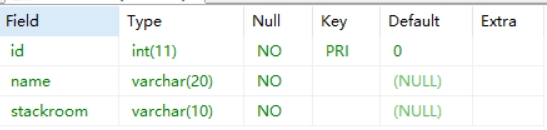
)

**COLLATE**='utf8\_general\_ci'

**ENGINE**=**InnoDB**

;

**desc** administrators;



**4.6 建立还书信息表（return\_table）**

**CREATE** **TABLE** `return\_table` (

`bookId` **INT**(11) **NOT** **NULL**,

`readerId` **INT**(11) **NOT** **NULL**,

`borrow\_date` **DATE** **NULL** **DEFAULT** **NULL**,

`return\_date` **DATE** **NULL** **DEFAULT** **NULL** **COMMENT** '实际还书情况',

**PRIMARY** **KEY** (`bookId`, `readerId`),

**INDEX** `readerId` (`readerId`, `bookId`),

**CONSTRAINT** `BOOKID\_R` **FOREIGN** **KEY** (`bookId`) **REFERENCES** `book` (`id`),

**CONSTRAINT** `READERID\_R` **FOREIGN** **KEY** (`readerId`) **REFERENCES** `reader` (`id`)

)

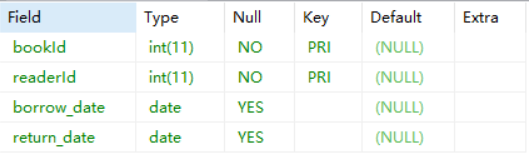
**COMMENT**='存储学生的归还信息'

**COLLATE**='latin1\_swedish\_ci'

**ENGINE**=**InnoDB**

;

**desc** return\_table;



**4.7建立罚单信息表（ticket）**

**CREATE** **TABLE** `ticket` (

`readerId` **INT**(11) **NOT** **NULL** **COMMENT** '学生编号',

`bookId` **INT**(11) **NOT** **NULL** **COMMENT** '书籍编号',

`over\_date` **INT**(11) **NULL** **DEFAULT** **NULL** **COMMENT** '超期天数',

`ticket\_fee` **FLOAT** **NULL** **DEFAULT** **NULL** **COMMENT** '处罚金额',

`payoff` **INT**(11) **NULL** **DEFAULT** **NULL**,

**PRIMARY** **KEY** (`readerId`, `bookId`),

**INDEX** `BOOKID\_T` (`bookId`),

**CONSTRAINT** `BOOKID\_T` **FOREIGN** **KEY** (`bookId`) **REFERENCES** `book` (`id`),

**CONSTRAINT** `READERID\_T` **FOREIGN** **KEY** (`readerId`) **REFERENCES** `reader` (`id`)

)

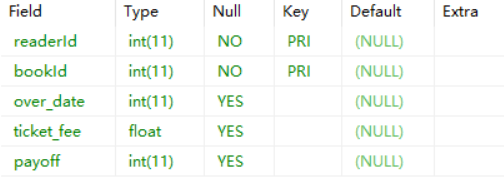
**COMMENT**='存储学生的罚单信息'

**COLLATE**='latin1\_swedish\_ci'

**ENGINE**=**InnoDB**

;

**desc** ticket;



**5 实施过程**

**5.1装载数据过程**

5.1.1 用DML语句对book表加入数据

**INSERT** **INTO** `library`.`book` (`id`, `name`, `author`, `publisher`, `num`, `sort`) **VALUES** ('8', '时事报告', '张习文', '时事报告杂志社', '1', '政治');

**INSERT** **INTO** `library`.`book` (`id`, `name`, `author`, `publisher`, `num`, `sort`) **VALUES** ('9', '计算机网络安全', '马利', '清华大学出版社', '1', '计算机');

**select** \* **from** book;



5.1.2 用DML语句对reader表加入数据

**INSERT** **INTO** `library`.`reader` (`id`, `name`, `class`, `department`) **VALUES** ('6', '张三', '1603', '机器人学院');

**INSERT** **INTO** `library`.`reader` (`id`, `name`, `class`, `department`) **VALUES** ('7', '李四', '1602', '金融学院');

**select** \* **from** reader;



5.1.3 用DML语句对administartors表加入数据

**INSERT** **INTO** `library`.`administrators` (`id`, `name`, `stackroom`) **VALUES** ('4', '王五', '外国文学');

**INSERT** **INTO** `library`.`administrators` (`id`, `name`, `stackroom`) **VALUES** ('5', '江六', '中国文学');

**select** \* **from** administrators;



**5.2设计视图**

5.2.1 在reader,borrow和book上创建借书者的全面信息视图reader\_borrow

**create** **view** reader\_borrow **as**

**select** reader.id **AS** reader\_id,reader.name **AS** reader\_name,book.id **AS** book\_id,book.name **AS** book\_name,borrow\_date,**addDate**(borrow\_date,30) expect\_return\_date

**from** reader,book,borrow

**where** reader.id = borrow.readerId **and** book.id = borrow.bookId;



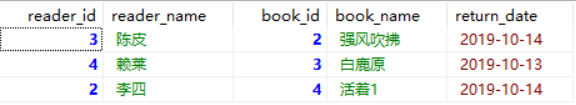
5.2.3 创建个人所有借书归还记录视图 reader\_borrow\_return

**create** **view** reader\_borrow\_return **as**

**select** reader.id **AS** reader\_id,reader.name **AS** reader\_name,book.id **AS** book\_id,book.name **AS** book\_name,return\_date

**from** reader,book,return\_table

**where** reader.id = return\_table.readerId **and** book.id = return\_table.bookId;



**5.3 存储过程设计**

5.3.1 产生罚单proc\_gen\_ticket

当日期超过预定归还日期，产生罚单，并将记录写入表ticket中，这个存储过程在定时器eventJob中调用

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `proc\_gen\_ticket`(

**IN** `currentdate` **date**

)

**BEGIN**

**replace** **into** ticket(readerId,bookId,over\_date,ticket\_fee,payoff)

**SELECT** reader\_id,book\_id,**dateDiff**(currentdate,reader\_borrow.expect\_return\_date),0.1\***datediff**(currentdate,reader\_borrow.expect\_return\_date),1

**FROM** reader\_borrow

**WHERE** currentdate>reader\_borrow.expect\_return\_date;

**END**

5.3.2 借书过程实现

1) 设计存储函数，func\_get\_booknum，返回书籍是否在架上

**CREATE** **DEFINER**=`root`@`%` **FUNCTION** `func\_get\_booknum`(

`book\_id` **INT**

)

**RETURNS** **int**(11)

**BEGIN**

**return**(**select** num **from** book **where** book.id=book\_id);

**END**

2) 设计存储过程proc\_borrow，调用func\_get\_booknum，判断书籍是否在架上，若为真，则借书成功，在borrow表中插入记录；否则提示失败

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `proc\_borrow`(

**IN** `reader\_id` **INT**,

**IN** `book\_id` **INT**,

**IN** `borrow\_date` **DATE**,

**IN** `expect\_return\_date` **DATE**

)

**BEGIN**

**if** func\_get\_booknum(book\_id) > 0 **then**

**insert** **into** borrow(readerId,bookId,borrow\_date,expect\_return\_date)

**values**(reader\_id,book\_id,borrow\_date,expect\_return\_date);

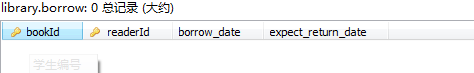
**else**

**select** -1 **as** msg;

**end** **if**;

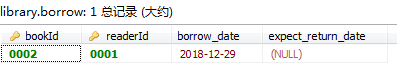
**END**

目前borrow记录如下

****

执行函数，读者0001借图书0002

**call** proc\_borrow('0001','0002',**CURDATE**());

****

****

执行函数，读者0002借图书0002

**call** proc\_borrow('0002','0002',**CURDATE**());

****

5.3.3 还书存储过程proc\_return

还书时，查看是否超期，即查询ticket表项，当发现超期，提示交罚单后再还书；如未超期，则记录归还项目到return\_table中，并且删除借书记录（以免还书后定时器仍然扫描这个记录）

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `proc\_return`(

**IN** `reader\_id` **INT**(11),

**IN** `book\_id` **INT**(11),

**IN** `return\_date` **DATE**

)

**BEGIN**

**declare** borrowdate **date**;

**if**(**select** payoff **from** ticket **where** ticket.readerId = reader\_id **and** ticket.bookId = book\_id) = 1 **then**

**select** -1 **as** msg;

**else**

**if**(**select** **count**(\*) **from** return\_table **where** return\_table.readerId = reader\_id **and** return\_table.bookId = book\_id ) > 0 **then**

**DELETE** **FROM** return\_table **WHERE** readerId = reader\_id;

**set** borrowdate = (**select** borrow\_date **from** borrow **where** borrow.readerId = reader\_id **and** borrow.bookId = book\_id);

**insert** **into** return\_table(readerId,bookId,borrow\_date,return\_date)

**values**(reader\_id,book\_id,borrowdate,return\_date);

**delete** **from** borrow

**where** borrow.readerId = reader\_id **and** borrow.bookId = book\_id;

**else**

**set** borrowdate = (**select** borrow\_date **from** borrow **where** borrow.readerId = reader\_id **and** borrow.bookId = book\_id);

**insert** **into** return\_table(readerId,bookId,borrow\_date,return\_date)

**values**(reader\_id,book\_id,borrowdate,return\_date);

**delete** **from** borrow

**where** borrow.readerId = reader\_id **and** borrow.bookId = book\_id;

**end** **if**;

**end** **if**;

**END**

读者0001借了0001图书，超期产生了罚单

****

**call** proc\_return('0001','0001',**CURDATE**());

****

交罚单

**call** proc\_payoff('0001','0001');

****

****

再次还书

****

5.3.4 交罚单存储过程proc\_payoff

修改罚单中payoff段为0，表明罚单已交

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `proc\_payoff`(

**IN** `reader\_id` **INT**,

**IN** `book\_id` **INT**

)

**BEGIN**

**update** ticket

**set** ticket.payoff = 0

**where** ticket.readerId = reader\_id **and** ticket.bookId = book\_id;

**select** 'succeed';

**END**

5.3.5 删除图书存储过程proc\_deleteBook

删除图书时，除了 book表，还要把关联表 return\_table 对应的书也删掉

**CREATE** **DEFINER**=`root`@`%` **PROCEDURE** `proc\_deleteBook`(

**IN** `book\_id` **INT**

)

**BEGIN**

**DELETE** **FROM** return\_table **WHERE** return\_table.bookId = book\_id;

**DELETE** **FROM** book **WHERE** book.id = book\_id;

**END**

**5.4触发器设计**

5.4.1 设计触发器borrow\_after\_insert

当某学生借书成功后，图书表相应的图书不在架上，变为0；

**CREATE** **TRIGGER** `borrow\_after\_insert` **AFTER** **INSERT** **ON** `borrow` **FOR** **EACH** **ROW** **BEGIN**

**update** book **set** num = num-1 **where** id = **new**.bookId;

**END**

在插入表borrow之前，id=0003的图书还在架上，为1



学生0001借了这本书后，在boorow中插入记录



在borrow中插入这条记录后，id=0003的图书，不在架上，为0

****

5.4.2 设计触发器return\_after\_insert

还书成功后，对应的书籍num变为1

**CREATE** **TRIGGER** `return\_after\_insert` **AFTER** **INSERT** **ON** `return` **FOR** **EACH** **ROW** **BEGIN**

**update** book **set** num = num+1 **where** id = **new**.bookId;

**END**

还书时在return\_table插入

****

此时图书归还架上

****

5.4.3 定义定时器（事件）eventJob

每天自动触发一次，扫描视图reader\_borrow，若发现当前有预期归还时间小于当前时间，则判断为超期，生成处罚记录，这个定时器将每天定时触发存储过程proc\_gen\_ticket

**CREATE** **EVENT** `eventJob`

**ON** **SCHEDULE**

**EVERY** 1 **DAY** **STARTS** '2018-08-01 16:22:42'

**ON** **COMPLETION** **PRESERVE**

**ENABLE**

**COMMENT** ''

**DO** **call** proc\_gen\_ticket(**CURDATE**())

读者0003借了图书0001，生成借书记录reader\_borrow视图



当他在1月9日前还书，没有生成罚单



当他在1月9日后还书，生成罚单





**6 系统实现**

**6.1 搭建前端框架**

6.1.1 搭建 vue-cli 脚手架

本系统使用 VUE 编写，所以需要搭建脚手架，使系统达到：提高效率、组件化、模块化、数据双向绑定的优秀特性

// 在项目的根目录下

npm install vue-cli -g

vue init webpack web-lib

cd web-lib

npm install

npm run dev

6.1.2 引入 axious、element库

因为有很多http请求，使用axious库，可以更方便的书写请求，并且易于管理；对于一个后台系统，其ui适合使用简洁的风格，element-ui即可满足

// 引入 aixos

npm i aixos vue-axios -s

在入口文件 main.js 里面加入

**import** axios **from** 'axios'

**import** VueAxios **from** 'vue-axios'

Vue.**use**(VueAxios, axios);

// 引入 elment-ui

npm i element-ui -S

在入口文件 main.js 里面加入

**import** ElementUI **from** 'element-ui';

**import** 'element-ui/lib/theme-chalk/index.css';

Vue.**use**(ElementUI);

6.1.3 上传到github仓库

需要对系统进行版本控制，在本地建立仓库后，与远程仓库进行连接

// 在项目目录中

git init

git remote **add** origin https://github.com/ZhangWei2222/web-lib.git

git pull

git **add** .

git **commit** -m '注释信息'

git push -u origin **master**

可在 <https://github.com/ZhangWei2222/web-lib> 查看

**6.2 使用 NodeJs 建立本地服务器**

使用 NodeJs 开发系统中间层接口，直接从 MySql 获取数据并通过 express 制作数据获取接口

// 在项目的根目录中

"use strict";

*/\* 引入express框架 \*/*

const express = **require**('express');

const app = express();

*/\* 引入cors \*/*

const cors = **require**('cors');

app.**use**(cors());

*/\* 引入body-parser \*/*

const bodyParser = **require**('body-parser');

app.**use**(bodyParser.**json**());

app.**use**(bodyParser.urlencoded({ **extended**: **false** }));

*/\* 引入mysql \*/*

const mysql = **require**('mysql');

const conn = mysql.createConnection({

**host** : '47.101.130.10',

**user** : 'root',

**password** : '1998WEI.',

**database** : 'library',

multipleStatements: **true**

})

conn.connect();

*/\* 监听端口 \*/*

app.listen(4444, () => {

console.**log**('——————————服务已启动——————————');

})

app.**get**('/', (req, res) => {

res.send('<p style="color:red">服务已启动</p>');

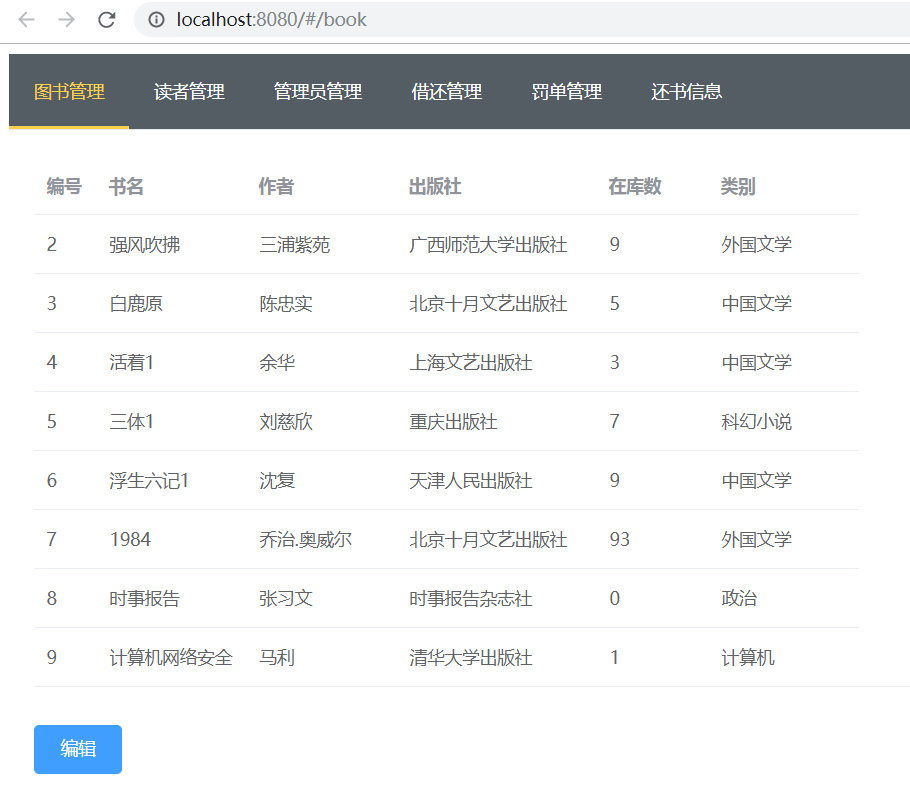
})

**6.3 图书管理-页面**

6.3.1 功能介绍

1).导航：/book

2).显示功能：展示图书信息



3).编辑功能：进入图书编辑界面，可对图书进行修改、删除、入库，操作成功/失败会有相应提示。

注1：修改、入库后需要点击完成，才能算操作完成

注2：编辑在库数，已限制只能键入数字

注3：记录均不可为空，完成提交时会检查，如为空，会提示，且无法提交





6.3.2 接口设计

// 书本列表

app.**get**('/api/getBooks', (req, res) => {

const sqlStr = 'SELECT \* FROM book'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 更新书本列表

app.post('/api/updateBook',(req,res) => {

let str = ''

**for**(var **key** **in** req.body) {

str += **key** + ' = "' + req.body[**key**] + '",'

}

str = str.**substr**(0, str.**length**-1);

const sqlStr = 'UPDATE book SET ' + str + ' WHERE id = ' + req.body.id

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 添加书本

app.post('/api/addBook',(req,res) => {

let str1 = '', str2 = ''

**for**(var **key** **in** req.body) {

str1 += **key** + ','

str2 += req.body[**key**] + ','

}

str1 = str1.**substr**(0, str1.**length**-1);

str2 = '"' + str2.**substr**(0, str2.**length**-2);

const sqlStr = 'INSERT INTO book (' + str1 + ')' + ' VALUES (' + str2 + ')'

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 删除书本

app.post('/api/delectBook',(req,res) => {

const sqlStr = 'DELETE FROM book WHERE id = ' + req.body.id

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

**6.4 读者管理-页面**

6.4.1 功能介绍

1).导航：/user

2).显示功能：展示读者信息



3).编辑功能：进入读者编辑界面，可对读者进行修改、删除、新增，操作成功/失败会有相应提示。

注1：修改、新增后需要点击完成，才能算操作完成

注2：编辑班级，已限制只能键入数字

注3：记录均不可为空，完成提交时会检查，如为空，会提示，且无法提交







6.4.2 接口设计

// 显示用户列表

app.**get**('/api/getUsers', (req, res) => {

const sqlStr = 'SELECT \* FROM reader'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 更新用户列表

app.post('/api/updateUser',(req,res) => {

let str = ''

**for**(var **key** **in** req.body) {

str += **key** + ' = "' + req.body[**key**] + '",'

}

str = str.**substr**(0, str.**length**-1);

const sqlStr = 'UPDATE reader SET ' + str + ' WHERE id = ' + req.body.id

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 添加用户

app.post('/api/addUser',(req,res) => {

let str1 = '', str2 = ''

**for**(var **key** **in** req.body) {

str1 += **key** + ','

str2 += req.body[**key**] + '","'

}

str1 = str1.**substr**(0, str1.**length**-1);

str2 = '"' + str2.**substr**(0, str2.**length**-2);

const sqlStr = 'INSERT INTO reader (' + str1 + ')' + ' VALUES (' + str2 + ')'

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 删除用户

app.post('/api/delectUser',(req,res) => {

const sqlStr = 'DELETE FROM reader WHERE id = ' + req.body.id

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

**6.5 管理员管理-页面**

6.5.1 功能介绍

1).导航：/admin

2).显示功能：展示管理员信息



3).编辑功能：进入管理员编辑界面，可对管理员进行修改、删除、新增，操作成功/失败会有相应提示。

注1：修改、新增后需要点击完成，才能算操作完成

注2：记录均不可为空，完成提交时会检查，如为空，会提示，且无法提交



6.5.2 接口设计

// 显示管理员列表

app.**get**('/api/getAdmin', (req, res) => {

const sqlStr = 'SELECT \* FROM administrators'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 更新管理员列表

app.post('/api/updateAdmin',(req,res) => {

let str = ''

**for**(var **key** **in** req.body) {

str += **key** + ' = "' + req.body[**key**] + '",'

}

str = str.**substr**(0, str.**length**-1);

const sqlStr = 'UPDATE administrators SET ' + str + ' WHERE id = ' + req.body.id

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 添加管理员

app.post('/api/addAdmin',(req,res) => {

let str1 = '', str2 = ''

**for**(var **key** **in** req.body) {

str1 += **key** + ','

str2 += req.body[**key**] + '","'

}

str1 = str1.**substr**(0, str1.**length**-1);

str2 = '"' + str2.**substr**(0, str2.**length**-2);

const sqlStr = 'INSERT INTO administrators (' + str1 + ')' + ' VALUES (' + str2 + ')'

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 删除管理员

app.post('/api/delectAdmin',(req,res) => {

const sqlStr = 'DELETE FROM administrators WHERE id = ' + req.body.id

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

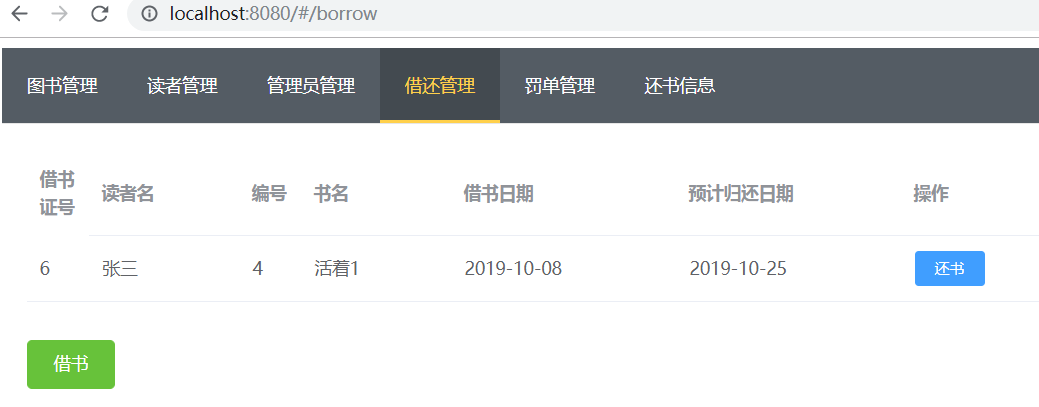
})

**6.6 借还管理-页面**

6.6.1 功能介绍

1).导航：/borrow

2).显示功能：展示借还信息



3).借书功能：新增借书记录，点击确认即完成操作；可按删除，取消操作

注1：记录均不可为空，完成提交时会检查，如为空，会提示，且无法提交

注2：读者名，书名采取下拉框的形式，选择完毕，唯一标识符借书证号、编号会自动填上

注3：每次只能进行一次借书操作，请完成新增操作再进行下面的操作

注4：每次借书，图书中的在库数会减少，为0的，无法借阅







4).还书功能：进行还书；如超过预计归还日期，产生罚单，需要到 罚单管理 进行缴费，才能进行归还

注：还书后，相应图书的在库数会增加





6.6.2 接口设计

// 显示借书列表

app.**get**('/api/getBorrow', (req, res) => {

const sqlStr = 'SELECT \* FROM reader\_borrow'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 获取 借书证号 列表

app.**get**('/api/getUserId', (req, res) => {

const sqlStr = 'SELECT id,name FROM reader'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 获取 编号 列表

app.**get**('/api/getBookId', (req, res) => {

const sqlStr = 'SELECT id,name FROM book'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 借书

app.post('/api/borrowBook',(req,res) => {

const sqlStr = 'call proc\_borrow("' + req.body.reader\_id + '","' + req.body.book\_id + '","' + req.body.borrow\_date + '","' + req.body.expect\_return\_date + '")'

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 还书

app.post('/api/returnBook',(req,res) => {

const sqlStr = 'call proc\_return("' + req.body.readerId + '","' + req.body.bookId + '","' + req.body.**curDate** + '")'

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

**6.7 罚单管理-页面**

6.7.1 功能介绍

1).导航：/ticket

2).显示功能：展示罚单信息，依据 是否欠费 进行排序，欠费记录为1，已缴费记录为0；可对已缴费的记录进行删除

注：系统每天会对借还列表进行查看，超过逾期，产生罚单



3).缴费功能：对欠费的记录进行缴费，之后可以归还书籍



6.7.2 接口设计

// 显示欠费列表

app.**get**('/api/getTicket', (req, res) => {

const sqlStr = 'SELECT \* FROM reader\_ticket'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})

// 缴费

app.post('/api/payTicket',(req,res) => {

const sqlStr = 'call proc\_payoff("' + req.body.readerId + '","' + req.body.bookId + '")'

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

// 删除已缴的欠费单

app.post('/api/delectTicket',(req,res) => {

const sqlStr = 'DELETE FROM ticket WHERE readerId = ' + req.body.readerId + ' AND bookId = ' + req.body.bookId

conn.**query**(sqlStr, (err,results) => {

**if**(err) **return** res.**json**({ **code**: -1, message: err })

res.**json**({ **code**: 10000, **data**: results })

})

})

**6.8 还书信息-页面**

6.8.1 功能介绍

1).导航：/return

2).显示功能：展示还书信息



6.8.2 接口设计

// 显示还书列表

app.**get**('/api/getReturn', (req, res) => {

const sqlStr = 'SELECT \* FROM reader\_borrow\_return'

conn.**query**(sqlStr, (error, results) => {

**if** (error) **return** res.**json**({ **code**: -1, message: error})

res.**json**({ **code**: 10000, **data**: results})

})

})