需要redis，mysql，springboot，一个云储存服务器阿里OSS，用来存放图片，pdf资源

**前后端同一约定格式**

{

code：状态码：20000表示处理成功，20001表示处理失败，20002表示服务异常

message：一些补充信息，出现异常时携带，例如注册时用户名重复

data：{ 返回的数据集

}

}

当前端收到错误的状态码的时候，可以使用message打印错误信息

输入格式由前端校验，后端只需要接收

用户权限验证暂不考虑

**用户相应操作模块：8001**

user表

{

uid：自增

userName:String

passWord:String,(MD5)加密

mail:String --前端校验是否为@结尾

debt: 欠费数额

phoneNumber:String --前端校验是否合法

}

提供用户登录，用户注册功能，其中密码找回采用邮件加验证码的形式

后端提供api：

**登录**

8001：/usercenter/login

post方式，请求格式：{

userName:xxx，

passWord:xxx

}

说明：该方法需要验证userName是否存在，验证passWord是否正确，成功之后，需要生成一个token，token内容是通过uid加密得到（这种加密一般可以使用JWT，参考https://blog.csdn.net/Mr\_Dracy/article/details/116738663），返回数据格式为

{

**. . .**

data:{

token:xxx

}

}

登录之后前端保留这个token，以cookie的形式记录下来，下一次访问就直接在url中附加这个token，后端每一次解密这个token从而获得用户id

登录成功后跳转到主页面

**注册**

8001: /usercenter/regist

post方式，请求格式：{

userName:

passWord:

mail:

phoneNumber:  
 }

后端验证注册时，要检查是否已经有重复用户名，如果有的话就终止操作，如果注册成功，就将信息放入数据库中，其中密码这一项需要经过MD5加密之后再放入数据库

前端获得返回结果，然后根据状态码进行注册判断，如果成功跳转至登录页面

**密码找回**

密码找回，首先是要通过邮件发送验证码，用户只有在输入正确验证码之后才能进行密码修改，验证码有效时长设置为5分钟，通过redis缓存存放。涉及到两个函数

8001：/email/sendcode --发送验证码

GET方式，后端随机生成一个六位数字，存入redis后，通过邮件发送到对方邮箱中

8001：/email/checkandchange --根据用户id从redis取出验证码验证后修改密码

POST方式，{

password:xxx

}

返回{

**....**

}

**查询用户个人信息**

8001：/usercenter/getuserinfo

GET

返回类型

{

uid：

userName:String

mail:String

phoneNumber:String

}

根据url的token获得id之后查表获得完整的用户信息，注意，密码不需要

**查询个人欠款：**

**8001: /usercenter/getuserdebt/{uid}**

欠款通过查询数据库中借阅表中的记录从而获得该用户逾期的总天数，按每天1元的价格费，返回

debtinfo{

uid:

debt: 欠款额

}

**图书服务模块：8002 isbn为主键**

**表设计**

book{

ISBN 默认13位

categoryID 分类ID

parentID 父分类ID

location ： 图书存放位置

ISBNcode url： 条形码链接

parentName 父分类名

bookName：

author: //作者

bookPrice： 默认0

createDate: 图书添加时间

discreption：图书描述

bookNum：图书的数量 默认0

bookRemain: 图书剩余库存

cover： 一个url，指向一个地址，存放图书封面的链接

pdf： 一个url，主要是用来存放图书pdf链接

}

借阅表

borrow{

userID

bookID

createTime 记录创建时间

returnTime 归还日期，为空则未归还

}

分类表：

category{

categoryID 自增

categoryName

parentID 父分类的categoryID

}

限制：分类最多两级

后端提供的api

**~~上传图书封面~~**

~~8002：/bookservice/uploadcover~~

~~post方法，接口参数为MultipartFile，接收前端上传的图片，然后将其上传到云端的储存器中，返回上传后得到的存储url，并发给前端~~

暂时废弃，因为isbn可以提供封面信息，此时后端直接通过前端上传的封面链接作为封面url并储存在数据库中

**上传图书pdf**

8002：/bookservice/uploadbookpdf

post方法，接口参数为MultipartFile，接收前端上传的pdf文件，然后将其上传到云端的储存器中，返回上传后得到的储存url，并发给前端

后端提供该方法，前端可以暂不实现

**添加分类**

8002：/bookservice/addcategoryinfo

post方法

请求格式：

{

categoryName:xxx,

parentName:xxx,

}

**获得分类信息**

8002：/bookservice/getbookcategoryinfo

GET

返回格式

{

code:xxx

message:xxx

data:{

parentCategories:

[

{

categoryID:xxx

categoryName:xxx

children:[

{

categoryID:xxx

categoryName:xxx

},

{

categoryID:xxx

categoryName:xxx

},

]

}，

{

categoryID:xxx

categoryName:xxx

children:[

{

categoryID:xxx

categoryName:xxx

},

{

categoryID:xxx

categoryName:xxx

},

]

}，

…

]

}

}

**上传图书信息**

**post方法中的信息是必要的，需要用户输入**

8002：/bookservice/addbookinfo

post 方法，参数

{

categoryID 分类ID

categoryName： 分类名

parentName 父分类名

parentID 父分类ID

location 图书存放位置

ISBN

}

后端调用https://m.jisuapi.com/api/isbn/ 中的api获取详细图书信息

同时根据isbn生成一个条形码，通过java工具类可做到，然后放在阿里云上，获得url存放至本地

储存时，如果isbn已存在，就终止并返回message

**修改图书信息 不可修改图书的描述信息，因为是通过isbn获得的**

8002: /bookservice/modifybookinfo

post方法，参数

{

ISBN

categoryID 分类ID

categoryName： 分类名

parentName 父分类名

parentID 父分类ID

location

bookNum：图书的数量

bookRemain ：剩余数量

pdf： 一个url，主要是用来存放图书pdf链接

}

通过bookID找到对应记录行然后更改

**删除图书信息**

8002：/bookservice/deleteBook/{ ISBN }

get

**通过图书id获得图书信息**

8002：/bookservice/getbookinfobyID/{ ISBN}

GET

bookInfo:{

ISBN：13位

categoryID 分类ID

parentID 父分类ID

ISBNcode url： 条形码链接

parentName 父分类名

bookName：

location

author: //作者

bookPrice： 默认0

createDate: 图书添加时间

isBorrowed: 是否借出 默认false

discreption：图书描述

bookNum：图书的数量 默认0

cover： 一个url，指向一个地址，存放图书封面的链接

pdf： 一个url，主要是用来存放图书pdf链接

}

**获得图书信息,带分页功能**

8002：/bookservice/getbook/{page}/{limit} //page，查询第page页；limit，一页有limit条记录

GET

booklist:

[

{

ISBN：13位

categoryID 分类ID

parentID 父分类ID

ISBNcode url： 条形码链接

parentName 父分类名

location

bookName：

author: //作者

bookPrice： 默认0

createDate: 图书添加时间

isBorrowed: 是否借出 默认false

discreption：图书描述

bookNum：图书的数量 默认0

cover： 一个url，指向一个地址，存放图书封面的链接

pdf： 一个url，主要是用来存放图书pdf链接

},

{

…

}

]

**获得图书信息，通过条件方式**

8002: /bookservice/booksearch/{page}/{limit}

POST

{

searchName

categoryID

parentID

}

后端通过传入数据做条件查询，如果其中某个为空，则不使用，searchName用模糊查询

如果categoryID为null，则默认查询该大类下的所有图书

**借阅图书**

8002:/bookservice/borrowbook/{ ISBN}

GET

后端判断借阅的时候，首先检查图书是否还有库存，没有则终止，其次，检查借阅者是否已经借阅了五本书，是的话就不允许借阅，都检查过后才能借阅。同时，只有欠款还清了才能借阅，欠款可以在借阅之前先进行计算，方法是通过远程调用获取该用户的debt，然后返回。借阅失败的话要在返回的message中添加失败的原因，成功了记得同步图书表对应的剩余数量，然后向借阅关系表中放一条记录

用户ID从url的token中获得

**归还图书**

8002：/bookservice/returnbook/{ ISBN}

GET

归还的时候，根据url的token获得用户ID，然后记得同步图书表对应的剩余数量，借阅关系表中的记录不必移除，而是将return标记为1，这样可以让用户查看所有历史借阅记录

**查看某一用户所有借阅记录**

8002: /bookservice/getborrowedhistory/{page}/{limit}

根据url中的token获得用户ID

borrowlist:[

{

userID

userName

bookName

ISBN

createTime 记录创建时间

dateToReturn 应还日期

isReturned： true or false

}，{

…

}

]

**续借记录**

8002:/bookservice/renewbook/{borrowId}/{day}

GET

**网站服务：8003**

**功能一 统计信息查询**

数据库表：statistics{

visitNumber int//当日访问人数，通过登录次数获得

borrowNumber int//当日总借阅数

returnNumber int//当日总归还数

createTime datetime //记录日期

}

**获得某个日期的请求记录**

8003：statistics/getvisitnumber/

post

{

year: 如果请求某年记录，就不要加上month，day，如果请求某月记录，就不要加上day，发送要么是year-month-day,要么是year-month，要么是year

month:

day:

}

statistics{

visitNumber: 这一 /年/月/日 登录人数可看做当日访问人数

borrowNumber: 这一 /年/月/日 借阅人数

returnNumber: 这一 /年/月/日 最热门借阅图书，该项通过跨服务调用

}

数据统计通过spring aop切面在每一次请求前拦截并添加到当日统计变量中，保证该变量线程安全，可以通过spring gateway统一请求处理，创建定时任务，每日定时将数据放入统计表中

**获得本周热门图书**

8003: statistics/getweekhotbook/

get

{

bookId: xxx

}

本周热门图书通过搜索借阅记录中的近一周内最多被借阅图书获得

**功能二 网站建议和反馈**

数据表

advice{

content: varchar(511)//反馈内容

createdate： 添加日期

qq: varchar(15) 联系方式，qq

}

**添加反馈**

**8003:** /station/advice/

post数据格式{

content: 内容

qq：

}

**功能三 支付管理**

数据库：order{//订单记录

order\_id:订单号：uid+创建日期 varchar(32)

create\_time:创建时间 datetime

uid：用户id

}

paylog{//订单支付成功记录

paylog\_id

order\_id

create\_time: datetime

}

**支付流程概述**

前端通过 8003:/pay/paydebt/{money} 来支付自己的欠款，后端生成一个订单，将订单号与相应许可发送给支付宝官方api，api文档在

<https://opendocs.alipay.com/open/02ekfg?scene=19>

中，通过api获得一个二维码地址，然后返回给前端

order:{

orderId:

payUrl: //二维码地址

}

前端通过二维码地址获取二维码然后显示在主界面，设置一个定时函数，每两秒钟向后端查询支付状态，url为

8003: /pay/getstatus/{orderId}

后端通过orderid向支付宝接口查询状态，后端返回时，code=20000表示成功支付，然后跳转到成功页面，否则返回20001表示未成功支付