**E-bookshop项目设计文档**

项目成员

童路勤 18301077

卢君亮 18301075

高梦楠 18301063

目录

[一、基本信息 3](#_Toc75216510)

[1.1开发人员 3](#_Toc75216511)

[1.2开发环境及工具 3](#_Toc75216512)

[二、总体设计 3](#_Toc75216513)

[2.1书城介绍及功能 3](#_Toc75216514)

[2.2书城使用流程图 4](#_Toc75216515)

[2.3书城架构图: 5](#_Toc75216516)

[2.4数据库设计: 5](#_Toc75216517)

[2.5 API设计 6](#_Toc75216518)

[2.5.1前端路由设计 6](#_Toc75216519)

[2.5.2 Restful API设计 6](#_Toc75216520)

[三、详细设计（Spring Cloud） 7](#_Toc75216521)

[3.1 Eureka（服务发现框架） 7](#_Toc75216522)

[3.1.1 Eureka Server 7](#_Toc75216523)

[3.1.2 Eureka Client 9](#_Toc75216524)

[3.2 Gateway 9](#_Toc75216525)

[3.2.1 Gateway\_service 9](#_Toc75216526)

[3.3业务服务 10](#_Toc75216527)

[3.3.1 Book\_service 10](#_Toc75216528)

[3.3.2 Collect\_service: 11](#_Toc75216529)

[3.4 Openfeign 11](#_Toc75216530)

[3.5 Hystrix 12](#_Toc75216531)

[3.6 Keyclock 13](#_Toc75216532)

# 一、基本信息

## 1.1开发人员

|  |  |
| --- | --- |
| 人员 | 职责 |
| 童路勤 | Keycloak对接 |
| 卢君亮 | 微服务配置 |
| 高梦楠 | 设计文档，数据库设计 |

## 1.2开发环境及工具

操作系统：Windows 10

开发语言：Java、Html、CSS、JavaScript等

开发IDE：IntelliJ IDEA

编译环境：Java JDK 1.8

开发框架：Spring Boot、Spring Cloud等

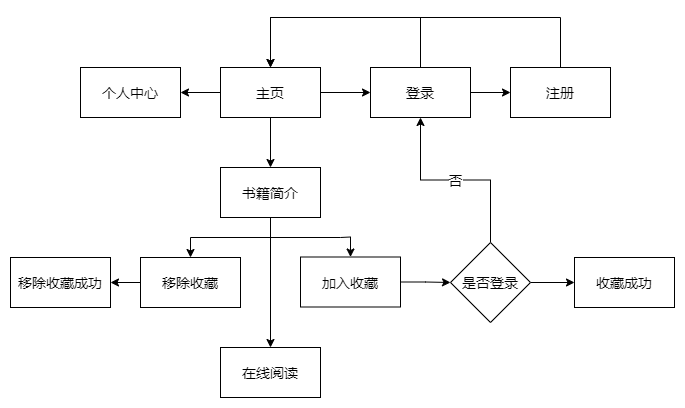
# 二、总体设计

## 2.1书城介绍及功能

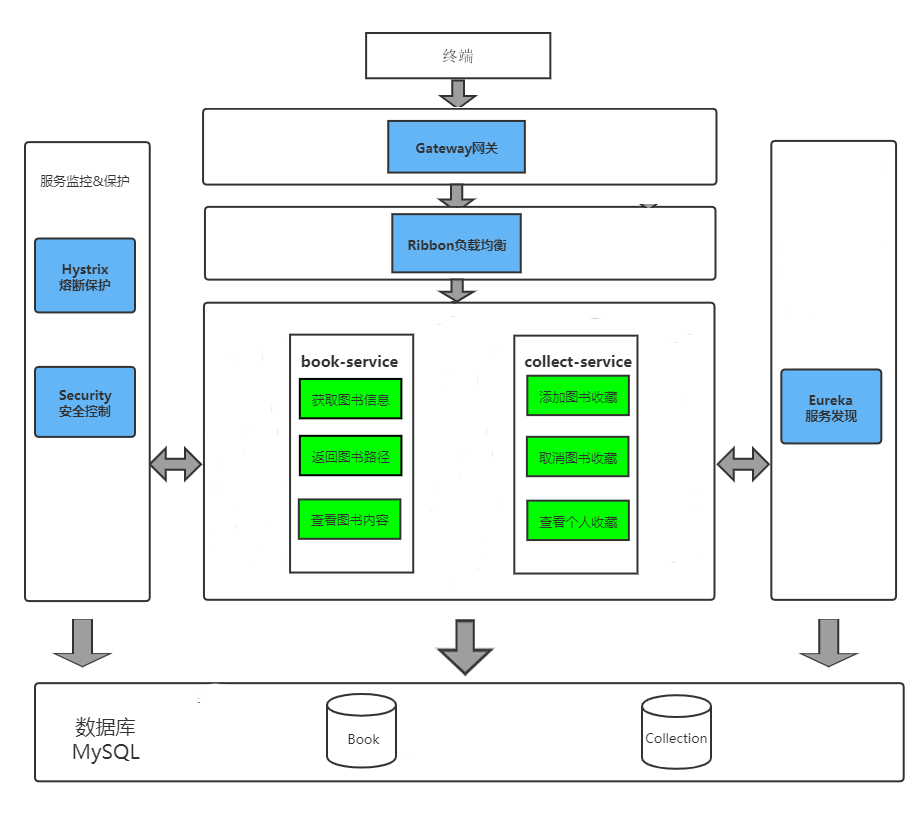
E-bookshop网站的开发主要包括后端数据库的建立，前端网页的设计，微服务的设计，以及如何用RESTful API进行前后端通信。在后端数据库设计中，我们确保数据一致和完整，而在前端网页中，我们力求做到页面简洁明了，方便操作。本次开发中，以Spring Cloud框架搭建微服务后台，使用Eureka进行服务发现、前端使用axios请求后端的REST API、Openfeign实现服务间RESTful通信、Hystrix实现熔断处理，利用keyclock软件来对实现Oauth2.0用户授权管理等。

项目主要功能有：登录、注册、查看图书上架信息，在线阅读书籍，添加或删除书籍收藏，修改个人信息等。

## 2.2书城使用流程图



## 2.3书城架构图:



## 2.4数据库设计:

|  |  |
| --- | --- |
| 表 | 描述 |
| Book | 电子书信息 |
| Collection | 用户收藏信息 |

**Book：**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| Title | Varchar | 书籍名称 |
| Author | Varchar | 作者 |
| mAbstract | Text | 简介 |
| Category | Varchar | 书籍分类 |
| filePath | Varchar | pdf文件路径 |
| Phases | Text | 章节目录 |
| imagePath | Varchar | 封面图片路径 |

**Collection：**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 描述 |
| Username | Varchar | 用户名 |
| Title | Varchar | 书籍名称 |

## 2.5 API设计

### 2.5.1前端路由设计

项目采用Vue-router history模式构建的前端路由

|  |  |
| --- | --- |
| URL | 描述 |
| / | 主页 |
| /abstract/{title} | 书籍介绍页 |
| /preview/{title} | 在线阅读页 |
| /personal | 个人信息页 |

### 2.5.2 Restful API设计

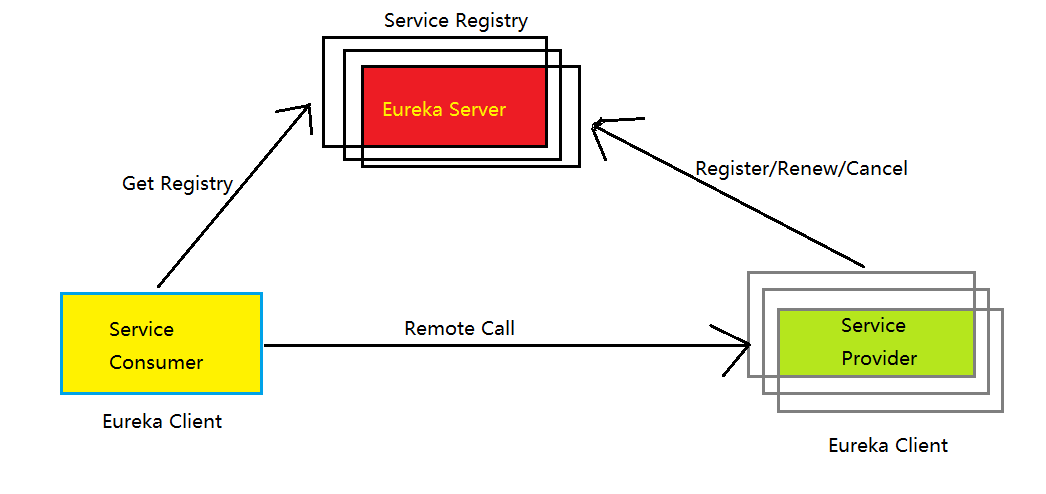
项目采用Spring Restful API设计

|  |  |  |
| --- | --- | --- |
| URL | HTTP方法 | 描述 |
| / | GET | 前端页面入口 |
| /books | GET | 获取书籍列表 |
| /books/{title} | GET | 获取某本书籍信息 |
| /collect/{username}/collections | GET | 获取用户收藏列表 |
| /collect/{username}/collections/{title} | GET | 获取用户对某本书的收藏信息 |
| /collect/{username}/collections/{title} | POST | 添加用户对某本书的收藏信息 |
| /collect/{username}/collections/{title} | DELETE | 删除用户对某本书的收藏信息 |

# 三、详细设计（Spring Cloud）

## 3.1 Eureka（服务发现框架）

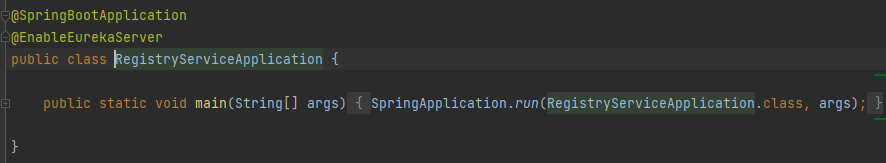
Eureka是Netflix开发的服务发现框架，SpringCloud将它集成在其子项目spring-cloud-netflix中，以实现SpringCloud的服务发现功能。Eureka包含两个组件：Eureka Server和Eureka Client。在应用启动后，将会向Eureka Server发送心跳,默认周期为30秒，如果Eureka Server在多个心跳周期内没有接收到某个节点的心跳，Eureka Server将会从服务注册表中把这个服务节点移除(默认90秒)。

**Eureka框架图：**

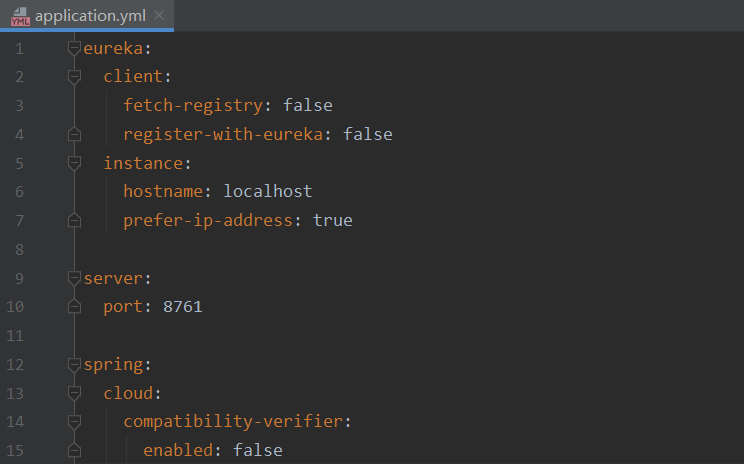
### 3.1.1 Eureka Server

Eureka Server提供服务注册服务，各个节点启动后，会在Eureka Server中进行注册，这样Eureka Server中的服务注册表中将会存储所有可用服务节点的信息，服务节点的信息可以在界面中直观的看到。

代码实现如下：

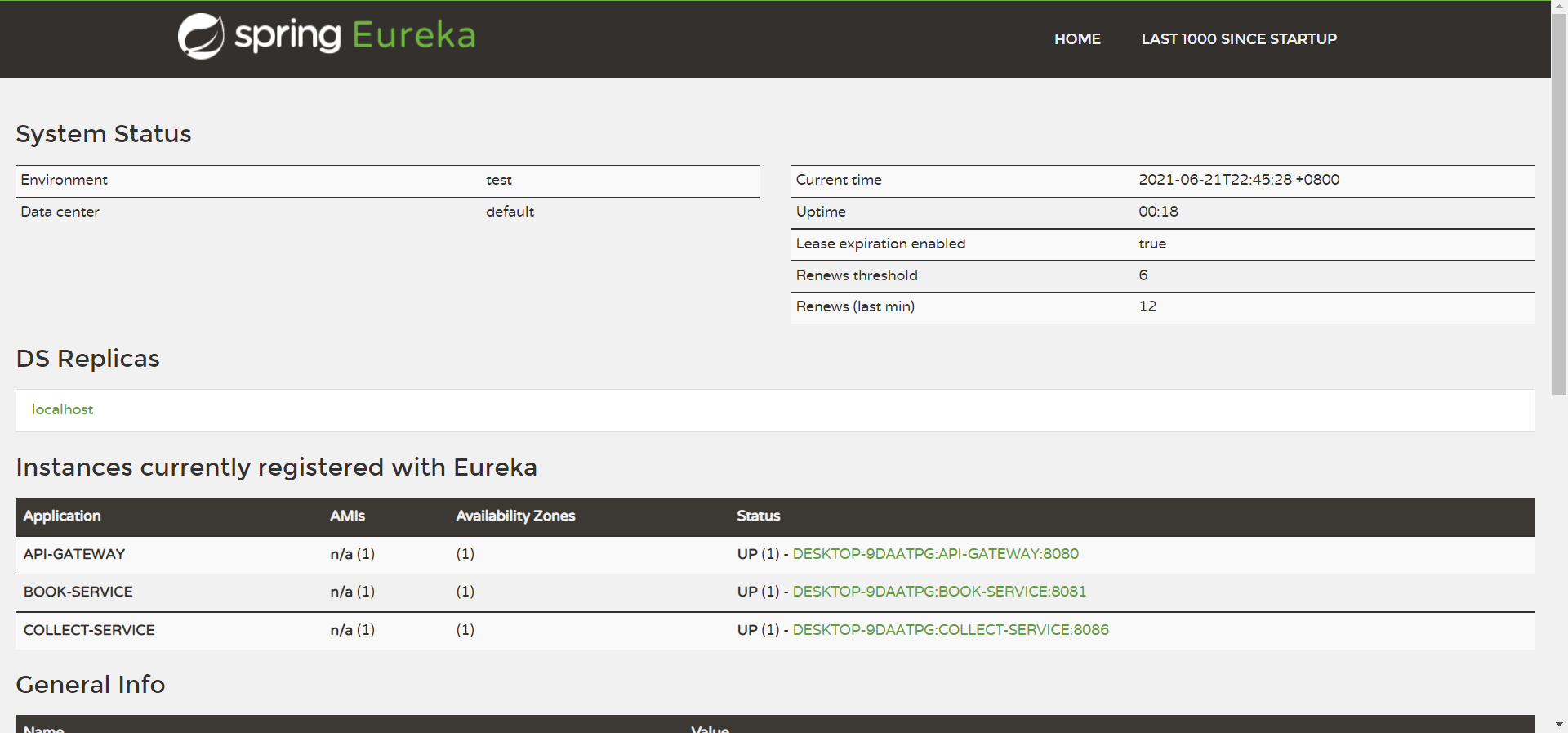


配置如下：



通过启动registry\_service中的RegistryServiceApplication打开Eureka Server。

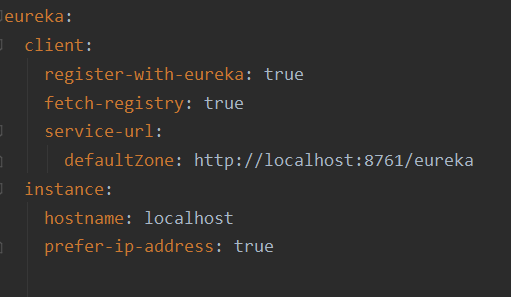
所有服务启动后，可以在Eureka管理页上查看：



### 3.1.2 Eureka Client

Eureka Client是一个java客户端，用于简化与Eureka Server的交互，客户端同时也就是一个内置的、使用轮询(round-robin)负载算法的负载均衡器。

配置如下：

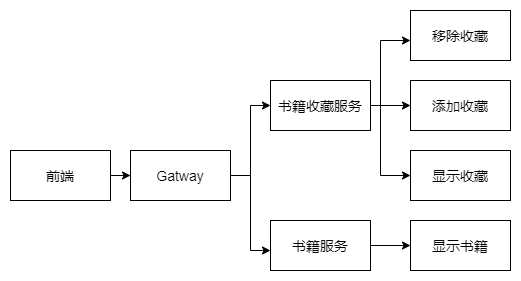


## 3.2 Gateway

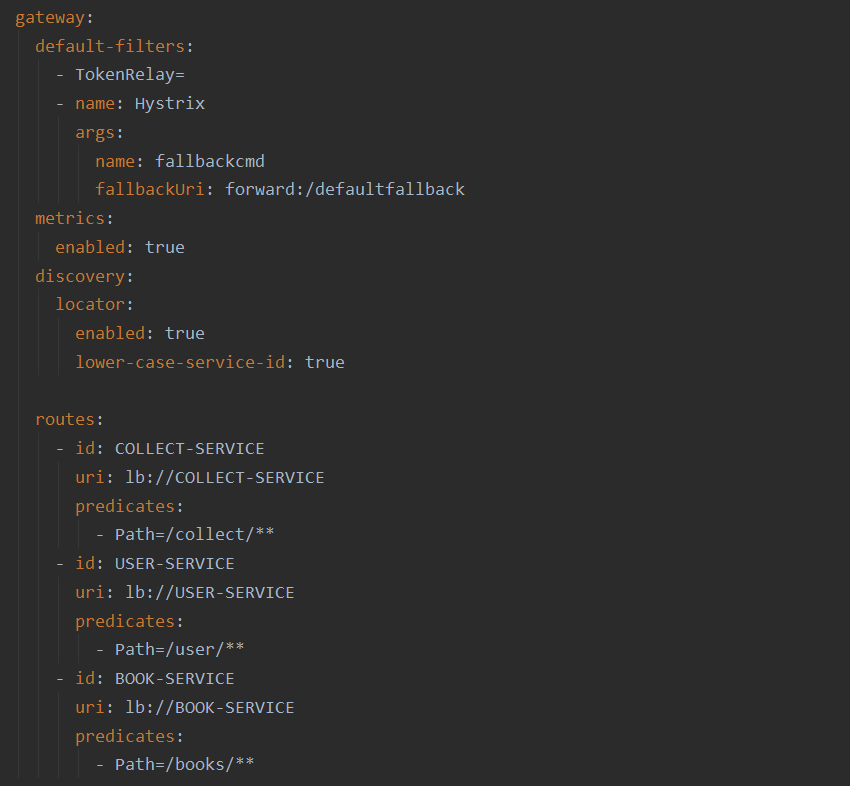
Spring Cloud Gateway 是 Spring Cloud 的一个全新项目，该项目是基于 Spring 5.0，Spring Boot 2.0 和 Project Reactor 等技术开发的网关，它旨在为微服务架构提供一种简单有效的统一的 API 路由管理方式。

### 3.2.1 Gateway\_service

提供网关服务，API网关是用户访问API的唯一入口，微服务架构通过引入微服务网关，将用户端与微服务的具体实现进行了解耦。



配置如图：



## 3.3业务服务

### 3.3.1 Book\_service

**BookController:**书本操作模块

getBooks()获得书籍列表

getBookByTitle(String title)获得特定书籍信息。

**Book**：里面包括book实体类以及赋值方法。

**BookService：**调用BookMapper接口，对书籍进行相关操作。

**BookMapper：**定义BookMapper接口给BookService。执行相应SQL语句向数据库中添加书籍信息，查找书籍信息，返回所有图书等。

### 3.3.2 Collect\_service:

**CollectController:**书籍收藏模块

getCollectionByUsername(String username)获得个人收藏信息

getCollection(String username, String title)查看书籍是否被收藏

addCollection(String username, String title)添加书籍收藏

deleCollection(String username, String title)删除书籍收藏

**Collection：**里面包括Collection实体类以及赋值方法

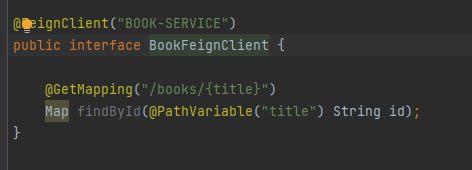
**CollectionService：**调用CollectionMapper接口，对书籍进行相关操作。

**CollectionMapper**：定义CollectionMapper接口给CollectionService，执行相应SQL语句向数据库中添加书籍收藏，通过用户名获取收藏信息，删除数据收藏记录。

## 3.4 Openfeign

OpenFeign是Spring Cloud在Feign的基础上支持了Spring MVC的注解。使用Feign调用Restful接口，feign内部基于Ribbon实现负载均衡，无需做特殊配置处理。在Spring Cloud Feign的实现下，只需创建一个接口并用注解的方式来配置它，即可完成对服务提供方的接口绑定，简化了在使用Spring Cloud Ribbon时自行封装服务调用客户端的开发量。  
定义BookFeignClient，接口@FeignClient使用fallbackFactory属性，可以操作异常信息，可操作性更强一些。在使用Spring注解来绑定具体该服务提供的REST接口。

代码实现如下：



之后在RestClientController来实现对Feign客户端的调用。使用@Autowired直接注入上面定义的BookFeignClient实例。

## 3.5 Hystrix

Hystrix是Spring Cloud提供的一种带有熔断机制的框架，由于在微服务系统中同一个操作会由多个不同的微服务来共同完成，所以微服务与微服务之间会由很多相互的调用，由于在分布式环境中经常会出现某个微服务节点故障的情况，所以会由调用失败发生，而熔断器的作用就是当出现远程调用失败的时候提供一种机制来保证程序的正常运行而不会卡死在某一次调用。

项目中Collect Service会调用Book Service来获取书籍的具体信息，当Book Service不可用时，需要进行熔断处理。在Application类添加@EnableCircuitBreaker注解，同时在Service里定义了collectionFallback()方法，并在需要熔断的方法上标注了@HystrixCommand(fallbackMethod = "collectionFallback")注解，从而在Book Service不可用时直接返回错误信息。

代码实现如图：

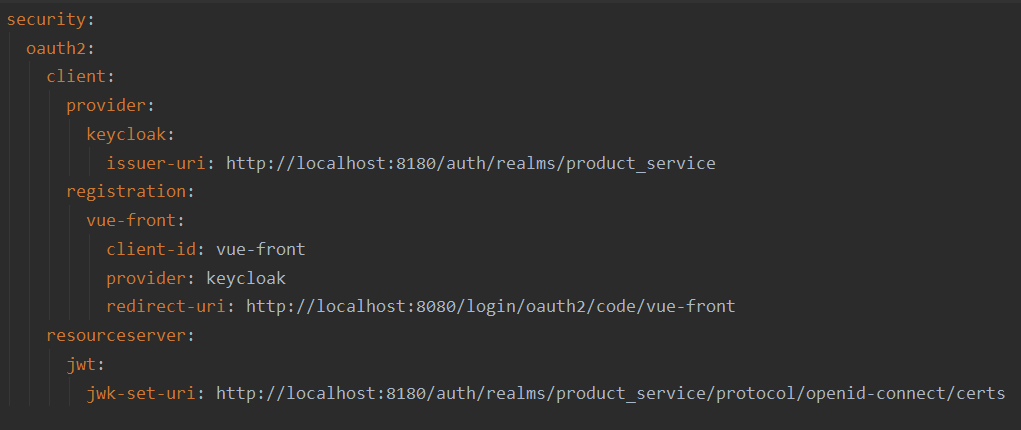


## 3.6 Keyclock

Keyclock基于OAuth2.0协议，实现鉴权和授权访问控制管理，是一个解决应用和服务身份验证与访问管理的开源工具。通过简单的配置即可达到保护应用和服务的目的。

**优点：**本应用不需要开发登录模块，验证用户和保存用户。Keycloak开发了用户管理，登录，注册，密码策略，安全问题，二步验证，密码重置等功能。登录，注册界面所需字段都是可配置，可自定义的。

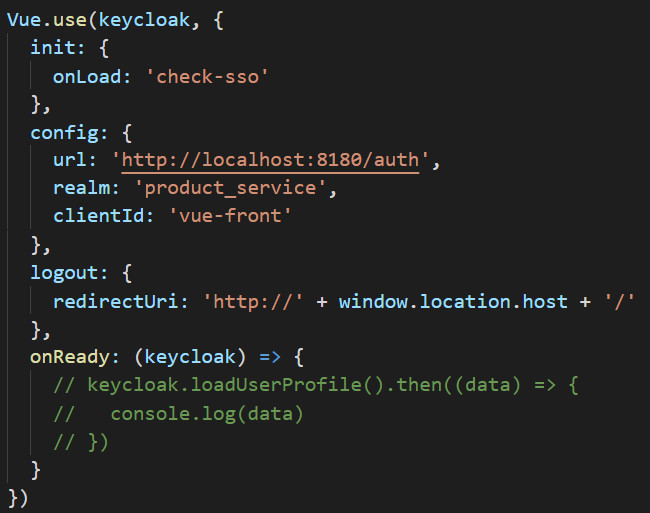
网关配置如下：



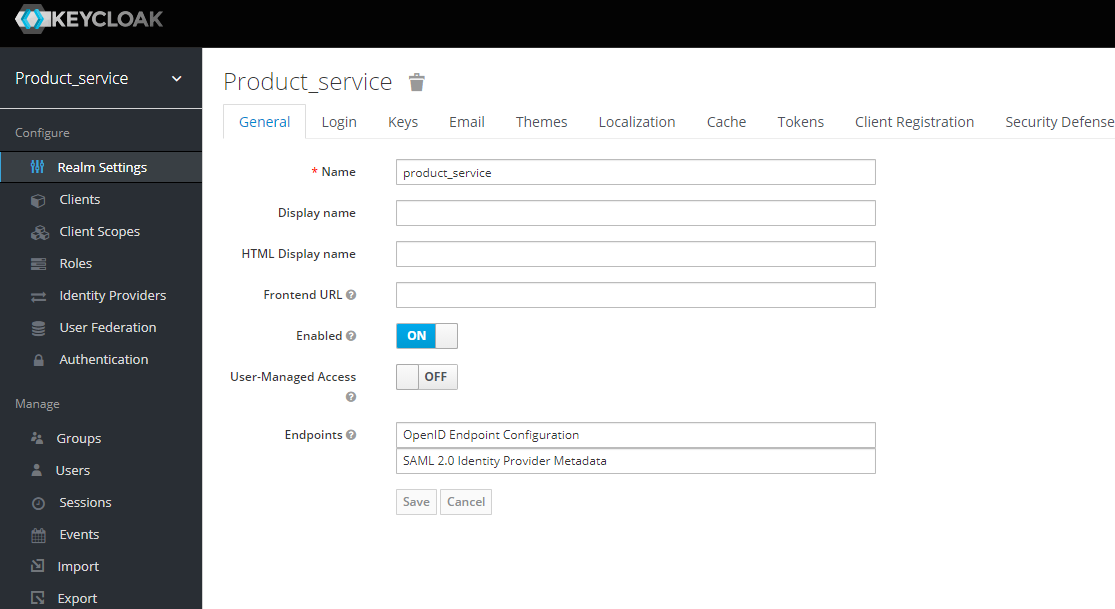
受保护的collect服务配置如下：



前端使用vue-keycloak.js进行配置如下：

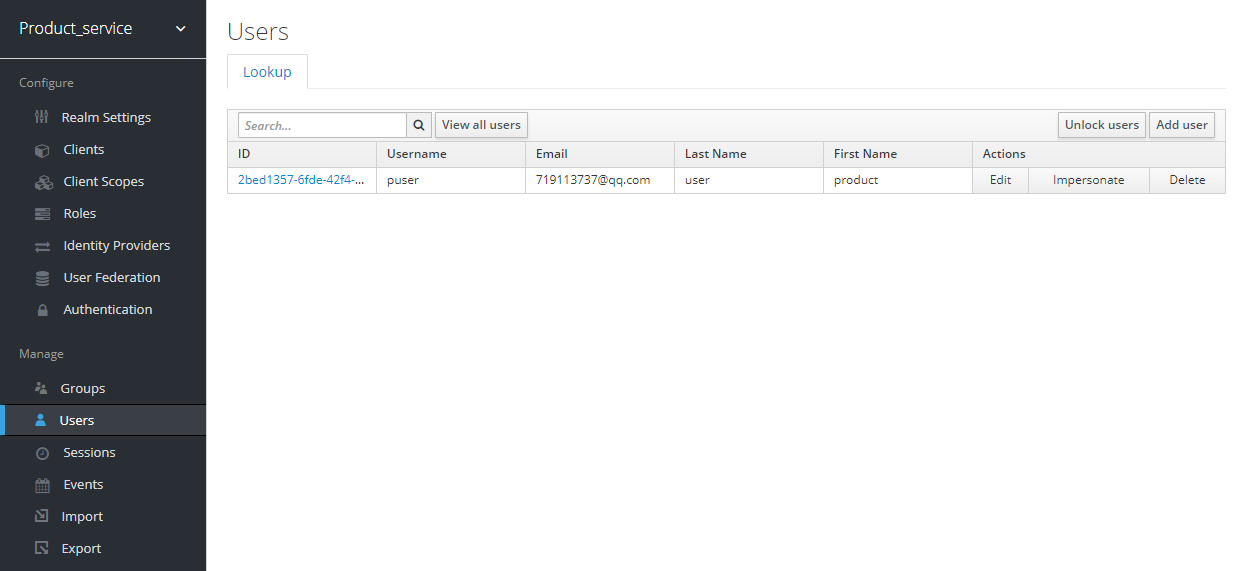


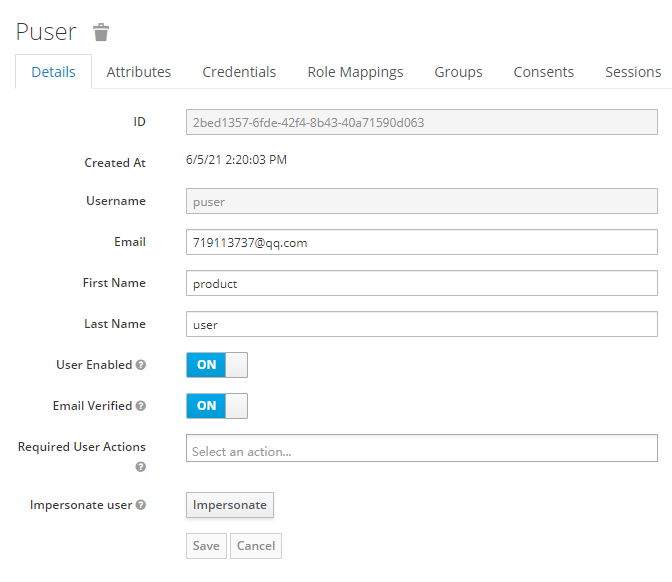
**后台管理界面（Admin Console）：**



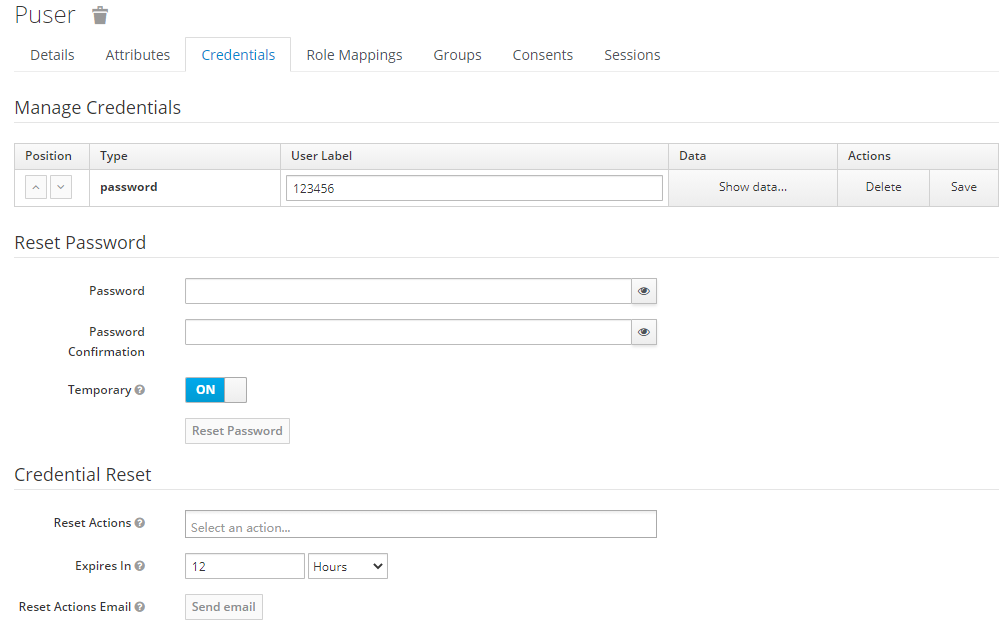
通过后台管理，管理员可以管理Keycloak所有方面的配置。可以开启关闭多种功能，配置身份代理和用户联盟。可以定义应用和服务，定义细粒的授权策略。管理用户，包括角色，权限，sessions等。

**账户管理界面（Account Management Console）:**





用户可以通过账户管理界面管理自己的信息，更新，设定二次验证。用户也可以管理自己的sessions。



在Credentials中也可以修改用户的密码。