****

详细设计规约

小组成员：1754130 文容智

1752210 黄安宁

1752262 龙宇

所在院系： 软件学院

学科专业： 软件工程

指导老师： 黄杰

目录

[1 引言 ２](#_Toc29526949)

[1.1 编写目的 ２](#_Toc29526950)

[1.2 系统包图 ３](#_Toc29526951)

[1.3 Presentation package ３](#_Toc29526952)

[1.4 数据库设计 ４](#_Toc29526953)

[1.5 模块设计 ５](#_Toc29526954)

[1.5.1 注册登录模块 ５](#_Toc29526955)

[1.5.2 博客管理模块 ６](#_Toc29526956)

[1.5.3 好友管理模块 ７](#_Toc29526957)

[2.3.4 用户管理和个人信息修改模块 ８](#_Toc29526958)

[2.3.5 评论管理模块 ９](#_Toc29526959)

[2.3.6 收藏夹管理模块 １０](#_Toc29526960)

[2.3.7 相册和图片管理模块 １１](#_Toc29526961)

[2 模块接口设计 １１](#_Toc29526962)

[2.1 注册登录模块 １１](#_Toc29526963)

[2.1.1 模块描述 １１](#_Toc29526964)

[2.1.2 功能 １１](#_Toc29526965)

[2.1.3 模块详细设计 １２](#_Toc29526966)

[2.2 博客管理 １６](#_Toc29526967)

[2.2.1 模块描述 １６](#_Toc29526968)

[2.2.2 功能 １６](#_Toc29526969)

[2.2.3 模块详细设计 １７](#_Toc29526970)

[2.3 好友管理模块 ２９](#_Toc29526971)

[2.3.1 模块描述 ２９](#_Toc29526972)

[2.3.2 功能 ２９](#_Toc29526973)

[2.3.3 模块详细设计 ３０](#_Toc29526974)

[2.4 用户管理和个人信息修改模块 ３７](#_Toc29526975)

[2.4.1 模块描述 ３７](#_Toc29526976)

[2.4.2 功能 ３７](#_Toc29526977)

[2.4.3 功能详细设计 ３８](#_Toc29526978)

[2.5 评论管理模块 ４６](#_Toc29526979)

[2.5.1 模块描述 ４６](#_Toc29526980)

[2.5.2 功能 ４６](#_Toc29526981)

[2.5.3 功能详细设计 ４６](#_Toc29526982)

[2.6 收藏夹管理 ５０](#_Toc29526983)

[2.6.1 模块描述 ５０](#_Toc29526984)

[2.6.2 功能 ５０](#_Toc29526985)

[2.6.3 功能详细设计 ５０](#_Toc29526986)

[2.7 相册和图片管理模块 ５６](#_Toc29526987)

[2.7.1 模块描述 ５６](#_Toc29526988)

[2.7.2 功能 ５６](#_Toc29526989)

[2.7.3 功能详细设计 ５７](#_Toc29526990)

[3 模块详细设计 ６３](#_Toc29526991)

[3.1 注册登录模块 ６３](#_Toc29526992)

[3.1.1 类的设计和接口实现 ６４](#_Toc29526993)

[3.1.2 说明： ６７](#_Toc29526994)

[3.2 博客管理模块 ６８](#_Toc29526995)

[3.2.1 类的设计和接口的实现： ６８](#_Toc29526996)

[3.2.2 说明 ７５](#_Toc29526997)

[3.3 好友管理模块 ７６](#_Toc29526998)

[3.3.1 类的设计和接口的实现 ７６](#_Toc29526999)

[3.3.2 说明 ７９](#_Toc29527000)

[3.4 用户管理和个人信息模块 ７９](#_Toc29527001)

[3.4.1 类的说明和接口的实现 ８０](#_Toc29527002)

[3.4.2 说明 ８２](#_Toc29527003)

[3.5 评论管理模块 ８３](#_Toc29527004)

[3.5.1 类的设计和接口的实现 ８３](#_Toc29527005)

[3.6 收藏夹管理模块 ８６](#_Toc29527006)

[3.6.1 类和详细接口设计 ８６](#_Toc29527007)

[3.7 相册和图片管理模块 ９０](#_Toc29527008)

[3.7.1 类和详细接口设计 ９０](#_Toc29527009)

[3.7.2 说明： ９３](#_Toc29527010)

# 引言

## 编写目的

《详细设计书》是项目设计的第二个阶段，这个阶段的主要任务是在《概要设计书》基础上，对概要设计中产生的功能模块进行过程描述，设计功能模块的内部细节，包括类的详细逻辑和接又及类之间的调用关系，为编写源代码提供必要的说明。

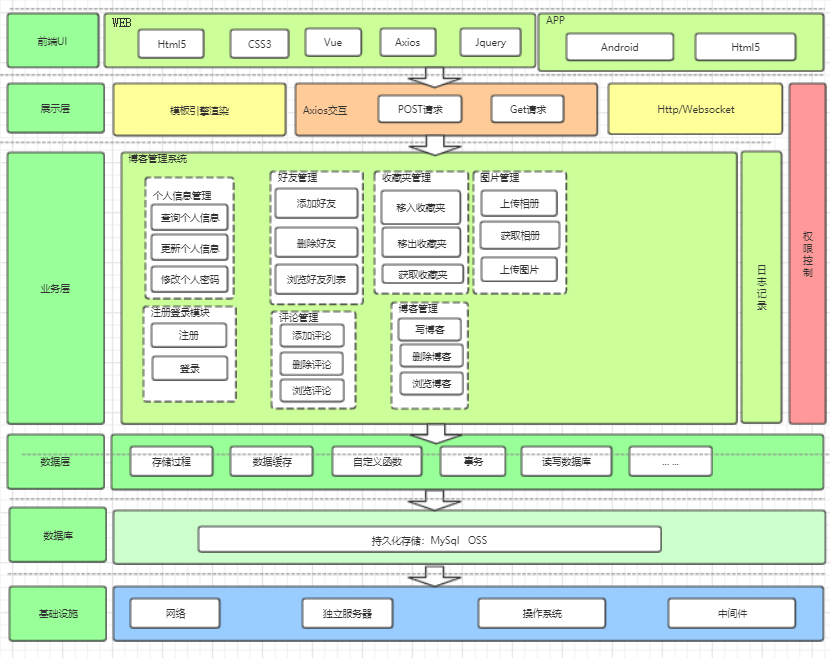
从该阶段开发正式进⼊软件的实际开发阶段本阶段完成系统的大致设计并明确系统的数据结构与软件结构。在软件设计阶段主要是把一个软件需求转化为软件表示的过程，这种表示只是描绘出软件的总的概貌。

本文档的目的旨在推动软件工程的规范化，使设计人员遵循统⼀的详细设计书写规范，节省制作文档的时间，降低系统实现的风险，做到系统设计资料的规范性与全面性，以利于系统的实现、测试、维护、版本升级等。也就是进⼀步细化软件设计阶段得出的软件总体概貌把它加工成在程序细节上⾮常接近于源程序的软件表示。

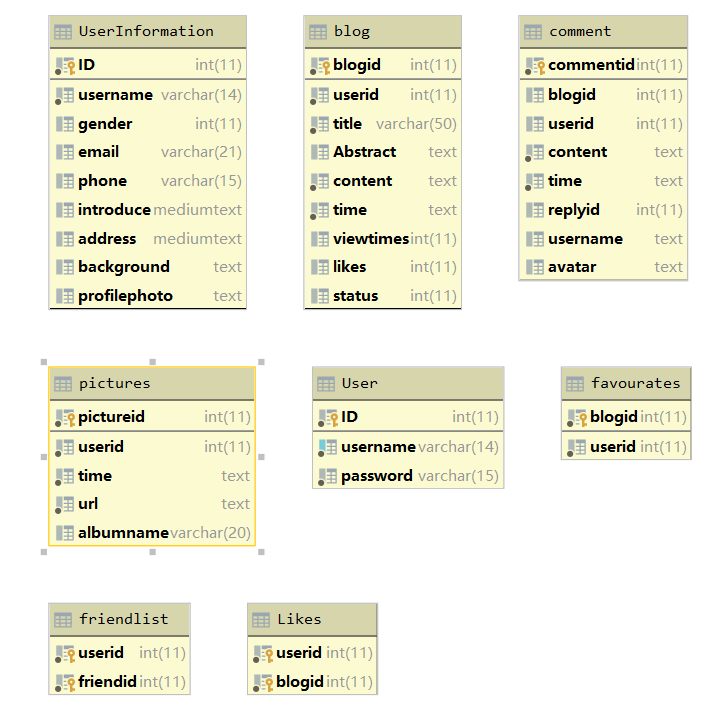
概要设计解决了软件系统总体结构设计的问题，包括整个软件系统的结构、模块划分、模块功能和模块间的联系等。详细设计则要解决如何实现各个模块的内部功能，即模块设计。具体的说，模块设计就是要详细描述各模块中包含的类的处理逻辑及其提供接又的参

数格式、请求方式。但这并不等同于系统实现阶段用具体的语言编码，它只是对实现细节作精确的描述，这样编码阶段就可以将详细设计中对功能实现的描述，直接翻译、转化为用某种程序设计语言书写的程序。详细设计的详细程度，应达到可以编写程序的水平。

## 系统架构

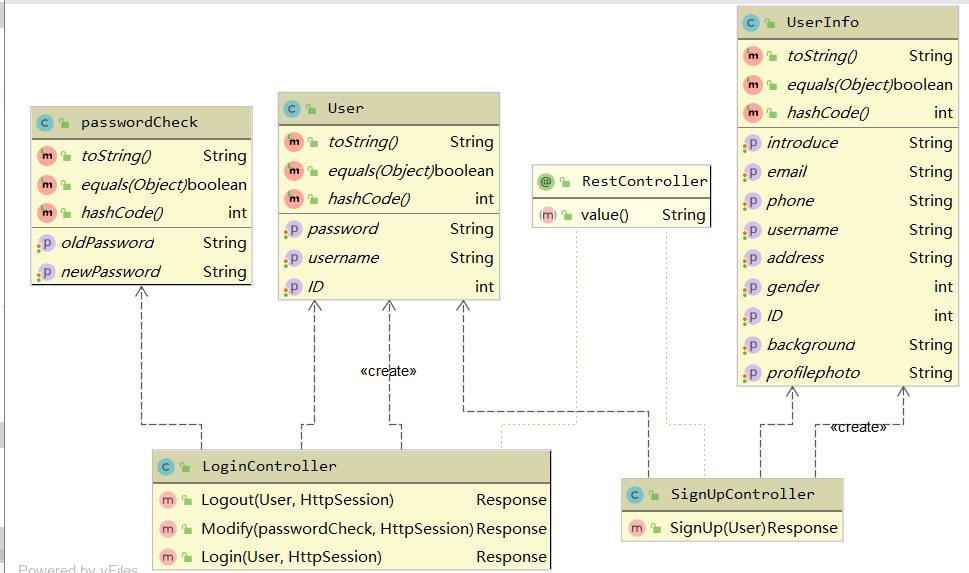


## 数据库设计



## 模块设计

### 注册登录模块

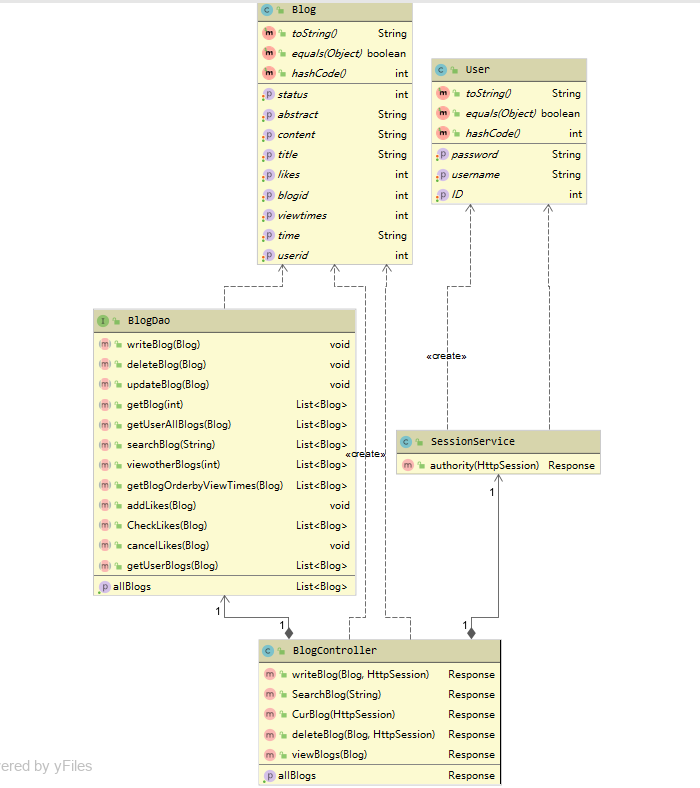


第一次使用平台的用户需要先注册，注册成功之后系统会跳转至登录页面，用户通过有效的用户名和密码登录系统，登录之后跳转至首页。

用户在个人页面查看自己的个人信息需要通过系统从数据库与区块链中获取保存在其中的信息。修改个人信息同样也需要通过系统将修改后的信息提交到数据库以及区块链。

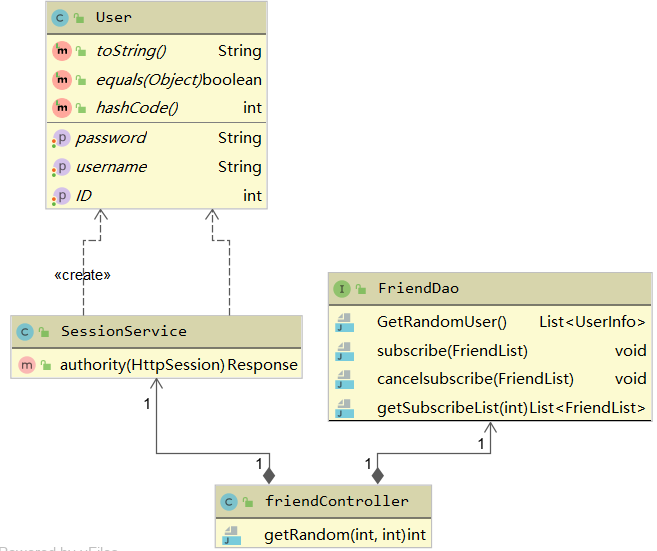
以上行为都必须与数据库以及区块链进行交互以获得以及更改对应的用户个人信息。类之间的调用关系如下图：

### 博客管理模块



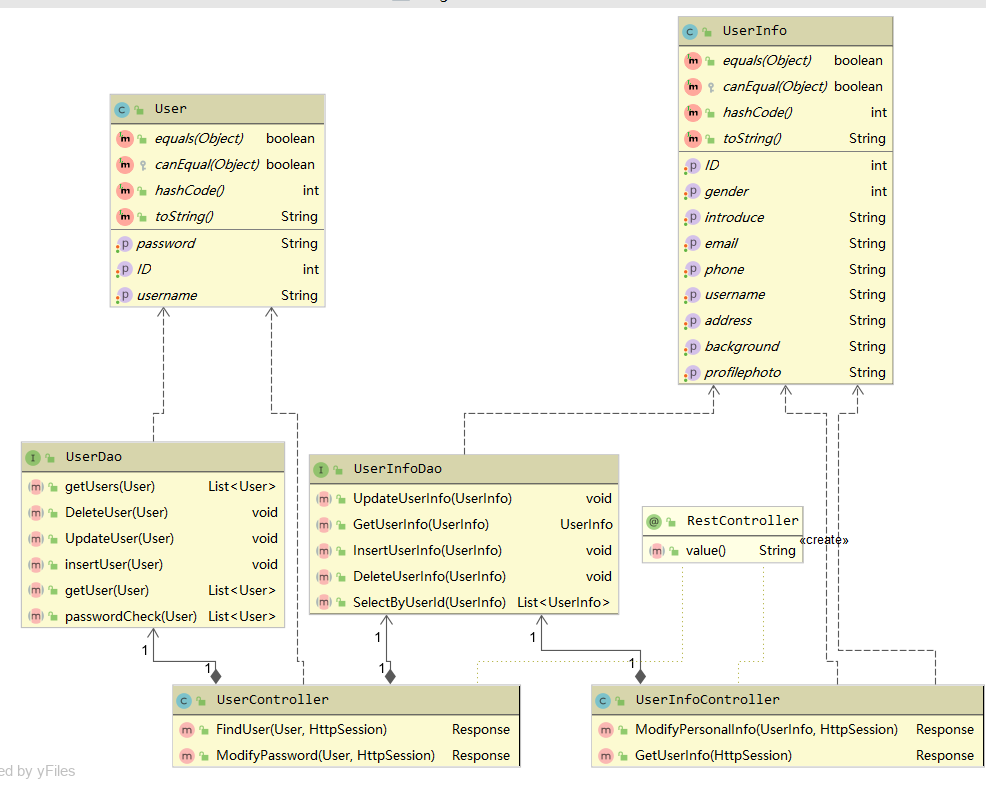
用户可以使用本模块进行对博客的基本操作，例如增删改查，访问其他用户的博客，同时可以对博客进行点赞和收藏。

### 好友管理模块



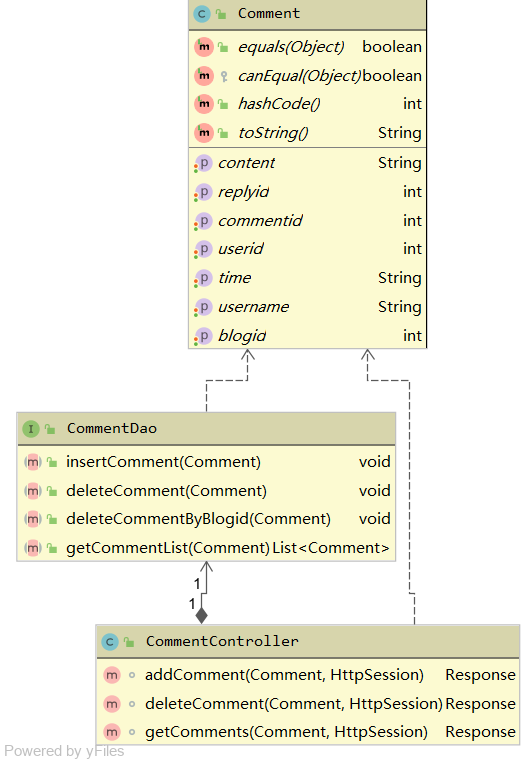
用户可以通过此模块进行对好友列表的增删改查，获取好友的个人信息，查看好友的博客等功能。

### 2.3.4 用户管理和个人信息修改模块



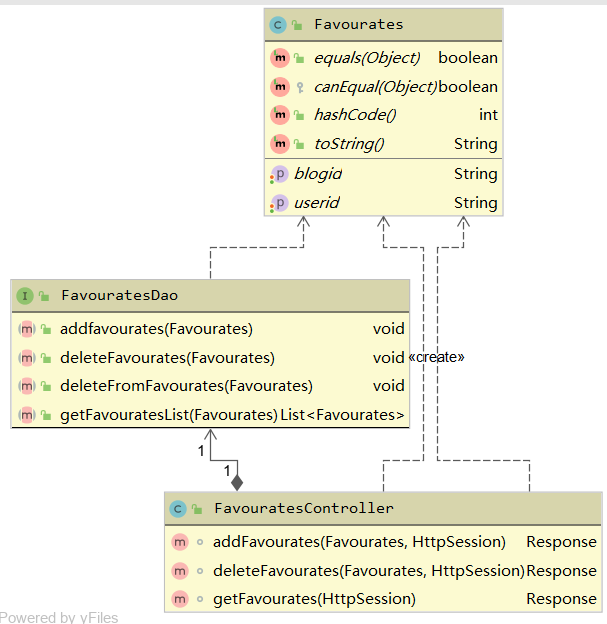
此模块为用户和管理员设计，用户可以通过此模块来修改个人信息以及密码。管理员可通过此模块对用户进行管理。

### 2.3.5 评论管理模块



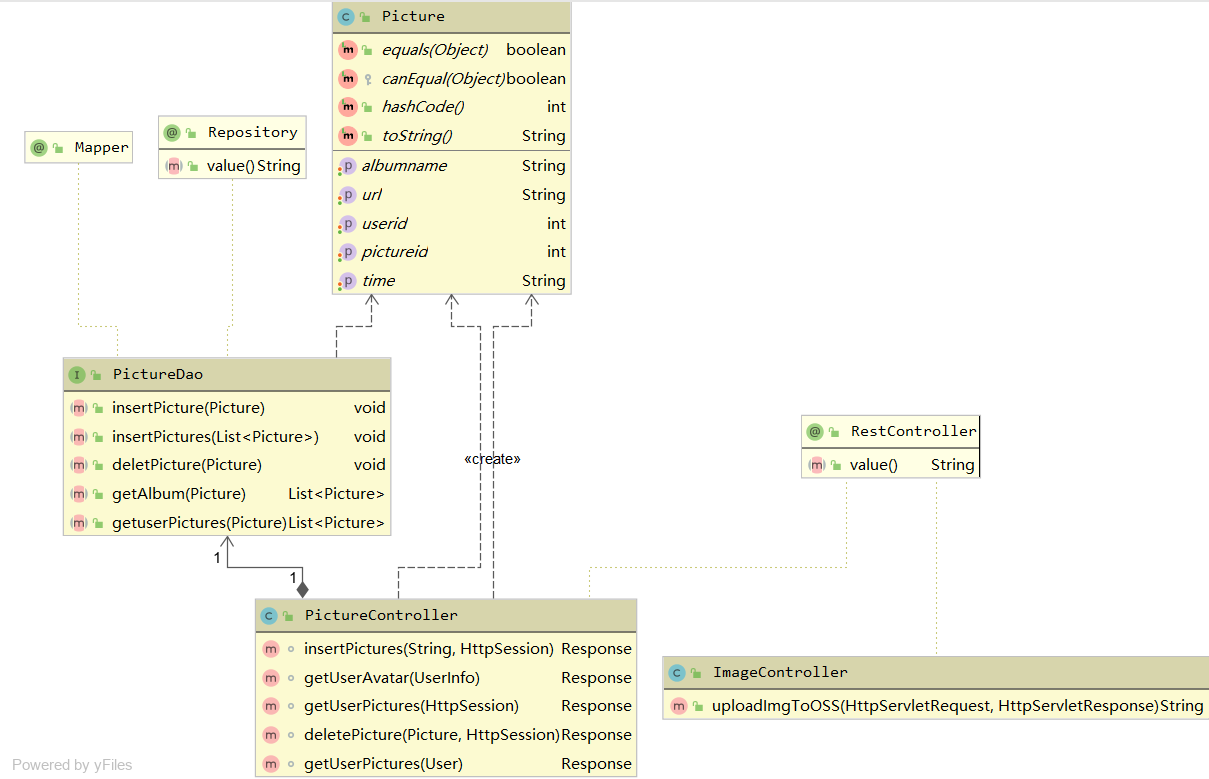
此模块为用户提供评论和留言功能，用户可以发表和删除自己的评论，管理员可以调用这些接口进行评论管理。

### 2.3.6 收藏夹管理模块



此模块提供给用户收藏夹的功能，用户可以收藏博客文章，对收藏夹进行增删改查。

### 2.3.7 相册和图片管理模块



本模块实现了相册的上传，删除个人相簿照片的功能，同时用户能访问他人相册。图片管理模块实现了对博客中上传头像的存储。

# 模块接口设计

## 注册登录模块

### 模块描述

用户使用此平台需要注册，注册后进行登录进入个人主页。

### 功能

1. 注册
2. 登录

### 模块详细设计

#### 用户登录详细设计

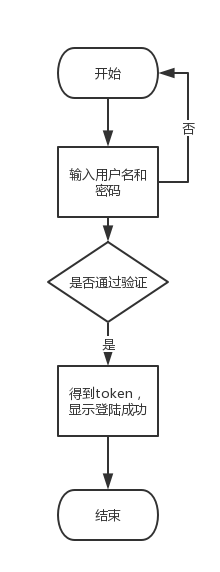
##### 输入项

1. 用户名
2. 用户密码

##### 输出项

1. 是否登陆成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/user/login |
| 路径参数/描述 | 请求参数1 username: 用户名  请求参数2 password: 用户密码 |
| 参数类型 | username: String  password: String |
| 描述 | 根据用户传入的参数，判断用户是否存在，如果不存在则将用户信息记录，返回“success”信息，如果失败返回失败相关信息。 |
| 返回值类型 | Json |
|  |  |

##### 限制条件及出错处理

1、用户密码必须为六位以上复杂密码，两次输入的密码必须一致，若不一致则弹窗提示用户密码不一致。

2、用户名未经注册，返回未找到此账号信息。

#### 用户注册详细设计

##### 输入项

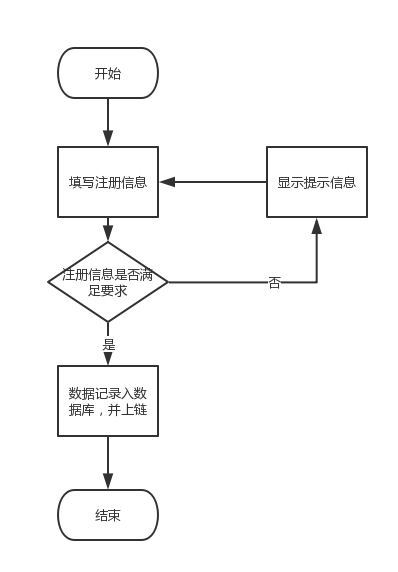
1. 用户姓名
2. 用户密码
3. 用户密码（输⼊两次密码，确认两次必须相同）

##### 输出项

1）用户注册成功

2）用户注册失败

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/user/SignUp |
| 路径参数/描述 | 请求参数1 username: 用户名  请求参数2 password: 用户密码 |
| 参数类型 | username: String  password: String |
| 描述 | 根据用户传入的参数，判断用户名密码是否正确，如果登录成功，返回“success”信息，如果失败返回失败相关信息。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1.若已存在相同用户名，则返回此用户名已注册。

## 博客管理

### 模块描述

本模块中，用户可以用来进行对博客的增删改查，访问他人博客，

### 功能

1. 写博客
2. 删除博客
3. 返回当前用户博客信息
4. 博客的关键字查询
5. 获取所有博客
6. 下线兼职信息

### 模块详细设计

#### 写博客详细设计

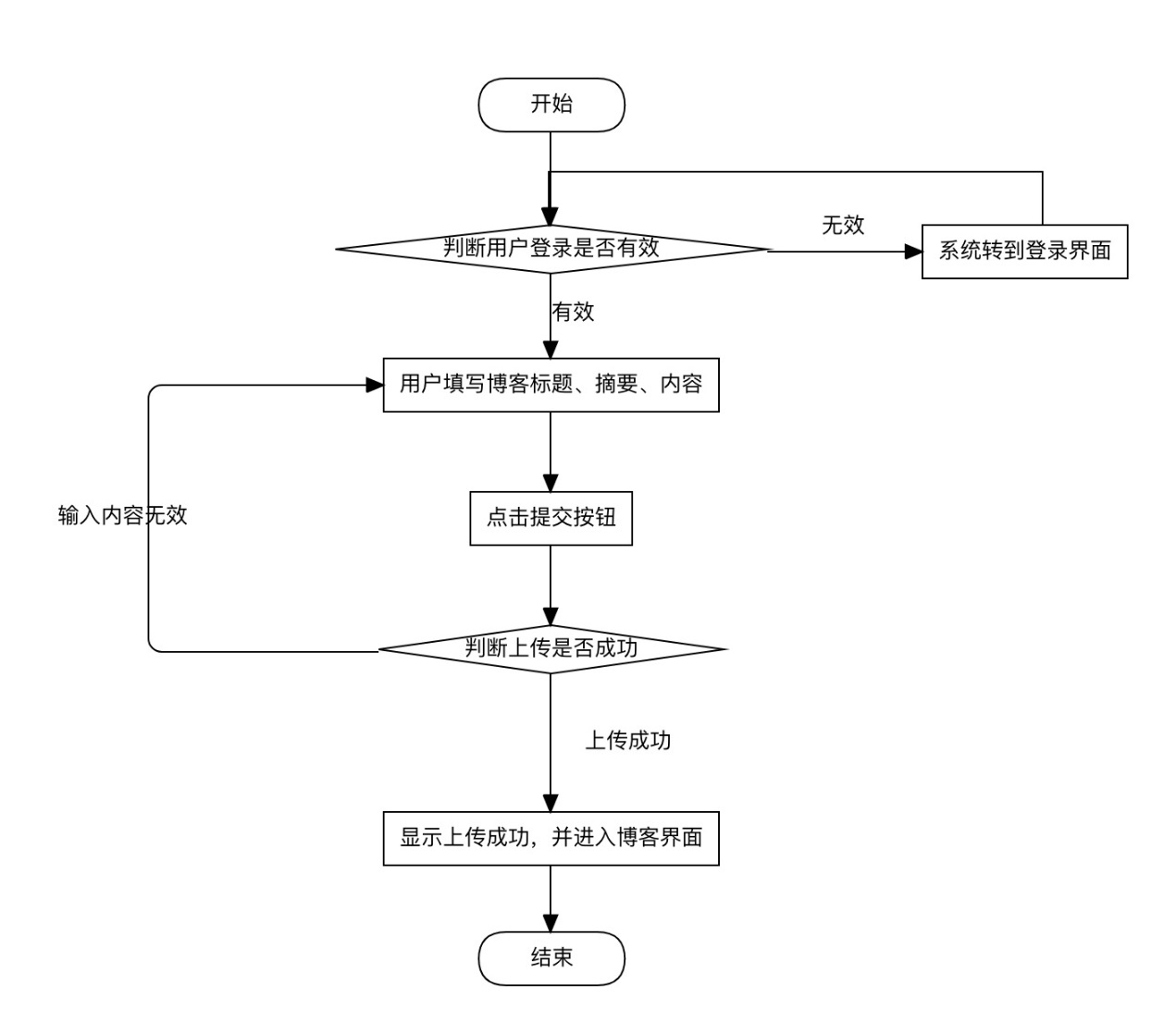
##### 输入项

1. 标题（title）
2. 摘要（Abstract）
3. 内容（content）

##### 输出项

1）是否发布成功

##### 逻辑流程



##### 接口设计（用户）

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | * http://[host]:8080/blog/write |
| 路径参数/描述 | 请求参数1 Title: 博客标题  请求参数2 Abstract: 博客摘要  请求参数3 Content: 博客内容 |
| 参数类型 | Title: String  Abstract:String  Content:String |
| 描述 | 用户上传博客的标题和摘要和博客内容，进行数据格式验证后上传数据库，返回查询信息。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1. 博客的标题在30字内，摘要在50字内，博客文本不超过65535字节。
2. 博客内容以富文本的形式存储。

#### 删除博客

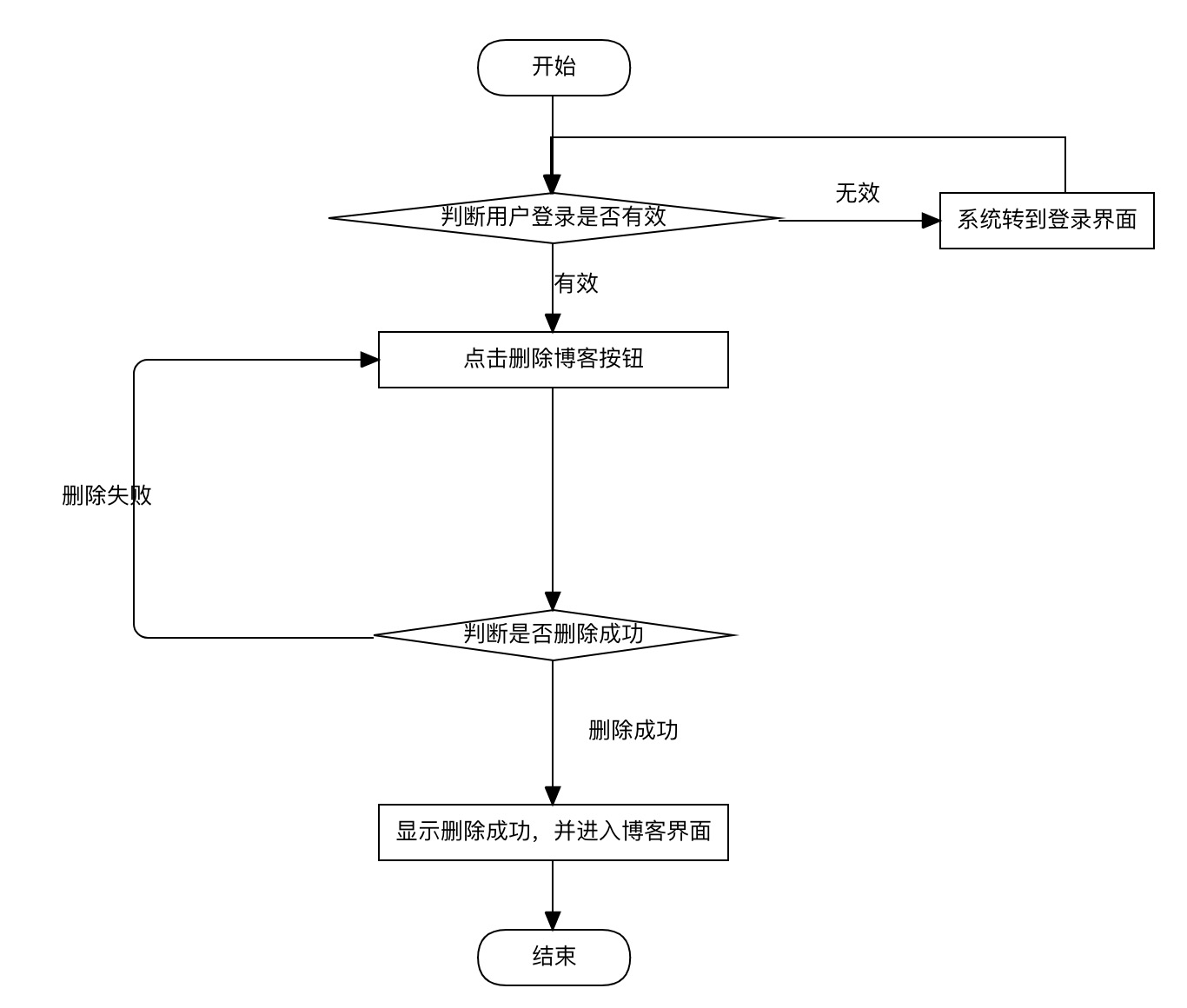
##### 输入项

1. 无

##### 输出项

1）是否删除成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/blog/delete |
| 路径参数/描述 | Blogid:博客ID |
| 参数类型 | Blogid:int |
| 描述 | 用户点击博客的删除按钮触发此接口，从数据库中删除此博客的信息，包括评论和收藏夹里关于此博客的信息。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1. 若博客不存在返回404。

#### 返回当前用户的博客信息

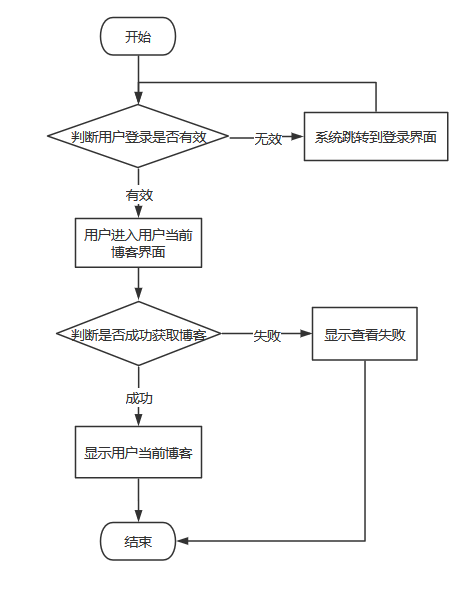
##### 3.2.3.3.1输入项

无

##### 3.2.3.3.2输出项

* 1. 当前用户的所有博客列表

##### 3.2.3.3.3逻辑流程



##### 3.2.3.3.4接口设计

|  |  |
| --- | --- |
| 请求方式 | GET |
| 服务路径 | http://[host]:8080/blog/getcur |
| 路径参数/描述 | 无 |
| 参数类型 | 无 |
| 描述 | 此接口会根据当前session中的“id”字段获取当前用户的id，然后到数据库里查找到此用户对应的全部博客文章然后返回 |
| 返回值类型 | Json |
| 示例 |  |

##### 3.2.3.3.5限制条件及出错处理

1、此功能模块会进行session验证，若用户未登录或者登录信息超时，会提示当前未登录。返回登录界面。

#### 获取给定id用户的所有博客（查看任意用户博客）

##### 3.2.3.4.1输入项

无

##### 3.2.3.4.2输出项

* 1. 给定id的用户的所有博客

##### 3.2.3.4.3 逻辑流程

##### 3.2.3.4.4 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/blog/view |
| 路径参数/描述 | 请求参数： userId: |
| 参数类型 | userID: String |
| 描述 | 用户点击进入别人的个人主页，点击他人个人主页的博客列表，查看他人所有博客。 |
| 返回值类型 | Json |
| 示例 |  |

#### 根据关键字查询博客内容

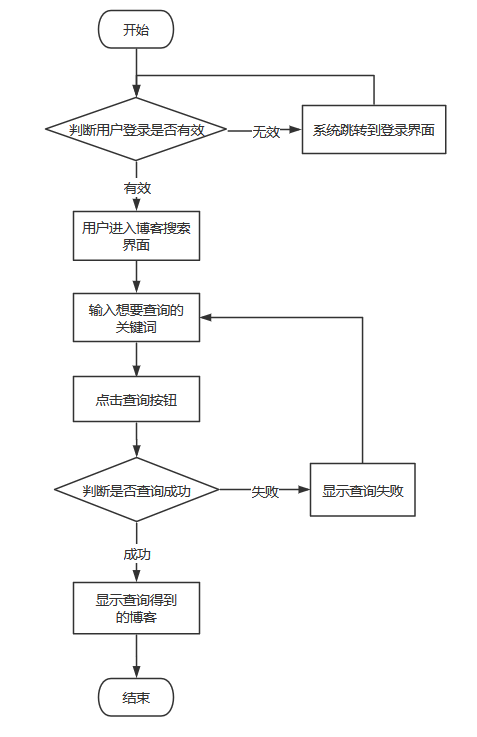
##### 输入项

搜索内容（查询关键字）

##### 输出项

搜索到的所有博客

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 |  |
| 服务路径 | http://[host]:8080/blog/search |
| 路径参数/描述 | 请求参数1 SearchString:查询关键字 |
| 参数类型 | searchString |
| 描述 | 用户在输入框中输入查询关键字，返回给用户标题或摘要或内容包含关键字的博客列表。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1. 关键字在30字内

#### 给博客点赞

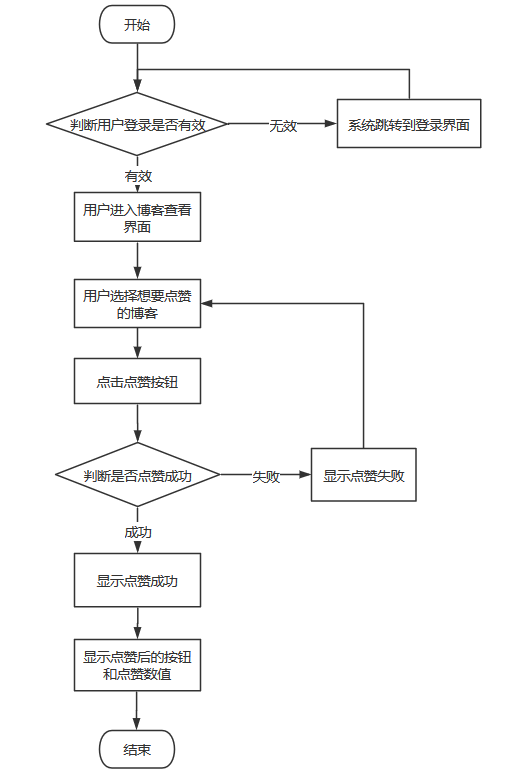
##### 输入项

无

##### 输出项

无

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/blog/likes |
| 路径参数/描述 | 无 |
| 参数类型 | 无 |
| 描述 | 用户点击点赞按钮，博客点赞数+1 同时将博客点赞数更新到数据库，并刷新此时的点赞数和点赞状态。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

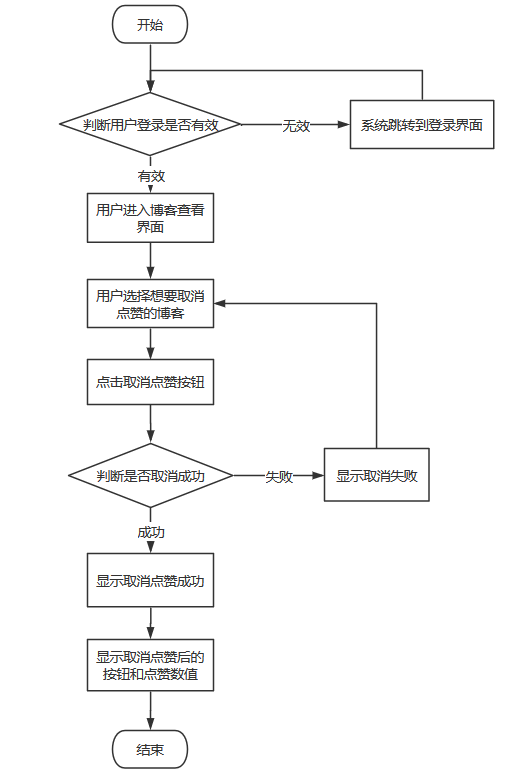
1.若用户已经给此博客点赞，此博客对当前用户显示点赞后的图标并更新点赞数，此时若再次点击点赞图标，则会触发取消点赞函数

#### 取消点赞

##### 输入项 无

##### 输出项 无

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/blog/cancellikes |
| 路径参数/描述 | 无 |
| 参数类型 | 无 |
| 描述 | 用户点赞后再次点击点赞按钮，此时会取消点赞，博客点赞数-1 同时将博客点赞数更新到数据库，并刷新此时的点赞数和点赞状态。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件以及出错处理

1.用户赞数为0的时候不会出现取消点赞按钮，即使出现Bug导致此时出现了取消点赞按钮，点击后也只会将点赞数清零，并恢复到“此用户并未对该博客点赞的状态”。

## 好友管理模块

### 模块描述

本模块给用户提供添加好友到关注列表，删除好友列表里的好友，获取好友列表的功能。此外还提供了随机推荐好友列表的接口。

### 功能

1. 添加好友
2. 删除好友列表中的好友
3. 获取好友推荐列表
4. 获取当前用户的好友列表
5. 获取指定id用户的好友列表

### 模块详细设计

#### 添加好友

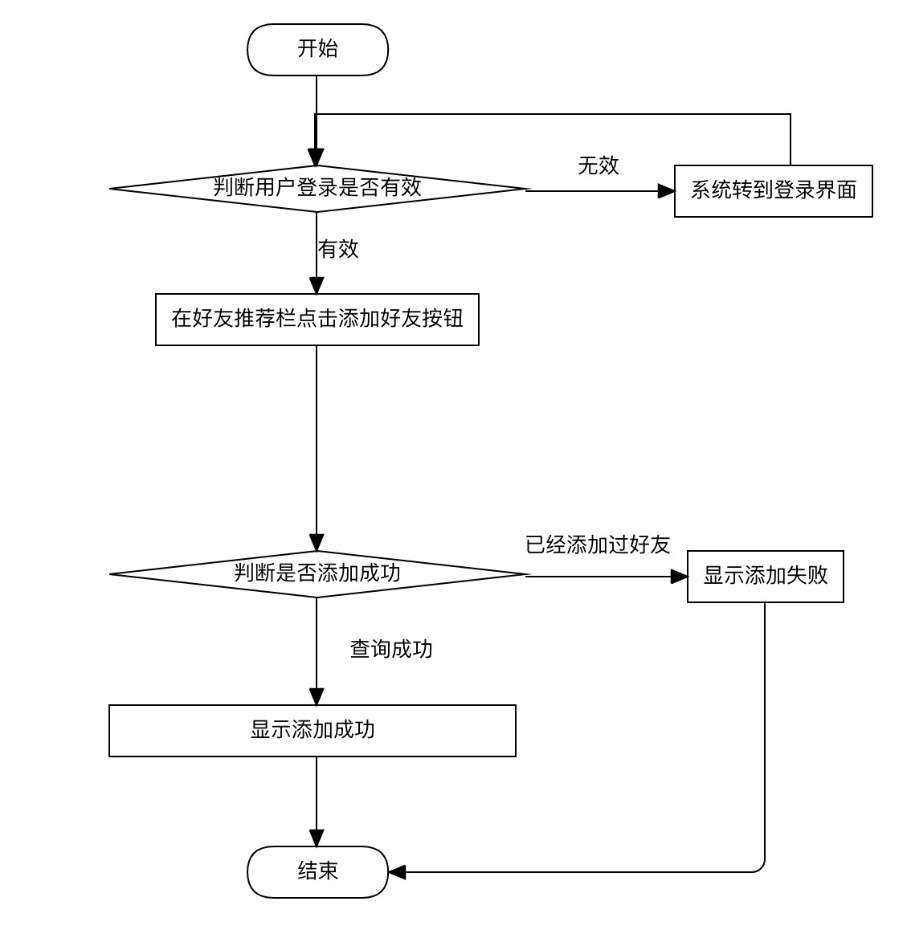
##### 输入项

无

##### 输出项

显示添加成功与否的信息

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/friend/subscribe |
| 路径参数/描述 | friendID(好友ID) |
| 参数类型 | friendID:int |
| 描述 | 当一个用户出现在随机关注列表里时，可以点击添加按钮添加到好友列表中。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1. 若用户已经在好友列表中，则会返回500，此时前端会弹出窗口：此用户已在好友列表中
2. 使用此功能的用户必须已登录且登录状态未过期。

#### 删除好友列表中的好友

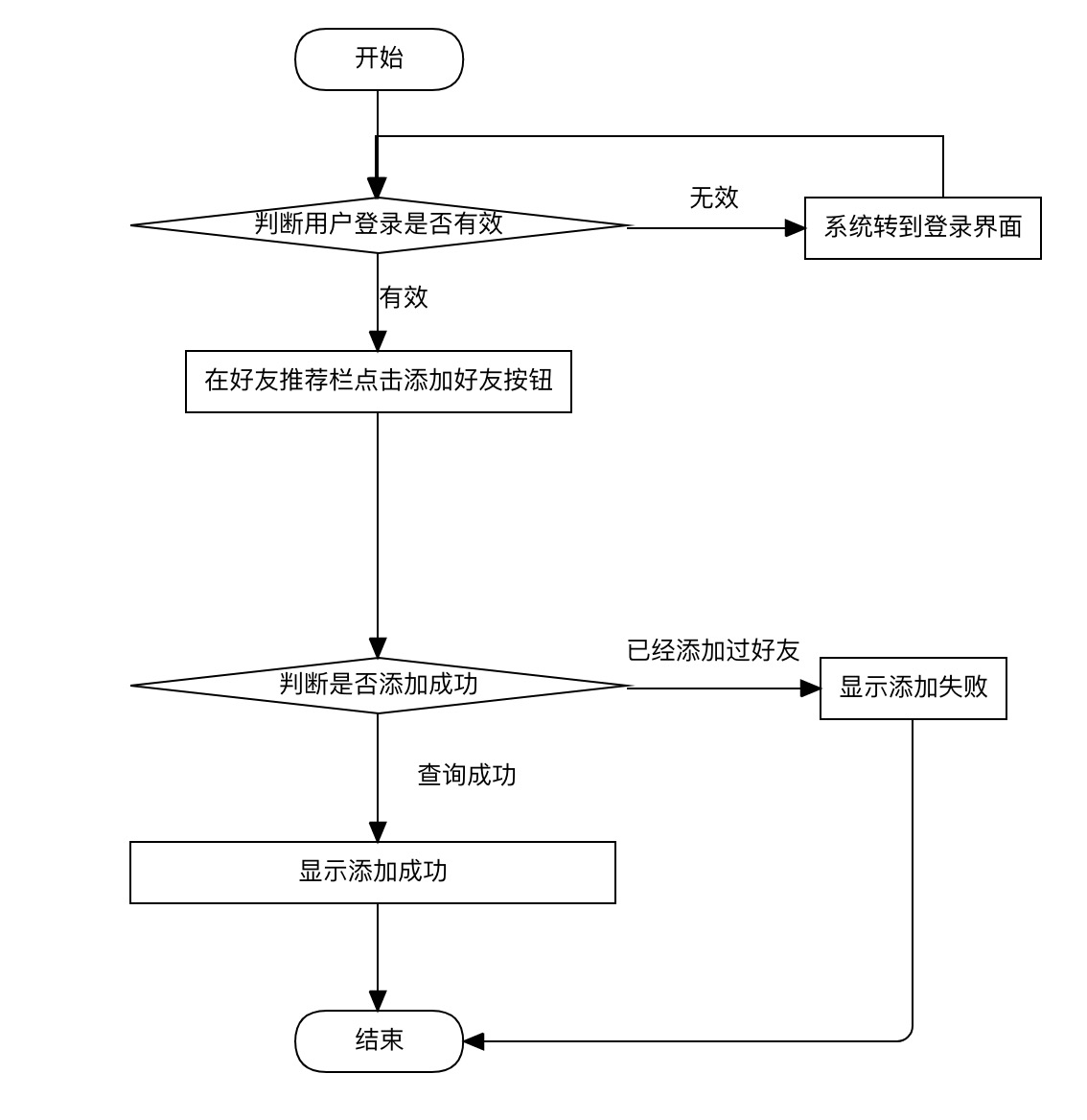
##### 输入项

无

##### 输出项

1. 显示删除成功与否的信息

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | DELETE |
| 服务路径 | http://[host]:8080/friend/delete |
| 路径参数/描述 | friendID(好友ID) |
| 参数类型 | friendID:int |
| 描述 | 用户点击好友列表中的删除按钮触发，删除列表中此好友并更新到数据库，随后返回成功与否并刷新页面。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1、使用此功能的用户必须已登录。

#### 获取好友推荐列表

##### 输入项

无

##### 输出项

一个推荐好友的列表

##### 逻辑流程

##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | GET |
| 服务路径 | http://[host]:8080/friend/recommend |
| 路径参数/描述 | 无 |
| 参数类型 | 无 |
| 描述 | 用户的个人主页右侧会出现好友的推荐列表，可以点击刷新键获取一个新的列表。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件以及出错处理

1、用户已登录，若用户的登录信息过期会跳转到登录页面。

#### 获取当前用户的好友列表

##### 输入项

无

##### 输出项

1. 返回当前用户的好友列表

##### 逻辑流程

##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | GET |
| 服务路径 | http://[host]:8080/friend/getList |
| 路径参数/描述 | 无 |
| 参数类型 | 无 |
| 描述 | 用户点击主页上的好友列表按钮，返回当前用户的好友列表并输出。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1、用户已登录，若用户的登录信息过期会跳转到登录页面。

#### 获取指定id的用户的好友列表（查看他人好友列表）

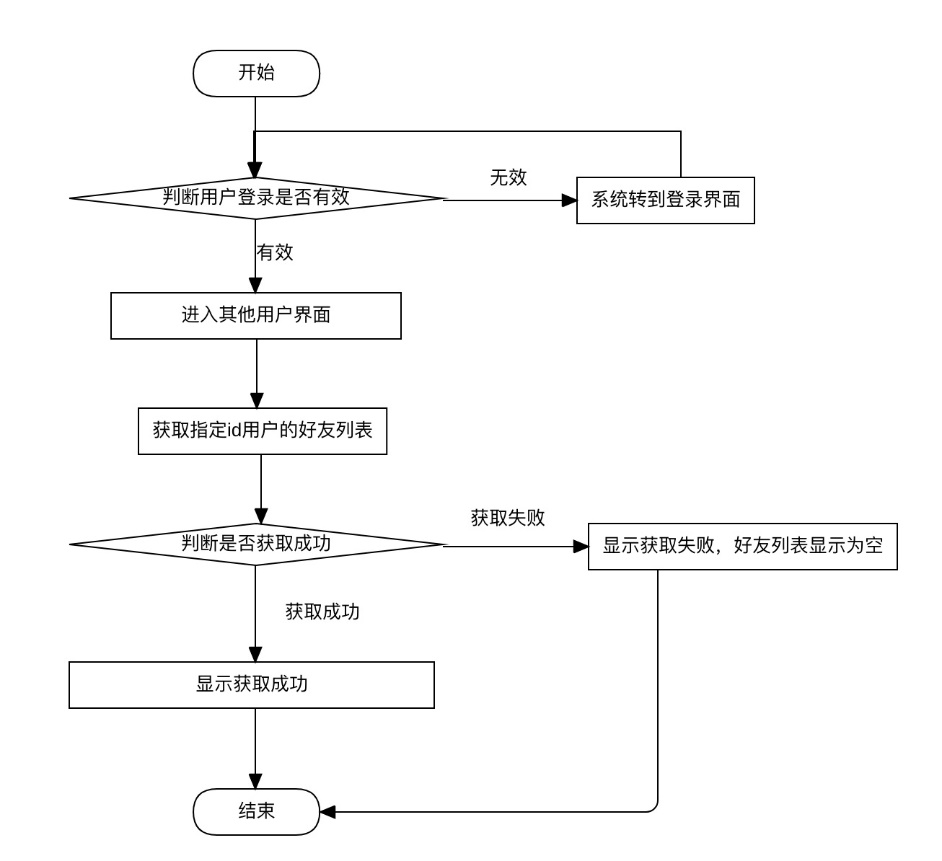
##### 输入项

无

##### 输出项

1. 显示指定id的用户的好友列表

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/friend/getFriends |
| 路径参数/描述 | 请求参数：userid(获取的用户的id) |
| 参数类型 | Userid: Int |
| 描述 | 用户在进入他人个人主页时，点击他的好友列表会获取到他人的好友列表。 |
| 返回值类型 | Json |
| 示例 |  |

##### 限制条件及出错处理

1、用户已登陆

## 用户管理和个人信息修改模块

### 模块描述

此模块提供给用户更改个人信息，修改密码。以及管理员和用户对用户的查询和提供给管理员的删除功能。

### 功能

1. 修改个人信息
2. 修改密码
3. 查询用户
4. 删除用户

### 功能详细设计

#### 修改个人信息

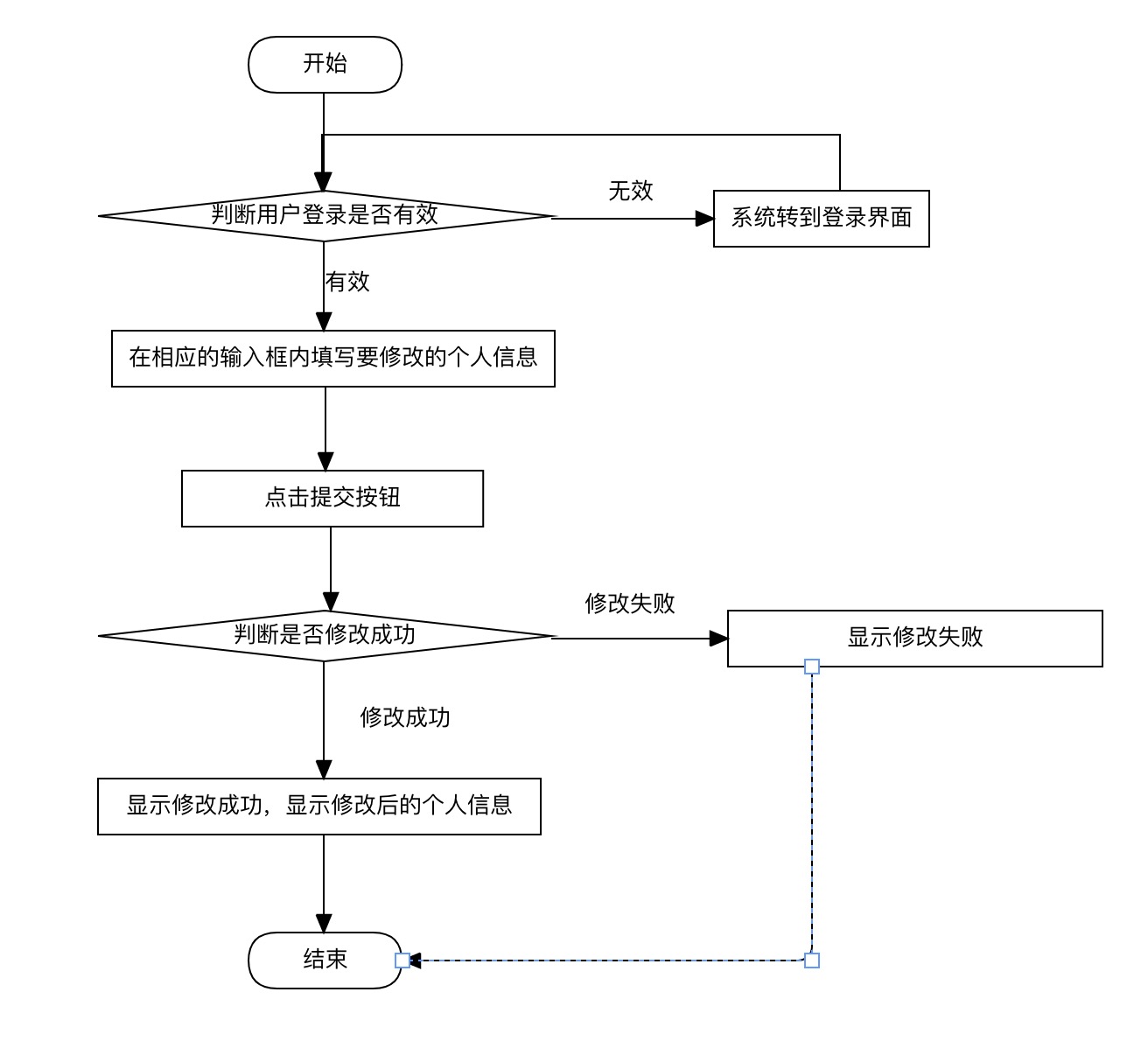
##### 输入项

1. 姓名
2. 性别（选择）
3. Email
4. 电话号码
5. 介绍
6. 地址

##### 输出项

修改是否成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/userInfo/modify |
| 路径参数/描述 | 选择参数1：姓名  选择参数2：性别（选择）  选择参数3：Email  选择参数4：电话号码  选择参数5：介绍  选择参数6：地址 |
| 参数类型 | 姓名：String  性别：int  Email:String  电话号码：String  介绍：String  地址：String |
| 描述 | 用户点击修改个人信息按钮进入个人信息修改界面。输入修改信息后 |
| 返回值类型 | Json |

##### 限制条件及出错处理：

1. 在使用此功能前，用户必须先登录。
2. 输入的信息必须合法（姓名必须在20个字符以内，电话号码不能出现字母，介绍和地址不超过200字）

#### 修改密码

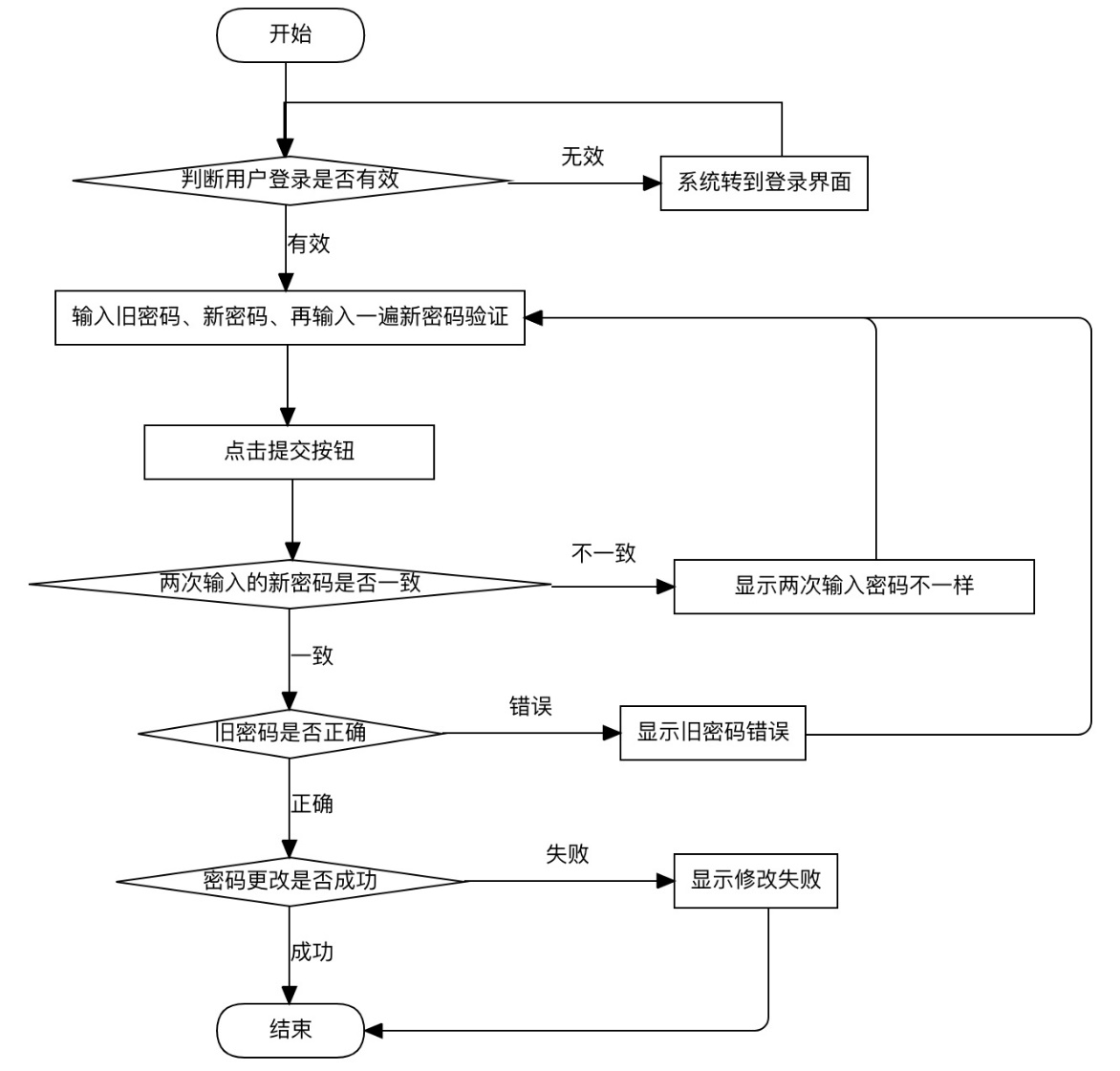
##### 输入项

旧密码和新密码

##### 输出项

返回修改是否成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/password/modify |
| 路径参数/描述 | 选择参数1：oldPassword  选择参数2：newPassword |
| 参数类型 | oldPassword：String  newPassword:String |
| 描述 | 用户点击修改密码页面，输入旧密码和新密码进行验证，若旧密码输入错误，则返回修改不成功，若输入正确，则返回修改成功。 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1. 使用此功能的用户必须已登录。
2. 新密码必须在20个字符内（由英文字母，数字，符号组成）

#### 查询用户（模糊查询用户）

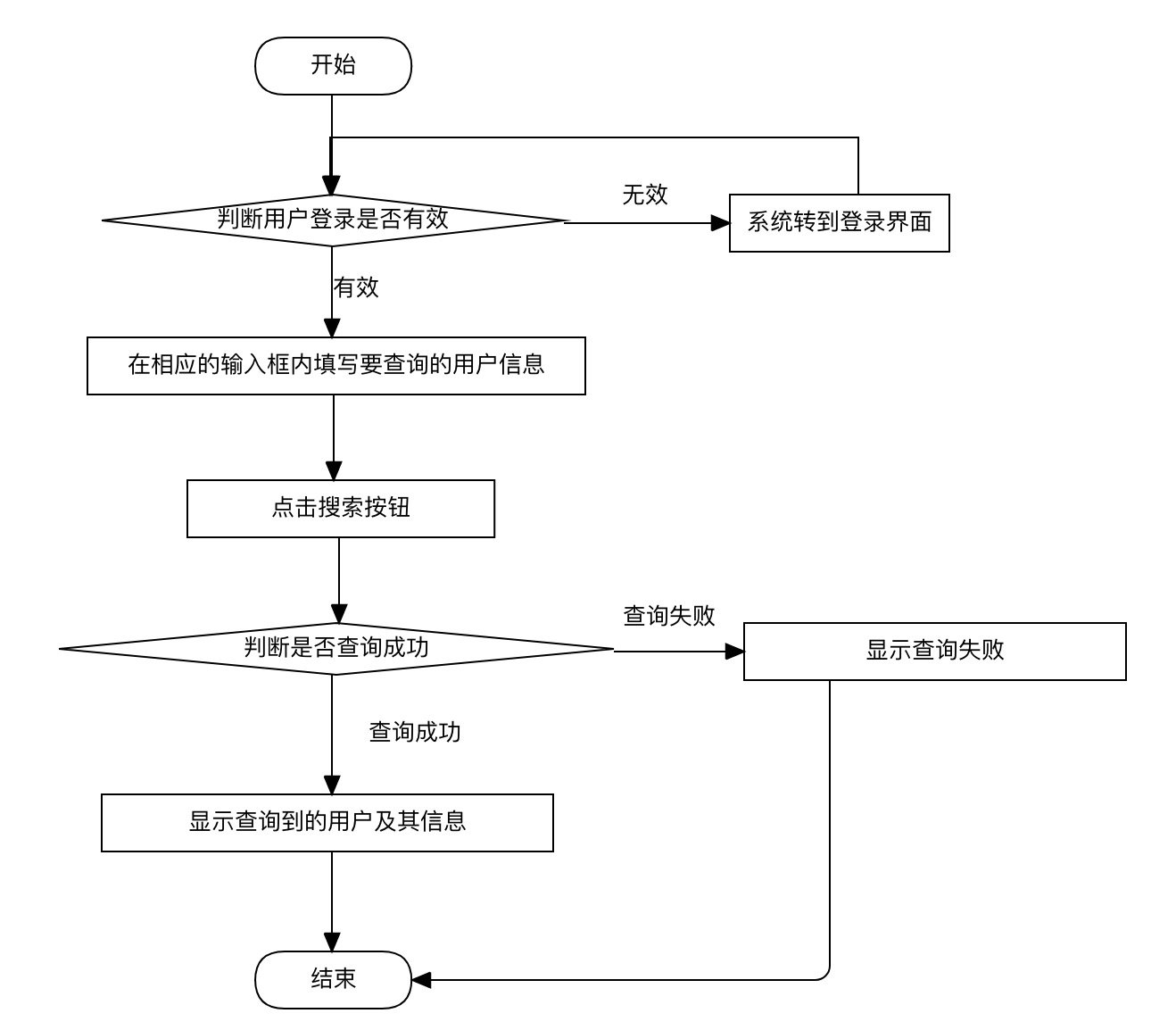
##### 输入项

用户输入需要查询的ID或者username

##### 输出项

返回一个UserList，它是符合条件的查询到的用户信息的List

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/user/find |
| 路径参数/描述 | 参数1:searchString(用户输入的查询字段) |
| 参数类型 | searchString:String |
| 描述 | 用户点击修改密码页面，输入旧密码和新密码进行验证，若旧密码输入错误，则返回修改不成功，若输入正确，则返回修改成功。 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1.用户必须先登录

#### 删除用户

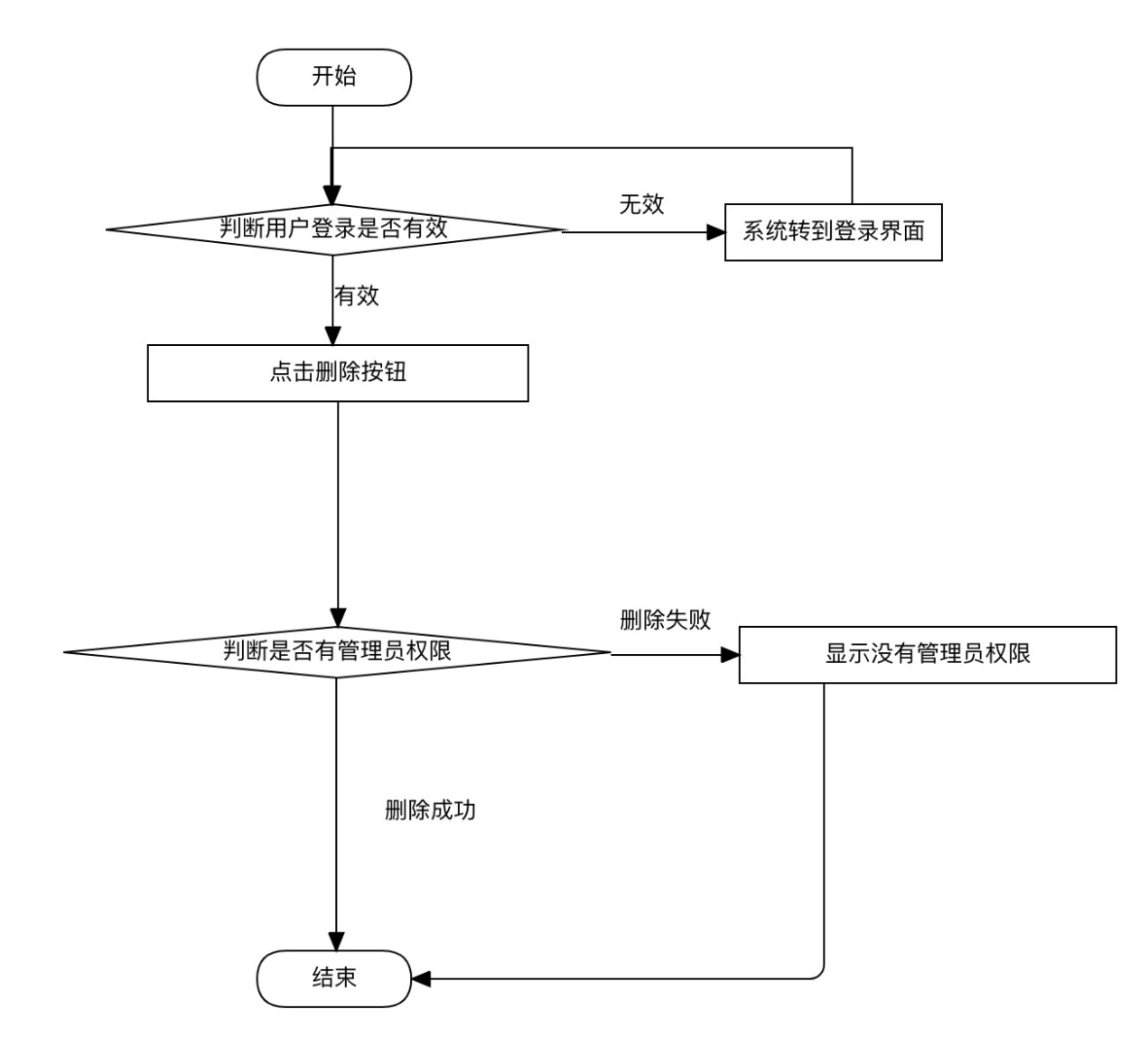
##### 输入项

无

##### 输出项

删除是否成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | DELETE |
| 服务路径 | http://[host]:8080/user/delete |
| 路径参数/描述 | 无 |
| 参数类型 | 无 |
| 描述 | 管理员进入管理员界面，点击删除用户按钮，弹出对话框是否删除，选择是，会验证管理员权限是否足够，若权限足够，则删除成功，否则会删除失败 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1．此功能只有权限足够的管理员在登录后才可用

## 评论管理模块

### 模块描述

此模块给用户和管理员的添加评论和删除评论的接口

### 功能

1. 添加评论
2. 删除评论

### 功能详细设计

#### 添加评论

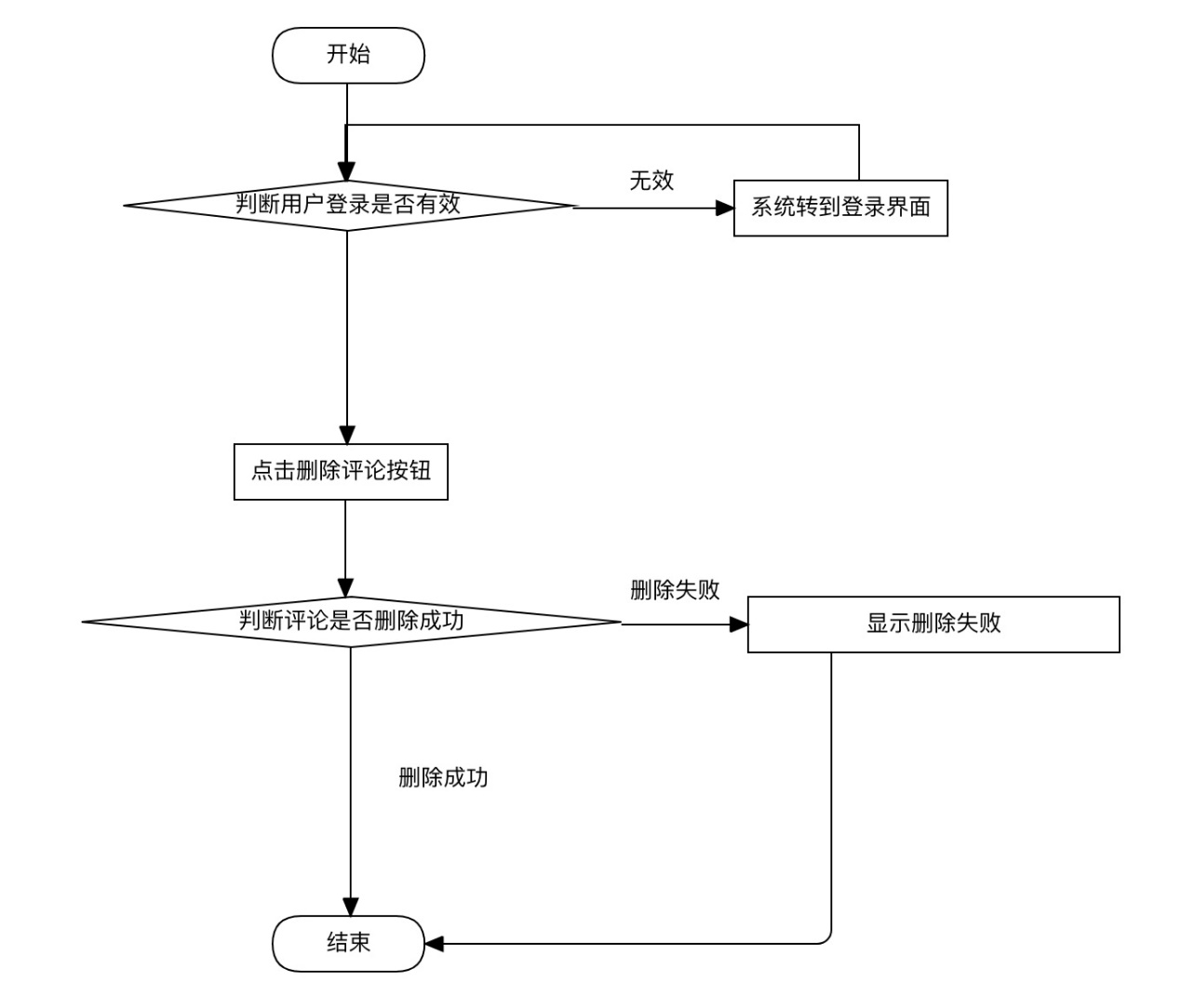
##### 输入项

用户输入评论内容

##### 输出项

返回评论是否添加成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/comment/add |
| 路径参数/描述 | 参数1：userID(用户的id)  参数2：blogID(博客的id) 参数3 content(评论的内容) |
| 参数类型 | userID:int  blogID:int  content:String |
| 描述 | 用户在博客界面下可以添加评论，评论支持富文本上传，输入评论后，点击添加评论按钮，返回是否输入成功。 |
| 返回值类型 | Json |

##### 限制条件及出错处理

1. 用户必须登陆后才能添加评论
2. 评论字段不能为空！

#### 删除评论

##### 输入项

无

##### 输出项

返回删除是否成功

##### 逻辑流程

##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | DELETE |
| 服务路径 | http://[host]:8080/comment/delete |
| 路径参数/描述 | 参数1：CommentID |
| 参数类型 | Comment：int |
| 描述 | 用户在博客界面下可以添加评论，评论支持富文本上传，输入评论后，点击添加评论按钮，返回是否输入成功。 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1. 用户只有登录后才能删除评论
2. 用户只能删除自己的评论

## 收藏夹管理

### 模块描述

### 功能

1. 添加到收藏夹
2. 从收藏夹中移除
3. 获取收藏夹的内容

### 功能详细设计

#### 添加到收藏夹

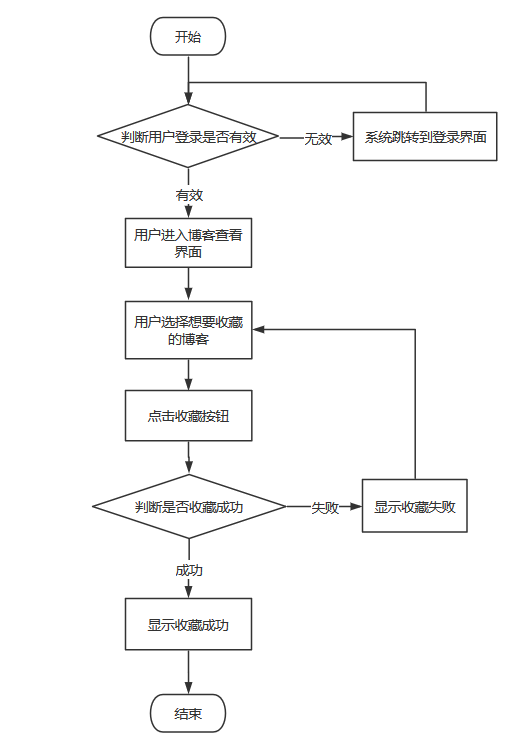
##### 输入项

无

##### 输出项

是否添加成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/favourates/add |
| 路径参数/描述 | 参数1：userID(用户的id)  参数2：blogID(博客的id) |
| 参数类型 | userID:int  blogID:int |
| 描述 | 用户点击博客下方的收藏按钮，返回是否输入成功。 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1．此功能必须登陆后使用

2.若用户已收藏博客，此时再点击收藏按钮，会提示此博客已收藏。

#### 从收藏夹中移除

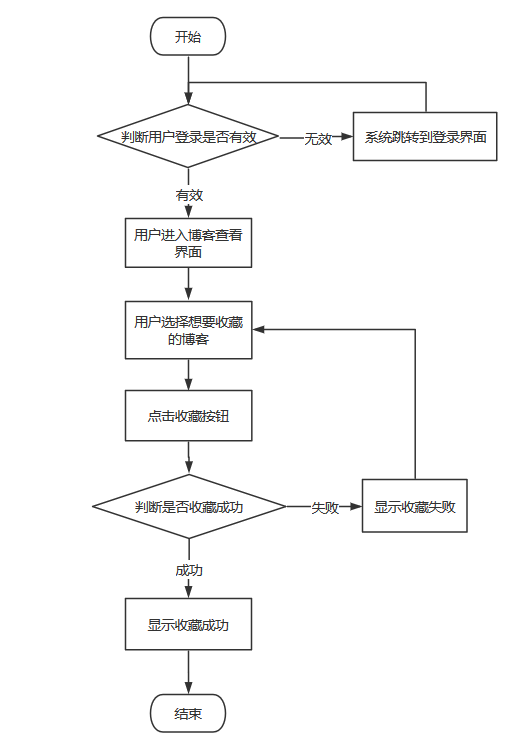
##### 输入项

无

##### 输出项

返回是否成功移除

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | DELETE |
| 服务路径 | http://[host]:8080/favourates/remove |
| 路径参数/描述 | 参数：blogID(博客的id) |
| 参数类型 | blogID:int |
| 描述 | 用户可以从收藏夹中移除已收藏的博客，点击移除按钮即可返回成功与否。 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

#### 获取收藏夹的内容

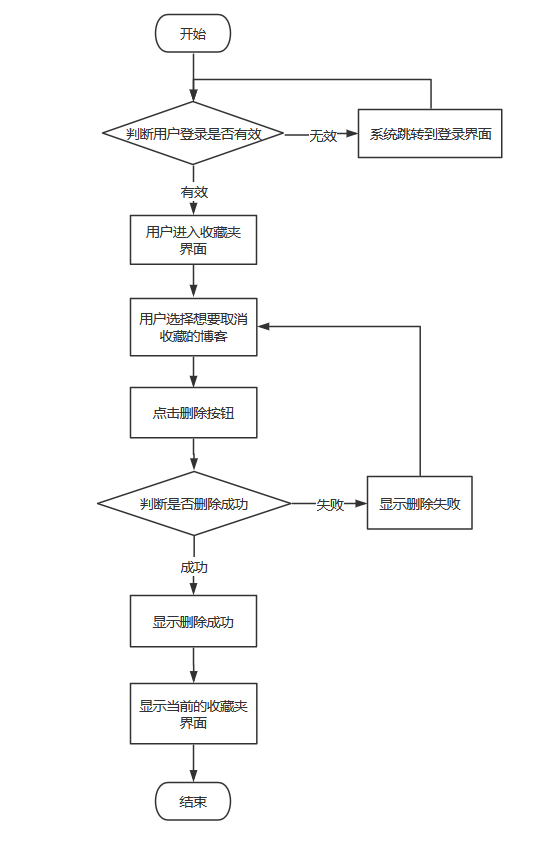
##### 输入项

无

##### 输出项

收藏夹的博客列表

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | GET |
| 服务路径 | http://[host]:8080/favourates/get |
| 路径参数/描述 | 参数1：userID(用户的id) |
| 参数类型 | userID:int |
| 描述 | 用户点击查看收藏夹的按钮，返回收藏夹的博客列表 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

## 相册和图片管理模块

### 模块描述

### 功能

1. 上传相册
2. 从相册中删除图片
3. 博客和评论中图片的上传
4. 查看相册

### 功能详细设计

#### 上传相册

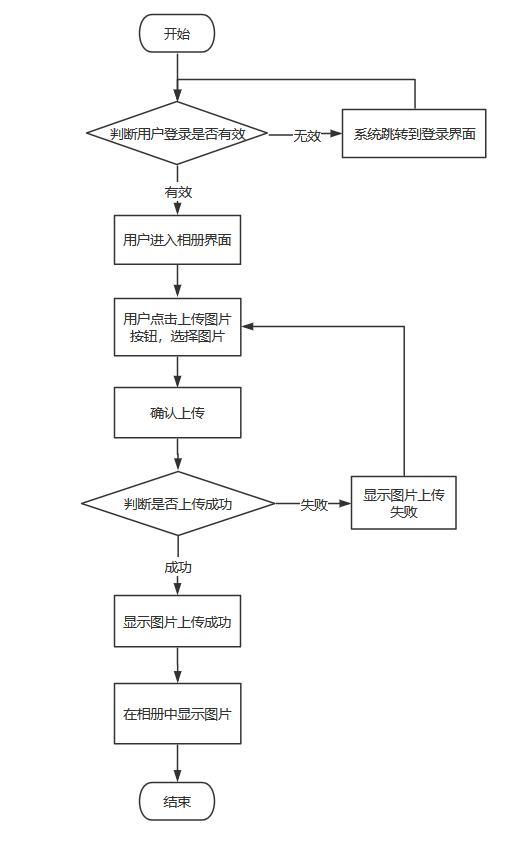
##### 输入项

用户上传一张或多张图片

##### 输出项

返回是否上传成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | POST |
| 服务路径 | http://[host]:8080/favourates/get |
| 路径参数/描述 | 参数1：userID(用户的id) |
| 参数类型 | userID:int |
| 描述 | 用户点击查看收藏夹的按钮，返回收藏夹的博客列表 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1. 登录后才能使用此功能。
2. 上传图片单张大小小于2MB

#### 从相册中删除图片

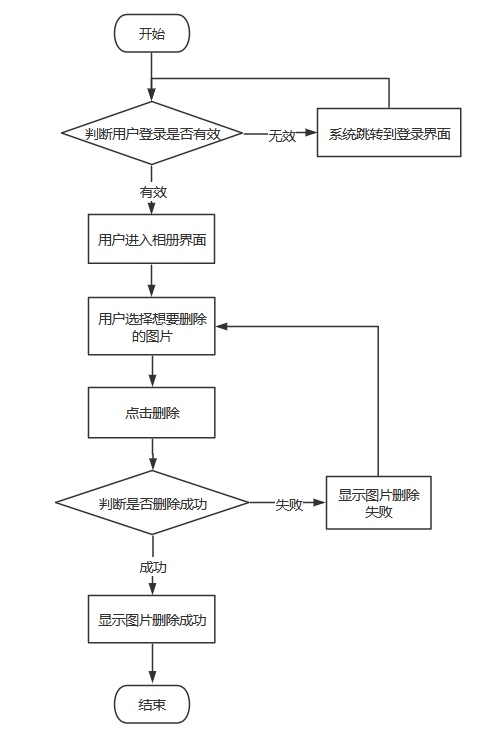
##### 输入项

无

##### 输出项

显示是否删除成功

##### 逻辑流程



##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | DELETE |
| 服务路径 | http://[host]:8080/picture/delete |
| 路径参数/描述 | 参数1：pictureID（想要删除的图片的ID） |
| 参数类型 | pictureID:int |
| 描述 | 用户点击图片的删除按钮，弹出是否删除的弹框，用户选择确认删除。 |
| 返回值类型 | Json |

##### 限制条件以及出错处理

1. 使用此功能必须先进行登录

#### 查看相册

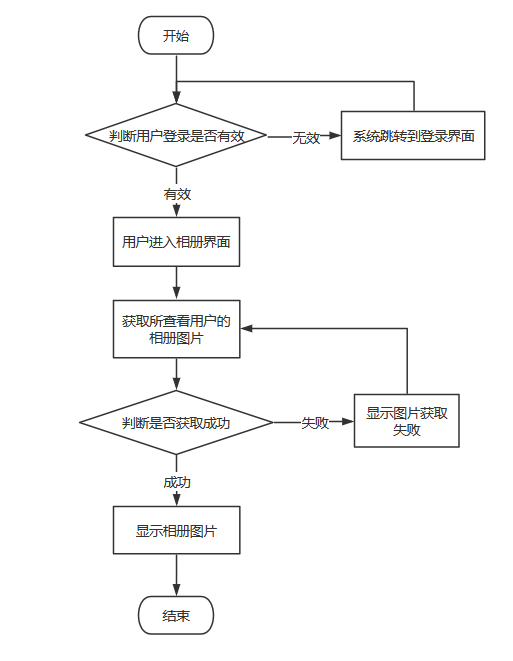
##### 输入项

无

##### 输出项

查看相册对应的图片

##### 逻辑流程

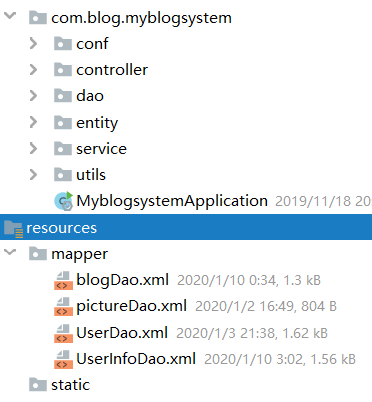


##### 接口设计

|  |  |
| --- | --- |
| 请求方式 | GET |
| 服务路径 | <http://[host]:8080/picture/get> |
| 路径参数/描述 | 参数1：userID（想要查看的相册的user的ID） |
| 参数类型 | pictureID:int |
| 描述 | 用户点击个人主页或者他人主页的相册，进入相册页面。 |
| 返回值类型 | Json |

# 模块详细设计

整体包结构如下：



Conf：全局配置类

Controller：控制类，对于不同的Http请求使用不同的相应

Dao:数据操作层的接口，把对数据库的操作封装为接口供Controller使用

Entity:实体类。

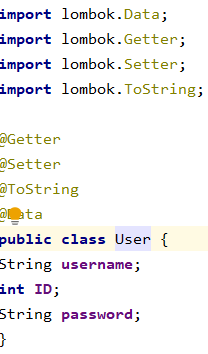
Service:为Controller的功能提供辅助

Mapper:Mybatis框架下将SQL语句进行封装实现Dao层的接口。

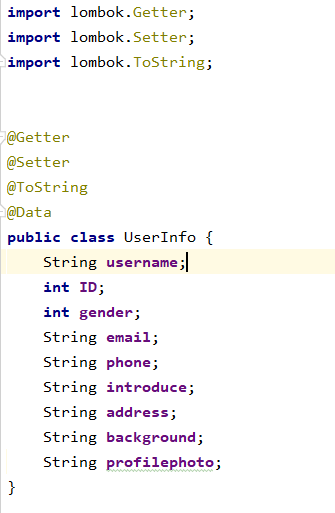
## 注册登录模块

### 类的设计和接口实现

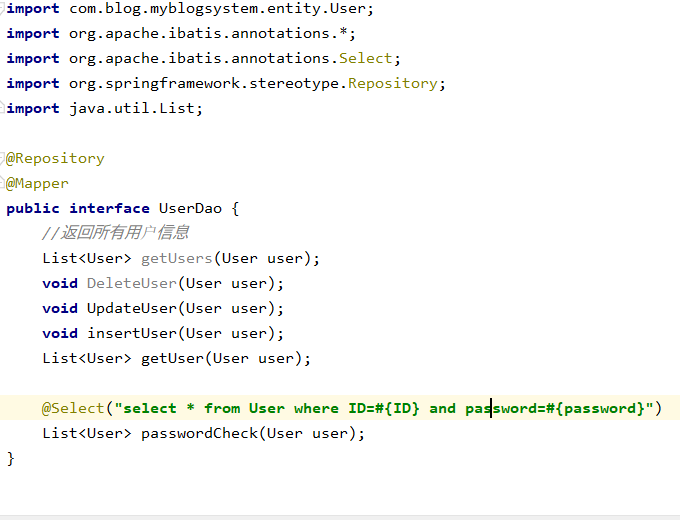
User类的设计：

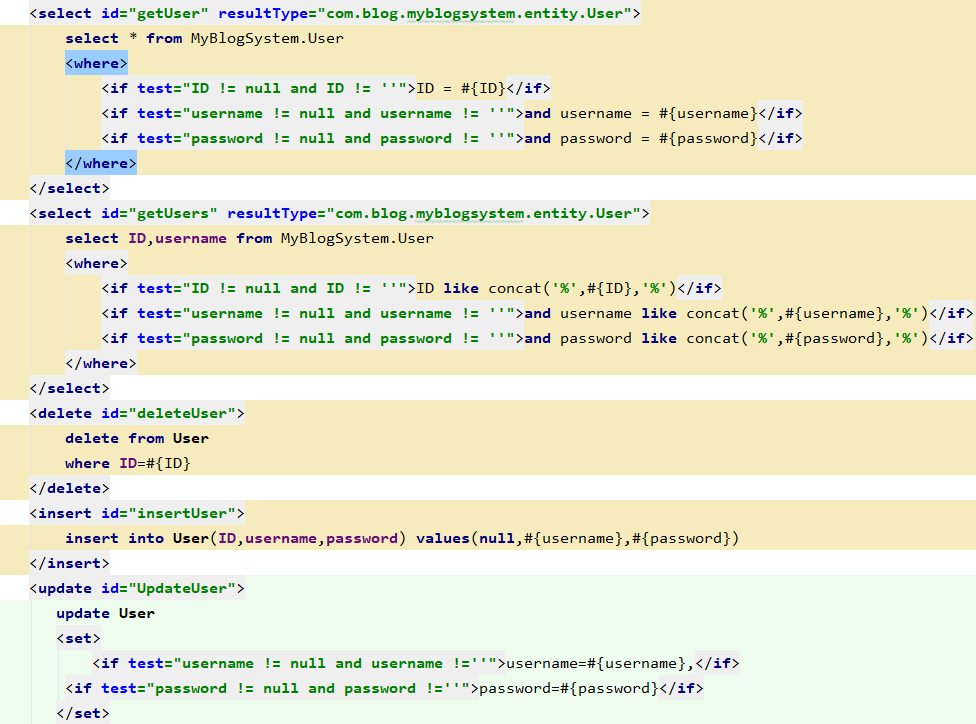


UserInfo类的设计：



Dao层接口和Mapper实现：

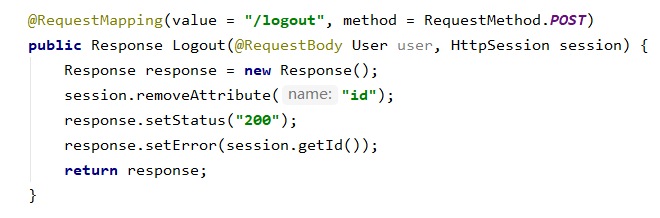




Controller:

LoginController





SignUpController:



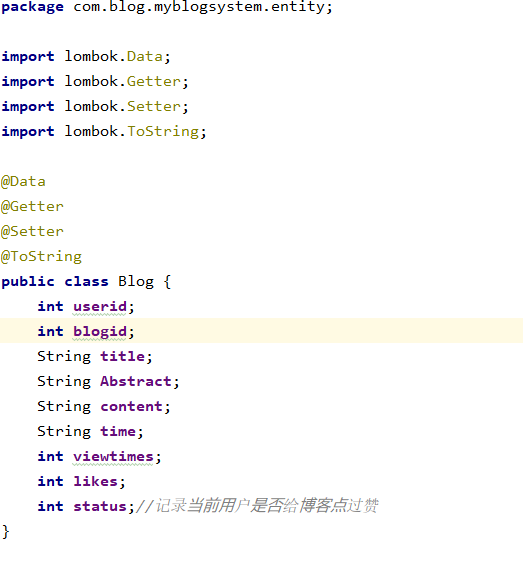
### 说明：

此模块通过LoginController和SignUpController来对登录和注册进行控制，登录时会进行账号验证，通过不同的错误类型返回给用户不同的反馈信息，登陆成功后会在Session中设置一个id来存储当前的userid,以方便前后端随时获取当前用户的状态和信息，当用户登出时，这个Session中的id属性被移除。

## 博客管理模块

### 类的设计和接口的实现：

#### Blog类



#### BlogController的实现：

##### 写博客



##### 查找博客



##### 获取当前用户的所有博客



##### 删除博客



##### 访问他人博客



##### 根据BlogID获取博客



##### 点赞功能的实现



##### 取消赞功能的实现



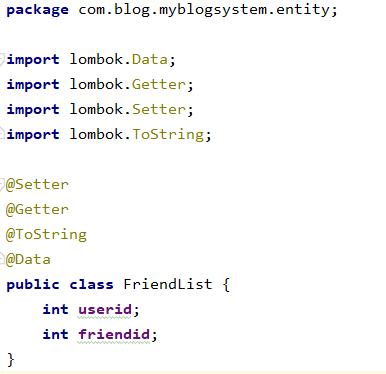
### 说明

此模块实现了博客的基本增删改查，各类访问接口和信息获取接口。最难实现的是点赞功能，由于前端实现点赞功能如果刷新就会丢失状态位，于是我在后端数据库给博客设置了一位状态位以及一个点赞表，每次刷新博客时会根据当前用户在点赞表里的数据来更新状态位，如果状态位为0，则证明当前用户还未给博客点过赞，前端会显示点赞图标，点击一下即可点赞，此时状态位更新为1，图标显示已点过赞，此时再次点击会触发取消赞的函数。使赞的数量减一，点赞状态恢复到未点赞。

## 好友管理模块

### 类的设计和接口的实现

#### FriendList的实现

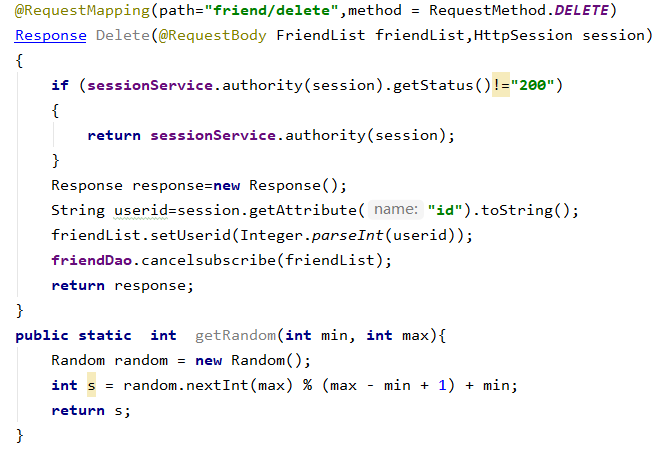


#### FriendController

##### 添加好友



##### 删除好友



##### 好友推荐



##### 获取好友列表



##### 获取朋友的朋友



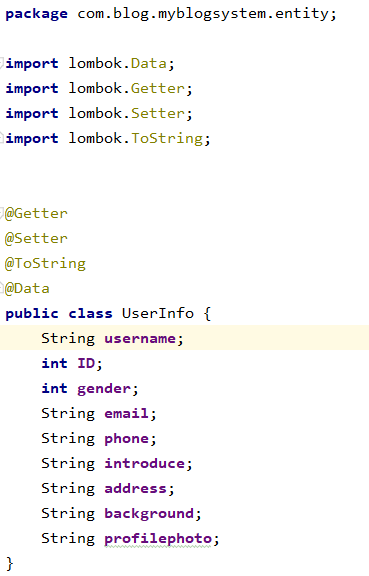
### 说明

此模块功能较为简单，采用的推荐机制为从数据库中随机获取用户推荐，而且并未设计将推荐列表中的用户加为好友后，此用户就从推荐列表移出的状态，日后此模块可以进行更多改进。

## 用户管理和个人信息模块

### 类的说明和接口的实现

#### UserInfo类



#### UserInfoController

##### 修改个人信息



##### 获取当前用户的个人信息



##### 获取指定用户的个人信息



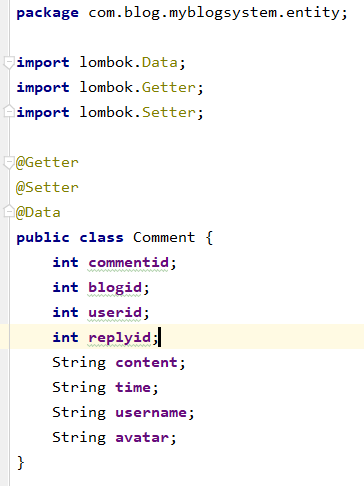
### 说明

此模块的功能较为简单，主要涉及用户信息的增删改查，因此要用Session来进行验证。

## 评论管理模块

### 类的设计和接口的实现

#### Comment类的设计



#### CommentController

##### 添加评论



##### 删除评论



##### 获取评论



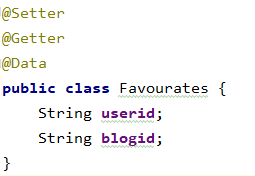
##### 获取评论状态



## 收藏夹管理模块

### 类和详细接口设计

#### Favourates类

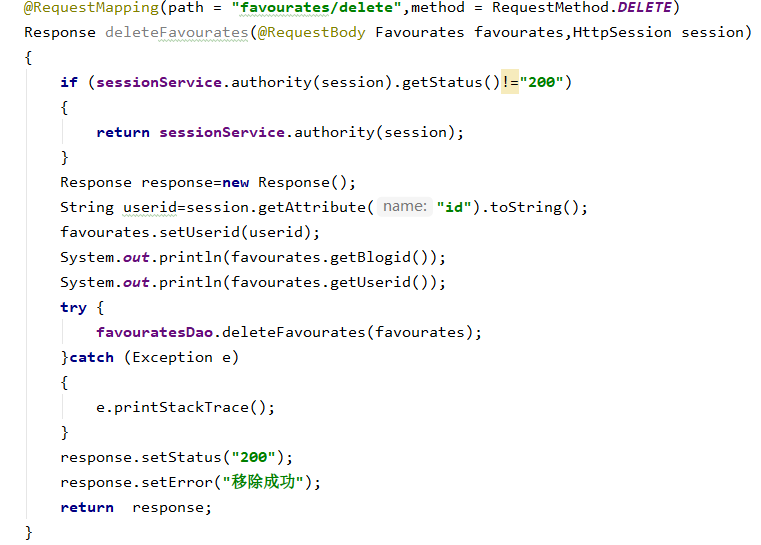


#### FavouratesController

##### 加入收藏夹



##### 从收藏夹中移除



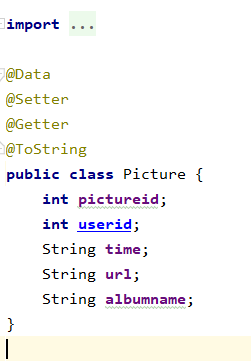
##### 获取收藏夹中的内容



## 相册和图片管理模块

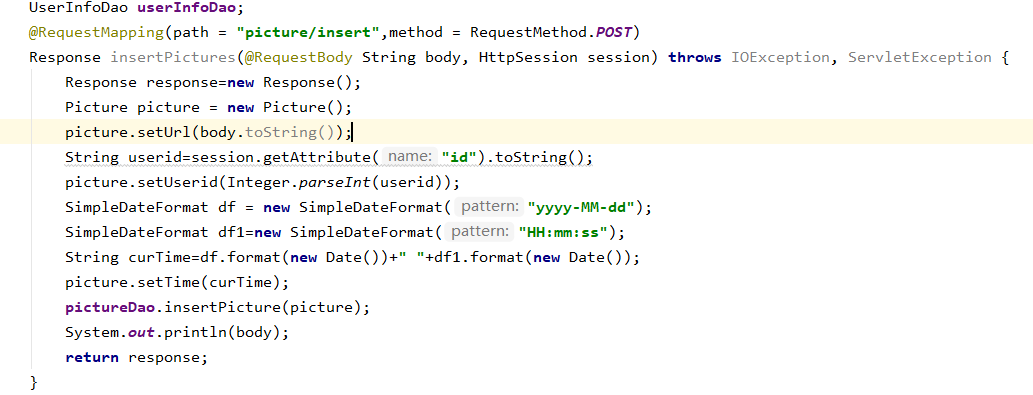
### 类和详细接口设计

#### Picture类

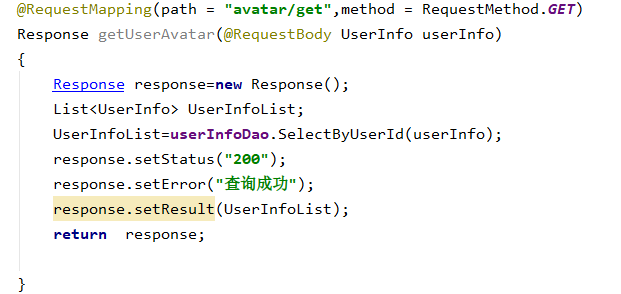


#### PictureController

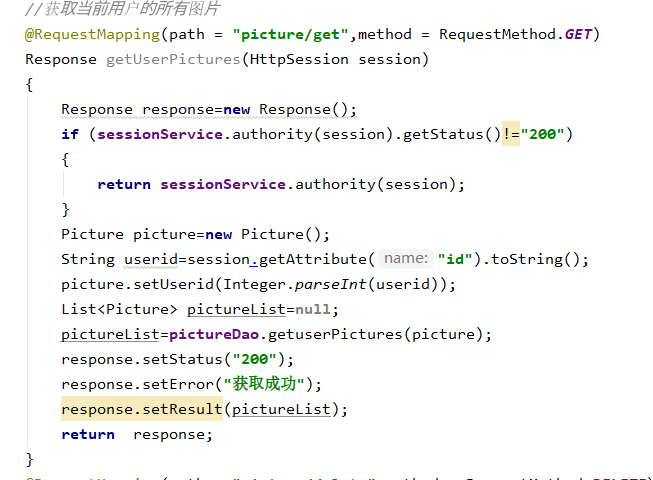
##### 向相册中插入一张图片



##### 根据ID获取用户头像



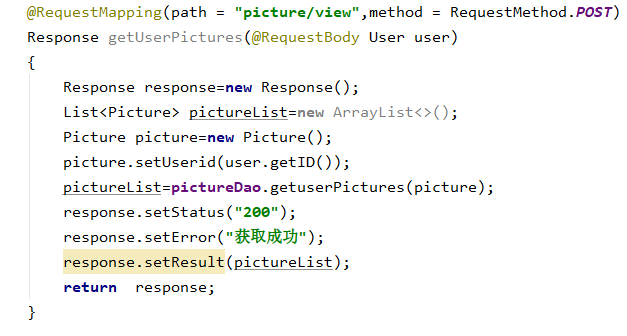
##### 获取当前用户的所有照片



##### 删除照片



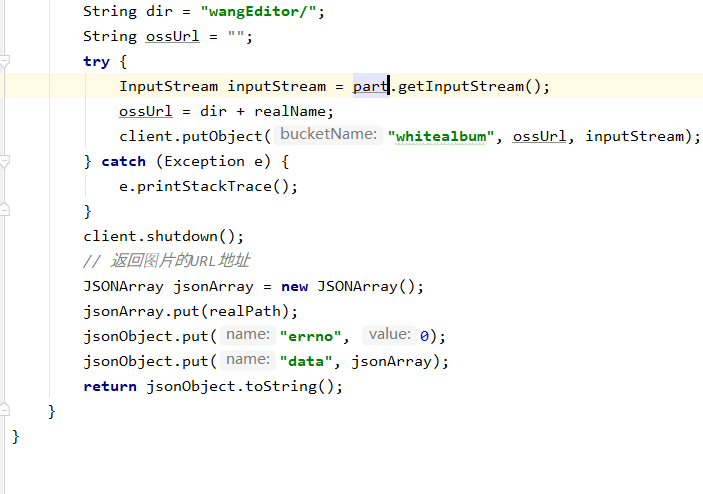
##### 访问他人相册



#### Image Controller

这部分主要涉及WangEditor和OSS的集成





### 说明：

本次相册和图像的存储使用了阿里云的OSS对象存储，即上传照片到云端，前端通过调用后端接口与OSS建立连接并上传，主要是相册和富文本编辑器在使用这个模块。前端上传后会返回给后端一个url。这个URL添加了随机后缀。所有相册中图片的URL均存储于相册中，每次执行相册的删除操作理论上都会进行以下两个步骤：1.通过API将云端照片删除 2.将数据库里的这条记录删除，但是考虑到我们这个项目开发时间不是很充足，用户的需求量也不是很大，就暂时没有处理这个问题，仅仅在数据库里删除了这条记录。这个模块以后会进行完善。