目录

[数据库 3](#_Toc90373631)

[Privilege（权限表） 3](#_Toc90373632)

[Role（角色表） 3](#_Toc90373633)

[Role\_ Privilege（角色-权限表） 4](#_Toc90373634)

[College（学院表） 4](#_Toc90373635)

[User（用户表） 4](#_Toc90373636)

[User\_Role（用户-角色表） 5](#_Toc90373637)

[User\_check（注册检验表） 5](#_Toc90373638)

[State（状态表） 5](#_Toc90373639)

[State\_text（状态-正文表） 5](#_Toc90373640)

[Like\_state（点赞状态表） 6](#_Toc90373641)

[Comment（评论） 6](#_Toc90373642)

[Album（相册表） 6](#_Toc90373643)

[Album\_picture（相册-图片表） 6](#_Toc90373644)

[Follow（关注列表） 7](#_Toc90373645)

[Banner（主页轮播图）【优先级低】 7](#_Toc90373646)

[API 7](#_Toc90373647)

[/file（文件） 8](#_Toc90373648)

[/privilege（权限） 8](#_Toc90373649)

[/select/${privilege\_id}（通过id查询权限） 8](#_Toc90373650)

[/selectAll（查询所有权限） 9](#_Toc90373651)

[/user/${user\_id}（查询用户id对应的权限） 9](#_Toc90373652)

[/role（角色） 10](#_Toc90373653)

[/insert（插入角色） 10](#_Toc90373654)

[/update/${role\_id}（修改角色） 10](#_Toc90373655)

[/delete/${role\_id}（删除角色） 11](#_Toc90373656)

[/select/${role\_id}（根据id查询角色） 11](#_Toc90373657)

[/selectAll（查询所有角色） 11](#_Toc90373658)

[/selectCondition（模糊条件查询角色） 12](#_Toc90373659)

[/user/${user\_id}（查询用户id对应的角色） 13](#_Toc90373660)

[/college（学院） 13](#_Toc90373661)

[/insert（添加学院） 13](#_Toc90373662)

[/update/${college\_id}（修改学院） 14](#_Toc90373663)

[/delete/${college\_id}（删除学院） 14](#_Toc90373664)

[/select/${college\_id}（根据id查询学院） 14](#_Toc90373665)

[/selectAll（查询所有学院） 15](#_Toc90373666)

[/selectCondition（模糊条件查询学院） 15](#_Toc90373667)

[/user（用户） 16](#_Toc90373668)

[/register（注册） 16](#_Toc90373669)

[/login（登陆） 17](#_Toc90373670)

[/logout（登出） 17](#_Toc90373671)

[/update/{$user\_id}（更新用户数据） 17](#_Toc90373672)

[/delete/{$user\_id}（删除用户） 18](#_Toc90373673)

[/select/${user\_id}（通过id查询用户） 18](#_Toc90373674)

[/selectAll（查询所有用户） 19](#_Toc90373675)

[/selectCondition（模糊条件查询用户） 19](#_Toc90373676)

[/followers（获取粉丝列表） 20](#_Toc90373677)

[/followings（获取关注列表） 21](#_Toc90373678)

[/follow/${user\_id}（关注某人） 21](#_Toc90373679)

[/unfollow/${user\_id}（取关某人） 21](#_Toc90373680)

[/album（相册） 22](#_Toc90373681)

[/insert（新建相册） 22](#_Toc90373682)

[/update/${album\_id}（修改相册） 22](#_Toc90373683)

[/delete/${album\_id}（删除相册） 23](#_Toc90373684)

[/select/${album\_id}（通过id查询相册） 23](#_Toc90373685)

[/selectAll（查询用户名下所有相册） 23](#_Toc90373686)

[/picture（相册图片） 24](#_Toc90373687)

[/insert/${album\_id}（上传图片） 24](#_Toc90373688)

[/delete/${picture\_id}（根据id删除图片） 24](#_Toc90373689)

[/selectAll/${album\_id}（查询相册id对应的所有图片） 24](#_Toc90373690)

[/state（状态） 25](#_Toc90373691)

[/insert（新建状态） 25](#_Toc90373692)

[/update/${state\_id}（修改状态） 25](#_Toc90373693)

[/delete/${state\_id}（删除状态） 26](#_Toc90373694)

[/select/${state\_id}（通过id查询状态） 26](#_Toc90373695)

[/selectAll（查询用户的所有状态） 27](#_Toc90373696)

[/selectCondition（条件查询状态） 27](#_Toc90373697)

[/recommend（主页推荐的状态） 28](#_Toc90373698)

[/like/${state\_id}（点赞状态） 28](#_Toc90373699)

[/unlike/${state\_id}（取消点赞状态） 29](#_Toc90373700)

[/comment（评论） 29](#_Toc90373701)

[/insert（添加评论） 29](#_Toc90373702)

[/delete/${comment\_id}（删除评论） 30](#_Toc90373703)

[/select/${comment\_id}（根据id获取评论） 30](#_Toc90373704)

[/selectAll/${state\_id}（获取一个状态的一级评论） 30](#_Toc90373705)

[/selectAllReply/${comment\_id}（获取一个评论下的所有子评论） 31](#_Toc90373706)

[/banner（轮播图）【优先级低】 32](#_Toc90373707)

[/insert（添加轮播图） 32](#_Toc90373708)

[/delete/${banner\_id}（根据id删除轮播图） 32](#_Toc90373709)

[/update/${banner\_id}（根据id修改轮播图） 32](#_Toc90373710)

[/select/${banner\_id}（根据id查询轮播图） 33](#_Toc90373711)

[/selectAll（查询所有轮播图） 33](#_Toc90373712)

[/selectValid（查询主页有效轮播图） 34](#_Toc90373713)

# 数据库

以下所有数据库表均省略以下两个字段

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Gmt\_create | Datetime | 创建时间 |
| Gmt\_modified | Datetime | 修改时间 |

以下所有数据表省略以下规则：

1. Id为0表示系统，常见于：直接操作数据库，由系统自动生成，没有该id等。所以所有id至少从1开始，0为保留id。
2. 一般权限-角色-用户还应该有签名（signature）字段，保证生成jwt的token。（后续再考虑是否添加）

## Privilege（权限表）

权限只能由直接操作数据库生成。并且每个的权限路由是由其父路由+自己的权限路由组合而成。

拥有父权限即拥有所有子权限。所以第一个权限“/”为最大的，意味可以访问所有API，一般是超级管理员。权限的父子关系可以参考目录。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Name | Varchar(20) | 权限名 |
| Url | Varchar(100) | 权限路由 |
| Parent | Int(20) | 父权限id |

## Role（角色表）

角色所对应的权限应当放在redis缓存中，并设置有效时间进行定期更新，如果前期来不及可以先不考虑。  
创建者id为0的角色表示基本角色，**不允许前台删除！！！**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Name | Varchar(20) | 角色名 |
| Description | Varchar(100) | 描述 |
| Create\_id | Int(20) | 创建者id |
| Modify\_id | Int(20) | 修改者id |

## Role\_ Privilege（角色-权限表）

创建者id为0的角色-权限，不允许删除。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Role\_id | Int(20) | 角色id |
| Privilege\_id | Int(20) | 权限id |
| Create\_id | Int(20) | 创建者id |
| Modify\_id | Int(20) | 修改者id |

## College（学院表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Name | Varchar(20)` | 学院名 |
| Description | Varchar(200) | 描述 |

## User（用户表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Student\_id | Int(20) | 学号 |
| Password | Varchar(50) | 密码 |
| Name | Varchar(20) | 姓名 |
| College\_id | Int(20) | 学院id |
| Gender | Int(5) | 性别（0保密，1男，2女） |
| Avater | Varchar(200) | 头像url |
| Description | Varchar(200) | 个性签名 |
| Birthday | Datatime | 生日 |
| Phone | Varchar(30) | 电话 |
| Email | Varchar(50) | 邮箱 |
| Sign\_state | Int(5) | 用户状态（0注册待审核，1正常） |
| Disable | Int(1) | 逻辑删除（0正常，1被逻辑删除） |

## User\_Role（用户-角色表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Role\_id | Int(20) | 角色id |
| User\_id | Int(20) | 用户id |
| Create\_id | Int(20) | 创建者id |
| Modify\_id | Int(20) | 修改者id |

## User\_check（注册检验表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| User\_id | Int(20)` | 用户id |
| Reviewer\_id | Int(20) | 审核人id |
| State | Int(20) | 状态（0待审核，1通过，2拒绝） |

## State（状态表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| User\_id | Int(20) | 用户id |
| Title | Varchar(50) | 标题 |
| Share\_state | Int(5) | 分享状态（0否，1是） |
| Disable | Int(5) | 逻辑删除（0正常，1·删除） |

## State\_text（状态-正文表）

分表，因为正文可能很长影响查阅速度

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| State\_id | Int(20) | 状态id |
| Text | Varchar(2000) | 正文 |
| Disable | Int(5) | 逻辑删除（0正常，1·删除） |

## Like\_state（点赞状态表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| State\_id | Int(20) | 状态id |
| User\_id | Int(20) | 点赞用户id |
| Disable | Int(5) | 逻辑删除（0正常，1·删除） |

## Comment（评论）

对于父id和回复id的说明。参考B站的设计，B站只有两层评论设计。第一层直接显示在视频下，第二层可以展开评论。展开的评论依附于第一层的评论，但同时可以回复第二层的其他评论。

|  |  |  |
| --- | --- | --- |
| Id | Int(20) | Id |
| State\_id | Int(20) | 状态id |
| User\_id | Int(20) | 评论用户id |
| Father\_id | Int(20) | 父评论id |
| Reply\_id | Int(20) | 回复评论id |
| Text | Varchar(200) | 评论正文 |
| Disable | Int(5) | 逻辑删除（0正常，1·删除） |

## Album（相册表）

封面如果被置空，应该默认填充默认封面的url。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| User\_id | Int(20) | 用户id |
| Name | Varchar(30) | 相册名 |
| Description | Varchar(200) | 相册描述 |
| Cover | Varchar(200) | 封面的url |
| Disable | Int(5) | 逻辑删除（0正常，1删除） |

## Album\_picture（相册-图片表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Album\_id | Int(20) | 相册id |
| Name | Varchar(50) | 图片备注 |
| url | Varchar(200) | 图片对应url |

## Follow（关注列表）

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| User\_id | Int(20) | 用户id |
| Follower\_id | Int(20) | 粉丝id |

## Banner（主页轮播图）【优先级低】

系统公告通过状态的方式发送（统一过程避免出现更多的麻烦）

State\_id为0表示不跳转到任何状态

没有逻辑删除的，不应该超过5个。

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| Id | Int(20) | Id |
| Create\_id | Int(20) | 创建者id |
| url | Varchar(200) | 轮播图的url |
| Comment | Varchar(20) | 备注信息 |
| State\_id | Int(20) | 点击后跳转到某个状态 |
| Disable | Int(5) | 逻辑删除（0正常，1删除） |

# API

Api请求参数与返回参数说明。

1. 所有参数的命名遵循**首单词小写的驼峰法**。Word有首字母自动大写，我懒得调了。
2. 如果是**查询**，则实体省略两个参数：
   1. Gmt\_date: 创建时间
   2. Gmt\_modied: 修改时间
3. 如果是**查询**，如无特殊说明，按照创建时间倒序返回，即最后创建的排在最前面

**响应格式**：

Code为状态码：

* 200：请求成功
* 403：权限拒绝（出现重复、没有访问api的权限、约束规则拒绝等）
* 404：请求的资源不存在（比如删除一个user，userid不存在）
* 500：内部错误

Msg为请求结果消息（如果有出错应当返回报错原因）：

Data数据内容（如果返回类型为数组，应当在data下放一个对象item）：

|  |
| --- |
| {      "code": 0,      "msg": "",      "data": {        }  } |

**请求格式**：

请求必定包含一个token字段，如果不需要token就留空

|  |
| --- |
| {      "token": “”,  } |

## /file（文件）

（这部分要看看minio的C#是如何实现的）

文件上传需要做单独的鉴权，评估能否上传【优先级低】。

## /privilege（权限）

权限不允许前台修改

### /select/${privilege\_id}（通过id查询权限）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Id: 权限id,  Name: 权限名,  url: 权限路由,  parent: 父权限id  } |

### /selectAll（查询所有权限）

权限并不多，路由权限最多一百个，可以直接返回而不必分页。

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Item: [  {  Id: 权限id,  Name: 权限名,  url: 权限路由,  parent: 父权限id  }  ]  } |

### /user/${user\_id}（查询用户id对应的权限）

该方法需要复杂的连表查询，但是只有管理员使用，可以不必太考虑性能。

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Item: [  {  Id: 权限id,  Name: 权限名,  url: 权限路由,  parent: 父权限id  }  ]  } |

## /role（角色）

### /insert（插入角色）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  } |

**响应：**

通过响应码表示请求成功/失败

成功应当返回角色id

|  |
| --- |
| Data: {  Id: 角色id  } |

### /update/${role\_id}（修改角色）

由系统创建的角色不允许修改

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /delete/${role\_id}（删除角色）

由系统创建的角色不允许删除

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${role\_id}（根据id查询角色）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Id: 角色id,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  } |

### /selectAll（查询所有角色）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: 角色id,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  }  ]  } |

### /selectCondition（模糊条件查询角色）

例如我搜索一个名字“信息”，那么对应的应该是搜索的“\*信息\*”，即模糊匹配，在select前后加\*即可。

**请求：**

|  |
| --- |
| Data: {  Token: token,  Id: 角色id,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  BeginDate: 创建日期下限,  endDate: 创建日期上限,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: 角色id,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  }  ]  } |

### /user/${user\_id}（查询用户id对应的角色）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Item: [  {  Id: 角色id,  Name: 角色名,  Description: 角色描述,  Privilege: [权限id数组]  }  ]  } |

## /college（学院）

### /insert（添加学院）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 学院名,  Description: 学院描述  } |

**响应：**

通过响应码表示请求成功/失败

成功应当返回学院id

|  |
| --- |
| Data: {  Id: college\_id  } |

### /update/${college\_id}（修改学院）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 学院名,  Description: 学院描述  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /delete/${college\_id}（删除学院）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${college\_id}（根据id查询学院）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Id: 学院id  Name: 学院名,  Description: 学院描述  } |

### /selectAll（查询所有学院）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: 学院id,  Name: 学院名,  Description: 学院描述  }  ]  } |

### /selectCondition（模糊条件查询学院）

例如我搜索一个名字“信息”，那么对应的应该是搜索的“\*信息\*”，即模糊匹配，在select前后加\*即可。

**请求：**

|  |
| --- |
| Data: {  Token: token,  Id: 学院id,  Name: 学院名,  Description: 学院描述,  BeginDate: 创建日期下限,  endDate: 创建日期上限,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: 学院id,  Name: 学院名,  Description: 学院描述  }  ]  } |

## /user（用户）

### /register（注册）

**请求：**

|  |
| --- |
| Data: {  StudentId: 学号,  Password: 密码,  Name: 姓名,  CollegeId: 学院id,  Gender: 性别,  Avater: 头像url,  Description: 个性签名,  Birthday: 生日,  Phone: 电话,  Email: 邮箱  } |

**响应：**

通过响应码表示请求成功/失败

如果成功应当返回token（注册成功之后自动登陆）和用户id。

|  |
| --- |
| Data: {  Id: user\_id,  Token: “”  } |

### /login（登陆）

**请求：**

|  |
| --- |
| Data: {  StudentId: 学号,  Password: 密码,  } |

**响应：**

通过响应码表示请求成功/失败

失败时token应当为空

|  |
| --- |
| Data: {  Token: token  } |

### /logout（登出）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

失败时token应当为空

|  |
| --- |
| Data: true/false |

### /update/{$user\_id}（更新用户数据）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 姓名,  CollegeId: 学院id,  Gender: 性别,  Avater: 头像url,  Description: 个性签名,  Birthday: 生日,  Phone: 电话,  Email: 邮箱  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /delete/{$user\_id}（删除用户）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${user\_id}（通过id查询用户）

如果user\_id为0，返回当前token对应用户的数据（即获取自己的用户数据）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Id:用户id  Name: 姓名,  CollegeName: 学院名,  Gender: 性别,  Avater: 头像url,  Description: 个性签名,  Birthday: 生日,  Phone: 电话,  Email: 邮箱  } |

### /selectAll（查询所有用户）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: user\_id  Name: 姓名,  CollegeId: 学院id,  Gender: 性别,  Avater: 头像url,  Description: 个性签名,  Birthday: 生日,  Phone: 电话,  Email: 邮箱  }  ]  } |

### /selectCondition（模糊条件查询用户）

例如我搜索一个名字“李”，那么对应的应该是搜索的“\*李\*”，即模糊匹配，在select前后加\*即可。

**请求：**

|  |
| --- |
| Data: {  Token: token,  Id: 用户id  Name: 姓名,  CollegeName: 学院名,  Gender: 性别, ,  BeginDate: 创建日期下限,  endDate: 创建日期上限,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: user\_id  Name: 姓名,  CollegeName: 学院名,  Gender: 性别,  Avater: 头像url,  Description: 个性签名,  Birthday: 生日,  Phone: 电话,  Email: 邮箱  }  ]  } |

### /followers（获取粉丝列表）

**请求：**

|  |
| --- |
| Data: {  Token: token  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  用户id数组  ]  } |

### /followings（获取关注列表）

一般APP的关注上限在5000左右。可按实际运行情况修改。

**请求：**

|  |
| --- |
| Data: {  Token: token  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  用户id数组  ]  } |

### /follow/${user\_id}（关注某人）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /unfollow/${user\_id}（取关某人）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

## /album（相册）

### /insert（新建相册）

用户id通过token获取。新建相册的封面应该为默认封面，后面可以修改为相册内图片。

相册数不应该超过20个，超过应该创建失败。

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 相册名,  Description: 相册描述  } |

**响应：**

通过响应码表示请求成功/失败

成功应当返回id

|  |
| --- |
| Data: {  Id: album\_id  } |

### /update/${album\_id}（修改相册）

只允许修改自己名下的相册

**请求：**

|  |
| --- |
| Data: {  Token: token,  Name: 相册名,  Description: 相册描述,  Cover: 封面url  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /delete/${album\_id}（删除相册）

只允许删除自己名下的相册

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${album\_id}（通过id查询相册）

用户只能查询自己名下的相册。

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Id: 相册id  Name: 相册名,  Description: 相册描述,  Cover: 封面url  } |

### /selectAll（查询用户名下所有相册）

因为相册数量不多，可以不做分页。**按照名字字典序排序**。

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Item: [  {  Id: album\_id  Name: 相册名,  Description: 相册描述,  Cover: 封面url  }  ]  } |

## /picture（相册图片）

### /insert/${album\_id}（上传图片）

这部分根据minio的C#实现情况再做设计。

### /delete/${picture\_id}（根据id删除图片）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /selectAll/${album\_id}（查询相册id对应的所有图片）

**请求：**

|  |
| --- |
| Data: {  Token: token  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: picture\_id  Name: 图片备注,  url: 图片url  }  ]  } |

## /state（状态）

### /insert（新建状态）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Title: 标题,  Text: 正文  shareState: 分享状态  } |

**响应：**

通过响应码表示请求成功/失败

成功应当返回id

|  |
| --- |
| Data: {  Id: state\_id  } |

### /update/${state\_id}（修改状态）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Title: 标题,  Text: 正文  shareState: 分享状态  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /delete/${state\_id}（删除状态）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${state\_id}（通过id查询状态）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  Id: 状态id  Title: 标题,  Text: 正文  shareState: 分享状态,  like: true/false（token对应用户是否有喜欢这篇状态）,  likeNumber: 点赞数  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  } |

### /selectAll（查询用户的所有状态）

返回的是token对应的user\_id的所有状态

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  id: 状态id,  Title: 标题,  shareState: 分享状态  like: true/false（token对应用户是否有喜欢这篇状态）,  likeNumber: 点赞数  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  }  ]  } |

### /selectCondition（条件查询状态）

该api不只是查一个用户的，可以查所有分享状态的

**请求：**

|  |
| --- |
| Data: {  Token: token,  User\_id: 用户id  Title: 标题  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  id: 状态id,  Title: 标题,  shareState: 分享状态  like: true/false（token对应用户是否有喜欢这篇状态）,  likeNumber: 点赞数  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  }  ]  } |

### /recommend（主页推荐的状态）

主页推荐的状态。设计一个算法根据热度+关注来随机生成一个用户的主页推荐。

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  id: 状态id,  Title: 标题,  shareState: 分享状态  like: true/false（token对应用户是否有喜欢这篇状态）,  likeNumber: 点赞数  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  }  ]  } |

### /like/${state\_id}（点赞状态）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /unlike/${state\_id}（取消点赞状态）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

## /comment（评论）

### /insert（添加评论）

父id为0表示表示状态的一级评论。Reply\_id为0表示不回复任何人。

**请求：**

|  |
| --- |
| Data: {  Token: token,  State\_id: 状态id,  Father\_id: 父id  Reply\_id: 回复id  Text: 正文  } |

**响应：**

通过响应码表示请求成功/失败

成功应当返回id

|  |
| --- |
| Data: {  Id: 评论id  } |

### /delete/${comment\_id}（删除评论）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${comment\_id}（根据id获取评论）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  State\_id: 状态id,  Father\_id: 父id  Reply\_id: 回复id  Text: 正文,  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  } |

### /selectAll/${state\_id}（获取一个状态的一级评论）

即father\_id=0 && state\_id = ${state\_id}

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: 评论id  State\_id: 状态id,  Father\_id: 父id  Reply\_id: 回复id  Text: 正文,  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  }  ]  } |

### /selectAllReply/${comment\_id}（获取一个评论下的所有子评论）

即父id为comment\_id

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  Id: 评论id  State\_id: 状态id,  Father\_id: 父id  Reply\_id: 回复id  Text: 正文,  userId: 作者id  username: 作者的名字  userAvater: 作者的头像  }  ]  } |

## /banner（轮播图）【优先级低】

### /insert（添加轮播图）

**请求：**

|  |
| --- |
| Data: {  Token: token,  url: 轮播图url  comment: 备注信息  state\_id: 跳转的状态id  } |

**响应：**

通过响应码表示请求成功/失败

成功应当返回id

|  |
| --- |
| Data: {  Id: 轮播图id  } |

### /delete/${banner\_id}（根据id删除轮播图）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /update/${banner\_id}（根据id修改轮播图）

**请求：**

|  |
| --- |
| Data: {  Token: token  url: 轮播图url  comment: 备注信息  state\_id: 跳转的状态id  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: true/false |

### /select/${banner\_id}（根据id查询轮播图）

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

|  |
| --- |
| Data: {  url: 轮播图url  comment: 备注信息  state\_id: 跳转的状态id  } |

### /selectAll（查询所有轮播图）

**请求：**

|  |
| --- |
| Data: {  Token: token,  Page: 页码（从0开始到total-1）,  pageSize: 页大小（应当大于等于1）  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Total: 按照当前pageSize有多少页  Item: [  {  id: 轮播图id,  url: 轮播图url  comment: 备注信息  state\_id: 跳转的状态id  }  ]  } |

### /selectValid（查询主页有效轮播图）

只返回前5个，多的不返回后面的。

**请求：**

|  |
| --- |
| Data: {  Token: token  } |

**响应：**

通过响应码表示请求成功/失败

返回一个数组

|  |
| --- |
| Data: {  Item: [  {  id: 轮播图id,  url: 轮播图url  comment: 备注信息  state\_id: 跳转的状态id  }  ]  } |