（红色的是请求的假数据，没有请求数据库；绿色部分暂时不可用）

# 一．数据字典表查询

## 获取所有省列表 /get\_province

* 请求方法：post
* 接口名称：/get\_province
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

}

* 返回结果：json

{

code：0 //状态码，

data：[

{

“province\_name”: //省名

“n\_province\_id”: //省的国家id

},

……….

]

}

* 注意事项：无

## 获取某省所有市的列表 /get\_city

* 请求方法：get
* 接口名称：/get\_city
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“n\_province\_id”: //省的国家id，查询某省的市的依据

}

* 返回结果：json

{

code：0 //状态码，表示成功

data：[

{

“city\_name”: //市名

“city\_id”: //市的id（非国家id）

},

……….

]

}

* 注意事项：无

## 获取所有行业（顶级）列表 /get\_industrial

* 请求方法：post
* 接口名称：/get\_industrial
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

}

* 返回结果：json

{

code：0 //状态码，

data：[

{

“industrial\_name”: //行业名

“industrial\_id”: //行业id

},

……….

]

}

* 注意事项：无

# 二．校友信息 增删改查API

## 分页获取校友信息列表 /get\_allschoolmate

* 请求方法：post
* 接口名称：/get\_allschoolmates
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“page\_toal”:”10” //显示的条数

“start”: ”1” //起始号

}

* 返回结果类型：json格式
* 返回结果字段：

{

total：10 //总共的条数，

start\_item：100 //当前页开始显示的条数编号，

end\_item：110 //当前页显示的最后一条编号，

code：0 //状态码，0表示返回成功

data：

[

{

update\_time: “2017-6-14 14:20:22” //更新时间

user\_id: “1”, //校友id

author\_id: “1”, //权限组id

user\_name：”左占平”, //校友姓名

user\_gender：”男” , //性别

user\_nation：”汉族”, //民族

user\_birthday：”1980-09-08”, //出生年月

user\_post: “销售经理”,

user\_workunit: “兰州电机厂”,

industrial\_id: “”,

industrial\_name: “”,

user\_workplace: “龚家湾”,

user\_province:”甘肃”,

user\_city: “兰州”,

user\_officephone: “”,

user\_wechat: “”,

user\_qq: “”,

user\_email: “”,

user\_cellphone: “”,

user\_education: [ //校友学籍信息，通过数组的方式全发过来

{

user\_edu: “本科”,

collage\_admin\_id:”12”,

collage\_admin\_name:”经济管理学院”,

collage\_edu\_id:”12”,

collage\_edu\_name:”经济管理学院”,

specialty\_id: “42”,

specialty\_name: “会计学”,

entrance\_year: “2001”,

grade\_year: “2005”,

class: “1”,

update\_time: “2017-6-14 14:20:22”

}, //一个校友的学籍信息结束

……….

]

}, //一个校友信息结束

……….

]

}

* 注意事项： 每条校友数据，应包含该校友的概要信息、学籍信息和工作信息。也就是说，除了历史信息以外，其他所有校友的可显示数据（来自数据库校友信息表和学籍信息表）都一并打包发过来。

## 多条件查询校友信息 /get\_alumnus\_by\_terms

* 请求方法：post
* 接口名称：/get\_alumnus\_by\_terms
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“terms”：tt //tt是查询条件，格式固定，在ajax请求前进行赋值和组装

“pages”： {  
 page: 1,  
 page\_row\_num: 10,  
 page\_total\_num: 1,  
 row\_total\_num: 0  
 }

}

//以下是tt的构建方式：

t = {

“user\_info”:

{ //这部分是校友个人及其工作信息条件，每个键值对的值默认为 “”

“user\_name”: “”,

“user\_gender”: “”,

“user\_age”: 29, //数值没有双引号

“user\_nation”: “”,

“user\_workunit”: “”,

“user\_post”: “”,

“user\_province”: “”,

“user\_city”: “”

},

“user\_education”:

{ //这是校友学籍信息的条件，每条默认值也是 “”

"specialty\_name": "计算机科学与技术系",  
"entrance\_year1": 2000,  
"entrance\_year2": "",  
"collage\_admin\_name": "",  
"user\_edu": ""

}

]

tt = JSON.stringify(t) //tt构建完毕

* 返回结果：json

{

code：0 //状态码，

data：[

{

“user\_name”: //校友姓名

“user\_edu”: //校友学籍

………. //包括校友基本信息和学籍信息

},

……….

],

pages: {  
 page:  
 page\_row\_num:   
 page\_total\_num:   
 row\_total\_num:   
}

}

* 注意事项：tt的构建单独做，t中每个key的默认值为 “”

## 删除某条校友信息/delete\_alumnus\_by\_id

* 请求方法：post
* 接口名称：/delete\_alumnus\_by\_id
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”: user\_pk, //用户登录时返回的user\_pk，存储在localstorage

“token”：token, //用于验证的token值，存储在localstorage

“alumnus\_ids”: data, //data的构建如下

}

//以下是data的构建方式：

da = [725,741] //要删除的校友的id，可删除多个校友，同事删除他们的学籍

data = JSON.stringify(da) //data构建完毕

* 返回结果：json

{

code：0 ，//状态码，

message：“”//成功失败信息

}

注意事项：data的构建单独做

## 修改某条校友信息/ update\_alumnus\_by\_id

* 请求方法：post
* 接口名称：/ update\_alumnus\_by\_id
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”: user\_pk, //用户登录时返回的user\_pk，存储在localstorage

“token”：token, //用于验证的token值，存储在localstorage

“alumnus\_id”: id, //要更新的校友ID。如果校友ID==user\_pk，说明是用户自己更新自己。如果校友ID != user\_pk，说明是管理员更新其他用户信息，需要进行权限控制

“alumnus\_data”：data //data是要更新的校友数据，在ajax请求前进行赋值和组装

}

//以下是data的构建方式：

da = {

“user\_info”: { //用户基本信息，来自于用户信息表

“user\_id”: “”,

“user\_name”: “”,

“user\_gender”: “”,

“user\_birthday”: “”,

“user\_nation”: “”,

“user\_officephone”: “”,

“user\_cellphone”: “”,

“user\_qq”: “”,

“user\_wechat”: “”,

“user\_email”: “”,

“user\_post”: “”,

“user\_workunit”: “”,

“industrial\_id”: “”,

“industrial\_name”: “”,

“user\_workplace”: “”,

“n\_province\_id”: “”,

“user\_province”: “”,

“user\_city”: “”,

“city\_id”: “”

},

“user\_education”: [

//用户的学籍信息，以对象数组的方式发送。

//如果用户没有更新学籍信息，则不发送此字段。这样可以避免无意义的传输和数据库查询。

{ //第一学历

“edu\_id”: “”,

“user\_edu”: “”,

“user\_id”: “”,

“collage\_admin\_id”: “”,

“collage\_admin\_name”: “”,

“collage\_edu\_id”: “”,

“collage\_edu\_name”: “”,

“specialty\_id”: “”,

“specialty\_name\_cn”: “”,

“entrance\_year”: “”,

“grade\_year”: “”,

“classes”: “”

},

…….

]

} //data封装结束

data = JSON.stringify(da) //data构建完毕

* 返回结果：json

{

code：0 ，//状态码，

message：“”//成功失败信息

}

注意事项：data的构建单独做，data中每个key的默认值为 “”，空值也会更新到数据库

## 获取某个校友的历史记录信息

~~请求方法：get~~

~~接口名称：/get\_history\_info~~

~~参数：{~~

~~“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage~~

~~“token”：token值 //用于验证的token值，存储在localstorage~~

~~“page\_toal”:”10” //显示的条数~~

~~}~~

~~返回结果类型：json格式~~

~~返回结果字段：~~

~~{~~

~~total： //总共的条数，~~

~~start\_item： //当前页开始显示的条数编号，~~

~~end\_item： //当前页显示的最后一条编号，~~

~~code：0 //状态码，0表示返回成功，~~

~~data：[~~

~~{~~

~~update\_time：”2017-1-2” //更新日期~~

~~user\_name：”~~~~左占平” //校友姓名~~

~~user\_workplace：”~~~~兰州理工大学” //工作单位~~

~~user\_province：”~~~~甘肃省” //所在省~~

~~user\_city：”~~~~兰州市” //所在市~~

~~user\_officephone：”~~~~0931-7723567” //办公电话~~

~~user\_cellphone：”13587980145” //手机~~

~~user\_wechat：”~~~~zuozhanping123” //微信号~~

~~user\_qq：”~~~~1872345673” //QQ号~~

~~},~~

~~……….~~

~~]~~

~~}~~

* ~~注意事项：无~~

# 三．学校信息 增删改查API

## ~~获取学校名称~~

~~请求方法：get~~

~~接口名称：/get\_school\_info~~

~~参数：{~~

~~“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage~~

~~“token”：token值 //用于验证的token值，存储在localstorage~~

~~}~~

~~返回结果类型：json格式~~

~~返回结果字段：~~

~~{~~

~~code：0 //状态码，0表示返回成功，~~

~~data：[~~

~~{~~

~~School\_name: “兰州理工大学” #学校名称~~

~~}~~

~~]~~

~~}~~

## ~~获取学院列表~~

~~请求方法：get~~

~~接口名称：/get\_collages\_info~~

~~参数：{~~

~~“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage~~

~~“token”：token值 //用于验证的token值，存储在localstorage~~

~~“page\_toal”:”10” //显示的条数~~

~~}~~

~~返回结果类型：json格式~~

~~返回结果字段：~~

~~{~~

~~total： //总共的条数，~~

~~start\_item： //当前页开始显示的条数编号，~~

~~end\_item： //当前页显示的最后一条编号，~~

~~code：0 //状态码，0表示返回成功，~~

~~data：[~~

~~{~~

~~},~~

~~]~~

~~}~~

## 根据学院ID，获取该学院下专业信息

## 添加学院

## 添加专业

# 四．校友组织 增删改查API

## 获取所有校友组织列表 /get\_organization

* 请求方法：post
* 接口名称：/get\_organization
* 是否需要登录：是
* 访问权限限制：
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

}

* 返回结果：json

{

code：0 //状态码，

data：[

{

org\_city: “兰州市” //兰州市

org\_contact\_info: ”null” //组织联系人信息

org\_found\_time: "2010-11-01" //组织创建日期

org\_id: ”1” //校友组织编号

org\_member\_num:”89” //组织会员数

org\_name: “兰州理工大学兰州校友会 //校友会名称 org\_province: “甘肃省” //校友会所在地

user\_admin\_name:”薛朱鹏” //校友组织管理员

},

……….

]

}

* 注意事项：无

## 多条件搜索校友组织列表

## 修改某校友组织

## 删除某校友组织

## 根据校友组织ID，查询所属成员信息

## 向校友组织添加成员

## 从校友组织删除成员

# 五．权限组 增删改查API

## 获取所有权限组列表/get\_authgroup\_all

* 请求方法：post
* 接口名称：/get\_authgroup\_all
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

}

* 返回结果：json

{

code：0 //状态码，

data：[

{

author\_group: “学院负责人” //权限组名

author\_id: ”4” //权限组号

create\_time: " 2017-06-29 00:38:40" //创建时间

update\_time: " 2017-06-29 00:38:40" //更新时间

},

……….

]

}

* 注意事项：无

## 修改权限组的基本信息/update\_authgroup\_baseinfo\_by\_id

* 请求方法：post
* 接口名称：/update\_authgroup\_baseinfo\_by\_id
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“author\_group\_baseinfo”：{

“author\_id”：4

“author\_group”： “超级管理员”

“is\_enable”:

“is\_edit”:

“is\_del”:

“um\_enable”：

“um\_add”：

“um\_del\_up”:

“um\_del\_up\_space”:

“um\_del\_up\_time”:

“um\_search”：

“um\_search\_space”：

“um\_search\_time”:

“um\_is\_export\_excel”:

“um\_is\_import\_excel”:

“as\_enable”：

“uim\_enable”：

“uim\_search”:

“uim\_add\_del\_up”:

“om\_enable”:

“om\_search”：

“om\_add\_del\_up”:

“am\_enable”：

}

}

* 返回结果：json

{

code：0 //状态码，

message：修改成功

}

* 注意事项：无

## 修改某权限组的权限（包括所有权限项）/update\_authgroup\_items\_by\_id

* 请求方法：post
* 接口名称：/update\_authgroup\_items\_by\_id
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“author\_group\_items”：{

“author\_id”：4

“author\_group”： “超级管理员”

}

}

* 返回结果：json

{

code：0 //状态码，

message：修改成功

}

* 注意事项：无

## 检查某权限组是否已有用户使用/check\_authgroup\_use\_by\_id

* 请求方法：post
* 接口名称：/check\_authgroup\_use\_by\_id
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“author\_id”： “新权限组名”

}

* 返回结果：json

{

code：0 //状态码，0表示成功，1表示不成功

data：{

author\_id://权限id

is\_use://是否被使用，0未使用，1有使用

use\_count:被授此权限组用户数量

}

}

* 注意事项：无

## 删除某权限组（如果该权限组已经被使用，则相关用户权限组变为默认权限组）/delete\_authgroup\_by\_id

* 请求方法：post
* 接口名称：/delete\_authgroup\_by\_id
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“author\_id”： “权限组id”

}

* 返回结果：json

{

code：0 //状态码，0表示成功，1表示不成功

message：//删除信息

}

* 注意事项：无

## 根据权限组ID，查询所属成员信息/ get\_alumnus\_by\_author\_id

* 请求方法：post
* 接口名称：/ get\_alumnus\_by\_author\_id
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“terms”：tt //tt是查询条件，格式固定，在ajax请求前进行赋值和组装

“pages”： {  
 page: 1,  
 page\_row\_num: 10,  
 page\_total\_num: 1,  
 row\_total\_num: 0  
 }

}

//以下是tt的构建方式：

t = {

“user\_info”:

{ //这部分是校友个人及其工作信息条件，每个键值对的值默认为 “”

“user\_author”: //非常重要，权限id可为空（即查询所有）

“user\_name”: “”,

“user\_gender”: “”,

“user\_age”: 29, //数值没有双引号

“user\_nation”: “”,

“user\_workunit”: “”,

“user\_post”: “”,

“user\_province”: “”,

“user\_city”: “”

},

“user\_education”:

{ //这是校友学籍信息的条件，每条默认值也是 “”

"specialty\_name": "计算机科学与技术系",  
"entrance\_year1": 2000,  
"entrance\_year2": "",  
"collage\_admin\_name": "",  
"user\_edu": ""

}

]

tt = JSON.stringify(t) //tt构建完毕

* 返回结果：json

{

code：0 //状态码，

data：[

{

“user\_name”: //校友姓名

“user\_edu”: //校友学籍

………. //包括校友基本信息和学籍信息

},

……….

],

pages: {  
 page:  
 page\_row\_num:   
 page\_total\_num:   
 row\_total\_num:   
}

}

* 注意事项：无

## 修改多个用户所属的权限组/ update\_alumnus\_authgroup\_by\_ids

* 请求方法：post
* 接口名称：/ update\_alumnus\_authgroup\_by\_ids
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“alumnus\_ids”： JSON.stringify([34,67,456])

“author\_id”： 2 //此处用权限组号

}

* 返回结果：json

{

code：0 //状态码，0表示成功，1表示不成功

message：修改成功

}

* 注意事项：无

## 新建权限组（可以克隆已有的权限组）/add\_authorgroup

* 请求方法：post
* 接口名称：/add\_authorgroup
* 是否需要登录：是
* 访问权限限制：最高管理员
* 请求参数：

{

“user”:user\_pk //用户登录时返回的user\_pk，存储在localstorage

“token”：token值 //用于验证的token值，存储在localstorage

“data”：{

“author\_group”： “新权限组名”

“clone\_id”：3 //克隆权限组的id

}

}

* 返回结果：json

{

code：0 //状态码，0表示成功，1表示不成功

message：修改成功

}

* 注意事项：无