API文档

请求编码: UTF-8

请求响应格式（header中Content-Type）: 'application/json'

请求地址格式: "http://aliyun.1230123.xyz:20080/api/v1/*{Module}*/*{Method}*?token=*{token}*"

所有的请求，凡是经过了代码逻辑捕获处理的，HTTP状态码均为200。不通过HTTP状态码表示请求是否成功。

所有成功的请求，按如下格式返回，下文中所示JSON无特殊说明均表示成功状态的data成员的要求。

```
{
  "success": true,
  "errCode": 0,
  "data": {
    // 填充对应的数据
    // 其中字符串形式"{DATA}_EMPTY"表示该字符串可为空串，但是必须存在该成员。其余的表示不能为空串
  }
}
```

所有错误的请求，按如下格式返回，下文中将以表格形式说明错误码对应的含义。

```
{
  "success": false,
  "errCode": 9, // 仅为示例，应替换为对应的错误码，类型为整型数字，不是字符串
  "data": {} // 仍要存在data成员，类型为空对象
}
```

errCode（后端与前端交换的数据）	errDescription（前端展示解释）
99999	未知错误

除Auth模块的方法之外，其余请求均会在query中传递token。各请求在进行处理前，都需要对token是否合法（即在Redis中是否存在与 *{用户ID}* 的对应关系）进行检查。普通用户和管理员用户应分别存储，以方便应对Admin模块的请求应判断该token是否对应管理员用户。若不合法，统一返回如下JSON。

```
{
  "success": false,
  "errCode": 99991,
  "data": {}
}
```

errCode	errDescription
99991	登录信息失效，请重新登录

1. Auth

1. login

Method	Content-Type	描述
POST	multipart/form-data	普通用户请求登录

表单内容（注：**Value类型并非Content Type，下同**）

Key	Value类型	描述
username	String	用户名
password	String	密码

成功返回

```
{
  "data": {
    "token": "{token}" // 返回已在Redis中保存的token
  }
}
```

错误代码

errCode	errDescription
100101	账号或密码错误
100102	该账号已被封禁

2. register

Method	Content-Type	描述
POST	multipart/form-data	普通用户请求注册

表单内容

Key	Value类型	描述
username	String	用户名
password	String	密码

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
100201	该用户名已被占用
100202	用户名或密码中包含非法字符

3. adminLogin

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求登录

表单内容（注：**Value类型并非Content Type，下同**）

Key	Value类型	描述
adminname	String	管理员用户名
password	String	密码

成功返回

```
{
  "data": {
    "token": "{token}" // 返回已在Redis中保存的token，该token应分别存储以标记这个token对应的是管理员用户
  }
}
```

错误代码

errCode	errDescription
100301	账号或密码错误

4. logout

Method	Content-Type	描述
POST	multipart/form-data	普通用户请求登出

成功返回。成功后该token从Redis数据库中删除，之后所有使用该token的请求均无效。若该token本身即非法，无需进行其他操作

```
{
  "data": {} // 空对象
}
```

5. adminLogout

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求登出

成功返回。成功后该token从Redis数据库中删除，之后所有使用该token的请求均无效。若该token本身即非法，无需进行其他操作

```
{
  "data": {} // 空对象
}
```

2. Admin

1. getAllUser

Method	Content-Type	描述
GET	/	管理员获取所有的***普通用户***信息

成功返回

```
{
  "data": {
    "users": [
      {
        "userid": userid, // 整型，{用户ID}
        "username": "{username}",  // 用户名
        "avatarurl": "{avatarurl}", // 头像图片完整URL地址，若用户未设置则为默认头像
        "studentid": "{studentid}_EMPTY", // 学号
        "isblock": true | false // 是否已经封禁
      },
      ...
      // 当前系统中没有普通用户，则该数组为空
    ]
  }
}
```

2. blockUser

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求封禁***普通用户***

注：封禁普通用户时，若该普通用户在Redis已经有token，要进行销毁。

表单内容

Key	Value类型	描述
userid	Integer	*{用户ID}*

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

注：若请求的*{用户ID}*对应的用户已被封禁，不会产生错误，确保其仍然处于封禁状态即可

errCode	errDescription
200201	请求封禁的普通用户不存在

3. unblockUser

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求解封***普通用户***

表单内容

Key	Value类型	描述
userid	Integer	*{用户ID}*

成功返回。

```
{
  "data": {} // 空对象
}
```

错误代码

注：若请求的*{用户ID}*对应的用户未被封禁，不会产生错误，确保其仍然处于未被封禁状态即可

errCode	errDescription
200301	请求解封的普通用户不存在

4. getAllExercise

Method	Content-Type	描述
GET	/	管理员获取所有的题目，**含被封禁的题目**

请求参数

Key	Value类型	描述
page	Integer	管理员获取所有的题目结果第 {page} 页

注：一次只返回最多20个题目。即查询的结果分页表示，按照 *{题目ID}* 从大到小进行排列，页大小为20

注：必须提供page，默认page参数为1

成功返回

```
f{
  "data": {
    "pages": pages, // 整型，表示总共的页数
    "thispage": [
      {
        "exerciseid": exerciseid, // 整型，{题目ID}
        "createusername": "{username}", // 创建该题目的用户的用户名
        "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
        "title": "{title}", // 题目的标题
        "content": "{content}", // 题目的正文
        "option": [
          // type == 0（判断题）或 type == 10（填空题）
          // 空数组
          // type == 1（单选题）或 type == 2（多选题）
          "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
          "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
          ...
          // 至少有2个选项，最多26个选项
        ],
        "answer": [
          // type == 0（判断题）
          "A" | "B", // A表示正确，B表示错误
          // type == 1（单选题）或 type == 2（多选题）
          "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
          // type == 10（填空题）
          "{answer}" // 表示正确答案的字符串
        ],
        "tag": [
```

```
          {
            "tagid": tagid, // 所在的题目组的 {题目组ID}
            "tagname": "{tagname}" // 所在的题目组的名称
          }
          ...
          // 至少在一个题目组当中
        ],
        "isBlock": true | false // 是否已经封禁
      }
      ......
      // 若所请求的页没有题目，则该数组为空
    ]
  }
}
```

5. blockExercise

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求封禁题目

表单内容

Key	Value类型	描述
exerciseid	Integer	*{题目ID}*

成功返回。

```
{
  "data": {} // 空对象
}
```

错误代码

注：若请求的 *{题目ID}* 对应的用户已被封禁，不会产生错误，确保其仍然处于封禁状态即可

errCode	errDescription
200501	请求封禁的题目不存在

6. unblockExercise

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求解封题目

表单内容

Key	Value类型	描述
exerciseid	Integer	*{题目ID}*

成功返回。

```
{
  "data": {} // 空对象
}
```

错误代码

注：若请求的 *{题目ID}* 对应的用户未被封禁，不会产生错误，确保其仍然处于未被封禁状态即可

errCode	errDescription
200601	请求解封的题目不存在

7. getAllAdmin

Method	Content-Type	描述
GET	/	管理员获取所有的***管理员用户***信息

成功返回

```
{
  "data": {
    "admins": [
      {
        "adminid": adminid, // 整型，{管理员ID}
        "adminname": "{adminname}",  // 管理员用户名
      },
      ...
      // 至少存在超级管理员用户root
    ]
  }
}
```

8. createAdmin

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求添加***管理员用户***

表单内容

Key	Value类型	描述
adminname	String	管理员用户名
password	String	密码

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
200801	该管理员用户名已被占用
200802	该管理员用户名或密码中包含非法字符

9. deleteAdmin

Method	Content-Type	描述
POST	multipart/form-data	管理员用户请求删除***管理员用户***

注：删除管理员用户时，若该管理员用户在Redis已经有token，要进行销毁。

表单内容

Key	Value类型	描述
adminid	Integer	*{管理员ID}*

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
200901	请求删除的管理员用户不存在
200902	请求删除的管理员用户为超级管理员root

3. UserInfo

1. getCurrentUserInfo

Method	Content-Type	描述
GET	/	获取当前用户的信息

成功返回

```
{
  "data": {
    "username": "{username}",  // 用户名
    "avatarurl": "{avatarurl}", // 头像图片完整URL地址，若用户未设置则为默认头像
    "studentid": "{studentid}_EMPTY", // 学号
  }
}
```

2. updateAvatar

Method	Content-Type	描述
POST	multipart/form-data	用户修改头像

表单内容

Key	Value类型	描述
newavatar	File	用户的新头像

成功返回

```
{
  "data": {
    "avatarurl": "{avatarurl}", // 新的头像图片完整URL地
  }
}
```

错误代码

errCode	errDescription
300201	新头像超出服务器限制

3. updateStudentID

Method	Content-Type	描述
POST	multipart/form-data	用户修改学号

表单内容

Key	Value类型	描述
newstudentid	String	用户的新学号

成功返回

```
{
  "data": {
    "studentid": "{studentid}", // 新的学号
  }
}
```

错误代码

errCode	errDescription
300301	新学号中包含非法字符

4. Exercise

1. createExercise

Method	Content-Type	描述
POST	application/json	创建题目

请求内容

```
{
  "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
  "title": "{title}", // 题目的标题
  "content": "{content}", // 题目的正文
  "option": [
    // type == 0（判断题）或 type == 10（填空题）
    // 空数组
    // type == 1（单选题）或 type == 2（多选题）
    "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
    "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
    ...
    // 至少有2个选项，最多26个选项
  ],
  "answer": [
```

```
    // type == 0（判断题）
    "A" | "B", // A表示正确，B表示错误
    // type == 1（单选题）或 type == 2（多选题）
    "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
    // type == 10（填空题）
    "{answer}" // 表示正确答案的字符串
  ],
  "tag": [
    tagid1 // 所在的题目组 {题目组ID}
    ...
    // 至少有一个题目组
  ]
}
```

成功返回

```
{
  "data": {
    "exerciseid": exerciseid, // 整型，{题目ID}
  }
}
```

错误代码

errCode	errDescription
400101	题目类型不受支持
400102	请求添加到的题目组不存在
400103	标题中包含非法字符或合规审查失败
400104	正文中包含非法字符或合规审查失败
400105	选项中包含非法字符或合规审查失败
400106	答案中包含非法字符或合规审查失败（仅填空题）

2. updateExercise

Method	Content-Type	描述
POST	application/json	更新 {题目ID} 对应的题目

请求内容

```
{
  "exerciseid": exerciseid, // 整型，{题目ID}
  "newdata": {
    "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
```

```
    "title": "{title}", // 题目的标题
    "content": "{content}", // 题目的正文
    "option": [
      // type == 0（判断题）或 type == 10（填空题）
      // 空数组
      // type == 1（单选题）或 type == 2（多选题）
      "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
      "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
      ...
      // 至少有2个选项，最多26个选项
    ],
    "answer": [
      // type == 0（判断题）
      "A" | "B", // A表示正确，B表示错误
      // type == 1（单选题）或 type == 2（多选题）
      "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
      // type == 10（填空题）
      "{answer}" // 表示正确答案的字符串
    ],
    "tag": [
      tagid1 // 所在的题目组 {题目组ID}
      ...
      // 至少有一个题目组
    ]
  }
}
```

成功返回

```
{
  "data": {
    "exerciseid": exerciseid, // 整型，{题目ID}
  }
}
```

错误代码

注：若该题目已被封禁，同样可以修改

errCode	errDescription
400201	当前用户无法管理该题目
400202	题目类型不受支持
400203	请求添加的题目组不存在
400204	标题中包含非法字符或合规审查失败
400205	正文中包含非法字符或合规审查失败
400206	选项中包含非法字符或合规审查失败
400207	答案中包含非法字符或合规审查失败（仅填空题）

3. getReachableExercise

Method	Content-Type	描述
GET	/	获取当前用户加入的所有的共享群组中的所有正在共享的题目，以及自己创建的所有的题目，**不包括被封禁的题目**

注：公共共享区为 *{共享群组ID}* 为1的群组，每个用户在注册时即加入，且不可退出。

注：一次只返回最多20个题目。即查询的结果分页表示，按照 *{题目ID}* 从大到小进行排列，页大小为20

请求参数

Key	Value类型	描述
page	Integer	获取的所有可查看的题目结果第 {page} 页

注：必须提供page，默认page参数为1

成功返回

```
{
  "data": {
    "pages": pages, // 整型，表示总共的页数
    "thispage": [
      {
        "exerciseid": exerciseid, // 整型，{题目ID}
        "createusername": "{username}", // 创建该题目的用户名
        "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
        "title": "{title}", // 题目的标题
        "content": "{content}", // 题目的正文
        "option": [
          // type == 0（判断题）或 type == 10（填空题）
          // 空数组
```

```
          // type == 1 (单选题) 或 type == 2 (多选题)
          "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
          "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
          ...
          // 至少有2个选项，最多26个选项
        ],
        "answer": [
          // type == 0 (判断题)
          "A" | "B", // A表示正确，B表示错误
          // type == 1 (单选题) 或 type == 2 (多选题)
          "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
          // type == 10 (填空题)
          "{answer}" // 表示正确答案的字符串
        ],
        "tag": [
          {
            "tagid": tagid, // 所在的题目组的 {题目组ID}
            "tagname": "{tagname}" // 所在的题目组的名称
          }
          ...
          // 至少在一个题目组当中
        ],
      }
      ......
      // 若所请求的页没有题目，则该数组为空
    ]
  }
}
```

4. getExerciseByID

Method	Content-Type	描述
GET	/	获取 {题目ID} 对应的题目

请求参数

Key	Value类型	描述
exerciseid	Integer	获取的题目的 {题目ID}

注：建议不检查该用户是否有权限访问这个题目

成功返回

```
{
  "data": {
    "isBlock": true | false, // 是否已经封禁
    "data": {
      // 题目存在且未被封禁
```

```
      "exerciseid": exerciseid, // 整型，{题目ID}
      "createusername": "{username}", // 创建该题目的用户名
      "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
      "title": "{title}", // 题目的标题
      "content": "{content}", // 题目的正文
      "option": [
        // type == 0（判断题）或 type == 10（填空题）
        // 空数组
        // type == 1（单选题）或 type == 2（多选题）
        "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
        "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
        ...
        // 至少有2个选项，最多26个选项
      ],
      "answer": [
        // type == 0（判断题）
        "A" | "B", // A表示正确，B表示错误
        // type == 1（单选题）或 type == 2（多选题）
        "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
        // type == 10（填空题）
        "{answer}" // 表示正确答案的字符串
      ],
      "tag": [
        {
          "tagid": tagid, // 所在的题目组的 {题目组ID}
          "tagname": "{tagname}" // 所在的题目组的名称
        }
        ...
        // 至少在一个题目组当中
      ],
      // 题目不存在或存在但是已被封禁，可以返回空对象
    }
  }
}
```

errCode	errDescription
400401	无此ID对应的题目

5. searchExercise

Method	Content-Type	描述
GET	/	在用户加入的所有的共享群组中的所有正在共享的题目，以及自己创建的所有的题目中搜索匹配的题目，**不包括被封禁的题目**

注：公共共享区为 *{共享群组ID}* 为1的群组，每个用户在注册时即加入，且不可退出。

注：一次只返回最多20个题目。即查询的结果分页表示，按照 *{题目ID}* 从大到小进行排列，页大小为20

请求参数

Key	Value类型	描述
page	Integer	获取的所有可查看的题目结果第 {page} 页
type	String	搜索的类型。"title"表示匹配题目的标题，"tag"表示匹配题目所在的题目组的名称
pattern	String	需要匹配的字符串，*暂只进行字符级的匹配*

注：必须提供page，默认page参数为1

成功返回

```
{
  "data": {
    "pages": pages, // 整型，表示总共的页数
    "thispage": [
      {
        "exerciseid": exerciseid, // 整型，{题目ID}
        "createusername": "{username}", // 创建该题目的用户名
        "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
        "title": "{title}", // 题目的标题
        "content": "{content}", // 题目的正文
        "option": [
          // type == 0（判断题）或 type == 10（填空题）
          // 空数组
          // type == 1（单选题）或 type == 2（多选题）
          "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
          "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
          ...
          // 至少有2个选项，最多26个选项
        ],
        "answer": [
          // type == 0（判断题）
          "A" | "B", // A表示正确，B表示错误
          // type == 1（单选题）或 type == 2（多选题）
          "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
          // type == 10（填空题）
          "{answer}" // 表示正确答案的字符串
        ],
        "tag": [
          {
            "tagid": tagid, // 所在的题目组的 {题目组ID}
            "tagname": "{tagname}" // 所在的题目组的名称
          }
          ...
          // 至少在一个题目组当中
        ],
```

```
      }
      ......
      // 若所请求的页没有题目，则该数组为空
    ]
  }
}
```

6. OCR

Method	Content-Type	描述
POST	multipart/form-data	请求OCR识别

表单内容

Key	Value类型	描述
file	File	请求OCR识别的文件，可以是图片，可以是PDF
page	Integer	请求OCR识别的页数，用来指定进行OCR识别的的页码，默认前端传入1，对于图片该值没有意义

成功返回

```
{
  "data": {
    "text": "{text}" // OCR的结果
  }
}
```

错误代码

errCode	errDescription
400601	*阿里云OCR服务错误（未使用云服务，已弃用）*
400602	本地OCR服务错误

7. getCommentByID

Method	Content-Type	描述
GET	/	获取 {题目ID} 对应的题目的讨论区

请求参数

Key	Value类型	描述
exerciseid	Integer	获取讨论区的题目的 {题目ID}

注：一次返回所有讨论。即查询的结果不分页，按照发表时间顺序从最近至最远排列

注：建议不检查该用户是否有权限访问这个题目

成功返回

```
{
  "data": {
    "comment": [
      {
        "createusername": "{username}", // 创建该讨论的的用户名
        "createavatarurl": "{avatarurl}", // 创建该讨论的的用户头像
        "time": "{YYYY-MM-DD HH:MM:SS}", // 讨论创建的时间，例如2024-01-01 23:59:59
        "content": "{content}", // 讨论的内容
      }
      ......
      // 若所请求的页题目没有讨论内容，则该数组为空
    ]
  }
}
```

8. addComment

Method	Content-Type	描述
POST	multipart/form-data	将讨论添加到 {题目ID} 对应的题目的讨论区

表单内容

Key	Value类型	描述
exerciseid	Integer	要添加到的题目的 {题目ID}
comment	String	讨论的内容

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
400801	无此ID对应的题目

5. Tag

1. createTag

Method	Content-Type	描述
POST	multipart/form-data	用户创建新的题目组

表单内容

Key	Value类型	描述
tagname	String	题目组名称

!!!CHEAT!!!注：tagname在演示时保证形如 {subject}-Chapter{chadpter num}

成功返回

```
{
  "data": {
    "tagid": tagid // 创建的题目组的 {题目组ID}
  }
}
```

错误代码

注：{题目组ID}是全局唯一的，但是题目组名称重复的检查范围仅为该用户创建的所有的题目组

errCode	errDescription
500101	请求创建的题目组名称已被占用

2. addExerciseToTag

Method	Content-Type	描述
POST	multipart/form-data	用户将题目加入到题目组中

表单内容

Key	Value类型	描述
tagid	Integer	题目组的 {题目组ID}
exerciseid	Integer	要加入到题目组的 {题目ID}

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
500201	题目组不存在或*当前用户无法管理该题目组*
500202	请求加入到题目组的题目不存在

3. getExerciseFromTag

Method	Content-Type	描述
GET	/	获取题目组中的所有题目，**不包括被封禁的题目**

注：公共共享区为 *{共享群组ID}* 为1的群组，每个用户在注册时即加入，且不可退出。

注：一次只返回最多20个题目。即查询的结果分页表示，按照 *{题目ID}* 从大到小进行排列，页大小为20

请求参数

Key	Value类型	描述
tagid	Integer	请求获取所有题目的题目组的 {题目组ID}
page	Integer	获取的所有可查看的题目结果第 {page} 页

注：必须提供page，默认page参数为1

注：建议不检查该用户是否有权限访问这个题目组

成功返回

```
{
  "data": {
    "pages": pages, // 整型，表示总共的页数
    "thispage": [
      {
        "exerciseid": exerciseid, // 整型，{题目ID}
```

```
        "createusername": "{username}", // 创建该题目的用户名
        "type": 0 | 1 | 2 | 10, // 题目的类型，0表示判断题，1表示单选题，2表示多选题，10表示填空题
        "title": "{title}", // 题目的标题
        "content": "{content}", // 题目的正文
        "option": [
          // type == 0（判断题）或 type == 10（填空题）
          // 空数组
          // type == 1（单选题）或 type == 2（多选题）
          "{optionA}", // A选项，不需要"A."开头直接返回选项内容即可
          "{optionB}" // B选项，不需要"B."开头直接返回选项内容即可
          ...
          // 至少有2个选项，最多26个选项
        ],
        "answer": [
          // type == 0（判断题）
          "A" | "B", // A表示正确，B表示错误
          // type == 1（单选题）或 type == 2（多选题）
          "A", "B", ... // 表示各个正确选项，单选题有且仅有一个，多选题至少2个，最多26个
          // type == 10（填空题）
          "{answer}" // 表示正确答案的字符串
        ],
        "tag": [
          {
            "tagid": tagid, // 所在的题目组的 {题目组ID}
            "tagname": "{tagname}" // 所在的题目组的名称
          }
          ...
          // 至少在一个题目组当中
        ],
      }
      ......
      // 若所请求的页没有题目，则该数组为空
    ]
  }
}
```

errCode	errDescription
500301	无此ID对应的题目组

4. getTagInfoByID

Method	Content-Type	描述
GET	/	获取题目组的信息

请求参数

Key	Value类型	描述
tagid	Integer	请求获取信息的题目组的 {题目组ID}

注：建议不检查该用户是否有权限访问这个题目组

成功返回

```
{
  "data": {
    "tagid": tagid, // 题目组的 {题目组ID}
    "tagname": "{tagname}", // 题目组的名称
    "createusername": "{username}", // 创建者的用户名
    "createavatarurl": "{avatarurl}", // 创建者的头像图片完整URL地址
    ]
  }
}
```

errCode	errDescription
500401	无此ID对应的题目组

5. getCurrentUserTag

Method	Content-Type	描述
GET	/	获取当前用户**创建的**所有题目组

成功返回

```
{
  "data": {
    "tag": [
      {
        "tagid": tagid, // 题目组的 {题目组ID}
        "tagname": "{tagname}" // 题目组的名称
      }
      ...
    ]
  }
}
```

6.Group

1. createGroup

Method	Content-Type	描述
POST	multipart/form-data	用户创建新的共享群组

表单内容

Key	Value类型	描述
groupname	String	共享群组名称

成功返回

```
{
  "data": {
    "groupid": groupid // 创建的共享群组的 {共享群组ID}
  }
}
```

2. deleteGroup

Method	Content-Type	描述
POST	multipart/form-data	用户删除共享群组

表单内容

Key	Value类型	描述
groupid	Integer	删除的共享群组的 {共享群组ID}

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
600201	删除的共享群组不存在
600202	当前用户无法管理该共享群组

3. joinGroup

Method	Content-Type	描述
POST	multipart/form-data	用户加入共享群组

表单内容

Key	Value类型	描述
groupid	Integer	加入的共享群组的 {共享群组ID}

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
600301	加入的共享群组不存在

4. exitGroup

Method	Content-Type	描述
POST	multipart/form-data	用户加入共享群组

表单内容

Key	Value类型	描述
groupid	Integer	退出的共享群组的 {共享群组ID}

注：若用户没有加入该共享群组，不会产生错误，确保其仍然处于未加入该共享群组的状态即可

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
600401	退出的共享群组不存在
600402	请求退出的共享群组为公共共享区
600403	请求退出的共享群组为自己创建的共享群组

5. addTagToGroup

Method	Content-Type	描述
POST	multipart/form-data	用户将题目组加入到共享群组中

表单内容

Key	Value类型	描述
groupid	Integer	要加入到的共享群组的 {共享群组D}
tagid	Integer	要加入的题目组的 {题目组ID}

成功返回

```
{
  "data": {} // 空对象
}
```

错误代码

errCode	errDescription
600501	题目组不存在或*当前用户无法管理该题目组*
600502	请求加入到共享群组不存在

6. getTagFromGroup

Method	Content-Type	描述
GET	/	获取共享群组中的所有题目组

注：公共共享区为 *{共享群组ID}* 为1的群组，每个用户在注册时即加入，且不可退出。

请求参数

Key	Value类型	描述
groupid	Integer	请求获取题目组的共享群组的 {共享群组ID}

注：建议不检查该用户是否有权限访问这个共享群组

成功返回

```
{
  "data": {
    "tag": [
      {
        "tagid": tagid, // 题目组的 {题目组ID}
        "tagname": "{tagname}", // 题目组的名称
        "createusername": "{username}", // 创建者的用户名
        "createavatarurl": "{avatarurl}" // 创建者的头像图片完整URL地址
      }
      ...
    ]
  }
}
```

errCode	errDescription
600601	无此ID对应的共享群组

7. getGroupInfoByID

Method	Content-Type	描述
GET	/	获取共享群组中的信息

注：公共共享区为 *{共享群组ID}* 为1的群组，每个用户在注册时即加入，且不可退出。

请求参数

Key	Value类型	描述
groupid	Integer	请求获取信息的共享群组的 {共享群组ID}

注：建议不检查该用户是否有权限访问这个共享群组

成功返回

```
{
  "data": {
    "groupid": groupid, // 共享群组的 {共享群组ID}
    "groupname": "{groupname}", // 共享群组的名称
    "createusername": "{username}", // 创建者的用户名
    "createavatarurl": "{avatarurl}", // 创建者的头像图片完整URL地址
    ]
  }
}
```

errCode	errDescription
600701	无此ID对应的共享群组

8. getCurrentUserGroup

Method	Content-Type	描述
GET	/	获取当前用户**创建和加入的**所有共享群组

成功返回

```
{
  "data": {
    "group": [
      {
        "groupid": groupid, // 共享群组的 {共享群组ID}
        "groupname": "{groupname}", // 共享群组的名称
        "createusername": "{username}", // 创建者的用户名
        "createavatarurl": "{avatarurl}", // 创建者的头像图片完整URL地址
      }
      ...
    ]
  }
}
```

7. Log

1. addWrongLog

Method	Content-Type	描述
POST	multipart/form-data	记录 {题目ID} 对应的题目做错一次

表单内容

Key	Value类型	描述
exerciseid	Integer	做错的题目的 {题目ID}

成功返回

```
{
  "data": {
    "timestamp": timestamp, // 整型，当前时间的时间戳
  }
}
```

错误代码

errCode	errDescription
700101	记录做错的题目不存在

2. addRightLog

Method	Content-Type	描述
POST	multipart/form-data	记录 {题目ID} 对应的题目做对一次

表单内容

Key	Value类型	描述
exerciseid	Integer	做对的题目的 {题目ID}

成功返回

```
{
  "data": {
    "timestamp": timestamp, // 整型，当前时间的时间戳
  }
}
```

错误代码

errCode	errDescription
700201	记录做对的题目不存在

3. getCurrentEvaluation

Method	Content-Type	描述
GET	/	获取当前学生的能力评价

成功返回

```
{
  "data": {
    "score": [
      score1, // 整型，对应日期的能力评价分数
      score2
      ...
    ],
    "time": [
      "{time1}",
      "{time2}" // 能力评价分数的日期
      ...
    ]
  }
}
```

4. getRecommendExercise

Method	Content-Type	描述
GET	/	在用户加入的所有的共享群组中的所有正在共享的题目，以及自己创建的所有的题目中推荐匹配的指定数量的题目

注：公共共享区为 *{共享群组ID}* 为1的群组，每个用户在注册时即加入，且不可退出。

请求参数

Key	Value类型	描述
pattern	String	需要匹配的字符串，与题目所在的题目组的名称进行匹配
quantity	Integer	推荐的题目的数量，若可推荐的且符合条件的题目中没有足够的题目，可少于所需数量

!!!CHEAT!!!注：可以不检查该用户是否有权限访问这个题目（保持与getExerciseByID方法相同即可）

!!!CHEAT!!!注：quantity在演示时为2或3，且可推荐的题目的数量是足够的，且可推荐的题目一定是用户可以访问的题目

成功返回

```
{
  "data": {
    "statisfy": true | false, // 是否满了推荐题目数量的需求
    "recommend": [
      exerciseid1 // 整型，{题目ID}
      ...
      // 至少返回一个题目
    ]
  }
}
```

错误代码

errCode	errDescription
700401	完成题目过少，无法推荐