详细设计说明书

1 引言

1.1 编写目的

该文档描述本项目的设计细节，供本项目开发人员、测试人员或相关专业人员阅读，以了解该项目设计细节，维持项目代码风格统一，保证程序实现的一致性。

1.2 背景

- **该软件系统名字：**“基于 Web 的教务管理系统”
- **任务提出者：**西安电子科技大学软件学院
- **开发者：**本团队（罗阳豪、赵善吉、梁一彤、白群迪）
- **用户：**西安电子科技大学软件学院学生、老师和教务管理人员
- **本程序的计算中心：**阿里云（www.aliyun.com）

1.3 定义

- 桌面 Web 客户端 /Web 前端 /前端 /客户端：在 Google Chrome 或 Mozilla Firefox 等桌面端浏览器可以直接通过因特网访问的该应用的用户图形交互界面；
- Web 应用后端 /后端：运行在 Linux 服务器上的，处理并响应用户请求，访问和操作数据库，实现业务逻辑的软件

1.4 参考资料

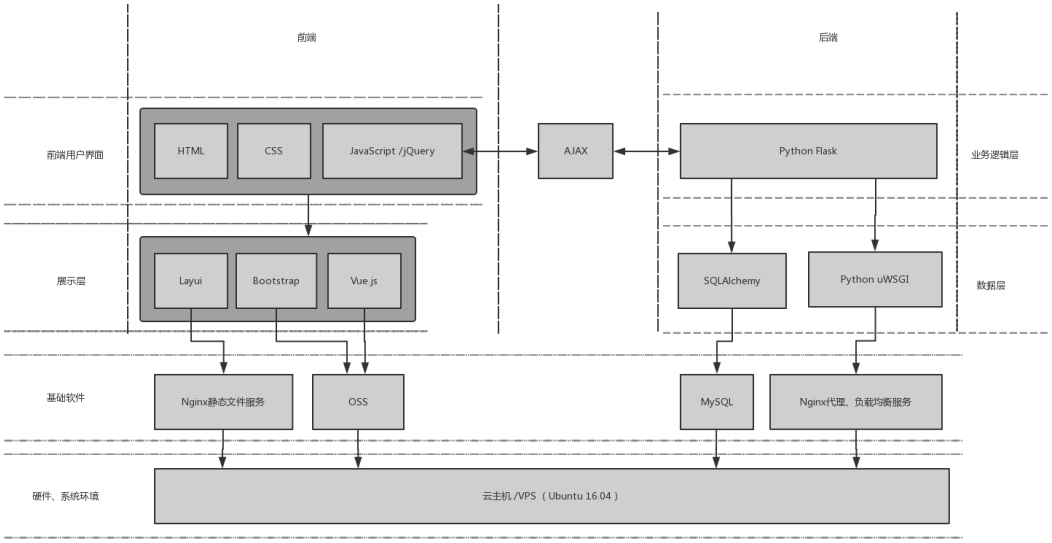
- Hypertext Transfer Protocol -- HTTP/1.1 (RFC 2616) https://tools.ietf.org/html/rfc2616
- Layui http://www.layui.com/doc/
- Boostrap https://getbootstrap.com/docs/4.1/getting-started/introduction/
- jQuery http://api.jquery.com/
- Vue.js https://cn.vuejs.org/v2/api/
- SQLAlchemy https://docs.sqlalchemy.org/en/latest/
- Flask http://flask.pocoo.org/docs/1.0/
- MySQL https://dev.mysql.com/doc/

2 程序系统的结构

该程序是一个典型的 Web 应用，严格按照前后端分离的方式开发。

前后端仅通过 AJAX 进行通信，他们分布在至少两个不同的服务器上。其中前端部署在一个 Nginx 的静态文件服务器，为了保证访问时效性，部分静态文件还被放置在 OSS 服务器上。而后端部署在另一个服务器上，使用 Nginx 作为反向代理及负载均衡服务器，以 uWSGI 作为 WSGI 容器。后端所依赖的 MySQL 服务器目前与后端部署在同一个服务器，但为了效率和稳定性也可以使用第三方的 MySQL 服务器。

该项目的体系结构的大致设计可见下图



3 “桌面 Web 客户端”设计说明

3.1 程序描述

该程序通过 Google Chrom 等 Web 浏览器运行，对用户提供友好的图形交互界面，获取用户操作并根据业务和功能需求与后端通信，实现数据可视化，和业务流程控制。

其由 HTML、CSS、JavaScript 等前端代码组成，使用前端 JavaScript 脚本 Vue.js，使用 JS 库 jQuery，使用 Bootstrap、Layui 等前端组件库或 CSS 框架

3.2 功能

1. 学生端：
   (1) 查看课表，按照时间顺序显示所参与的课程、上课时间、课室安排
   (2) 查看选课，显示所有可选课和选课状态
   (3) 选课，选课或取消选课

(4) 查看考试安排及考试成绩，按照时间顺序显示所参与考试科目、时间、试室安排
(5) 查看课程成绩，显示所有参与的课程、任课老师、课程成绩

2. 教师端：
(1) 查看课表，按照时间顺序显示所参与的课程、上课时间、课室安排
(2) 查看学生名单，显示某教学班全部学生学号、姓名、成绩
(3) 录入、修改成绩，修改所任教某教学班学生成绩表
(4) 查看考试安排及考试成绩，按照时间顺序显示所参与考试科目、时间、试室安排

3. 领导端：
暂未完成….

4. 普通管理员：
(1) 查看并增加、删除、修改账户
(2) 查看并增加、删除、修改考试安排
(3) 查看并增加、删除、修改课程信息
(4) 查看并增加、删除、修改教学班信息
(5) 排课，设置教学班与时间课室的关系，生成课表

5. Root 管理员：
(1) 查看数据库，在 Web 界面查看数据库中任何一表
(2) 修改数据，在 Web 界面可以修改数据库中任何一表

3.3 性能

时间开销

1. 静态页面加载时间： 少于 1s；
2. 后端数据加载超时时间：少于 5s
3. 数据渲染时间： 少于 1s

内存开销

内存开销取决与用户所使用的浏览器版本，

硬盘开销

理论上无需在客户端进行硬盘读写，但是其实际情况是依赖客户所使用浏览器的具体实现的

3.4 输人项

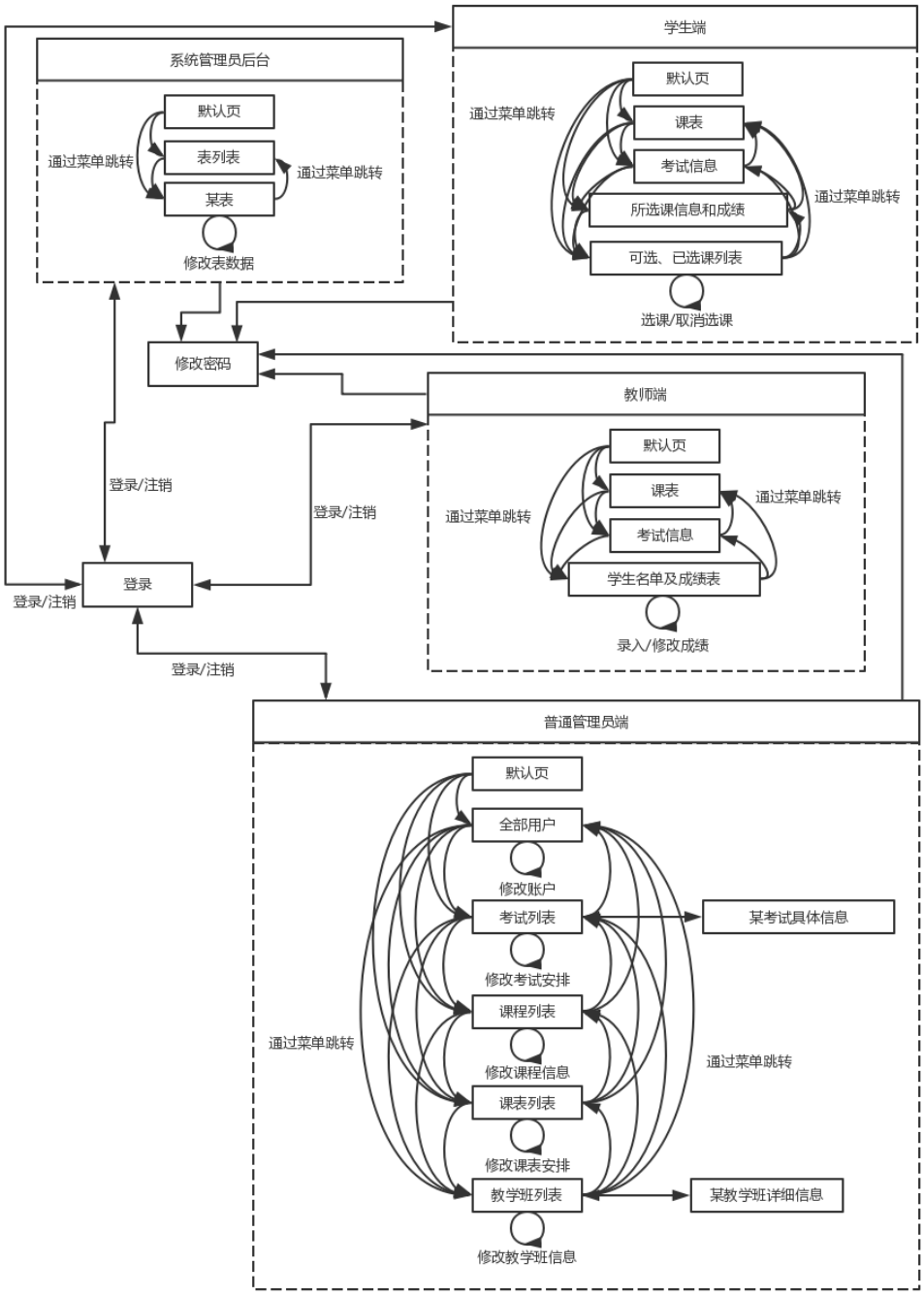
该程序的输入就是用户在浏览器上对页面所作的单击等操作，和简单的表单填写

3.5 输出项

根据具体业务逻辑，输出为带有查询数据的页面

3.6 流程逻辑

以下为页面跳转图



3.7 接口

前端与后端通过 HTTP 协议进行接口通讯，交换业务所需的数据。接口定义和实现分为两个阶段，第一阶段仅实现该系统最基础的功能，其余功能留作第二阶段实现
两阶段接口具体定义见 附录 1：第一阶段接口文档 和 附录 2：第二阶段接口文档

3.8 注释设计

1. 在每份代码的头部使用注释，说明代码作者，最后修改时间，主要功能
2. 代码中的注释全部独立成行，避免代码行末尾注释
3. 对每个函数的功能，参数，返回值作说明
4. 在其他需要的地方适当增加注释

3.9 限制条件

1. 网络：客户端本身连同所有数据通过因特网传输，客户计算机必须能够正常访问因特网
2. 浏览器：改客户端为 Web 应用，必须在 Web 浏览器上运行。支持 Google Chrome 或 Mozilla Firefox 等浏览器
3. 后端：所有客户数据都需通过接口从后端获得，所以在运行客户端之前需保证后端正在运行并且可访问

3.10 测试计划

1. 用户测试，通过点击页面，观察页面响应进行测试
2. 测试 Mozilla Firefox 浏览器和 Google Chrome 浏览器上的兼容情况，其他不对 IE 系列浏览器做兼容性测试
3. 通过浏览器的开发者工具检查，前端是否正确响应用户行为，发送正确的 AJAX 请求

3.11 尚未解决的问题

1. 数据加密传输，目前应用运行在 HTTP 服务器上，需迁移至 HTTPS
2. 人机验证，需添加人机验证，确保访问该应用的是真实的人而非网络机器人
3. 更直观的数据呈现方式，充分利用图表、柱状图、扇形图、热点图等方式直观呈现数据

4“Web 应用后端”设计说明

4.1 程序描述

该部分程序用于接收前端发送的请求，查询数据库获得所需数据，根据接口定义返回一定格式的数据作为响应

其是运行在 Linux 主机上的 Python3.6 程序，使用 Flask 框架的 Web 应用程序，通过 Nginx 代理网络请求，以 uWSGI 作为 WSGI 容器，通过 SQLAlchemy ORM 框架操作 MySQL 数据库

4.2 功能

该部分程序为桌面 Web 客户端提供数据接口，具体功能细节见“接口”（节 4.6）

4.3 性能

时间开销

1. 单条数据库查询指令耗时：少于 1s；
2. 单个接口请求响应时间： 少于 3s；

灵活性

1. 接口设计避免依赖具体实现，当切换后端实现时不会影响接口
2. 允许在不停止服务的情况下修改代码并自动更新其服务
3. 使用 SQLAlchemy ORM 操作数据库，当切换数据库时，不影响后端实现

4.4 输人项

该部分程序输入项为符合接口文档的 HTTP 请求，详细格式和内容见 HTTP/1.1 文档(RFC 2616)，其余详见“接口”（节 4.6）

4.5 输出项

该部分程序输出项为符合接口文档的 HTTP 响应，响应数据是以 JSON 字符串格式表示的。详见“接口”（节 4.6）

4.6 接口

该部分程序对上层应用，即桌面 Web 客户端提供基于 HTTP 协议的数据接口，具体接口定义见附录 1：第一阶段接口文档 和 附录 2：第二阶段接口文档

该部分程序对下层应用，即 MySQL 数据库使用基于 TCP 的协议进行接口通讯，通过 ORM 框架 SQLAlchemy 封装其通讯，具体接口定义见 SQLAlchemy 官方文档

4.7 存储分配

该部分程序需要使用 MySQL 数据库存储应用所必需的用户资料。总共包括 14 张数据表，数据库模式设计细节见附录 3：数据库模式设计文档

4.8 注释设计

1. 在每份代码的头部使用注释，说明代码作者，最后修改时间，主要功能
2. 代码中的注释全部独立成行，避免代码行末尾注释
3. 对每个函数的功能，参数，返回值作说明
4. 在其他需要的地方适当增加注释

4.9 限制条件

1. 网络，运行该程序的主机必须能够正常连接互联网，并且被通过公网 ip 寻址并访问
2. 数据库，该程序的数据存储和查询必须依赖 MySQL 数据库，需保证所连接的 MySQL 服务器可用
3. 该程序仅是数据接口，接口的访问需要通过用户代理，如桌面 Web 客户端
4. 数据查询的时效性，事物的完整性，很多情况下是要依赖所使用的数据库的

4.10 测试计划

使用 Postman 等接口测试工具，根据接口文档测试后端数据接口。

4.11 尚未解决的问题

1. 更高的数据库查询效率，目前仅简单地使用 SQLAlchemy 的 ORM 操作数据库，没有仔细优化查询语句
2. 负载均衡，如果需要应对选课等高并发问题，就需要使用负载均衡等方法保证响应时效性
3. 容灾备份，目前还没有应对数据库或服务器崩溃造成数据丢失的机制

附录 1：第一阶段接口文档

Host: http://xxx.xxx.xxx.xxx（不便于在此公开）

账户

登录

Request:

URL: `/login`

POST

```
post_data = {
    'account': 'balabala',
    'password': 'ba1aba1aba1aba1aba1aba1aba1aba1a'  // 32 位 MD5
}
```

Response:

- 成功： '{"type": "someone"}', 200 （ 'someone' ，根据登陆者身份返回身份说明'student' 、 'instructor' 、'superior' 、 'admin' 或 'root' ）
- 类型错误： '{"message": "type error"}', 400
- 数据缺失： '{"message":"data missing"}', 400
- 密码错误： '{"message": "password error"}', 403
- 用户不存在： '{"message": "no account"}', 404

总是通过 session 记录登录状态

我是谁

Request:

URL: `/who_am_i`

GET

Response:

- 成功： `'{"type": "", "name": ""}', 200`
- 未登录： `"no login", 401`

修改密码

Request:

URL: `/change_passwd`

POST

```
post_data = {
    'old_password': 'ba1aba1aba1aba1aba1aba1aba1aba11',  // 32 位 MD5
    'new_password': 'ba1aba1aba1aba1aba1aba1aba1aba1a'  // 32 位 MD5
}
```

Response:

- 成功： `'{"message": "change successful"}', 200`
- 旧密码错误： `'{"message": "password error"}', 403`
- 未登录： `"no login", 401`
- 账号不存在： `'{"message": "no account"}', 404`

登出

Request:

URL: `/logout`

GET

Response:

- 成功： '{"message": "logout successful"}', 200
- 未登录： "no login", 401

学生端功能

获取课表

Request:

URL: `/student/mine_class`

GET

Response:

- 未登录 401
- 成功：JSON, 200

```
JSON = [
    {
        'course_id': '',
        'course_name': '',
        'instructor_name': ['', ''],  // 如果有多位老师
        'classroom_id': '',
        'week': '',
        'day': '',
        'section': ''
    },
    // {...},
]
```

- 缺少信息 {'message':'data missing'} 403
- 没有课程 {'message':'no course'} 404

获取成绩单

Request:

URL: `/student/mine_grade`

GET

Response:

- 未登录 401
- 成功：JSON, 200

```
JSON = [
    {
        'course_id': '',
        'name': '',  // 课程名
        'type': '',  // 课程属性
        'credit': '',
        'grade': ''
    },
    // {...},
]
```

- {'message':'no course'} 404

查看可选课信息

Request:

URL: `/student/classes_list`

GET

Response:

- 未登录 401
- 成功：JSON, 200

```
JSON = [
    {
        "course_info": {
            "course_id": "",
            "course_name": "",
            "type": "",  // 课程属性
            "credit": "",
            "period": "",
        },
        "class_id" : "",
        "last_people": "",  // 剩余人数
        "selected": "",  // 是否已选
        "instructor_name": ["", ""],
        "classroom_id": "",
        "time": [
            {"section": "", "day": ""},
            // {...},
        ],
    },
    // {...},
]
```

- {'message': 'no course'}, 404

选课

Request:

URL: `/student/choice_class`

GET

```
get_data = {'class_id': ''}
```

Response:

- 未登录：401
- 已选： {'message':'had cancel'} 200
- 成功：200
- 不能选课 {'message':'can't choose'} 403
- 其他 {'message':'no class'} 404

查看考试信息

Request:

URL: `/student/exam_info`

GET

Response:

- 未登录：
- 成功：`JSON, 200`

```
JSON = [
    {
        'course_name': '',
        'classroom_id': '',
        'date': '',
        'time': '',
        'exam_grade': ''
    },
    // {...},
]
```

- 没有考试信息 {'message':'no exam'}, 404

教师端功能

获取课程表

Request:

URL: `/teacher/mine_class`

GET

Response:

- 未登录：401
- 成功：JSON, 200

```
JSON = [
    {
        'class_id': '',
        'course_id': '',
        'course_name': '',
        'classroom_id': '',
        'week': '',
        'day': '',
        'section': ''
    },
    // {...},
]
```

- {'message':'no class'} 404

获取所授课程信息

Request:

URL: `/teacher/class_info`

GET

Response:

- 未登录：401
- 成功：JSON, 200

```
JSON = [
    {
        'class_id': '',
        'course_id': '',
        'course_name': '',
        'type': '',  // 课程属性
        'classroom_id': '',
        'time': [
            {'day': '', 'section': ''},
            // {...}
```

```
        ],  // 如果一周要上多节
    },
    // {...},
]
```

- {'message':'no class'} 404

获取某教学班学生名单

Request:

URL: /teacher/class_people

GET

```
get_data = {'class_id': ''}
```

Response:

- 未登录：
- 成功：JSON, 200

```
JSON = [
    {
        'student_id': '',
        'student_name': '',
        'grade': ''
    },
    // {...},
]
```

- 该课程任课老师不是你 403
- 其他 {'message':'no data'},401

录成绩

Request:

URL: /teacher/insert_grade

POST

// 写进 request.data ,用 JSON 格式

```
post_data = {
    'class_id': '',
    'grade': [
        {
            'student_id': '',
            'grade': ''
        },
        // {...},
    ]
}
```

Response:

- 成功：200
- 未登录：401
- 该课程任课老师不是你 403
- 课程不存在 404
- 错误数据 '{"student_id": []}', 200

获取考试安排

***仅能看教师自己所上班级的考试安排**

Request:

URL: /teacher/exam_info

GET

Response:

- 未登录：
- 成功：JSON, 200

```
JSON = [
```

```
    {
        'course_name': '',
        'classroom_id': [],
        'date': '',
        'time': '',
        // 'exam_grade': '',暂未加入，给什么考试数据？平均数？
    },
    // {...},
]
```

- 其他

系统管理员

获取表名列表

Request:

URL: `/root/root_show_tables`

GET

Response:

- 未登录：
- 成功：`JSON, 200`

```
JSON = [
    'table_name1',
    'table_name2',
    // ...
]
```

- 其他

获取某表

Request：

URL:/root/root_get_table

GET

```
get_data = {'table_name': ,}
// 放在 request.form['table_name'] 里就可以
```

Response:

- 未登录：
- 成功：JSON，200

```
JSON = {
    'format': ['col_name1', 'col_name2', '...'],  // 每一列的列名，主键放在第一列
    'data': [
        ('row1_col1', 'row1_col2', 'row1_col3', 'row1_...'),
        ('row2_col1', 'row2_col2', 'row2_col3', 'row2_...'),
        // [...],
}
```

- 表名错误 403

修改某表

Request:

URL: /root_change_table

POST

```
post_data = {
    'table_name': '',
    'type': '',  // 'DELETE', 'INSERT'或 'UPDATE'
    'primary_key': '',  // 一个主键的值，用于'DELETE'、'UPDATE'操作的 WHERE 字句做比较，'INSERT'的话这个值为空
    'data': ['col1', 'col2', '...']  // 用于'UPDATE'或'INSERT'插入或修改的数据，'DELETE'的话该值为空
}
```

附录 2：第二阶段接口文档

Host: http://xxx.xxx.xxx.xxx:xxxxx（不便于在此公开）

普通管理员

列出所有用户

Request

URL: `/admin/account_list`
GET

Response

- 成功：JSON, 200

```
JSON = [
    {
        'account': '',
        'name': '',
        'type',
    },
    // {...},
]
```

- 未登录：401

修改用户 x

Request

URL: `/admin/change_account`
POST

```
post_data = {
    'change_type': '',  // 修改类型，可以是'DELETE', 'UPDATE', 'INSERT'
    'detail': {},  // 具体的修改数据，根据修改类型的不同，此处结构会有不同，见下
}

if (post_data['change_type'] == 'DELETE')
```

```
    post_data['detail'] = {
        'account': '',
        'type': '' // 同 update.
    };
else if (post_data['change_type'] == 'UPDATE')
    post_data['detail'] = {
        'account': '',
        'name': '',
        'type': '' // 全部字母小写 比如 student/instructor/admin/superior 便于查表
        'passwd': '',  // 除了'account'项必须提供，其他几项只提供需要修改的项，如果某项未提供则表示该项维持原值
    };
else if (post_data['change_type'] == 'INSERT')
    post_data['detail'] = {
        'account': '',
        'name': '',
        'type': '',
        'passwd': '',
        'year':  //整数，type == student 时需要，没有的话默认为当前年份
    };
```

Response

- 成功： 200
- 未登录： 401
- 操作不允许： 403

Tip: 不能修改 root 账户和 admin 账户

查看考试安排

Request

URL: /admin/exam_info
GET

Response

- 成功： JSON, 200

```
JSON = [
    {
        'exam_id': '',
        'date': '',
```

```
        'time': '',
        'classroom_id': '',
        'class_id': '',
    },
    // {...},
]
```

- 未登录： 401

查看某场考试具体信息

Request

URL: /admin/exam_info
GET

```
get_data = {'exam_id': ''} // 放在 request.form 里，我通过
request.form.get('exam_id') is None 来判断是哪种请求
```

Response

- 成功： JSON, 200

```
JSON = {
    'exam_id': '',
    'date': '',
    'time': '',
    'classroom_id': '',
    'class_id': '',
    'course_ id': '',
    'course_name': '',
    'instructor_name': ['', ''],
    'student': [
        {'student_id': '', 'name': ''},
    ],
}
```

- 未登录： 401

修改考试信息（暂时搁置）

Request

URL: /admin/change_exam
POST

```
post_data = {

}
```

Response

- 成功： 200
- 未登录： 401

获取全部教学班信息

请求和 exam_info 相同

Request

URL: /admin/class_list
POST

Response

- 成功： JSON, 200

```
JSON = [
    {
        'class_id': '',
        'course_id': '',
        'course_name': '',
        'classroom_id': '',
        'instructor_name': ['', ''],
    },
    // {...},
]
```

- 未登录： 401
- 失败 403
- 成功 200

获取某教学班详细信息

Request

URL: /admin/class_list
GET

```
get_data = {'class_id': ''}
```

Response

- 成功： JSON, 200

```
JSON = {
    'class_id': '',
    'classroom_id': '',
    'instructor': ['', ''],
    'course_info': {
        'course_id': '',
        'course_name': '',
        'type': '',
        'credit': '',
        'period': '',
    },
    'student': [
        {'student_id': '', 'name': ''},
        // {...},
    ]
}
```

- 未登录： 401
- 失败： 403
- 成功: 200

修改教学班信息（暂时搁置）

获取所有课程信息

Request

URL: /admin/course_list
GET

Response

- 成功： JSON, 200

```
JSON = [
    {
        'course_id': '',
        'course_name': '',
        'type': '',
```

```
        'credit': '',
        'period': ''
    },
    // {...},
]
```

- 未登录： 401

修改课程信息

Request

URL: /admin/change_course
POST

```
post_data = {
    'change_type': '',  // 修改类型，可以是'DELETE', 'UPDATE', 'INSERT'
    'detail': {}  // 具体的修改数据，根据修改类型的不同，此处结构会有不同，见下
}

if (post_data['change_type'] == 'DELETE')
    post_data['detail'] = {
        'course_id': ''
    };
else if (post_data['change_type'] == 'UPDATE')
    post_data['detail'] = {
        'course_id': '',
        'course_name': '',
        'credit': '',
        'type': '',
        'period': ''  // 除了'course_id'项必须提供，其他几项只提供需要修改的项，
如果某项未提供则表示该项维持原值
    };
else if (post_data['change_type'] == 'INSERT')
    post_data['detail'] = {
        'course_id': '',
        'course_name': '',
        'type': '',
        'credit': '',
        'period': ''
    };
```

Response

- 成功： 200

- 未登录： 401
- 失败： 403

获取所有课表

Request

URL: /admin/schedule_list
GET

Response

- 成功： JSON, 200

```
JSON = [
    {
        'schedule_id': // 数据库中的主键,int
        'class_id': '',
        'course_name': '',
        'classroom_id': '',
        'week': '',
        'day': '',
        'section': '',
    },
    // {...},
]
```

修改课表

Request

URL: /admin/change_schedule
POST

```
post_data = {
    'change_type': '',  // 修改类型，可以是'DELETE', 'UPDATE', 'INSERT'
    'detail': {}  // 具体的修改数据，根据修改类型的不同，此处结构会有不同，见下
}

if (post_data['change_type'] == 'DELETE')
    post_data['detail'] = {
       'schedule_id': // int 类型
    };
else if (post_data['change_type'] == 'UPDATE')
```

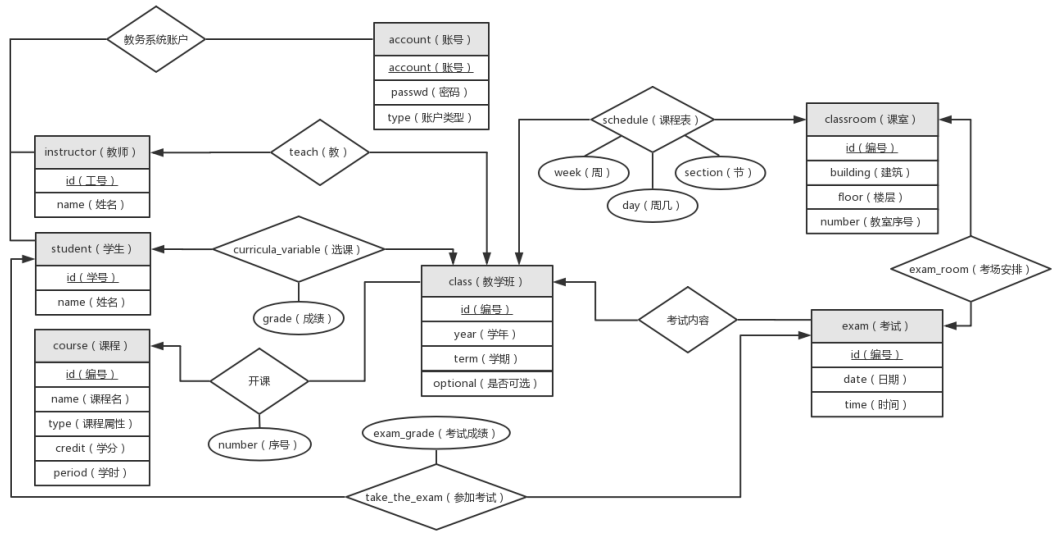
```
    post_data['detail'] = {
        'schedule_id': // int 类型
        'class_id': '',
        'classroom_id': '',
        'week': '',
        'day': '',
        'section': ''
    };
else if (post_data['change_type'] == 'INSERT')
    post_data['detail'] = {
        // 不用发 insert_id, orm 自动生成
        'class_id': '',
        'classroom_id': '',
        'week': '',
        'day': '',
        'section': ''
    };
```

Response

- 成功： 200
- 未登录： 401
- 失败: 403

附录 3：数据库模式设计文档

E-R 图



SQL

```sql
CREATE TABLE account (
  account VARCHAR(32) PRIMARY KEY NOT NULL,  # 账号，全部写成整形数
  passwd VARCHAR(32) NOT NULL,               # 密码
  type VARCHAR(16) NOT NULL                  # 账户类型
);


CREATE TABLE student (
  id VARCHAR(11) PRIMARY KEY NOT NULL,  # 学号
  name VARCHAR(32) NOT NULL             # 姓名
  year INT NOT NULL                     # 入学年份
);

CREATE TABLE instructor (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 编号
  name VARCHAR(32) NOT NULL             # 姓名
);
```

```
CREATE TABLE SUPERIOR(
   id VARCHAR(16) PRIMARY KEY NOT NULL,  # 编号
  name VARCHAR(32) NOT NULL
 );

CREATE TABLE ADMIN (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 编号
  name VARCHAR(32) NOT NULL            # 姓名
);

CREATE TABLE course (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 课程编号
  name VARCHAR(64) NOT NULL,           # 课程名
  type VARCHAR(16) NOT NULL,           # 课程属性
  credit NUMERIC(3, 1) NOT NULL,        # 学分
  period NUMERIC(5, 1) NOT NULL         # 学时
);

CREATE TABLE classroom (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 课室编号
  building VARCHAR(32) NOT NULL,        # 建筑名
  floor INT NOT NULL,                  # 楼层
  number INT NOT NULL                  # 序号
);


CREATE TABLE class (
  id VARCHAR(32) PRIMARY KEY NOT NULL,  # 教学班编号
  year INT NOT NULL,                   # 开课学年
  term INT NOT NULL,                   # 开课学期
  course_id VARCHAR(16) NOT NULL,       # 所开课程
  number INT NOT NULL,                 # 序号
  optional BOOL,                       # 是否可以选课
  FOREIGN KEY (course_id) REFERENCES course(id)
);

CREATE TABLE exam (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 考试编号
  class_id VARCHAR(32) NOT NULL,        # 课程 id
  date DATE NOT NULL,                  # 日期 格式为年/月/日
  time TIME NOT NULL,                  # 时间 格式为小时:分钟， 按 24 小时制
  FOREIGN KEY (class_id) REFERENCES class(id)
);
```

```
CREATE TABLE curricula_variable (
  student_id VARCHAR(11) NOT NULL, # 学生学号
  class_id VARCHAR(32) NOT NULL,   # 教学班 id
  grade NUMERIC(3, 1) NOT NULL,    # 课程成绩
  PRIMARY KEY (student_id, class_id),
  FOREIGN KEY (student_id) REFERENCES student(id),
  FOREIGN KEY (class_id) REFERENCES class(id)
);

CREATE TABLE teach (
  instructor_id VARCHAR(16) NOT NULL,  # 教师编号
  class_id VARCHAR(32) NOT NULL,
  PRIMARY KEY (instructor_id, class_id),
  FOREIGN KEY (instructor_id) REFERENCES instructor(id),
  FOREIGN KEY (class_id) REFERENCES class(id)
);

CREATE TABLE schedule (
  id INTEGER NOT NULL,
  class_id VARCHAR(32) NOT NULL,
  classroom_id VARCHAR(16) NOT NULL,  # 课室 id
  week INT NOT NULL,                  # 周
  day INT NOT NULL,                   # 天
  section INT NOT NULL,               # 节
  PRIMARY KEY (id),
  FOREIGN KEY (class_id) REFERENCES class(id),
  FOREIGN KEY (classroom_id) REFERENCES classroom(id)
);

CREATE TABLE exam_room (
  exam_id VARCHAR(16) NOT NULL,
  classroom_id VARCHAR(16) NOT NULL,
  PRIMARY KEY (exam_id, classroom_id),
  FOREIGN KEY (exam_id) REFERENCES exam(id),
  FOREIGN KEY (classroom_id) REFERENCES classroom(id)
);

CREATE TABLE take_the_exam (
  student_id VARCHAR(11) NOT NULL,    # 学生学号
  exam_id VARCHAR(16) NOT NULL,       # 考试 id
  exam_grade NUMERIC(3, 1) NOT NULL,  # 考试成绩
  PRIMARY KEY (student_id, exam_id),
  FOREIGN KEY (student_id) REFERENCES student(id),
```

```
  FOREIGN KEY (exam_id) REFERENCES exam(id),
);
```

简要说明

```
CREATE TABLE account (
  account VARCHAR(32) PRIMARY KEY NOT NULL,  # 账号
  passwd VARCHAR(32) NOT NULL,               # 密码
  type VARCHAR(16) NOT NULL                  # 账户类型
);
```

- account 如果是学生、教师账户则为其 id，如果是领导、管理员则随意定（以 u 开头，后面随意定（'u'保证不会和学生、教师重复））
- passwd 密码的 32 位 MD5 摘要
- type 学生为'student'，教师为'instructor'，领导为'superior'，普通管理员为'admin'，系统管理员为'root'

```
CREATE TABLE student (
  id VARCHAR(11) PRIMARY KEY NOT NULL,  # 学号
  name VARCHAR(32) NOT NULL             # 姓名
);
```

- id 同西电学号格式，11 位
- name 学生姓名

```
CREATE TABLE instructor (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 编号
  name VARCHAR(32) NOT NULL             # 姓名
);
```

- id 格式为名缩写加姓拼音缀数字，如'yhluo'（罗阳豪），如有重复则后缀数字"yhluo_1"，
- name 教师名字

```
CREATE TABLE course (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 课程编号
  name VARCHAR(64) NOT NULL,            # 课程名
  type VARCHAR(16) NOT NULL,            # 课程属性
  credit NUMERIC(3, 1) NOT NULL,        # 学分
  period NUMERIC(5, 1) NOT NULL         # 学时
);
```

- id 参考西电课程号
- name 课程名

- type 必修为 1，校任选为 2，人文类选修为 3，学院选修为 4
- credit 、 period 学分、学时，保留小数点后一位

```
CREATE TABLE classroom (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 课室编号
  building VARCHAR(32) NOT NULL,        # 建筑名
  floor INT NOT NULL,                   # 楼层
  number INT NOT NULL                   # 序号
);
```

- id 通过 building 、 floor 、 number 生成，如 (“B-206”, “B”, 2, 6), ("EI-208", "EI", 2, 8)

```
CREATE TABLE class (
  id VARCHAR(32) PRIMARY KEY NOT NULL,  # 教学班编号
  year INT NOT NULL,                    # 开课学年
  term INT NOT NULL,                    # 开课学期
  course_id VARCHAR(16) NOT NULL,       # 所开课程
  number INT NOT NULL,                  # 序号
  optional BOOL,                        # 是否可以选课
  FOREIGN KEY (course_id) REFERENCES course(id)
);
```

“教学班”并不是一般所说的班级，是指一个学期、一门课的一个班，比如 2018 年上半年高数 II 的某个班

- id 格式“课程编号+_序号”（如，"AM1001L_1"）
- term 上半年为 2，下半年为 1（或者理解为第一学期为 1，第二学期为 2）
- number 该学期该门课开了多个班的，序号由 1 开始递增

```
CREATE TABLE exam (
  id VARCHAR(16) PRIMARY KEY NOT NULL,  # 考试编号
  class_id VARCHAR(32) NOT NULL,        # 课程 id
  date DATE NOT NULL,                   # 日期
  time TIME NOT NULL,                   # 时间
  FOREIGN KEY (class_id) REFERENCES class(id)
);
```

- id 格式“考试的教学班 id+_表示考试性质的字符串（比如期中考‘m’，期末考‘f’）”
- date ， 日期格式“YYYY-MM-DD”
- time， 时间格式“hh:mm:ss”

```
CREATE TABLE curricula_variable (
  student_id VARCHAR(11) NOT NULL, # 学生学号
```

```
  class_id VARCHAR(32) NOT NULL,   # 教学班 id
  grade NUMERIC(3, 1) NOT NULL,    # 课程成绩
  PRIMARY KEY (student_id, class_id),
  FOREIGN KEY (student_id) REFERENCES student(id),
  FOREIGN KEY (class_id) REFERENCES class(id)
);
```

表示学生应上的课

- 课程成绩如果还没录入则为-1

```
CREATE TABLE teach (
  instructor_id VARCHAR(16) NOT NULL,  # 教师编号
  class_id VARCHAR(32) NOT NULL,
  PRIMARY KEY (instructor_id, class_id),
  FOREIGN KEY (instructor_id) REFERENCES instructor(id),
  FOREIGN KEY (class_id) REFERENCES class(id)
);
```

表示某课的任课老师，一节课有多位老师可列多条

```
CREATE TABLE schedule (
  class_id VARCHAR(32) NOT NULL,
  classroom_id VARCHAR(16) NOT NULL,  # 课室 id
  week INT NOT NULL,                  # 周
  day INT NOT NULL,                   # 天
  section INT NOT NULL,               # 节
  PRIMARY KEY (class_id, classroom_id),
  FOREIGN KEY (class_id) REFERENCES class(id),
  FOREIGN KEY (classroom_id) REFERENCES classroom(id)
);
```

- week，第几周，从 1 开始递增
- day，周一到周六是 1 到 6，周日是 0
- section，一天 5 节，上午两节，下午两节，晚上一节，分别是 1 到 5

```
CREATE TABLE exam_room (
  exam_id VARCHAR(16) NOT NULL,
  classroom_id VARCHAR(16) NOT NULL,
  PRIMARY KEY (exam_id, classroom_id),
  FOREIGN KEY (exam_id) REFERENCES exam(id),
  FOREIGN KEY (classroom_id) REFERENCES classroom(id)
);
```

表示考场安排

```
CREATE TABLE take_the_exam (
  student_id VARCHAR(11) NOT NULL,    # 学生学号
```

```
  exam_id VARCHAR(16) NOT NULL,       # 考试 id
  exam_grade NUMERIC(3, 1) NOT NULL,  # 考试成绩
  PRIMARY KEY (student_id, exam_id),
  FOREIGN KEY (student_id) REFERENCES student(id),
  FOREIGN KEY (exam_id) REFERENCES exam(id),
);
```

往 exam 插入一条记录同时根据 class_id 查找应该参加该考试的学生，并将相应结果插入该表

- exam_grade 考试成绩，不同于上述的课程成绩，成绩未录入时为-1