RESTFul API Design

School

1. GET api/school

a. Description

Get schools list. b. Return Format: **json**

```
{
  "school_ids":[1, 2, 3, 4]
}
```

2. GET api/school/{school_id}

a. Description

Get a specific school with id `school_id` b. Return Format: **json**

```
{
        "school_id": 1,
        "school_name": "SUSTech",
        "website": "sustc.edu.cn",
        "description": "Best university of the world."
}
```


3. POST /api/school + json

a. Description

Add a school to database.

4. PUT /api/school/{school_id} + json

a. Description **Update** a school in database. b. Request Body: **json**


```
{
        "school_name": "SUSTech",
        "website": "sustc.edu.cn",
        "description": "Best university of the world."
}
```


5. DELETE /api/school/{schoold_id}

a. Description

Delete a school from database.

6. GET api/school/{school_id}/courses

a. Description

Get all courses id of the school with id `school_id`. b. Return Format: **json**


```
{
  "course_ids": [1, 2, 3, 4]
}
```


~~7.~~ GET api/school/{schoold_id}/tree

a. Description

Get a json that represents the relationship of all courses in that school.

b. Return Format: **json** Generated automatically by Javascript.


```
{
        "version": "2.0",
        "refs": {
                "1": {
                        "_classPath": "Q.Position.CENTER_MIDDLE"
                }
        },
        "datas": [
```

```
                {
                        "_className": "Q.Text",
                        "json": {
                                "name": "计算机导论",
                                "styles": {
                                        "label.position": {
                                                "$ref": 1
                                        },
                                        "label.anchor.position": {
                                                "$ref": 1
                                        },
                                        "label.background.color": "#dadada",
                                        "label.size": {
                                                "_className": "Q.Size",
                                                "json": {
                                                        "width": 150,
                                                        "height": 70
                                                }
                                        },
                                        "label.radius": 0,
                                        "label.font.size": 18,
                                        "label.color": "#555",
                                        "border.width": 1,
                                        "border.color": "#555",
                                        "border.radius": 0
                                },
                                "location": {
                                        "x": -776.6396587982824,
                                        "y": -285.00512875536464
                                }
                        },
                        "_refId": "7048"
                }
  ]
}
```


~~8.~~ POST /api/school/{school_id}/tree + json

a. Description

Add courses relation tree for a school. POST body is a json same as above.

~~9.~~ PUT /api/school/{school_id}/tree + json

a. Description

Modify courses relation tree for a school. PUT body is a json same as above.

~~10~~. DELETE /api/school/{school_id}/tree

a. Description

Delete courses relation tree for a school.

Course

1. GET api/course

a. Description

Get courses list. b. Return Format: **json**

```
{
  "course_ids":[1, 2, 3, 4]
}
```

Note: this command return all courses id stored in the databases, not only the courses in one school!

2. GET api/course/{course_id}

a. Description

Get a specific course with id `course_id`. b. Return Format: **json**

```
{
  "course_id": 1,
  "course_code": "CS303",
  "course_name": "Database Principle",
  "description": "this is a course",
  "professor": "John",
  "author": 12,
  "school": 1
}
```

Note: The Response does not contain the syllabus(Which is PDF format). So you should send another request to server in order to get it.

3. POST api/course + json

a. Description

Add a specific course.

b. Procedures:

1. First we need to post a json which contains the basic infomation of the course except the syllabus file to the server.

Request Body: **json**

```
{
  "course_code": "CS303",
  "course_name": "Database Principle",
  "description": "this is a course",
  "professor": "John",
  "author": 12,
  "school": 1
}
```

2. If the json is accepted by the server, the server will return the id of the the course. Then we post syllabus with this id inorder to associate it with the course we created previously.

Request Body: **multipart form data**

[file]	syllabus.pdf

4. PUT api/course/{course_id} + json

a. Description **Update** a specific course (when you need to update the syllabus, use **8**).

5. DELETE api/course/{course_id}

a. Description

Delete a specific course with id `course_id`.

~~6.~~ GET api/course/{course_id}/syllabus

a. Description

Get a syllabus of a course with id `course_id` b. Return Format: **PDF**

~~7.~~ POST api/course/{course_id}/syllabus + PDF

a. Description

Add a syllabus of a course with id `course_id`

~~8.~~ PUT api/course/{course_id}/syllabus + PDF

a. Description

Update a syllabus of a course with id `course_id`

~~9.~~ DELETE api/course/{course_id}/syllabus

a. Description

Delete a syllabus of a course with id `course_id`

User

1. GET api/user

a. Description

Return all users in the databases b. Return Format: **json**

```
{
  "user_ids": [1, 2, 3, 4]
}
```

2. GET api/user/{user_id}

a. Description

Get a specific user with id `user_id` b. Return Format: **json**

```
{
  "name": "John",
  "email": "John@gmail.com",
  "password": "123456",
  "school": "SUSTech"
}
```

Note: When logged in, the user_id of current user will be stored in cookie.

3. POST api/user + json

a. Description

Add a user to database.

4. PUT api/user + json

a. Description

Update a user to database.

5. DELETE api/user

a. Description

Delete a user to database.