《信息系统集成课程实习》

实习一报告

学　　院:　　遥感信息工程学院

班　　级:　　20F10

学　　号:　　2020302131249

姓　　名:　　马文卓

实习地点:　　教学实验大楼 101 机房

指导教师:　　王华敏

2022 年　　12 月　　6 日

目录

一. 实验概述

1. 实验代号

本次实验的代号为 dace。

2. 实验目的

本实验考察学生对 Web Service 理论掌握的情况，通过一个具体的 Web 应用的前后端编写，实现相应的 REST风格 Web API，服务于前端网页、以及各种 API 调用。通过该实验过程，学生可以了解 REST 架构风格的特点、集合类资源的 WebAPI 设计方式、以 JSON 为格式的表述方法，从而加深对 Web Service 理论的理解。

二. 项目介绍

1. 项目概述

本项目名为【MyData】，是一款学生信息收集工具。MyData 旨在方便快速地收集学生的姓名、学号、电话、邮箱、专业、兴趣、出生日期共 7 项信息，并存储在数据库当中以供使用。为了用户方便，MyData 提供了网页版和命令行版本。网页版有着简洁美观的界面 UI，而命令行版本则具有强大的功能且方便。

针对于具体实现，网页前端使用了 html+css+javascript 三件套，其中使用 JQuery 的 jajx 进行 http 请求。命令行 App 则使用 Python 进行编写，使用 requests 库进行 http 请求。服务器端也使用 Python 编写，并且使用 flask 框架响应客户端的 http 请求。数据存储在 MySQL 数据库 students 的 student 表中。

2. 项目结构

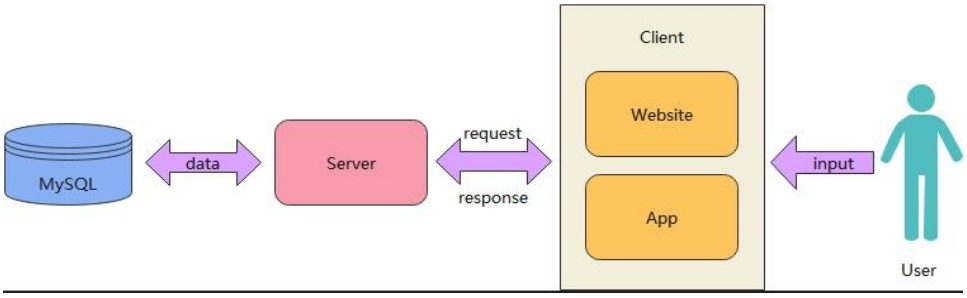


Figure 1:项目结构图

3. 功能介绍

MyData 作为学生信息收集工具，数据的基本操作均有体现，本节只是大体罗列主要功能，功能的详细介绍请见各自对应章节。

（1）网页

1>信息收集：收集学生信息提交给服务端
2>信息展示：展示已有信息（或查询的结果）
3>重置表格：清空表格中的已填信息
4>信息验证：验证所填信息是否合乎规范
5>信息刷新：刷新展示界面
6>信息查询：查询想要的数据展示在展示界面
7>信息更改：对指定学号的信息进行数据项更改
8>信息删除：删除指定学号的信息
9>操作提示：对本网站的相关信息和操作进行提示

（2）App

1>数据展示：罗列出数据库中已有信息
2>增添数据：增加一条数据
3>数据验证：验证增添的数据中数据项是否合乎规范
4>删除数据：删除一条指定数据
5>更新数据：对指定数据进行更新
6>查询数据：查询想要的数据
7>帮助：展示 App 的帮助提示
8>版本：展示与 MyData 相关的版本信息

4. 文件组织

本节主要介绍本项目的文件组织结构，方便查看。具体如下：

- dace：包含项目所有内容
  - client：存储客户端相关文件
    - client.html：网页客户端
    - App.py：命令行客户端
    - template.json：资源模板
    - layui：网页客户端所引用的“饿了吗”UI 样式
  - server：存储服务端相关文件
    - server.py：服务器
    - data.sql：初始数据
  - pictures：存储报告中所用到的图表
  - video：存储功能展示视频
    - video1：网页前端功能展示
    - video2：命令行 App 功能展示
  - dace.doc：项目报告
  - ReadMe.md：项目的 readme

5. 使用方法

本节介绍本项目的正确启动方法，具体如下。

1>安装 MySQL（如果已有可跳过此步骤）

2>启动 MySQL 服务（这个步骤不可或缺，每次使用 MyData 前都要确保 MySQL 的服务已经开启，否则 MyData 服务器无法连接服务器）

```
start mysql net
```

3>创建一个名为 students 的 MySQL 数据库，或者在 server.py 中更改数据库名称为你数据库的名称

```
mysql -h localhost -u root -p
create database students;
```

4>在 students 数据库中创建一个名为 student 的表

```
create table student(name varchar(20) not null,
id varchar(20) not null,
phonenumber varchar(20) not null,
email varchar(40) not null,
major varchar(40) not null,
interest varchar(200),
birthday varchar(20) not null,
primary key (id));
```

```
mysql> desc student;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| name        | varchar(20)  | NO   |     | NULL    |       |
| id          | varchar(20)  | NO   | PRI | NULL    |       |
| phonenumber | varchar(20)  | NO   |     | NULL    |       |
| email       | varchar(40)  | NO   |     | NULL    |       |
| major       | varchar(40)  | NO   |     | NULL    |       |
| interest    | varchar(200) | YES  |     | NULL    |       |
| birthday    | varchar(20)  | NO   |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Figure 2:表结构

5>启动服务器：运行 server.py

```
python server.py
```

6>启动客户端：在你的浏览器中打开 client.html 或者运行 App.py

```
python App.py
```

*可以先把初始数据 data.sql 导入数据库的 student 表中，再使用哦

三．网页前端

1. 界面 UI

Figure 3:网页界面

本次项目网页前端的界面设计秉承简单大方、简洁美观的原则，为了更好的达到这种效果，我们借用了“饿了吗”的 UI 库进行构造。

可以看到界面整体上分成三块区域：信息录入区域、信息展示区域和功能栏。

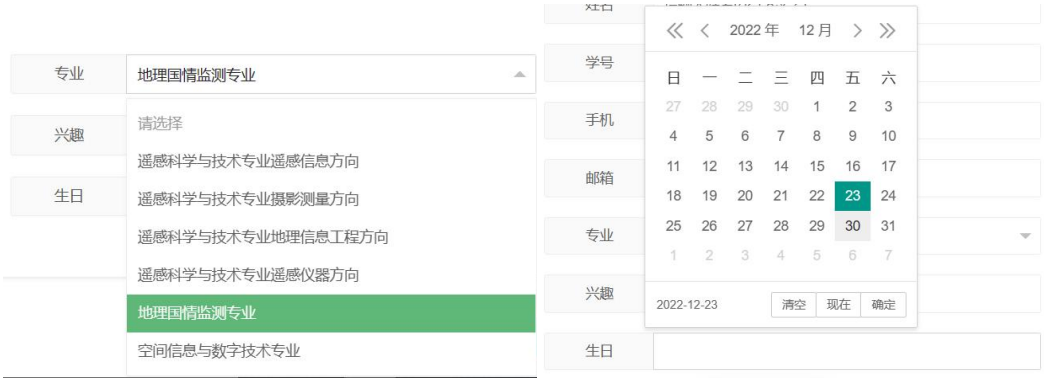


Figure 4:选项设计

信息录入区主要是一个面板当中放入一个表单加上两个按钮所组成。关于表单的颜色选择，我们选择了较为浅的黑灰色，旨在给用户一种比较舒服的效果。提交按钮由于是主要的功能按钮，所以选择页面的主题颜色，经过多次改进之后，该按钮呈现出一种渐变青绿色，颇为好看。重置按钮由于是辅助按钮，所以设置为白色，且大小较小，显得更加契合。对于表单中特殊选项的选择，我也有设计：专业栏使用下拉框进行选择，而生日栏则使用日期小弹窗进行选择，被选中的选项都是用主题绿色。

信息展示区由一个面板中放入一个表格组成。表格的表头颜色设置为与左侧的表单一致，更加美观。并且设置为可以左右上下滚动，以展示出全部信息。

下方的功能栏的设计灵感来自于苹果的程序坞，既能很好的展示功能按键又

可以提升整个界面的美观程度。功能栏里一共五个功能按钮，采用白色图标来释义，主题颜色作为背景。

2. 功能介绍

网页前端主要包括以下功能，对于其具体的效果可以观看 video 文件夹中的 video1.mp4 文件。

（1）提交数据

填写完表单所有内容之后点击提交按钮，验证成功后数据会以 json 格式发送到服务器，存储到数据库当中。提交成功之后上方会出现提示，并且将已有的全部数据以表格的形式展示在右侧的信息展示区域（可以左右拖动滑轮查看全部信息）。

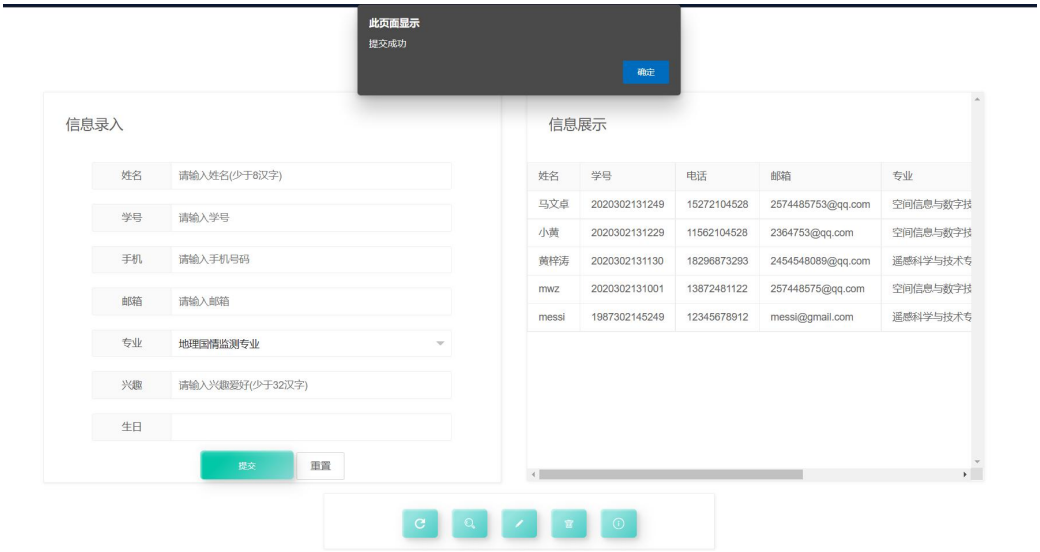


Figure 5:提交数据

（2）验证数据

对于填入表单的数据进行验证，验证规则如下：

①姓名不超过 8 个字符且不能带数字

②学号为 13 位数字，开头四位代表入学年份，应为 1900-2022 间的一年

③手机为 11 位数字，且开头第一位应为 1

④邮箱应该包含@和.，且两者不能在开头和结尾

⑤兴趣不超过 32 个字符

⑥生日格式应该为“年-月-日”，且时间因为 1920-2022 之间的一天

在实际操作当中，如果不符合数据规范，中上方会给出相应提示。

Figure 6:验证提示

（3）重置数据

对于未提交的表单数据，可以按下重置按钮一键清空表单。

（4）刷新数据

点击功能栏第一个刷新按钮，网页会再次向服务器发起请求，获取数据库中所有数据，展示在右边的信息展示区域，并给出刷新成功的提示，否则给出刷新失败的提示。

Figure 7:刷新数据

（5）查询数据

点击功能栏第二个查询按钮，正上方弹出的查询框内提示了查询格式为【字段名=查询值】，若按照正确的查询格式进行查询，则给出查询成功的提示，并将结果展示在右侧的信息展示栏当中；若查询格式错误，则给出查询格式错误的提示。

Figure 8:查询数据

（6）更新数据

点击功能栏的第三个更新按钮，在正上方弹出的更新框内提示了更新格式为

【学号&权限密码&字段名 1=更新值 1&字段名 2=更新值 2&...】。这里的更新是更新指定学号的数据。同时为了安全性考虑，我们设置了权限密码的机制，只有权限密码正确时才可进行更新，否则提醒“密码错误您无此权限”。若格式正确且密码正确，则执行更新，并且将更新后的结果展示在右侧的信息展示区域。若格式错误则给出提示。若更新的学号不存在，则给出“无此学号”的提示。

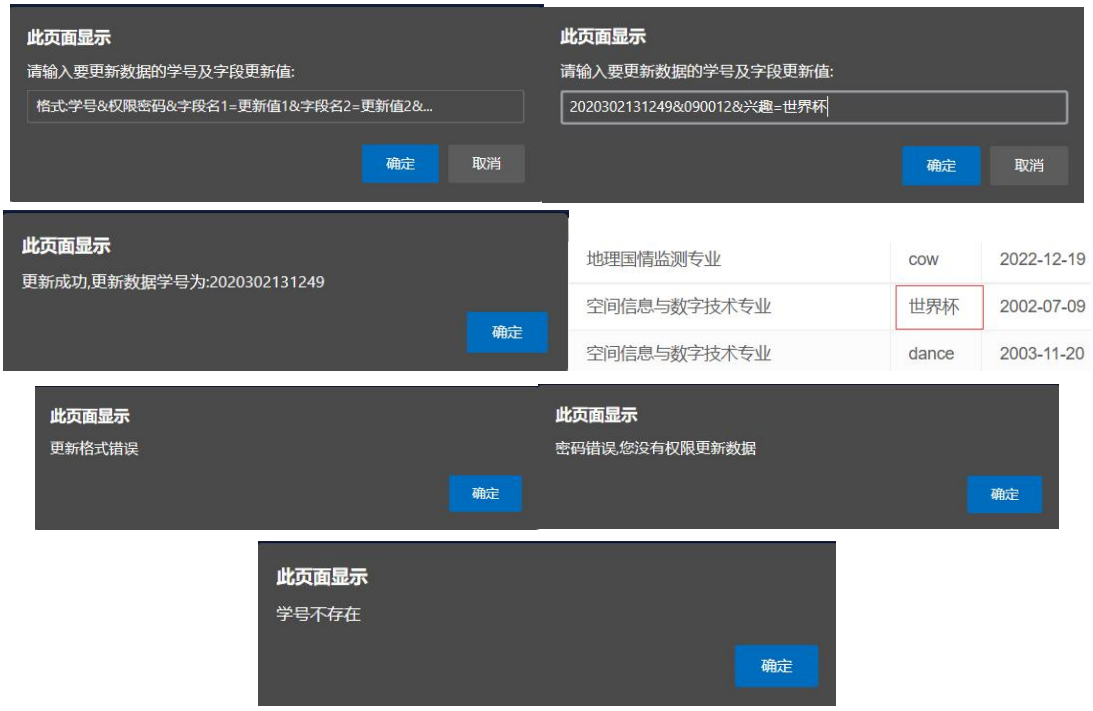


Figure 9:更新数据

（7）删除数据

点击功能栏的第四个删除按钮，正上方弹出的删除框内提示了删除数据的格式为【学号&权限密码】。为了安全，删除数据我们也设置了权限密码，只有格式正确且密码正确时才执行删除，若学号存在，则删除该条数据后展示剩余数据在有侧的信息展示栏当中，若学号不存在，则给出提示。

Figure 10:删除数据

（8）操作提示

点击功能栏最后一个提示按钮，在正上方弹出的小窗口内会展示关于项目的基础信息如作者、核心技术、版本信息等，并且会展示网页的操作提示。

Figure 11:操作提示

3. 代码概述

如上文所述，MyData 的网页前端由 html+css+javascript 编写，采用 jQuery 的 jajx 进行 http 请求。其中关于 html 和 css 就不过多赘述（详细可见 client.html 文件，其中有详细注释）。这里主要对 javascript 部分进行重点叙述。

（1）整体架构

```
<script>
    //日期选择
    layui.use('laydate', function () {…
    //获取数据
    function getData() {…
    //验证数据
    function validData(data) {…
    //提交数据
    function submitData() {…
    //刷新数据
    function refreshData(){…
    //查询数据
    function retrieveData(){…
    //更新数据
    function updateData(){…
    //删除数据
    function deleteData(){…
    //提示数据
    function helpData(){…
    //重置数据
    function resetData() {…
    //表格渲染
    function showData(data){…
</script>
```

Figure 12:js 代码架构

如上图所示，网页前端主要由 11 个函数构成，分别实现不同的功能。其中 submitData()、refreshData()、retrieveData()、updateData()、deleteData()涉及到向

服务器发送请求并接受来自服务器的相应。validData()负责对表单数据进行正则验证；getData()负责收集表单数据并进行格式转换；resetData()负责重置表单；showData()主要是帮助将数据美观的展示在右侧的信息展示区当中；helpData()负责给出操作提示。

（2）jajx 发送请求

Http 定义了四种常用的与服务器交互的方法，分别是 GET、POST、Put、DELETE。

• GET 请求会向服务器发送索取数据的请求，从而来获取信息，该请求就像数据库的 select 操作一样，只是用来查询一下数据，不会修改、增加数据，不会影响资源的内容，即该请求不会产生副作用。无论进行多少次操作，结果都是一样的。

• PUT 请求是向服务器端发送数据的，从而改变信息，该请求就像数据库的 update 操作一样，用来修改数据的内容，但是不会增加数据的种类等，也就是说无论进行多少次 PUT 操作，其结果并没有不同。

• POST 请求同 PUT 请求类似，都是向服务器端发送数据的，但是该请求会改变数据的种类等资源，就像数据库的 insert 操作一样，会创建新的内容。几乎目前所有的提交操作都是用 POST 请求的。

• DELETE 请求顾名思义，就是用来删除某一个资源的，该请求就像数据库的 delete 操作。

本网页使用 jajx 进行 http 请求，其中增添数据使用 Post 请求；查询\刷新数据使用 GET 请求；更新数据使用 PUT 请求；删除数据使用 DELETE 请求。其具体写法如下图所示：

```
$.ajax({//向服务器发送数据
    type: 'GET',
    url: "http://localhost:5500/refreshData?method=descend",
    contentType: 'application/json; charset=UTF-8',
    success: function (res) { //成功的话，得到消息
        console.log('刷新成功');
        alert("刷新成功");
        showData(res);//刷新表格
    },
    error: function (responseStr) {
        console.warn('刷新失败', responseStr);
        alert("刷新失败");
    }
});
```

```
$.ajax({//向服务器发送数据
    type: 'POST',
    url: "http://localhost:5500/submitData",
    data: JSON.stringify(data),  //转化字符串
    contentType: 'application/json; charset=UTF-8',
    success: function (res) { //成功的话，得到消息
        console.log('提交成功');
        alert("提交成功");
        showData(res);//刷新表格
    },
    error: function (responseStr) {
        console.warn('提交失败', responseStr);
        alert("提交失败");
    }
});
```

```
$.ajax({//向服务器发送数据
    type: 'PUT',
    url: "http://localhost:5500/updateData",
    data: JSON.stringify(data),  //转化字符串
    contentType: 'application/json; charset=UTF-8',
    success: function (res) { //成功的话，得到消息
        if(res.length==0){
            console.warn('学号不存在');
            alert('学号不存在');
        }
        else{
            console.log('更新成功');
            alert("更新成功,更新数据学号为:"+arr[0]);
            showData(res);//刷新表格
        }
    },
    error: function (responseStr) {
        console.warn('更新失败', responseStr);
        alert("更新失败");
    }
});
```

```
$.ajax({//向服务器发送数据
    type: 'DELETE',
    url: "http://localhost:5500/deleteData",
    data: JSON.stringify({'id':arr[0]}),  //转化字符串
    contentType: 'application/json; charset=UTF-8',
    success: function (res) { //成功的话，得到消息
        console.log(res);
        if(res.length==0){
            alert('查无此学号');
            console.warn('查无此学号');
        }
        else{
            console.log('删除成功');
            alert("删除成功,删除数据学号为:"+text);
            showData(res);//刷新表格
        }
    },
    error: function (responseStr) {
        console.warn('删除失败', responseStr);
        alert("删除失败");
    }
```

Figure 13:jajx 的四种请求

（3）幂等性及安全性讨论

幂等性：就是用户对于同一操作发起的一次请求或者多次请求的结果是一致的，不会因为多次点击而产生了副作用。安全性则是指操作的安全与否。根据所学知识可知，GET 请求安全且幂等，POST 请求既不安全也不幂等，PUT 和 DELETE 请求均不安全但幂等。

考虑到 PUT 和 DELETE 请求的不安全性，我们通过增加权限密码的操作来限制数据的删除和更新，这样给数据库当中的数据增添了一份保障。

```
if(text.indexOf("&")!=-1){//格式正确
    var arr=text.split("&");
    if(arr[1]=='090012'){//权限密码正确
        $.ajax({//向服务器发送数据
```

Figure 14:权限密码

考虑到 POST 方法的不幂等性，在客户端我们采用【提交之后确认】的方式来避免多次重复提交；同时服务端在将数据写入数据库当中时，我们使用【replace】语句（即存在相同数据则替换，而不是重复添加）来避免多次重复请求。

```
success: function (res) { //成功的话，得到消息
    console.log('提交成功');
    alert("提交成功");
    showData(res);//刷新表格
},
```

Figure 15:提交之后确认

（4）正则验证

对于数据的格式规范我们采用正则表达式的方式在客户端进行验证（验证规则见本章第 2 节）。具体如下：

1>验证是否填写完整

```
if(data.name==''||data.id==''||data.phonenumber==''||data.email==''||data.major==''||data.interest==''||data.birthday==''){
    console.warn('EmptyErr:请填写所有信息');
    return 0;
}
```

Figure 16:完整性验证

2>验证姓名

```
//验证姓名
var regex1 = /\d/;
if(regex1.test(data.name) === true) {//不能有数字
    console.warn("student-numberErr:请输入正确的姓名");
    return -1;
}
```

Figure 17:姓名验证

3>验证学号

```
// 验证学号
var regex2 = /^(19\d{11}|20([0-1]\d{10}|2[0-2]\d{9}))$/;
if(regex2.test(data.id) === false) {
    console.warn("student-numberErr:请输入正确的学号");
    return -2;
}
```

Figure 18:学号验证

4>验证电话

```
//验证电话
var regex3 = /^[1]\d{10}$/;
if(regex3.test(data.phonenumber) === false) {
    console.warn("tele-numberErr:请输入正确的手机号码");
    return -3;
}
```

Figure 19:电话验证

5>验证邮箱

```
//验证邮箱
var regex4 = /^[A-Za-z\d]+([-_.][A-Za-z\d]+)*@([A-Za-z\d]+[-.])+[A-Za-z\d]{2,4}$/;
if(regex4.test(data.email) === false) {
    console.warn("emailErr:请输入正确的邮箱地址");
    return -4;
}
```

Figure 20:邮箱验证

6>验证生日

```
//验证生日
var regex5 = /^((19[2-9]\d{1})|(20(([0-1][0-9])|(2[0-2]))))\-((0?[1-9])|(1[0-2]))\-((0?[1-9])|([1-2][0-9])|30|31)$/;
if(regex5.test(data.birthday) === false) {
    console.warn("emailErr:请输入正确的生日");
    return -5;
}
```

Figure 21:生日验证

（5）代码规范

本着代码规范的原则，本网页在后滩输出了几乎所有的错误警告和中间结果以便于更新代码、修改 bug。同时在重要位置均有注释。

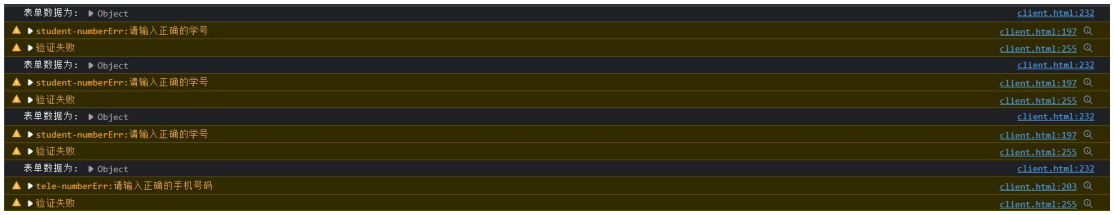


Figure 22:网页控制台信息

四. 命令行 App

1. 功能介绍

命令行 App 主要包括以下功能，对于其具体的效果可以观看 video 文件夹中的 video2.mp4 文件。

（1）罗列数据

输入【App -l|--list descend|ascend】即可罗列出数据库当中所有数据，其中默认为为 descend 排列（除非指定排序规则为 ascend）。值得注意的是，作为保护机制，当输入【App】时，也会以 descend 的顺序罗列出数据。

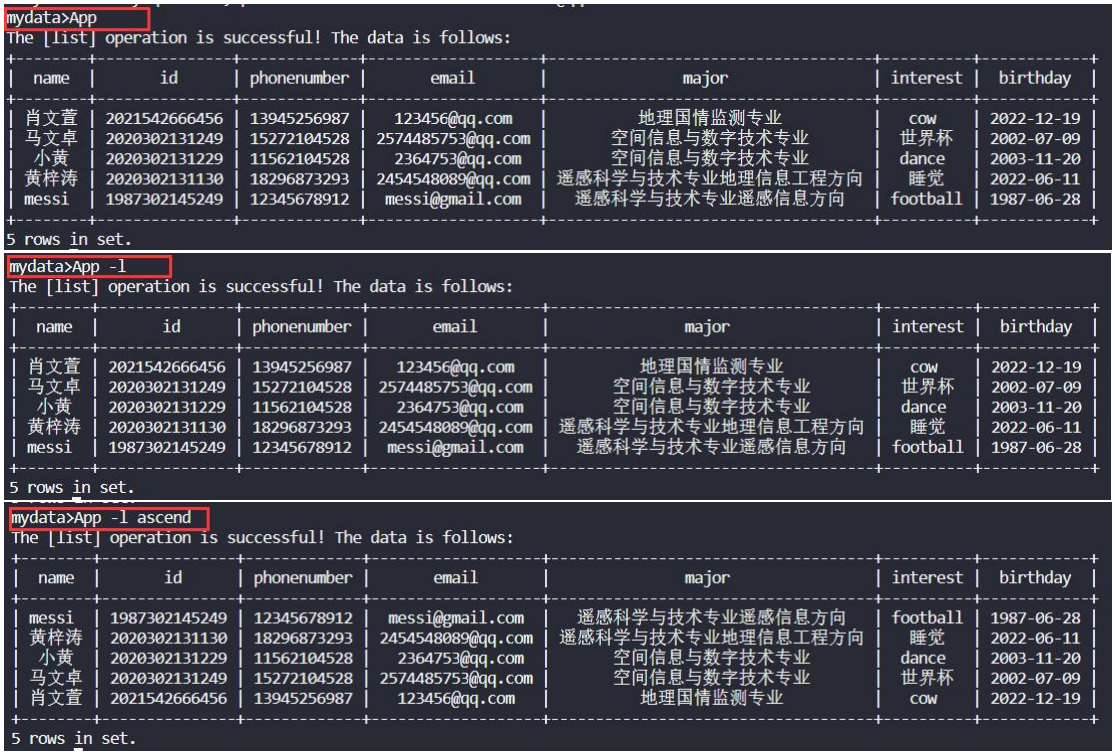


Figure 23:罗列数据

（2）增添数据

输入【App -a|--add -n|--name [value1] -i|--id [value2] -p|--phonenumber [value3] -e|--email [value4] -m|--major [value5] -h|--hobby [value6] -b|--birthday [value7]】即可增加一条数据。

Figure 24:增添数据

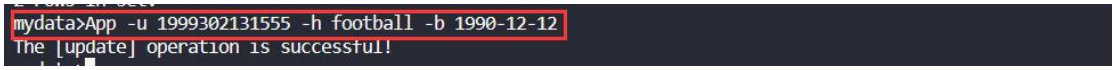
（3）查询数据

输入【App -r|--retrieve [field] [value]】即可查询出 field=value 的数据，并且展示出来。

Figure 25:查询数据

（4）更新数据

输入【App -u|--update [id] [field1] [value1] [field2] [value2] ...】即可对 id=[id] 的数据进行相应字段的更新。



name	id	phonenumber	email	major	interest	birthday
肖文萱	2021542666456	13945256987	123456@qq.com	地理国情监测专业	cow	2022-12-19
马文卓	2020302131249	15272104528	2574485753@qq.com	空间信息与数字技术专业	世界杯	2002-07-09
小黄	2020302131229	11562104528	2364753@qq.com	空间信息与数字技术专业	dance	2003-11-20
黄梓涛	2020302131130	18296873293	2454548089@qq.com	遥感科学与技术专业地理信息工程方向	睡觉	2022-06-11
c罗	1999302131555	13768596677	3578609@qq.com	遥感科学与技术专业摄影测量方向	football	1990-12-12
messi	1987302145249	12345678912	messi@gmail.com	遥感科学与技术专业遥感信息方向	football	1987-06-28

Figure 26:更新数据

（5）删除数据

输入【App -d|--delete [id]】即可删除数据中 id=[id]的条目。

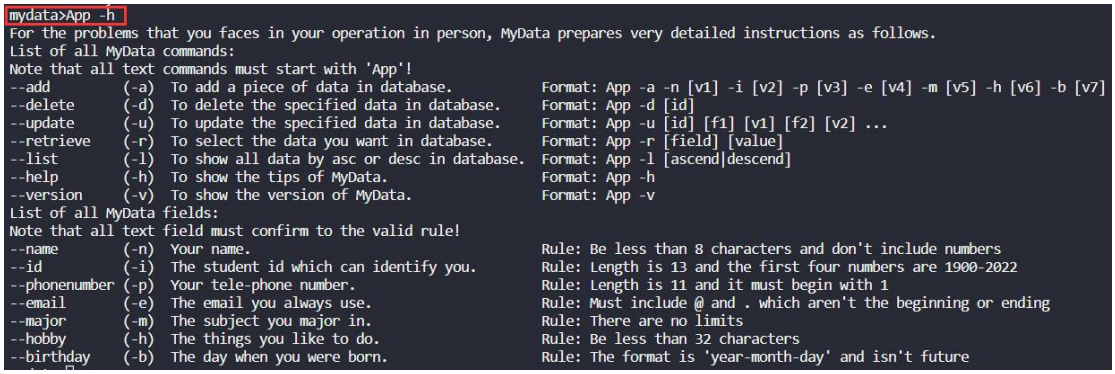
```
mydata>App -d 1999302131555
The [delete] operation is successful!
```

name	id	phonenumber	email	major	interest	birthday
肖文萱	2021542666456	13945256987	123456@qq.com	地理国情监测专业	cow	2022-12-19
马文卓	2020302131249	15272104528	2574485753@qq.com	空间信息与数字技术专业	世界杯	2002-07-09
小黄	2020302131229	11562104528	2364753@qq.com	空间信息与数字技术专业	dance	2003-11-20
黄梓涛	2020302131130	18296873293	2454548089@qq.com	遥感科学与技术专业地理信息工程方向	睡觉	2022-06-11
messi	1987302145249	12345678912	messi@gmail.com	遥感科学与技术专业遥感信息方向	football	1987-06-28

Figure 27:删除数据

（6）查阅帮助文档

输入【App -h|--help】即可显示出帮助文档，其中包含了操作命令、具体功能解释、使用格式、字段大全、字段规则等。

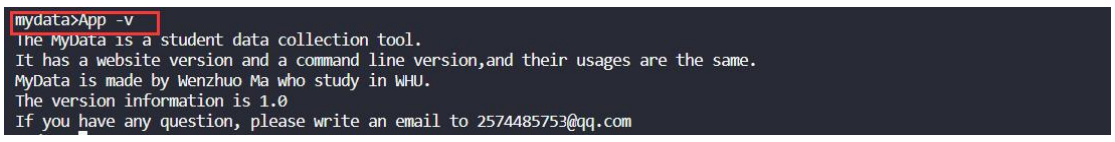


```
mydata>App -h
For the problems that you faces in your operation in person, MyData prepares very detailed instructions as follows.
List of all MyData commands:
Note that all text commands must start with 'App'!
--add         (-a)  To add a piece of data in database.          Format: App -a -n [v1] -i [v2] -p [v3] -e [v4] -m [v5] -h [v6] -b [v7]
--delete      (-d)  To delete the specified data in database.    Format: App -d [id]
--update      (-u)  To update the specified data in database.    Format: App -u [id] [f1] [v1] [f2] [v2] ...
--retrieve    (-r)  To select the data you want in database.     Format: App -r [field] [value]
--list        (-l)  To show all data by asc or desc in database. Format: App -l [ascend|descend]
--help        (-h)  To show the tips of MyData.                 Format: App -h
--version     (-v)  To show the version of MyData.              Format: App -v
List of all MyData fields:
Note that all text field must confirm to the valid rule!
--name        (-n)  Your name.                                  Rule: Be less than 8 characters and don't include numbers
--id          (-i)  The student id which can identify you.      Rule: Length is 13 and the first four numbers are 1900-2022
--phonenumber (-p)  Your tele-phone number.                     Rule: Length is 11 and it must begin with 1
--email       (-e)  The email you always use.                   Rule: Must include @ and . which aren't the beginning or ending
--major       (-m)  The subject you major in.                   Rule: There are no limits
--hobby       (-h)  The things you like to do.                  Rule: Be less than 32 characters
--birthday    (-b)  The day when you were born.                 Rule: The format is 'year-month-day' and isn't future
```

Figure 28:查阅帮助文档

（7）查阅版本信息

输入【App -v|--version】即可显示版本信息，其中包含了作者、版本号、问题提交渠道等信息。



```
mydata>App -v
The MyData is a student data collection tool.
It has a website version and a command line version,and their usages are the same.
MyData is made by Wenzhuo Ma who study in WHU.
The version information is 1.0
If you have any question, please write an email to 2574485753@qq.com
```

Figure 29:查阅版本信息

（8）退出

输入【App -e|--exit】即可退出 MyData。

```
mydata>App -e
Byebye~
```

Figure 30:退出 MyData

（9）错误提示

除了以上的功能以外，MyData 的命令行版本与其网页版一样具有非常完整

的错误提示系统。其错误系统一共包含 27 种错误，可分为三大类别：命令语法错误、数据验证错误、http 请求错误。

命令语法类错误主要是每种命令应该合乎规范。例如，命令前未添加【App】、同时使用多个操作如【App -a -d】、缺少字段如【App -a -n value -i value】、多余字段如【App -l -n】、字段错误【App -a -c】、操作错误【App -c】等等。下面列举出其中几种：

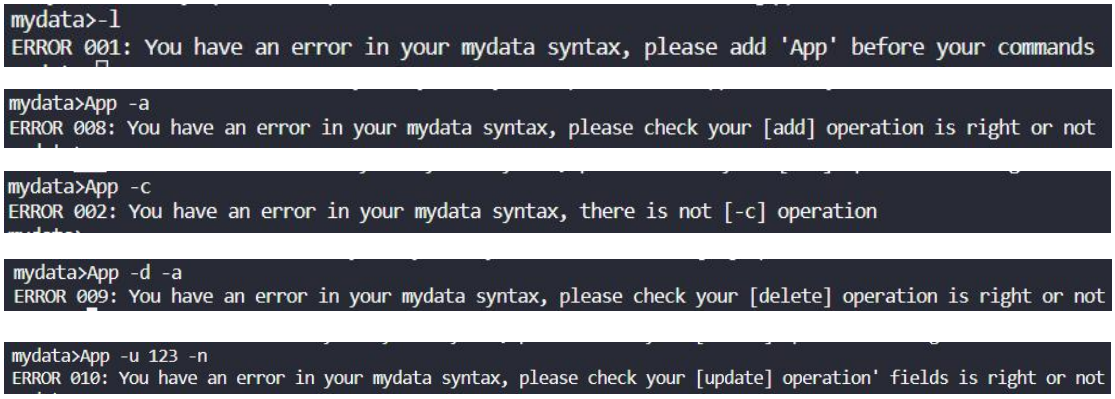


Figure 31:命令语法类错误示例

数据验证类错误即与网页前端的验证规则一致，若在添加或更新数据时数据不合乎规范，则出现相应的错误提示。下面列举几种：

Figure 32:数据验证类错误示例

http 请求类错误主要是客户端与服务端交互出现的错误。如请求失败（会打印出详细错误信息和状态码）、删除和更新的学号不存在等。

Figure 33:http 请求类错误示例

2. 代码概述

（1）整体架构

本命令行 App 版本主要由一个类及 11 个函数组成。其中 add()、retrieve()、update()、list()、delete()涉及到与服务端的交互。listen()负责监听终端的输入；help()负责展示帮助文档；version()展示版本信息；exit()退出程序；display()美观地展示数据；valid()验证数据格式是否合乎规范。具体如下图所示：

Figure 34:App 代码架构

（2）requests 发送请求

本 App 程序使用 python 的 requests 库来向服务器发送 http 请求。和网页版一样，增添数据使用 POST 请求，查询数据使用 GET 请求，删除数据使用 DELETE 请求，更新数据使用 PUT 请求。

```
def retrieve(self,data):
    '''查询操作'''                    GET请求
    try:
        res=requests.get("http://localhost:5500/retrieveData?flag={}&value={}".format(self.fieldMap[data[0]],data[1]))
        res.raise_for_status()#获取异常
        result=res.json()#获取返回数据
        print('The [retrieve] operation is successful! The result is followed:')
        self.display(result)
    except HTTPError as e:
        print('ERROR 17: The [retrieve] operation is failed. The status code is '+str(res.status_code))
        print('The detail is followed:\n',e)
```

Figure 35:GET 请求

```
def add(self,data):
    '''添加操作'''
    submit={}
    for i in range(0,len(data),2):#提交数据预处理
        submit[self.fieldMap[data[i]]]=data[i+1]
    submit=json.dumps(submit)                    POST请求
    try:
        res=requests.post("http://localhost:5500/submitData", headers={"Content-Type": "application/json"},data=submit)
        res.raise_for_status()#获取异常
        print('The [add] operation is successful!')
    except HTTPError as e:
        print('ERROR 16: The [add] operation is failed. The status code is '+str(res.status_code))
        print('The detail is followed:\n',e)
```

Figure 36:POST 请求

```python
def update(self,data):
    '''更新操作'''
    #处理数据
    submit={'id':data[0]}
    for i in range(1,len(data),2):
        submit[self.fieldMap[data[i]]]=data[i+1]
    submit=json.dumps(submit)
    try:
        res=requests.put("http://localhost:5500/updateData",headers={"Content-Type": "application/json"},data=submit)
        res.raise_for_status()#获取异常
        result=res.json()#获取返回数据
        if result==[]:
            print('ERROR 18: The id you want to update doesn\'t exist.')
        else:
            print('The [update] operation is successful!')
    except HTTPError as e:
        print('ERROR 19: The [update] operation is failed. The status code is '+str(res.status_code))
        print('The detail is followed:\n',e)
```

PUT请求

Figure 37:PUT 请求

```python
def delete(self,data):
    '''删除操作'''
    submit={'id':data}
    try:
        res=requests.delete("http://localhost:5500/deleteData",json=submit)
        res.raise_for_status()#获取异常
        result=res.json()#获取返回数据
        if result==[]:
            print('ERROR 20: The id you want to delete doesn\'t exist.')
        else:
            print('The [delete] operation is successful!')
    except HTTPError as e:
        print('ERROR 21: The [delete] operation is failed. The status code is '+str(res.status_code))
        print('The detail is followed:\n',e)
```

DELETE请求

Figure 38:DELETE 请求

（3）正则验证

和上面网页端一样，数据格式的验证主要还是通过正则表达式进行验证，这里就不过多赘述。

```python
def valid(self,data):
    '''验证'''
    for i in range(0,len(data),2):#逐项验证
        if self.fieldMap[data[i]]=='name':#验证姓名
            if bool(re.search(r'\d',data[i+1]))==True or len(data[i+1])>8:#名字不能含有数字且不能超过8个字符
                print('ERROR 22: The [name] field is not valid.')
                return False
        if self.fieldMap[data[i]]=='id':#验证学号
            if bool(re.search(r'^(19\d{11}|20([0-1]\d{10}|2[0-2]\d{9}))$',data[i+1]))==False:#学号13字符，开头四位代表年份(
                print('ERROR 23: The [id] field is not valid.')
                return False
        if self.fieldMap[data[i]]=='phonenumber':#验证电话
            if bool(re.search(r'^[1]\d{10}$',data[i+1]))==False:#电话11位数字，开头为1
                print('ERROR 24: The [phonenumber] field is not valid.')
                return False
        if self.fieldMap[data[i]]=='email':#验证邮箱
            if bool(re.search(r'^[A-Za-z\d]+([-_.][A-Za-z\d]+)*@([A-Za-z\d]+[-.])+[A-Za-z\d]{2,4}$',data[i+1]))==False:#
                print('ERROR 25: The [email] field is not valid.')
                return False
        if self.fieldMap[data[i]]=='interest':#验证兴趣
            if len(data[i+1])>32:#兴趣不能超过32个字符
                print('ERROR 26: The [interest] field is not valid.')
                return False
        if self.fieldMap[data[i]]=='birthday':#验证生日
            if bool(re.search(r'^((19[2-9]\d{1})|(20(([0-1][0-9])|(2[0-2]))))\-((0?[1-9])|(1[0-2]))\-((0?[1-9])|([1-2][0-
                print('ERROR 27: The [birthday] field is not valid.')
                return False
    return True
```

Figure 39:数据验证

（4）鲁棒性

上文提到本 App 有三类共 27 种错误提示，可以非常好的帮助用户定位错误位置。（可见本章功能介绍）

同时为了保证程序运行的稳定性，所有可能出现错误的操作（特别是与服务器之间的交互）均使用 try...except...语句，若果出错会打印出相应的错误信息并且不会阻碍 App 的正常运行。

```
try:
    res=requests.get("http://localhost:5500/refreshData?method={}".format(method))#发送get请求
    res.raise_for_status()#获取异常
    result=res.json()#获取返回数据
    print('The [list] operation is successful! The data is follows:')
    self.display(result)
except HTTPError as e:
    print('ERROR 15: The [list] operation is failed. The status code is '+str(res.status_code))
    print('The detail is followed:\n',e)
```

Figure 40:异常获取

五．服务器

1. 代码概述

（1）总体架构

MyData 的服务器使用 Python 的 flask 架构编写，使用 pymysql 与数据库进行交互。其一共由一下 13 个函数构成，右侧的五个为路由函数，负责接收客户端发送来的请求，并对其进行响应。左侧的 connectDB()、closeDB()负责控制数据库连接的开启和关闭；start()用于启动服务器；add()、retrieve()、update()、delete()负责对数据库表格进行增删改查的操作；transform()负责数据的格式转换，用于将数据库中得到的数据转换为便于客户端使用的形式传输过去。

```
def connectDB():···

def closeDB(db,cursor):···

def start():···

def add(db,cursor,data):···

def retrieve(db,cursor,flag='*',data='*',method='descend'):

def update(db,cursor,data):···

def delete(db,cursor,flag,data):···

def transform(data):···
```

```
@app.route('/submitData',methods=['POST'])
@cross_origin()#解决跨域请求
def submitData():···

@app.route('/refreshData',methods=['GET'])
@cross_origin()#解决跨域请求
def refreshData():···

@app.route('/retrieveData',methods=['GET'])
@cross_origin()#解决跨域请求
def retrieveData():···

@app.route('/updateData',methods=['PUT'])
@cross_origin()#解决跨域请求
def updateData():···

@app.route('/deleteData',methods=["DELETE"])
@cross_origin()#解决跨域请求
def deleteData():···
```

Figure 41:MyData 服务器代码架构

（2）Flask 响应请求

Flask 架构主要是通过路由的形式来接收客户端的请求，并且进行响应。其也可限制可接收的请求类型，这里对应客户端相应操作的请求类型，编写相应的服务端接收函数。其处理流程基本为：接收客户端数据->按要求对数据库进行操作->对数据库返回的数据进行处理->发送给客户端。

```python
@app.route('/submitData',methods=['POST'])
@cross_origin()#解决跨域请求
def submitData():
    '''数据提交'''
    print('-----------提交数据开始-----------')
    db,cursor=connectDB()#连接数据库
    data=request.get_json()#获取数据
    print('接收到数据:',data)
    add(db,cursor,data)#插入数据
    result=retrieve(db,cursor,'*','*')#查询获得数据库中所有数据
    result=transform(result)#格式转换
    closeDB(db,cursor)#关闭数据库连接
    print('-----------提交数据结束-----------')
    return result
```

Figure 42:POST 请求接收函数

```python
@app.route('/retrieveData',methods=['GET'])
@cross_origin()#解决跨域请求
def retrieveData():
    print('-----------查询数据开始-----------')
    db,cursor=connectDB()#连接数据库
    data={'flag':request.args.get('flag'),'value':request.args.get('value')}#获取数据
    print("查询【{}={}】的数据".format(data['flag'],data['value']))
    result=retrieve(db,cursor,data['flag'],data['value'])#查询
    result=transform(result)#格式转换
    closeDB(db,cursor)#关闭数据库连接
    print('-----------查询数据结束-----------')
    return result
```

Figure 43:GET 请求接收函数

```python
@app.route('/updateData',methods=['PUT'])
@cross_origin()#解决跨域请求
def updateData():
    print('-----------更新数据开始-----------')
    db,cursor=connectDB()#连接数据库
    data=request.get_json()#获取数据
    result=retrieve(db,cursor,'id',data['id'])#查询是否存在这个学号
    if result==():
        print('学号不存在')
        result=[]
    else:
        print('学号存在')
        update(db,cursor,data)#更新数据
        result=retrieve(db,cursor,'*','*')
        result=transform(result)
    closeDB(db,cursor)#关闭数据库连接
    print('-----------更新数据结束-----------')
    return result
```

Figure 44:PUT 请求接收函数

```python
@app.route('/deleteData',methods=["DELETE"])
@cross_origin()#解决跨域请求
def deleteData():
    print('-----------删除数据开始-----------')
    db,cursor=connectDB()#连接数据库
    data=request.get_json()#获取数据
    result=[]
    if retrieve(db,cursor,'id',data['id'])==():#查询看是否存在此学号
        print('学号不存在')
        result=[]
    else:
        delete(db,cursor,'id',data['id'])#删除数据
        result=retrieve(db,cursor,'*','*')#查询
        result=transform(result)#格式转换
    closeDB(db,cursor)#关闭数据库连接
    print('-----------删除数据结束-----------')
    return result
```

Figure 45:DELETE 请求接收函数

（3）数据库操作

主要通过 pymysql 对 mysql 数据库进行操作。操作内容主要是通过 sql 语句进行增删改查。值得注意的是，为了保证服务器平稳运行，所有的数据库操作都是用 try...except...语句进行异常抛出，且若出现异常打印错误信息的同时还会对数据库操作进行回滚。具体如下：

```python
def add(db,cursor,data):
    '''增加数据'''
    #如果之前不存在，则插入；如果之前存在，则更新
    sql='replace into student values (\'{}\',\'{}\',\'{}\',\'{}\',\'{}\',\'{}\',\'{}\');'\
        .format(data['name'],data['id'],data['phonenumber'],\
        data['email'],data['major'],data['interest'],data['birthday'])
    try:
        cursor.execute(sql)#执行sql语句
        db.commit()#提交事务
        print("成功插入:",data)
    except Exception as e:
        print("插入操作出现错误:{}".format(e))
        db.rollback()# 回滚所有更改
```

Figure 46:数据库操作-增

```python
def retrieve(db,cursor,flag='*',data='*',method='descend'):
    '''查询数据'''
    if method=='descend':
        method='desc'
    else:
        method='asc'
    sql=''
    if flag=='*' and data=='*':
        sql='select * from student order by id {};'.format(method)
    else:
        sql='select * from student where {} = \'{}\' order by id {};'.format(flag,data,method)
    try:
        cursor.execute(sql)
        result = cursor.fetchall()#获取查询结果
        print("查询【{}={}】成功".format(flag,data))
        return result
    except Exception as e:
        print("查询操作出现错误:{}".format(e))
        return None
```

Figure 47:D 数据库操作-查

```python
def update(db,cursor,data):
    '''更新数据'''
    sql='update student set '
    for i in data.keys():
        if i=='id':
            continue
        sql+=i+' = '+'\''+data[i]+'\','
    sql=sql[:-1]
    sql+=' where id = \'{}\';'.format(data['id'])
    print(sql)
    try:
        cursor.execute(sql)#执行sql语句
        db.commit()#提交事务
        print("成功更新:",data)
    except Exception as e:
        print("更新操作出现错误:{}".format(e))
        db.rollback()# 回滚所有更改
```

Figure 48:数据库操作-改

```python
def delete(db,cursor,flag,data):
    '''删除数据'''
    sql=''
    if flag=='*' and data=='*':#删除表中所有数据
        sql='truncate table student;'
    else:
        sql='delete from student where {} = \'{}\';'.format(flag,data);
    try:
        cursor.execute(sql)#执行sql语句
        db.commit()#提交事务
        print("成功删除"+flag+'为'+data+'的数据')
    except Exception as e:
        print("删除操作出现错误:{}".format(e))
        db.rollback()# 回滚所有更改
```

Figure 49:数据库操作-删

（4）跨域问题

对于网页客户端曾出现的跨域问题，这是由于同源策略所引起的，导致 jajx 当中无法进入 success 部分，无法完成交互。解决办法是，引入 flask_cors 库，通过在路由函数之前添加【@cross_origin()】来解决。具体如下：

```
Access to XMLHttpRequest at 'http://localhost:8090/user/getotp' from origin 'null' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.   getotp.html:1
Failed to load resource: net::ERR_FAILED   localhost:8090/user/getotp:1
```

```python
@app.route('/refreshData',methods=['GET'])
@cross_origin()#解决跨域请求
def refreshData():…
```

Figure 50:跨域问题及其解决

（5）后台记录

为了保证 MyData 的稳定运行及后续更新，本次代码打印了几乎所有的服务器活动到终端。我们可以通过终端监视客户端做了什么，服务器哪里出现了问题

等。

```
Press CTRL+C to quit
-----------刷新数据开始-----------
连接mysql数据库students成功
查询【*=*】成功
数据库连接关闭成功
刷新成功
-----------刷新数据结束-----------
127.0.0.1 - - [23/Dec/2022 23:46:31] "GET /refreshData?method=descend HTTP/1.1" 200 -
-----------刷新数据开始-----------
连接mysql数据库students成功
查询【*=*】成功
数据库连接关闭成功
刷新成功
-----------刷新数据结束-----------
127.0.0.1 - - [23/Dec/2022 23:52:47] "GET /refreshData?method=descend HTTP/1.1" 200 -
-----------提交数据开始-----------
连接mysql数据库students成功
127.0.0.1 - - [24/Dec/2022 00:01:00] "POST /submitData HTTP/1.1" 400 -
```

Figure 51:服务器后台信息

2. API 测试

对于服务器的 API 我们使用 curl 进行了测试，结果如下。

（1）list 操作

• 命令： curl http://localhost:5500/refreshData?method=descend

```
(myPython) D:\本科\信息集成与管理\实习\mwz\dace>curl http://localhost:5500/refreshData?method=descend
[{"birthday":"2022-12-19","email":"123456@qq.com","id":"2021542666456","interest":"cow","major":"\u5730\u7406\u56fd\u60c5\u76d1\u6d4b\u4e13\u4e1a"
,"name":"\u8096\u6587\u8431","phonenumber":"13945256987"},{"birthday":"2002-07-09","email":"2574485753@qq.com","id":"2020302131249","interest":"\u
4e16\u754c\u676f","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b57\u6280\u672f\u4e13\u4e1a","name":"\u9a6c\u6587\u5353","phonenumber":"15272104
528"},{"birthday":"2003-11-20","email":"2364753@qq.com","id":"2020302131229","interest":"dance","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b5
7\u6280\u672f\u4e13\u4e1a","name":"\u5c0f\u9ec4","phonenumber":"11562104528"},{"birthday":"2022-06-11","email":"2454548089@qq.com","id":"202030213
1130","interest":"\u7761\u89c9","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u5730\u7406\u4fe1\u606f\u5de5\u7a0b\u65b9\u5411","
name":"\u9ec4\u6893\u6d9b","phonenumber":"18296873293"},{"birthday":"1987-06-28","email":"messi@gmail.com","id":"1987302145249","interest":"footba
ll","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u9065\u611f\u4fe1\u606f\u65b9\u5411","name":"messi","phonenumber":"12345678912
"}]
```

Figure 52:curl 测试 list 接口

（2）add 操作

• 命令： curl http://localhost:5500/submitData -H "Content-Type: application/json" -X POST -d "{\"name\":\"jack\",\"id\":\"2020302131002\",\"phonenumber\":\"15272104545\",\"email\":\"25744@qq.com\",\"major\":\"remote sense\",\"interest\":\"sleep\",\"birthday\":\"2000-01-01\"}"

```
(myPython) D:\本科\信息集成与管理\实习\mwz\dace>curl http://localhost:5500/submitData -H "Content-Type: application/json" -X POST -d "{\"name\":\"
jack\",\"id\":\"2020302131002\",\"phonenumber\":\"15272104545\",\"email\":\"25744@qq.com\",\"major\":\"remote sense\",\"interest\":\"sleep\",\"bir
thday\":\"2000-01-01\"}"
[{"birthday":"2022-12-19","email":"123456@qq.com","id":"2021542666456","interest":"cow","major":"\u5730\u7406\u56fd\u60c5\u76d1\u6d4b\u4e13\u4e1a
","name":"\u8096\u6587\u8431","phonenumber":"13945256987"},{"birthday":"2002-07-09","email":"2574485753@qq.com","id":"2020302131249","interest":"\u
4e16\u754c\u676f","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b57\u6280\u672f\u4e13\u4e1a","name":"\u9a6c\u6587\u5353","phonenumber":"15272104
528"},{"birthday":"2003-11-20","email":"2364753@qq.com","id":"2020302131229","interest":"dance","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b5
7\u6280\u672f\u4e13\u4e1a","name":"\u5c0f\u9ec4","phonenumber":"11562104528"},{"birthday":"2022-06-11","email":"2454548089@qq.com","id":"202030213
1130","interest":"\u7761\u89c9","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u5730\u7406\u4fe1\u606f\u5de5\u7a0b\u65b9\u5411","
name":"\u9ec4\u6893\u6d9b","phonenumber":"18296873293"},{"birthday":"2000-01-01","email":"25744@qq.com","id":"2020302131002","interest":"sleep","m
ajor":"remote sense","name":"jack","phonenumber":"15272104545"},{"birthday":"1987-06-28","email":"messi@gmail.com","id":"1987302145249","interest"
:"football","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u9065\u611f\u4fe1\u606f\u65b9\u5411","name":"messi","phonenumber":"123
45678912"}]
```

name	id	phonenumber	email	major	interest	birthday
肖文萱	2021542666456	13945256987	123456@qq.com	地理国情监测专业	cow	2022-12-19
马文卓	2020302131249	15272104528	2574485753@qq.com	空间信息与数字技术专业	世界杯	2002-07-09
小黄	2020302131229	11562104528	2364753@qq.com	空间信息与数字技术专业	dance	2003-11-20
黄梓涛	2020302131130	18296873293	2454548089@qq.com	遥感科学与技术专业地理信息工程方向	睡觉	2022-06-11
jack	2020302131002	15272104545	25744@qq.com	remote sense	sleep	2000-01-01
messi	1987302145249	12345678912	messi@gmail.com	遥感科学与技术专业遥感信息方向	football	1987-06-28

Figure 53:curl 测试 add 接口

（3）retrieve 操作

• 命令： curl -X GET -G --data-urlencode "flag=name" --data-urlencode "value=messi" -i http://localhost:5500/retrieveData

```
(myPython) D:\本科\信息集成与管理\实习\mwz\dace>curl -X GET -G --data-urlencode "flag=name" --data-urlencode "value=messi" -i http://localhost:550
0/retrieveData
HTTP/1.1 200 OK
Server: Werkzeug/2.2.2 Python/3.9.7
Date: Sat, 24 Dec 2022 05:47:12 GMT
Content-Type: application/json
Content-Length: 241
Access-Control-Allow-Origin: *
Connection: close

[{"birthday":"1987-06-28","email":"messi@gmail.com","id":"1987302145249","interest":"football","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f
\u4e13\u4e1a\u9065\u611f\u4fe1\u606f\u65b9\u5411","name":"messi","phonenumber":"12345678912"}]
```

Figure 54:curl 测试 retrieve 接口

（4）update 操作

•命令：curl http://localhost:5500/updateData -H "Content-Type: application/json" -X PUT -d "{\"id\":\"2020302131002\",\"name\":\"Merry\"}"

```
(myPython) D:\本科\信息集成与管理\实习\mwz\dace>curl http://localhost:5500/updateData -H "Content-Type: application/json" -X PUT -d "{\"id\":\"202
0302131002\",\"name\":\"Merry\"}"
[{"birthday":"2022-12-19","email":"123456@qq.com","id":"2021542666456","interest":"cow","major":"\u5730\u7406\u56fd\u60c5\u76d1\u6d4b\u4e13\u4e1a"
,"name":"\u8096\u6587\u8431","phonenumber":"13945256987"},{"birthday":"2002-07-09","email":"2574485753@qq.com","id":"2020302131249","interest":"\u
4e16\u754c\u676f","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b57\u6280\u672f\u4e13\u4e1a","name":"\u9a6c\u6587\u5353","phonenumber":"15272104
528"},{"birthday":"2003-11-20","email":"2364753@qq.com","id":"2020302131229","interest":"dance","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b5
7\u6280\u672f\u4e13\u4e1a","name":"\u5c0f\u9ec4","phonenumber":"11562104528"},{"birthday":"2022-06-11","email":"2454548089@qq.com","id":"202030213
1130","interest":"\u7761\u89c9","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u5730\u7406\u4fe1\u606f\u5de5\u7a0b\u65b9\u5411","
name":"\u9ec4\u6893\u6d9b","phonenumber":"18296873293"},{"birthday":"2000-01-01","email":"25744@qq.com","id":"2020302131002","interest":"sleep","m
ajor":"remote sense","name":"Merry","phonenumber":"15272104545"},{"birthday":"1987-06-28","email":"messi@gmail.com","id":"1987302145249","interest
":"football","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u9065\u611f\u4fe1\u606f\u65b9\u5411","name":"messi","phonenumber":"12
345678912"}]
```

name	id	phonenumber	email	major	interest	birthday
肖文萱	2021542666456	13945256987	123456@qq.com	地理国情监测专业	cow	2022-12-19
马文卓	2020302131249	15272104528	2574485753@qq.com	空间信息与数字技术专业	世界杯	2002-07-09
小黄	2020302131229	11562104528	2364753@qq.com	空间信息与数字技术专业	dance	2003-11-20
黄梓涛	2020302131130	18296873293	2454548089@qq.com	遥感科学与技术专业地理信息工程方向	睡觉	2022-06-11
Merry	2020302131002	15272104545	25744@qq.com	remote sense	sleep	2000-01-01
messi	1987302145249	12345678912	messi@gmail.com	遥感科学与技术专业遥感信息方向	football	1987-06-28

Figure 55:curl 测试 update 接口

（5）delete 操作

•命令：curl http://localhost:5500/deleteData -H "Content-Type: application/json" -X DELETE -d "{\"id\":\"2020302131002\"}"

```
(myPython) D:\本科\信息集成与管理\实习\mwz\dace>curl http://localhost:5500/deleteData -H "Content-Type: application/json" -X DELETE -d "{\"id\":\"
2020302131002\"}"
[{"birthday":"2022-12-19","email":"123456@qq.com","id":"2021542666456","interest":"cow","major":"\u5730\u7406\u56fd\u60c5\u76d1\u6d4b\u4e13\u4e1a"
,"name":"\u8096\u6587\u8431","phonenumber":"13945256987"},{"birthday":"2002-07-09","email":"2574485753@qq.com","id":"2020302131249","interest":"\u
4e16\u754c\u676f","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b57\u6280\u672f\u4e13\u4e1a","name":"\u9a6c\u6587\u5353","phonenumber":"15272104
528"},{"birthday":"2003-11-20","email":"2364753@qq.com","id":"2020302131229","interest":"dance","major":"\u7a7a\u95f4\u4fe1\u606f\u4e0e\u6570\u5b5
7\u6280\u672f\u4e13\u4e1a","name":"\u5c0f\u9ec4","phonenumber":"11562104528"},{"birthday":"2022-06-11","email":"2454548089@qq.com","id":"202030213
1130","interest":"\u7761\u89c9","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u5730\u7406\u4fe1\u606f\u5de5\u7a0b\u65b9\u5411","
name":"\u9ec4\u6893\u6d9b","phonenumber":"18296873293"},{"birthday":"1987-06-28","email":"messi@gmail.com","id":"1987302145249","interest":"footba
ll","major":"\u9065\u611f\u79d1\u5b66\u4e0e\u6280\u672f\u4e13\u4e1a\u9065\u611f\u4fe1\u606f\u65b9\u5411","name":"messi","phonenumber":"12345678912
"}]
```

name	id	phonenumber	email	major	interest	birthday
肖文萱	2021542666456	13945256987	123456@qq.com	地理国情监测专业	cow	2022-12-19
马文卓	2020302131249	15272104528	2574485753@qq.com	空间信息与数字技术专业	世界杯	2002-07-09
小黄	2020302131229	11562104528	2364753@qq.com	空间信息与数字技术专业	dance	2003-11-20
黄梓涛	2020302131130	18296873293	2454548089@qq.com	遥感科学与技术专业地理信息工程方向	睡觉	2022-06-11
messi	1987302145249	12345678912	messi@gmail.com	遥感科学与技术专业遥感信息方向	football	1987-06-28

Figure 56:curl 测试 delete 接口

3. 资源模板

本项目还定义了资源模板，也可以修改资源模板中的数据，然后通过 curl 上传数据，从而进行添加数据。具体命令为：

curl -H "Content-Type: application/json" -X POST -d @client\template.json http://localhost:5500/submitData

Figure 57:资源模板

Figure 58:上传资源模板

六．实验心得

通过本次实习我对于 rest 风格下的 http 请求有了更加全面深入的认识，对于 GET、POST、PUT、DELETE 几种常用的请求类型理解地更加透彻。得益于本次实习，体验了一把全栈程序员的快乐，自己搭建前端和后端，自己进行接口测试，都是一种难得的体验，值得一提的是，在编写这个项目的时候真正体会到了一些简单的工作实际上要花费大量的时间。例如网页的样式设计，当时花了一整天就想要把它设计的美观又简洁；包括一些简单的错误验证，例如命令行 App 中如何保证只有输入正确时才做出相应的操作就是一门繁杂的学问。

总而言之，本次实习让我收获颇丰，同时也感谢老师的悉心指导。