



## 《信息系统集成与管理》 实验二 dace 描述文档

学    院：    遥感信息工程学院

班    级：    20F10

学    号：    2020302131201

姓    名：    常耀文

指导教师：    王华敏

2022 年 12 月 15 日

## 目录

1. 实习概述.....	3
1.1 实习目的.....	3
1.2 实习任务.....	3
1.3 实习所用软件梗概.....	4
1.3.1 Visual Studio Code.....	4
1.3.2 Pycharm.....	4
2. 实习环境概述.....	5
2.1 编程环境.....	5
2.2 软件环境.....	5
2.3 设备环境.....	5
3. 实验设计思路与细节设计.....	6
3.1 总体设计思路.....	6
3.2 实验细节设计.....	7
4. 实习内容.....	7
4.1 实习软件安装与环境配置.....	7
4.1.1 安装 Vscode.....	7
4.1.2 安装 Pycharm.....	7
4.2 编写网页前端，实现对人员信息的基本采集.....	7
4.2.1 编写前端提交页面.....	7
4.2.2 对提交信息进行验证.....	8
4.2.3 修改信息提交页面的显示方式.....	10
4.2.4 全部界面的设计.....	10
4.2 编写服务端代码，响应前端的请求.....	12
4.3 编写 API.....	15
4.3.1 提供对所有条目的列表获取的 API 服务.....	15
4.3.2 修改资源模板.....	15
4.3.3 提供新增的 API 服务.....	16
4.4 API 后端逻辑实现.....	21
4.4APP 实现.....	24
4.4.1 APP 设计思路.....	24
4.4.2 help 实现.....	25
4.4.3 add 实现.....	26
4.4.4 Delete 实现.....	26
4.4.5 Update 实现.....	26
4.4.6 list 实现.....	27
4.4.7 APP 核心代码实现思路.....	28
5. 实习问题与解决方式.....	30
5.1 Java Script 代码报错.....	30
5.2 修改页面显示时出错.....	31
5.3 Curl post 时输入中文会出现乱码.....	31
5.4 符合题目条件编写代码.....	31
6. 特别说明.....	31

7.	实习心得.....	32
----	-----------	----

## 1. 实习概述

### 1.1 实习目的

本次实习主要考察对于 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、包含各种 API 调用操作。通过该实践操作过程,学生可以了解 REST 架构风格的特点、学习如何集合类资源的 Web API 设计方式、通过 JSON 为格式的表述方法,加深对 Web Service 理论的理解,最后达到相应的实习目的与实习效果。

### 1.2 实习任务

本次实习主要针对业务需求,设计资源模版,模板的设计参考理论课所讲的 collection+json 样式。其次,安装业务功能,设计网页端需要的各种 REST 服务,以及要求的 API 服务,最后,根据实验要求,实现前后端代码,并进行测试。具体技术准备流程如下:

- 操作系统。学生根据自己的情况,在 Windows 或 Linux 中完成实验
- 服务端编程语言。尽管实验为大家准备了 node.js 基础代码,但学生可以根据自己偏好,选择不同的语言完成实验
- API 测试环境。在 curl 或者 postman 两种中选择一个。Windows 环境中需要自行安装 curl 程序包,postman 从官网下载试用版(Windows 或者 Linux 均可)
- 客户端 App 编程语言。客户端对自定义的 API 编程,以完成本实验指定的业务目标。  
客户端编程语言可以从以下的语言中任选一个:
  - Python
  - Node.js
  - C++
  - Java

## 1.3 实习所用软件梗概

### 1.3.1 Visual Studio Code

Visual Studio Code（简称“VS Code”）是 Microsoft 在 2015 年 4 月 30 日 Build 开发者大会上正式宣布一个运行于 Mac OS X、Windows 和 Linux 之上的，针对于编写现代 Web 和云应用的跨平台源代码编辑器，[2] 可在桌面上运行，并且可用于 Windows，macOS 和 Linux。它具有对 **JavaScript, TypeScript 和 Node.js** 的内置支持，并具有丰富的其他语言（例如 C++，C#，Java，Python，PHP，Go）和运行时（例如 .NET 和 Unity）扩展的生态系统。



Fig1.Vs code

### 1.3.2 Pycharm

PyCharm 是一种 Python IDE（Integrated Development Environment，集成开发环境），带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、项目管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，以用于支持 Django 框架下的专业 Web 开发。



Fig2.Pycharm

## 2. 实习环境概述

实验环境是实验操作的必不可少的步骤，本部分本节将从编程环境、软件环境，电脑硬件设别三方面对实验环境进行讨。

### 2.1 编程环境

本次实习主要采用了朱老师提供的示例 Node.js 搭建后端，使用了”http”,”crypto”,”fs”等多个模块，编写了”Templates”,”Messages”模块。使用 python 实现了命令行 app，调用了”request”,”time”等库。

编程环境	调用模块
Node.JS	http
	Crypto
	Fs
Python	Request
	Time

Table1.编程环境对应表

### 2.2 软件环境

本次实验采用个人计算机实现，所选取的软件也是从官网下载，具体的软件与软件版本之间的关系如下表所示：

软件名称	安装版本
VS CODE	1.67.2
PYCHARM	2021.3.2

Table2.软件版本表

### 2.3 设备环境

本次使用电脑设备与系统环境如下图所示：

项目	值
操作系统名称	Microsoft Windows 11 家庭中文版
版本	10.0.22521 版本 22521
其他操作系统描述	没有资料
操作系统制造商	Microsoft Corporation
系统名称	LAPTOP-QRU07UUI
系统制造商	LENOVO
系统型号	82AV
系统类型	基于 x64 的电脑
系统 SKU	LENOVO_MT_82AV_BU_idea_FM_Legion Y7000 2020
处理器	Intel(R) Core(TM) i5-10200H CPU @ 2.50GHz, 2496 Mhz, 4 个内核, ...
BIOS 版本/日期	LENOVO EFCN31WW, 2020/4/27
SMBIOS 版本	3.2
嵌入式控制器版本	1.31
BIOS 模式	UEFI
主板制造商	LENOVO
主板产品	LVNB161216
主板版本	SDK0L77769 WIN
平台角色	移动
安全启动状态	启用
PCR7 配置	需要提升才能查看
Windows 目录	C:\WINDOWS
系统目录	C:\WINDOWS\system32
启动设备	\Device\HarddiskVolume1
区域设置	中国
硬件抽象层	版本 = "10.0.22521.819"
用户名	LAPTOP-QRU07UUI\lenovo
时区	中国标准时间
已安装的物理内存(RAM)	16.0 GB
总的物理内存	15.9 GB
可用物理内存	7.86 GB
总的虚拟内存	34.8 GB

### 3. 实验设计思路与细节设计

本部分主要对于本次实验进行思路与总体架构进行设计，并且提出总体的技术目标与实现流程。

#### 3.1 总体设计思路

本次实习的任务均需要采用朱老师提供的 NodeJs 源代码完成。通过对于任务的分析，构建起任务的总体思路：

1. 首先搭建 Node. Js 后端服务器，通过使用 Node. Js 服务搭建起网页服务端，确保可以正常执行页面
2. 在已有代码的基础上进行加工修改，对于页面进行设计，并且对于输入数据的信息进行核查，确保电话，学号等信息的位数正确。
3. 设计 API，利用 API 进行测试，利用 API 实现对于数据条目的增删改查
4. 使用 Python 设计客户端 APP 编程，利用命令行执行 Python 程序，然后利用 Python 调用设计的 API，实现对于数据的操作
5. 测试程序的可行性

本次实习严格遵循 REST 的架构风格，设计的流程图如下所示：



Fig4.实验设计流程图

## 3.2 实验细节设计

本次实验设计的细节主要集中在 API 的设计，在 API 设计的时候要注意 REST 架构需要的幂等性，在设计时要注意有关细节的程序设计，在删除的时候，也有保证同一条目不能够被重复删除。

## 4. 实习内容

### 4.1 实习软件安装与环境配置

在开始实习前需要进行实习软件的下载与配置，本次实习主要安装两个软件，Vscode 与 Pycharm。

#### 4.1.1 安装 Vscode

通过 Vscode 的官方网站 (<https://code.visualstudio.com/>) 下载自定义版本的 Vscode，然后根据安装指导，一步一步进行安装即可。

#### 4.1.2 安装 Pycharm

安装 python 编辑器时能够通过 Pycharm 在 windows 端的官方网站 (<https://www.jetbrains.com/pycharm/download/#section=windows>) 下载自定义版本的 Pycharm，然后根据安装指导，一步一步进行安装即可。

### 4.2 编写网页前端，实现对人员信息的基本采集

首先从 FTP 服务器上下载朱老师的示例代码作为空项目在 VScode 中导入，并清空 data 目录内已有数据。

#### 4.2.1 编写前端提交页面

我们需要实现采集人员基本信息，包括：姓名、学号、邮箱、手机号码、个人兴趣，共 5 项内容，我使用 form 表单，内含 5 个 input 标签提交信息，5 个标签均不可为空。具体实现流程如下：

1. 打开 templates 文件夹中的 list.html，修改 body 中的信息以满足要求。
2. 首先将 h2 标签和其后 p 标签中的信息改为实验课指导中的内容，然后对 form 表单进行修改
3. 对最外边 form 标签绑定提交前检查函数，对每行使用 div 标签，其内部加上两个 label

标签和一个 input 标签，并设定相应的内容，具体情况如图中所示。

```
<form action="{@host}/messages" method="post" onsubmit="return check(this)">
  <div>
    <label>&nbsp;姓名&nbsp;</label>
    <input type="text" name="name" value="" required="true" maxlength="8"/>
    <label style="color: red"></label>
  </div>
  <div>
    <label>&nbsp;学号&nbsp;</label>
    <input type="text" name="studentId" value="" required="true"/>
    <label style="color: red"></label>
  </div>
  <div>
    <label>&nbsp;邮箱&nbsp;</label>
    <input type="email" name="email" value="" required="true"/>
    <label style="color: red"></label>
  </div>
  <div>
    <label>手机号码</label>
    <input type="tel" name="tel" value="" required="true"/>
    <label style="color: red"></label>
  </div>
  <div>
    <label>个人兴趣</label>
    <input type="text" name="interest" value="" required="true" maxlength="32"/>
    <label style="color: red"></label>
  </div>
  <input type="submit" />
</form>
```

Fig5.form 表单设计



Fig6.form 表单与初始界面展示

## 4.2.2 对提交信息进行验证

要求对姓名、个人兴趣采取字符长度限制（姓名不超过 8 个中文字符，个人兴趣不超过 32 个中文字符），因此在姓名和兴趣的 input 标签中添加 maxlength 限制。并且对学号、手机号、邮箱采取正则匹配，templates 目录下的 script.js，定义之前绑定在 form 表单上的 js 函数，在函数中对 form 表单中的各项进行验证。其中对学号、邮箱、手机号码采取正则匹配验证，姓名要求小于八位，学号要求为 13 位数字，邮箱要求合理格式，手机号码要求为以 1 开头，第二位不为 1, 2 的数字，且为 11 位。具体的 js 代码如下图所示：



```
// uses bootstrap for styling
window.onload = function() {
    var elm, coll, i, x;

    // style the body
    elm = document.getElementsByTagName('div')[0];
    if(elm) {elm.className = 'hero-unit';}

    // style the nav links
    coll = document.getElementsByTagName('a');
    for(i=0, x=coll.length; i<x; i++) {
        if(coll[i].parentNode.className==='links') {
            coll[i].className = 'btn btn-primary btn-large';
        }
    }

    // style the message details
    elm = document.getElementsByTagName('dl')[0];
    if(elm) {elm.className='dl-horizontal';}

    // style the input form
    elm = document.getElementsByTagName('form')[0];
    if(elm) {
        elm.className='form-inline';
        // 正则匹配
    }

    coll = document.getElementsByTagName('input');
    for(i=0, x=coll.length; i<x; i++) {
        if(coll[i].getAttribute('type')==='submit') {
            coll[i].className='btn';
        }
    }
}

function check(form){
    let name_valid = form.name.value.length <= 8;
    let interest_valid = form.interest.value.length <= 32;
    let studentId_valid = /^d{13}$/.test(form.studentId.value);
    let email_valid = /^[w@w+(\.w+)+$]/.test(form.email.value);
    let tel_valid = /^[13-9][0-9]\d{8}$/.test(form.tel.value);
    if (name_valid && interest_valid && studentId_valid && email_valid && tel_valid)
        return true;
    else
    {
        if (!name_valid)
            form.name.nextElementSibling.innerText = '名字不得长于8字符';
        else
            form.name.nextElementSibling.innerText = '';
        if (!studentId_valid)
            form.studentId.nextElementSibling.innerText = '学号需为13位数字';
        else
            form.studentId.nextElementSibling.innerText = '';
        if (!email_valid)
            form.email.nextElementSibling.innerText = '邮箱不合标准';
        else
            form.email.nextElementSibling.innerText = '';
        if (!tel_valid)
            form.tel.nextElementSibling.innerText = '电话不合标准';
        else
            form.tel.nextElementSibling.innerText = '';
        if (!interest_valid)
            form.interest.nextElementSibling.innerText = '个人兴趣不得长于32字符';
        else
            form.interest.nextElementSibling.innerText = '';
        return false;
    }
}
```

Fig7.信息验证 JavaScript 代码设计

在客户端对于信息验证功能进行验证操作如下，如下首先输入 23 位为一的手机号码，会以红字的形式提示错误，14 位学号，也会提示学号需要 13 位数字，提示错误。正确输入后不会出现红字部分，且跳转到 Message 界面。展示结果如下图所示：

Fig8.信息验证结果展示

当邮箱没有@时，也会提示错误，当姓名长度超过 8 位后，无法再次进行输入，而是会将 8 位之后的字符自动省略掉，以下将展示这两种错误情况：

Fig9.错误类型结果展示

在 data 目录下发现，本条数据已经成功录入，下图为本次创建的用户信息数据，通过时间比对与信息条目研究，发现可以和其功能相对应。

**《信息系统集成与管理》**  
**实验二 dace 描述文档**  
**2020302131201-常耀文**

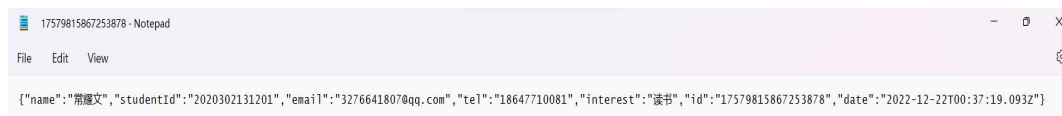


Fig10.成功录入结果展示

### 4.2.3 修改信息提交页面的显示方式

原本的信息提交界面提交成功后，会在页面下方出现链接，通过链接可以跳转到相应的个人信息界面，为了信息的方便显示，将在信息展示界面展示全部的信息，打开 `app.js`，并跳转到函数 `formatHtmlList(list)`，将原有的显示 `message` 字段改为显示为提交的个人信息字段，具体信息展示如下图所示：

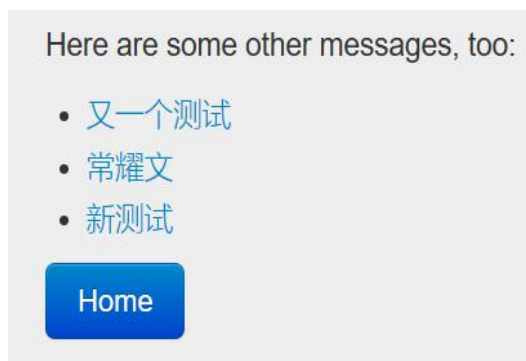


Fig11.修改前显示效果



Fig12.修改后显示效果

最终个人信息采集表的界面的显示效果如下图所示：



Fig13.个人信息采集表界面

### 4.2.4 全部界面的设计

由于界面并非本次实习的重点，因此通过修改朱老师所给的代码，构建了本次个人信息提交的全部界面：

1. 首先是 `home` 界面，`home` 界面主要是对于网站的基本介绍，通过两个超链接按钮可以实现页面之间的跳转



Fig14.home 界面

2. 其次是个人信息收集界面，由于本界面在之前已经有了较多的叙述，因此在这里不过多叙述

Fig15.个人信息收集界面

3. 接下来展示的是本次的 About 界面,通过点击 home 界面的关于本网站可以跳转到 About 界面，下图为 About 界面：

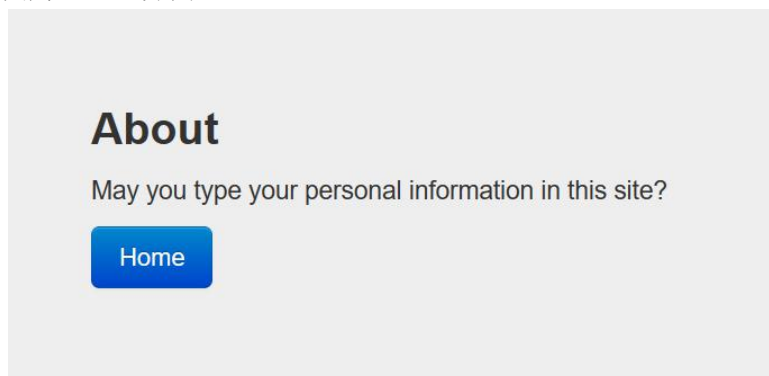


Fig16.About 界面

4. 通过点击个人信息收集界面的收集到的个人信息，可以跳转到个人详细信息收集界面，具体效果如下图所示：



Fig17.个人信息展示界面

## 4.2 编写服务端代码，响应前端的请求

由于本次服务端代码采用 Nodejs 实现，因此在编写服务端代码之前，需要对于 Nodejs 此次工作的逻辑进行了解，Nodejs 后端处理网页前端消息的流程如下图：

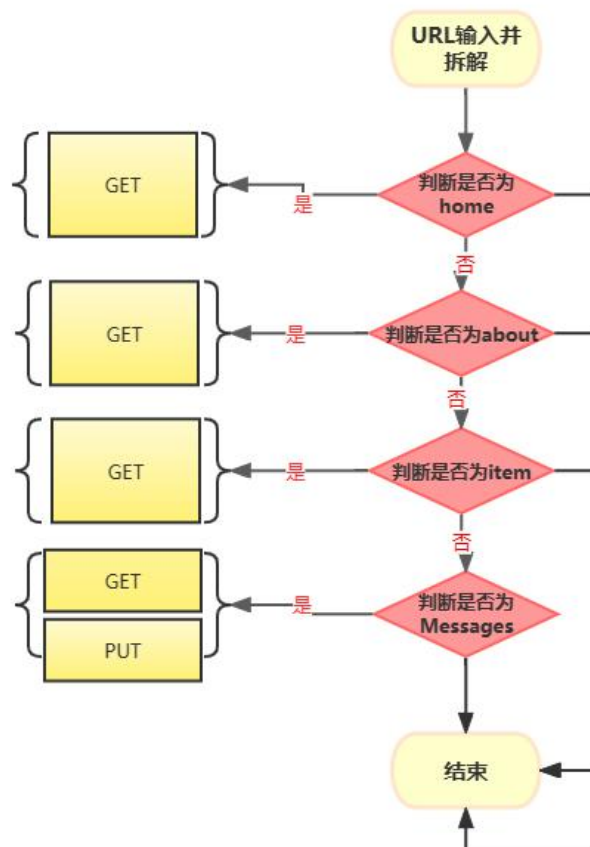


Fig18.页面前端消息处理流程逻辑图

首先使用 http 模块在本机 1337 端口创建服务，用 handler()函数处理请求和响应，如下图所示：

```
// register listener for requests
http.createServer(handler).listen(port);
```

Fig19.http 创建服务

handler()函数在收到请求后，通过对 url 进行拆解分析，使用正则匹配的方法，分析该请求的来源是网页前端还是 API，本节中主要描述网页前端。如下图所示：

```
function handler(req, res) {
    var segments, i, x, parts, flg;

    // set root
    root = 'http://' + req.headers.host;

    // parse incoming request URL
    parts = [];
    segments = req.url.split('/');
    for(i=0, x=segments.length; i<x; i++) {
        if(segments[i]!='') {
            parts.push(segments[i]);
        }
    }
}
```

Fig20.url 获取并拆解

```
// handle routing
flg=false;

// home
if(reHome.test(req.url)) {
    flg=true;
    if(req.method==='GET') {
        sendHtmlHome(req, res);
    }
    else {
```

Fig21.正则匹配确定来源

通过对 url 的正则匹配可以分析出该请求的网页来源，然后分析其请求方法为”POST”还是”GET”。对于不同页面，”GET”方法获取的内容是不同的。在 home 页面与 about 页面，”GET”方法实现的功能是近似一致的（此处以 about 页面为例），都是将网页模板发送至前端显示，情况如下图所示：

```
// about
if(flg===false && reAbout.test(req.url)) {
    flg=true;
    if(req.method==='GET') {
        sendHtmlAbout(req, res);
    }
    else {
        sendHtmlError(req, res, 'Method Not Allowed', 405);
    }
}
```

Fig22. about 页面的 GET 形式

```
function sendHtmlAbout(req, res) {
    var t;

    try {
        t = templates('about.html');
        t = t.replace(/{@host}/g, root);
        sendHtmlResponse(req, res, t, 200);
    }
    catch (ex) {
        sendHtmlError(req, res, 'Server Error', 500);
    }
}
```

Fig23. about 发送页面模板函数

但是在提交提交个人信息的 messages 网页，”GET”方法不仅可以用于发送页面模板，还可以传递本地存储的数据到 list 页面进行显示

```
// list
if(flg===false && reList.test(req.url)) {
    flg=true;
    switch(req.method) {
        case 'GET':
            sendHtmlList(req, res);
            break;
        case 'POST':
            postHtmlItem(req, res);
            break;
        default:
            sendHtmlError(req, res, 'Method Not Allowed', 405);
            break;
    }
}
```

```
function sendHtmlList(req, res) {
    var t, rtn, list, lmDate;

    try {
        rtn = messages('list');
        list = rtn.list;
        lmDate = rtn.lastDate;
        t = templates('list.html');
        t = t.replace(/{@host}/g, root);
        t = t.replace(/{@messages}/g, formatHtmlList(list));
        sendHtmlResponse(req, res, t, 200, new Date(lmDate).toGMTString());
    }
    catch (ex) {
        sendHtmlError(req, res, 'Server Error', 500);
    }
}
```



Fig24. list 页面的 GET 形式

Fig25. list 发送页面模板函数

在 sendHtmlList 函数中出现了 messages 模块，这是自定义的模块，用于与本地文件系统交互，此处通过 messages('list') 获取本地文件系统储存的所有数据与最新一条数据的时间，其具体实现过程如下。

```
function getList(arg) {
    var coll, item, list, i, x, lastDate;

    lastDate = null;
    coll = [];
    list = fs.readdirSync(folder);
    for(i=0,x=list.length;i<x;i++) {
        item = JSON.parse(fs.readFileSync(folder+list[i]));
        if(arg) {
            if(item.title.indexOf(arg)!==-1) {
                if(lastDate===null || lastDate<new Date(item.date)) {
                    lastDate = new Date(item.date);
                }
                coll.push(item);
            }
        }
        else {
            if(lastDate===null || lastDate<new Date(item.date)) {
                lastDate = new Date(item.date);
            }
            coll.push(item);
        }
    }
    return {list:coll, lastDate:lastDate};
}
```

Fig26.Message Js 获取数据条目

在提交个人信息的 list 页面，有一个向后端提交一条数据的”POST”方法，当写完所有的数据并且点击提交后，通过调用 postHtmlItem() 函数来实现，后端在向本地文件系统写入一条数据后，向前端发送一个状态码为”303”的回复，指导前端进行下一步操作——再向后端发送”GET”请求。

```
function postHtmlItem(req, res) {
    var body, item, rtn, lmDate;

    body = '';
    req.on('data', function(chunk) {
        body += chunk.toString();
    });

    req.on('end', function() {
        try {
            item = messages('add', querystring.parse(body)).item;
            res.writeHead(303, 'See Other', {'Location': root+'/messages/'+item.id});
            res.end();
        }
        catch (ex) {
            sendHtmlError(req, res, 'Server Error', 500);
        }
    });
}
```

=

Fig27. postHtmlItem()

当 postHtmlItem() 函数在收到客户端发送的数据后，先对数据进行解析和反序列化，后

调用 messages 模块在本地文件系统中写入数据。在写入时同时赋予 id，记录时间。当进入到 item 界面时，其”GET”方法流程与 messages 页面 GET 方法思路大体一致，区别在于：后者在后端本地文件系统中获取了所有的数据，前者是根据数据 id 获取了具体的一条数据。而非获取后端种存储的全部数据。

## 4.3 编写 API

### 4.3.1 提供对所有条目的列表获取的 API 服务

打开 app.js，转到 formatAPIList(list)函数，将输出信息由 message 改为学生相关信息，具体如下图所示。

```
function formatAPIList(list) {  
    var i, x, rtn, item;  
  
    rtn = [];  
    for(i=0,x=list.length; i<x; i++) {  
        item = {};  
        item.href = root + '/api/' + list[i].id;  
        item.data = [];  
        item.data.push({name:"name", value:list[i].name});  
        item.data.push({name:"studentId", value:list[i].studentId});  
        item.data.push({name:"email", value:list[i].email});  
        item.data.push({name:"tel", value:list[i].tel});  
        item.data.push({name:"interest", value:list[i].interest});  
        item.data.push({name:"date_posted", value:list[i].date});  
        rtn.push(item);  
    }  
    return JSON.stringify(rtn, null, 4);  
}
```

Fig28. 修改 message 信息成为学生相关信息

### 4.3.2 修改资源模板

打开 templates 目录中的 collection.js，将 template->data 中的 text 信息改为对应的学生相关信息，具体如下图所示。

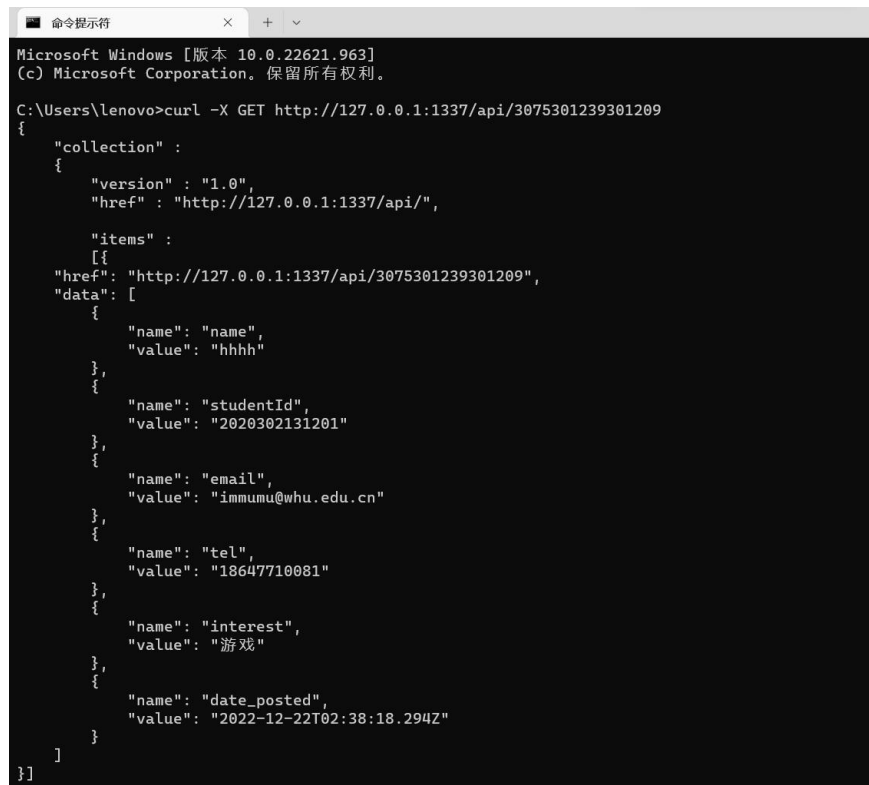
```
{  
  "collection" :  
  {  
    "version" : "1.0",  
    "href" : "{@host}/api/",  
    "items" :  
    {@list}  
    ,  
    "template" : {  
      "data" : {  
        {  
          "prompt" : "name of student", "name" : "name", "value" : ""  
        }  
        {  
          "prompt" : "studentId of student", "name" : "studentId", "value" : ""  
        }  
        {  
          "prompt" : "email of student", "name" : "email", "value" : ""  
        }  
        {  
          "prompt" : "telephone of student", "name" : "tel", "value" : ""  
        }  
        {  
          "prompt" : "personal interest of student", "name" : "interest", "value" : ""  
        }  
      }  
    }  
  }  
}
```

Fig29. 修改 collection.js

然后将 test 目录下的 zhu1.js 文件定义为资源模板，成为自定义的资源模板

以数据 3075301239301209 为例，使用命令

curl -X GET http://127.0.0.1:1337/api/3075301239301209，传回数据如下图。



```
Microsoft Windows [版本 10.0.22621.963]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lenovo>curl -X GET http://127.0.0.1:1337/api/3075301239301209
{
  "collection" :
  {
    "version" : "1.0",
    "href" : "http://127.0.0.1:1337/api/",
    "items" :
    [
      {
        "href": "http://127.0.0.1:1337/api/3075301239301209",
        "data": [
          {
            "name": "name",
            "value": "hhhh"
          },
          {
            "name": "studentId",
            "value": "2020302131201"
          },
          {
            "name": "email",
            "value": "immumu@whu.edu.cn"
          },
          {
            "name": "tel",
            "value": "18647710081"
          },
          {
            "name": "interest",
            "value": "游戏"
          },
          {
            "name": "date_posted",
            "value": "2022-12-22T02:38:18.294Z"
          }
        ]
      }
    ]
  }
}
```

Fig30. 使用 API 测试获取数据

### 4.3.3 提供新增的 API 服务

由于实习任务的要求，服务端需要能够通过 API 的方式提供相应的服务，API 主要实现对于条目的增删改查。

#### 4.3.3.1 对于条目进行新增

使用 Curl 命令进行新增：

```
curl -X POST http://127.0.0.1:1337/api/ -H "Content-type:application/json" -d
{"template":{"data":[{"name":"name","value":"changyaowen"}, {"name":"studentId","value":"2020302131201"}, {"name":"email","value":"2020302131201@whu.edu.cn"}, {"name":"tel","value":"18647710081"}, {"name":"interest","value":"play game"}]}}
```

或者通过自定义的资源模板上传信息，打开 test/zhu1.js，修改内容如下：



```
{
  "template" : {
    "data" : [
      { "name" : "name", "value" : "常耀文"},
      { "name" : "studentId", "value" : "2020302131201"},
      { "name" : "email", "value" : "2020302130023@whu.edu.cn"},
      { "name" : "tel", "value" : "18647710081"},
      { "name" : "interest", "value" : "看书"}
    ]
  }
}
```

FIg 31 zhu1. js 文件内容

通过运行，发现上传成功：

```
8081486890204845 X
data > 8081486890204845
1 {"name":"changyaowen","studentId":"2020302131201","email":"2020302131201@whu.edu.cn","tel":"18647710081","interest":"play game","id":"8081486890204
```

Fig32. 通过 curl 命令不利用模板 API 增加结果

对于操作结果进行分析，发现 POST 方法并不幂等，因此在多次上传相同数据时，由于时间不同，都会被存储。

```
server > data > 17472033787110441 > ...
1 [{"name":"常耀文","studentId":"2020302131201","email":"3276641807@qq.com","tel":"18647710081","interest":"看书","id":"17472033787110441",
2 "date":"2022-12-25T05:55:31.546Z"}]

server > data > 11832442686561495 > ...
1 [{"name":"常耀文","studentId":"2020302131201","email":"3276641807@qq.com","tel":"18647710081","interest":"看书","id":"11832442686561495",
2 "date":"2022-12-22T02:51:44.507Z"}]
```

Fig33. POST 方法不具有幂等性

#### 4. 3. 3. 2 对于条目进行删除

对条目的删除，举例如下：

Curl 命令:

`curl -X DELETE http://127.0.0.1:1337/api/3075301239301209` 在删除前还存在着 3075301239301209 的数据,

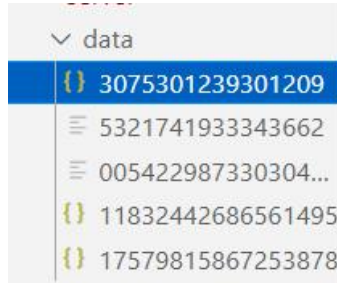


Fig34. Delete 方法操作前

在命令行执行命令:

```
C:\Users\lenovo>curl -X DELETE http://127.0.0.1:1337/api/3075301239301209
```

Fig35. 删除数据

执行命令后, 结果为:

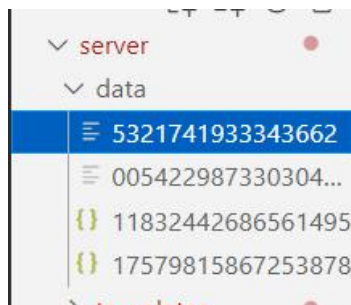


Fig36. 执行命令后数据条目消失

DELETE 方法具有幂等性, 因此多次执行此 Curl 命令, 对系统的影响是一致的。如图, 仅有第一次删除是成功的, 后续尝试删除以状态码 500 报错告终, 无法重复对于同一条数据进行删除操作。

```
C:\Users\lenovo>curl -X DELETE http://127.0.0.1:1337/api/3075301239301209
C:\Users\lenovo>curl -X DELETE http://127.0.0.1:1337/api/3075301239301209
{"collection":{"version":"1.0","href":"http://127.0.0.1:1337/api/","error":{"title":"Server Error","code":500}}}
C:\Users\lenovo>
```

Fig37. 删除具有幂等性

#### 4.3.3.3 对于条目进行修改

对于条目进行修改, 同样可以使用 Curl 命令或者利用资源模板进行更正:

✓ CURL 命令: `curl -X PUT http://127.0.0.1:1337/api/11832442686561495 -H`

```
"Content-type:application/json" -d  
  
{"template":{"data":[{"name":"name","value":"testPut"}, {"name":"studentId","value":"2019302131201"}, {"name":"email","value":"4555666@qq.com"}, {"name":"tel","value":"14567996654"}, {"name":"interest","value":"sleep"}]}}
```

✓ 资源模板:

```
Curl -X PUT http://127.0.0.1:1337/api/11832442686561495 -H  
"Content-type:application/json" -d @F:\dace\test\zhu1.js
```

修改前数据如图所示:

```
server > data > 11832442686561495 > ...  
1 {"name":"常耀文","studentId":"2020302131201","email":"3276641807@qq.com","tel":"18647710081","interest":"看书","id":"11832442686561495",  
2 "date":"2022-12-22T02:51:44.507Z"}
```

Fig38. 修改前数据

命令行窗口执行 put 的 curl 命令, 返回修改后的数据内容。

```
.\Users\lenovo>curl -X DELETE http://127.0.0.1:1337/api/3075301239301209  
{"collection":{"version":"1.0","href":"http://127.0.0.1:1337/api/","error":{"title":"Server Error","code":500}}}  
.\Users\lenovo>curl -X PUT http://127.0.0.1:1337/api/11832442686561495 -H "Content-type:application/json" -d {"template":{"data":[{"name":"name",  
value":"testPut"}, {"name":"studentId","value":"2019302131201"}, {"name":"email","value":"4555666@qq.com"}, {"name":"tel","value":"14567996654"}, {"name":"interest","value":"sleep"}]}}}  
  
{"collection":  
  {  
    "version": "1.0",  
    "href": "http://127.0.0.1:1337/api/",  
    "items":  
      [{  
        "href": "http://127.0.0.1:1337/api/11832442686561495",  
        "data": [  
          {  
            "name": "name",  
            "value": "testPut"  
          },  
          {  
            "name": "studentId",  
            "value": "2019302131201"  
          },  
          {  
            "name": "email",  
            "value": "4555666@qq.com"  
          },  
          {  
            "name": "tel",  
            "value": "14567996654"  
          },  
          {  
            "name": "interest",  
            "value": "sleep"  
          },  
          {  
            "name": "date_posted",  
            "value": "2022-12-22T02:51:44.507Z"  
          }  
        ]  
      }  
    ]  
  }  
}
```

Fig39. 修改后数据改变

**《信息系统集成与管理》**  
**实验二 dace 描述文档**  
**2020302131201-常耀文**

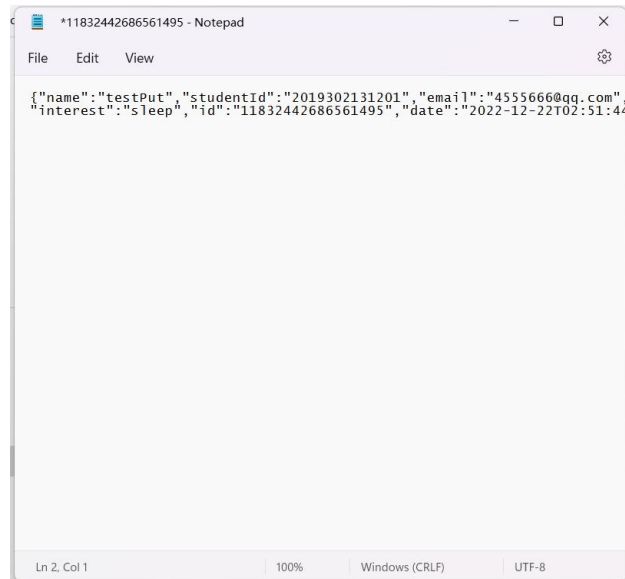


Fig40. 修改后本地数据改变

验证 PUT 方法的幂等性：多次执行上述 curl 命令，命令行窗口每次返回的数据相同，在服务端查看，数据未发生改变。



Fig41. 多次修改后没有发生改变

#### 4.3.3.3 对条目列表进行访问

Curl 命令为：

`curl -X GET http://127.0.0.1:1337/api/`

执行命令后，会按照预先定义的模板形式返回服务器所有的数据文件：

```
C:\Users\lenovo>curl -X GET http://127.0.0.1:1337/api/
{"collection": {"version": "1.0", "href": "http://127.0.0.1:1337/api/", "items": [{"href": "http://127.0.0.1:1337/api/11832442686561495", "data": [{"name": "name", "value": "testPut"}, {"name": "studentId", "value": "2019302131201"}, {"name": "email", "value": "4555666@qq.com"}, {"name": "tel", "value": "14567996654"}, {"name": "interest", "value": "sleep"}, {"name": "date_posted", "value": "2022-12-22T02:51:44.587Z"}]}, {"href": "http://127.0.0.1:1337/api/17579815867253878", "data": [{"name": "name", "value": "常耀文"}, {"name": "studentId", "value": "2020302131201"}, {"name": "email", "value": "3276641807@qq.com"}, {"name": "tel", "value": "18647710881"}, {"name": "interest", "value": "读书"}, {"name": "date_posted", "value": "2022-12-22T08:37:19.893Z"}]}]}}
```

Fig42. 获取全部条目

由于仅保存了两条数据，在这里全部显示。并且验证 GET 方法的幂等性：多次执行上述 curl 命令，命令行窗口每次返回的内容相同，并未对服务端的数据本身产生影响。

## 4.4 API 后端逻辑实现

在开发相应的 API 时，需要注意到 API 后端逻辑的实现，首先列上关于 API 的处理流程框图：

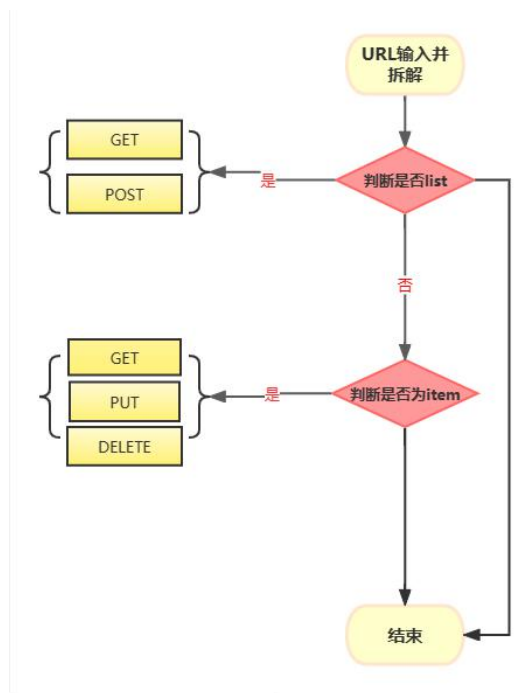


Fig43. 对 API 的处理逻辑

通过对 url 进行拆解分析，使用正则匹配的方法，分析该请求的来源是 API List 还是 API Item，然后判断请求方法为 POST、GET、PUT、DELETE 哪一种，进行进一步的处理操作。其中，POST、GET 处理流程与网页前端+Nodejs 后端的 POST、GET 处理思路基本一致，故本节主要描述 DELETE 和 PUT 的过程。

### ➤ Delete

服务端收到 API 发送的 DELETE 请求后，将被删除文件的 id 等参数传入函数 removeAPIItem()中，removeAPIItem()函数调用 messages 模块，对本地文件系统中对应 id 的数据文件进行删除，在 messages 模块中，通过调用 fs 模块的 unlinkSync()函数对对应 id 的文件进行删除。

<pre>// API Item if(flag===false &amp;&amp; reAPIItem.test(req.url)) {   flag=true;   switch(req.method) {     case 'GET':       sendAPIItem(req, res, parts[1]);       break;     case 'PUT':       updateAPIItem(req, res, parts[1]);       break;     case 'DELETE':       removeAPIItem(req, res, parts[1]);       break;     default:       sendAPIError(req, res, 'Method Not Allowed', 405);       break;   } }</pre>	<pre>function removeAPIItem(req, res, id) {   var t;    try {     messages('remove', id);     t = templates('collection.js');     t = t.replace(/@host/g, root);     t = t.replace(/@list/g, formatAPIList(messages('list')));     res.writeHead(204, 'No Content', cjHeaders);     res.end();   }   catch(ex) {     sendAPIError(req, res, 'Server Error', 500);   } }</pre>
<pre>function main(action, arg1, arg2) {   var rtn;    switch(action) {     case 'list':       rtn = getList();       break;     case 'item':       rtn = getItem(arg1);       break;     case 'add':       rtn = addItem(arg1);       break;     case 'update':       rtn = updateItem(arg1, arg2);       break;     case 'remove':       rtn = removeItem(arg1);       break;     default:       break;   }   return rtn; }</pre>	<pre>function removeItem(id) {   fs.unlinkSync(folder+id);   return getList(); }  function makeId() {   var tmp, rtn;    tmp = Math.random();   rtn = String(tmp);   rtn = rtn.substr(2);   return rtn; }  /* eof */</pre>

Fig44. 删除的逻辑实现

### ➤ Put

服务端收到 API 发送的 PUT 请求后，将被更新条目的参数到传入函数 updateAPIItem()中，在函数中服务端读取了修改数据的内容，将其作为 collection，随 id 一同传入 messages 模块中，messages 模块对本地文件系统修改后，将修改后数据传回 updateAPIItem()函数中，服务端再调用 sendAPIItem()函数将修改后数据发送至调用 API 端。在 messages 模块中，调用了 updateItem()函数对本地文件进行修改，即先根据 id 获取数据信息的 collection，而后在 collection 中对指定修改内容进行修改。下图为 PUT 的实现逻辑。



<pre>function updateItem(id, item) {     var current;     current = getItem(id).item;     current.name = item.name;     current.studentId = item.studentId;     current.email = item.email;     current.tel = item.tel;     current.interest = item.interest;     //current.date = new Date(); // put方法是幂等的     fs.writeFileSync(folder+id, JSON.stringify(current));     return getItem(id); }</pre>	<pre>function updateAPIItem(req, res, id) {     var body, item, msg;      body = '';     req.on('data', function(chunk) {         body += chunk;     });      req.on('end', function() {         try {             msg = JSON.parse(body);             let data = {}, datalist = msg.template.data;             for(i = 0; i &lt; datalist.length; ++i)                 data[datalist[i].name] = datalist[i].value;             item = messages('update', id, data).item;             sendAPIItem(req, res, id);         }         catch(ex) {             sendAPIError(req, res, 'Server Error', 500);         }     }); }</pre>
<pre>function main(action, arg1, arg2) {     var rtn;      switch(action) {         case 'list':             rtn = getList();             break;         case 'item':             rtn = getItem(arg1);             break;         case 'add':             rtn = addItem(arg1);             break;         case 'update':             rtn = updateItem(arg1, arg2); //arg1为文件id, arg2为新             break;         case 'remove':             rtn = removeItem(arg1);             break;         default:             break;     }      return rtn; }</pre>	<pre>// API Item if(flag===false &amp;&amp; reAPIItem.test(req.url)) {     flag=true;     switch(req.method) {         case 'GET':             sendAPIItem(req, res, parts[1]);             break;         case 'PUT':             updateAPIItem(req, res, parts[1]);             break;         case 'DELETE':             removeAPIItem(req, res, parts[1]);             break;         default:             sendAPIError(req, res, 'Method Not Allowed', 40             break;     } }</pre>

Fig45. PUT 的逻辑实现

但是在 PUT 方法的实现于逻辑研究时因为与同学讨论产生了一些疑惑，通过对于问题的研究得出了以下结论：

1. 首先之前对于 PUT 方法的描述是具有错误的，在下图蓝框中可以发现，原本代码为”current = getItem(id)”，但细心观察发现 getItem() 函数返回值为 collection 形式：{item:item, lastDate:lastDate}，结合函数下文，current 应该被赋值为函数返回值的 item 项，故此处修改为”current = getItem(id).item”，因此在这里需要对于代码进行修改。

```
//之前 current = getItem(id), 没有item
current = getItem(id).item;
current.name = item.name;
current.studentNumber = item.studentNumber;
current.emailAddress = item.emailAddress;
current.phoneNumber = item.phoneNumber;
current.interest = item.interest;
//current.date = new Date(); // put方法是幂等的
```

2. 在 `current.date = new Date()` 部分，这里的时间按照理解有两种解释，一种是最近修改的时间，如果是这样，不该被注释，而理解为上传的时间的话，那么由于 PUT 幂等性，不能修改时间，就不能持续更新 `date` 了，所以应该注释掉。

## 4. 4APP 实现

根据本次实习要求，需要自行设计一个客户端 APP，而且以命令行的方式加以实现，具体的实习要求如下所示：

1. 添加一条新的条目
2. 删除一条已存在的条目。条目不存在时，给予错误提示
3. 修改一条已存在的条目的内容，条目不存在时，给予错误提示
4. 对所有的条目列表输出，并可以指定按照条目中的时间，升序或降序列表

由于在之前已经有过利用 Python 进行命令行编程的经验，因此本次开发 APP 使用 Python 实现。

### 4. 4. 1APP 设计思路

通过命令行中输入指令，利用 APP 程序对指令进行字符串拆解、功能分析，然后使用 `requests` 库调用服务器端的不同 API，以实现增加，删除，修改，查找的功能。设计的流程框图如下图所示：

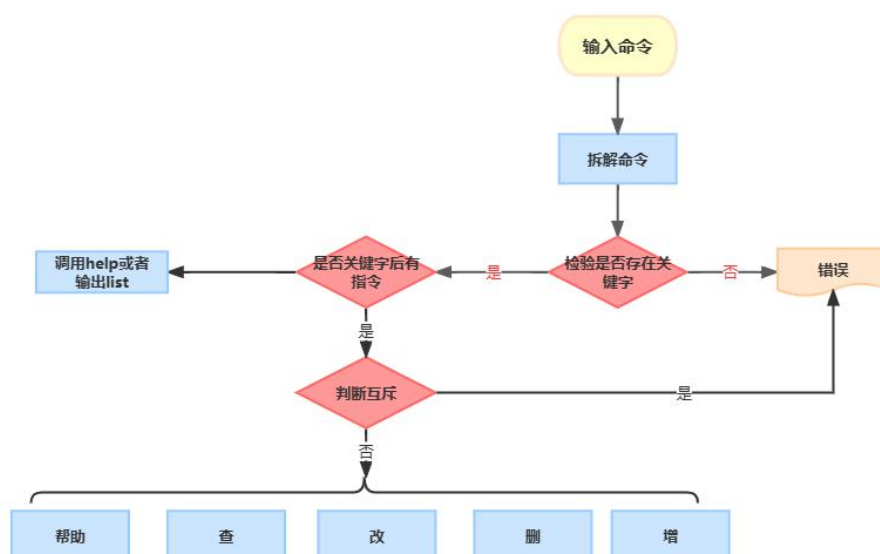


Fig46. APP 逻辑设计



本次 APP 实现的想法是通过在命令行中输入命令，然后利用函数解析命令，从而设计不同的函数，调用不同的 API 实现 APP 的功能。

## 4.4.2help 实现

首先书写帮助文档，当接受到 help 指令时，打印帮助文档的内容，如果有指令错误，程序也会提示输入 app -h/--help 寻求帮助。帮助文档如下图所示：

```
*****说明*****
1.所有的选项均为非强制选项（即：如果运行时无选项，则按照"-list=ascend"操作（因为这是一个安全操作）；如果有选项，则按照选项操作；
2.互斥选项：-a | -d | -u | -l ,即该4个选项只能同时选择一个；
3.联合选项：-u id {-n | -i | -m | -b}，即-u选项必须附带最少后面的四个选项之一才能工作，其中-u --update使用方式相同，意义等价

-h | --help: 输出本app的用法帮助

-a | --add: 增加一个新的条目，该选项需要配合以下4个选项（4个选项须同时使用）：
    -n | --name={增加的姓名}
    -i | --id={增加的学号}
    -m | --mobile={增加的手机号}
    -h | --hobby={增加的兴趣}
示例：app>>app -a -n Jack -i 202030213189 -t 18777954123 -b basketball

-d | --delete={id}：对编号为id的条目做删除操作，id必须输入
示例：app>>app -d 11832442686561495

-u | --update={id}：对编号为id的条目做修改操作，该操作需要配合以下4个选项（可独立，也可组合使用）：
    -n | --name={更新后的姓名}
    -i | --id={更新后的学号}
    -t | --telephone={更新后的手机号}
    -h | --h={更新后的兴趣}
示例：app>>app -u 11832442686561495 -n Jack -i 2020302111111 -m 15995159515 -b basketball

-l | --list={mode}：当mode是"descend"时【默认】，按照时间降序排列；当mode是"ascend"，升序排列，输出到终端显示
示例：app>>app -l ascend
```

Fig47. 帮助文档

运行以后的展示如下图：

```
PS H:\AIXiao> python app.py
请输入样式为 app -h/--help类型的指令:app -h
*****说明*****

1.所有的选项均为非强制选项（即：如果运行时无选项，则按照"-list=ascend"操作（因为这是一个安全操作）；如果有选项，则按照选项操作；
2.互斥选项：-a | -d | -u | -l ,即该4个选项只能同时选择一个；
3.联合选项：-u id {-n | -i | -m | -b}，即-u选项必须附带最少后面的四个选项之一才能工作，其中-u --update使用方式相同，意义等价

-h | --help: 输出本app的用法帮助

-a | --add: 增加一个新的条目，该选项需要配合以下4个选项（4个选项须同时使用）：

    -n | --name={增加的姓名}

    -i | --id={增加的学号}
```

Fig48. 运行帮助

#### 4.4.3add 实现

在实现添加一个条目时，必须要把所有的信息都添加到，即如此添加：

-a | --add: 增加一个新的条目，该选项需要配合以下 4 个选项（4 个选项须同时使用）：

-n | --name={增加的姓名}

-i | --id={增加的学号}

-m | --mobile={增加的手机号}

-b | --hobby={增加的兴趣}

示例：app>>app -a -n Jack -i 202030213189 -m 18777954123 -b basketball

因此在输入时，四个信息都不可缺少，必须全部输入才能够正确添加信息

下图为程序运行结果与过程：

```
PS H:\AIXiao> python app.py
请输入样式为 app -h/--help 类型的指令 :app -a -n Jack -i 202030213189 -t 18777954123 -o basketball
上传数据成功
```

Fig49. 依照案例运行 add

```
server > data > {} 4438242551083096 > ...
1  {"name":"Jack","studentId":"202030213189","email":"null","tel":"18777954123","interest":"basketball","id":"4438242551083096","date":"2022-12-23T13:0
```

Fig50. 检验数据确实上传成功

#### 4.4.4Delete 实现

实现删除操作时，必须具备的是有一个 id 编号，通过对于 id 编号的删除进而删除该条目数据，示例如下：

-d | --delete={id}：对编号为 id 的条目做删除操作，id 必须输入，示例：app>>app -d 11832442686561495

```
请输入样式为 app -h/--help 类型的指令 :app -a -n Jack -i 202030213189 -t 18777954123 -o basketball
上传数据成功
请输入样式为 app -h/--help 类型的指令 :app -d 11832442686561495
删除数据成功
```

Fig51. 依照案例运行 delete

```
server
  data
    {} 4438242551083096
    {} 17579815867253878
  templates
```

```
server > data > {} 11832442686561495 >
1  studentId:"2019302131201",
```

Fig52. 在 Vscode 中发现确实删除了数据

#### 4.4.5Update 实现

实现 update 时，需要注意可以对多条目操作，也可以对单条目操作，示例如下：-u | --update={id}：对编号为 id 的条目做修改操作，该操作需要配合以下 4 个选项（可独立，也

可组合使用):

```
-n | --name={更新后的姓名}  
-i | --id={更新后的学号}  
-m | --telephone={更新后的手机号}  
-b | --hobby={更新后的兴趣}
```

示例: `app>>app -u 17579815867253878 -n Jack -i 2020302111111 -m 15995159515 -b basketball`  
但是需要有具体的数据存在, 如果不存在会出现提示, 不存在此条数据:

```
请输入样式为 app -h/--help 类型的指令: app -u 11832442686561495 -n Jack -i 2020302111111 -m 15995159515 -o basketball  
不存在此条数据, 请检查输入 id
```

Fig53. 不存在数据

不存在数据会提示出现检查输入 ID 样式。修改成为已存在数据的 ID 后, 会出现提示更新成功, 如下图所示:

```
请输入样式为 app -h/--help 类型的指令: app -u 17579815867253878 -n Jack -i 2020302111111 -t 15995159515 -o basketball  
更新后数据为:  
{'name': 'name', 'value': 'Jack'}  
{'name': 'studentId', 'value': '2020302111111'}  
{'name': 'email', 'value': '3276641807@qq.com'}  
{'name': 'tel', 'value': '15995159515'}  
{'name': 'interest', 'value': 'basketball'}  
{'name': 'date_posted', 'value': '2022-12-22T00:37:19.093Z'}
```

Fig54. 数据更新成功

```
server > data > 1 17579815867253878 > interest  
1 {"name":"Jack","studentId":"2020302111111","email":"3276641807@qq.com","tel":"15995159515","interest":"basketball","id":"17579815867253878","date":
```

Fig55. 查看数据, 发现更新正常

## 4.4.6 list 实现

list 是通过列举出全部的数据条目进行显示, 默认是按照数据降序排列, 使用如下:

`-l | --list={mode}`: 当 mode 是 "descend" 时【默认】, 按照时间降序排列; 当 mode 是 "ascend", 升序排列, 输出到终端显示

示例: `app>>app -l ascend`

升序排列结果:

```
请输入样式为 app -h/--help 类型的指令: app -l ascend  
{'href': 'http://127.0.0.1:1337/api/17579815867253878', 'data': [{'name': 'name', 'value': 'Jack'}, {'name': 'studentId', 'value': '2020302111111'}, {'name': 'email', 'value': '3276641807@qq.com'}, {'name': 'tel', 'value': '15995159515'}, {'name': 'interest', 'value': 'basketball'}, {'name': 'date_posted', 'value': '2022-12-22T00:37:19.093Z'}]}  
{'href': 'http://127.0.0.1:1337/api/4438242551083096', 'data': [{'name': 'name', 'value': 'Jack'}, {'name': 'studentId', 'value': '20203021389'}, {'name': 'email', 'value': 'null'}, {'name': 'tel', 'value': '18777954123'}, {'name': 'interest', 'value': 'basketball'}, {'name': 'date_posted', 'value': '2022-12-23T13:03:40.687Z'}]}
```

Fig56. 升序排列结果

```
请输入样式为 app -h/--help 类型的指令: app -l  
{'href': 'http://127.0.0.1:1337/api/4438242551083096', 'data': [{'name': 'name', 'value': 'Jack'}, {'name': 'studentId', 'value': '20203021389'}, {'name': 'email', 'value': 'null'}, {'name': 'tel', 'value': '18777954123'}, {'name': 'interest', 'value': 'basketball'}, {'name': 'date_posted', 'value': '2022-12-23T13:03:40.687Z'}]}  
{'href': 'http://127.0.0.1:1337/api/17579815867253878', 'data': [{'name': 'name', 'value': 'Jack'}, {'name': 'studentId', 'value': '2020302111111'}, {'name': 'email', 'value': '3276641807@qq.com'}, {'name': 'tel', 'value': '15995159515'}, {'name': 'interest', 'value': 'basketball'}, {'name': 'date_posted', 'value': '2022-12-22T00:37:19.093Z'}]}
```

Fig57. 默认降序排列结果

#### 4.4.7 APP 核心代码实现思路

本次设计 APP 需要先手输入字符串，然后根据字符串解析指令，通过不同的指令分析不同的函数，从而达到解决问题的目的，解析指令如下图：

```
1. while(1):
2.     cmd = input("请输入样式为 app -h/--help 类型的指令:")
3.     temp_order = cmd.split() # 多空格字符串分割
4.     Fcmd = []
5.     for item in temp_order:
6.         item0 = item.split("=")
7.         for tempitem in item0:
8.             Fcmd.append(tempitem)
```

通过将指令中的字符存储为按照空格隔开的的字符串列表从而实现目的，嵌套 while 循环保证可以一直运行相关的代码，保证程序持续执行

当接受到指令后，传入设计的 GERTOREDER 函数：

```
1. # 通过 sys 获取命令行字符串实现分割指令，整体程序运行逻辑在本函数中
2. def GETORDER(cmd): # 处理输入命令
3.     # 判断命令是否以 app 开头，并判断命令有无选项
4.     if cmd[0] != "app":
5.         print("非正确使用，如继续使用，请输入 app -h/--help")
6.     elif (cmd[0] == "app" and len(cmd) == 1):
7.         # 无选项，按照"--list=descend"处理
8.         res = get_list('descend')
9.         if res[0] == 1: # 获取信息成功
10.             dataList = eval(res[1])
11.             dataList = dataList['collection']['items']
12.             for item in dataList:
13.                 print(item)
14.         else:
15.             print(res[1]) # 获取信息失败
16.     else:
17.         # 检查是否存在命令互斥情况，存在命令互斥会导致系统不稳定，
            因此需要考虑
18.         if checkorder(cmd) == False:
19.             print("命令互斥，请检查输入命令")
20.         else: # 不存在命令互斥，对命令进行分析
21.             if cmd[1] == "-h" or cmd[1] == "--help": # 输出
                对该App 的用法帮助
22.                 mainhelp(cmd)
23.             elif cmd[1] == "-a" or cmd[1] == "--add": # 增
                加一条新的条目
24.                 mainadd(cmd)
```

```
25.         elif cmd[1] == "-d" or cmd[1] == "--delete": #  
            对编号id 的条目删除操作  
26.             maindelete(cmd)  
27.         elif cmd[1] == "-u" or cmd[1] == "--update": #  
            对编号id 的条目进行修改  
28.             mainupdate(cmd)  
29.         elif cmd[1] == "-l" or cmd[1] == "--list": # 根  
            据mode 对所有数据按时间升序、降序排列  
30.             mainlist(cmd)  
31.         else:  
32.             print("指令有误, 请重新输入!")
```

GETORDER 函数通过检验互斥性后对于指令解析, 实现不同的功能, 互斥性检验的代码如下:

```
1. def checkorder(cmd): # 检查功能间是否互斥  
2.     List = ["-h", "--help", "-a", "--add", "-d", "--delete",  
            "-u", "--update", "-l", "--list"]  
3.     RCmd = cmd[1]  
4.     flag = True  
5.     for tcmd in List:  
6.         if tcmd != RCmd:  
7.             if tcmd in cmd:  
8.                 flag = False  
9.                 break  
10.        else:  
11.            flag = True  
12.    return flag
```

通过判断指令中是否包含多个关键词, 比如“app -h -u”包含关键词-h, -u,说明指令互斥, 因此需要对于指令字符列表判断:

```
1. def checkorder(cmd): # 检查功能间是否互斥  
2.     List = ["-h", "--help", "-a", "--add", "-d", "--delete",  
            "-u", "--update", "-l", "--list"]  
3.     RCmd = cmd[1]  
4.     flag = True  
5.     for tcmd in List:  
6.         if tcmd != RCmd:  
7.             if tcmd in cmd:  
8.                 flag = False  
9.                 break  
10.        else:  
11.            flag = True  
12.    return flag
```

判断完互斥性后, 需要对于指令做出操作, 这里使用了 Request 库进行交互操作, 在开发代码时, 想到的是利用写好的 API 直接调用, 在这里通过修改 curl 命令为 python 代码调用 API



，利用这个网站 ([Convert curl commands to Python \(curlconverter.com\)](http://curlconverter.com)) 可以实现将 curl 命令转换为 python 代码的操作，从而设计出对于不同的指令进行不同操作的 python 代码，由于报告篇幅缘由，不在这里对于代码逻辑进行过多介绍，粘贴出核心代码供参考：

<pre>01. def post_item(add): # 上传一条数据 02.     headers = { 03.         'Content-type': 'application/json', 04.     } 05.     json_data = { 06.         'template': { 07.             'data': [ 08.                 { 09.                     'name': 'name', 10.                     'value': '', 11.                 }, 12.                 { 13.                     'name': 'studentId', 14.                     'value': '', 15.                 }, 16.                 { 17.                     'name': 'email', 18.                     'value': 'null', 19.                 }, 20.                 { 21.                     'name': 'tel', 22.                     'value': '', 23.                 }, 24.                 { 25.                     'name': 'interest', 26.                     'value': '', 27.                 }, 28.             ], 29.         }, 30.     } 31.     # 对post方法的参数进行解析 32.     i = 0 33.     while i &lt; len(add): 34.         eCmd = add[i] 35.         eData = add[i + 1] 36.         if (eCmd == "-n" or eCmd == "--name"): # 姓名 37.             json_data['template']['data'][0]['value'] = eData 38.         elif (eCmd == "-i" or eCmd == "--id"): # 学号 39.             json_data['template']['data'][1]['value'] = eData 40.         elif (eCmd == "-m" or eCmd == "--telephone"): # 手机号 41.             json_data['template']['data'][3]['value'] = eData</pre>	<pre>71. 72. def get_list(sortMethod="descend"): # 获取所有数据，默认参数为降序排列 73.     headers = { 74.         'Content-type': 'application/json', 75.     } 76.     response = requests.get('http://127.0.0.1:1337/api/', headers=headers) 77. 78.     if response.status_code != 200 and response.status_code != 201: 79.         return [0, response.content.decode("utf-8")] 80. 81.     else: 82.         dataCol = eval(response.content.decode("utf-8")) 83.         data = dataCol["collection"]["items"] 84.         timeOfData = list() 85.         # 获取每一条记录的时间 86.         for item in data: 87.             timeOfData.append(item['data'][5]['value']) 88.         # 时间格式转换，str-&gt;float时间戳 89.         for i in range(len(timeOfData)): 90.             timeOfData[i] = time_convert(timeOfData[i]) 91.         # 记录原始数据下标 92.         orderOfData = list() 93.         for i in range(len(timeOfData)): 94.             orderOfData.append(i) 95. 96.         i = 0 97.         while i &lt; len(timeOfData): 98.             j = i + 1 99.             while j &lt; len(timeOfData): 100.                 if timeOfData[i] &lt; timeOfData[j]: 101.                     tempOrder = orderOfData[i] 102.                     orderOfData[i] = orderOfData[j] 103.                     orderOfData[j] = tempOrder 104.                 tempTime = timeOfData[i]</pre>
<pre>def get_item(itemid): # 获取一条数据     url = "http://127.0.0.1:1337/api/" + itemid     headers = {         'Content-type': 'application/json',     }     try:         response = requests.get(url, headers=headers)     except:         return [0, "请检查服务器连接"]     if response.status_code != 200:         return [0, "不存在此条数据，请检查输入id"]     else:         return [1, response.content.decode("utf-8")]  def put_item(input): # 更新一条数据     # 获取本id数据，以此为模板     id= input[2]     resGet = get_item(id)     if resGet[0] == 0:         return resGet     else:         temp1 = eval(resGet[1])["collection"]["items"]         data = temp1[0]['data']         i = 3         while i &lt; len(input):             if input[i] == '-n' or input[i] == '--name':                 data[0]['value'] = input[i + 1]             elif input[i] == '-i' or input[i] == '--id':                 data[1]['value'] = input[i + 1]             elif input[i] == '-m' or input[i] == '--telephone':                 data[3]['value'] = input[i + 1]             elif input[i] == '-b' or input[i] == '--hobby':</pre>	<pre>def put_item(input): # 更新一条数据     # 获取本id数据，以此为模板     id= input[2]     resGet = get_item(id)     if resGet[0] == 0:         return resGet     else:         temp1 = eval(resGet[1])["collection"]["items"]         data = temp1[0]['data']         i = 3         while i &lt; len(input):             if input[i] == '-n' or input[i] == '--name':                 data[0]['value'] = input[i + 1]             elif input[i] == '-i' or input[i] == '--id':                 data[1]['value'] = input[i + 1]             elif input[i] == '-m' or input[i] == '--telephone':                 data[3]['value'] = input[i + 1]             elif input[i] == '-b' or input[i] == '--hobby':                 data[4]['value'] = input[i + 1]             else:                 return [0, "更新用户信息时输入了无法识别的参数，请键入app -h查看帮助"]         i += 2  # 请取api实现更新 headers = {     'Content-type': 'application/json', } json_data = {     'template': {         'data': data,     }, } url = 'http://127.0.0.1:1337/api/' + id try:     response = requests.put(url, headers=headers, json=json_data) except:     return [0, "请检查服务器连接"]</pre>

Fig58. 调用 API 代码

最后，利用 if 语句判断执行命令，调用不同模块的函数，就实现了 API 的功能，本节 API 设计也就圆满结束。

## 5. 实习问题与解决方式

### 5.1 Java Script 代码报错

需要检查脚本中的语法，尤其检查是否正确的空行或者缩进。其次，脚本模块必须指定

输出字段，输出字段是新定义的变量名称，如果没有指定输出字段直接运行该脚本，或者将输出字段定义为输入变量都会报错。

## 5.2 修改页面显示时出错

在修改页面使其显示信息更加详细时，出现了一系列的问题，经过检查发现是因为在 Message.js 中没有及时修改相应的数值传输造成的错误，要注意字段在传递过程中的对应关系，才能避免错误的发生。

## 5.3 Curl post 时输入中文会出现乱码

通过查询资料了解到是因为编码方式的问题导致错误，由于 Windows 命令行本身的编码方式为 GBK，通过调整系统的编码方式为 UTF-8 可以使得中文字符正确输入，但由于系统的编码方式改变会导致较多的编码发生改变，所以本次展示中没有进行演示。

## 5.4 符合题目条件编写代码

实现实习指导中在 APP 操作时，忽略掉了邮箱一栏信息的操作，无论在增加还是更新时，都是默认邮箱作为空值实现的，利用 APP 存入数据后邮箱仍为空值，并不是因为操作错误，而是遵守了实习的要求，如果老师后续需要更改要求，可以修改代码实现邮箱的 APP 操作。

```
-n | --name={更新后的姓名}:  
-i | --id={更新后的学号}:  
-m | --mobile={更新后的手机号}  
-b | --hobby={更新后的兴趣爱好}
```

## 6. 特别说明

本次实验中由于数据在不断替换与更新，因此可能在实际举例中有些数据被本人操作删除了，如果老师在运行程序时，出现了数据不存在的条目，可以从下方选择数据进行验证（以下的数据是按照数据的 ID 号列出）：

1. 17579815867253878
2. 17472033787110441
3. 8081486890204845
4. 6294181661093061
5. 4438242551083096
6. 4042149832377966

以上数据均为最后文件夹中存在的数据，可以实际用于测验 API 与 APP 设计的功能。

## 7. 实习心得

经过了一周的信息系统集成与管理实习，我获得了很多深刻的体会。首先，实践可以大大促进对理论的理解。在进行一个学期的《信息系统集成与管理》理论课程的学习过程中，我对于有些概念和方法的理解是模糊的和不确定的，总觉得关于一些集成方式与网络设计方法距离自己太过遥远，但是通过实习，使我明白了原来理论并非遥不可及，我们可以利用理论真真切切地完成成果，自我实现网站的服务端与客户端。

其次，理论对实践具有很强的指导作用。倘若一位同学之前没有学习过相关理论课程，或是对于理论课程中的概念浅尝辄止，那么在实操的过程中会将会很难建立起前后步骤之间的关系。比如在检查 PUT 方法错误的时候，如果不清楚幂等性相关概念，是无法对朱老师所给代码进行透彻分析的。

然后，除了对于软件本身，我还掌握了理论知识对应的实际操作。该方面在代码编写的获取与操作中最为明显，例如 APP 的编写等。令我对于这些理论知识有了直观的理解。此外，我还在实习操的时候回顾了先行学科的知识点，明白了实际必须要联系理论才能法辉出它的最大价值。

最后，感谢朱老师和王老师对于我们的教导，经过一周的实习，我感受到了自身的进步，收获了知识与能力，在未来，一定会积极运用本次实践课上所收获的知识！