Python大作业

阿里云盘地址： https://www.aliyundrive.com/s/hUVTUX9PCyP 点击链接保存，或者复制本段内容，打开「阿里云盘」APP ，无需下载极速在线查看，视频原画倍速播放。

百度网盘地址：链接：https://pan.baidu.com/s/13sMZ5fY86bo4yMvtMFgBnw?pwd=37js
提取码：37js

码云仓库地址：https://gitee.com/xxxxasd/python-bighomework
（如果这个项目帮助到你了，麻烦点个赞，谢谢）

技术栈

vue2 + vuex + vue-router + webpack + ES6/7 +element-ui+ animate.css + flex弹性布局 + axios+ qs

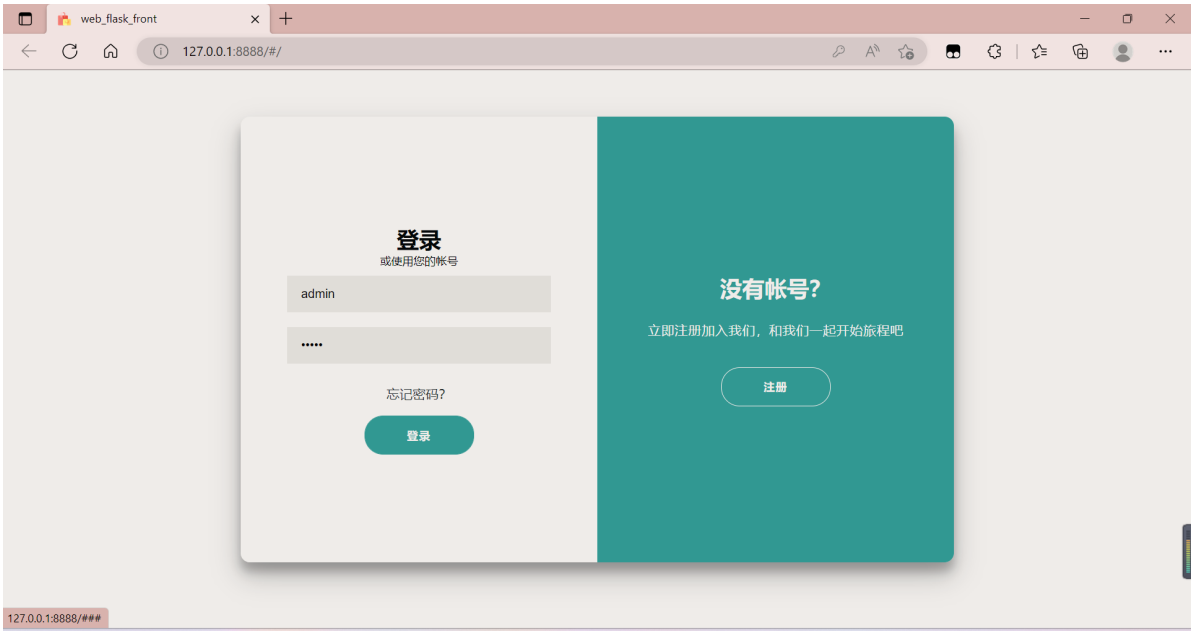
python + flask + mysql + flask_sqlalchemy + flask_mail + 爬虫

一.介绍

对拉勾网爬取招聘信息，并展示在网页中。实现对招聘信息的增删改查。前后端分离，前端采用vue2，后端采用flask.前台实现功能对职位的增删改查，以及登录注册功能，后端则提供相应的接口。主要有两个页面，登录

注册和主页面。

1.登录注册



二.主页面

二.软件架构

本项目为前后端分离项目，前端采用目前的主流框架vue ,后端则采用 flask。

(1).后端:

1.采用 flask_sqlalchemy 处理连接数据库 。简化对数据库的操作，。以下为连接数据库的代码，

```
# 数据库的配置信息
HOSTNAME = '127.0.0.1'
PORT = '3306'
DATABASE = 'flask_web'
USERNAME = 'root'
PASSWORD = 'admin123'
DB_URI = 'mysql+pymysql://{}:{}@{}:{}/{}?charset=utf8'.format(USERNAME,
PASSWORD, HOSTNAME, PORT, DATABASE)
SQLALCHEMY_DATABASE_URI = DB_URI
SQLALCHEMY_TRACK_MODIFICATIONS = False
```

2，采用面向对象思想建立数据表，同时使用 flask_sqlalchemy 简化建表语句。

```
class UserModel(db.Model):
    __tablename__ = "user"
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    # varchar, null=0
    username = db.Column(db.String(100), nullable=False)
    password = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(100), nullable=False, unique=True)
    Join_time = db.Column(db.DateTime, default=datetime.now)

    def dict(self):
        return {
            'id': self.id,
            'name': self.username,
            # 'password': self.password
            "email": self.email,
```

```
            "Join_time": self.Join_time
        }
```

3，使用flask内置的 Blueprint 写接口，最好分文件写，架构更清晰。在主文件中应用。

```
# Auth.py文件
from flask import Blueprint
#登录注册接口  地址前缀要加/auth
bp = Blueprint("Auth", __name__, url_prefix="/auth")


# User.py文件
from flask import Blueprint
#职位相关的接口  地址前缀要加/api
bp = Blueprint("user", __name__, url_prefix="/api")


# app.py
# 引入并使用
from blueprints.user import bp as user_bp
from blueprints.Auth import bp as Auth_bp
app.register_blueprint(user_bp)
app.register_blueprint(Auth_bp)
```

4，使用CORS实现后端跨域，至关重要 没有这一步，程序就瘫痪了。以下为跨域的官方解释。（现在只要知道会用就行）

出于浏览器的同源策略限制。同源策略（Sameoriginpolicy）是一种约定，它是浏览器最核心也最基本的安全功能，如果缺少了同源策略，则浏览器的正常功能可能都会受到影响。可以说Web是构建在同源策略基础之上的，浏览器只是针对同源策略的一种实现。同源策略会阻止一个域的javascript脚本和另外一个域的内容进行交互。所谓同源（即指在同一个域）就是两个页面具有相同的协议（protocol），主机（host）和端口号（port）。

```
from flask_cors import CORS
CORS(app, resources=r'/*')
#后端实现跨域
```

5，对本项目的数据显示，使用 requests re解析对拉勾网的职位信息进行爬取，先以csv形式保存于本地，再从本地读取并设置相应字段存入数据库，并在前端进行展示。

```
# 1.导入工具库
import csv
import requests
import re

from spider.mysql import CSV_SQL

url = 'https://www.lagou.com/wn/zhaopin'
```

```
headers = {
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko)'
                  ' Chrome/98.0.4758.82 Safari/537.36'
}
response = requests.get(url=url, headers=headers)
# print(response.text)
str = re.compile(
    '<div class="item__1ORTO">.*?<a>(?P<positionName>.*?)<!-- -->(?P<address>.*?)
</a><span>(?P<time>.*?)'
    '</span></div><div class="p-bom__JlNur"><span ''class="money__3Lkgq">(?
P<money>.*?)</span>(?P<education>.*?)'
    '</div>.*?<a>.*?</a></div><div class="industry__1HBkr">.*?</div></div>''<div
class="com-logo'
    '__1QOwC"><img src="(?P<posturl>.*?)" .*?</div><div class="il__3lk85">(?
P<jobNature>.*?)</div></div>'
    '</div>', re.S)
html_data = re.findall(str, response.text)
# print(html_data)
for it in html_data:
    # print(it)
    positionName = it[0]
    address = it[1].strip("[]")
    publishtime = it[2]
    money = it[3]
    education = it[4]
    posturl = it[5]
    jobNature = it[6].strip("”“”")
    # print('正在爬取：', positionName)
    with open('拉勾网.csv', mode='a', encoding='utf-8', newline='') as csvfile:
        csvwriter = csv.writer(csvfile, delimiter=',')  # delimiter=','  csv数据
的分隔符
        csvwriter.writerow(
            [positionName, address, publishtime, money, education, posturl,
jobNature])  # 序列化数据，写入csv
print("爬取成功")
CSV_SQL() #  写入数据库函数
```

```
import pymysql

# 自定义函数CSV_SQL() 写入数据库
def CSV_SQL():
    sql_conn = pymysql.connect(host='127.0.0.1', port=3306, user='root',
                               password='admin123', db='flask_web',
charset='utf8', connect_timeout=1000)
    # 创建数据库对象
    cursor = sql_conn.cursor()
    with open('C:\PycharmProjects\pythonProject\web\spider\拉勾网.csv',
encoding='utf-8') as line_1:
        # 依次读取CSV文件的每一行
        for line_2 in line_1.readlines():
            # strip() 方法用于移除字符串头尾指定的字符（默认为空格或换行符）或字符序列“
            line_2 = line_2.strip()
            # split() 通过指定分隔符对字符串进行切片，这里指定','，而“-1”表示分隔所有
            list_1 = line_2.split(',', -1)
```

```
            # 执行插入表数据语句
            sql_3 = 'INSERT INTO csv_position_py (positionName, address,
publishtime, money, education, ' \
                    'posturl,jobNature) value (%s, %s, %s, %s, %s, %s, %s) '
            cursor.execute(sql_3,
                           (list_1[0], list_1[1], list_1[2], list_1[3],
list_1[4], list_1[5], list_1[6]))
    sql_conn.commit()  # 提交事务
    sql_conn.close()  # 关闭连接
    print('写入数据库成功')
```

(2).前端

（此不属于Python内容，故不做过多介绍。如老师不相信，可向我联系）

1.主要框架为vue，使用vue脚手架创建项目，

2.使用 animate.css 来实现动画过度 。

3.使用 axios 发起AJAX 请求，请求后端资源，

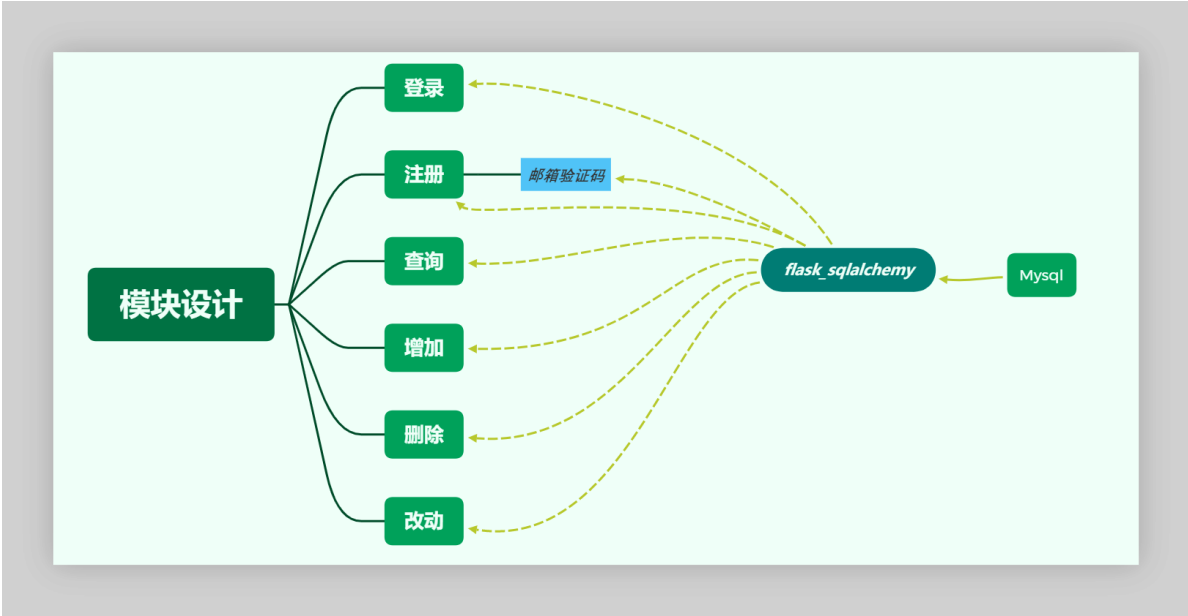
4.使用 element-ui 快速搭建网站。

5.使用 vue-router 进行路由跳转。

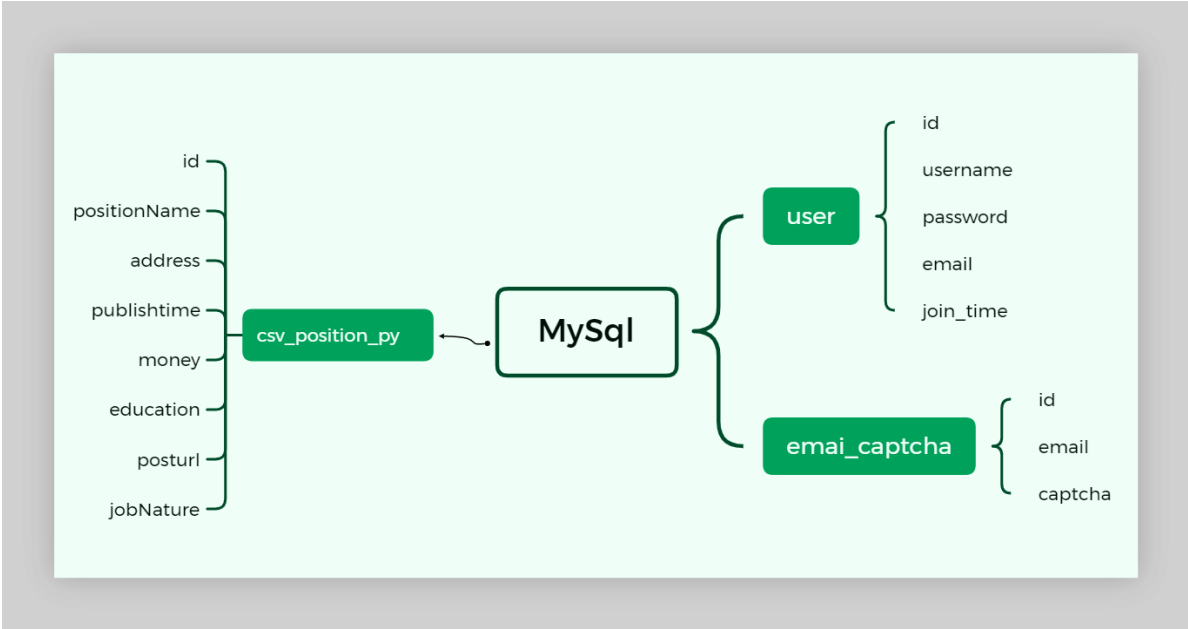
6.使用 qs 进行对象与字符串之间转换。

三.功能实现

总览



数据库设置



1.登录功能

请求为 post, 请求后调用user_login函数，获取登录账号密码，，进行数据库比对，错误返回相应信息，

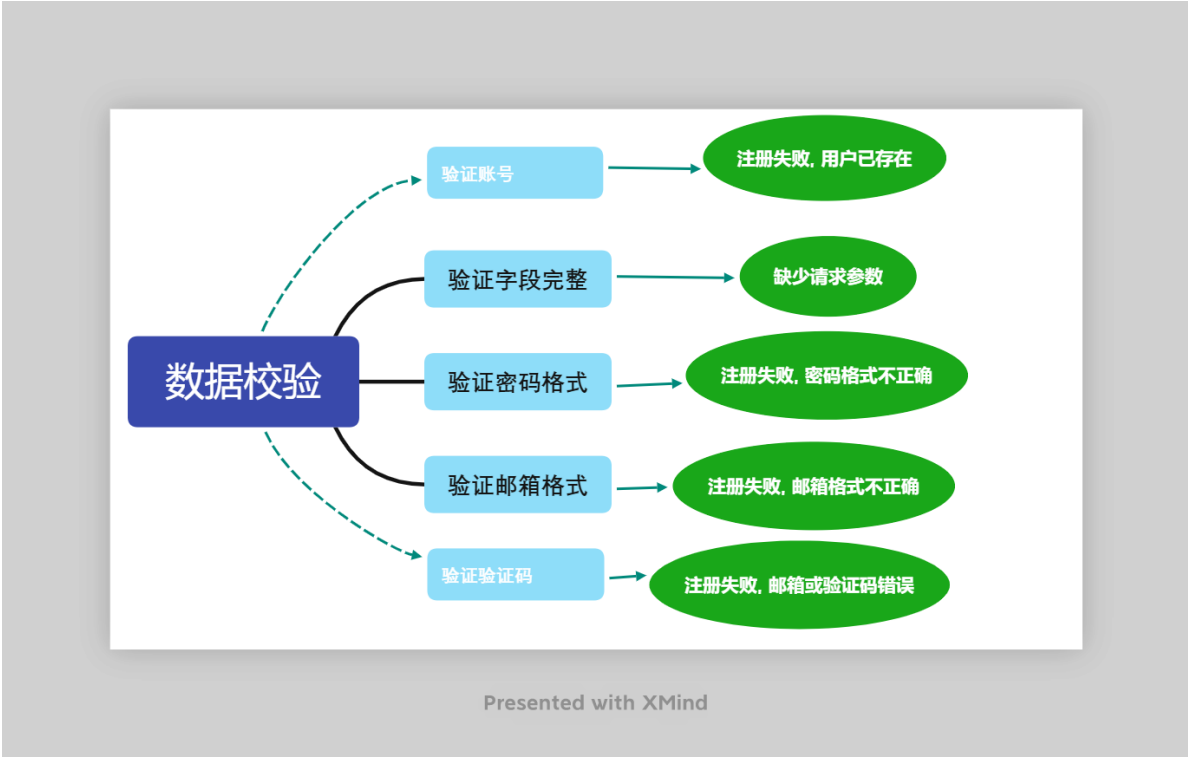
```
@bp.route('/login', methods=['POST'])
def user_login():
    username = request.values.get('username')
    password = request.values.get('password')
    print(username, password)
    # 2. 处理请求参数缺少
    if not all([username, password]):
        return r(code=401, msg='登录, 缺少请求参数')
    # 3. 验证账号和密码
    user = UserModel.query.filter_by(username=username,
password=password).first()
    # 4. 用户不存在, 直接返回
    if not user:
        return r(code=404, msg='用户名或密码错误')
    else:
        # 5. 保存用户状态到 session
        session['user_info'] = user.dict()
        return r(msg='登录成功', data=user.dict())
```

2.获取验证码功能

需预先进行邮箱的配置

```
# 邮箱配置
MAIL_SERVER = "smtp.qq.com"
MAIL_USE_SSL = True
MAIL_PORT = 465
MAIL_USERNAME = "1521891733@qq.com"
MAIL_PASSWORD = "lxngxstqmmcahgif"
MAIL_DEFAULT_SENDER = "1521891733@qq.com"
```

用户点击获取验证码后，调用接口，获取验证码，发送到用户所填写的邮箱。




```
@bp.route('/captcha', methods=["POST"])
def get_email_captcha():
    email = request.values.get('email')
    print(email)
    source = string.digits * 4
    captcha = random.sample(source, 4)
    captcha = "".join(captcha)
    message = Message(subject="拉勾网注册验证码", recipients=[email], body=f"您的验证码是:{captcha}")
    mail.send(message)
    # 用数据库表的方式存储
    email_captcha = EmailCaptchaModel(email=email, captcha=captcha)
    db.session.add(email_captcha)
    db.session.commit()
    return jsonify({"code": 200, "message": "", "data": None})
```

3.注册功能

请求也为post 调用user_register函数，获取账号密码和邮箱，进行数据验证后往数据库中添加。

```
@bp.route('/register', methods=['POST'])
def user_register():
    username = request.values.get('username')
    password = request.values.get('password')
    email = request.values.get("email")
    code = request.values.get("code")
    # 2. 处理请求参数缺少
    if not all([username, password, email, code]):
        return r(code=401, msg='注册失败，缺少请求参数', )
    # 3. 根据 user_name 字段判断用户是否注册过
    str2 = r"^(?![0-9]+$)(?![a-zA-Z]+$)[0-9A-Za-z]{8,15}$"
    if not re.match(str2, password):
```

```
        return r(code=405, msg='注册失败，密码格式不正确', )
    str1 = r"^[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+){0,4}@[a-zA-Z0-9_-]+(\.[a-zA-Z0-
9_-]+){0,4}$"
    if not re.match(str1, email):
        return r(code=404, msg='注册失败，邮箱格式不正确', )
    captcha_model = EmailCaptchaModel.query.filter_by(email=email,
captcha=code).first()
    if not captcha_model:
        return r(code=406, msg='注册失败，邮箱或验证码错误' )
    try:
        user = UserModel.query.filter_by(username=username).first()
        if user:
            return r(code=403, msg='注册失败，用户已存在')
    except Exception as e:
        print(e, '[Error] in [/user] [POST] when select MYSQL user where name=[]
.')
        return r(code=402, msg='服务器内部出错')
    # 4. 注册用户到数据库
    user = UserModel(username=username, password=password, email=email)
    try:
        db.session.add(user)
        db.session.commit()  # 提交事务
    except Exception as e:
        db.session.rollback()  # 异常时回滚
        print(e, '[Error] in [/user] [POST] when inserting a user into MySQL.')
        return r(code=402, msg='服务器内部出错')
    print('新用户注册成功:  ', user.dict())
    db.session.delete(captcha_model)
    db.session.commit()
    return r(code=200, msg='注册成功', data=user.dict())
```

4.查询所有职位

查询功能，请求为get，循环每一项改为字典形式，最后以json 形式返回到前端.

```
# 查询所有
@bp.route("/user", methods=['GET'])
def index():
    # userList: List[PositionModel] = PositionModel.query.all()
    # return r(code=200, msg='用户查询成功', data=model_list_to_dict(userList))
    result = PositionModel.query.all()
    payload = []
    for i in result:
        payload.append(i.dict())
    return jsonify(payload)
```

5.删除

删除功能，请求为get 请求参数为职位的id, 在数据库中找到再删除，

```
# 删除
@bp.route("/delete/<int:del_id>")
def delete(del_id):
    """删除数据"""
    # 方式1: 先查后删除
    Position = PositionModel.query.filter_by(id=del_id).first()
    # 删除数据
    db.session.delete(Position)
    # 提交会话 增删改都要提交会话
    db.session.commit()
    return r(code=200, msg='删除成功')
```

6.增加

增加功能，请求为post ，接受表单数据，判断是否合法，再提交数据库，

```
@bp.route("/add", methods=["POST"])
def add_user():
    msg = "添加成功！"
    print(request.json)
    if "positionName" in request.json and "address" in request.json and
"publishtime" in request.json and "money" in \
            request.json and "education" in request.json and "posturl" in
request.json and "jobNature" in request.json:
        new_user = PositionModel()
        new_user.positionName = request.json["positionName"]
        new_user.address = request.json["address"]
        new_user.publishtime = request.json["publishtime"]
        new_user.money = request.json["money"]
        new_user.education = request.json["education"]
        new_user.posturl = request.json["posturl"]
        new_user.jobNature = request.json["jobNature"]
        db.session.add(new_user)
        db.session.commit()
    else:
        msg = "添加失败！缺少参数"
    return r(code=201, msg=msg)
```

7.改

改变职位信息功能，请求为post ，接受表单数据，根据id 进行改变。

```
# 改
@bp.route("/update", methods=["POST"])
def update_user():
    msg = "修改成功！"
    print(request.json)
    if "positionName" in request.json and "address" in request.json and
"publishtime" in request.json and "money" in \
            request.json and "education" in request.json and "posturl" in
request.json and "jobNature" in request.json:
        update_id = request.json["id"]
        update_user = PositionModel.query.get(update_id)
```

```
        update_user.positionName = request.json["positionName"]
        update_user.address = request.json["address"]
        update_user.publishtime = request.json["publishtime"]
        update_user.money = request.json["money"]
        update_user.education = request.json["education"]
        update_user.posturl = request.json["posturl"]
        update_user.jobNature = request.json["jobNature"]
        db.session.commit()
    else:
        msg = "修改失败！"
    return r(code=201, msg=msg)
```

四.心得体会

首先最开始，我们要明确一点，学习编程不是一日之功，需要每天投入时间学习，也不可纸上谈兵，需要自己亲自操作，不动手就不会发现问题，动了手印象才会深刻，记得更靠。

通过此次总时长超过30个小时的开发，我从零基础学flask web 框架，因以前学过express MySQL学起来还是比较快的， 一开始要充分想好自己要做什么项目，要实现什么功能，以及后期一些模板的加入，考虑代码的健壮性这是必备的，不然后期加功能的时候就会想我一样。。改动的地方非常多。还容易出现bug 。

我是一名编程爱好者，享受着解决 bug 后的喜悦，以及看着满屏的代码给自己一种莫名的喜欢，有时候遇到一个bug ，改来改去，头皮发麻，各种网上查找资料，那种感觉真的很痛苦。但当解决后。就想如获新生一样。

通过比次实训，我学到了flask 框架的基本用法。能基本掌握开发流程和规范，以及对Python 数据类型掌握的更深，了解更多Python 的库，

五.不足

1，前端没有做数据检验，说走提交的数据都有后端来处理，增大了服务器的负担。
2，本次使用操作数据库的工具是flask _sqlarcmy 简化sql 语句，应加强对sql 语句的书写和使用。
3，对 ui 不是特别敏感，导致做出来的页面比较丑。
4，使用爬虫爬取数据时，使用了re 正则表达式去匹配。因不熟其他方法，。导致re 正则非常长，下一步可以采用很好的解析方式。
5，请求后端返回数据时，没有形成一套体系，返回的状态码比较乱，以及数据处理和返回结构不够清晰。