

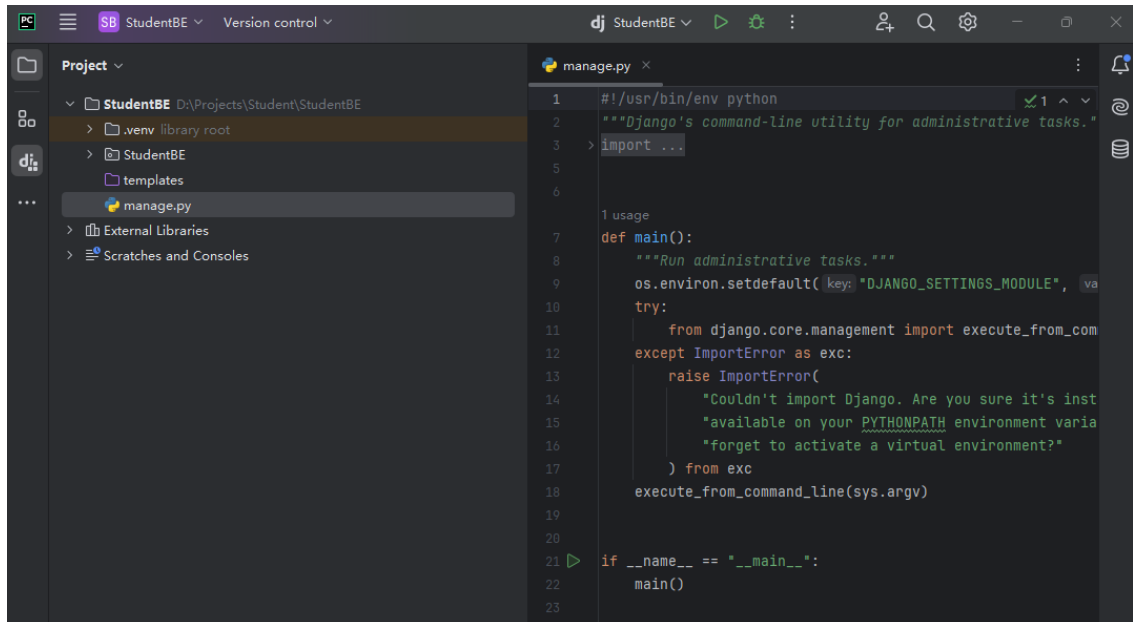
Python+Vue+Django前后端分离项目实战【Student】

Python+Vue+Django前后端分离项目实战【Student】

- 1.后端项目创建和数据库准备
- 2.前端Vue项目的初始化
- 3.引入Element UI和页面结构
- 4.使用Element UI实现侧边栏
- 5.使用Element UI实现表单布局
- 6.使用Element UI中表格展示数据
- 7.使用Element UI实现分页展示
- 8.实现获取所有学生的后端接口
- 9.使用Axios请求后端接口的数据
- 10.解决Ajax跨域请求
- 11.把后端接口返回的数据展示在表格中
- 12.实现表格展示数据的分页功能
- 13.实现学生信息查询的后端接口
- 14.完成学生信息查询的功能
- 15.学生管理系统增删改查的总体概述
- 16.使用Element UI实现弹出框的表单
- 17.美化Element UI弹出框表单
- 18.浅拷贝和深拷贝
- 19.三种状态加载同一个表单
- 20.表单提交前的校验
- 21.从后端校验学号是否存在
- 22.完成学号是否存在的校验
- 23.表单的校验和重置
- 24.实现添加学生的后端接口
- 25.完成学生添加的功能
- 26.完成学生修改功能
- 27.完成学生的删除功能
- 28.完成学生批量删除的功能
- 29.使用Element完成表单中照片的显示
- 30.照片上传、存储、展示的过程分析
- 31.完成上传照片后端接口
- 32.实现照片的上传和展示
- 33.实现照片存储在数据库
- 34.读取Excel文件数据到Python
- 35.实现Excel批量导入数据的后端接口
- 36.完成Excel批量导入的功能
- 37.完成导出到Excel的功能

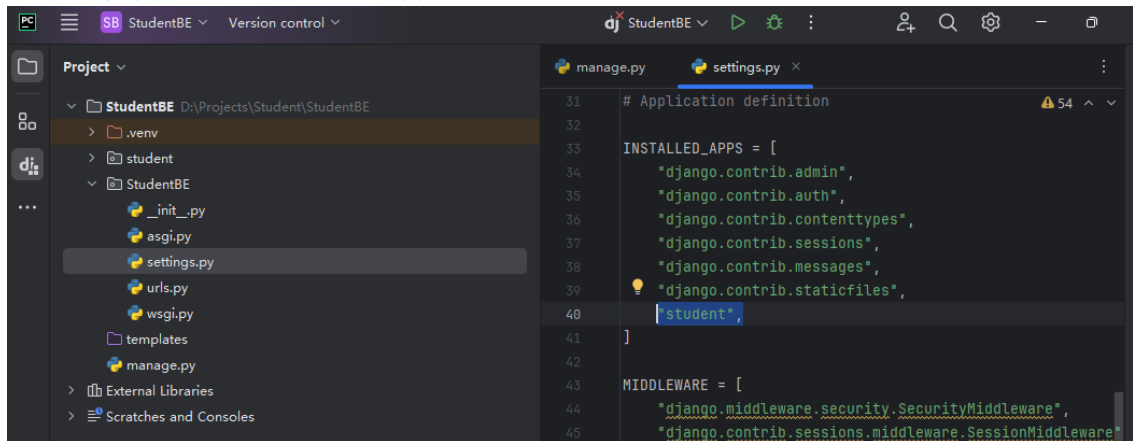
1.后端项目创建和数据库准备

1. Pycharm新建一个项目，选择Django，然后取一个合适的名字，其他内容默认。

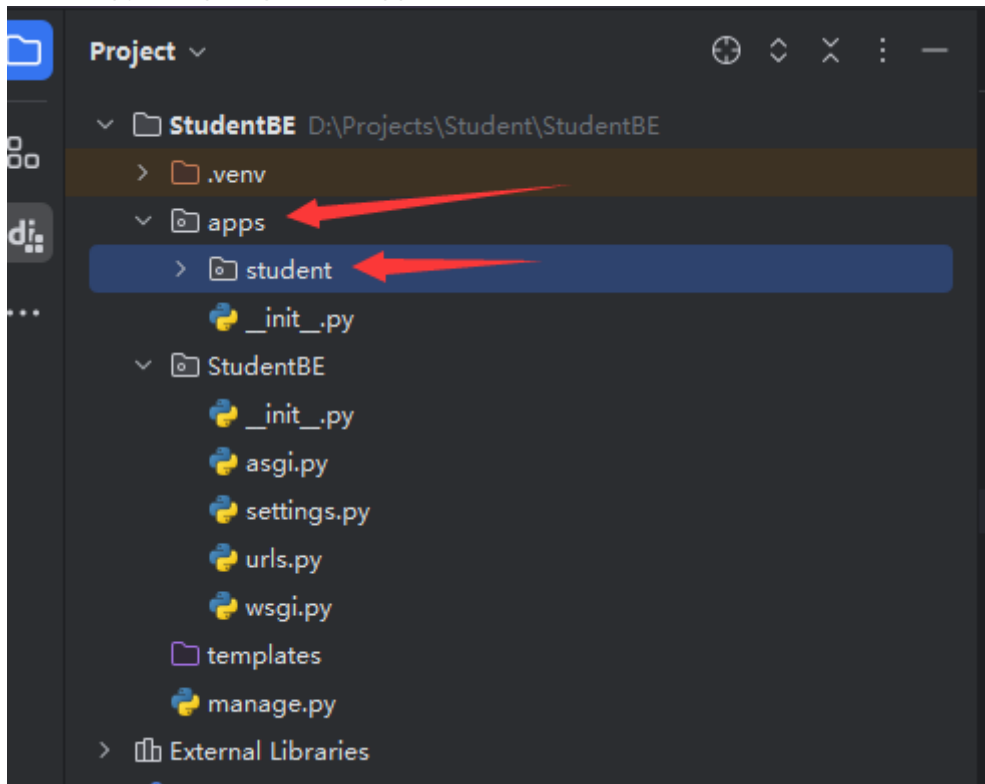


2. 使用指令python manage.py startapp student创建app。

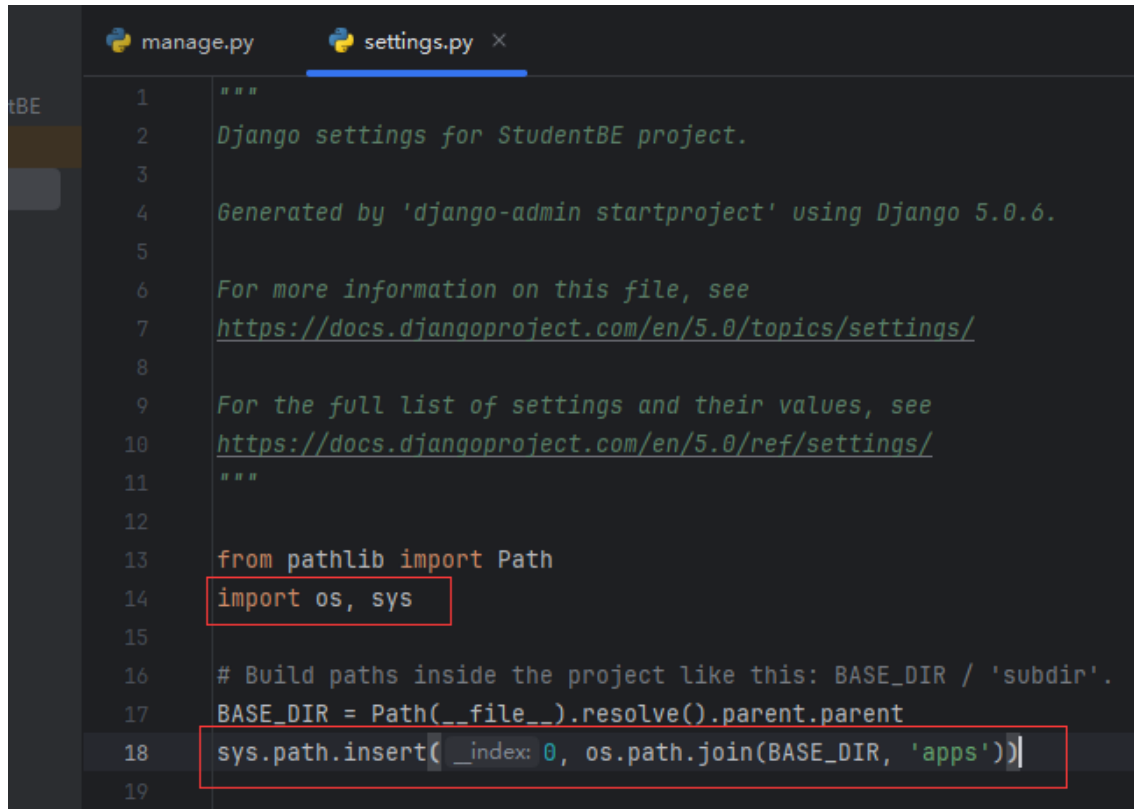
3. 在settings.py中注册student这个app。



4. 创建一个python package用来放app，然后把student拖进去。

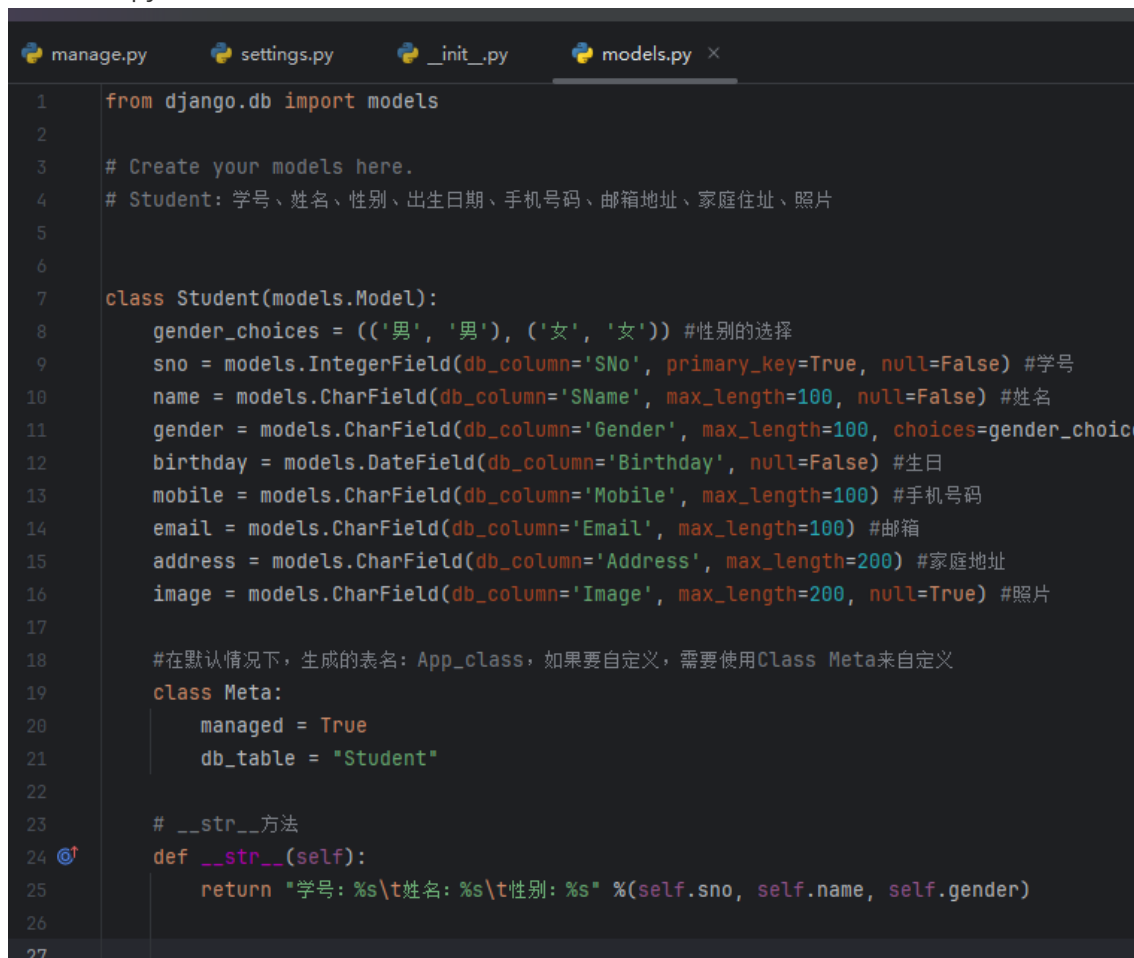


5. 把apps标记为source root，然后在settings.py中加入apps这个路径。



```
1 """
2 Django settings for StudentBE project.
3
4 Generated by 'django-admin startproject' using Django 5.0.6.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/5.0/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/5.0/ref/settings/
11 """
12
13 from pathlib import Path
14 import os, sys
15
16 # Build paths inside the project like this: BASE_DIR / 'subdir'.
17 BASE_DIR = Path(__file__).resolve().parent.parent
18 sys.path.insert(0, os.path.join(BASE_DIR, 'apps'))
19
```

6. 在models.py中新建一个Student类，用于映射。



```
1 from django.db import models
2
3 # Create your models here.
4 # Student: 学号、姓名、性别、出生日期、手机号码、邮箱地址、家庭住址、照片
5
6
7 class Student(models.Model):
8     gender_choices = (('男', '男'), ('女', '女')) # 性别的选择
9     sno = models.IntegerField(db_column='SNo', primary_key=True, null=False) # 学号
10    name = models.CharField(db_column='SName', max_length=100, null=False) # 姓名
11    gender = models.CharField(db_column='Gender', max_length=100, choices=gender_choices)
12    birthday = models.DateField(db_column='Birthday', null=False) # 生日
13    mobile = models.CharField(db_column='Mobile', max_length=100) # 手机号码
14    email = models.CharField(db_column='Email', max_length=100) # 邮箱
15    address = models.CharField(db_column='Address', max_length=200) # 家庭地址
16    image = models.CharField(db_column='Image', max_length=200, null=True) # 照片
17
18    # 在默认情况下，生成的表名: App_class，如果要自定义，需要使用Class Meta来自定义
19    class Meta:
20        managed = True
21        db_table = "Student"
22
23    # __str__方法
24    def __str__(self):
25        return "学号: %s\t姓名: %s\t性别: %s" % (self.sno, self.name, self.gender)
26
27
```

7. `from django.db import models`

```

# Create your models here.
# Student: 学号, 姓名, 性别, 出生日期, 手机号码, 邮箱地址, 家庭住址, 照片

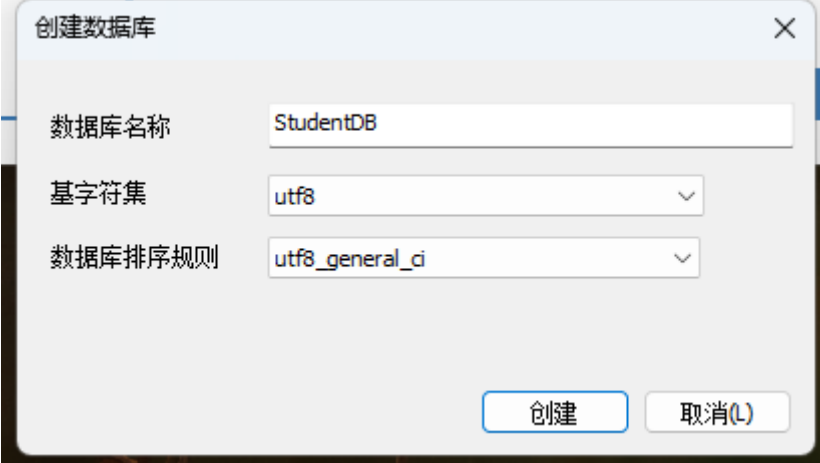
class Student(models.Model):
    gender_choices = (('男', '男'), ('女', '女'))
    sno = models.IntegerField(db_column="SNo", primary_key=True, null=False)
# 学号, 不允许为空, 主键
    name = models.CharField(db_column="SName", max_length=100, null=False)
# 姓名, 最长100个字符, 不允许为空
    gender =
models.CharField(db_column="Gender", max_length=100, choices=gender_choices) #
性别, 选项选择
    birthday = models.DateField(db_column="Birthday", null=False) # 出生日期,
不允许为空
    mobile = models.CharField(db_column="Mobile", max_length=100) # 手机号码,
    email = models.CharField(db_column="Email", max_length=100) # 邮箱地址
    address = models.CharField(db_column="Address", max_length=200) # 家庭住址
    image = models.CharField(db_column="Image", max_length=200, null=True) #
照片

# 在默认情况下, 生成的表名: App_class, 如果要自定义, 需要使用Class Meta来自定义
class Meta:
    managed = True
    db_table = "Student"

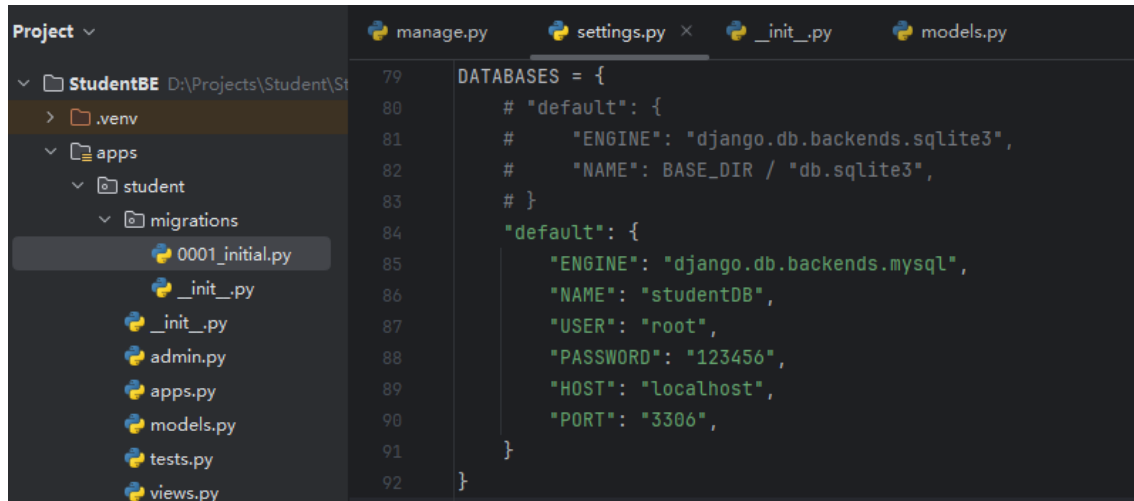
# __str__方法
def __str__(self):
    return "学号:%s\t姓名:%s\t性别:%s" %(self.sno, self.name, self.gender)

```

8. 新建数据库。

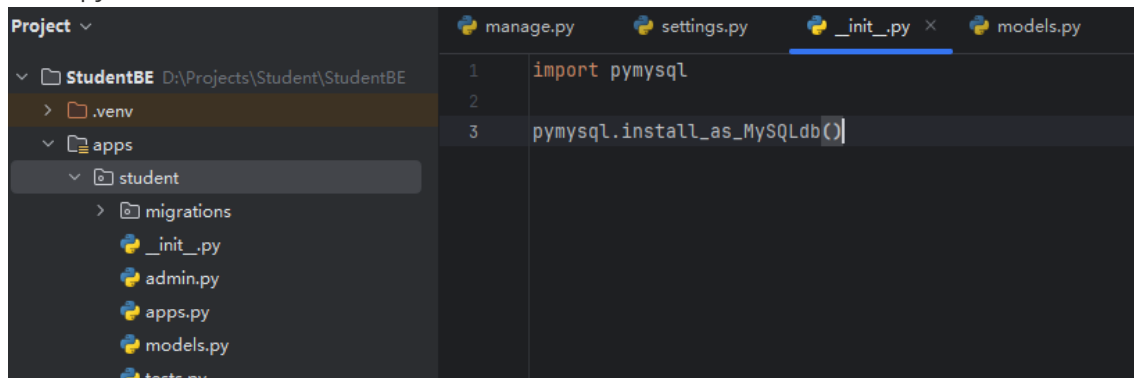


9. 在settings中DATANASES的设置。



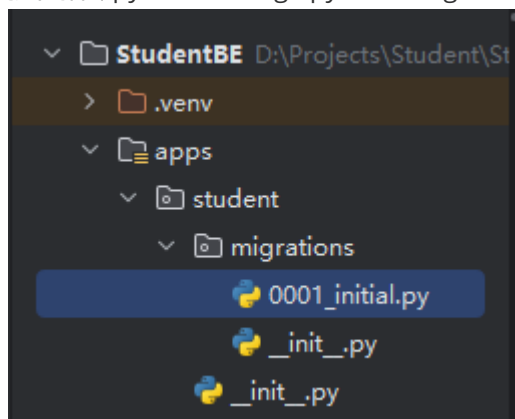
```
79 DATABASES = {
80     # "default": {
81     #     "ENGINE": "django.db.backends.sqlite3",
82     #     "NAME": BASE_DIR / "db.sqlite3",
83     # }
84     "default": {
85         "ENGINE": "django.db.backends.mysql",
86         "NAME": "studentDB",
87         "USER": "root",
88         "PASSWORD": "123456",
89         "HOST": "localhost",
90         "PORT": "3306",
91     }
92 }
```

10. 在init.py中引入模块。

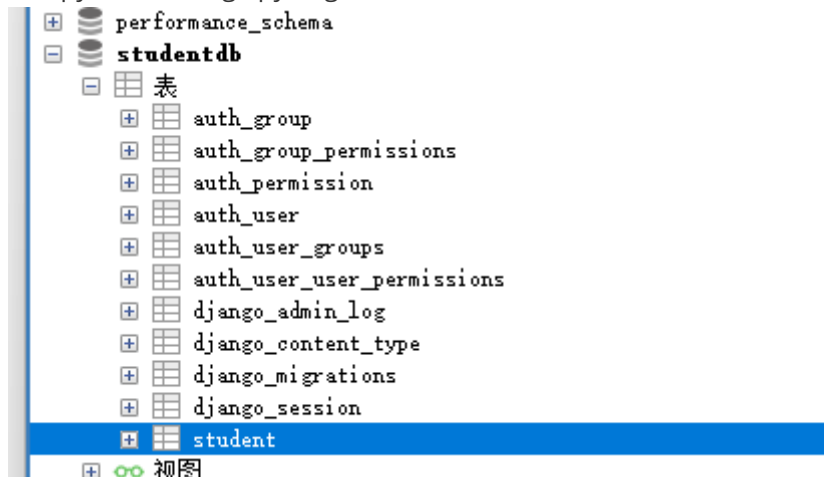


```
1 import pymysql
2
3 pymysql.install_as_MySQLdb()
```

11. 执行指令python manage.py makemigrations, 生成本地脚本



12. 执行python manage.py migrate, 生成表格。

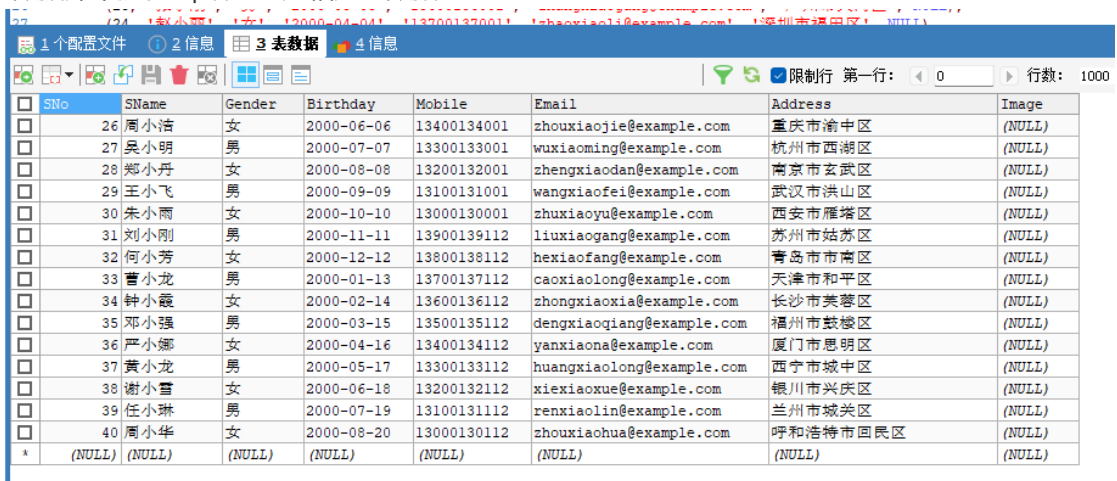


13. -- 插入学生数据

```
INSERT INTO student (SNO, SName, Gender, Birthday, Mobile, Email, Address,
Image)
VALUES
(1, '张三', '男', '2000-01-01', '13800138000', 'zhangsan@example.com', '北京市朝阳区', NULL),
(2, '李四', '女', '2000-02-02', '13900139000', 'lisi@example.com', '上海市浦东新区', NULL),
(3, '王五', '男', '2000-03-03', '13600136000', 'wangwu@example.com', '广州市天河区', NULL),
(4, '赵六', '女', '2000-04-04', '13700137000', 'zhaoliu@example.com', '深圳市南山区', NULL),
(5, '钱七', '男', '2000-05-05', '13500135000', 'qianqi@example.com', '成都市武侯区', NULL),
(6, '孙八', '女', '2000-06-06', '13400134000', 'sunba@example.com', '重庆市渝中区', NULL),
(7, '周九', '男', '2000-07-07', '13300133000', 'zhoujiu@example.com', '杭州市西湖区', NULL),
(8, '吴十', '女', '2000-08-08', '13200132000', 'wushi@example.com', '南京市玄武区', NULL),
(9, '郑十一', '男', '2000-09-09', '13100131000', 'zhengshi@example.com', '武汉市洪山区', NULL),
(10, '王十二', '女', '2000-10-10', '13000130000', 'wangshier@example.com', '西安市雁塔区', NULL),
(11, '朱十三', '男', '2000-11-11', '13900139111', 'zhushisan@example.com', '苏州市姑苏区', NULL),
(12, '刘十四', '女', '2000-12-12', '13800138111', 'liushisi@example.com', '青岛市市南区', NULL),
(13, '何十五', '男', '2000-01-13', '13700137111', 'heshiwu@example.com', '天津市和平区', NULL),
(14, '曹十六', '女', '2000-02-14', '13600136111', 'caoshiyi@example.com', '长沙市芙蓉区', NULL),
(15, '钟十七', '男', '2000-03-15', '13500135111', 'zhongshiqi@example.com', '福州市鼓楼区', NULL),
(16, '邓十八', '女', '2000-04-16', '13400134111', 'dengshiba@example.com', '厦门市思明区', NULL),
(17, '严十九', '男', '2000-05-17', '13300133111', 'yanshijiu@example.com', '西宁市城中区', NULL),
(18, '黄二十', '女', '2000-06-18', '13200132111', 'huangershisi@example.com', '银川市兴庆区', NULL),
(19, '谢廿一', '男', '2000-07-19', '13100131111', 'xiexishi@example.com', '兰州市城关区', NULL),
(20, '任廿二', '女', '2000-08-20', '13000130111', 'renershier@example.com', '呼和浩特市回民区', NULL),
(21, '王小明', '男', '2000-01-01', '13800138001', 'wangxiaoming@example.com', '北京市海淀区', NULL),
(22, '李小红', '女', '2000-02-02', '13900139001', 'lixiaohong@example.com', '上海市静安区', NULL),
(23, '张小刚', '男', '2000-03-03', '13600136001', 'zhangxiaogang@example.com', '广州市天河区', NULL),
(24, '赵小丽', '女', '2000-04-04', '13700137001', 'zhaoxiaoli@example.com', '深圳市福田区', NULL),
(25, '孙小伟', '男', '2000-05-05', '13500135001', 'sunxiaowei@example.com', '成都市锦江区', NULL),
(26, '周小洁', '女', '2000-06-06', '13400134001', 'zhouxiaojie@example.com', '重庆市渝中区', NULL),
```

```
(27, '吴小明', '男', '2000-07-07', '13300133001',
'wuxiaoming@example.com', '杭州市西湖区', NULL),
(28, '郑小丹', '女', '2000-08-08', '13200132001',
'zhengxiaodan@example.com', '南京市玄武区', NULL),
(29, '王小飞', '男', '2000-09-09', '13100131001',
'wangxiaofei@example.com', '武汉市洪山区', NULL),
(30, '朱小雨', '女', '2000-10-10', '13000130001', 'zhuxiaoyu@example.com',
'西安市雁塔区', NULL),
(31, '刘小刚', '男', '2000-11-11', '13900139112',
'liuxiaogang@example.com', '苏州市姑苏区', NULL),
(32, '何小芳', '女', '2000-12-12', '13800138112',
'hexiaofang@example.com', '青岛市市南区', NULL),
(33, '曹小龙', '男', '2000-01-13', '13700137112',
'caoxiaolong@example.com', '天津市和平区', NULL),
(34, '钟小霞', '女', '2000-02-14', '13600136112',
'zhongxiaoxia@example.com', '长沙市芙蓉区', NULL),
(35, '邓小强', '男', '2000-03-15', '13500135112',
'dengxiaoqiang@example.com', '福州市鼓楼区', NULL),
(36, '严小娜', '女', '2000-04-16', '13400134112', 'yanxiaona@example.com',
'厦门市思明区', NULL),
(37, '黄小龙', '男', '2000-05-17', '13300133112',
'huangxiaolong@example.com', '西宁市城中区', NULL),
(38, '谢小雪', '女', '2000-06-18', '13200132112',
'xiexiaoxue@example.com', '银川市兴庆区', NULL),
(39, '任小琳', '男', '2000-07-19', '13100131112',
'renxiaolin@example.com', '兰州市城关区', NULL),
(40, '周小华', '女', '2000-08-20', '13000130112',
'zhouxiaohua@example.com', '呼和浩特市回民区', NULL);
```

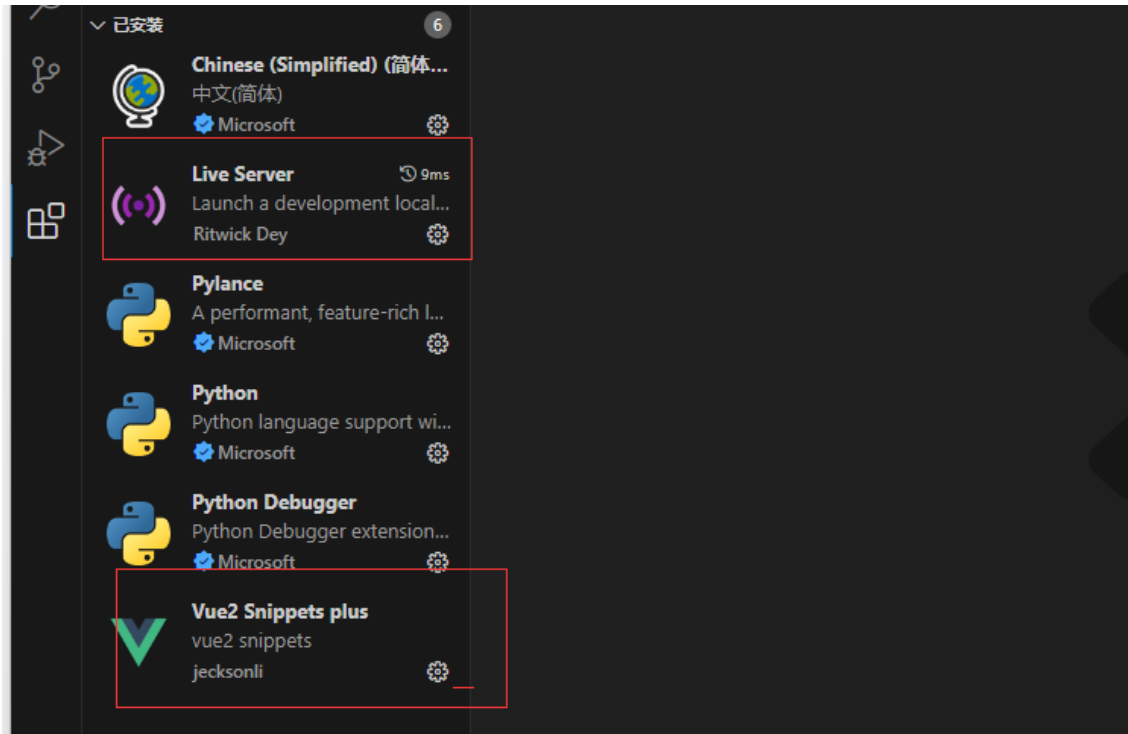
14. 在数据库中执行sql语句创建相应的数据。



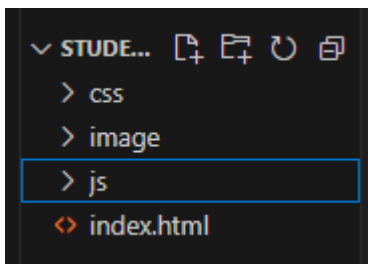
SNo	SName	Gender	Birthday	Mobile	Email	Address	Image
26	周小洁	女	2000-06-06	13400134001	zhouxiaojie@example.com	重庆市渝中区	(NULL)
27	吴小明	男	2000-07-07	13300133001	wuxiaoming@example.com	杭州市西湖区	(NULL)
28	郑小丹	女	2000-08-08	13200132001	zhengxiaodan@example.com	南京市玄武区	(NULL)
29	王小飞	男	2000-09-09	13100131001	wangxiaofei@example.com	武汉市洪山区	(NULL)
30	朱小雨	女	2000-10-10	13000130001	zhuxiaoyu@example.com	西安市雁塔区	(NULL)
31	刘小刚	男	2000-11-11	13900139112	liuxiaogang@example.com	苏州市姑苏区	(NULL)
32	何小芳	女	2000-12-12	13800138112	hexiaofang@example.com	青岛市市南区	(NULL)
33	曹小龙	男	2000-01-13	13700137112	caoxiaolong@example.com	天津市和平区	(NULL)
34	钟小霞	女	2000-02-14	13600136112	zhongxiaoxia@example.com	长沙市芙蓉区	(NULL)
35	邓小强	男	2000-03-15	13500135112	dengxiaoqiang@example.com	福州市鼓楼区	(NULL)
36	严小娜	女	2000-04-16	13400134112	yanxiaona@example.com	厦门市思明区	(NULL)
37	黄小龙	男	2000-05-17	13300133112	huangxiaolong@example.com	西宁市城中区	(NULL)
38	谢小雪	女	2000-06-18	13200132112	xiexiaoxue@example.com	银川市兴庆区	(NULL)
39	任小琳	男	2000-07-19	13100131112	renxiaolin@example.com	兰州市城关区	(NULL)
40	周小华	女	2000-08-20	13000130112	zhouxiaohua@example.com	呼和浩特市回民区	(NULL)
*	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

2.前端Vue项目的初始化

1. 在Student下新建一个StudentFE用来存储前端相关内容，然后在VScode中安装Vue 2 Snippets和Live Server两个扩展。



2. 新建一个index.html，然后新建三个文件夹用于存放图像、css、js文件。



3.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>学生信息管理系统</title>
  <!-- 引入外部的样式文件 -->
  <link rel="stylesheet" href="./css/index.css">
  <!-- 使用CDN引入Vue模块 -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2.7.16"></script>
</head>
<body>
  <div id="app">
    {{msg}}
  </div>

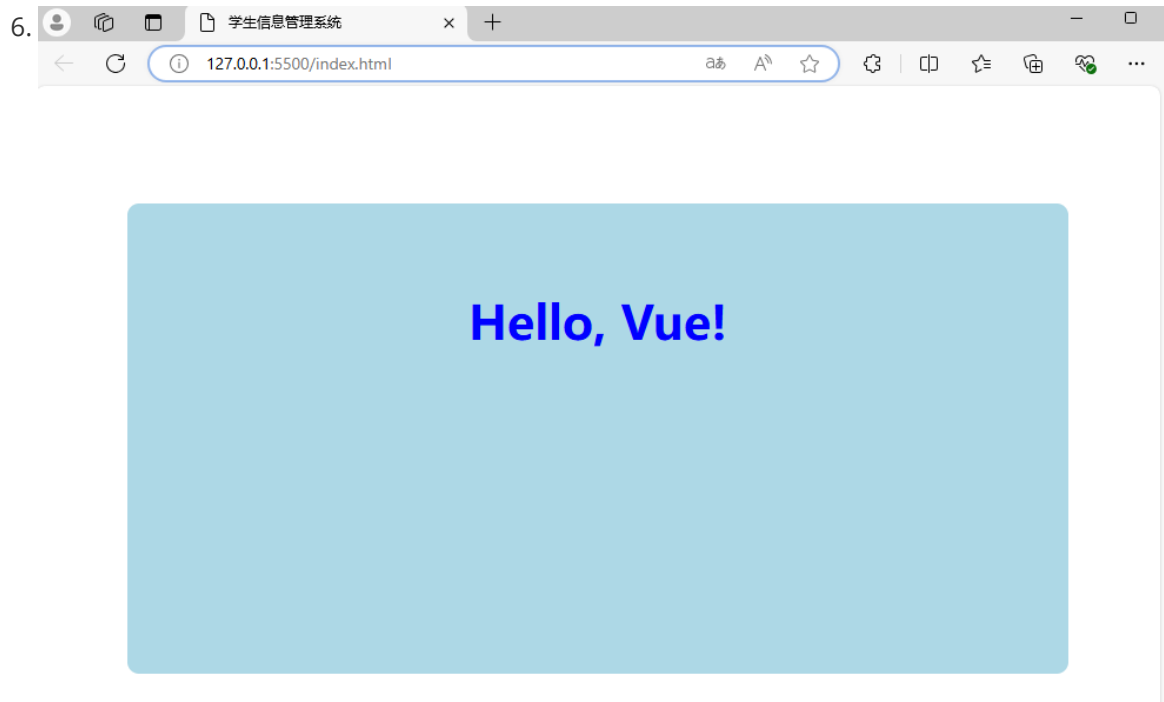
</body>
</html>
<!-- 引入Vue代码 -->
<script src="./js/index.js"></script>
```


4.

```
#app{
  width: 800px;
  height: 400px;
  background-color: lightblue;
  margin: 100px auto;
  font-size: 40px;
  font-weight: bold;
  line-height: 200px;
  text-align: center;
  border-radius: 10px;
  color: blue;
}
```

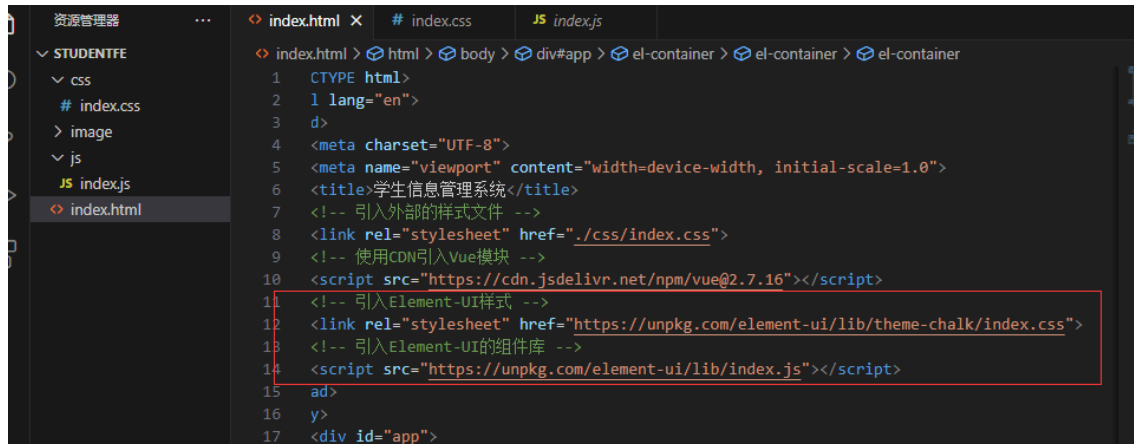
5.

```
const app = new Vue({
  el: "#app",
  data: {
    msg: "Hello, Vue!",
  }
})
```



3.引入Element UI和页面结构

1. 到element官网的组件模块找到CDN，引入样式和组件库，注意一定要放到Vue后面，不然不生效。



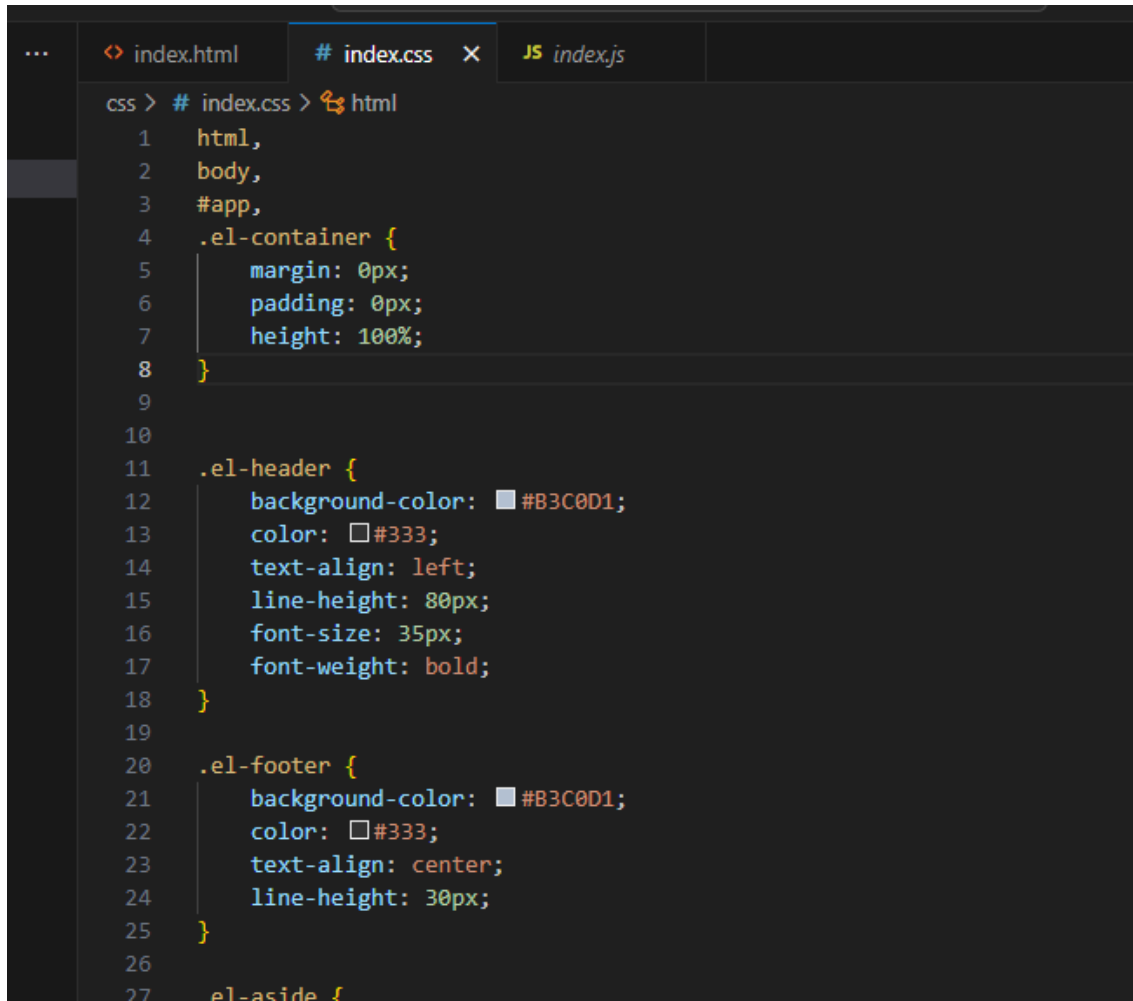
```
index.html > html > body > div#app > el-container > el-container > el-container
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>学生信息管理系统</title>
7    <!-- 引入外部的样式文件 -->
8    <link rel="stylesheet" href="./css/index.css">
9    <!-- 使用CDN引入Vue模块 -->
10   <script src="https://cdn.jsdelivr.net/npm/vue@2.7.16"></script>
11   <!-- 引入Element-UI样式 -->
12   <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
13   <!-- 引入Element-UI的组件库 -->
14   <script src="https://unpkg.com/element-ui/lib/index.js"></script>
15   <div id="app">
```

2. 然后找到组件的布局容器，找到一种合适的形式，把代码复制下来到body里面。



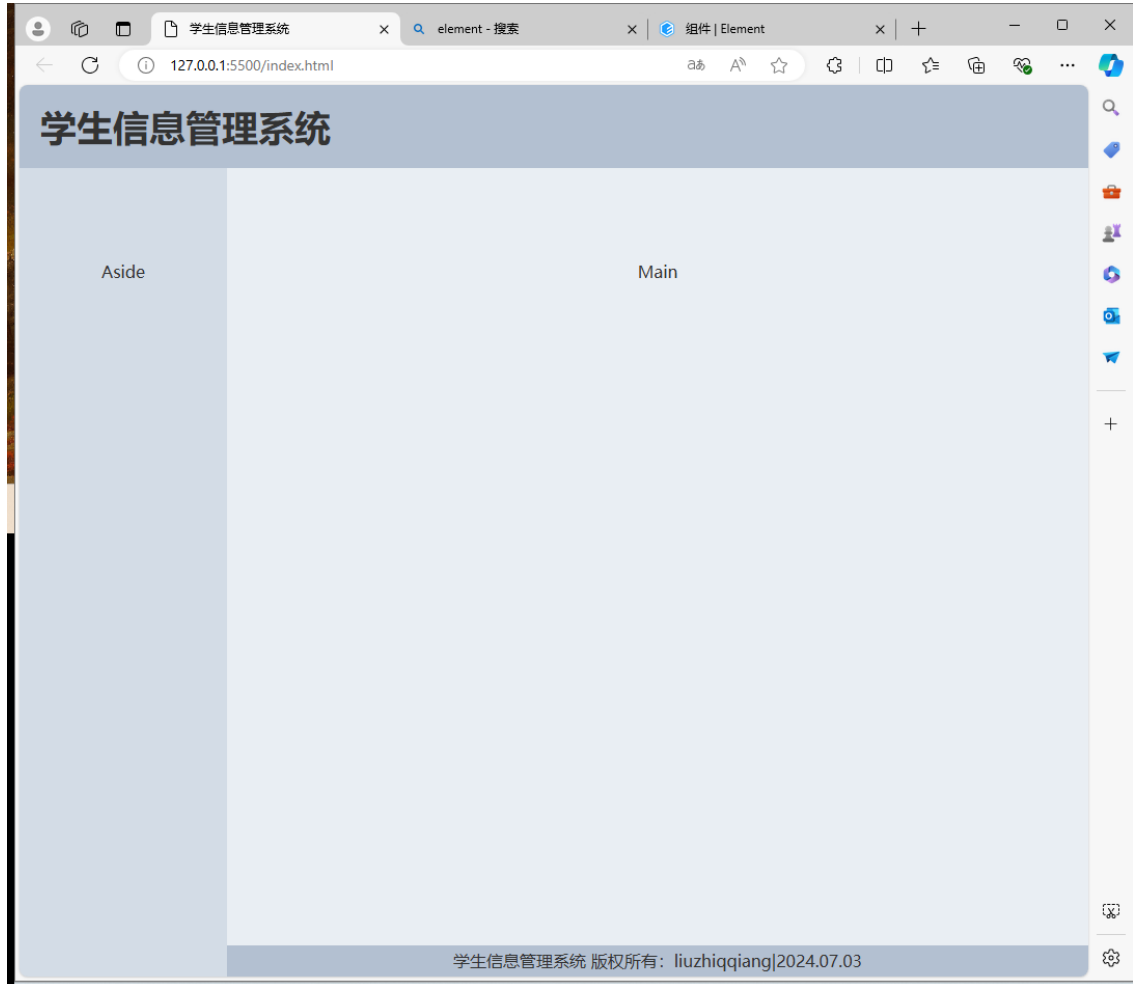
```
15 </head>
16 <body>
17   <div id="app">
18     <el-container>
19       <el-header style="height: 80px;">学生信息管理系统</el-header>
20       <el-container>
21         <el-aside width="200px">Aside</el-aside>
22         <el-container>
23           <el-main>Main</el-main>
24           <el-footer style="height: 30px;">学生信息管理系统 版权所有: liuzhiqqiang|20
25         </el-container>
26       </el-container>
27     </el-container>
28   </div>
29
30 </body>
```

3. 对应的，也要把样式都复制到css中，另外也可以添加一些其他的样式。保证页面占据整个浏览器。



```
css > # index.css > html
1  html,
2  body,
3  #app,
4  .el-container {
5      margin: 0px;
6      padding: 0px;
7      height: 100%;
8  }
9
10
11 .el-header {
12     background-color: #B3C0D1;
13     color: #333;
14     text-align: left;
15     line-height: 80px;
16     font-size: 35px;
17     font-weight: bold;
18 }
19
20 .el-footer {
21     background-color: #B3C0D1;
22     color: #333;
23     text-align: center;
24     line-height: 30px;
25 }
26
27 .el-aside {
```

4. 然后就编程这样了。



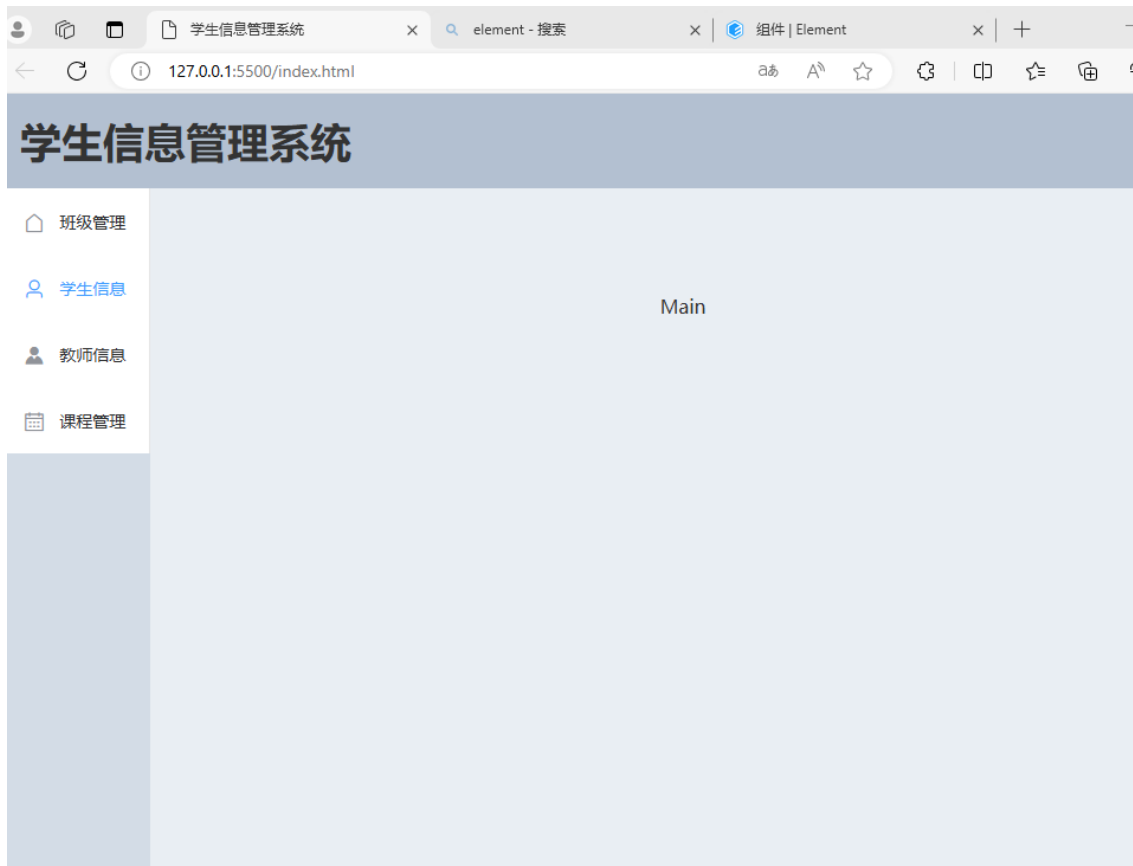
4.使用Element UI实现侧边栏

1. 找到组件中的导航菜单，选择一种合适的菜单的代码复制到index.html中的Aside中。可以自己修改一下，简化一下页面。

```
... < index.html x # index.css JS index.js
< index.html > html > body > div#app > el-container > el-container > el-aside
2 <html lang="en">
4 <head>
15 <script src="https://unpkg.com/element-ui/lib/index.js"></script>
16 </head>
17
18 <body>
19 <div id="app">
20 <el-container>
21 <el-header style="height: 80px;">学生信息管理系统</el-header>
22 <el-container>
23 <el-aside width="200px">
24 <el-menu default-active="1" class="el-menu-vertical-demo">
25 <el-menu-item index="1">
26 <i class="el-icon-menu"></i>
27 <span slot="title">导航1</span>
28 </el-menu-item>
29 <el-menu-item index="2">
30 <i class="el-icon-menu"></i>
31 <span slot="title">导航二</span>
32 </el-menu-item>
33 <el-menu-item index="3">
34 <i class="el-icon-document"></i>
35 <span slot="title">导航三</span>
36 </el-menu-item>
37 <el-menu-item index="4">
38 <i class="el-icon-setting"></i>
39 <span slot="title">导航四</span>
40 </el-menu-item>
41 </el-menu></el-aside>
42 <el-container>
43 <el-main>Main</el-main>
44 <el-footer style="height: 30px;">学生信息管理系统 版权所有:liuzhiqqian
45 </el-container>
```

2. 找到组件中的图标，把侧边栏的图标都改成合适的。

```
<el-header style="height: 80px;">学生信息管理系统</el-header>
<el-container>
  <el-aside width="130px">
    <el-menu default-active="1" class="el-menu-vertical-demo">
      <el-menu-item index="1">
        <i class="el-icon-house"></i>
        <span slot="title">班级管理</span>
      </el-menu-item>
      <el-menu-item index="2">
        <i class="el-icon-user"></i>
        <span slot="title">学生信息</span>
      </el-menu-item>
      <el-menu-item index="3">
        <i class="el-icon-s-custom"></i>
        <span slot="title">教师信息</span>
      </el-menu-item>
      <el-menu-item index="4">
        <i class="el-icon-date"></i>
        <span slot="title">课程管理</span>
      </el-menu-item>
    </el-menu></el-aside>
  <el-container>
    <el-main>Main</el-main>
```

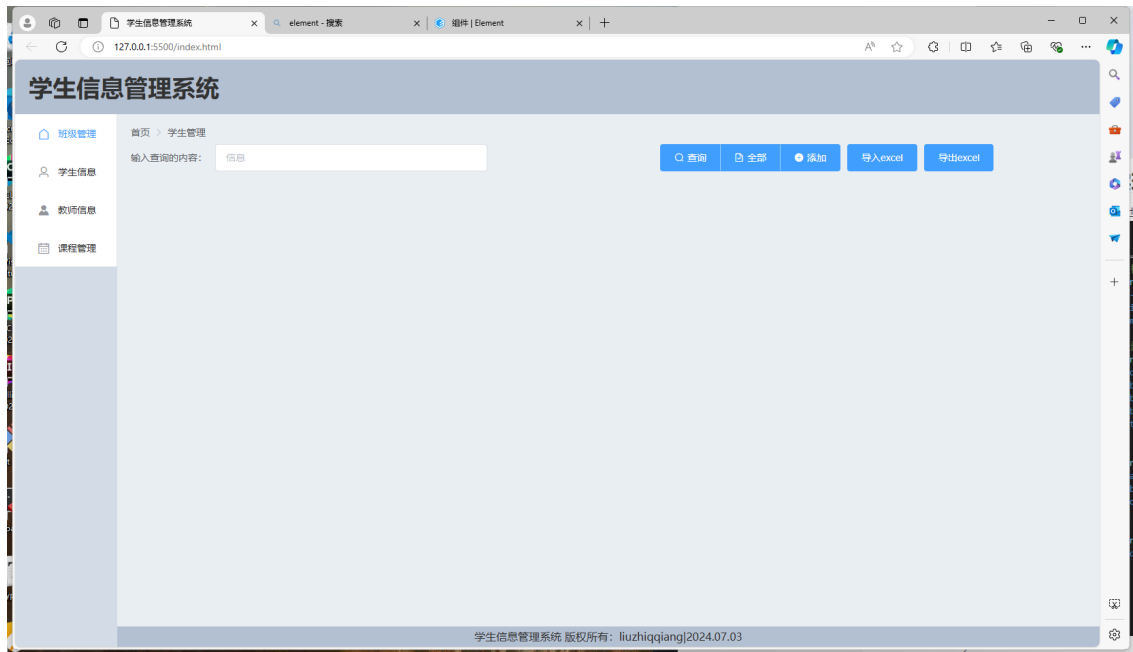


5.使用Element UI实现表单布局

1. 在页面中添加表单，相应的内容也去Element中找就行了。

```
53     </el-breadcrumb>
54     <!-- 表单 -->
55     <el-form :inline="true" style="margin-top: 10px;">
56       <!-- 查询输入框 -->
57       <el-col :span="10">
58         <el-form-item label="输入查询的内容：">
59           <el-input placeholder="信息" style="width: 400px;"></el-input>
60         </el-form-item>
61       </el-col>
62       <!-- 查询、全部、添加按钮 -->
63       <el-col :span="8" style="text-align: right; padding-right: 10px;">
64         <el-button-group>
65           <el-button type="primary" icon="el-icon-search">查询</el-button>
66           <el-button type="primary" icon="el-icon-document-checked">全部</el-button>
67           <el-button type="primary" icon="el-icon-circle-plus">添加</el-button>
68         </el-button-group>
69       </el-col>
70       <!-- 导入按钮 -->
71       <el-col :span="1.5" style="text-align: left;">
72         <el-upload>
73           <el-button type="primary">导入excel</el-button>
74         </el-upload>
75       </el-col>
76       <!-- 导出按钮 -->
77       <el-col :span="3" style="text-align: left; padding-left: 10px;">
78         <el-button type="primary">导出excel</el-button>
79       </el-col>
80     </el-row>
81   </el-form>
82 </el-main>
83
84
```

2.



6.使用Element UI中表格展示数据

1. 因为还未涉及到与后端的交互，因此先在Vue中的data中手动创建三个学生数据。

```
js > JS index.js > [?] app
1  const app = new Vue({
2    el: "#app",
3    data:{
4      msg: "Hello, Vue!",
5      students:[
6        {
7          sno:1, name:'刘一', gender:"男", birthday:"2000-01-01",mobile:'1380013800',
8          email:"zhangsan@example.com", address:"北京市朝阳区"
9        },
10       {
11         sno:2, name:'李四', gender:"女", birthday:"2000-02-02",mobile:'1390013900',
12         email:"lisi@example.com", address:"上海市浦东新区"
13       },
14       {
15         sno:3, name:'王五', gender:"男", birthday:"2000-03-03",mobile:'1360013600',
16         email:"wangwu@example.com", address:"广州市天河区"
17       },
18     ]
19   }
20 })
```

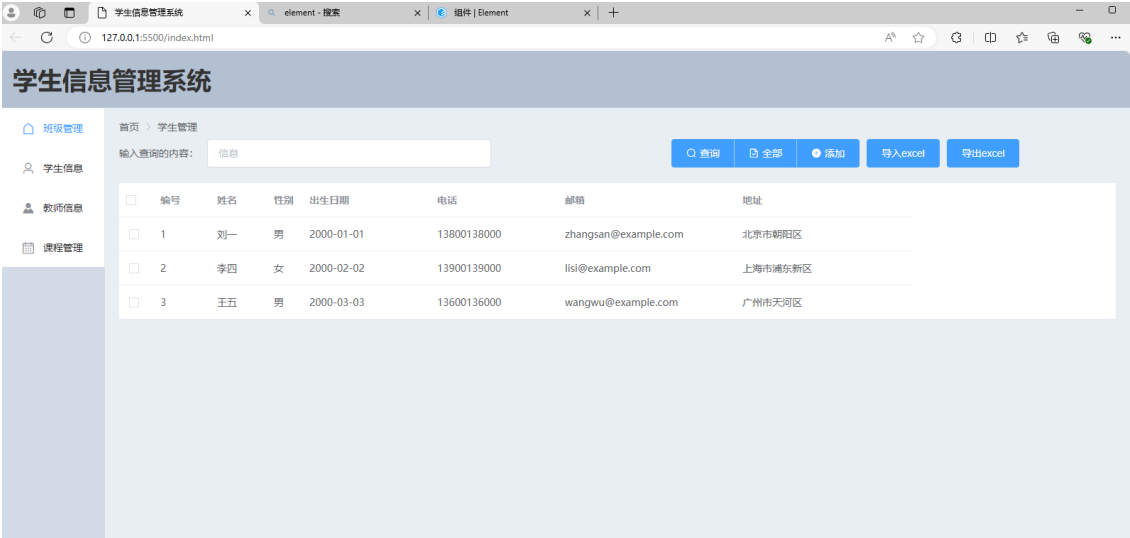
2. 添加表格和复选框。

```
<el-container>

  <!-- 表格 -->
  <el-table :data="students" style="width: 100%">
    <el-table-column type="selection">
    </el-table-column>
    <el-table-column prop="sno" label="编号" width="80">
    </el-table-column>
    <el-table-column prop="name" label="姓名" width="80">
    </el-table-column>
    <el-table-column prop="gender" label="性别" width="50">
    </el-table-column>
    <el-table-column prop="birthday" label="出生日期" width="180">
    </el-table-column>
    <el-table-column prop="mobile" label="电话" width="180">
    </el-table-column>
    <el-table-column prop="email" label="邮箱" width="250">
    </el-table-column>
    <el-table-column prop="address" label="地址" width="250">
    </el-table-column>
  </el-table>

</el-main>
```

3. 然后就能在网页中显示刚刚手动编辑的三条学生数据了。



4. 添加三个合适的按钮用于查看、编辑、删除信息。

```
<!-- 表格 -->
<el-table :data="students" style="width: 100%">
  <el-table-column type="selection" align="center">
  </el-table-column>
  <el-table-column prop="sno" label="编号" width="80" align="center">
  </el-table-column>
  <el-table-column prop="name" label="姓名" width="100" align="center">
  </el-table-column>
  <el-table-column prop="gender" label="性别" width="50" align="center">
  </el-table-column>
  <el-table-column prop="birthday" label="出生日期" width="120" align="center">
  </el-table-column>
  <el-table-column prop="mobile" label="电话" width="120" align="center">
  </el-table-column>
  <el-table-column prop="email" label="邮箱" width="300" align="center">
  </el-table-column>
  <el-table-column prop="address" label="地址" width="300" align="center">
  </el-table-column>
  <el-table-column label="操作" width="250" align="center">
    <el-button type="success" icon="el-icon-check" circle size="mini"></el-button>
    <el-button type="primary" icon="el-icon-edit" circle size="mini"></el-button>
    <el-button type="danger" icon="el-icon-delete" circle size="mini"></el-button>
  </el-table-column>
</el-table>
```

5.

学生信息管理系统

首页 > 学生管理

输入查询的内容: 信息

查询 全部 添加 导入excel 导出excel

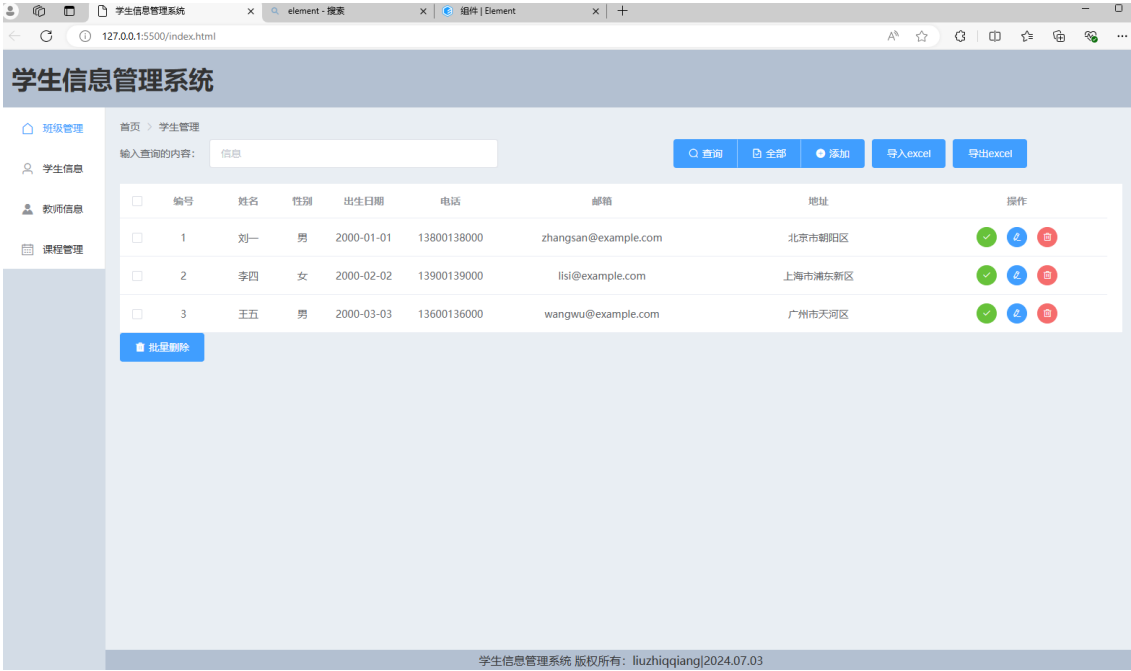
	编号	姓名	性别	出生日期	电话	邮箱	地址	操作
<input type="checkbox"/>	1	刘一	男	2000-01-01	13800138000	zhangsan@example.com	北京市朝阳区	✓ ✎ ✖
<input type="checkbox"/>	2	李四	女	2000-02-02	13900139000	lisi@example.com	上海市浦东新区	✓ ✎ ✖
<input type="checkbox"/>	3	王五	男	2000-03-03	13600136000	wangwu@example.com	广州市天河区	✓ ✎ ✖

7.使用Element UI实现分页展示

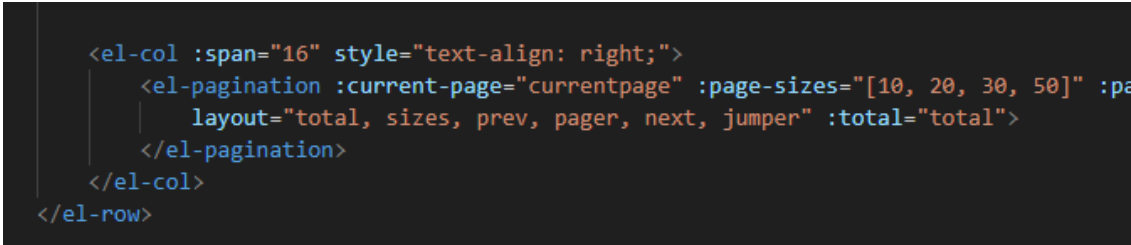
1. 设置分页展示框架，本质上是一行两个部分。

```
6 </el-table>
7
8 <!-- 分页展示 -->
9 <el-row>
10   <el-col :span="8" style="text-align: left;">
11     <el-button type="primary" icon="el-icon-delete-solid">批量删除
12   </el-col>
13
14   <el-col :span="16" style="text-align: right;">
15
16   </el-col>
17 </el-row>
18
19
20 </el-main>
```

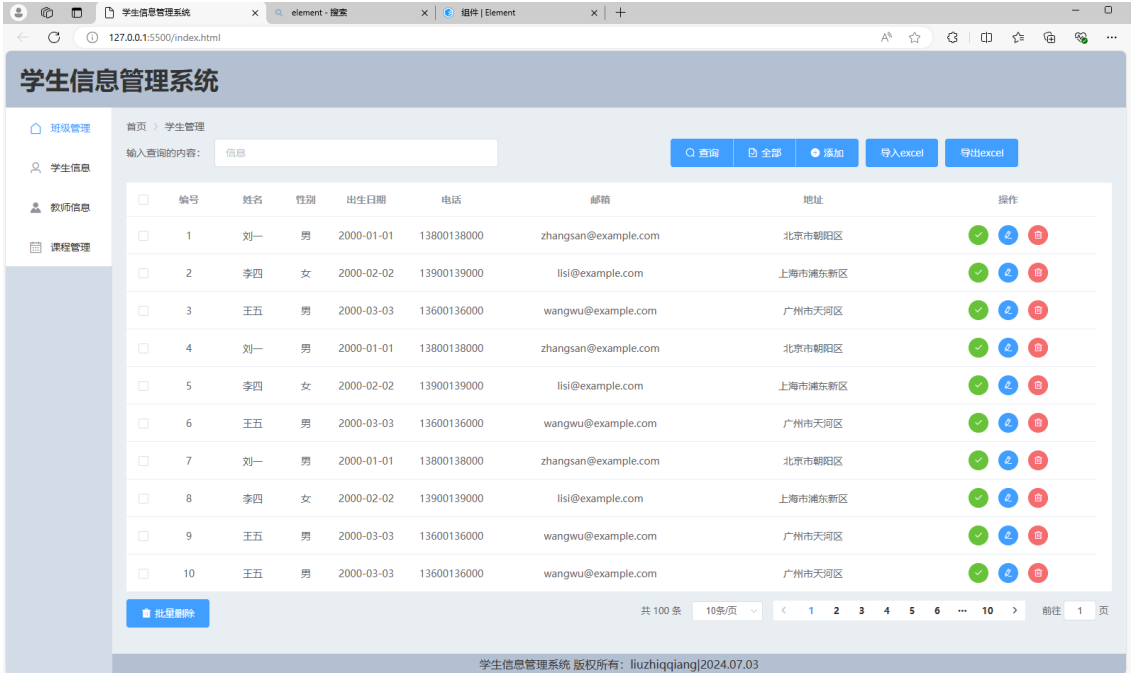
2. 结果：



3. 分页展示。

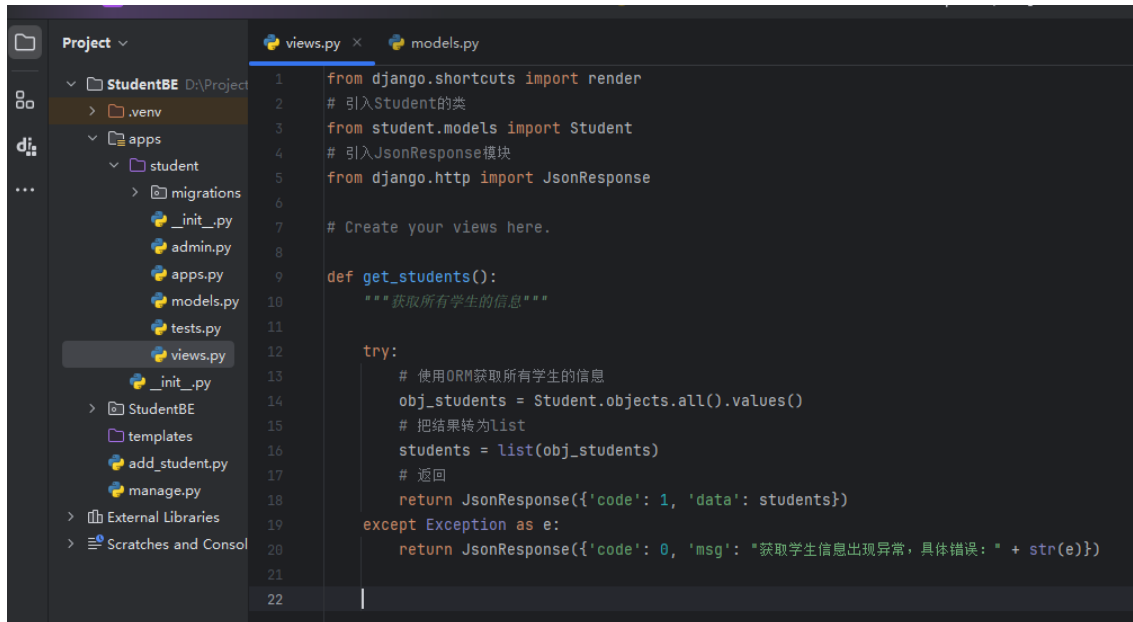


4. 结果：



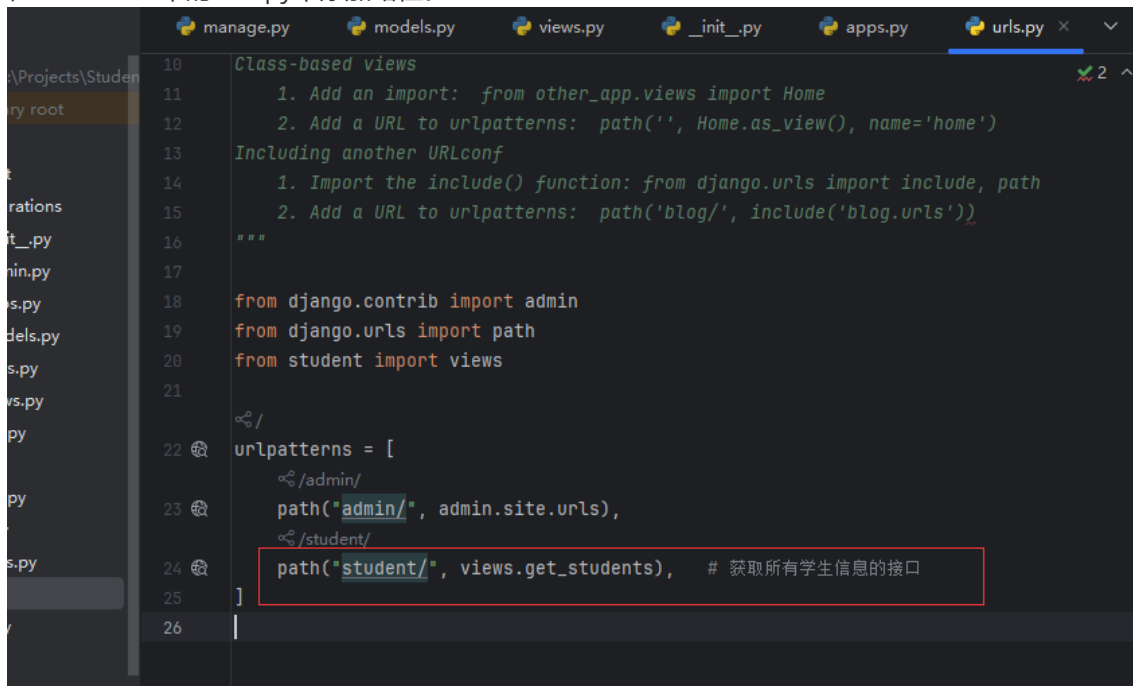
8.实现获取所有学生的后端接口

1. 在后端的student这个app中的views.py中创建获取学生信息的接口。



```
1 from django.shortcuts import render
2 # 引入Student的类
3 from student.models import Student
4 # 引入JsonResponse模块
5 from django.http import JsonResponse
6
7 # Create your views here.
8
9 def get_students():
10     """ 获取所有学生的信息 """
11
12     try:
13         # 使用ORM获取所有学生的信息
14         obj_students = Student.objects.all().values()
15         # 把结果转为list
16         students = list(obj_students)
17         # 返回
18         return JsonResponse({'code': 1, 'data': students})
19     except Exception as e:
20         return JsonResponse({'code': 0, 'msg': "获取学生信息出现异常, 具体错误: " + str(e)})
21
22
```

2. 在StudentBE中的urls.py中添加路径。



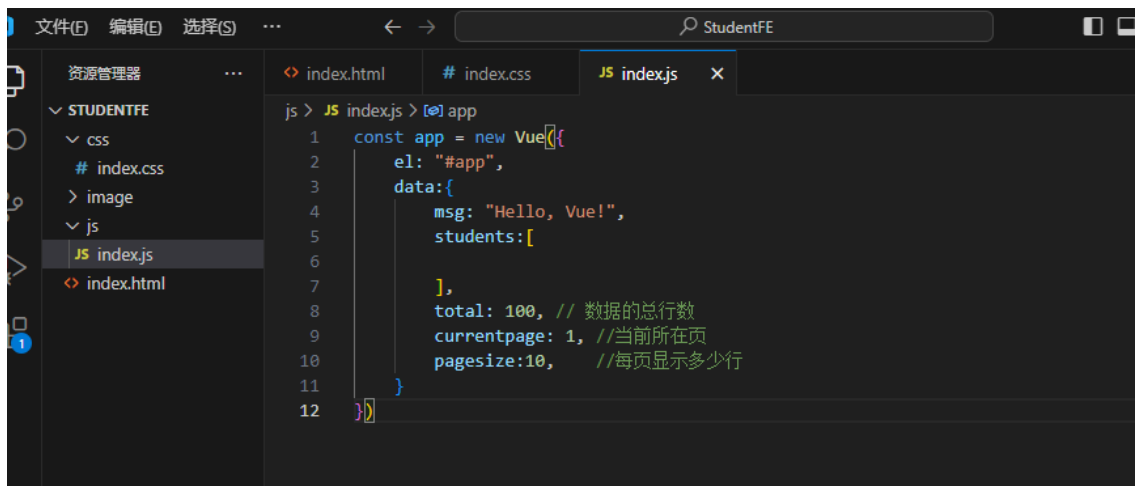
```
10 Class-based views
11 1. Add an import: from other_app.views import Home
12 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17
18 from django.contrib import admin
19 from django.urls import path
20 from student import views
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('student/', views.get_students), # 获取所有学生信息的接口
25 ]
26
```

3. 注意!!!!!!一定要将apps.py中的name改成student, 原本是apps.tudent, 因为前面把student这个app拉到apps下了, 并且apps已经被添加到默认查找源, 所以在所有地方都要改为student而不需要前缀apps.了。不然报错就是: RuntimeError: Model class student.models.Student doesn't declare an explicit app_label and isn't in an application in INSTALLED_APPS.
4. 修改好了过后在运行, python manage.py runserver 8080, 因为我默认的端口他说被占用了, 所以加了一个8080作为端口, 然后在浏览器中访问/student就可以看到数据库中查询的内容了。

```
1 {
2   "code": 1,
3   "data": [
4     {
5       "sno": 1,
6       "name": "张三",
7       "gender": "男",
8       "birthday": "2000-01-01",
9       "mobile": "13800138000",
10      "email": "zhangsan@example.com",
11      "address": "北京市朝阳区",
12      "image": null
13    },
14    {
15      "sno": 2,
16      "name": "李四",
17      "gender": "女",
18      "birthday": "2000-02-02",
19      "mobile": "13900139000",
20      "email": "lisi@example.com",
21      "address": "上海市浦东新区",
22      "image": null
23    },
24    {
25      "sno": 3,
26      "name": "王五",
27      "gender": "男",
28      "birthday": "2000-03-03",
29      "mobile": "13600136000",
30      "email": "wangwu@example.com",
31      "address": "广州市天河区",
32      "image": null
33    },
34    {
35      "sno": 4,
36      "name": "赵六",
37      "gender": "女",
38      "birthday": "2000-04-04",
39      "mobile": "13700137000",
40      "email": "zhaoliu@example.com",
41    }
42  ]
43 }
```

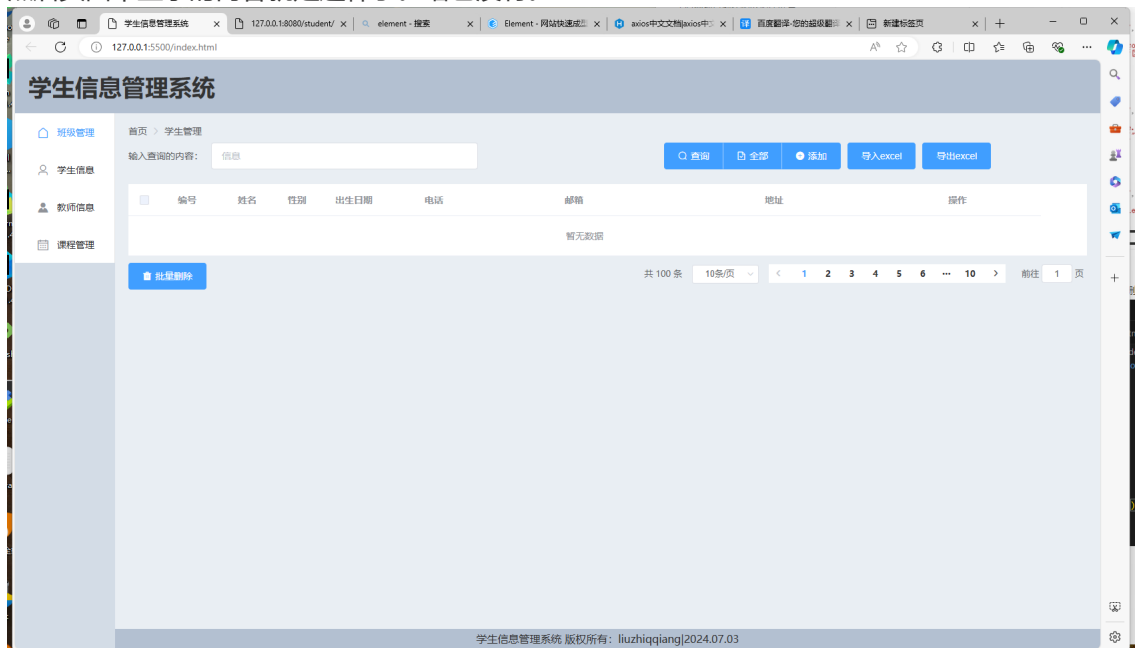
9.使用Axios请求后端接口的数据

1. 引入axios组件库。
2. 把原来前端Vue中的手动添加的数据都删除。

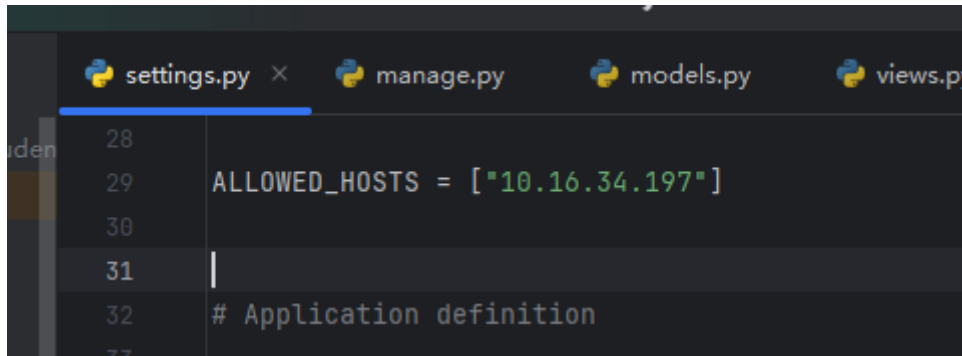


```
js > JS index.js > [0] app
1  const app = new Vue({
2    el: "#app",
3    data:{
4      msg: "Hello, Vue!",
5      students:[
6
7    ],
8    total: 100, // 数据的总行数
9    currentpage: 1, //当前所在页
10   pagesize:10, //每页显示多少行
11  },
12  })
```

3. 然后页面中显示的内容就是这样了。啥也没有。

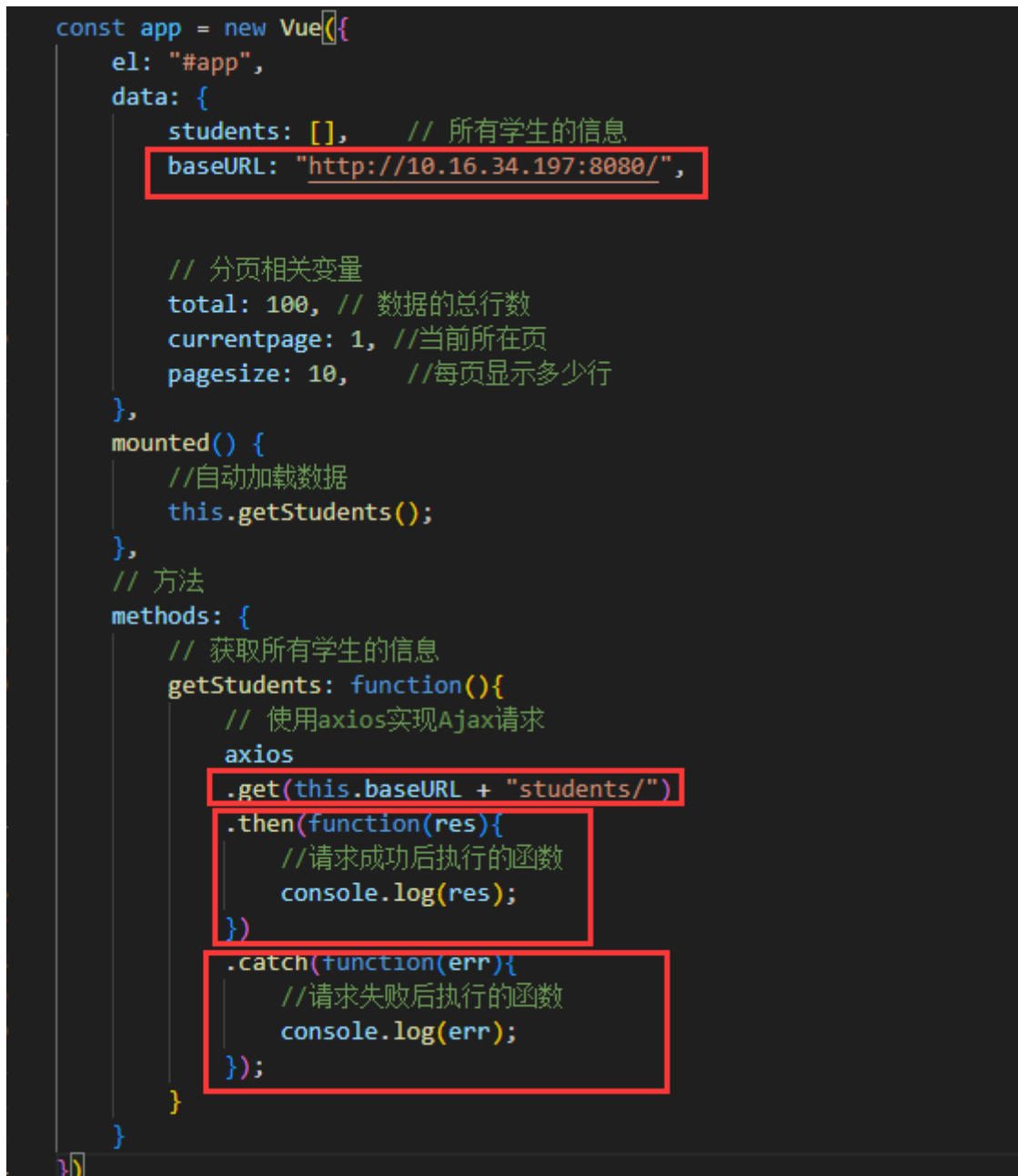


4. cmd中用ipconfig查看本机ip地址。目前开发是前后端都在一台主机上，所有默认地址都是127.0.0.1，但是如果以后要在不同机器上的话，就要配置一些相关内容，现在就弄好。



```
28
29 ALLOWED_HOSTS = ["10.16.34.197"]
30
31
32 # Application definition
```

5. 这样即使后端和前端不在一台主机上，前端机器也能随便访问后端了。运行的时候就用python manage.py runserver 10.16.34.197:8080就行了。但是貌似需要在同一局域网下。
6. 然后在前端Vue中添加上前面设置的网址。

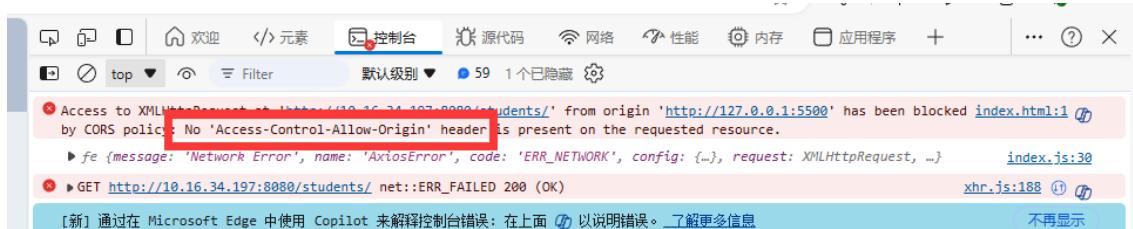


```
const app = new Vue({
  el: "#app",
  data: {
    students: [], // 所有学生的信息
    baseUrl: "http://10.16.34.197:8080/",

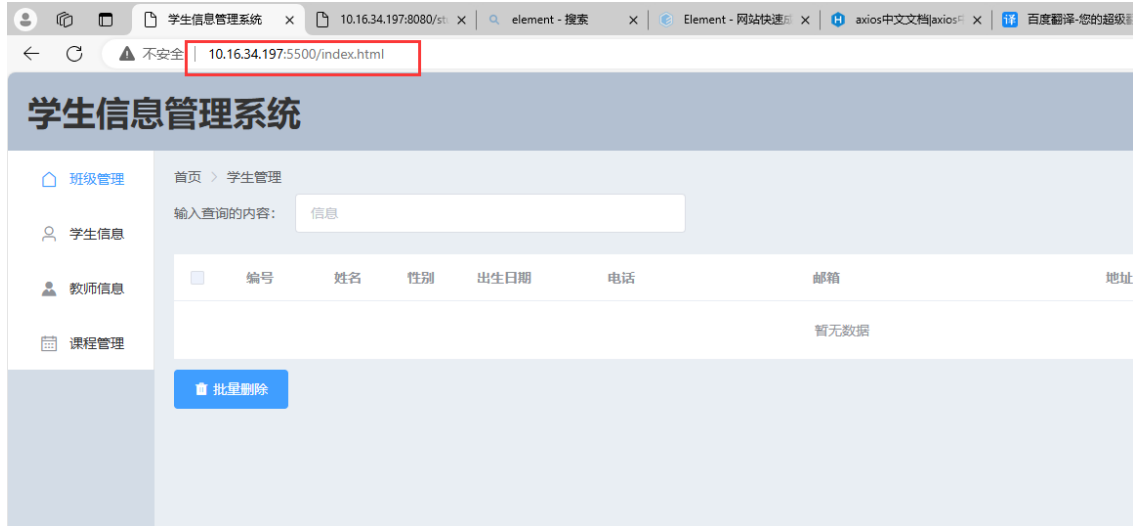
    // 分页相关变量
    total: 100, // 数据的总行数
    currentpage: 1, // 当前所在页
    pagesize: 10, // 每页显示多少行
  },
  mounted() {
    // 自动加载数据
    this.getStudents();
  },
  // 方法
  methods: {
    // 获取所有学生的信息
    getStudents: function(){
      // 使用axios实现Ajax请求
      axios
        .get(this.baseUrl + "students/")
        .then(function(res){
          // 请求成功后执行的函数
          console.log(res);
        })
        .catch(function(err){
          // 请求失败后执行的函数
          console.log(err);
        });
    }
  }
});
```

10.解决Ajax跨域请求

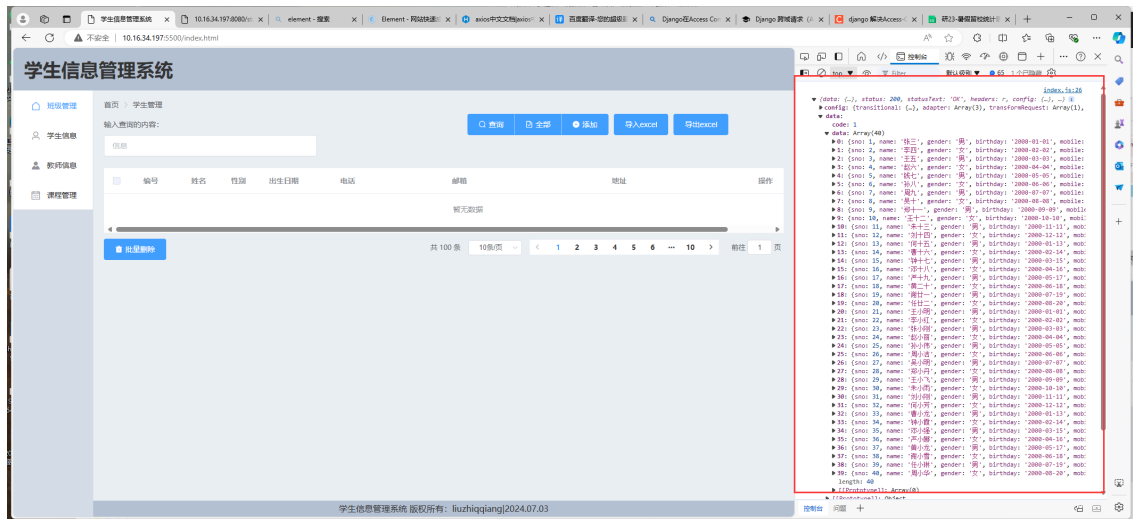
1. 但是刷新页面会发现跨域问题。



2. 后端解决跨域访问[Django 跨域请求 \(Access-Control-Allow-Origin\) | 极客教程 \(geek-docs.com\)](#)
3. 然后在前面网页刷新就不会报错了。
4. 前端也可以通过IP地址进行访问:



5. 刷新之后也能看到后端数据了。



11.把后端接口返回的数据展示在表格中

1. 完成getStudent函数的功能，然后添加成功或错误的提示信息。

```
js > JS index.js > app > methods > getStudents > then() callback > message
1  const app = new Vue({
18
19    // 方法
20    methods: {
21      // 获取所有学生的信息
22      getStudents: function(){
23        // 记录this的地址
24        let that = this
25        // 使用axios实现Ajax请求
26        axios
27        .get(this.baseUrl + "students/")
28        .then(function(res){
29          //请求成功后执行的函数
30          if (res.data.code == 1){
31            //把数据给students
32            that.students = res.data.data;
33            //提示
34            that.$message({
35              showClose: true,
36              message: '数据加载成功!',
37              type: 'success'
38            });
39          }
40          else{
41            that.$message({
42              showClose: true,
43              message: '数据加载错误!',
44              type: 'error'
45            });
46          }
47        })
48        .catch(function(err){
49          //请求失败后执行的函数
50          console.log(err);
51        });
52      }
53    }
54  })
```

2. 主要要用that这个变量把this的地址备份一下，不然后面再用this的话会被axios的this给覆盖。

学生信息管理系统

数据加载成功!

首页 > 班级管理

输入查询的内容: 信息

查询 全部 添加 导入excel 导出excel

<input type="checkbox"/>	编号	姓名	性别	出生日期	电话	邮箱	地址	操作
<input type="checkbox"/>	1	张三	男	2000-01-01	13800138000	zhangsan@example.com	北京市朝阳区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	2	李四	女	2000-02-02	13900139000	lisi@example.com	上海市浦东新区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	3	王五	男	2000-03-03	13600136000	wangwu@example.com	广州市天河区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	4	赵六	女	2000-04-04	13700137000	zhaoliu@example.com	深圳市南山区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	5	钱七	男	2000-05-05	13500135000	qianqi@example.com	成都市武侯区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	6	孙八	女	2000-06-06	13400134000	sunba@example.com	重庆市渝中区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	7	周九	男	2000-07-07	13300133000	zhoujiu@example.com	杭州市西湖区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	8	吴十	女	2000-08-08	13200132000	wushi@example.com	南京市玄武区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	9	郑十一	男	2000-09-09	13100131000	zhengshi@example.com	武汉市洪山区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	10	王十二	女	2000-10-10	13000130000	wangshier@example.com	西安市雁塔区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	11	朱十三	男	2000-11-11	13900139111	zhushisan@example.com	苏州市姑苏区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
<input type="checkbox"/>	12	刘十四	女	2000-12-12	13800138111	liushisi@example.com	青岛市市南区	<input checked="" type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>

- 3.

12. 实现表格展示数据的分页功能

1. 在getStudents中还要更新数据总行数、然后调用getPageStudents函数去更新当前页的数据。实现后记录如下。

```
2. const app = new Vue({
  el: "#app",
  data: {
    students: [],    // 所有学生的信息
    pageStudents: [], // 分页后当前页的学生信息

    baseUrl: "http://10.16.34.197:8080/",

    // 分页相关变量
    total: 0, // 数据的总行数
    currentpage: 1, // 当前所在页
    pagesize: 10, // 每页显示多少行
  },
  mounted() {
    // 自动加载数据
    this.getStudents();
  },

  // 方法
  methods: {
    // 获取所有学生的信息
    getStudents: function(){
      // 记录this的地址
      let that = this
      // 使用axios实现Ajax请求
      axios
        .get(this.baseUrl + "students/")
        .then(function(res){
          // 请求成功后执行的函数
          if (res.data.code == 1){
            // 把数据给students
            that.students = res.data.data;
            // 获取总行数
            that.total = that.students.length;
            // 获取当前页的数据
            that.getPageStudents();
            // 提示
            that.$message({
              showClose: true,
              message: '数据加载成功!',
              type: 'success'
            });
          }
          else{
            that.$message({
              showClose: true,
              message: '数据加载错误!',
              type: 'error'
            });
          }
        })
    }
  }
});
```



```

    })
    .catch(function(err){
        //请求失败后执行的函数
        console.log(err);
    });
},

//获取当前页的学生数据
getPageStudents: function(){
    //清空pageStudents中的数据
    this.pageStudents = [];
    //获得当前页的数据
    for(let i = (this.currentpage - 1) * this.pagesize; i <
this.total; i++){
        //把这些遍历到的数据添加到pageStudents中
        this.pageStudents.push(this.students[i]);
        if (this.pageStudents.length == this.pagesize)
            break;
    }
}
}
})

```

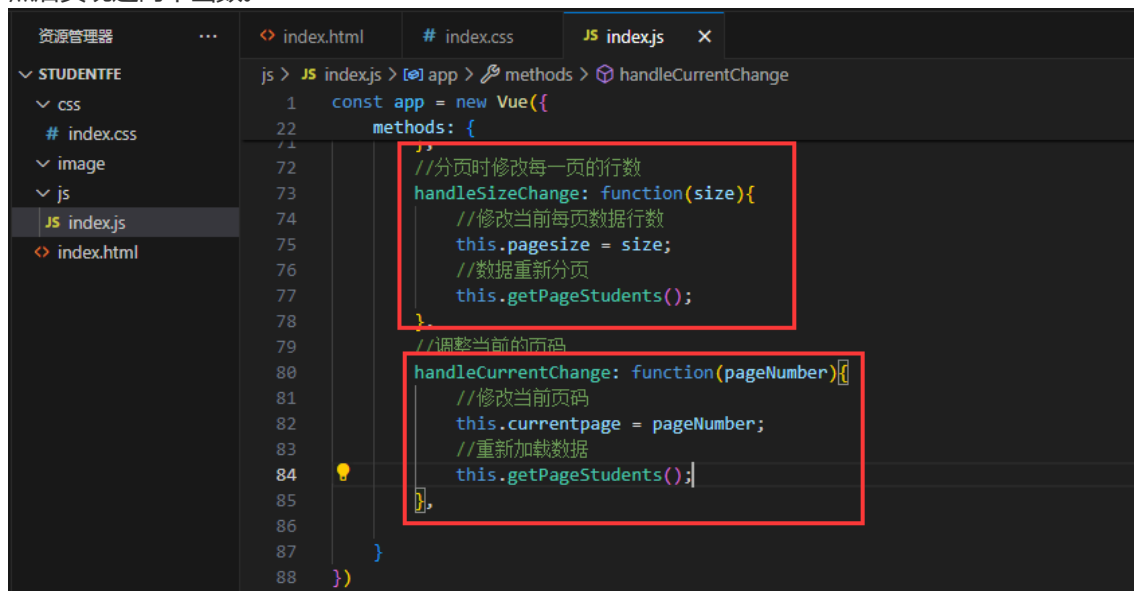
3. 这个时候前端就能正常只显示10条数据了，但是下面的分页选项没有任何作用，所以要把之前删掉的绑定给加上。

```

1  <!-- 分页展示 -->
2  <el-row style="padding-top: 10px;">
3      <el-col :span="8" style="text-align: left;">
4          <el-button type="primary" icon="el-icon-delete-solid">批量删除</el-button>
5      </el-col>
6
7      <el-col :span="16" style="text-align: right;">
8          <el-pagination @size-change="handleSizeChange"
9              @current-change="handleCurrentChange":current-page="currentpage" :
10              layout="total, sizes, prev, pager, next, jumper" :total="total">
11          </el-pagination>
12      </el-col>
13  </el-row>

```

4. 然后实现这两个函数。

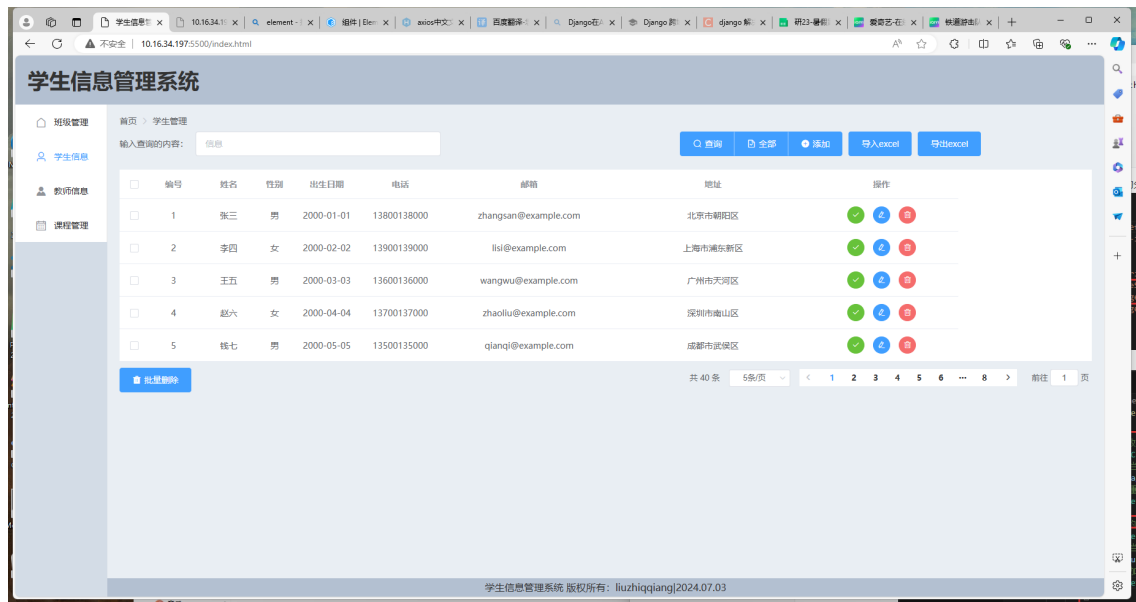


```

js > JS index.js > [app] > methods > handleCurrentChange
1  const app = new Vue({
22     methods: {
23         //分页时修改每一页的行数
24         handleSizeChange: function(size){
25             //修改当前每页数据行数
26             this.pagesize = size;
27             //数据重新分页
28             this.getPageStudents();
29         },
30         //调整当前的页码
31         handleCurrentChange: function(pageNumber){
32             //修改当前页码
33             this.currentpage = pageNumber;
34             //重新加载数据
35             this.getPageStudents();
36         },
37     },
38 })

```

5. 然后就一切ok了。



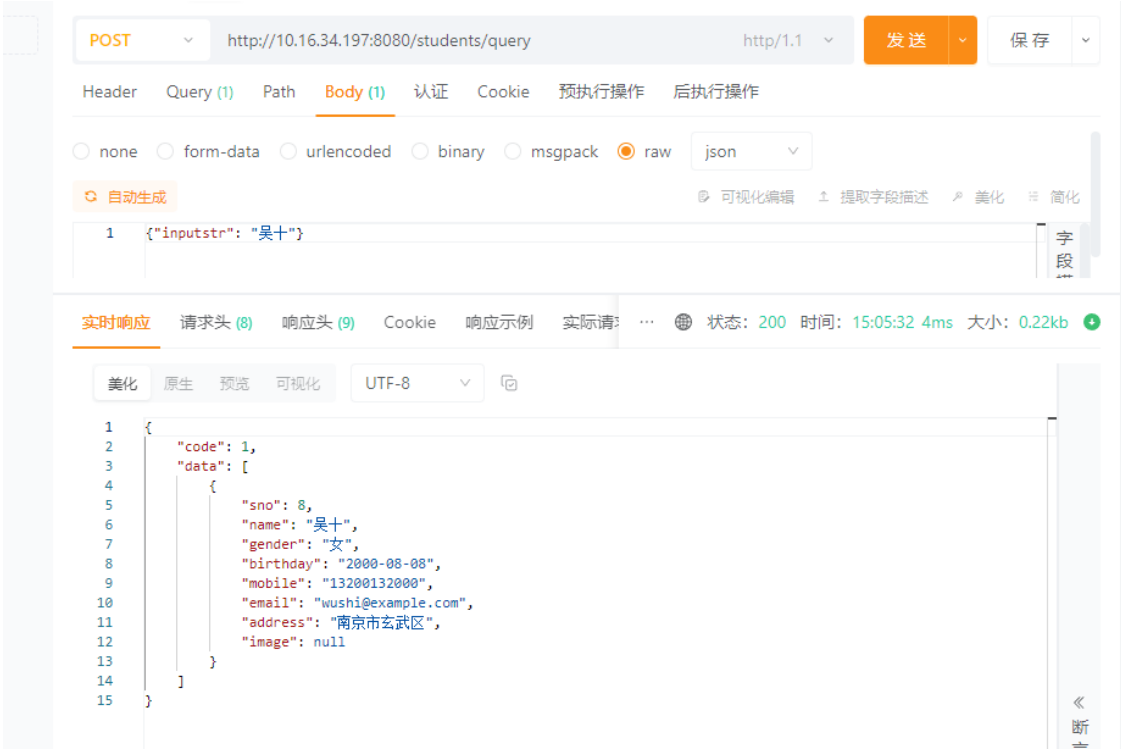
13.实现学生信息查询的后端接口

1. 实现查询函数query_students。

```
views.py | urls.py | settings.py
28 | /students/query 1 usage
29 | def query_students(request):
30 |     # 查询学生信息
31 |     # 获取所有学生的信息
32 |     # 接口传递过来的查询条件 Axios默认是json格式---字典类型 ('inputstr') --data['inputsre']
33 |     data = json.loads(request.body.decode('utf-8'))
34 |
35 |     try:
36 |         # 使用ORM获取满足条件的学生信息
37 |         obj_students = Student.objects.filter(Q(sno__icontains=data['inputstr']) | Q(name__ic
38 |             Q(gender__icontains=data['inputstr']) | Q(mobile__icontains=data['inputstr']) |
39 |             Q(email__icontains=data['inputstr']) | Q(address__icontains=data['inputstr'])).va
40 |
41 |         # 把结果转为List
42 |         students = list(obj_students)
43 |
44 |         # 返回
45 |         return JsonResponse({'code': 1, "data": students})
46 |     except Exception as e:
47 |         # 如果出现异常，返回
48 |         return JsonResponse({'code': 0, 'msg': "查询学生信息异常，具体错误: " + str(e)})
```

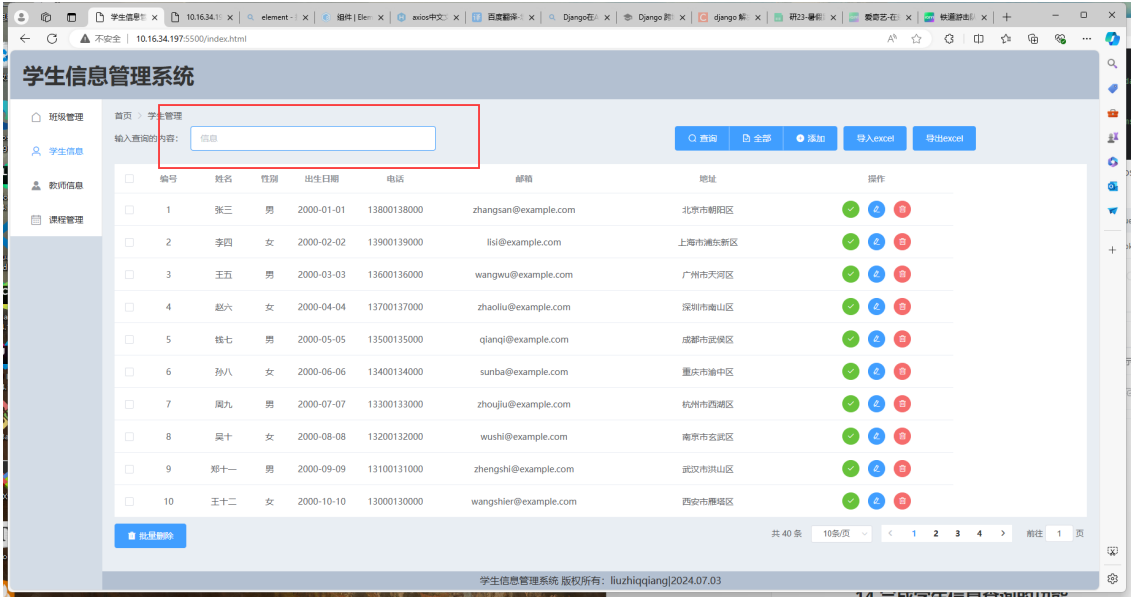
2. 然后下来一个apipost，运行起来后端服务后，去api中用post请求，在body的row中选择json格式，然后使用关键字模拟查询。

3. 点击发送就跨域模拟访问了。

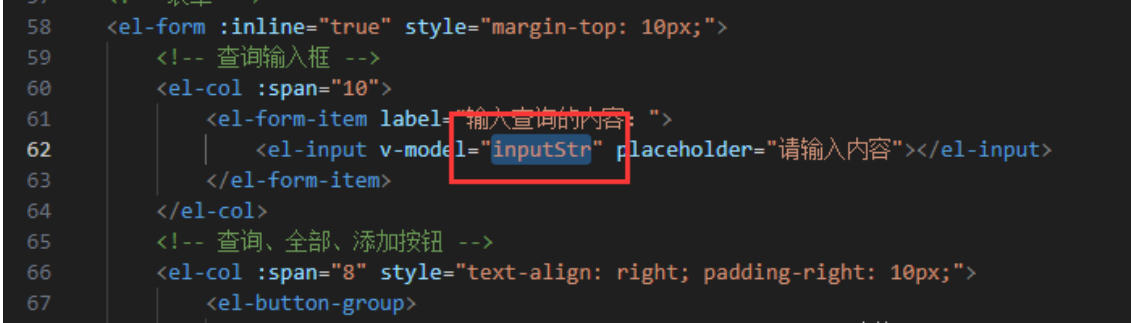


14.完成学生信息查询的功能

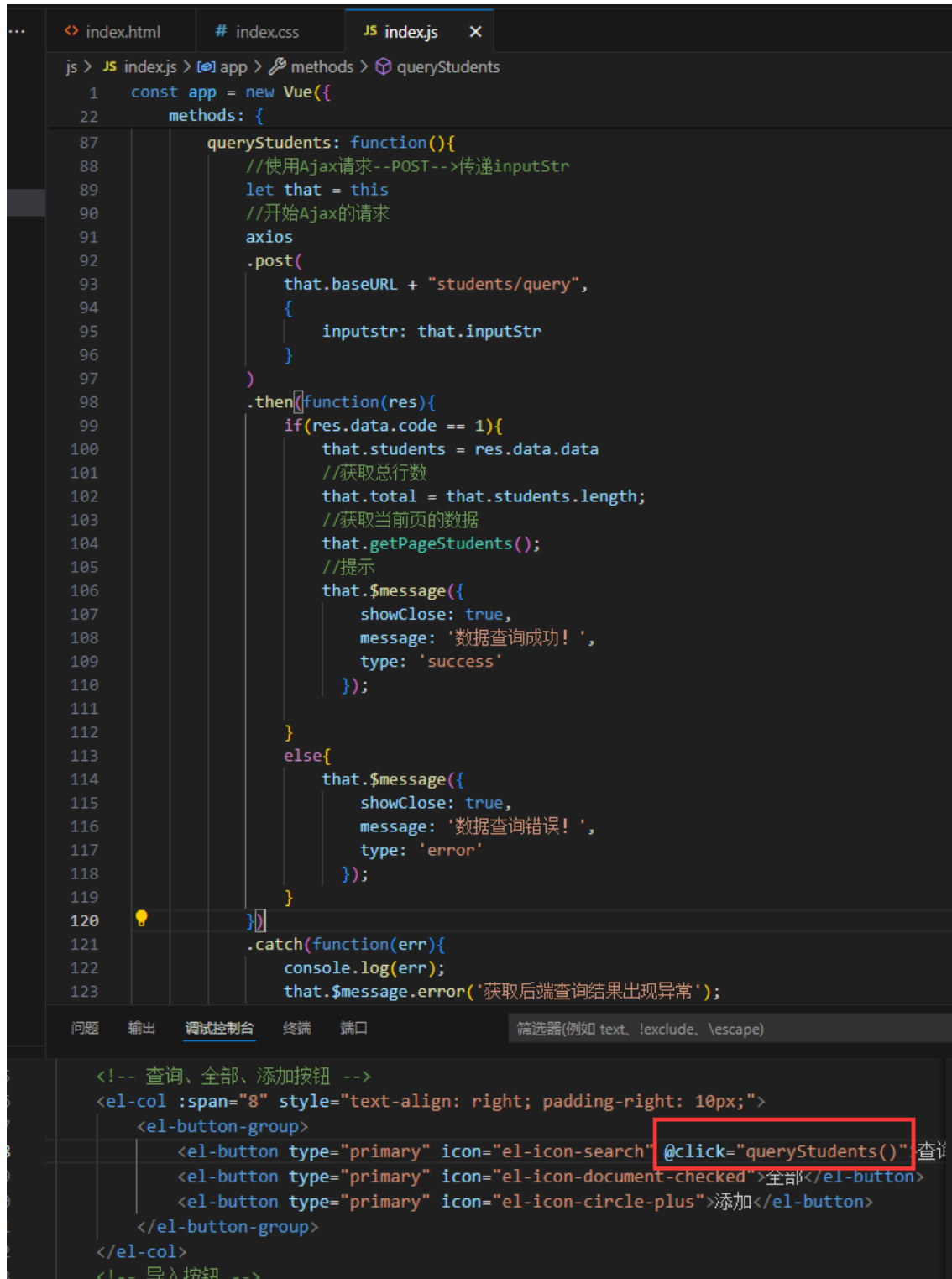
1. 现在在前端的输入框出入内容是不接受任何信息的。因为没有变量与它做捆绑。



2. 与Vue里面的inputStr变量绑定就可以了。



3. 实现查询函数，并且与查询按钮绑定。



The image shows a code editor with two files: index.html and index.js. The index.js file contains a Vue instance with a queryStudents method. The method uses axios to send a POST request to the backend. It then checks the response code and updates the UI accordingly. The index.html file shows a button with the @click event bound to the queryStudents method.

```
js > JS index.js > app > methods > queryStudents
1  const app = new Vue({
22  methods: {
87    queryStudents: function(){
88      //使用Ajax请求--POST-->传递inputStr
89      let that = this
90      //开始Ajax的请求
91      axios
92      .post(
93        that.baseUrl + "students/query",
94        {
95          inputstr: that.inputStr
96        }
97      )
98      .then(function(res){
99        if(res.data.code == 1){
100          that.students = res.data.data
101          //获取总行数
102          that.total = that.students.length;
103          //获取当前页的数据
104          that.getPageStudents();
105          //提示
106          that.$message({
107            showClose: true,
108            message: '数据查询成功!',
109            type: 'success'
110          });
111        }
112        else{
113          that.$message({
114            showClose: true,
115            message: '数据查询错误!',
116            type: 'error'
117          });
118        }
119      })
120    }
121    .catch(function(err){
122      console.log(err);
123      that.$message.error('获取后端查询结果出现异常');
124    })
125  }
126})
```

```
<!-- 查询、全部、添加按钮 -->
<el-col :span="8" style="text-align: right; padding-right: 10px;">
  <el-button-group>
    <el-button type="primary" icon="el-icon-search" @click="queryStudents()" 查询
    <el-button type="primary" icon="el-icon-document-checked">全部</el-button>
    <el-button type="primary" icon="el-icon-circle-plus">添加</el-button>
  </el-button-group>
</el-col>
<!-- 导入按钮 -->
```

4. 然后绑定全部按钮，因为全部按钮和getStudents的功能是一样的，所以直接绑定getStudents就可以了。

15. 学生管理系统增删改查的总体概述

- 增：增加
- 删：删除
- 改：修改
- 查：查看

16. 使用Element UI实现弹出框的表单

```

<el-dialog title="学生详细信息" :visible.sync="dialogVisible" width="50%"
align="center">

    <el-form ref="form" label-width="80px"

inline="true">

        <el-form-item label="学号" style="width: 40%;">
            <el-input></el-input>
        </el-form-item>
        <el-form-item label="姓名" style="width: 40%;">
            <el-input></el-input>
        </el-form-item>
        <el-form-item label="性别" style="width: 40%;">
            <el-select placeholder="请选择性别">
                <el-option label="男" value="男"></el-
option>

                <el-option label="女" value="女"></el-
option>

            </el-select>
        </el-form-item>
        <el-form-item label="日期">
            <el-date-picker type="date" placeholder="选择
日期">

            </el-date-picker>
        </el-form-item>
        <el-form-item label="手机号码" style="width:
40%;">

            <el-input></el-input>
        </el-form-item>
        <el-form-item label="邮箱地址" style="width:
40%;">

            <el-input></el-input>
        </el-form-item>
        <el-form-item label="家庭住址" style="width:
40%;">

            <el-input></el-input>
        </el-form-item>
        <el-form-item label="详细信息" style="width:
40%;">

            <el-input></el-input>
        </el-form-item>
        <div>
            <span>
                <el-button type="primary" >确 定</el-
button>

                <el-button type="info">取 消</el-button>
            </span>
        </div>
    </el-form>
</el-dialog>

```

17美化Element UI弹出框表单

```

<!-- 弹出框的学生明细表单 -->
    <el-dialog title="学生详细信息"
:visible.sync="dialogVisible" width="50%" :close-on-click-modal="false">

```

```

        <el-form inline="true" style="margin-left: 10px;"
label-width="100px" label-position="right" ref="form" label-width="80px"
inline="true">

            <el-form-item label="学号">
                <el-input suffix-icon="el-icon-edit"></el-
input>

            </el-form-item>
            <el-form-item label="姓名">
                <el-input suffix-icon="el-icon-edit"></el-
input>

            </el-form-item>
            <el-form-item label="性别">
                <el-select placeholder="请选择性别">
                    <el-option label="男" value="男"></el-
option>

                    <el-option label="女" value="女"></el-
option>

                </el-select>
            </el-form-item>
            <el-form-item label="日期">
                <el-date-picker type="date" placeholder="选择
日期">

                </el-date-picker>
            </el-form-item>
            <el-form-item label="手机号码">
                <el-input suffix-icon="el-icon-edit"></el-
input>

            </el-form-item>
            <el-form-item label="邮箱地址">
                <el-input suffix-icon="el-icon-edit"></el-
input>

            </el-form-item>
            <el-form-item label="家庭住址">
                <el-input suffix-icon="el-icon-edit"></el-
input>

            </el-form-item>
            <el-form-item label="详细信息">
                <el-input suffix-icon="el-icon-edit"></el-
input>

            </el-form-item>
            <div>
                <span>
                    <el-button type="primary" >确 定</el-
button>

                    <el-button type="info">取 消</el-button>
                </span>
            </div>
        </el-form>
    </el-dialog>

```

18.浅拷贝和深拷贝

19.三种状态加载同一个表单

- 添加：所有表单的条目为空

- 查看明细：所有表单加载当前行的数据，所有数据不能修改
- 修改：所有表单加载当前行的数据，学号不能修改

20.表单提交前的校验

21.从后端校验学号是否存在

1. 添加校验函数。

```
50  
51 def check_sno(request):  
52     # 判断要添加的学号是否存在  
53     data = json.loads(request.body.decode('utf-8'))  
54  
55     try:  
56         obj_student = Student.objects.filter(sno=data['sno'])  
57         if obj_student.count() == 1:  
58             return JsonResponse({'code': 1, 'exists': True})  
59         else:  
60             return JsonResponse({'code': 1, 'exists': False})  
61     except Exception as e:  
62         return JsonResponse({'code': 0, 'msg': "校验学号失败，具体原因: " + str(e)})  
63
```

2. 添加路径。

```
21  
22 urlpatterns = [  
23     path("admin/", admin.site.urls),  
24     path("students/", views.get_students), # 获取所有学生信息的接口  
25     path("students/query", views.query_students), # 查询学生信息的接口  
26     path("students/check_sno", views.check_sno), # 校验要添加的学号  
27 ]  
28
```

3. 在apipost里面访问，实现成功。



22.完成学号是否存在的校验


1. 把Vue中的data变成一个对象，然后实现自定义校验规则。

```
2     el: '#app',
3     data() {
4         // 校验学号是否存在
5         const rulesSno = (rule, value, callback) =>{
6             //使用Axios进行校验
7             axios.post(
8                 this.baseUrl + 'students/check_sno',
9                 {
10                     sno: value,
11                 }
12             )
13             .then((res)=>{
14                 //请求成功
15                 if(res.data.code === 1){
16                     if(res.data.exists){
17                         callback(new Error("学号已存在! "));
18                     }
19                     else{
20                         callback();
21                     }
22                 }
23                 //请求失败
24                 else{
25                     callback(new Error("校验学号后端出现异常"))
26                 }
27             })
28             .catch((err)=>{
29                 //如果请求失败,在后台打印
30                 console.log(err);
31             });
32         }
33
34         return {
35             students: [], // 所有学生的信息
36         }
37     }
38 }
```

2. 绑定

```
},
rules: {
  sno: [
    { required: true, message: "学号不能为空", trigger: 'blur' },
    { pattern: /^[9][5]\d{3}$/, message: "学号必须是95开头的5位数", trigger: 'blur' },
    { validator: rulesSno, trigger: 'blur' },
  ],
  name: [

```

3.  image-20240705103855760

23.表单的校验和重置


```

251     },
252     //提交学生的表单(添加、修改)
253     submitStudentForm(formName) {
254         this.$refs[formName].validate((valid) => {
255             if (valid) {
256                 alert('submit!');
257             } else {
258                 console.log('error submit!!');
259                 return false;
260             }
261         });
262     },
263
264 }
265 })

```

24.实现添加学生的后端接口

1. 改了一下sql语句，把学号姓名这些东西改了一下。

2. `INSERT INTO student (SNo, SName, Gender, Birthday, Mobile, Email, Address, Image)`
`VALUES`
 (2024001, '张三', '男', '2000-01-01', '13800138000', 'zhangsan@qq.com',
 '北京市朝阳区', NULL),
 (2024002, '李娜', '女', '2000-02-02', '13900139000', 'lina@163.com', '上海
 市浦东新区', NULL),
 (2024003, '王伟', '男', '2000-03-03', '13600136000', 'wangwei@126.com',
 '广州市天河区', NULL),
 (2024004, '赵丽', '女', '2000-04-04', '13700137000', 'zhaoli@gmail.com',
 '深圳市南山区', NULL),
 (2024005, '钱勇', '男', '2000-05-05', '13500135000', 'qianyong@qq.com',
 '成都市武侯区', NULL),
 (2024006, '孙娟', '女', '2000-06-06', '13400134000', 'sunjuan@163.com',
 '重庆市渝中区', NULL),
 (2024007, '周军', '男', '2000-07-07', '13300133000', 'zhoujun@gmail.com',
 '杭州市西湖区', NULL),
 (2024008, '吴萍', '女', '2000-08-08', '13200132000', 'wuping@qq.com', '南
 京市玄武区', NULL),
 (2024009, '郑华', '男', '2000-09-09', '13100131000', 'zhenghua@126.com',
 '武汉市洪山区', NULL),
 (2024010, '王强', '女', '2000-10-10', '13000130000',
 'wangqiang@gmail.com', '西安市雁塔区', NULL),
 (2024011, '朱明', '男', '2000-11-11', '13900139111', 'zhuming@qq.com', '苏
 州市姑苏区', NULL),
 (2024012, '刘婷', '女', '2000-12-12', '13800138111', 'liuting@163.com',
 '青岛市市南区', NULL),
 (2024013, '何浩', '男', '2000-01-13', '13700137111', 'hehao@gmail.com',
 '天津市和平区', NULL),
 (2024014, '曹洁', '女', '2000-02-14', '13600136111', 'caojie@qq.com', '长
 沙市芙蓉区', NULL),
 (2024015, '钟强', '男', '2000-03-15', '13500135111',
 'zhongqiang@gmail.com', '福州市鼓楼区', NULL),

(2024016, '邓丽', '女', '2000-04-16', '13400134111', 'dengli@163.com', '厦门市思明区', NULL),
(2024017, '严军', '男', '2000-05-17', '13300133111', 'yanjun@gmail.com', '西宁市城中区', NULL),
(2024018, '黄萍', '女', '2000-06-18', '13200132111', 'huangping@qq.com', '银川市兴庆区', NULL),
(2024019, '谢华', '男', '2000-07-19', '13100131111', 'xiehua@163.com', '兰州市城关区', NULL),
(2024020, '任娟', '女', '2000-08-20', '13000130111', 'renjuan@gmail.com', '呼和浩特市回民区', NULL),
(2024021, '王小明', '男', '2000-01-01', '13800138001', 'wangxiaoming@qq.com', '北京市海淀区', NULL),
(2024022, '李小红', '女', '2000-02-02', '13900139001', 'lixiaohong@163.com', '上海市静安区', NULL),
(2024023, '张小刚', '男', '2000-03-03', '13600136001', 'zhangxiaogang@gmail.com', '广州市天河区', NULL),
(2024024, '赵小丽', '女', '2000-04-04', '13700137001', 'zhaoxiaoli@qq.com', '深圳市福田区', NULL),
(2024025, '孙小伟', '男', '2000-05-05', '13500135001', 'sunxiaowei@163.com', '成都市锦江区', NULL),
(2024026, '周小洁', '女', '2000-06-06', '13400134001', 'zhouxiaojie@gmail.com', '重庆市渝中区', NULL),
(2024027, '吴小明', '男', '2000-07-07', '13300133001', 'wuxiaoming@qq.com', '杭州市西湖区', NULL),
(2024028, '郑小丹', '女', '2000-08-08', '13200132001', 'zhengxiaodan@163.com', '南京市玄武区', NULL),
(2024029, '王小飞', '男', '2000-09-09', '13100131001', 'wangxiaofei@gmail.com', '武汉市洪山区', NULL),
(2024030, '朱小雨', '女', '2000-10-10', '13000130001', 'zhuxiaoyu@qq.com', '西安市雁塔区', NULL),
(2024031, '刘小刚', '男', '2000-11-11', '13900139112', 'liuxiaogang@163.com', '苏州市姑苏区', NULL),
(2024032, '何小芳', '女', '2000-12-12', '13800138112', 'hexiaofang@gmail.com', '青岛市市南区', NULL),
(2024033, '曹小龙', '男', '2000-01-13', '13700137112', 'caoxiaolong@qq.com', '天津市和平区', NULL),
(2024034, '钟小霞', '女', '2000-02-14', '13600136112', 'zhongxiaoxia@163.com', '长沙市芙蓉区', NULL),
(2024035, '邓小强', '男', '2000-03-15', '13500135112', 'dengxiaoqiang@gmail.com', '福州市鼓楼区', NULL),
(2024036, '严小娜', '女', '2000-04-16', '13400134112', 'yanxiaona@qq.com', '厦门市思明区', NULL),
(2024037, '黄小龙', '男', '2000-05-17', '13300133112', 'huangxiaolong@163.com', '西宁市城中区', NULL),
(2024038, '谢小雪', '女', '2000-06-18', '13200132112', 'xiexiaoxue@gmail.com', '银川市兴庆区', NULL),
(2024039, '任小琳', '男', '2000-07-19', '13100131112', 'renxiaolin@qq.com', '兰州市城关区', NULL),
(2024040, '周小华', '女', '2000-08-20', '13000130112', 'zhouxiaohua@163.com', '呼和浩特市回民区', NULL);

3. 实现添加学生的后端函数。

```
views.py  models.py  urls.py  settings.py
59
60  def add_student(request):
61      # 添加学生到数据库
62      # 接收前端传递过来的值
63      data = json.loads(request.body.decode('utf-8'))
64      # 添加到数据库
65      try:
66          obj_student = Student(sno=data['sno'], name=data['name'], gender=data['gender'],
67                               birthday=data['birthday'], mobile=data['mobile'],
68                               email=data['email'], address=data['address'])
69          # 执行添加
70          obj_student.save()
71
72          # 肯定是要更新页面中的学生内容的。
73          # 使用ORM获取所有学生信息
74          obj_students = Student.objects.all().values()
75          # 把结果转为List
76          students = list(obj_students)
77          # 返回
78          return JsonResponse({'code': 1, 'data': students})
79      except Exception as e:
80          return JsonResponse({'code': 0, 'msg': '添加到数据库出现异常，具体原因: ' + str(e)})
81
82
```

4. apipost一下。



5. 发现已经加进去了。

SNo	SName	Gender	Birthday	Mobile	Email	Address	Image
2024027	吴小明	男	2000-07-07	13300133001	wuxiaoming@qq.com	杭州市西湖区	(NULL)
2024028	郑小丹	女	2000-08-08	13200132001	zhengxiaodan@163.com	南京市玄武区	(NULL)
2024029	王小飞	男	2000-09-09	13100131001	wangxiaofei@gmail.com	武汉市洪山区	(NULL)
2024030	朱小雨	女	2000-10-10	13000130001	zhuxiaoyu@qq.com	西安市雁塔区	(NULL)
2024031	刘小刚	男	2000-11-11	13900139112	liuxiaogang@163.com	苏州市姑苏区	(NULL)
2024032	何小芳	女	2000-12-12	13800138112	hexiaofang@gmail.com	青岛市市南区	(NULL)
2024033	曹小龙	男	2000-01-13	13700137112	caoxiaolong@qq.com	天津市和平区	(NULL)
2024034	钟小霞	女	2000-02-14	13600136112	zhongxiaoxia@163.com	长沙市芙蓉区	(NULL)
2024035	邓小强	男	2000-03-15	13500135112	dengxiaoliang@gmail.com	福州市鼓楼区	(NULL)
2024036	严小娜	女	2000-04-16	13400134112	yanxiana@qq.com	厦门市思明区	(NULL)
2024037	黄小龙	男	2000-05-17	13300133112	huangxiaolong@163.com	西宁市城东区	(NULL)
2024038	谢小雪	女	2000-06-18	13200132112	xiexiaoxue@gmail.com	银川市兴庆区	(NULL)
2024039	任小琳	男	2000-07-19	13100131112	renxiaolin@qq.com	兰州市城关区	(NULL)
2024040	周小华	女	2000-08-20	13000130112	zhouxiaohua@163.com	呼和浩特市回民区	(NULL)
2024041	刘晓	男	2001-05-24	15313907254	liuxiao@qq.com	上海市浦东新区	(NULL)
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

25. 完成学生添加的功能

1. 点击确定按钮的时候提交表单，然后调用提交表单函数，并且判断是属于修改添加，如果是添加那么再进一步调用submitAddStudent函数。

```

    },
    // 添加学生
    submitAddStudent() {
        // 定义that
        let that = this;
        // 执行Axios请求
        axios
            .post(that.baseUrl + 'students/add_student', that.studentForm)
            .then(res => {
                if (res.data.code === 1) {
                    //把数据给students
                    that.students = res.data.data;
                    //获取总行数
                    that.total = that.students.length;
                    //获取当前页的数据
                    that.getPageStudents();
                    //提示
                    that.$message({
                        showClose: true,
                        message: '数据添加成功!',
                        type: 'success'
                    });
                    //关闭窗口
                    that.closeDialogForm('studentForm');
                }
                else {
                    that.$message.error(res.data.msg);
                }
            })
            .catch(err => {
                console.log(err);
                that.$message.error('获取后端查询结果出现异常');
            })
    },
    // 修改学生

```

26.完成学生修改功能

1. 添加update_student函数

```
82 def update_student(request):
83     # 添加学生到数据库
84     # 接收前端传递过来的值
85     data = json.loads(request.body.decode('utf-8'))
86     # 添加到数据库
87     try:
88         # 先把原来的删掉，然后直接复制添加的部分是可以的，当然也可以一个一个修改。
89         # 先找到然后直接删除
90         obj_student = Student.objects.get(sno=data['sno'])
91         obj_student.delete()
92         # 然后添加
93         obj_student = Student(sno=data['sno'], name=data['name'], gender=data['gender'],
94                               birthday=data['birthday'], mobile=data['mobile'],
95                               email=data['email'], address=data['address'])
96         # 执行添加
97         obj_student.save()
98
99         # 肯定是要更新页面中的学生内容的。
100        # 使用ORM获取所有学生信息
101        obj_students = Student.objects.all().values()
102        # 把结果转为List
103        students = list(obj_students)
104        # 返回
105        return JsonResponse({'code': 1, "data": students})
106    except Exception as e:
107        return JsonResponse({'code': 0, 'msg': '修改数据库出现异常，具体原因: ' + str(e)})
```

2. 前端和增加学生信息的动作是一样的。只不过路径改变了。

```
97 const app = new Vue({
    methods: {
306 submitUpdateStudent() {
307     // 定义that
308     let that = this;
309     // 执行Axios请求
310     axios
311     .post(that.baseURL + 'students/update_student', that.studentForm)
312     .then(res => {
313         if (res.data.code === 1) {
314             //把数据给students
315             that.students = res.data.data;
316             //获取总行数
317             that.total = that.students.length;
318             //获取当前页的数据
319             that.getPageStudents();
320             //提示
321             that.$message({
322                 showClose: true,
323                 message: '数据修改成功!',
324                 type: 'success'
325             });
326             //关闭窗口
327             that.closeDialogForm('studentForm');
328         }
329         else {
330             that.$message.error(res.data.msg);
331         }
332     })
333     .catch(err => {
334         console.log(err);
335         that.$message.error('修改时获取后端查询结果出现异常');
336     })
337     },
338 },
339 })
```

3. 但是发现在修改的时候还会校验学号是否存在。

修改学生明细

* 学号

2024000

* 姓名

李晓雪

* 性别

女

* 出生日期

2020-06-24

* 手机号码

13982945612

* 邮箱地址

liujuan@163.com

* 家庭住址

上海市浦东新区

确定

取消

4. 在校验学号是否存在的时候，判断是否isEdit，如果是的话就直接callback。

```

1  el: "#app",
2  data() {
3    // 校验学号是否存在
4    const rulesSno = (rule, value, callback) => {
5      //使用Axios进行校验
6      if(this.isEdit)
7        callback();
8      axios.post(
9        this.baseUrl + 'students/check_sno',
10        {
11          sno: value,
12        }
13      )
14    }
15  }
16 }

```

27.完成学生的删除功能

1. 前面几个功能都实现之后，会发现后面的功能都非常简单。

2. 实现后端接口。

```

def delete_student(request):
    # 接收前端传递过来的值
    data = json.loads(request.body.decode('utf-8'))
    # 添加到数据库
    try:
        # 先找到然后直接删除
        obj_student = Student.objects.get(sno=data['sno'])
        obj_student.delete()

        # 肯定是要更新页面中的学生内容的。
        # 使用ORM获取所有学生信息
        obj_students = Student.objects.all().values()
        # 把结果转为List
        students = list(obj_students)
        # 返回
        return JsonResponse({'code': 1, "data": students})
    except Exception as e:
        return JsonResponse({'code': 0, 'msg': '删除数据库出现异常，具体原因: ' + str(e)})

```

3. 前端需要实现一个提示信息，并且传递的数据是一行的数据，代码很长，具体看实现。



28.完成学生批量删除的功能

1. 后端实现遍历删除。

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# @Time : 2024/4/1 14:00
# @Author : 王伟
# @File : delete_students.py
# @Software : PyCharm

from flask import request, jsonify
from flask_restful import Resource
from models import Student

class DeleteStudents(Resource):
    def delete(self):
        # 接收前端传递过来的值
        data = json.loads(request.body.decode('utf-8'))
        # 添加到数据库
        try:
            # 遍历每一个学生并删除
            for stu in data['students']:
                obj_student = Student.objects.get(sno=stu['sno'])
                obj_student.delete()

            # 肯定是要更新页面中的学生内容的。
            # 使用ORM获取所有学生信息
            obj_students = Student.objects.all().values()
            # 把结果转为List
            students = list(obj_students)
            # 返回
            return JsonResponse({'code': 1, 'data': students})

        except Exception as e:
            return JsonResponse({'code': 0, 'msg': '批量删除数据库出现异常，具体原因: ' + str(e)})
```

2. 前端需要实现复选框的绑定和用于post的函数，代码很乱，具体看代码。

29.使用Element完成表单中照片的显示

添加学生明细

* 学号

* 姓名

* 性别

请选择性别

* 出生日期

选择日期

* 手机号码

* 邮箱地址

* 家庭住址

确定

取消

30.照片上传、存储、展示的过程分析

存储通常有两种方法：

- 一种是把照片序列化成文本，把文本存储在数据库。这种方法只适用于较小的图片。
- 另一种是把照片存储到后端文件夹中，把路径存储在数据库中。

如何保证上传照片不重名：

- 方案一：以当前的日期时间+随机数命名。
- 方案二：使用python中uuid模块生成不相同的名字。

31.完成上传照片后端接口

步骤：

1. 配置media文件夹。
2. settings中设定media。
 - `MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')`
 - `MEDIA_URL = '/media/'`
3. 设置URL。

- ```
from django.conf import settings
from django.conf.urls.static import static
添加这行
urlpatterns += static(settings.MEDIA_URL, document_root =
settings.MEDIA_ROOT)
```

4. 完成接口函数代码。



1. 在后端创建一个文件夹media用于后期存放图片文件的。
2. 设置文件上传后保存的路径已经外部访问时的url。

```
settings.py x urls.py views.py admin.py
130
131 STATIC_URL = "static/"
132
133
134
135 # 设置上传文件的目录和外部访问的路径
136 MEDIA_ROOT = os.path.join(BASE_DIR, 'media/') # 有文件上传后就把文件存到这个路径
137 MEDIA_URL = '/media/' # 外部访问的时候通过这个url进去访问
138
139 # Default primary key field type
140 # https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field
141
```

3. 设置URL。

```
settings.py urls.py x views.py admin.py
19 from django.urls import path
20 from student import views
21 from django.conf import settings
22 from django.conf.urls.static import static
23
24 urlpatterns = [
25 path('/admin/', admin.site.urls),
26 path('/students/', views.get_students), # 获取所有学生信息的接口
27 path('/students/query', views.query_students), # 查询学生信息的接口
28 path('/students/check_sno', views.check_sno), # 校验要添加的学号的接口
29 path('/students/add_student', views.add_student), # 添加学生的接口
30 path('/students/update_student', views.add_student), # 添加学生的接口
31 path('/students/delete_student', views.delete_student), # 删除学生的接口
32 path('/students/delete_students', views.delete_students), # 删除学生的接口
33]
34
35 # 添加这行——允许所有的media文件被访问
36 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
37
```

#### 4. 使用uuid和hash实现唯一id。


```
1 usage
2
3 def get_unique_id():
4 # 获取uuid的随机数
5 uuid_val = uuid.uuid4()
6
7 # 获取uuid的随机数字字符串
8 uuid_str = str(uuid_val).encode('utf-8')
9
10 # 获取md5实例
11 md5 = hashlib.md5()
12
13 # 拿取uuid的md5摘要
14 md5.update(uuid_str)
15
16 # 返回固定长度的字符串
17 return md5.hexdigest()
```

#### 5. 实现上传图片的后端函数。

```
6 def upload_img(request):
7 """接收上传的文件"""
8
9 # 接收上传的文件—缓存中
10 receive_file = request.FILES.get('avatar')
11
12 # 判断，是否有文件
13 if not receive_file:
14 return JsonResponse({'code': 0, 'msg': '图片不存在! '})
15
16 # 获得一个唯一的名字: uuid + hash
17 new_name = get_unique_id()
18
19 # 准备写入的URL
20 file_path = os.path.join(settings.MEDIA_ROOT, new_name + os.path.splitext(receive_file)[1])
21
22 # 开始写入到本地磁盘
23 try:
24 f = open(file_path, 'wb')
25 # 多次写入
26 for i in receive_file.chunks():
27 f.write(i)
28 # 写完之后要关闭
29 f.close()
30 # 返回
31 return JsonResponse({'code': 1, 'name': new_name + os.path.splitext(receive_file)[1]})
32
33 except Exception as e:
34 return JsonResponse({'code': 0, 'msg': str(e)})
```

## 32.实现照片的上传和展示

修改学生明细



\* 学号

2024001

\* 姓名

张三

\* 性别

男

\* 出生日期

2000-01-01

\* 手机号码

13800138000

\* 邮箱地址

zhangsan@qq.com

\* 家庭住址

北京市朝阳区

确定

取消

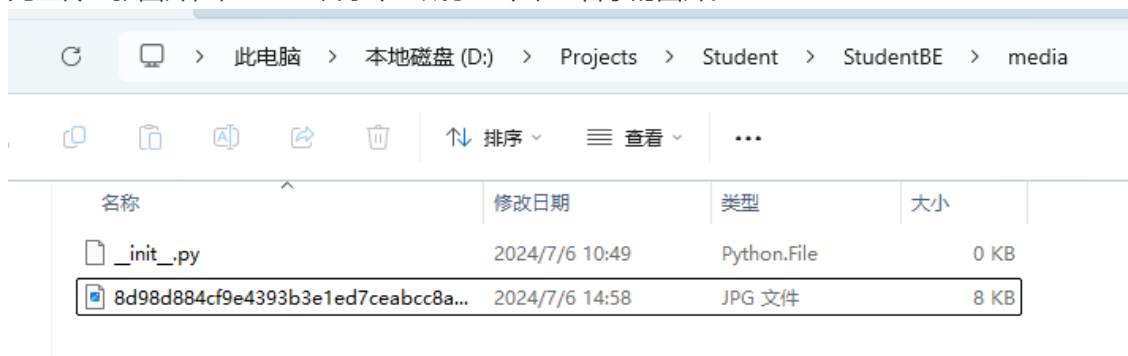
调试了老半天，不知道哪错了，但是上传成功了，后端也成功存储到了本地，就是前面样子hi不显示，所以应该是路径的问题，后面重写了一下那些路径就ok了。

### 33.实现照片存储在数据库

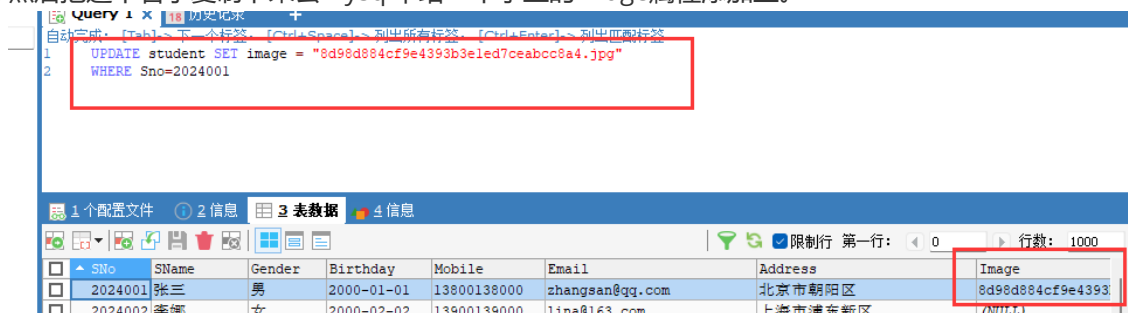
目前只是上传了，但是文件没有存到数据库，所以也就没办法保存，关闭之后再打开就没有了。

- 查看
- 添加
- 修改

1. 先上传一张图片，在media目录下生成了一个唯一名字的图片。



2. 然后把这个名字复制下来去mysql中给一个学生的image属性添加上。



.....完成了图片的查看、修改、添加。

## 34.读取Excel文件数据到Python

模块: openpyxl

1. 在student这个app中的tests.py中测试读取excel文件。

```
from django.test import TestCase

Create your tests here.
import openpyxl

def read_excel_dict(path: str):
 """读取Excel的数据，存储为字典---[{}, {}, {}...]"""
 # 实例化一个workbook
 workbook = openpyxl.load_workbook(path)
 # 实例化一个sheet
 sheet = workbook['students']
 # 定义一个变量存储最终的数据--[]
 students = []
 # 准备key
 keys = ['sno', 'name', 'gender', 'birthday', 'mobile', 'email', 'address',
 'image']
 # 遍历
 for row in sheet.rows:
 # 定义一个临时的字典
 temp_dict = dict()
 # 组合值和key
 for index, cell in enumerate(row):
 # 组合
 temp_dict[keys[index]] = cell.value
 # 附件到list中
 students.append(temp_dict)
 return students

if __name__ == '__main__':
 path = 'D:/Projects/Student/students.xlsx'
 students = read_excel_dict(path)
 # 输出
 print(students)
```

## 35.实现Excel批量导入数据的后端接口

有两种实现的方法：

- 在前端读取后，把读取的数据提交给后端存储
- 在前端直接把文件给后端，后端负责读取、存储到数据库

```
def read_excel_dict(path: str):
 """读取Excel的数据，存储为字典---[{}, {}, {}...]"""
 # 实例化一个workbook
```

```

workbook = openpyxl.load_workbook(path)
实例化一个sheet
sheet = workbook['students']
定义一个变量存储最终的数据--[]
students = []
准备key
keys = ['sno', 'name', 'gender', 'birthday', 'mobile', 'email', 'address',
'image']
遍历
for row in sheet.rows:
 # 定义一个临时的字典
 temp_dict = dict()
 # 组合值和key
 for index, cell in enumerate(row):
 # 组合
 temp_dict[keys[index]] = cell.value
 # 附件到list中
 students.append(temp_dict)
return students

def import_students_excel(request):
 """从Excel批量导入学生信息"""
 # =====1.接收Excel文件存储到Media文件夹
 receive_file = request.FILES.get('excel')
 if not receive_file:
 return JsonResponse({'code':0, 'msg':'Excel文件不存在! '})

 try:
 # 这里我的实现是文件名都统一改为excel_for_students。
 # 这样后面再上传文件的时候就直接把前面的覆盖了，不然以后会有特别多的文件。
 new_name = 'excel_for_students'
 file_path = os.path.join(settings.MEDIA_ROOT, new_name +
os.path.splitext(receive_file.name)[1])
 f = open(file_path, 'wb')
 # 多次写入
 for i in receive_file.chunks():
 f.write(i)
 # 要关闭
 f.close()

 except Exception as e:
 return JsonResponse({'code':0, 'msg':str(e)})

 # =====2.读取存储在Media文件夹的数据
 students = read_excel_dict(file_path)
 # =====3.把读取的数据存储到数据库
 # 定义几个变量: success, error, errors
 success, error, errors = 0, 0, []

 # 开始遍历
 for stu in students:
 try:
 # keys = ['sno', 'name', 'gender', 'birthday', 'mobile', 'email',
 'address', 'image']

```

```

 obj_student = Student(sno=stu['sno'], name=stu['name'],
gender=stu['gender'],
 birthday=stu['birthday'],
mobile=stu['mobile'],
 email=stu['email'], address=stu['address'],
image='041abd5bcf337655efb2443e5ae57a94.jpg')
 obj_student.save()
 # 计数
 success += 1

 except:
 # 如果失败了
 error += 1
 errors.append(stu['sno'])

4. 返回--导入信息（成功：5， 失败4--（sno）），所有学生
obj_students = Student.objects.all().values()
students = list(obj_students)
return JsonResponse({'code': 1, 'success': success, 'error': error,
'errors': errors, 'data': students})

```

## 36.完成Excel批量导入的功能

1. 完成上传后还要注意，没有对学号查重。
2. 对学号进行查重。

## 37.完成导出到Excel的功能

- 在后端数据库读取数据
- 导出到Media文件夹 (\*.xlsx)
- 返回这个文件的URL
- 前端Window.open(url)即可下载