《软件工程》

CodeAC队伍课程设计

**姓名：**罗格、刘诗法、刘修远、付俊博

**班级：**21级计算机基地班

**指导老师：**徐玉生

**日期：**2024/4/23

**目录**

[**一、文档简介 3**](#_Toc164973701)

[**1.1 产品目标 3**](#_Toc164973702)

[**1.2 产品范围 3**](#_Toc164973703)

[**1.3 术语和缩略词 3**](#_Toc164973704)

[**1.4 参考资料 3**](#_Toc164973705)

[**1.5 软件需求规格书其余部分 3**](#_Toc164973706)

[**二、产品概述 3**](#_Toc164973707)

[**2.1 产品背景 3**](#_Toc164973708)

[**2.2 产品功能 3**](#_Toc164973709)

[**2.3 用户特征 3**](#_Toc164973710)

[**2.4 约束条件 3**](#_Toc164973711)

[**2.4.1 设计约束 3**](#_Toc164973712)

[**2.4.2 过程约束 3**](#_Toc164973713)

[**三、具体需求 3**](#_Toc164973714)

[**3.1 外部接口要求 3**](#_Toc164973715)

[**3.1.1 用户界面 4**](#_Toc164973716)

[**3.1.2 硬件接口 4**](#_Toc164973717)

[**3.1.3 软件接口 4**](#_Toc164973718)

[**3.1.4 通信接口 4**](#_Toc164973719)

[**3.2 数据定义 4**](#_Toc164973720)

[**3.2.1 ER图 4**](#_Toc164973721)

[**3.2.2 数据流图 4**](#_Toc164973722)

[**3.3 质量需求 4**](#_Toc164973723)

[**3.3.1 兼容性 4**](#_Toc164973724)

[**3.3.2 安全性 4**](#_Toc164973725)

[**3.3.3 维护性 4**](#_Toc164973726)

[**3.3.4 可扩展性 4**](#_Toc164973727)

[**四、附录 4**](#_Toc164973728)

# 一、文档简介

## 1.1 产品目标

该产品旨在为用户提供一种便捷的在线方式，以可视化图像的形式展现多个气象站的不同气象数据，使用户能够轻松查阅特定时间段内的各项气象信息，如温度、风速和湿度等。同时，还赋予了管理员在后台管理权限，以便他们能灵活地对用户数据及各类气象数据进行添加、删除、修改和查询等操作。

## 1.2 产品范围

此软件工程课程设计项目是由我们小组合作完成的环境监测站数据管理系统，该系统基于Django框架构建，实现了前台和后台的综合功能。前台为普通访客提供了地图展示、站点信息浏览以及数据可视化服务，用户可通过地图交互获取站点详细数据，数据显示采用了Echarts库生成统计分析图表，并具备移动端数据浏览功能。后台则为管理员提供了用户数据和气象数据的CURD（增删改查）操作界面，以及用户账号的管理功能。

系统设计满足了基本要求，包括数据统计分析、地图集成、分布式部署（数据存储在数据库服务器，程序运行在云服务器），本地版本采用了Sqlite作为默认数据库，同时也支持华为云存储和其他数据库选项如MongoDB、Redis等。

系统特色在于良好的项目结构，通过封装数据库访问操作，确保了代码的模块化和易维护性；前端页面运用Bootstrap表格样式展示站点信息，地图展示利用JavaScript实现交互性，数据展示页面采用了轮播图设计，增强了用户体验。同时，后台页面简洁明了，借助simpleui优化布局，便于管理员高效执行各种管理操作。

系统从NASA网站**[[1]](#footnote-1)**获取原始数据，并将之转化为直观的可视化信息，包括站点编号、名称、地理位置、时间和多维度气象指标（如露点、气压、温度、可见度、风向和风速），并支持数据的可靠性和统计分析展示。项目还特别注重安全性，确保数据通过Web接口以JSON格式安全推送至数据库。最终，系统成功部署在华为云上，面向公网提供稳定的服务访问。

## 1.3 术语和缩略词

在讨论的课程设计中最重要的术语和缩略词如下所示：

1. **Django框架**：用于构建Web应用程序的Python高级框架，本次课程设计的核心技术之一。
2. **ECharts**：用于实现数据可视化分析的JavaScript库，帮助展示青藏高原环境监测站的数据统计图表。
3. **分布式部署**：将数据存储在数据库服务器，程序运行在云服务器上，提高了系统的稳定性和扩展性。
4. **NASA数据源**：来自美国国家海洋和大气管理局(NOAA)下属NCEI网站的气象数据源。
5. **数据库表设计**：设计符合需求的数据库表结构，设计的数据库表包括datasource和new\_table表，用于存储站点数据和统计数据。
6. **轮播图设计**：用于数据展示界面的交互式设计，允许用户浏览不同类型的站点数据图表。
7. **CURD操作**：在后台管理系统中，完成对用户数据和气象数据的增删改查功能。
8. **前端地图交互**：使用ECharts库结合地图API，实现用户点击地图上站点时的数据展示和导航。
9. **用户权限管理**：区分访客用户和管理员用户的不同权限，提供用户账号的注册、登录、管理和删除功能。
10. **云服务集成**：将项目部署在华为云服务器上，并使用云数据库或对象存储服务进行数据存储和分发。

## 1.4 参考资料

1. **NASA网站的气象数据源网址**：<https://www.ncei.noaa.gov/maps/daily/>
2. **华为云服务官方网址**：<https://www.huaweicloud.com/>
3. **Echarts库官方网址**：<https://echarts.apache.org/zh/index.html>
4. **Django框架官方文档网址**：<https://www.djangoproject.com/start/overview/>
5. **HTML教程网址**：<https://developer.mozilla.org/zh-CN/docs/Learn/HTML>
6. **CSS教程网址**：<https://developer.mozilla.org/zh-CN/docs/Learn/CSS>
7. **JavaScript教程网址**：<https://developer.mozilla.org/zh-CN/docs/Learn/JavaScript>

## 1.5 软件需求规格书其余部分

接下来的内容组织如下，首先我会在产品概述即第二部分从产品的背景、功能出发，深入阐述产品的孕育环境及其核心功能特点，随后剖析目标用户群体的特征以及他们在使用产品时的特定需求。紧接着，我们会详尽列举在设计和实施过程中需遵循的各项约束条件，包括设计层面的技术制约因素，以及项目推进过程中的内外部限制。

在第三部分具体需求的论述中，首要探讨的是产品与外界交互的各种接口要求，不仅涵盖用户界面的人机交互设计，还会详细介绍与硬件设备、其他软件系统间的协同机制以及通信接口的标准和要求。接着，我们将通过实体关系图（ER图）和数据流图等形式，直观描绘数据结构和信息流转的内在逻辑。此外，本部分还将对产品质量提出严格标准，涉及兼容性需求以适应多样化的运行环境，安全性需求以保障数据和用户隐私的安全，以及维护性和可扩展性需求以确保产品在生命周期内易于维护升级和适应未来需求变化。

最后，在第四部分附录中，我们将提供额外的参考资料、图表、规范文件和补充说明等重要信息，以供读者更全面地理解本软件需求规格书的内容和上下文。通过对以上各个方面的细致梳理和严谨表述，力求确保软件开发团队能够准确把握客户需求，遵循既定规范，高效高质量地完成软件产品的设计与开发工作。

# 二、产品概述

## 2.1 产品背景

产品立足于环境监测数据管理，运用Python和Django框架构建一体化信息系统。对接NASA气象数据，实现监测站环境数据自动化收集与呈现，支持实时及历史记录查询。系统采取分布式部署，云端存储与计算分离，提供前端交互式地图和移动端浏览体验。利用Echarts进行可视化分析，满足科研人员及公众的多元化需求，兼具模块化、可扩展和易维护特性，确保长期稳定运行与未来升级。

## 2.2 产品功能

产品功能主要包括：

* 1. **数据管理**：系统具备对环境监测站数据的增删改查(CRUD)功能，支持后台管理员对气象数据和用户数据的完整管理，包括但不限于温度、风速、湿度、PM2.5等实时及历史数据。
  2. **前端展示**：访客用户可以通过前端界面查看地图上各个监测站的位置信息及数据概览，支持站点详情的点击跳转，提供数据统计分析图表，如利用Echarts库绘制折线图、柱状图等。
  3. **地图交互**：用户可以通过地图点击获取特定站点详细信息，并能跳转至数据展示页面，页面上展示包括站点地理位置、检测数据在内的多维度信息。
  4. **数据可视化分析**：对监测数据进行统计分析，展示每种数据序列的最高值、最低值、平均值、方差、变异系数以及数据可信度等关键指标。
  5. **用户账户管理**：包括访客登录、注册、管理员登录以及用户账号的增删改查操作，后台对用户权限进行细致划分和管理。

## 2.3 用户特征

用户特征包括：

1. **多样化角色**：包含普通访客用户与管理员两类，访客可浏览站点信息和数据，管理员享有数据管理权限，执行CURD操作。
2. **数据需求强烈**：用户关注青藏高原环境监测站的各项数据，如温度、风速、湿度、PM2.5等，希望通过可视化图表与地图交互了解实时和历史数据。
3. **移动端便捷访问**：用户群体期待能在移动端高效浏览数据，要求界面友好、操作便捷。
4. **统计分析需求**：用户需要借助Echarts等工具进行数据深度挖掘和统计分析，以便做出决策或科研用途。
5. **安全性意识**：对于数据安全和隐私保护有较高要求，支持安全的数据推送机制，避免直接插入数据库的风险。
6. **地理信息偏好**：用户关心站点地理位置信息，并期望通过地图功能直观展现监测站分布及数据关联性。

综合来看，用户群体既涵盖了一般公众用户对环境数据的关注，也包括专业技术人员对数据管理和分析的需求。

## 2.4 约束条件

### 2.4.1 设计约束

1. **技术选型受限**：必须采用Python编程语言和Django框架进行开发，以确保与现有项目技术和团队技能匹配。
2. **数据源规定**：必须对接NASA提供的公开气象数据源，且在必要时可与其他合规数据源整合。
3. **数据安全与隐私保护**：设计时须遵守相关法律法规，确保所有数据加密存储和传输，防止敏感信息泄露。
4. **性能要求**：系统应具有高效的检索、加载速度，满足大量数据的实时显示和分析需求。
5. **响应式布局**：前端设计需适应不同终端设备，包括桌面端和移动端的屏幕尺寸和分辨率。
6. **数据完整性与一致性**：数据库设计需考虑事务处理和备份恢复策略，保证数据的完整性和一致性

### 2.4.2 过程约束

1. **时间进度限制**：项目开发需按照预定的时间表进行，按时交付每个阶段成果。
2. **成本控制**：在满足功能需求的同时，需控制开发成本，合理利用云服务资源。
3. **合规性审查**：在整个开发过程中，需定期进行内部和外部合规性审查，确保符合行业标准和规定。
4. **版本控制与迭代**：采用有效的版本控制系统，明确迭代计划和周期，保证项目按版本稳步进展。
5. **测试验证**：开发完成后需进行全面的功能测试、性能测试和兼容性测试，确保系统稳定可靠。
6. **文档完备性**：在整个开发流程中，需同步编写和完善技术文档。

# 三、具体需求

## 3.1 外部接口要求

### 3.1.1 用户界面

本节介绍我们网站的实现的功能，包括前台的首页、数据展示界面以及登录和注册界面的功能实现，后台对用户数据和气象数据的操作步骤和注意事项。可供访问的网址为[**http://121.36.95.49:8000/**](http://121.36.95.49:8000/)，如果想要运行本地版需要先安装django库以及运行命令**pip install django-simpleui**，然后才可在与manage.py文件同一文件夹下运行命令**python manage.py runserver**运行程序。

#### 3.1.1.1 前台-首页

 首页内容如下：

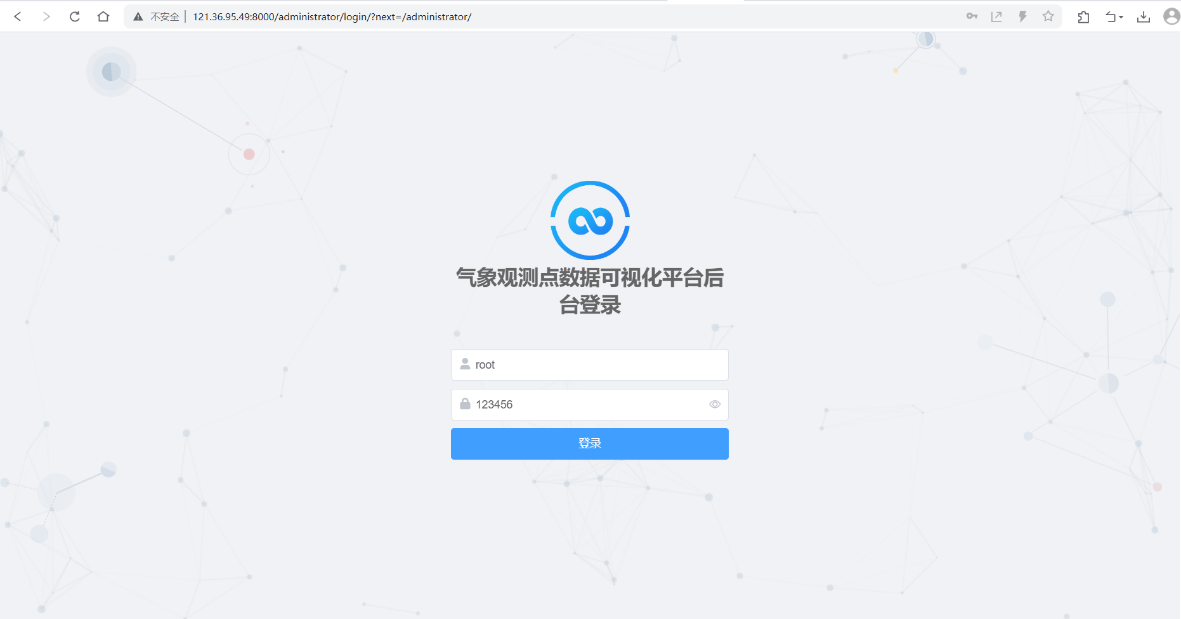
**图一 首页**

数据地图展示页面包括了三个主要区域：头部、地图展示区域和站点信息表格区域。头部提供访客登录和注册、后台登录和联系我们的链接，方便用户进行相关操作。访客登录界面和注册界面如图二所示：

当访客登录或者注册成功后，首页中左上角会变为图三所示的样子：

**图三 访客登录或者注册成功**

**图二 访客登录界面和注册界面**

 或者是点击管理员登录，这里我们推荐使用的管理员账户是**root/123456**：

**图四 管理员登录**

站点信息表格区域也就是右侧的滑动框图，其使用Bootstrap表格样式展示站点的名称和数据记录条数。支持滚动，方便查看大量站点信息。

地图交互指的是用户点击地图上的站点时，通过JavaScript监听器获取站点信息。然后页面跳转至数据展示页面，并将站点信息作为参数传递给子页面。这样做使得用户的交互感较强，且可视化方便用户查询相关站点的信息。

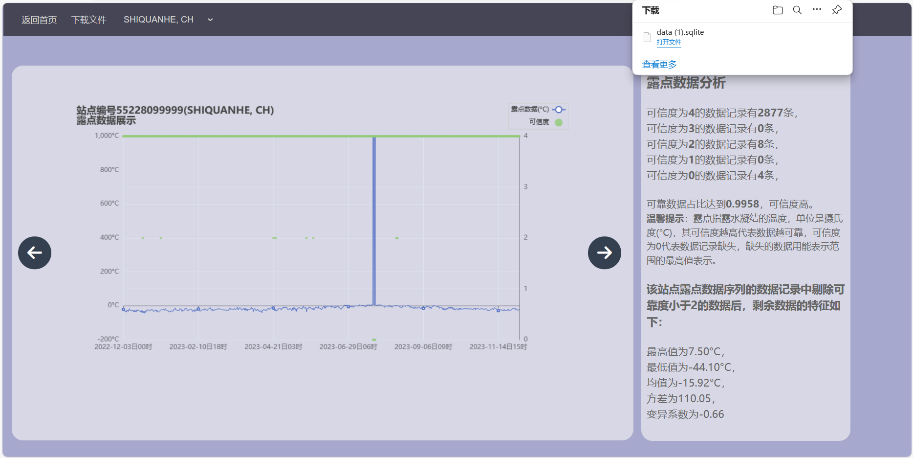
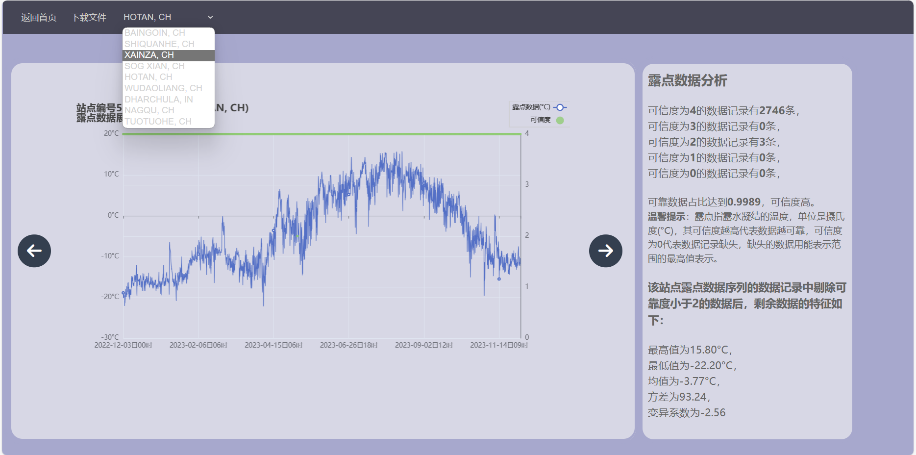
#### 3.1.1.2 前台-数据展示界面

 点击首页中地图上的站点标识会跳转到数据展示界面，其样式如图五所示：

**图五 数据展示界面样式**

图五中页面分为了三部分：顶部页面头、交互式Line图区域、对数据序列的分析框图区域。

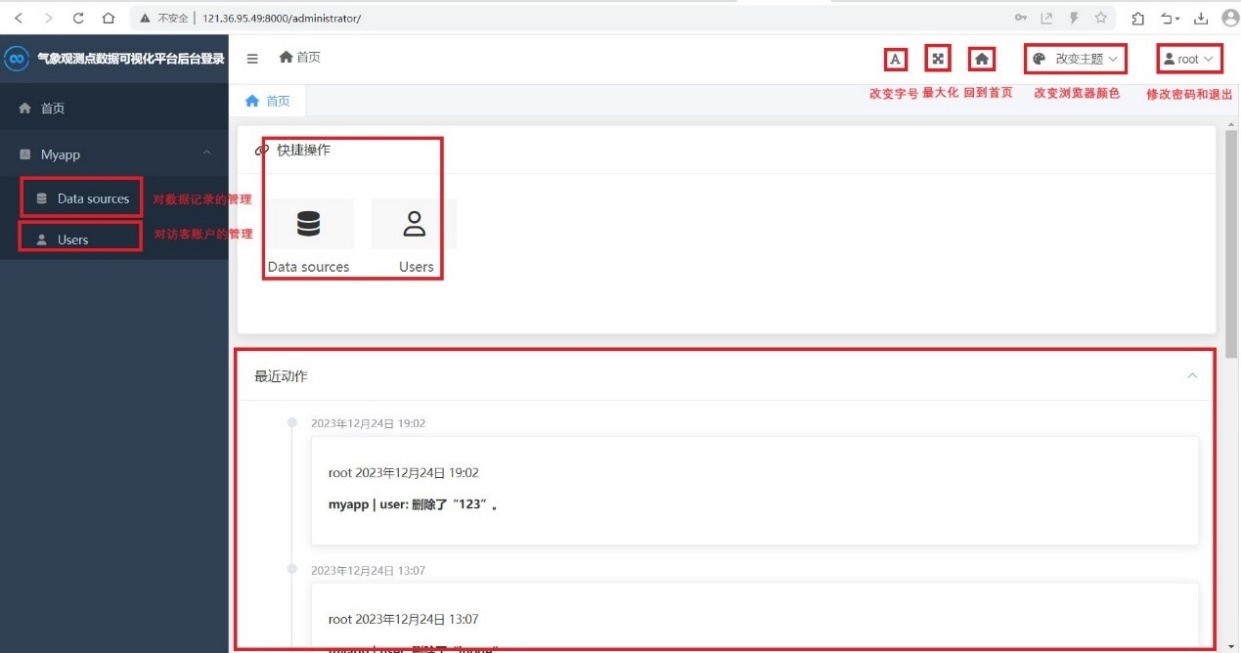
在顶部页面头中，一共有三个按钮，分别是“返回首页”、“下载文件”、下拉框图。点击“返回首页”时会跳转到首页，点击下载文件时会下载.sqlite数据源文件供用户使用，点击下拉框时选择指定站点可以跳转到其具体展示页面，图六和图七分别是点击下载文件和下拉框时的截图：

 在交互式Line图区域中，有向左和向右两个箭头以及中间的数据序列的Line图展示，当用户点击向左按钮时，交互式Line图区域中的展示片会向左滑动一张，而对数据序列的分析框图区域中的展示片会向上滑动一张。我们的页面会展示指定站点的检测到的露点数据、气压数据、气温数据、可视距离数据、风向数据和风速数据。而在对数据序列的分析框图区域，我们会针对每一种数据序列的最高值、最低值、均值、方差和变异系数以及数据记录的可信度进行分析。

**图七 选择站点**

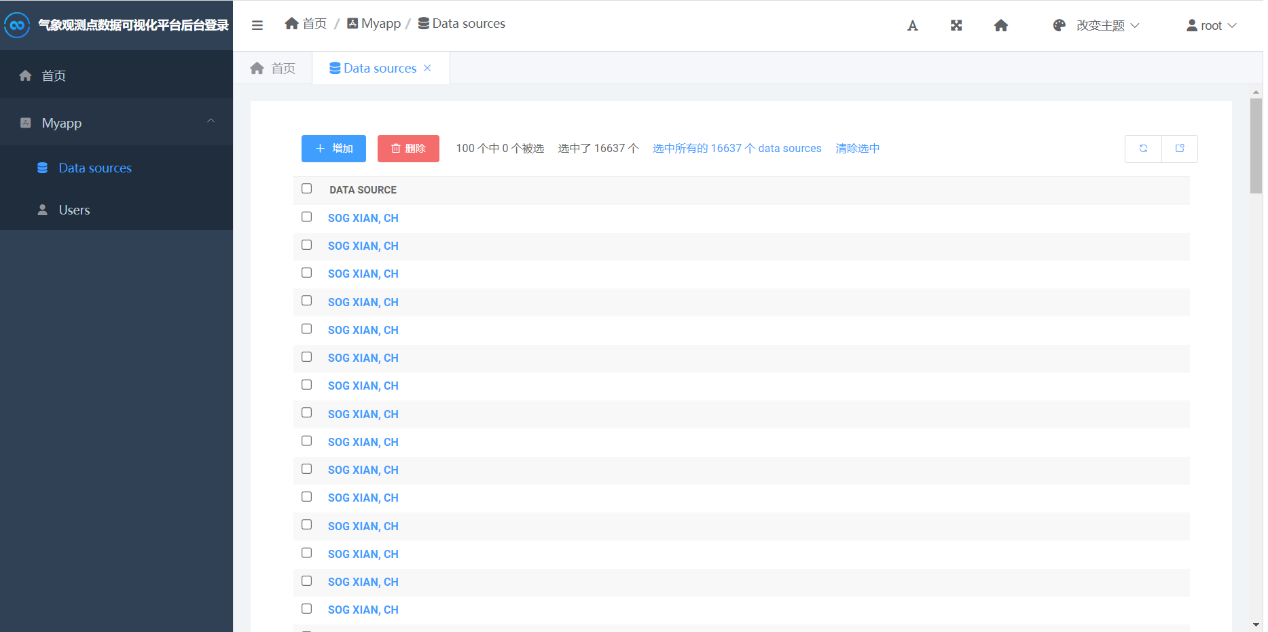
**图六 下载文件**

#### 3.1.1.3 后台-用户数据和气象数据的CURD

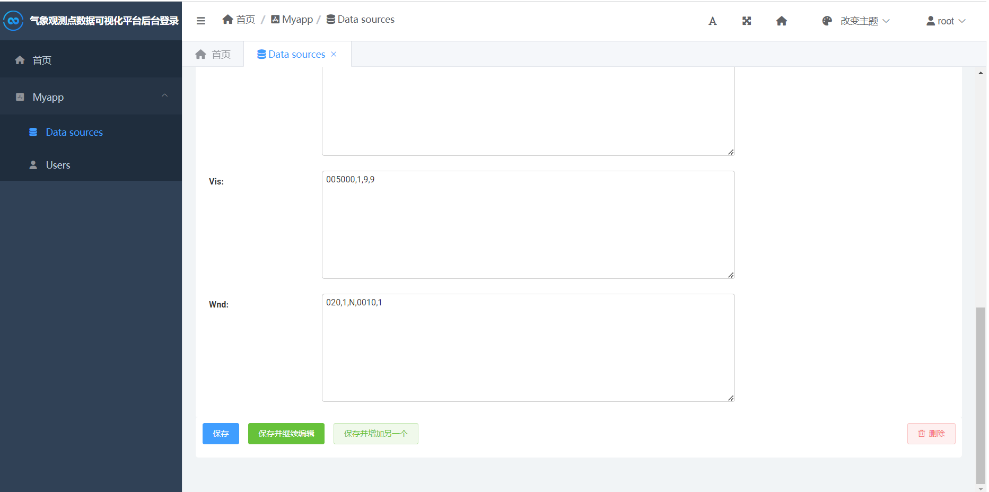
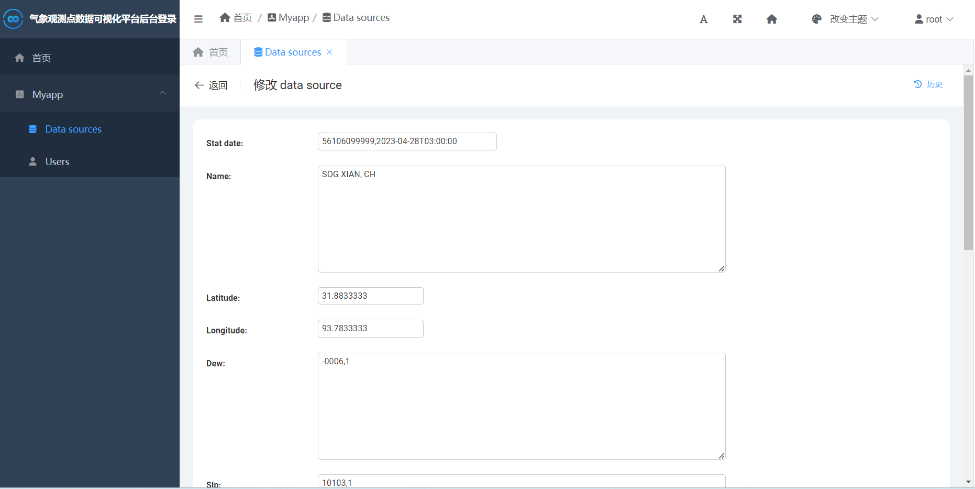
 在利用管理员账户**root/123456**进入后台后，后台的界面及功能区域展示在图八中：

**图八 后台功能区域展示**

左侧为功能栏，默认展示界面为首页部分，首页中有快捷操作以及最近动作两部分构成，方便管理员直接对后台功能整体有一个大概的了解。本后台采用**simpleui**进行优化，使得页面看起来更为整洁，同时使用更为方便。  
 然后是展示数据记录的页面：

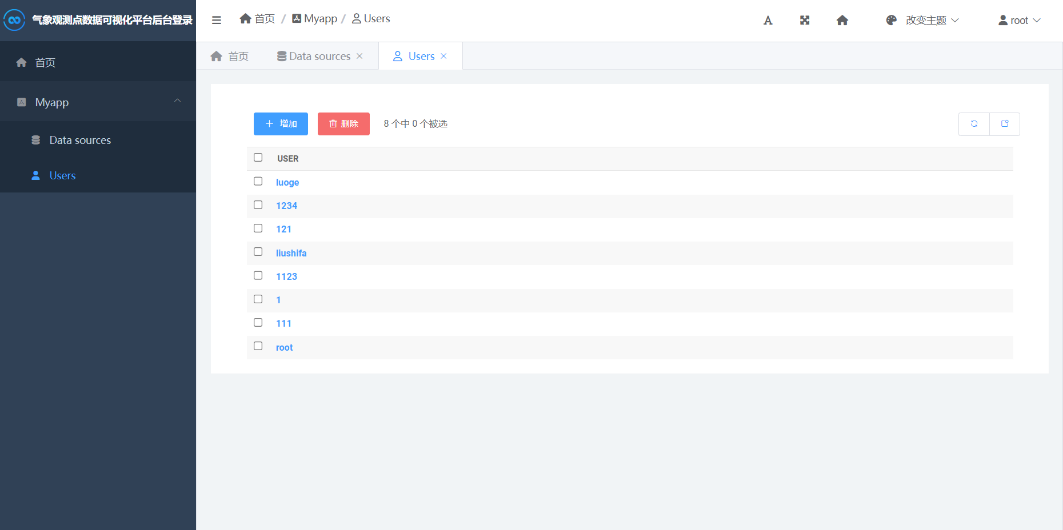
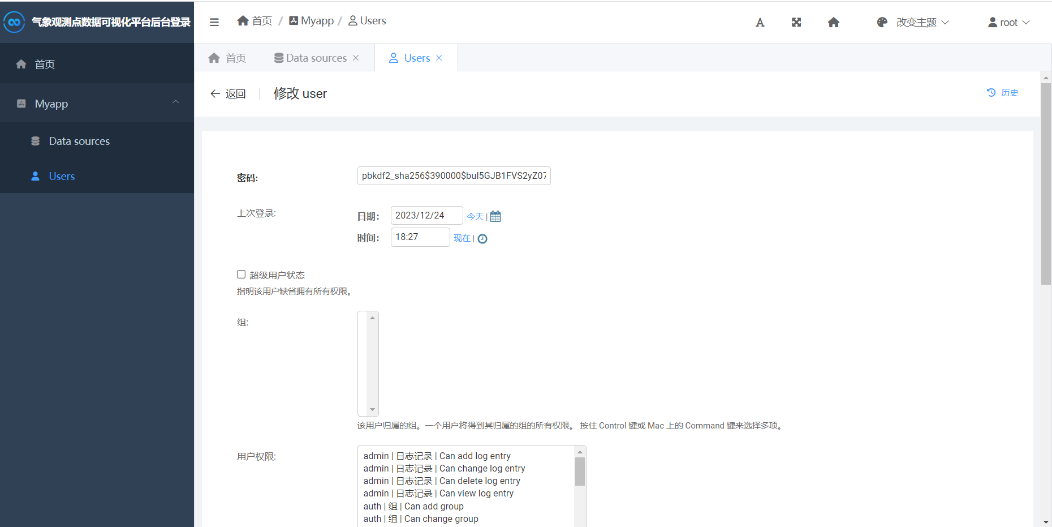
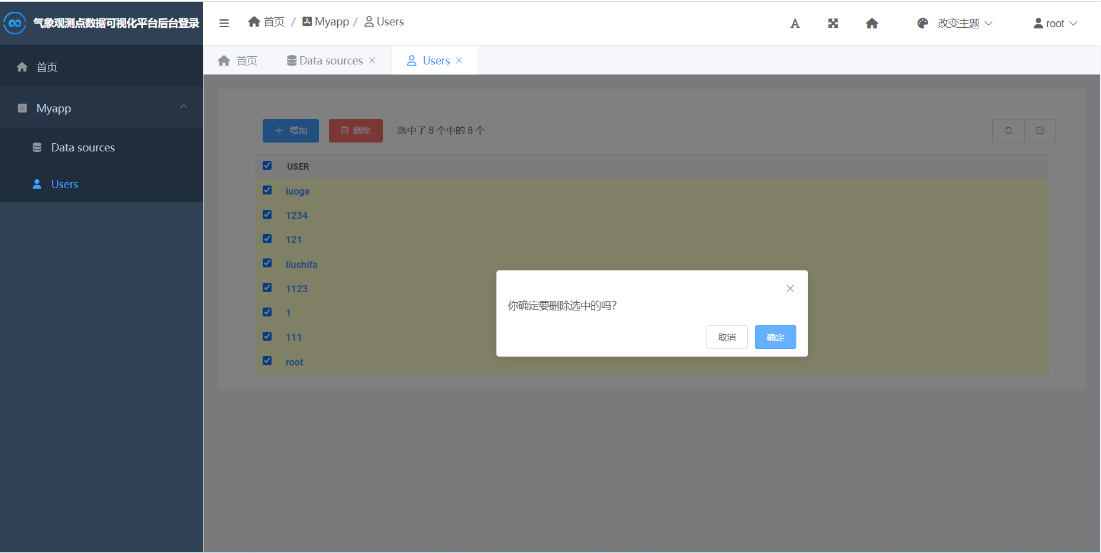
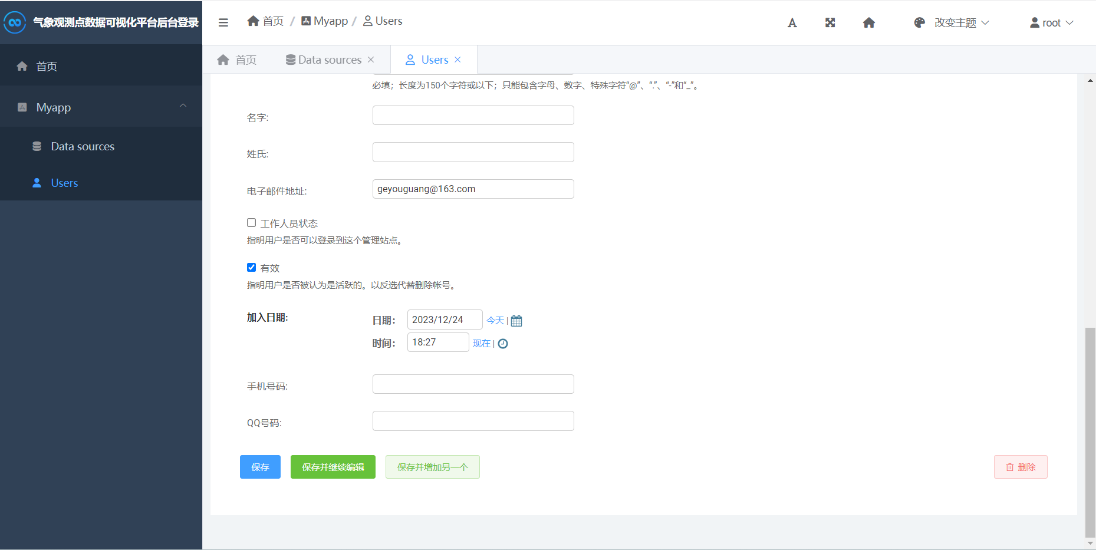
 首先展示了当前观测站的，此处具有数据的增删改查功能。上部分增加和删除两个按钮分别对应了数据的增删，同时显示出当前具有的数据量总数。此外，每个数据站前对应一个复选框，使得数据删除可以批量进行操作。

**图九 后台展示数据记录的页面**

 点击某条数据记录进入，查看修改和保存功能：

**图十 对后台所展示数据记录的修改页面**

此时会显示出数据站id以及数据导入时间、数据站名称、纬度、经度、露点等数据以供管理员进行查看和修改操作。

 以下是对数据记录的操作界面，我们还可以对用户账号进行操作：

**图十四 后台批量删除无用的用户账号**

**图十三 后台修改用户账号的信息页面(二)**

**图十二 后台修改用户账号的信息页面(一)**

**图十一 后台展示用户账号**

图十一中展示了用户账号的记录，图十二和图十三是对用户账号的修改和查看，可以得到用户的权限信息以及其他信息，图十四是删除无用用户账号的信息页面。

### 3.1.2 硬件接口

硬件接口主要用于系统与各类环境监测设备（如温度传感器、风速仪等）之间的数据交互。通过标准化的通信协议（如Modbus、MQTT等）将传感器数据实时采集到系统中，确保数据的准确性和完整性。同时，系统还需与云存储服务器或本地硬件设施（如硬盘阵列）进行高效数据交换，实现数据的存储与备份。**由于我们采用的是来自美国国家海洋和大气管理局(NOAA)下属NCEI网站的气象数据源，因此我们不需要关注硬件接口，而只需要学会使用已有的气象数据。**

### 3.1.3 软件接口

服务接口 (Service Interfaces)：①MySQLTool类用于提供数据库访问接口，设计用于系统与数据库系统MySQL之间的交互，包括数据的查询、插入、更新和删除操作，封装了SQL操作，以确保数据的一致性和安全性。②模块间通信接口：不同功能模块之间的API调用接口，比如用户管理模块与数据分析模块之间的接口，以便模块间数据流转和协同工作。

### 3.1.4 通信接口

通信接口确保了系统在网络环境中的数据传输安全和高效。我们采用HTTPS协议保证数据传输的加密安全，同时支持WebSocket等实时通信协议，实现实时数据流的推送。此外，系统针对分布式部署，设计了高效稳定的内部通信机制，确保多个节点间的数据同步和任务协调。

## 3.2 数据定义

### 3.2.1 ER图

### 3.2.2 数据流图

## 3.3 质量需求

### 3.3.1 兼容性

在我们开发的环境监测信息系统中，兼容性是至关重要的质量需求之一。系统需确保在不同操作系统、浏览器环境下均能稳定运行，这意味着前端应用要充分兼容Windows、Mac OS、Linux等操作系统上的主流浏览器，包括最新的Chrome、Firefox、Safari以及IE/Edge浏览器的不同版本。另外，系统还需要支持在各种屏幕尺寸和分辨率的设备上无缝切换，特别是移动端设备如iPhone、iPad以及各类Android设备。在后端，系统需兼容不同的数据库类型，如MySQL、PostgreSQL等，以及与第三方服务或硬件设备（如传感器设备）的接口兼容。

### 3.3.2 安全性

系统在设计和实现过程中，高度重视数据安全与隐私保护。所有数据传输通过HTTPS协议加密，确保数据在互联网上的安全传输。同时，采用严格的身份认证机制，如密码强度策略、二次验证等，防止未经授权的访问。对于用户敏感信息，如个人信息和监测数据，系统将在数据库中进行加密存储。除此之外，系统还内置防御XSS、CSRF等多种网络攻击手段，定期进行安全审计与漏洞扫描，及时修复潜在安全风险。

### 3.3.3 维护性

为了提高系统的可维护性，我们采用模块化设计，各功能模块相互独立，便于进行单元测试和维护升级。系统生成详细的日志记录，包括错误日志和操作日志，有利于定位问题和优化性能。同时，采用敏捷开发和DevOps理念，配置自动化测试、持续集成和持续部署（CI/CD）工具链，简化上线流程，提升维护效率。

### 3.3.4 可扩展性

针对未来可能出现的业务增长和技术变革，系统设计之初就着重考虑了可扩展性。采用分层架构和微服务设计，新功能模块可以独立开发和部署，不影响现有系统稳定运行。数据库采用分布式架构设计，可根据数据量的增长进行水平或垂直扩展。与此同时，API接口设计遵循开放、通用的原则，方便与其他系统集成或在未来进行功能拓展，确保系统能随业务发展需求而灵活调整和扩展。

# 四、附录

## 4.1 数据库表设计

设计数据表前，我们需要先明确数据源以及字段。数据来源于NASA网站**https://www.ncei.noaa.gov/maps/daily/**。对于每个站点，我们存储的数据包括它的站点编号、名称、纬度、经度、记录的时间(每三小时会记录一次数据)、露点、海平面压力、温度、可见度、风向角和风速。对应数据源中的字段分别为**STATION、NAME、LATITUDE、LONGITUDE、DATE、DEW、SLP、TMP、VIS、WND**。对应的具体含义见下表：

**表一 原始数据记录的含义**

|  |  |
| --- | --- |
| **原始数据字段** | **含义** |
| **STATION** | **每个站点唯一的编号** |
| **NAME** | **例如“BAINGOIN, CH”，BAINGOIN是地名班戈县(西藏那曲市下辖县)，CH是China的缩写** |
| **LATITUDE** | **观测站的纬度** |
| **LONGITUDE** | **观测站的经度** |
| **DATE** | **记录的时间，每三小时会记录一次数据，例如“2022-12-03T00:00:00”** |
| **DEW** | **露点(露水凝结的温度)，例如“-0275,1”，-0275的缩放因子是10，单位是摄氏度，说明实际的温度是-27.5℃，如果为+9999代表数据缺失，不用管后面的数据可靠性分级。逗号后面的1代表观测到的数据的可靠性，说明这个观测值很可靠** |
| **SLP** | **海平面压力，缩放因子为10，例如“10140,1”，表明是1014.0 hPa，为99999则代表数据缺失，不用管数据可靠性分级** |
| **TMP** | **温度，缩放因子为 10，当温度数据缺失时，使用 +9999 表示，例如对于“-0056,1”，记录的温度是-5.6℃，可靠性很高。** |
| **VIS** | **可见度，包括能够看到并辨认物体的水平距离等数据，例如“030000,1,9,9”，其中030000代表可见度为30000米，999999代表数据缺失，后面两个数据都是9，不使用** |
| **WND** | **例如“240,1,N,0070,1”，不关注第三个数据，第一个数据是240°，是风向角，它是指从正北开始，按照顺时针方向测量的风吹来的方向与正北之间的角度，999代表数据缺失。第四个数据即0070，0070的缩放因子是10，单位是m/s，表示风速是7m/s，9999代表数据缺失。** |

此外，每个数据都有对应的可靠性分级，逗号后面就是可靠性分级，具体的分级为：

**0: 通过了粗略限制检查。**

**1: 通过了所有的质量控制检查。**

**2：可疑，可能存在问题。**

**3: 错误的，可能包含错误。**

**4: 通过了粗略限制检查，数据来自NCEI数据源。**

**5: 通过了所有的质量控制检查，数据来自NCEI数据源。**

**6: 可疑，数据来自NCEI数据源。**

**7: 错误的，数据来自NCEI数据源。**

**9: 如果元素存在，通过了粗略限制检查。**

**A: 数据值被标记为可疑，但被接受为有效值。**

**C: 从自动气象观测系统（AWOS）接收的温度和露点温度以整数摄氏度报告。自动质量控制标记这些值，但它们被接受为有效。**

**I: 数据值最初不在数据中，但由验证器插入。**

**M: 根据国家气象局（NWS）或联邦航空管理局（FAA）提供的信息对值进行手动更改。**

**P: 数据值最初没有标记为可疑，但被验证器替换。**

**R: 数据值被NCEI软件计算的值替换。**

**U: 数据值被编辑值替换。**

然后我们就可以开始设计数据库中的表，首先要确定一个主键。对于每条数据记录而言，我们只能通过其所属站点编号和具体时间将它和其它数据记录区分开，因此，我们需要将这两个原始字段拼接起来组成数据库中存储数据源的表的主键。最后设计的表及对应的字段如下：

**表二 存储了数据源的表datasource的字段**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **含义** |
| **stat\_date** | **VARCHAR(50)** | **主键，能够标识特定站点在指定时间的数据记录** |
| **name** | **TEXT** | **非空，表示监测站点的名称** |
| **latitude** | **REAL** | **表示纬度** |
| **longitude** | **REAL** | **表示经度** |
| **dew** | **TEXT** | **非空，表示露点** |
| **slp** | **TEXT** | **非空，表示海平面气压** |
| **tmp** | **TEXT** | **非空，表示温度** |
| **vis** | **TEXT** | **非空，表示能见度** |
| **wnd** | **TEXT** | **非空，表示风向和风速** |

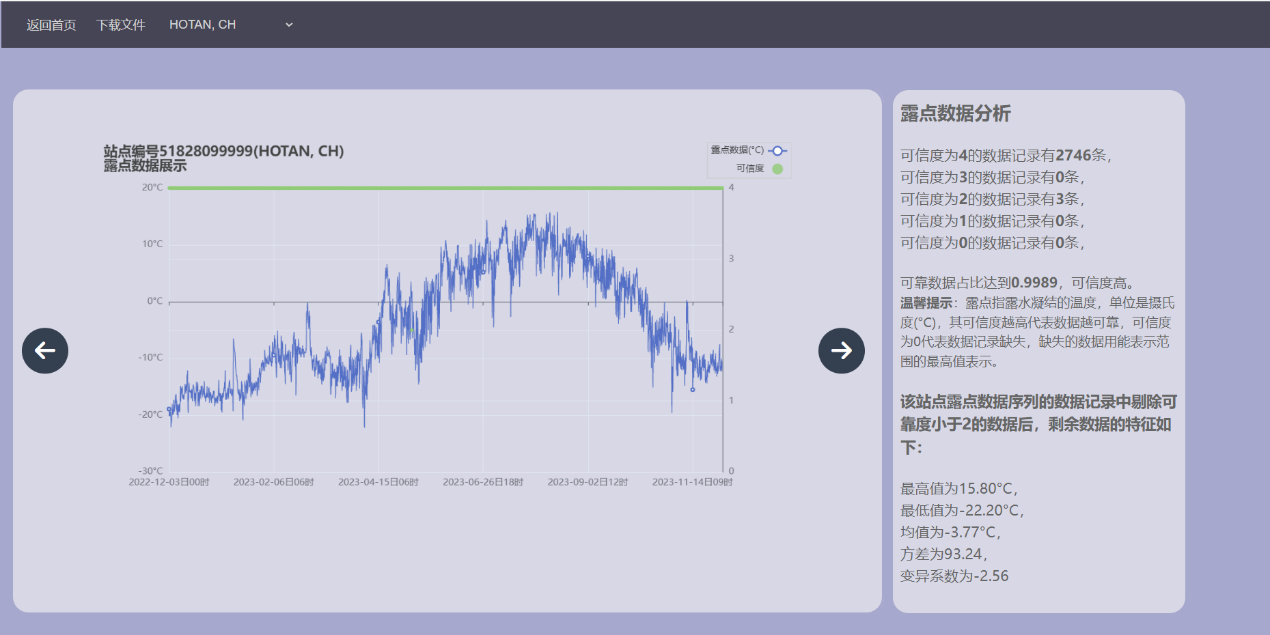
**表三 汇总每个监测站点的相关信息的表new\_table的字段**

|  |  |  |
| --- | --- | --- |
| **字段** | **类型** | **含义** |
| **name** | **TEXT** | **非空，表示监测站点的名称** |
| **latitude** | **REAL** | **表示纬度** |
| **longitude** | **REAL** | **表示经度** |
| **data\_count** | **INTEGER** | **表示与该数据源相关的记录数量** |

**new\_table表的数据可以通过触发器（triggers）实时更新，以保持与 datasource 表的同步。**

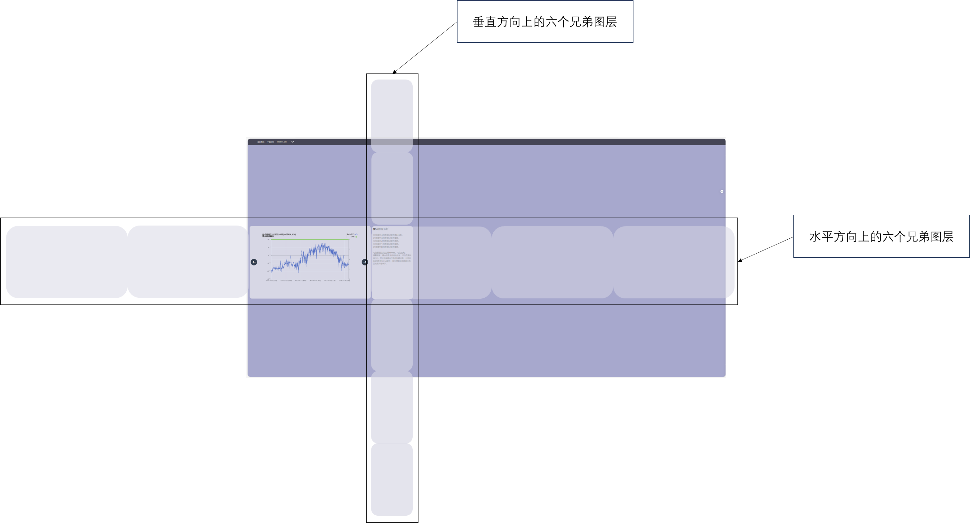
## 4.2 数据展示界面模板设计

编写的模板文件为/myapp/templates/showdata.html，设计的难点和重点分为两部分，一个是轮播图的设计，另外一个是echart图像对应的html文件的实时渲染和导入。

 首先是轮播图的设计，轮播图的整体视图如下：

**图十五 轮播图的整体视图**

左边是一个动态交互展示数据的图层，右边是对数据记录的可信度分析图层。点击向左的按钮时，左边的图层向左滑动，右边的图层会向上滑动，这种动态效果依托于/myapp/static/js/scripts.js文件实现。

 在showdata.html文件中，我们先为两个轮播图分别设计了一个父对象元素，并展示每个站点的六类信息，因此它们实际上各有六个子图层：

**图十六 轮播图的实际视图**

对于小图层C，它们是最小的展示元素，然后横向和纵向的六个小图层各自组成了一个整体并撑起了自己的父对象元素B，也就是一个长方框，然后每一个长方框又有一个自己的父对象元素A。由于有六个C，因此我们将父对象元素A的长度设置成B的1/6，而宽度等于B的。然后我们在scripts.js文件中设置点击相应按钮后会产生的动作，具体的函数如下：

const updateClick = (e) => {  
 //Calculate the uodated html index and text index based on the button clicked

***htmlIndex*** += e.target.id === "next" ? 1 : -1;  
 ***textIndex*** += e.target.id === "next" ? 1 : -1;  
 slideHtml(***htmlIndex***);  
 slideText(***textIndex***);  
}

其中htmlIndex和textIndex分别是横向图层的索引以及纵向涂层的索引，对应的横向最左边和纵向最上的图层索引为0。当点击事件发生时，这个函数就会被调用，然后根据我们为两个按钮编号的别称(prev以及next)就可以判断更新后的横向和纵向下一个被展示的图层。slideHtml函数如下：

const slideHtml = () => {  
 //Calculate the updates html index  
 //console.log("htmls："+htmls.length);  
 ***htmlIndex*** = (***htmlIndex*** === 6 ? 0 : (***htmlIndex*** === -1 ? 5 : ***htmlIndex***));  
 //Updaate the carousel display to show the specified html  
 ***carousel***.style.transform = `translate(-${***htmlIndex*** \* 1150}px)`  
}

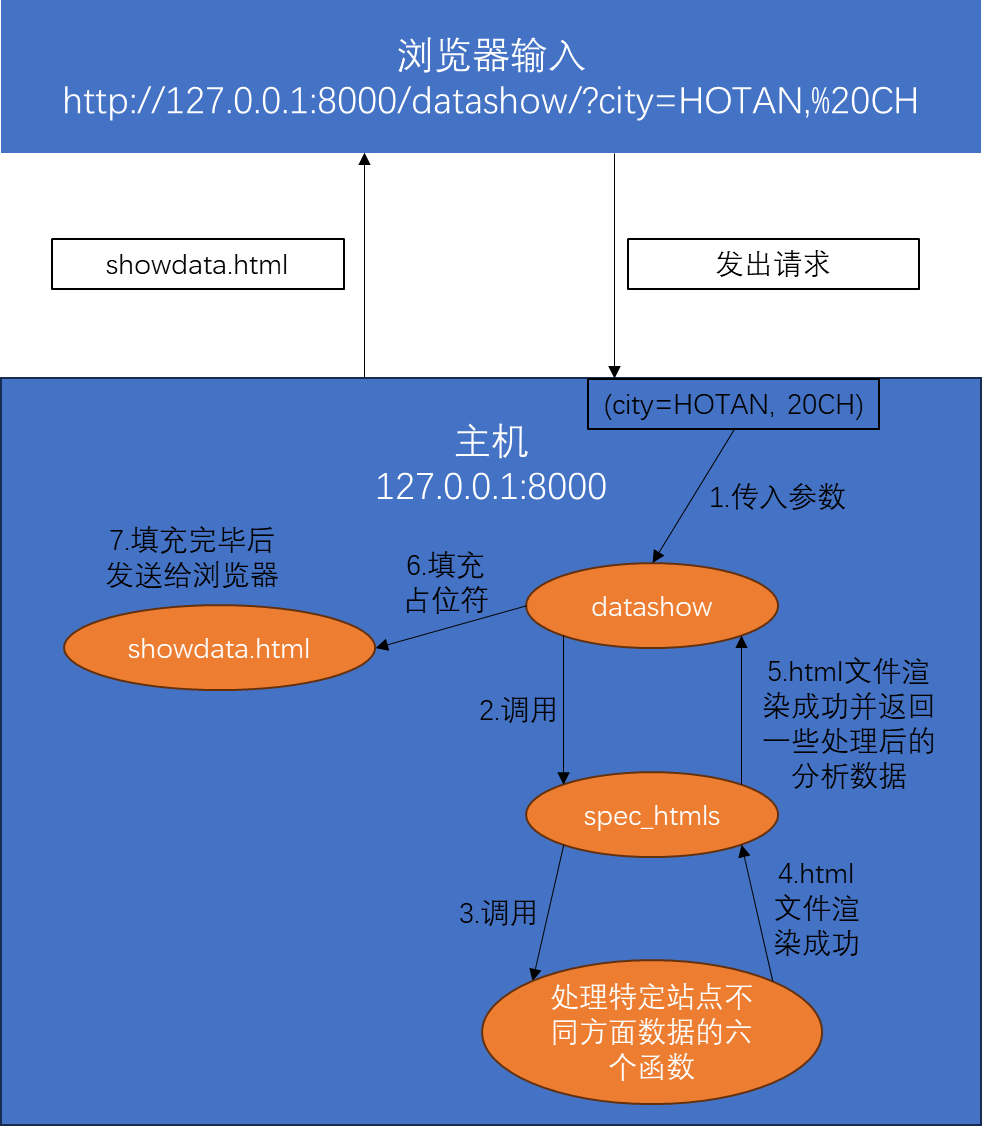
为了实现轮播，我们需要在图层索引为6或者-1的时候分别更新图层索引为0和5，这两个操作的含义是最右边图层展示后要回到最左边图层、最左边图层展示后要回到最右边图层。slideText函数功能类似，不再赘述。

然后是echart图像对应的html文件的实时渲染和导入，这个问题需要通过嵌套网页的标签iframe实现，具体是在图层C的父对象B中插入六个iframe元素：

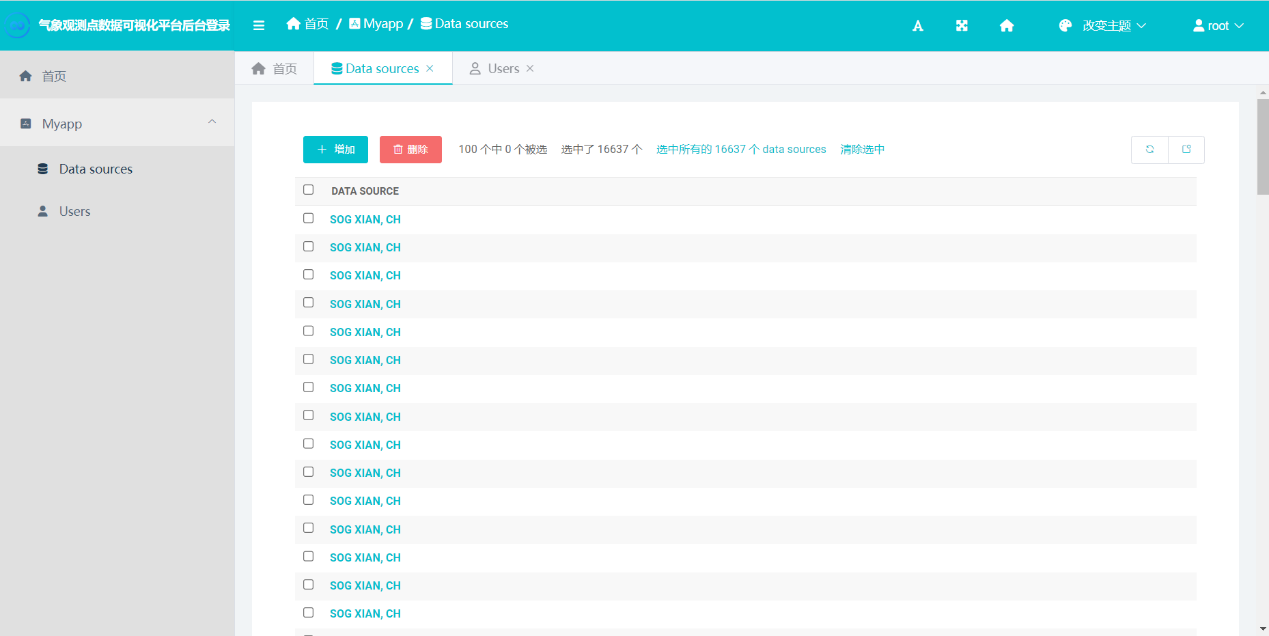
<div class="carousel">  
 <iframe src="../static/html/{{ stat }}/dew.html"></iframe>  
 <iframe src="../static/html/{{ stat }}/slp.html"></iframe>  
 <iframe src="../static/html/{{ stat }}/tmp.html"></iframe>  
 <iframe src="../static/html/{{ stat }}/vis.html"></iframe>  
 <iframe src="../static/html/{{ stat }}/wnddire.html"></iframe>  
 <iframe src="../static/html/{{ stat }}/wndsped.html"></iframe>  
</div>

{{stat}}占位符是站点编号，后续会在views.py中的datashow函数传入这个占位符具体的值。

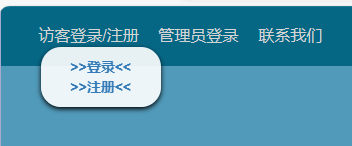
## 4.3 数据展示界面模板导入数据

每一次进入/datashow/?city=HOTAN,%20CH等类似界面，就会触发datashow函数的调用，将city=后面的参数解码后传送给该函数。而在该函数中又会调用spec\_htmls函数来重新渲染出这个参数所对应站点的各类数据图像的html文件，存入/static下方的/html目录中。具体来说，spec\_html函数内部会调用另外六个函数来获取该站点的某一类数据，每个函数通过每个站点的名称从数据库中检索出这个站点的所有数据，然后会在内部生成echarts图表并将其写入到指定的html文件中。图示过程如下：

**图十七 数据展示界面模板导入数据流程**

 前台和后台都有登录功能，前台的登录功能可供访客使用，后台的登录功能供管理员使用。而在后台，我们还可以对数据记录进行操作。这两种具体功能的实现依托于models.py文件中定义的User对象和DataSource对象以及forms.py中定义的LoginForm对象和RegForm对象。当我们在admin.py文件中注册好了User对象和DataSource对象，然后就可以直接调用已经制作好的后台实现对用户和数据记录的增删改查操作。

**图十八 后台数据记录展示**

 具体的实现逻辑是当访客进入登录或者注册界面时，首先分别会调用views.py中的do\_login函数和do\_reg函数，这两个函数会解析接收到发送来的POST信息时，会有具体的逻辑操作。do\_login函数会根据提交的表单信息来解析出用户名和密码，然后通过中间件authenticate来验证这个账户是否有资格访问后台。do\_reg则会根据读取到的用户的用户名、邮件和密码来为该用户创建一个账户，并重定向回到首页。另外我们还在首页/welcome中设置了登录或者注册成功的页面和访客页面不同的地方：

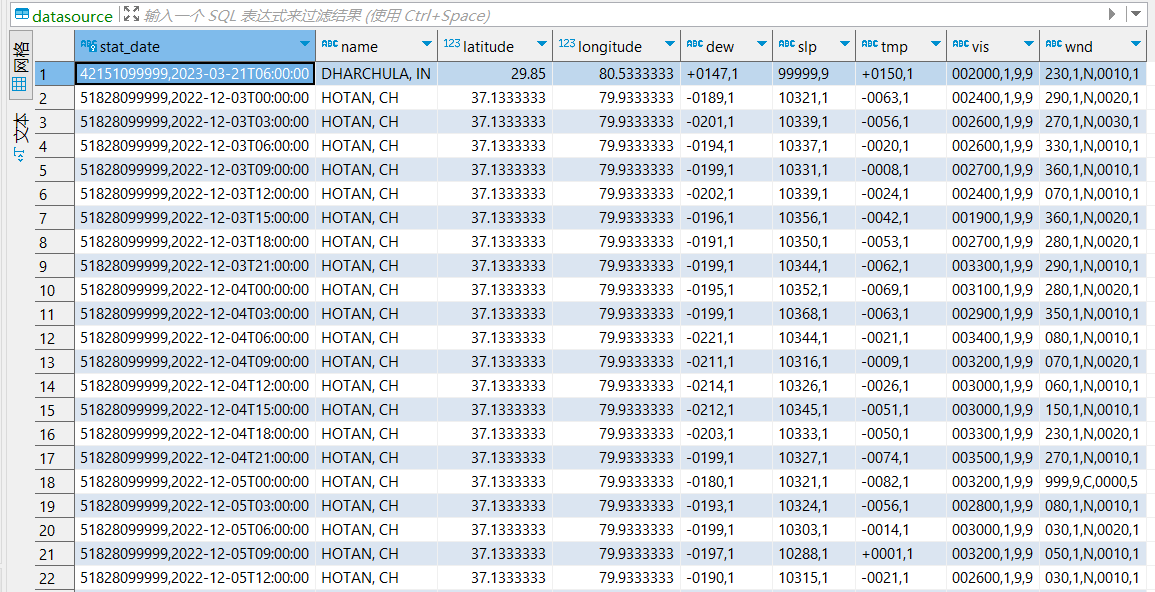
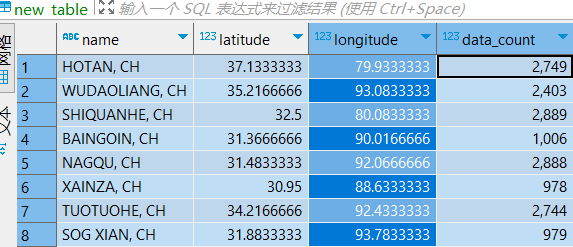
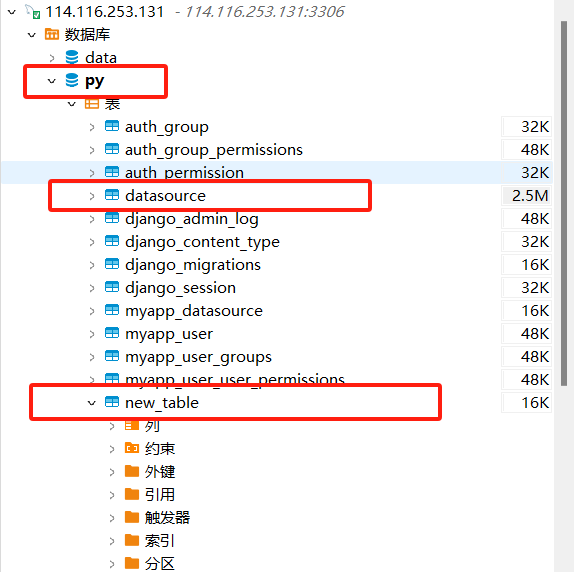
**图十九 登录前后界面变化**

可以看到，登录成功和登陆失败的图标是不一样的。然后我们可以选择登出并且选择管理员登录，管理员账户为root/123456，登录成功后就可以进入后台。

## 4.4 分布式设计

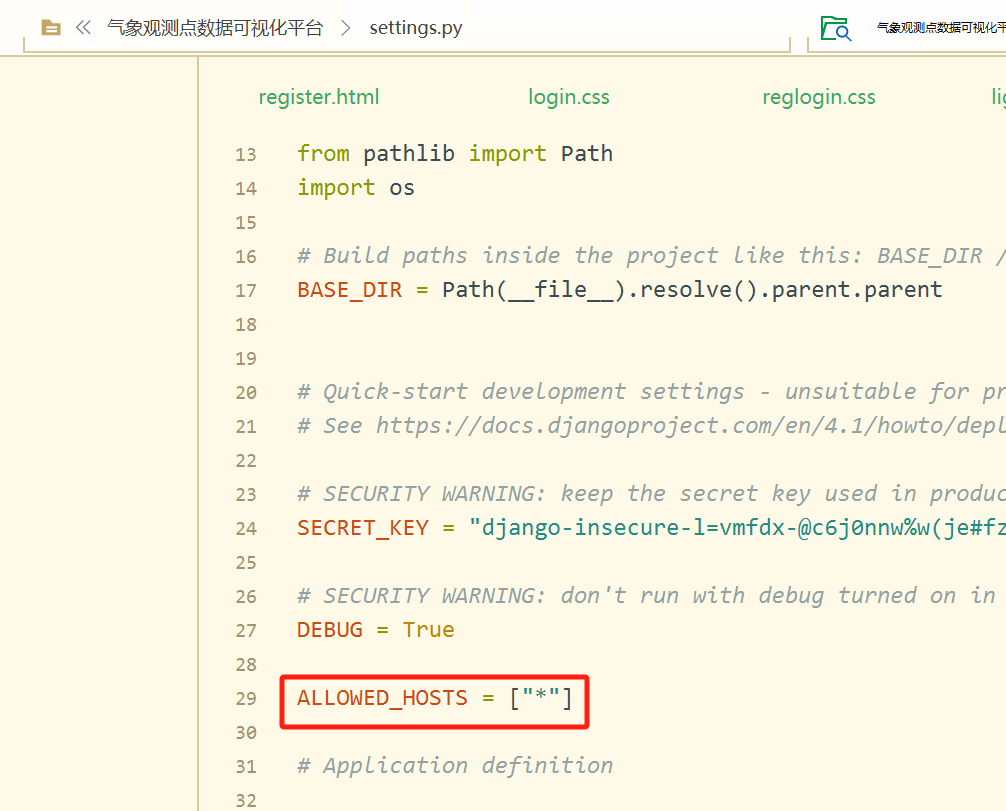
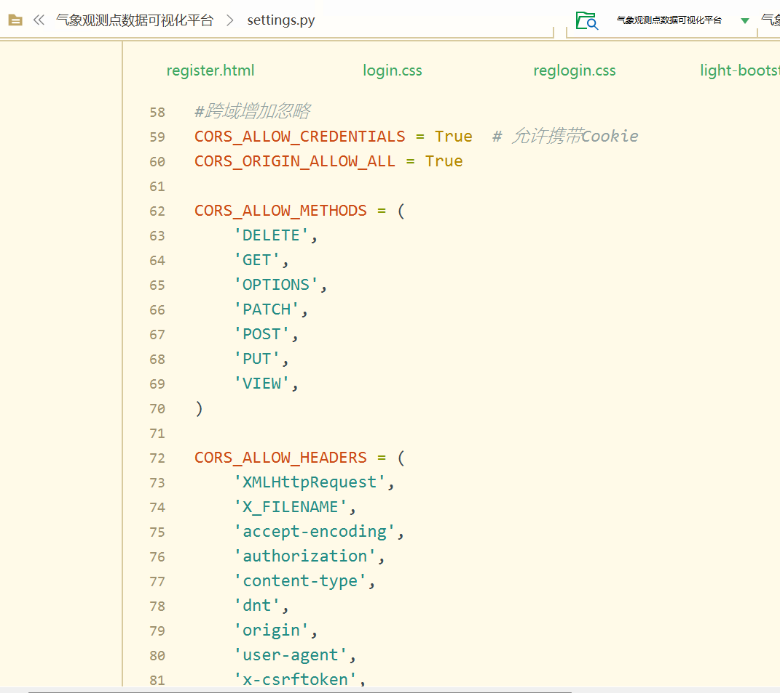
 我们的数据存储在数据库服务器中，而程序运行在云服务器上。数据库服务器的部署细节如图二十所示：

**图二十 数据库服务器配置**

 使用DBeaver展示数据表datasource和new\_table：

**图二十一 py数据库中存储了datasource和new\_table数据表**

**图二十二 datasource和new\_table数据表**

 然后是对云服务器的部署，项目整合之后上传至服务器，在服务器上首先使用python manage.py makemigrations生成迁移文件，再使用python manage.py migrate同步数据库，本例中因为将项目整体上传所以只使用（python manage.py migrate）即可。服务器在部署过程中有以下几个位置需要改动：

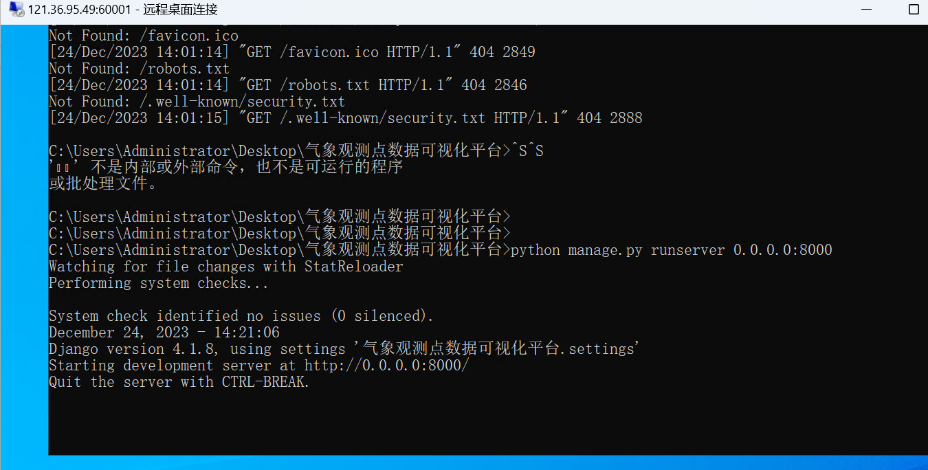
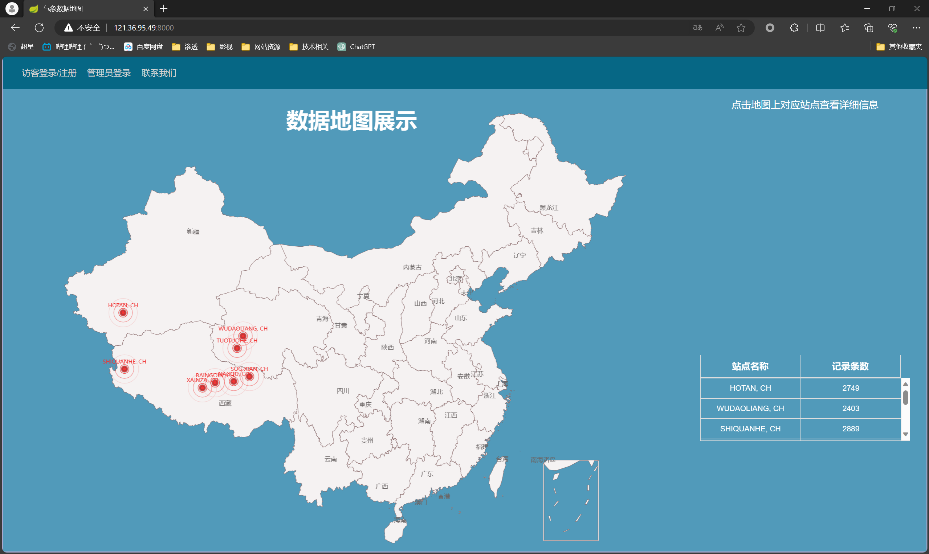
**图二十四 对setting.py文件的修改(二)**

**图二十三 对setting.py文件的修改(一)**

图二十三中表示允许用于指定可以访问应用程序的域名或 IP 地址，“\*”表示允许以任意方式访问（外网使用“http://121.36.95.49:8000” 可以访问，服务器本地使用127.0.0.1:8000也可以访问）。另外，在部署过程中还存在另一个问题是跨域访问的问题，由于浏览器的同源策略，导致部署完成后外部主机无法直接通过ip+端口方式访问，此时对settings.py做如图二十四所示的修改。允许跨域访问，然后使用**https://121.36.95.49:8000**即可成功访问。之后还有一个Django 报错**django.core.exceptions.ImproperlyConfigured: Error loading MySQLdb module**的异常情况，解决方法为在**\_\_init\_\_.py**中加入如下两行代码：

import pymysql  
pymysql.install\_as\_MySQLdb()

之后使用命令**python manage.py runserver 0.0.0.0:8000**成功启动项目(centos以及ubuntu同理，此处以windows服务器为例，实测centos同样步骤也可运行)：

成功运行截图如下，至此服务器配置完毕：

**图二十六 成功运行截图(二)**

**图二十五 成功运行截图(一)**

1. <https://www.ncei.noaa.gov/maps/daily/> [↑](#footnote-ref-1)