

《GIS 工程设计与实践》课程报告
基于 WebGIS 的新冠疫情数据
可视化系统设计与实现

姓名：顾宇

学号：2018302141300

目录

1	开发背景	1
2	需求分析	1
3	总体设计	3
3.1	系统框架设计	3
3.2	功能模块设计	4
3.3	界面设计	4
4	数据库设计	5
5	详细设计	8
6	功能实现	10
6.1	数据获取与处理	10
6.2	专题图制作	11
6.3	地图数据发布	11
6.4	主界面功能实现	12
6.5	搜索界面功能实现	16
6.6	静态页面部署	16
7	心得体会	17
8	附件材料说明	18

1 开发背景

2020年初起，Covid-19新冠疫情在全球范围蔓延。国家卫健委在2020年2月4日发布了《国家卫生健康委办公厅关于加强信息化支撑新型冠状病毒感染的肺炎疫情防控工作的通知》，其中明确指出要“强化与工信、公安、交通运输等部门的信息联动，形成公路、铁路、民航、通讯、医疗等疫情相关多方多源数据监测、交换、汇聚、反馈机制，利用大数据技术对疫情发展进行实时跟踪、重点筛查、有效预测，为科学防治、精准施策提供数据支撑。”基于此文件，诸多门户网站开始整理疫情相关的信息，推出专门追踪新冠疫情走势、更新新冠疫情信息的网站或客户端，具有代表性的有丁香园的“全球新冠肺炎疫情地图”和阿里集团的“阿里健康APP”等；全球范围内的新冠疫情数据可视化系统也相继推出，如Johns Hopkins大学推出“COVID-19 Dashboard by the Center for Systems Science and Engineering (CSSE)”。

为了更好地对疫情数据进行可视化，充分了解国内疫情的发展趋势及现状，本次大作业我完成了对基于Web平台的新冠疫情WebGIS系统开发。WebGIS是传统的地理信息系统(GIS)在网络上的延伸和发展，利用互联网对地理空间数据进行发布和应用，使空间数据的共享和互操作变得更为便捷。本系统采取前后端分离的形式，前端采用了Node.js和jquery框架，搭配ArcGIS API for Javascript、高德地图API等地图接口实现新冠疫情数据可视化、数据分析与数据查询等功能；采用ArcGIS Server服务器发布地图服务，Http-server部署本地静态页面；后端采用PostgreSQL+PostGIS数据库管理数据，并利用Python爬取疫情数据，实现了对疫情数据的获取与管理。

2 需求分析

新冠疫情WebGIS系是将实时的疫情信息与地理空间信息相结合，实现疫情数据的快速发布、查询检索以及有关统计分析的网络信息平台。主要用户是关心疫情发展的普通大众，因此系统界面必须简单、友好，数据可视化成果必须直观，同时具备较为完善的功能。结合Web地理信息系统的普遍功能和疫情系统的实际需求，遵循科学性、实用性、可拓展性和开放性等开发原则，系统应实现以下功能：

1. 地图相关基本功能

(a) 地图浏览功能

实现基本的地图浏览功能，如放大、缩小、平移、漫游显示等。用户可以将地图移动到感兴趣的区域，根据感兴趣的区域大小或清晰程度放大、缩小显示范围。该系列基本功能可以通过系统的工具栏控件实现。

(b) 图层控制功能

实现对地图图层的分层显示和控制管理功能。本系统应通过不同色块和标记的组合，直观表现各省份的疫情感染情况，制作多张疫情相关专题图。同时需要通过添加图层管理控件，实现不同图层间的切换和显示。此外，如果要添加多源数据，如卫星遥感数据和开放街道地图等，应通过图层管理控件实现不同源地图数据的显示，便于用户直观、流畅地浏览各类疫情数据。

(c) 地图查询功能

查询功能是信息发布系统的重要功能，用户可以通过多种方式灵活查询数据。对于地理信息系统而言，用户可以通过属性信息查询空间数据，也可以通过地图上的空间位置查询对应的属性信息。例如，在本系统中可实现通过点击省份位置查询该省疫情感染人数、死亡人数等疫情信

息的空间查询功能。同时，也应具备通过输入具体地理地址，查询周边疫情感染人数及最近感染人员地址的属性查询功能。

(d) 地图导出

用户可选择将现有的地图显示切片或查询结果进行保存并导出成新的地图文件。同时，考虑开发更为全面的统计输出功能，如对发布的各种专题图信息进行统计，生成报表、柱状图、饼状图并与原始地图一起输出。

2. 维护管理相关功能

(a) 网站管理

用户都是通过浏览器访问地图，因此，该地理信息系统需要兼容各类浏览器。同时，考虑到本系统打算采用开源框架进行搭建，由于开源框架更新迅速，需安排专人定期对网站后台进行维护，包括数据库维护、网站信息发布系统维护等；同时，由于疫情信息具有很高的实时性，需要定期对地图发布的信息进行更新以确保其正确性和有效性。

(b) 服务配置与用户管理

管理员将待发布的.mxd 文件配制成服务，可以对地图的各种功能进行设置，如服务的共享性，服务最大排队等待时间等。对于各种开通的服务，如服务启动、暂停、编辑、停止和删除，需定期进行管理。

而用户管理则主要指用户信息添加、删除、修改与用户权限控制，即针对不同用户的请求做不同的权限设置。对于系统管理员级别的用户可以使用系统的配置及编辑功能，对服务器的系统参数、数据进行更新和修改，而一般的用户应只具备浏览、查询、统计制图等基础功能。

3. 疫情可视化相关功能

(a) 疫情感染与治愈情况多粒度评价及展示

针对全国区域内的多维疫情数据，应对数据进行分类整理，从不同角度展示疫情蔓延及治愈情况；与此同时，应对全国、省、市、区、街道等不同区域层次的疫情数据进行多粒度展示，保证用户获得感兴趣区域的特殊疫情数据。

(b) 疫情数据分析统计功能

应结合原始地图和地址信息，对现有疫情数据进行简单分析，生成报表、散点图、柱状图、饼状图等直观展示抗疫结果的图标，为后续疫情决策提供参考。

除了上述功能以外，新冠疫情 WebGIS 系统还应满足一下非功能性需求：

1. 系统性能：保证系统数据来源可靠、实时，对信息查询的响应时间不得超过 5 秒
2. 系统开放性：系统要求与不同的地理信息系统，不同格式的地理信息数据相兼容；系统提供的功能应该能够支持不同语言的操作系统。
3. 系统可移植性：整个系统应具有很高的可靠性，稳定性，满足在不同设备上连续稳定运行的要求。
4. 数据管理性能：系统应满足数据保存至当前一年半内的数据容量管理需求，支持未来三年内的数据增长的管理需求。

3 总体设计

3.1 系统框架设计

本新冠疫情 WebGIS 系统采用浏览器/服务器模式 (Browser / Server, B / S), 主要分为数据库层、逻辑服务层和 Web 端表现层 3 层结构。系统采取前后端分离的形式。前端采用了 Node.js 和 jquery 框架，搭配 ArcGIS API for Javascript、高德地图 API 等地图接口建立新冠疫情网页，实现了基本的地图显示功能和省、市、区不同层次的新冠疫情数据可视化、数据分析与数据查询等功能；逻辑服务层采用 ArcGIS Server 服务器发布地图服务，保证后台数据与前端网页的对接与瓦片地图发布，同时利用 Http-server 这一轻量级 Web 服务器发布本地静态网页；后端采用 PostgreSQL + PostGIS 数据库管理数据，并利用 Python 爬取丁香园网页和 CBNData 网页发布的疫情数据，实现了对疫情数据的快速获取与管理。

系统总体框架图如下图 3.1 所示。

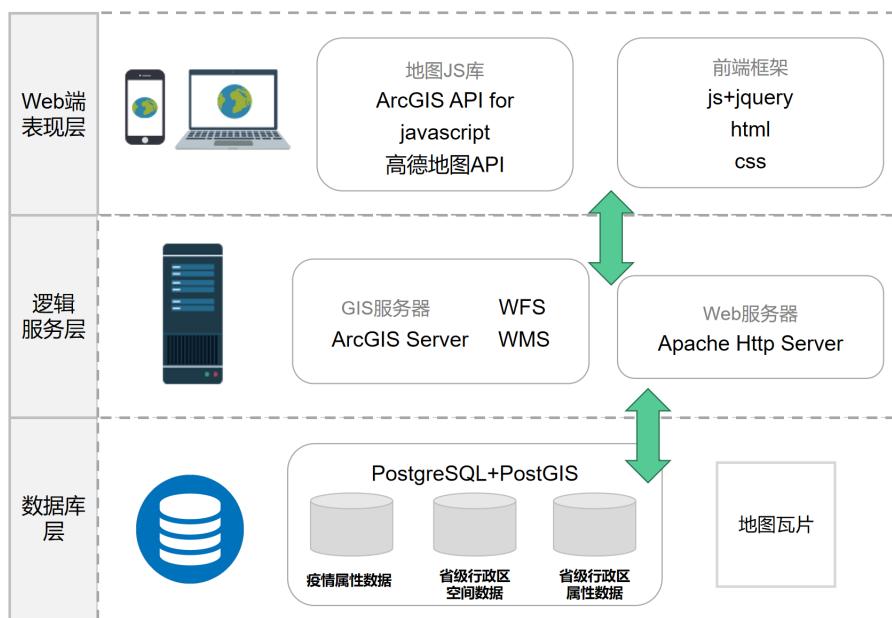


图 3.1. 系统总体框架图

系统开发环境及软件配置如下表 3.1 所示。

表 3.1. 系统开发环境及软件配置

用途	名称
操作系统	Windows 10
开发语言	JavaScript, Python 3.8.7
集成开发环境（前端+后端）	VSCode
地图服务器	ArcGIS Server 10.2
前端框架	Node.js 14.16.1, jquery 3.2.1
地图开发框架	ArcGIS API for Javascript 3.5, 高德地图 JS API
浏览器	Chrome
空间属性数据库	PostgreSQL 13 + PostGIS 3.1.1

3.2 功能模块设计

模块是采用结构化设计方法进行系统总体设计的一个重要概念。因此，在系统总体设计中，一项主要的工作就是确定功能模块结构。

本新冠疫情 WebGIS 系统主要分为三个功能模块：地图基本模块、全国疫情可视化模块和地区疫情可视化模块。系统功能模块设计结构图如下图 3.2 所示。

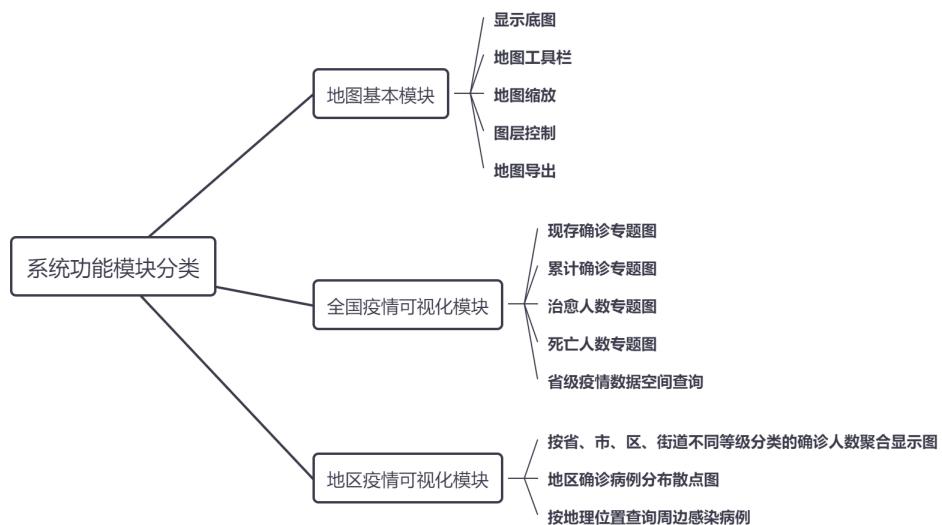


图 3.2. 功能模块设计结构图

“地图基本模块”实现了基本的地图浏览功能，如地图显示、放大、缩小、平移等。用户可以将地图移动到感兴趣的区域，根据感兴趣的区域大小或清晰程度放大、缩小显示范围。该系列基本功能可以通过地图右上角的工具栏控件实现。同时，通过“图层控制”控件实现不同专题图和多源数据（地图、影像）之间的切换。

“全国疫情可视化模块”中，本系统制作了多张疫情数据专题图，主要内容为根据不同颜色分层设色地显示各省份的疫情现存确诊人数、累计确诊人数、治愈人数、死亡人数等信息；同时，系统实现了通过点击地图上的省份位置查询该省份疫情信息的空间查询功能。

“地区疫情可视化模块”中，首先实现了按省、市、区、街道不同等级分类的确诊人数聚合图的显示，通过鼠标缩放可以查看不同等级不同区域的确诊人数。另外，系统提供了按地理位置查询周边感染病例的属性查询功能。系统首先获取用户当前地理位置，提示用户周边疫情情况；用户同样可以通过在搜索框输入特定地址，查询到该地址附近的疫情感染情况。

3.3 界面设计

系统共有一个主界面（如图 3.3）和一个跳转界面（如图 3.4）。主界面用于实现全国新冠疫情数据专题图的显示和空间查询功能，通过“图层控制”控件实现不同专题图的切换；通过主界面的“周边疫情”

按钮可以跳转到周边疫情查询界面，用于实现属性查询功能以及显示不同等级确诊人数聚合图的功能。

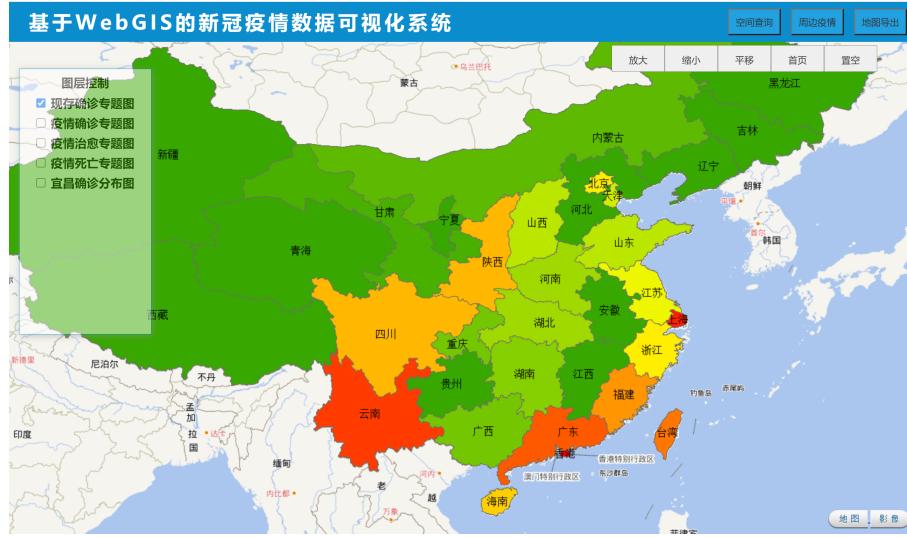


图 3.3. 主界面



图 3.4. 通过主界面的“周边疫情”按钮跳转到搜索界面

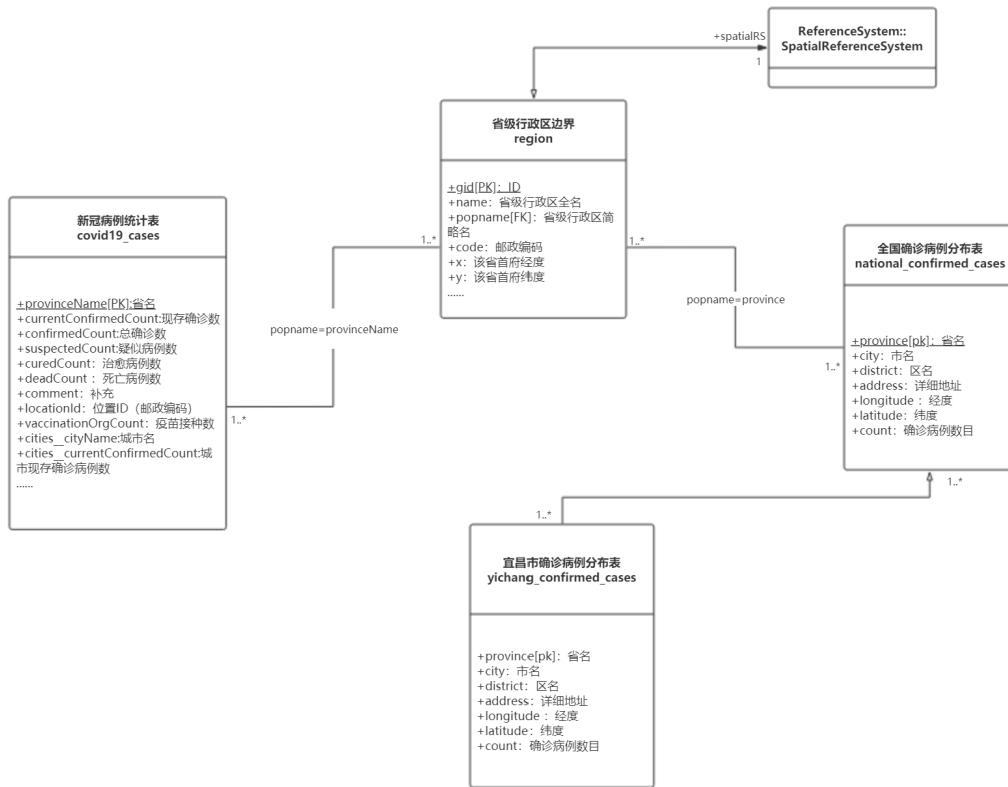
4 数据库设计

本新冠疫情 WebGIS 系统采用 PostgreSQL 作为属性数据库, PostGIS 支持空间数据部分, 利用 Python 爬取网页疫情数据, 与数据库建立连接并导入数据。

其中, 属性数据库部分主要管理四类表格: 新冠病例统计表 covid19_cases, 省级行政区边界 region, 全国确诊病例分布表 national_confirmed_cases 和全国确诊病例分布 national_confirmed_cases; 空间数据库主要管理省级行政区边界 region 的矢量数据。需要注明的是, 本系统中聚合显示图和周边病例查询需要采用的省、市、区详细地图 (包含街道地址等信息) 采用 API 形式导入, 并利用高德地图 API 的地理编码功能实现属性数据和空间数据的挂接。由于此部分仅包含前端开发的内容, 不利用数据库进行数据管理, 因此空间数据库设计较为简单。若未来有进一步开发需要, 再将详细的市、区边界及详细的街道

信息导入数据库中。

数据库概念模型的 UML 类图如下图 4.5;



*注：本系统中省、市、区详细地图（包含街道地址等信息）采用API形式导入，不采用数据库进行管理，因此数据库设计中不包含省、市、区详细地图的空间、属性数据。

图 4.5. 数据库概念模型

属性数据表及其关键字段如下表 4.2, 4.3, 4.4。

表 4.2. 新冠病例统计表

字段名	说明	数据类型
provinceName[PK]	省名	Text
currentConfirmedCount	现存确诊数	Integer
confirmedCount	总确诊数	Integer
suspectedCount	疑似病例数	Integer
curedCount	治愈病例数	Integer
deadCount	死亡病例数	Integer
comment	补充	Text
locationId	位置 ID (邮政编码)	Integer
vaccinationOrgCount	疫苗接种数	Integer
cities__cityName	城市名	Text
cities__currentConfirmedCount	城市现存确诊病例数	Integer

表 4.3. 省级行政区边界属性表

字段名	说明	数据类型
province[pk]	省名	Text
city	市名	Text
district	区名	Text
address	详细地址	Text
longitude	经度	Float
latitude	纬度	Float
count	确诊病例数目	Integer

表 4.4. 全国确诊病例分布表

字段名	说明	数据类型
gid[PK]	ID	Integer
name	省级行政区全名	Text
popname[FK]	省级行政区简略名	Text
code	邮政编码	Integer
x	该省首府经度	Float
y	该省首府纬度	Float

5 详细设计

本节将主要讲解系统“地区疫情”模块中“按地理位置查询周边感染病例”功能设计与实现。本功能基于高德地图 JS API 实现，生成了一个基于原生 js 的纯前端网页（search.html）。数据来源于 CBNData 的疫情确诊分布统计。

1. 数据获取

要实现这个功能，首先需要获取含有确诊人员地理位置信息的数据。为获取实时更新的确诊者数据，我们需要 html 的脚本中引入 Web API 中的 `fetch()` 方法，以爬取 <https://assets.cbndata.org/2019-ncov/data.json?ncovData=t> 网页的实时数据，得到.json 文件。

而后，根据.json 文件的解码方式，通过 Python 将其导出为.csv 文件并导入数据库 PostgreSQL 中，为后续功能做准备。需要说明的是，获取的数据当中部分数据缺失经纬度坐标，仅有地址信息；有部分缺失详细地址，仅有省市信息。对于仅有地址、没有经纬度坐标的数据，我们通过高德地图 API 的地理编码插件 Geocoder 进行地理编码，获取经纬度；对于仅有省市信息的数据，我们将其经纬度坐标简单设置为该市经纬度。

2. 当前位置周边疫情信息展示

我们希望获取用户当前的地理位置，并自动弹出提示页面，向用户展示当前地理位置的周边确诊情况，提醒用户做好防护。

该页面的底图及地址信息由高德地图 JS API 提供，对高德地图 JS API 的 `AMap.map` 类进行实例化，加载地图图层。

获取地理位置的功能由高德地图 JS API 的 `AMap.Geolocation()` 方法实现。`AMap.Geolocation` 插件融合了浏览器定位、高精度 IP 定位、安卓定位 sdk 辅助定位等多种手段，提供了获取当前准确位置、获取当前城市信息、持续定位（浏览器定位）等功能。基于 IP 地址的定位只能获取到城市级别信息，如果想获取到具体的位置就要借助浏览器定位。

周边确诊情况的展示由自己编写的 `showInfo()` 方法实现。我们通过 `new AMap.InfoWindow()` 创建信息窗口对象，并利用该对象的 `setContent()` 方法编写展示的内容，同时通过 `setPosition()` 定位信息窗口到当前位置。

3. 按地理位置查询周边疫情

按地理位置搜索的功能由高德地图 JS API 的 `AMap.AutoComplete()` 方法辅助实现。`AMap.AutoComplete` 根据输入关键字提示匹配信息，可将 Poi 类型和城市作为输入提示的限制条件。我们根据输入的完整地址或下拉框选取的地址获取该地址的经纬度数据。而后，我们以查询位置为圆心，1km 和 3km 为半径设定搜索范围，搜索周边 1 公里、3 公里内的确诊病例数。最后通过上述 `showInfo()` 方法和实例化的信息窗口对象 `infoWin` 提示当前位置及周边 1 公里内、3 公里内疫情信息。

该功能设计与实现流程图如下图 5.6：

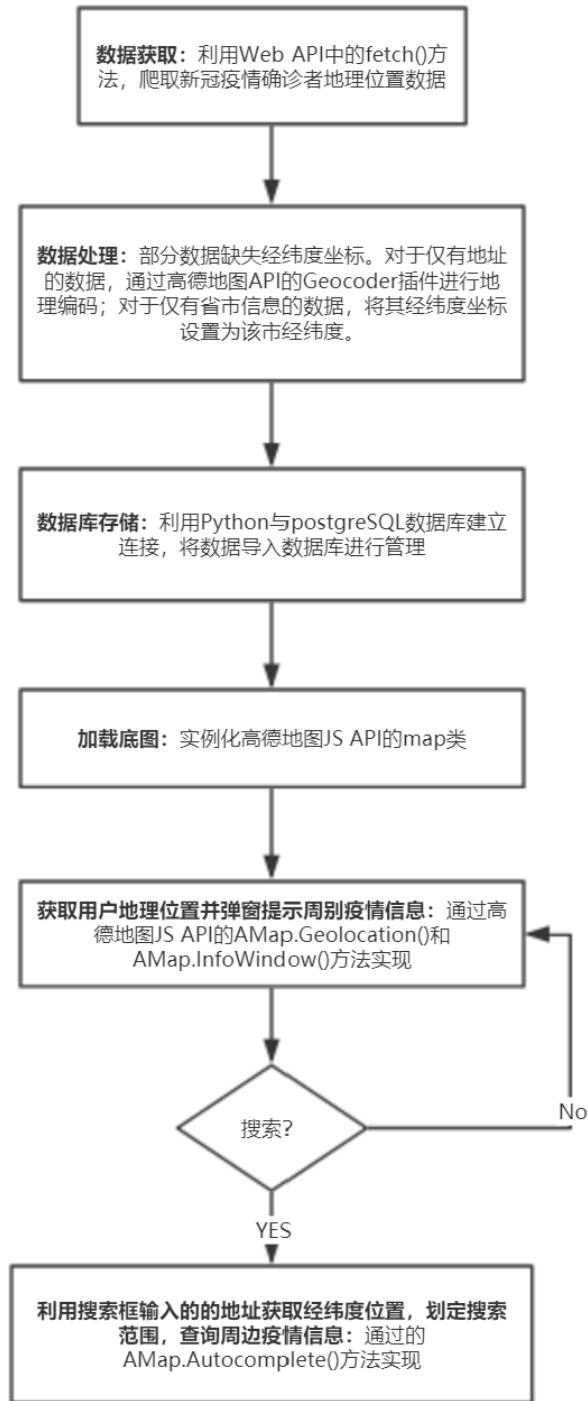


图 5.6. 详细设计流程图

6 功能实现

本节将介绍系统功能的具体实现，包括数据获取与处理、专题图制作、地图数据发布、主界面功能实现、搜索界面功能实现、静态页面部署几部分。系统总体功能模块设计见“总体设计”一节及图 3.2。

6.1 数据获取与处理

要实现图 3.2 中介绍的系统功能，我们需要获取以下几类信息：含现存确诊病例数、累计确诊病例数、死亡人数、治愈人数等信息的新冠疫情统计数据；全国省级行政区边界及属性信息；全国累计确诊病例位置及分布。省级行政区边界的 shp 文件易于获取，此处不再赘述。

为获取现有的新冠疫情统计数据，我编写了 crawl.py 文件，利用 python 的 requests 库中的 requests.get(url) 方法，爬取了丁香园推出的全球新冠肺炎疫情地图网页 (<https://ncov.dxy.cn/ncov/h5/view/pneumonia>)，辅助后续专题图的制作。爬虫部分关键代码如下图 6.7：此外，我们还需要获取带位置信息的全国累计确

```
import requests
import pprint
import re
import json

def write_to_file(item):
    with open('ylqing.json', 'w', encoding='utf-8') as f:
        f.write(json.dumps(item, indent=4, ensure_ascii=False))
    f.close()

result = requests.get(
    'https://ncov.dxy.cn/ncov/h5/view/pneumonia?scene=2&clicktime=1579583352&enterid=1579583352&from=timeline&isappinstalled=1')
url_text = result.content.decode()

url_result = re.search(r'window.getAreaStat = (.*?);}])catch', url_text, re.S)
texts = url_result.group()

texts = texts.replace('window.getAreaStat = ', '')
texts = texts.replace(')catch', '')
c = json.loads(texts)
# pprint.pprint(c)
write_to_file(c)
```

图 6.7. 爬取数据部分关键代码

诊病例分布。利用 JS Web API 中的 fetch() 方法获取 <https://assets.cbnadata.org/2019-nCoV/data.json?ncovData=t> 网页中实时更新的确诊病例位置。

获取的数据文件格式为.json，将其转为 csv 格式存储。而后，同样通过 python 与 PostgreSQL 数据库中创建的 covid19 数据库建立连接，将 csv 文件存储到 covid19 数据库中，关键代码如下图 6.8。同时，利用 postGIS Manager 将省级行政区的空间数据 (shapefile) 导入到 covid19 数据库中，导入后结果如下图 6.9。

```
import psycopg2
from sqlalchemy import create_engine
import pandas as pd
import os

# csv文件所在的文件夹
# 如果只有一个文件
# Folder_Path = r'C:\Users'

table1 = pd.read_csv('china_data\\yichang_comfirmed_cases.csv', header=0)

connect = create_engine(
    'postgresql+psycopg2://'+ 'postgres' + ':' + '123' +
    '@127.0.0.1' + ':' + str(5432) + '/' + 'covid19')

pd.io.sql.to_sql(table1, name='yichang_comfirmed_cases', con=connect, index=False,
                 if_exists='replace', chunksize=1000)

connect.dispose()
```

图 6.8. 数据库连接部分关键代码

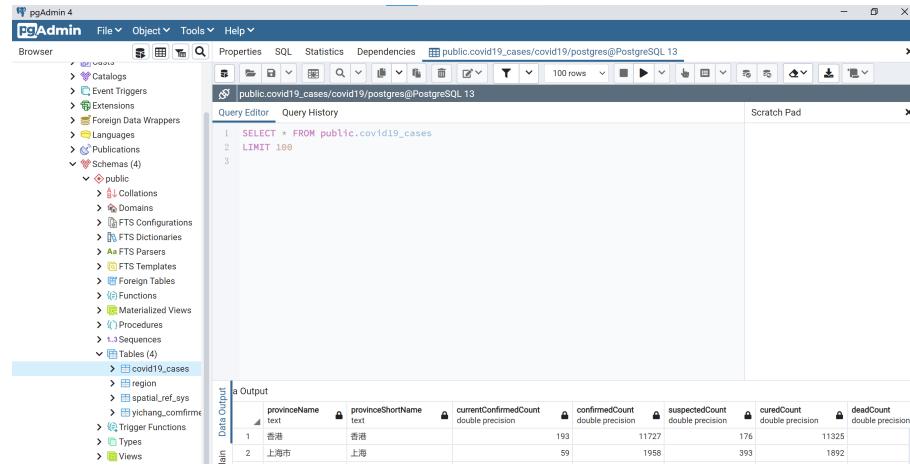


图 6.9. covid19 数据库

6.2 专题图制作

为制作全国各省疫情现存确诊病例数、累计确诊病例数、治愈人数、死亡人数几张专题图，我们利用 ArcGIS 的 join and relate 功能，将各省疫情数据表与省级行政区边界表利用省名这一关键字进行连接，然后根据各省的现存确诊病例数、累计确诊病例数、治愈人数、死亡人数划分出几个组，每组对应的矢量多边形设置不同的颜色，分层显示各省疫情情况（绿色代表轻微，红色代表严重，颜色由绿到红过渡），并将各省省名作为文本注记添加到对应位置。专题图示例如图 6.10 所示。最后将各张专题图作为 ArcGIS 图层输出。

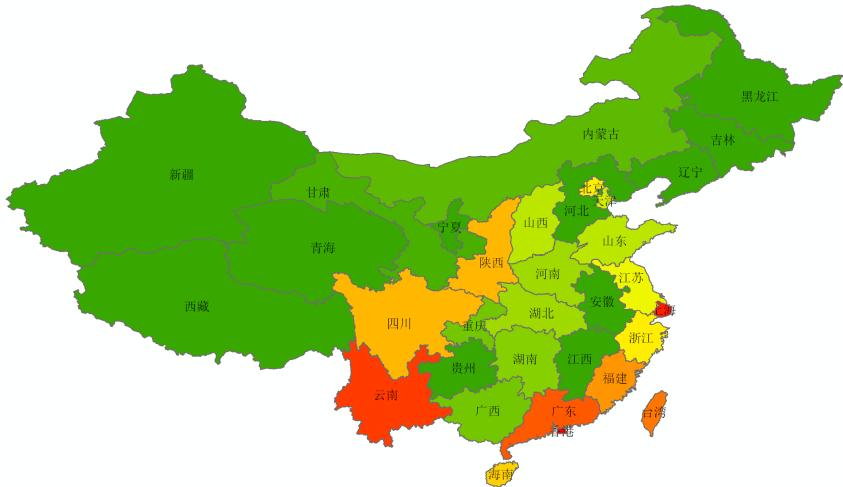


图 6.10. 现存确诊专题图

由于时间限制，本系统的这个空间分析功能在后端 ArcGIS 上实现，后续考虑持续开发系统功能，结合实时更新的疫情数据，将本空间分析功能移植到 WebGIS 上。

6.3 地图数据发布

本系统通过 ArcGIS for Server 服务器进行地图数据的发布。

利用 ArcGIS Server Manager 将上述的 ArcMap 输出的专题图部署到 ArcGIS Server 上，方便后续

WebGIS 系统通过引用 URL 对各图层的调用和现实。例如，将现存确诊专题图部署到服务器上，示例结果如图 6.11。



图 6.11. 发布现存确诊专题图

6.4 主界面功能实现

主界面主要实现以下几个功能：(1) 基本地图功能；(2) 疫情专题图的显示；(3) 某市确诊病例散点图制作及显示（以宜昌市为例）；(4) 省级疫情数据空间查询。

1. 基本地图功能

基本地图功能主要通过系统右上角工具栏及右下角的控件实现。地图底图采用的是天地图，首先将天地图矢量地图图层、影像地图图层、标注图层的.js 文件下载到本地并在脚本中实例化对象。通过编写 Showvce() 和 ShowImage() 并对地图/影像切换控件创建 onclick 事件实现矢量地图和影影像地图的切换。关键代码如下图 6.12；切换效果如下图 6.13,6.14。

```

vcemap = new TDTLayer();
imgmap = new TDTImagesLayer();
label = new TDTAnnoLayer();
china_base = new ArcGISDynamicMapServiceLayer("http://
    id: "china_base"
);
SHGIS.map = new Map("yongxinmap", SHGIS.mapOptions);
SHGIS.map.removeAllLayers();
SHGIS.Showvce();
});

//影像地图
SHGIS.Showimage = function () {
    isload = true;
    SHGIS.map.removeAllLayers();
    SHGIS.map.addLayer(imgmap);
    SHGIS.map.addLayer(label);
};

//矢量地图
SHGIS.Showvce = function () {
    isload = false;
    SHGIS.map.removeAllLayers();
    SHGIS.map.on("zoom-end", function () {
        if (isload == false) {
            if (SHGIS.map.getZoom() > 15) {
                SHGIS.map.addLayer(vcemap);
                SHGIS.map.addLayer(label);
                SHGIS.map.addLayer(china_base);
                isload == true;
            }
        }
    });
    SHGIS.map.addLayer(vcemap);
    SHGIS.map.addLayer(label);
    SHGIS.map.addLayer(china_base);
};

```

图 6.12. 底图加载部分代码



图 6.13. 加载矢量地图底图

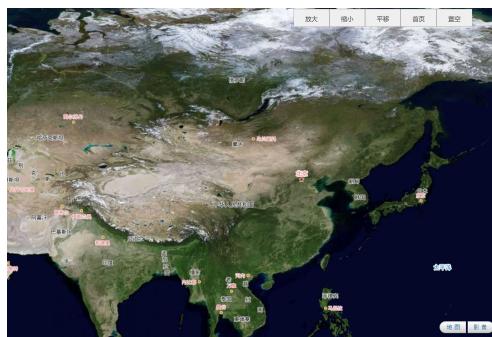


图 6.14. 加载影像地图底图

地图的放大、缩小、平移、返回首页、置空功能通过右上角工具栏控件实现，该控件的 onclick 事件会触发 toolClickHandle() 函数。其中，放大、缩小、平移功能可以通过鼠标滚轮和鼠标拖动实现，也可以点击工具栏对应控件后，在图中选取感兴趣的区域并缩放到该位置，它们通过 ArcGIS API for Javascript 中 Navigation 类的 ZOOM_IN、ZOOM_OUT、PAN 等属性实现。首页功能将返回到原始地图中心点 (110.24, 35.25)，而置空功能主要用于清除图层及空间查询的结果。关键代码如下图 6.15。

```
toolClickHandle = function (item) {
    require([
        "esri/map", "esri/toolbars/navigation", "esri/geometry/Point", "esri/SpatialReference", "dojo/domReady!"
    ], function (Map, Navigation, Point, SpatialReference) {
        if (navToolbar === null) {
            navToolbar = new Navigation(SHGIS.map);
        }

        navToolbar.deactivate();
        if (item === '1') {
            navToolbar.activate(Navigation.ZOOM_IN);
            return;
        }
        if (item === '2') {
            navToolbar.activate(Navigation.ZOOM_OUT);
            return;
        }
        if (item === '3') {
            navToolbar.activate(Navigation.PAN);
            return;
        }
        if (item === '4') {
            var point = new Point([110.24, 35.25], new SpatialReference({
                wkid: 4326
            }));
            SHGIS.map.centerAndZoom(point, 3);
            return;
        }
        if (item === '5') {
            navToolbar.deactivate();
            SHGIS.map.graphics.clear();
            if (SHGIS.map.getLayer('chinafeaturelayer')) {
                SHGIS.map.removeLayer(SHGIS.map.getLayer('chinafeaturelayer'));
            }
            return;
        }
    });
}
```

图 6.15. 地图工具箱代码

此外，右上角的“地图导出”控件会触发 mapPrintOutClickHandle() 函数，通过 ArcGIS API for Javascript 的 PrintTemplate 类实现地图导出功能。关键代码如下图 6.16。

```

mapPrintOutClickHandle = function () {
    require(["esri/tasks/PrintTask", "esri/tasks/PrintTemplate", "esri/tasks/PrintParameters"], function (PrintTask,
        var url = 'http://123.15.42.106:6080/arcgis/rest/services/Utilities/PrintingTools/GPServer/ExportToWebMap'
        var printTask = new PrintTask(url);

        var template = new PrintTemplate();
        template.exportOptions = {
            width: 500,
            height: 400,
            dpi: 96
        };
        template.format = "PDF";
        template.layout = "MAP_ONLY";
        template.preserveScale = false;
        var params = new PrintParameters();
        params.map = SHGIS.map;
        params.template = template;
        params.template = template;
        alert("地图正在导出，请稍后...");
        printTask.execute(params, function (evt) {
            window.open(evt.url, "_blank");
        });
    });
}

```

图 6.16. 地图导出部分代码

2. 疫情专题图的显示

疫情专题图的显示通过“图层控制”checkbox 控件实现，该控件的 onclick 事件会触发 layerControlClick() 函数，该函数的传入参数 item 表示需要加载的图层名称。由于之前已经完成了专题图的制作，此处只需要将 GIS 服务器发布的地图图层加载到地图底图上即可，通过 ArcGIS API for Javascript 的 addLayer() 函数实现。例如，加载现存确诊疫情专题图，效果如下图 6.17 所示。



图 6.17. 加载现存确诊疫情专题图

3. 某市确诊病例散点图制作及显示（以宜昌市为例）

宜昌市确诊病例散点图同样通过“图层控制”控件实现，该控件的 onclick 事件会触发 loadScatterPoints() 函数。该函数通过 ArcGIS API for Javascript 的 geometry/Point 类，获取湖北省宜昌市每个确诊病例对应的经纬度坐标，创建点对象，并通过 Graphic 类创建图上的点，将点对象显示在地图中对应的经纬度位置。关键代码如下图 6.18；散点图显示效果如下图 6.19。

```

loadScatterPoints = function () {
    require(["esri/geometry/Point", "esri/SpatialReference", "esri/symbols/SimpleMarkerSymbol", "esri/graphic", "esri/sym
        var simpleMarkerSymbol = new SimpleMarkerSymbol(SimpleMarkerSymbol.STYLE_CIRCLE, 4,
            new SimpleLineSymbol($simpleLineSymbol(STYLE_SOLID,
                new Color([255, 0, 0]), 1),
                new Color([255, 0, 0])));
        ygData.map(item => {
            var point = new Point(item.longitude, item.latitude, new SpatialReference({
                wkid: 4326
            }));
            var graphic = new Graphic(point, simpleMarkerSymbol);
            SHGIS.map.graphics.add(graphic);
        })
        var point = new Point([111.5, 30.75], new SpatialReference({
            wkid: 4326
        }));
        SHGIS.map.centerAndZoom(point, 8)
    })
}

```

图 6.18. 散点图部分代码

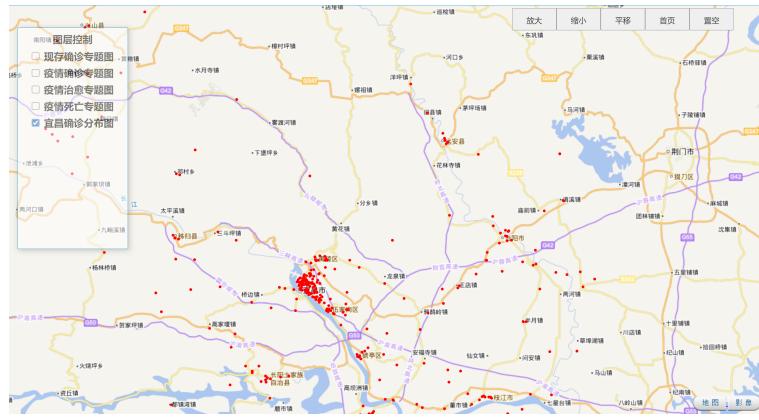


图 6.19. 宜昌市确诊病例分布散点图

4. 省级疫情数据空间查询

省级疫情数据空间查询功能通过“空间查询”button 空间实现，该控件的 onclick 事件会触发 menuClickHandle() 函数。该函数通过 ArcGIS API for Javascript 的 InfoTemplate 类，创建信息窗口，通过 InfoTemplate 的 setContent() 设置显示每个省的现存确诊数、累计确诊数、死亡人数、治愈人数等信息，并在点击地图中省份的空间位置时进行定位并弹出信息窗体。关键代码如下图 6.20；空间查询效果示例如下图 6.21。

```
menuClickHandle = function () {
    require(["esri/layers/FeatureLayer", "esri/InfoTemplate"], function (FeatureLayer, InfoTemplate) {
        if (!SHGIS.map.getLayer("chinafeatureLayer") === undefined) {
            var infoTemplate = new InfoTemplate("${provincesh}");
            var featureLayer = new FeatureLayer("http://123.15.42.106:6080/arcgis/rest/services/sqbd/test_china/MapServer/1");
            featureLayer.setLayerName("chinafeatureLayer");
            featureLayer.mode = FeatureLayer.MODE_ONDEMAND;
            infoTemplate.setTitle("疫情数据");
            infoTemplate.setContent("现存确诊: ${currentCon}<br>确诊数量: ${confirmedC}<br>疑似数量: ${suspectedC}<br>治愈数量: ${curedC}<br>死亡数量: ${deadC}");
            SHGIS.map.addLayer(featureLayer);
        }
    })
}
```

图 6.20. 空间查询部分代码

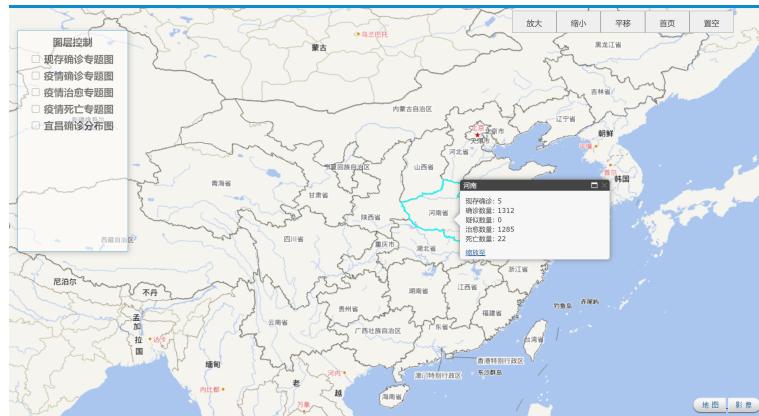


图 6.21. 查询河南省疫情信息

6.5 搜索界面功能实现

由于需要获取详细的街道地址信息、调用高德地图 API 的地理编码功能，搜索界面 (search.html) 引入了高德地图 API 进行实现。本界面主要实现以下功能：(1) 显示按省、市、区、街道不同等级分类的确诊人数聚合图；(2) 按地理位置查询周边感染病例。其中，数据获取过程和功能 (2) 的设计与实现已在“详细设计”一节进行详细介绍，此处不再赘述。

而功能 (1) 首先导入高德地图底图和原始数据.json 文件后。而后通过自己编写的 renderCluster() 方法将原始数据点根据经纬度聚类。再通过实例化高德地图 API 的 Marker 类为各个聚类创建 markers，最后通过实例化高德地图 API 的 MarkerClusterer 类显示按照经纬度聚类的聚类标签。关键代码如下图 6.22；聚合显示效果图如下图 6.23。

```
function renderCluster(points) {
    for (var i = 0; i < points.length; i += 1) {
        if ((points[i]['longitude']) && (points[i]['latitude'])) {
            markers.push(new AMap.Marker({
                position: [points[i]['longitude'] - 0, points[i]['latitude'] - 0],
                content: '

' + points[i].city + '

',
                offset: new AMap.Pixel(-15, -15)
            }));
        } else {
            // console.log(points[i].city)
        }
    }
    addCluster();
}

function addCluster() {
    var sts = [
        {url: "https://a.amap.com/jsapi_demos/static/images/blue.png",
        size: new AMap.Size(32, 32),
        offset: new AMap.Pixel(-16, -16)},
        {url: "https://a.amap.com/jsapi_demos/static/images/green.png",
        size: new AMap.Size(32, 32),
        offset: new AMap.Pixel(-16, -16)},
        {url: "https://a.amap.com/jsapi_demos/static/images/orange.png",
        size: new AMap.Size(36, 36),
        offset: new AMap.Pixel(-18, -18)},
        {url: "https://a.amap.com/jsapi_demos/static/images/red.png",
        size: new AMap.Size(48, 48),
        offset: new AMap.Pixel(-24, -24)},
        {url: "https://a.amap.com/jsapi_demos/static/images/darkRed.png",
        size: new AMap.Size(48, 48),
        offset: new AMap.Pixel(-24, -24)}
    ];
    cluster = new AMap.MarkerClusterer(map, markers, {
        gridSize: 50
    });
}
```

图 6.22. 聚合显示部分关键代码

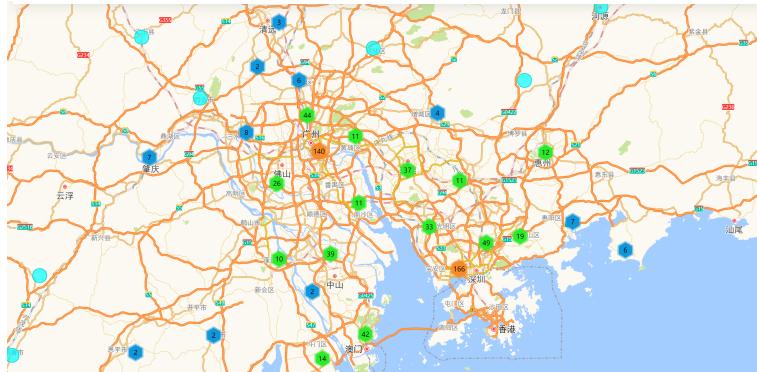


图 6.23. 聚合显示效果图

功能 (2) 的搜索功能及提示窗口如示意图 6.24。

6.6 静态页面部署

最后，在完成了主界面 (main.html) 和搜索界面 (search.html) 两个静态界面的编写后，我们需要把它们部署到本地服务器上，才能在浏览器中进行显示。

本系统采用的是 http-server 服务器。http-server 是一个基于 node.js 的轻量级 http 服务器。它可以使任意一个目录成为服务器的目录，完全抛开后台的沉重工程，直接运行想要的 js 代码，其轻量级、功

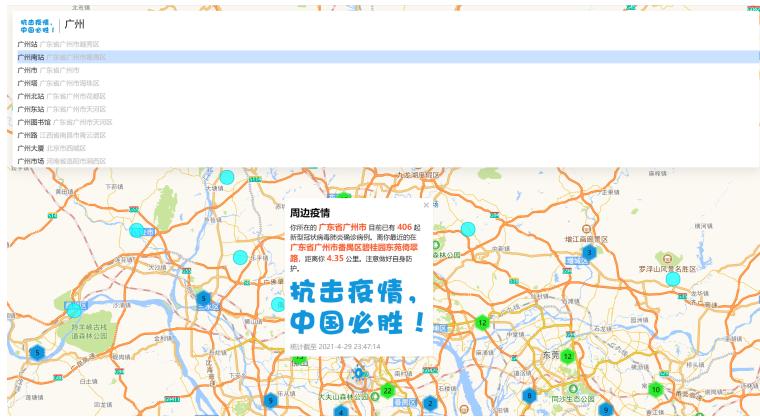


图 6.24. 按地理位置查询周边感染病例

能简捷的特性也适用于本简单 WebGIS 系统静态页面的部署。最后的界面效果见“界面设计”一节中的图 3.3, 图 3.4。

7 心得体会

尽管时间紧张，但还是决定在本次报告的最后附上一点简短却真诚的心得体会。

本次 WebGIS 系统是我第一次接触前端开发的内容，之前只简略的接触过少许 javascript+html+css 网页编程的内容，用起来完全算不上得心应手，完整开发一整个 WebGIS 系统更是无从谈起。因此，本次 WebGIS 系统的搭建于我而言完全是从 0 到 1 的过程。

在开始编写这个系统的一周里，我几乎可以说是毫无头绪，不仅全然不熟悉与 javascript 框架相关的内容，对于前端开发需要用到的工具，如服务器、Web 接口、网络爬虫等内容更是一窍不通。之前学习、使用过的编程语言基本都限于科学计算和数据分析的范畴，对此次网络程序开发帮助不大。好在通过查阅各类轻量级 WebGIS 系统开发的书籍、博客、视频等等资料，利用一周的时间总算是勉强厘清整个系统开发的流程，决定了需要使用的框架和工具。

完成整个系统开发又花了整整一周的时间。尽管只是少许简单的入门功能，对于初次接触前端开发的我来说每一步都走得如履薄冰。后来，我决定把困难的任务拆分成一个个子模块，告诉自己，一天实现能一个模块的功能就算成功。在完成了主界面的开发后，我对主界面的现有功能和界面设计感到不甚满意。且主界面中的数据无法自动完成数据更新，需要人工处理，实用性不强。于是，我又依据 github 中制作的一些疫情地图，再根据文档学习了一些高德地图 JS API 的功能，在主界面的基础上增加了一个用于属性查询和分类聚合显示的 search 界面，这个界面是依靠 js 脚本的纯前端界面，里面的数据可以实现实时更新。尽管由于时间有限，我尚未将这个框架移植到原本开发的主界面中，但既已完成第一步，后面的再调整工作相比不会太困难。

可惜的是，由于本人在网页编程和前端开发方面的基础实在太薄弱，且能真正投入这个系统开发的时间始终有限，在本系统开发中仍然留下了许多的遗憾。例如，我一直对网页的界面布局和可视化不太满意，这个仿佛诞生于世纪初的 Windows XP 系统上的笨拙网页显得有些落伍，不美观，也不优雅，且在数据分析方面的功能似乎也有些缺乏。本来想引入 ECharts 对网页界面和数据分析效果进行一些改进，无奈时间限制也未能付诸实践。希望日后有机会再对这个系统一些功能和代码框架做进一步的完善。

总的来说，通过这次 WebGIS 系统开发，我初步接触到了 Web 开发的内容，基本掌握了 Web 开发的

整体流程，迈出了系统开发实践的第一步，也算是收获颇丰。尽管这个系统与诸如 Johns Hopkins 大学开发的疫情网页那样优雅、可用、cutting-edge 的新冠疫情可视化系统仍存在着天壤之别，但我仍希望能以此大作业为契机，以后能够持续地接触前端开发的相关内容，早日拥有成为项目经理的能力。本系统做得不足的地方也望老师和使用者海涵。

衷心感谢赵前胜与申丽丽老师为本门课教学实践付出的心血。

8 附件材料说明

表 8.5. 附件材料说明

名称	是否提交	数量	说明
报告文档	√	1	文件名：2018302141300-顾宇
录屏	√	1	文件名：2018302141300-顾宇-录屏
代码	√	1	压缩包中的 code 文件夹。包含 main.html、search.html 两个 html 网页（使用时需先部署到本地服务器）；css 文件夹包含离线 css 样式表；js 和 static 文件夹包括离线 jquery 和 bootstrap 库以及天地图底图；python 文件夹包含用 python 编写的代码；imgs 文件夹包含引用的图片；data 文件夹包含所需要的数据。
数据	√	1	压缩包中的 data 文件夹

封底

课程报告承诺：我承诺该课程报告所有内容由本人独立完成，没有抄袭他人的成果，如有违反，本人愿意承担所有后果，接受学校相关的纪律处分。

电子签名（手写拍照）： 顾宇

日期：2021. 04. 30

报告评语：

第一部分得分	
第二部分得分	
第三部分得分	
第四部分得分	
总评成绩	
指导教师：	
年 月 日	