

**物联网项目说明文档**

**Covid-19疫情展示及预测系统**

**课程名称：物联网应用概论**

**院 系：软件学院**

**小组成员：陈诺 2052538**

**倪韵旖2050235**

**罗敏琪 2054402**

**杨明月 2053303**

**白钰 2053171**

**二零二二年十二月**

1. **项目选题**
2. **概述**

我国疫情防控形势持续向好、生产生活秩序加快恢复的态势不断巩固和拓展，统筹推进疫情防控和经济社会发展工作取得积极成效，这样的成绩得益于全国上下的共同努力和付出。但随着全国各地区相继解封，企业先后复工，国内外人口流动和聚集增加带来的疫情反弹风险依然存在，特别是国际疫情快速蔓延带来的输入性风险增加，这意味着我们的疫情保卫战还未结束，虽胜利在即，但仍要慎终如始。

为加强新冠状病毒肺炎疫情防控工作，做好外来及流动人员管理、企业复工、学校开学、社区管理、交通管控等防疫工作，我们本次的项目希望实现的是疫情防控数据可视化平台。

疫情防控数据可视化平台对接多方数据资源，包括丁香网数据和传感器数据，经过数据模型研判分析，将不同类型、不同维度的疫情维稳数据以不同的方式展示出来，文图结合，清晰、直观展示疫情态势监测数据。

本平台通过数据和、折线图和柱状图的方式展现全球、全国和各个省份的实时数据，并作出预测。

1. **需求说明**
   1. **目标用户**

主要满足3类用户：**大众用户、政企用户**和患者用户。其中主要是前2者。大众用户是指像我们普遍受此前疫情影响生活、工作等方面的大众群体。政企用户是指政府和企业机构，同样受此次疫情影响，对机构的运作肯定也是有影响的，他们需要基于此做合适的决策，保障企业和员工的安全。患者用户是指受此次疫情传播切身影响到的用户，包括确诊、疑似、接触、被动隔离等，这类用户对医学信息的获取会更为迫切。

* 1. **需求**
     1. 发布端(传感器部分):模拟全球及中国地区数据发布；
     2. 数据网络传输部分:发布和订阅数据通过网络(定义报文或 MOTT 报文格式)传输发布到服务，代理端及订阅端；
     3. 数据处理部分:服务端，代理端或发布端，接收，存储并展示联 COVID-19相关疫情数据，并具备简单分析预测数据功能；
     4. 服务端，代理端和采集发布端可以部署在局域网内不同主机；
     5. **信息范围**

**我们主要展示累计确诊、剩余疑似、累计治愈和累计死亡的人数。在空间上，我们分别展示世界范围、全国和各省份的信息；在时间上，展示当天的相关数据。**

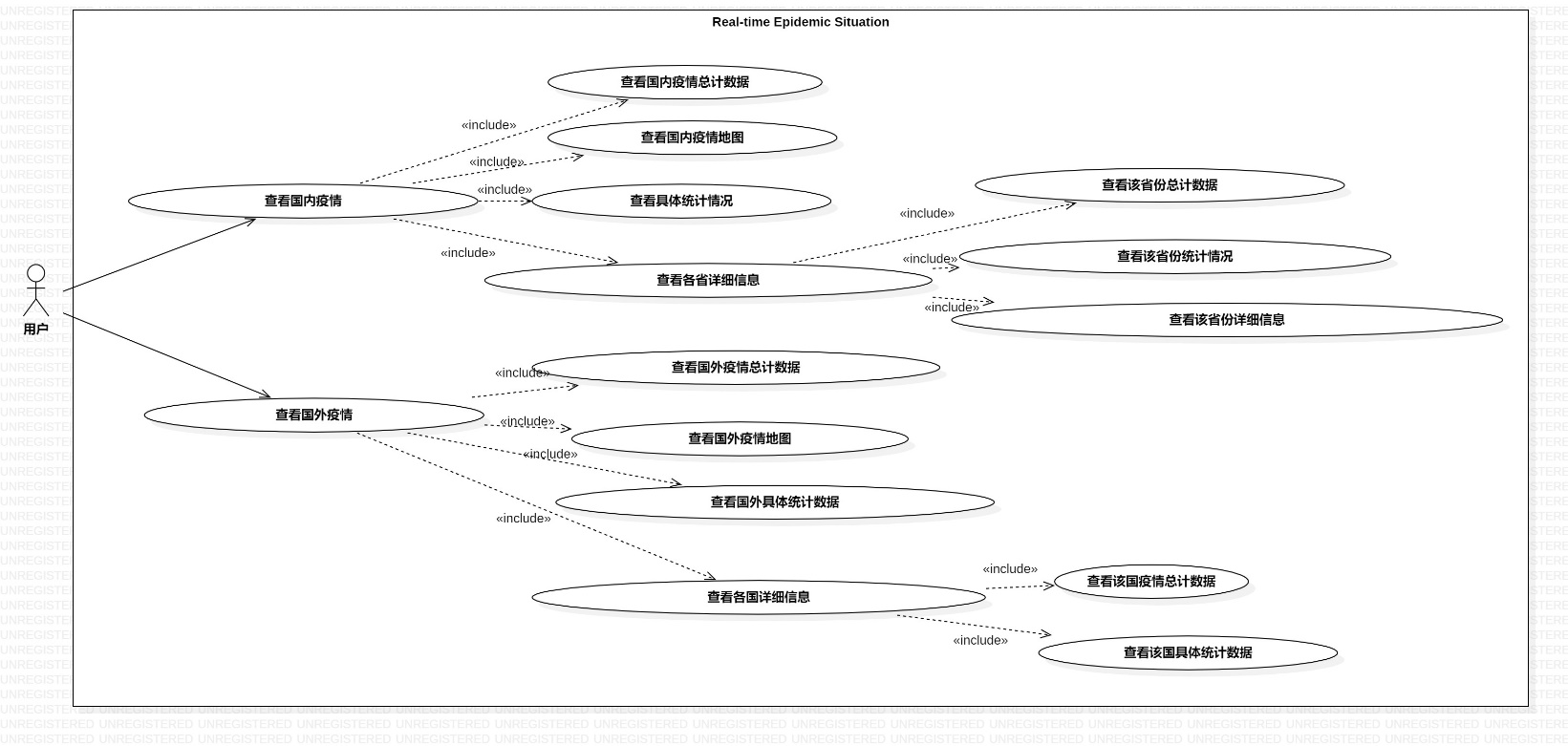
* + 1. **展示形式**

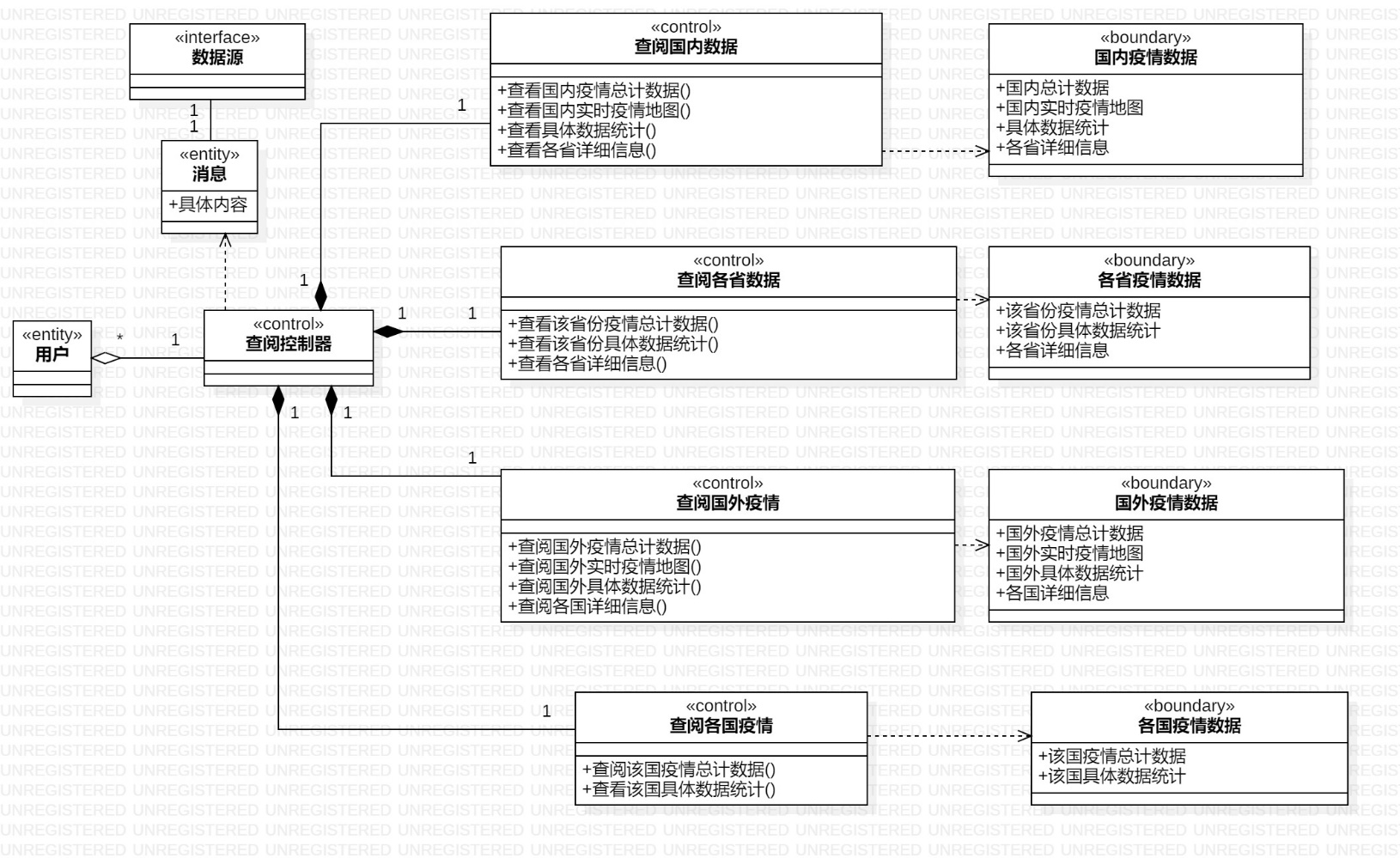
**主要包括三种数据展示形式：文本、疫情地图及图表。**

**以全国页面为例。疫情地图会以省份为单位进行划分，以不同颜色标明各省份的现有确诊人数，如下图1所示。图表包括柱状图和折线图，其中折线图可以以日期为单位展示四种数据的变化趋势。**

* 1. **UML**

项目用例图：



项目类图：

1. **MQTT**
   1. **MQTT特点**

MQTT 是一个客户端服务端架构的发布/订阅模式的消息传输协议。它的设计思想是轻巧、开放、 简单、规范，因此易于实现。这些特点使得它对很多场景来说都是很好的选择，包括受限的环境如机器与机器的通信（M2M）以及物联网环境（IoT），这些场景要求很小的代码封装或者网络带宽非常昂贵。

本协议运行在 TCP/IP，或其它提供了有序、可靠、双向连接的网络连接上。

它有以下特点：

• 使用发布/订阅消息模式，提供了一对多的消息分发和应用之间的解耦；

• 消息传输不需要知道负载内容；

• 提供三种等级的服务质量：

• “最多一次”，尽操作环境所能提供的最大努力分发消息。消息可能会丢失。例如，这个 等级可用于环境传感器数据，单次的数据丢失没关系，因为不久之后会再次发送；

• “至少一次”，保证消息可以到达，但是可能会重复；

• “仅一次”，保证消息只到达一次。例如，这个等级可用在一个计费系统中，这里如果 消息重复或丢失会导致不正确的收费；

• 很小的传输消耗和协议数据交换，最大限度减少网络流量 ；

• 异常连接断开发生时，能通知到相关各方。

* 1. **MQTT原理说明**
     1. MQTT协议实现方式
* 实现MQTT协议需要：客户端和服务器端；
* MQTT协议中有三种身份：发布者（Publish）、代理（Broker）（服务器）、订阅者（Subscribe）。其中，消息的发布者和订阅者都是客户端，消息代理是服务器，消息发布者可以同时是订阅者；
* MQTT传输的消息分为：主题（Topic）和负载（payload）两部分：Topic，可以理解为消息的类型，订阅者订阅（Subscribe）后，就会收到该主题的消息内容（payload）；payload，可以理解为消息的内容，是指订阅者具体要使用的内容。
  + 1. 网络传输与应用消息

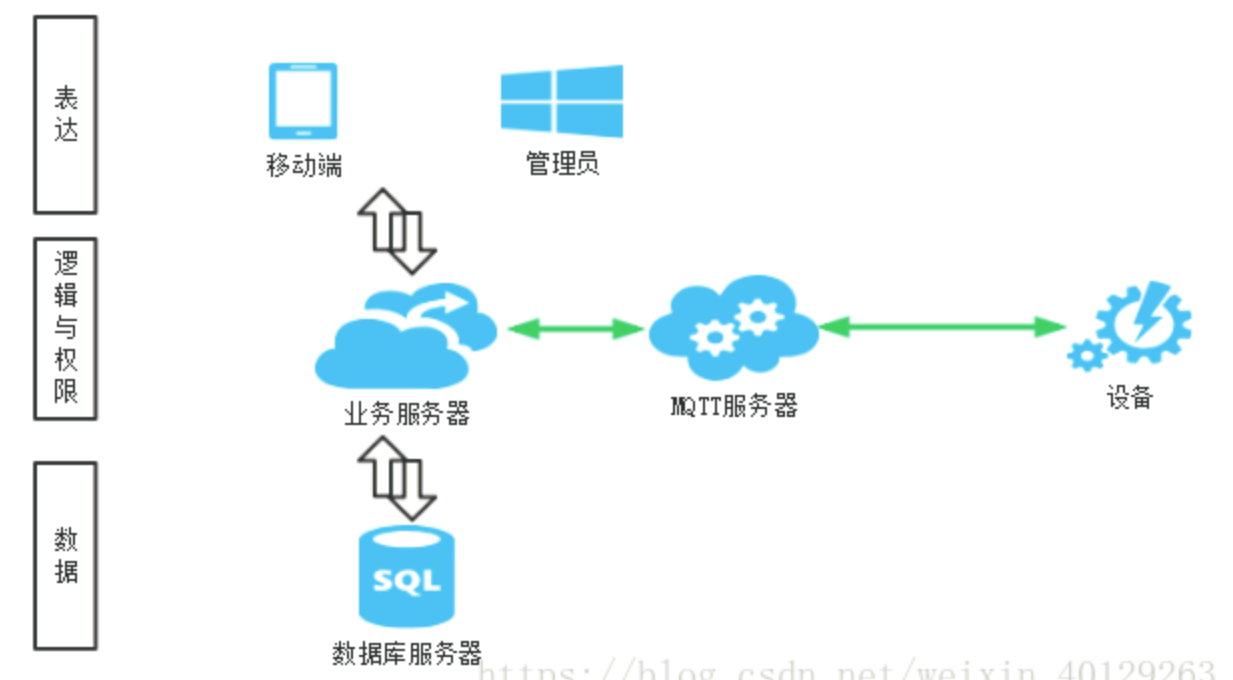
MQTT会构建底层网络传输：它将建立客户端到服务器的连接，提供两者之间的一个有序的、无损的、基于字节流的双向传输。当应用数据通过MQTT网络发送时，MQTT会把与之相关的服务质量（QoS）和主题名（Topic）相关联。

* + 1. MQTT客户端

一个使用MQTT协议的应用程序或者设备，它总是建立到服务器的网络连接。客户端可以：

* 发布其他客户端可能会订阅的信息
* 订阅其它客户端发布的消息
* 退订或删除应用程序的消息
* 断开与服务器连接
  + 1. MQTT服务端

MQTT服务器以称为“消息代理”（Broker），可以是一个应用程序或一台设备。它是位于消息发布者和订阅者之间，它可以：

* 接受来自客户的网络连接
* 接受客户发布的应用信息
* 处理来自客户端的订阅和退订请求
* 向订阅的客户转发应用程序消息
  + 1. MQTT协议中的订阅、主题、会话

订阅（Subscription）：订阅包含主题筛选器（Topic Filter）和最大服务质量（QoS）。订阅会与一个会话（Session）关联。一个会话可以包含多个订阅。每一个会话中的每个订阅都有一个不同的主题筛选器；

会话（Session）：每个客户端与服务器建立连接后就是一个会话，客户端和服务器之间有状态交互。会话存在于一个网络之间，也可能在客户端和服务器之间跨越多个连续的网络连接。

主题名（Topic Name）：连接到一个应用程序消息的标签，该标签与服务器的订阅相匹配。服务器会将消息发送给订阅所匹配标签的每个客户端。

主题筛选器（Topic Filter）：一个对主题名通配符筛选器，在订阅表达式中使用，表示订阅所匹配到的多个主题。

负载（Payload）：消息订阅者所具体接收的内容。

* + 1. MQTT协议中的方法

MQTT协议中定义了一些方法（也被称为动作）， 来于表示对确定资源所进行操作。 这个资源可以代表预先存在的数据或动态生成数据，这取决于服务器的实现。通常来说，资源指服务器上的文件或输出。

Connect，等待与服务器建立连接；

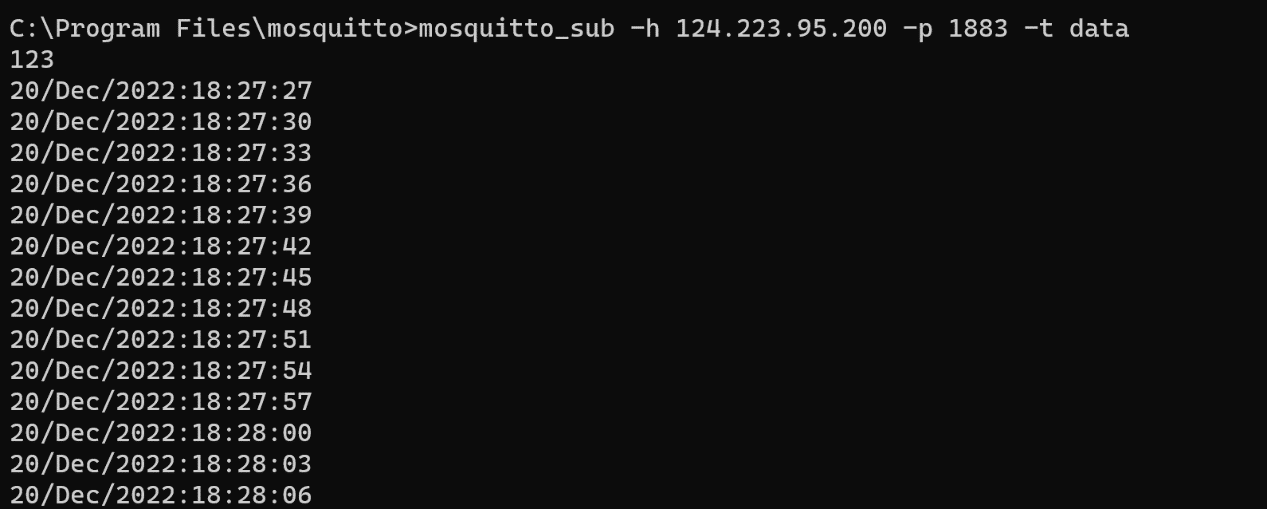
Disconnect，等待MQTT客户端完成所做的工作，并与服务器断开TCP/IP会话；

Subscribe，等待完成订阅；

UnSubscribe，等待服务器取消客户端的一个或多个topics订阅；

Publish，MQTT客户端发送消息请求，发送完成后返回应用程序线程。

* 1. **MQTT通讯协议（mosquitto）发布订阅**
     1. 在官网进行下载安装；

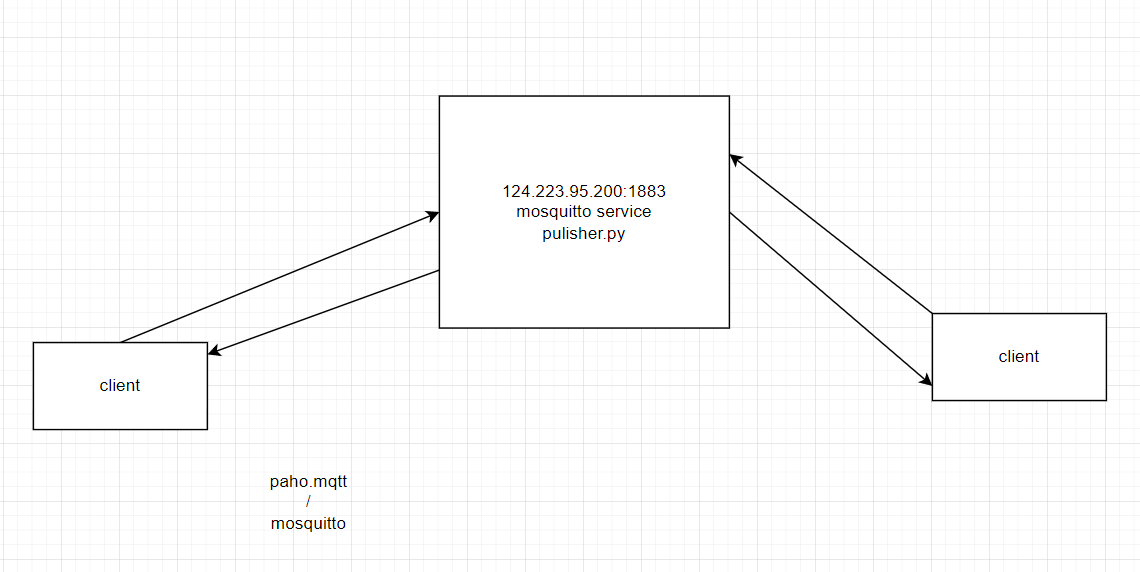
试运行如下：

* + 1. mosquito命令

|  |  |  |
| --- | --- | --- |
| **场景** | **命令** | **分析** |
| 启动代理服务 | mosquitto -v | 【-v】打印更多的调试信息 |
| 订阅主题 | mosquitto\_sub -v -t sensor | 【-t】指定主题，此处为sensor 【-v】打印更多的调试信息 |
| 发布内容 | mosquitto\_pub -t sensor -m 12 | 【-t】指定主题【-m】指定消息内容 |

1. **项目架构**

架构图：



以云服务器为中介，两台客户机通过树莓派进行数据的发布和订阅。客户机还可以通过web server与树莓派间接相连进行订阅和发布。

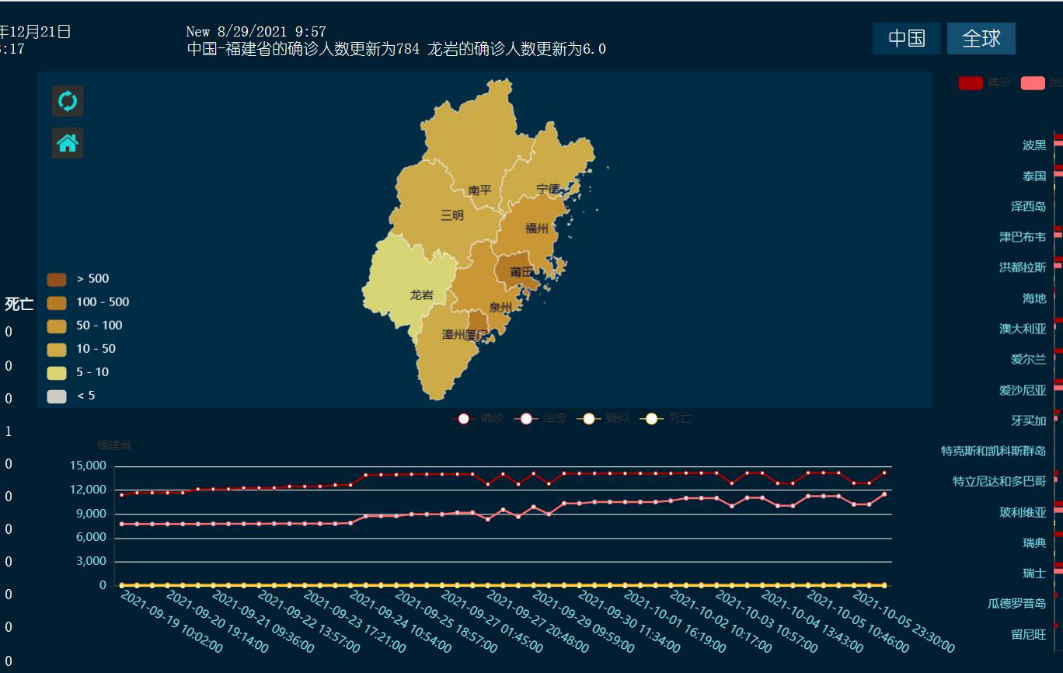
1. **项目实现**
2. **前端实现**

前端页面用html、css、js实现，接口选用ajax，搭建了一个响应式网页，其中页面布局分为五个模块，上端为页面基本信息和mqtt接受的实时消息展示，左侧为全国各省的累计确诊、疑似、治愈、死亡的数据，底部为历史数据、右侧为全球数据，中间为地图界面，悬浮鼠标可以看到信息详情，点击具体的省可进入小地图，也会更新左侧底部、右侧的数据。

大部分利用AJAX请求后端，将相应的数据用echarts进行图表化的展示。主要有:

* 1. 中国各省份的地图与最新新冠数据展示；
  2. 全国各省（某省各市）最新新冠数据表格；
  3. 某省按时间轴的数据的变化；
  4. 全世界的最新数据的柱状图展示。

展示如下：

地图

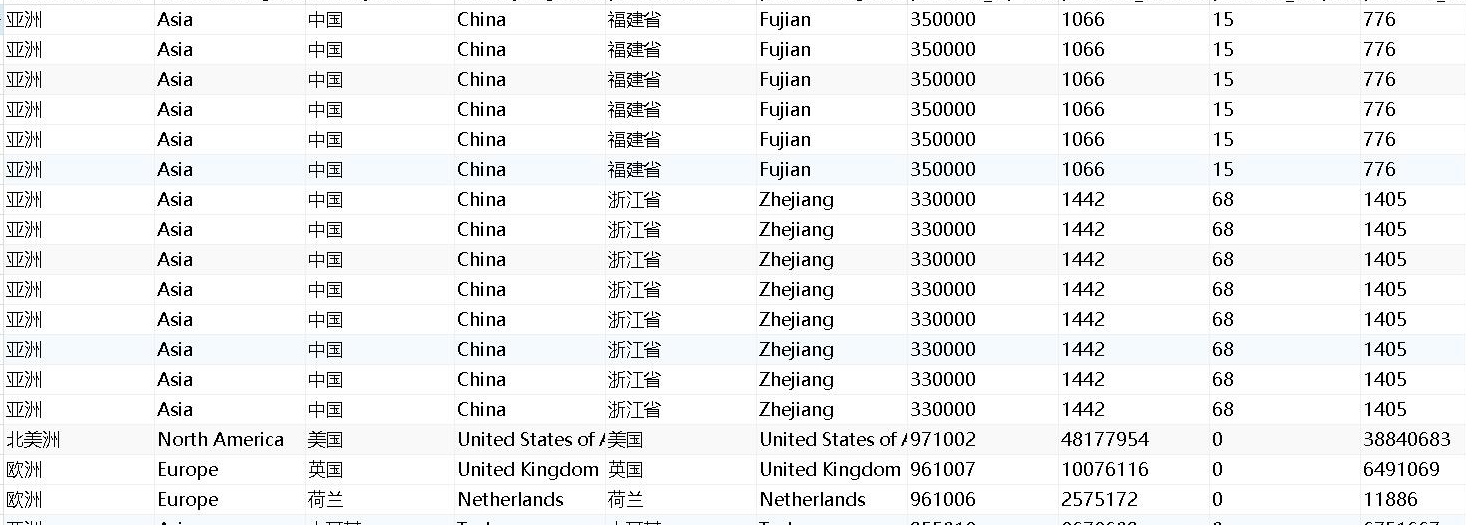
中度可信度描述已自动生成

1. **后端实现**

后端用python实现，选择了flask框架，前端展示的数据通过python从数据库中提取并在后端进行格式转换传输给前端。当订阅的mqtt主题发送了新消息时，将执行两步操作，首先将该信息进行数据处理，展示在前端的信息展示界面，其次将数据插入数据库，当前端点击刷新按钮时就可以看到新增信息。

使用python中的panda进行数据处理，由于提供的数据集数据较多，因此按照降低响应时间，减少无用数据的原则对数据进行清洗分类，将数据整理为世界疫情、国内疫情，并单独保存了最新一天的数据。以天为单位进行数据更新，因此通过MySQL进行了类型转换，将updateTime时间格式精确到天，清除重复数据，有效降低了数据量，减小交互的开销。

实现如下所示：



1. **MQTT Broker**

算法如下：

# encoding: utf-8  
   
import paho.mqtt.client as mqtt  
import time  
import datetime  
import json  
import pandas as pd  
  
HOST = "127.0.0.1"  
PORT = 1883  
  
def start():  
    client = mqtt.Client()  
    client.connect(HOST, PORT, 120)  
    df = pd.read\_csv(&apos;new.csv&apos;)      
    print(df.info())  
    df\_dict = dict()  
    client.loop\_start()  
    i=0  
    while True:  
        df\_dict[&apos;continentName&apos;]             =      df.loc[i][&apos;continentName&apos;]         
        df\_dict[&apos;continentEnglishName&apos;]      =      df.loc[i][&apos;continentEnglishName&apos;]          
        df\_dict[&apos;countryName&apos;]               =      df.loc[i][&apos;countryName&apos;]          
        df\_dict[&apos;countryEnglishName&apos;]        =      df.loc[i][&apos;countryEnglishName&apos;]           
        df\_dict[&apos;provinceName&apos;]              =      df.loc[i][&apos;provinceName&apos;]           
        df\_dict[&apos;provinceEnglishName&apos;]       =      df.loc[i][&apos;provinceEnglishName&apos;]           
        df\_dict[&apos;province\_zipCode&apos;]          =      str(df.loc[i][&apos;province\_zipCode&apos;])           
        df\_dict[&apos;province\_confirmedCount&apos;]   =      str(df.loc[i][&apos;province\_confirmedCount&apos;])           
        df\_dict[&apos;province\_suspectedCount&apos;]   =      str(df.loc[i][&apos;province\_suspectedCount&apos;])           
        df\_dict[&apos;province\_curedCount&apos;]       =      str(df.loc[i][&apos;province\_curedCount&apos;])           
        df\_dict[&apos;province\_deadCount&apos;]        =      str(df.loc[i][&apos;province\_deadCount&apos;])           
        # df\_dict[&apos;updateTime&apos;]                =      str(df.loc[i][&apos;updateTime&apos;])           
        df\_dict[&apos;updateTime&apos;]                =      str(datetime.datetime.now().strftime("%Y/%m/%d %H:%M"))           
        df\_dict[&apos;cityName&apos;]                  =      str(df.loc[i][&apos;cityName&apos;])           
        df\_dict[&apos;cityEnglishName&apos;]           =      str(df.loc[i][&apos;cityEnglishName&apos;])           
        df\_dict[&apos;city\_zipCode&apos;]              =      str(df.loc[i][&apos;city\_zipCode&apos;])       
        df\_dict[&apos;city\_confirmedCount&apos;]       =      str(df.loc[i][&apos;city\_confirmedCount&apos;])          
        df\_dict[&apos;city\_suspectedCount&apos;]       =      str(df.loc[i][&apos;city\_suspectedCount&apos;])          
        df\_dict[&apos;city\_curedCount&apos;]           =      str(df.loc[i][&apos;city\_curedCount&apos;])          
        df\_dict[&apos;city\_deadCount&apos;]            =      str(df.loc[i][&apos;city\_deadCount&apos;])               
                           
        # print(df\_dict)  
        print(json.dumps(df\_dict).replace(&apos;"nan"&apos;,&apos;None&apos;))  
        client.publish("data",json.dumps(df\_dict),qos=1)   
        time.sleep(10)  
        i+=1  
       
if \_\_name\_\_ == &apos;\_\_main\_\_&apos;:  
    start()

【Broker如下】

