

COVID-19 数据发布、订阅、分析预测系统

小组成员：1854025 杨晶 1950081 田同轩 1951725 邓若涛 1952730 金湘雯

目录

物联网—COVID-19 数据发布、订阅、分析预测系统.....	1
1.项目简介.....	2
1.1 项目背景.....	2
1.2 项目目标.....	2
2.功能介绍.....	2
2.1 主页面相关功能.....	2
2.1.1 展示当前时间.....	2
2.1.2 展示联系方式与友情链接.....	3
2.2 世界疫情相关功能.....	4
2.2.1 展示世界疫情概况.....	4
2.2.2 3D 热力图展示全球国家累计确诊	5
2.2.3 展示选中国家概况数据.....	6
2.2.4 展示 TOP10 国家详细数据	7
2.2.5 展示 TOP10 国家随时间变化数据	8
2.2.6 TOP10 国家疫情预测	9
2.2.7 发布某个国家新数据.....	10
2.3 国内疫情展示功能.....	11
2.3.1 展示国内疫情数据情况.....	12
2.3.2 中国热力图展示现有/累计确诊	13
2.3.3 国内疫情实时新闻播报.....	14
2.3.4 国内疫情随时间变化展示.....	15
2.3.5 国内疫情趋势预测.....	16
2.3.6 各省份疫情详细数据展示.....	17
2.3.7 展示所选省份各市累计/现有确诊	18
2.3.8 MQTT 协议实现邮箱订阅	20
3. 数据库设计.....	20
3.1 数据库环境.....	20
3.2 数据库设计思想.....	21
3.3 数据库框架.....	21
3.4 数据库详细信息.....	21
4. 框架设计.....	23
4.1 全局框架.....	23
4.2 前端框架	23
4.3 后端框架.....	24
5. 项目亮点.....	25
5.1 核心算法.....	25
5.2 MQTT.....	26
5.2.1 MQTT 原理.....	26

5.2.2 MQTT 实现.....	26
5.3 Redis.....	27
5.3.1 Redis 原理.....	27
5.3.2 Redis 实现.....	27

1.项目简介

1.1 项目背景

本项目背景基于 2020 年 1 月爆发的新冠疫情，截至 2021 年年末，全球疫情的增速和确诊人数仍在上升，疫情本身仍未完全得到控制。除去医疗卫生等直接防疫手段，疫情防控同样需要基于疫情的实时数据不断升级改善防疫政策。当前，各国疾控中心均需要基于大数据的疫情数据统计系统，用于数据统计，数据分析以及数据预测等需求。

1.2 项目目标

本项目旨在为医务工作者和普通民众构建出一个疫情大数据可视化系统，帮助疫情防控机构更直观的展示全球疫情分布情况和我国各省份的详细疫情数据，同时利用现有算法和技术对未来疫情趋势进行预测以帮助人们更好的制定疫情防控政策，提前预知可能风险。

本项目以物联网为出发点，同样希望通过疫情数据的发布，代理和处理来模拟一个物联网的场景，并尝试将 MQTT 通信协议应用到项目当中。MQTT 本身的低带宽，即时性同样和本项目的基本需求相适配，利用现有的 MQTT 平台将此协议应用到本项目中能够帮助我们更好的去理解物联网本身的真正魅力。

2.功能介绍

2.1 主页面相关功能

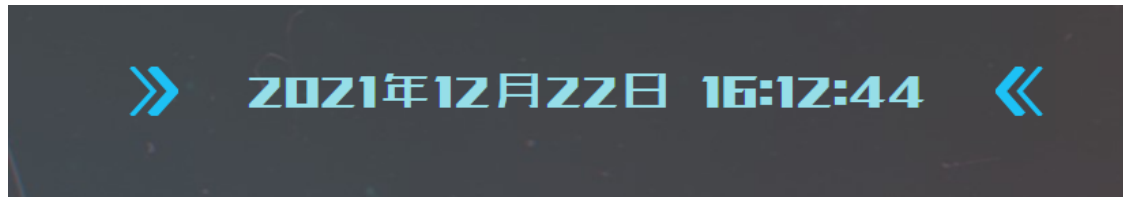
2.1.1 展示当前时间

1) 用例描述

用例名称	展示当前时间
标识符	UC101
说明	页面右上角可以展示当前时间

用例名称	展示当前时间
前置条件	用户已成功登录进入疫情可视化系统
后置条件	无
基本操作流程	1.系统请求当前时间 2.系统将时间渲染到右上角
可选操作流程	无

2) 功能截图



2.1.2 展示联系方式与友情链接

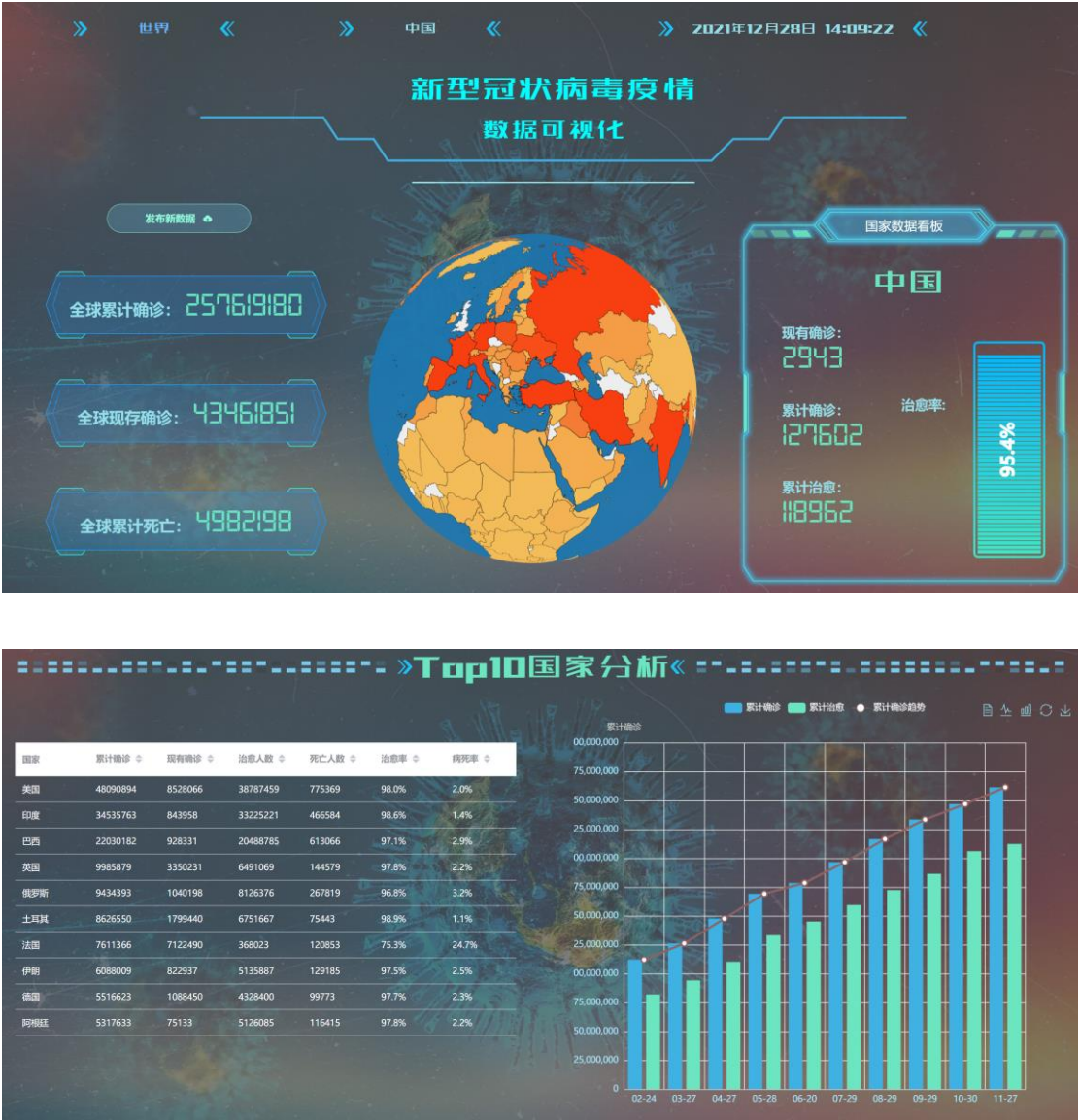
1) 用例描述

用例名称	展示联系方式、友情链接
标识符	UC202
说明	页面展示同济大学软件学院联系方式、友情链接
前置条件	用户已成功登录进入疫情可视化系统面
后置条件	无
基本操作流程	1.系统将对对应组件渲染到页脚
可选操作流程	无

2) 功能截图



2.2 世界疫情相关功能



2.2.1 展示世界疫情概况

1) 用例描述

用例名称	展示世界疫情概况
标识符	UC201
说明	页面展示世界疫情累计确诊、现存确诊、累计死亡
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情页面
后置条件	无
基本操作流程	1.系统像后端请求最新确诊数据 2.系统将数据渲染到世界疫情页面上端三个展示框上

用例名称	展示世界疫情概况
可选操作流程	无

2) 功能截图



2.2.2 3D 热力图展示全球国家累计确诊

1) 用例描述

用例名称	展示全球各个国家累计确诊
标识符	UC202
说明	页面通过自动旋转的 3D 地球渲染热力图展示全球累计确诊情况
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情界面
后置条件	无

用例名称	展示全球各个国家累计确诊
基本操作流程	1.系统向后端请求各个国家累计确诊数据 2.系统将数据渲染到 3D 地球热力图上
可选操作流程	3.用户可以对地球进行放大拖拽

2) 功能截图



2.2.3 展示选中国家概况数据

1) 用例描述

用例名称	展示选中国家概况数据
标识符	UC203
说明	页面展示鼠标悬浮的国家的现有确诊、累计确诊、累计治愈和治愈率
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情界面
后置条件	无

用例名称	展示选中国家概况数据
基本操作流程	1.用户将鼠标悬浮在某个国家上 2.系统在已请求到的数据中查询该国家 3.系统将数据渲染世界疫情页面右端框架上
可选操作流程	无

2) 功能截图



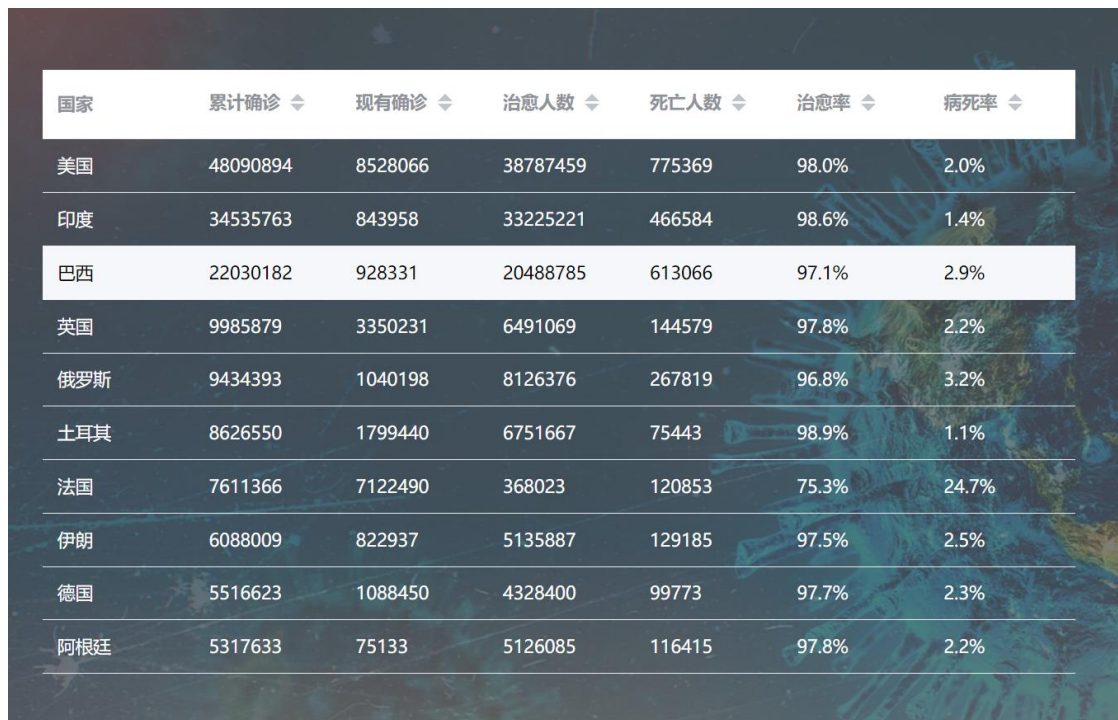
2.2.4 展示 TOP10 国家详细数据

1) 用例描述

用例名称	展示 TOP10 国家详情数据
标识符	UC204
说明	页面通过表格展示 TOP10 累计确诊的国家累计确诊、现有确诊、治愈人数、死亡人数、治愈率和病死率
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情界面
后置条件	无

用例名称	展示 TOP10 国家详情数据
基本操作流程	1.系统向后端请求 TOP10 国家数据 2.系统将数据渲染世界疫情页面下方表格中
可选操作流程	3.用户通过表格的排序功能对比查看 TOP10 国家数据

2) 功能截图



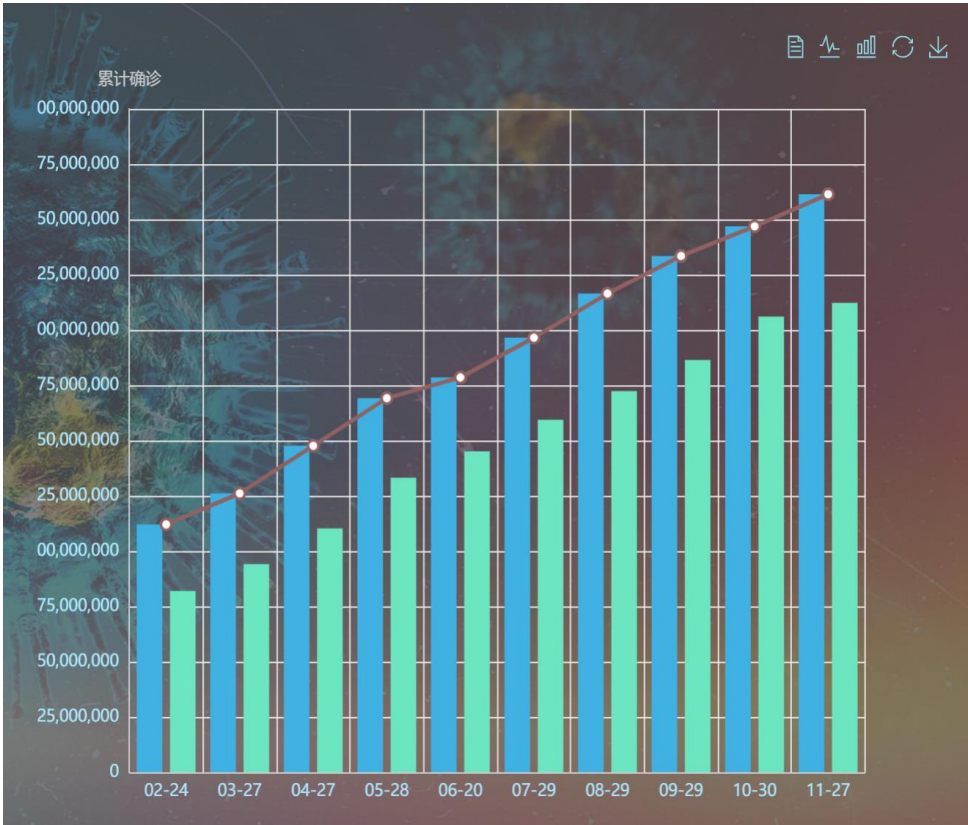
国家	累计确诊	现有确诊	治愈人数	死亡人数	治愈率	病死率
美国	48090894	8528066	38787459	775369	98.0%	2.0%
印度	34535763	843958	33225221	466584	98.6%	1.4%
巴西	22030182	928331	20488785	613066	97.1%	2.9%
英国	9985879	3350231	6491069	144579	97.8%	2.2%
俄罗斯	9434393	1040198	8126376	267819	96.8%	3.2%
土耳其	8626550	1799440	6751667	75443	98.9%	1.1%
法国	7611366	7122490	368023	120853	75.3%	24.7%
伊朗	6088009	822937	5135887	129185	97.5%	2.5%
德国	5516623	1088450	4328400	99773	97.7%	2.3%
阿根廷	5317633	75133	5126085	116415	97.8%	2.2%

2.2.5 展示 TOP10 国家随时间变化数据

1) 用例描述

用例名称	展示 TOP10 国家随时间变化数据
标识符	UC205
说明	页面通过柱状图展示所选国家累计确诊和累计治愈人数
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情界面
后置条件	无
基本操作流程	1.用户在表格中选中某个国家 2.系统将该国家的数据渲染到表格右端的柱状图中
可选操作流程	3.用户可以选择右上角工具箱进行相关操作

2) 功能截图

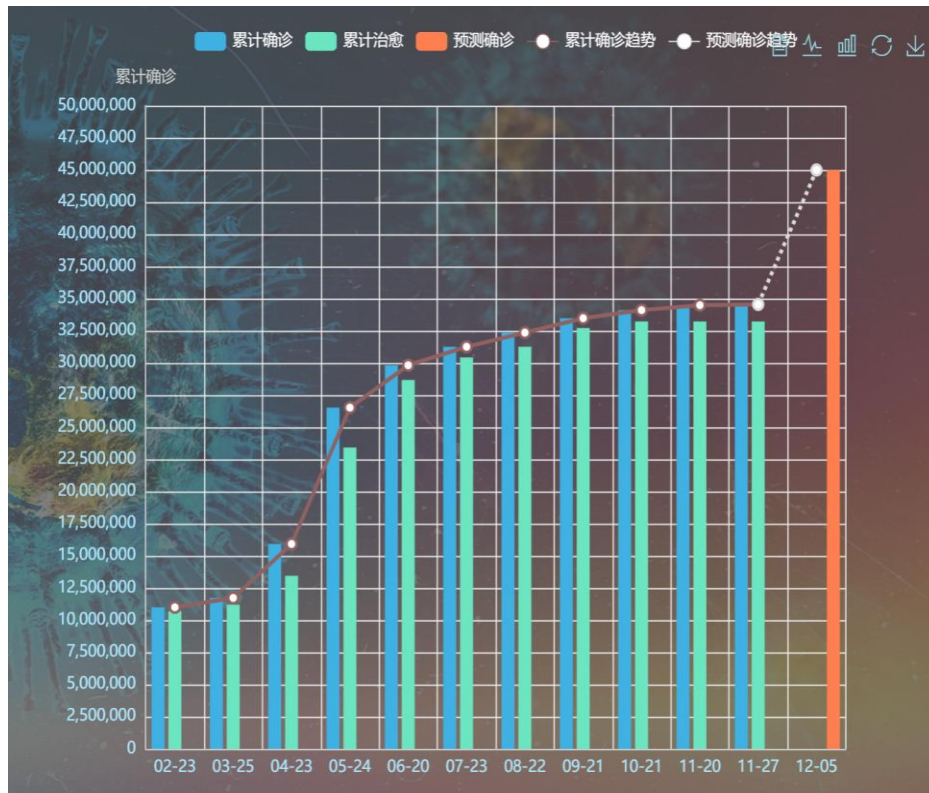


2.2.6 TOP10 国家疫情预测

1) 用例描述

用例名称	预测 TOP10 累计确诊未来趋势
标识符	UC206
说明	页面通过折线图展示所选国家累计确诊的未来变化预测趋势
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情界面
后置条件	无
基本操作流程	1.用户在表格中选中某个国家 2.系统将该国家的预测数据用折线的形式渲染到表格右端的柱状图中
可选操作流程	无

2) 功能截图



2.2.7 发布某个国家新数据

1) 用例描述

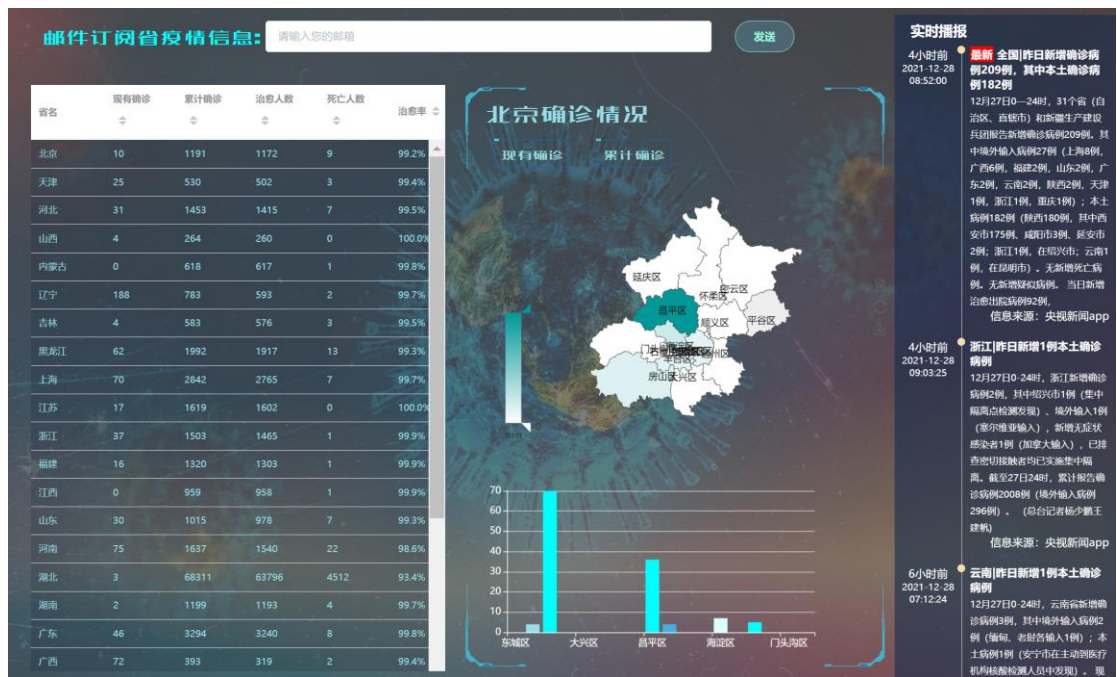
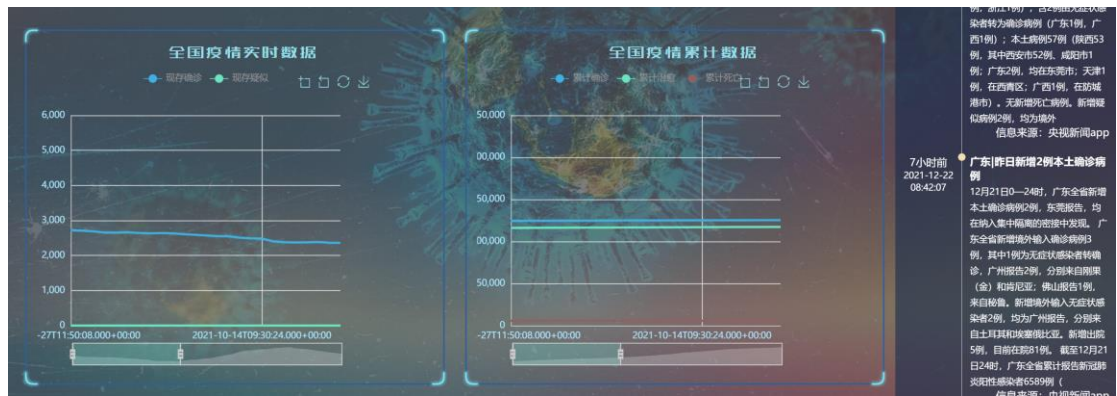
用例名称	预测 TOP10 累计确诊未来趋势
标识符	UC207
说明	疫情数据发布专员发布某个国家新确诊数据
前置条件	用户已成功登录进入疫情可视化系统并进入世界疫情界面
后置条件	无
基本操作流程	1.用户点击“发布数据”功能 2.用户下拉选择国家，输入新增加的数据 3.系统将该国家的新数据渲染到页面上
可选操作流程	无

2) 功能截图



2.3 国内疫情展示功能





2.3.1 展示国内疫情数据情况

1) 用例描述

用例名称	展示中国疫情概况
标识符	UC301
说明	页面展示中国疫情累计确诊、现存确诊、治愈率和死亡率
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情页面
后置条件	无
基本操作流程	<ol style="list-style-type: none"> 1.系统像后端请求最新确诊数据 2.系统将数据渲染到世界疫情页面左上端展示框上
可选操作流程	无

2) 功能截图

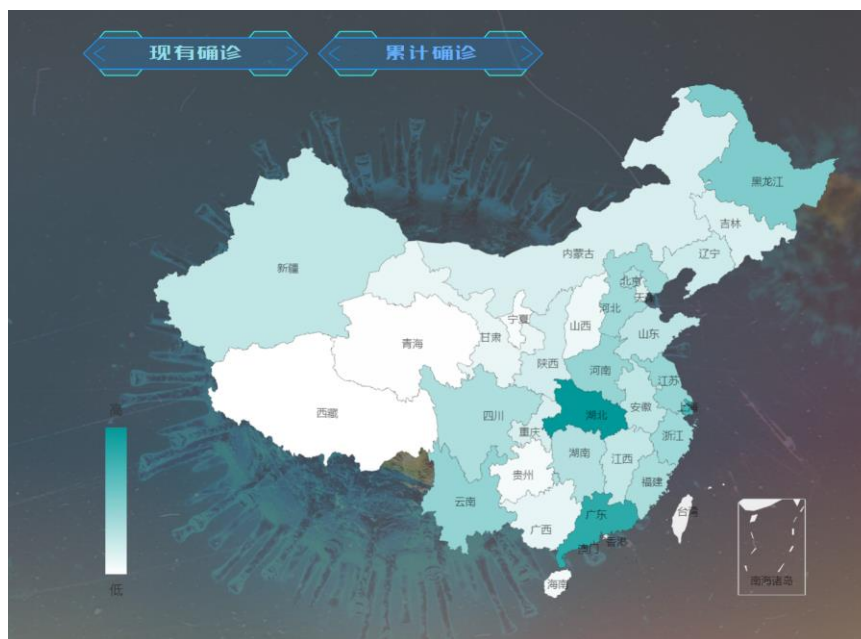


2.3.2 中国热力图展示现有/累计确诊

1) 用例描述

用例名称	通过中国地图热力图展示现有/累计确诊
标识符	UC302
说明	页面通过中国地图渲染热力图展示全球各个省份现有/累计确诊情况
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面
后置条件	无
基本操作流程	1.用户通过按钮选择现有确诊/累计确诊 2.系统向后端请求各个省份对应确诊数据 2.系统将数据渲染到中国地图热力图上
可选操作流程	无

2) 功能截图



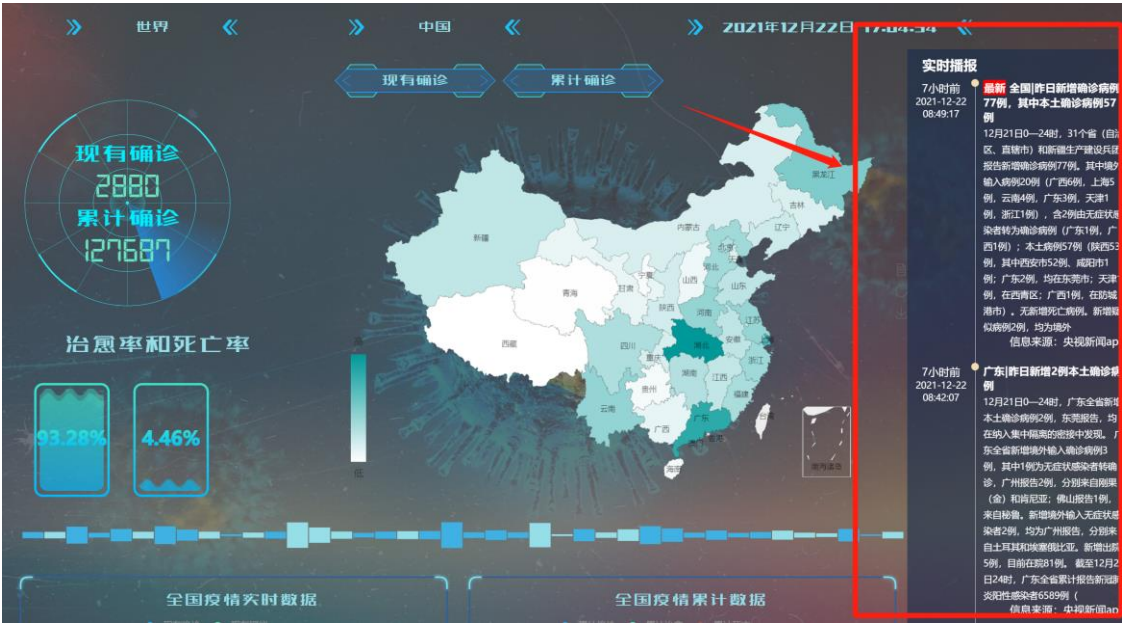
2.3.3 国内疫情实时新闻播报

1) 用例描述

用例名称	国内疫情实时新闻播报
标识符	UC303
说明	页面通过在右端悬浮展示最新的疫情新闻播报
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面

用例名称	国内疫情实时新闻播报
后置条件	无
基本操作流程	1.系统向接口请求最新相关新闻和对应时间 2.系统将新闻渲染到右端悬浮框上
可选操作流程	3.用户点击新闻跳转到新闻网站

2) 功能截图

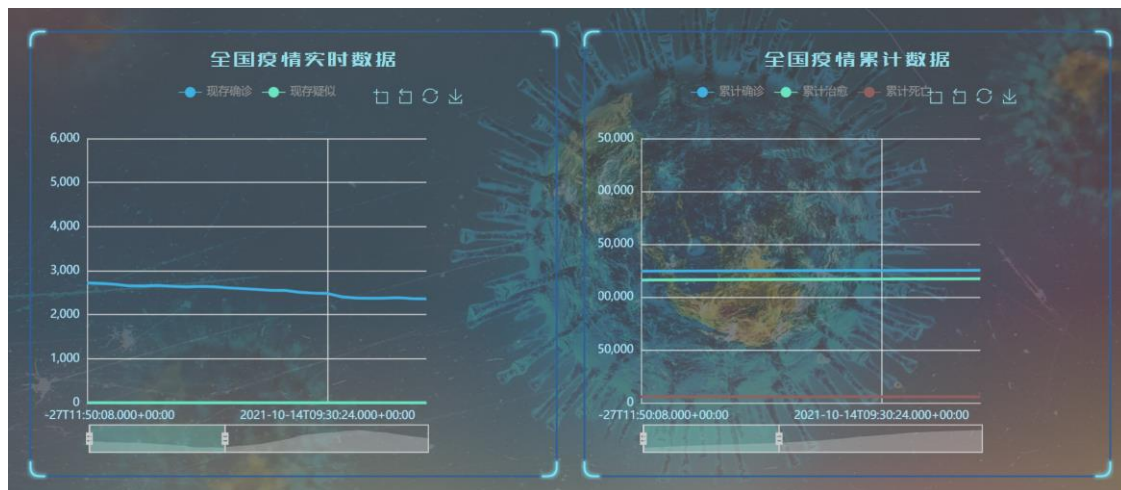


2.3.4 国内疫情随时间变化展示

1) 用例描述

用例名称	国内疫情随时间变化展示
标识符	UC304
说明	页面通过中国地图渲染热力图展示全球各个省份现有/累计确诊情况
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面
后置条件	无
基本操作流程	1.系统向后端请求国内疫情随时间确诊数据 2.系统将数据渲染到两张折线图上
可选操作流程	3.用户可通过折线图上方选择框选择要渲染的折线图内容

2) 功能截图



2.3.5 国内疫情趋势预测

1) 用例描述

用例名称	国内疫情趋势预测
标识符	UC305
说明	页面将对于国内疫情累计确诊的预测数据渲染到折线图中
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面
后置条件	无
基本操作流程	<ol style="list-style-type: none"> 1.系统调用中国疫情预测数据文件 2.系统将数据渲染到折线图上
可选操作流程	无

2) 功能截图



2.3.6 各省份疫情详细数据展示

1) 用例描述

用例名称	各省份疫情详细数据展示
标识符	UC306
说明	页面通过中国地图渲染热力图展示全球各个省份现有/累计确诊情况
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面
后置条件	无
基本操作流程	<ol style="list-style-type: none"> 1.系统向后端请求国内疫情随时间确诊数据 2.系统将数据渲染到两张折线图上
可选操作流程	<ol style="list-style-type: none"> 3.用户可通过折线图上方选择框选择要渲染的折线图内容

2) 功能截图

省名	现有确诊	累计确诊	治愈人数	死亡人数	治愈率
北京	10	1191	1172	9	99.2%
天津	25	530	502	3	99.4%
河北	31	1453	1415	7	99.5%
山西	4	264	260	0	100.0%
内蒙古	0	618	617	1	99.8%
辽宁	188	783	593	2	99.7%
吉林	4	583	576	3	99.5%
黑龙江	62	1992	1917	13	99.3%
上海	70	2842	2765	7	99.7%
江苏	17	1619	1602	0	100.0%
浙江	37	1503	1465	1	99.9%
福建	16	1320	1303	1	99.9%
江西	0	959	958	1	99.9%
山东	30	1015	978	7	99.3%
河南	75	1637	1540	22	98.6%
湖北	3	68311	63796	4512	93.4%
湖南	2	1199	1193	4	99.7%
广东	46	3294	3240	8	99.8%
广西	72	393	319	2	99.4%
重庆	5	610	500	6	99.9%

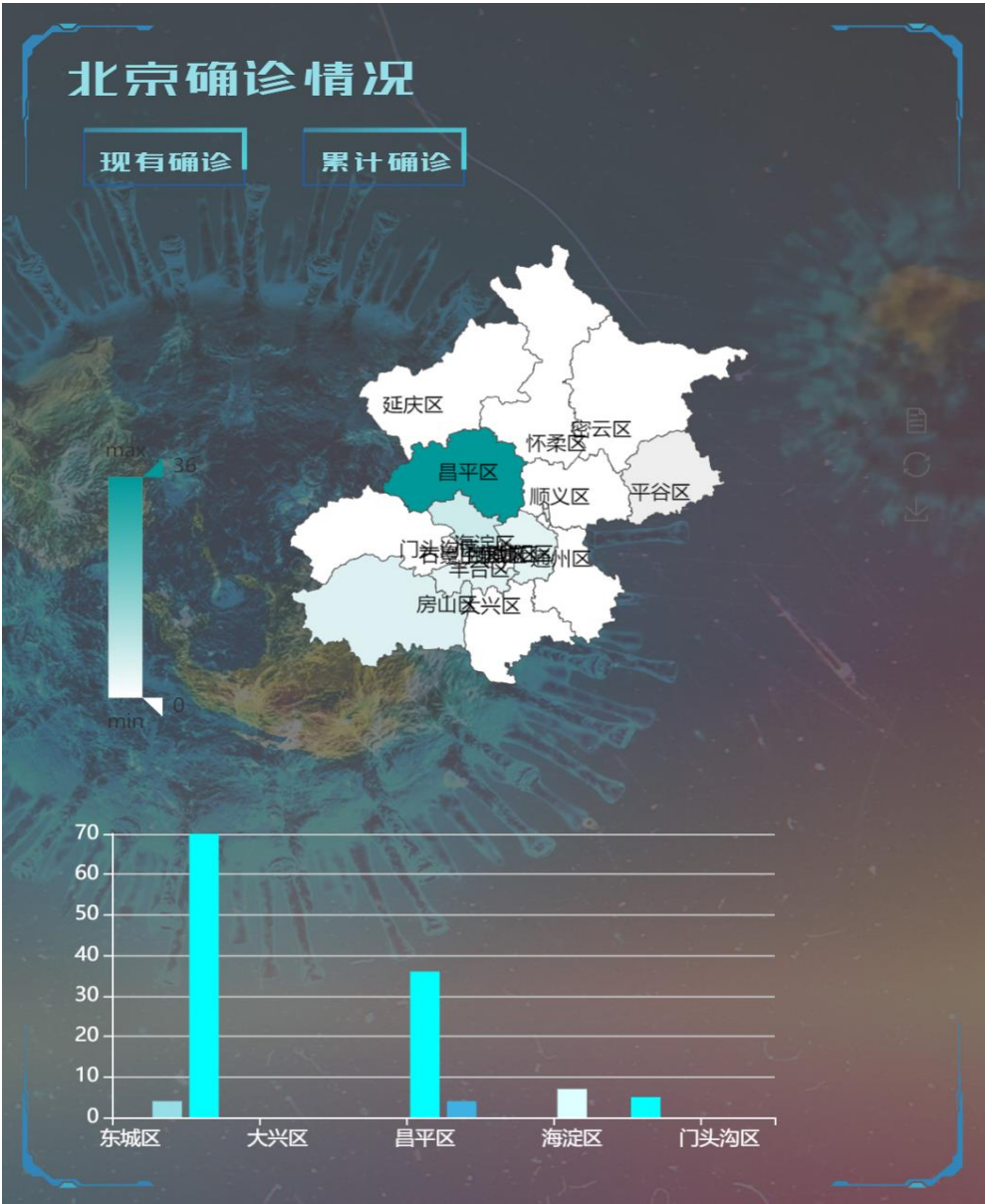
2.3.7 展示所选省份各市累计/现有确诊

1) 用例描述

用例名称	展示所选省份各市累计/现有确诊
标识符	UC307
说明	页面通过省地图和柱状图渲染展示各个市区现有/累计确诊情况
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面
后置条件	无

用例名称	展示所选省份各市累计/现有确诊
基本操作流程	1.用户选择省份 2.系统向后端请求所选省份各市区数据 3.系统将数据渲染到地图热力图和柱状图上
可选操作流程	无

2) 功能截图

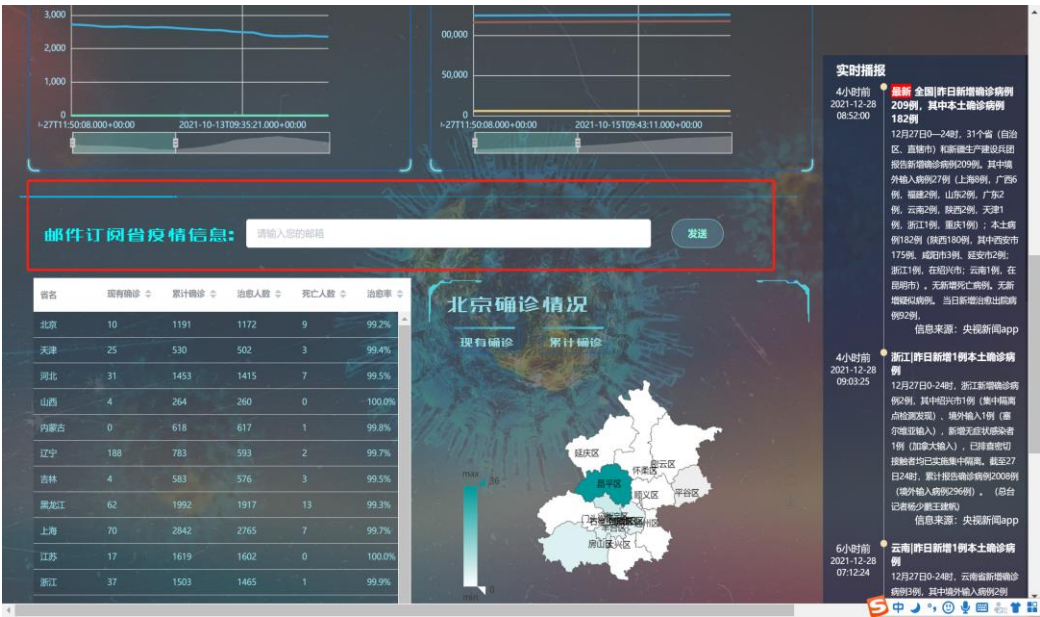


2.3.8 MQTT 协议实现邮箱订阅

1) 用例描述

用例名称	预测 TOP10 累计确诊未来趋势
标识符	UC308
说明	用户通过邮箱订阅全国疫情数据
前置条件	用户已成功登录进入疫情可视化系统并进入中国疫情界面
后置条件	无
基本操作流程	1.用户在邮箱输入框里输入邮箱 2.用户点击发送 3.系统向后端订阅 4.后端服务器向用户邮箱发布数据
可选操作流程	无

2) 功能截图



3.数据库设计

3.1 数据库环境

本次项目我们根据我们的需求，选择了较为轻量级的 `Mysql` 数据库。为了更好的调用与修

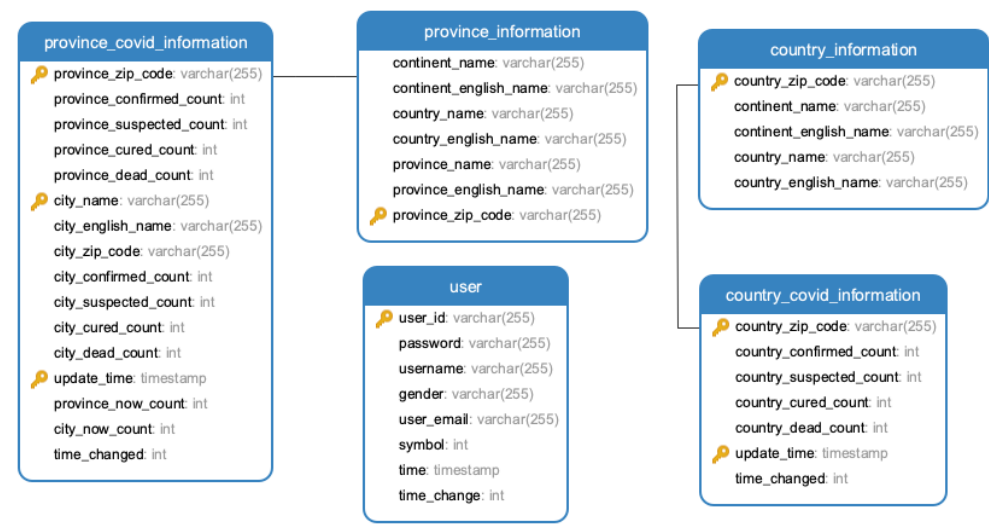
改，我们将数据库放在阿里云的 Linux 云服务器上：URL: 101.132.138.14, port: 3306, named: demo 。

3.2 数据库设计思想

本系统的数据库按照面向对象的思想设计，相应的实体类，从实体类中生成相应的数据库表，数据表中的关系，反映出对象之间的关系。

3.3 数据库框架

编号	表名	表的用途
1	Country_information	存放国家的地理信息
2	Country_covid_information	存放国家的疫情信息
3	Province_information	存放中国省份的地理信息
4	Province_covid_information	存放中国省份的疫情信息
5	User	管理员的账号



3.4 数据库详细信息

Country_information

列名	名称	数据类型	字段类型	长度
country_zip_code	国家编号	varchar(255)	varchar	255
continent_name	大陆中文名	varchar(255)	varchar	255

continent_english_name	大陆英文名	varchar(255)	varchar	255
country_name	国家中文名	varchar(255)	varchar	255
country_english_name	国家英文名	varchar(255)	varchar	255

Country_covid_information

列名	名称	数据类型	字段类型	长度
country_zip_code	国家编号	varchar(255)	varchar	255
country_confirmed_count	国家累计确诊人数	int	int	255
country_suspected_count	国家现有疑似病例	int	int	255
country_cured_count	国家累计痊愈人数	int	int	255
country_dead_count	国家累计死亡人数	int	int	255
update_time	更新时间	timestamp	timestamp	0

Province_information

continent_name	大陆名	varchar(255)	varchar	255
continent_english_name	大陆英文名	varchar(255)	varchar	255
country_name	国家名	varchar(255)	varchar	255
country_english_name	国家英文名	varchar(255)	varchar	255
province_name	省份名	varchar(255)	varchar	255
province_english_name	省份英文名	varchar(255)	varchar	255
province_zip_code	省份编号	varchar(255)	varchar	255

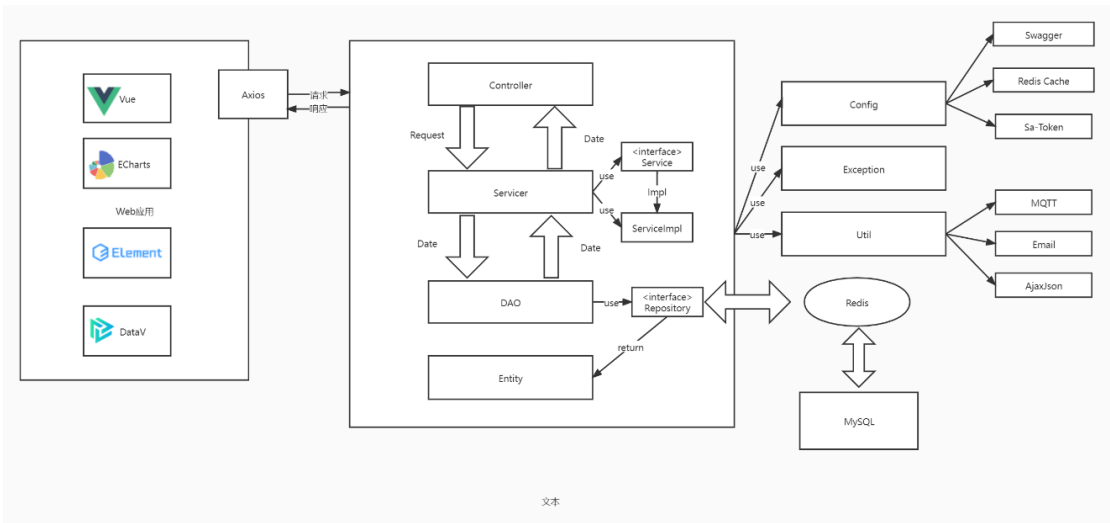
Province_covid_information

列名	名称	数据类型	字段类型	长度
province_zip_code	省份编号	varchar(255)	varchar	255
province_confirmed_count	省份累计人数	int	int	255
province_suspected_count	省份疑似人数	int	int	255
province_cured_count	省份累计痊愈	int	int	255
province_dead_count	省份累计死亡	int	int	255
city_name	城市名	varchar(255)	varchar	255
city_english_name	城市英文名	varchar(255)	varchar	255
city_zip_code	城市编号	varchar(255)	varchar	255
city_confirmed_count	城市累计确诊	int	int	255
city_suspected_count	城市现有疑似	int	int	255
city_cured_count	城市累计痊愈	int	int	255
city_dead_count	城市累计死亡	int	int	255

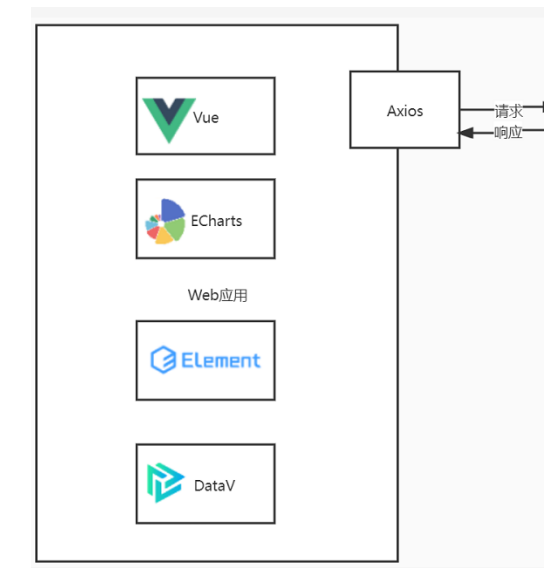
update_time	更新时间	timestamp	timestamp	0
province_now_count	省份现有人数	int	int	255
city_now_count	城市现有人数	int	int	255

4. 框架设计

4.1 全局框架



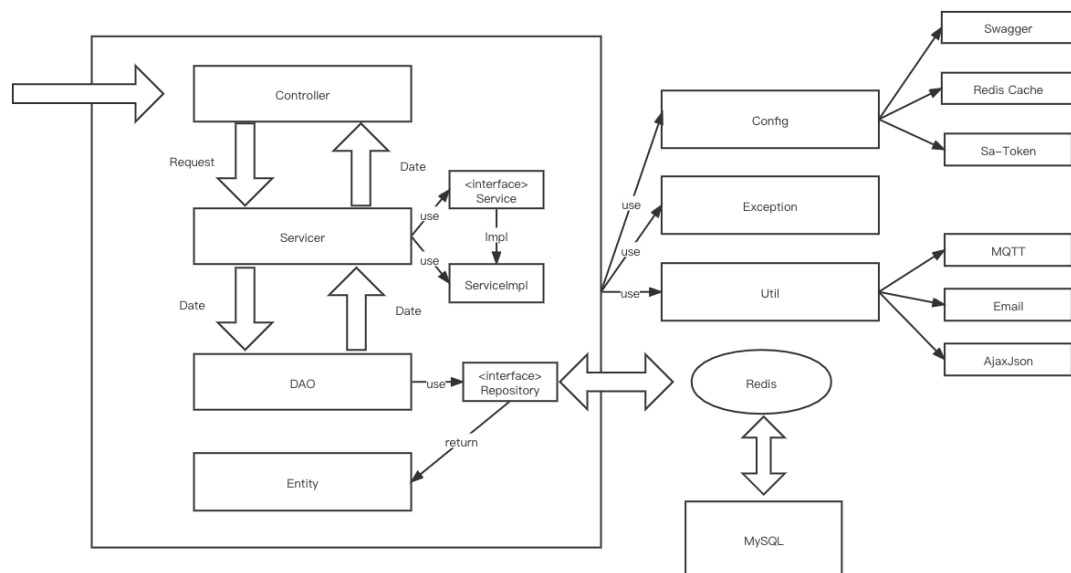
4.2 前端框架



前端基于 Web 应用，采用 Vue 框架，并通过 ElementUI，DataV，ECharts 等组件库快速动态的大屏数据展示页面，通过 axios 发送 http 请求。无论是发布端还是代理端，均通过运行

于现代浏览器的 Web 应用使用被系统，发送的信息应被同一转化为 JSON 格式的数据发送到服务层。

4.3 后端框架



后端主要采用了 **Springboot + jpa** 的框架搭建，它主要由以下的组件所构成：

- Controller**：与外部世界交互并直接调用 **Servicer** 来处理业务逻辑。
- Servicer**：处理业务逻辑。我们使用接口和 **impl** 来减少耦合。
- DAO**：我们主要利用 **jpa** 框架封装好的 **repository** 的接口类进行对数据库的查询。
- Entity**：根据业务逻辑和数据库进行设计，主要用于承载具体的数据

除了必要的控件外，我们还设计了一些重要的实体类。

—**Config**：一些全局的配置，包括了跨越配置，鉴权配置，缓存管理配置，文档接口配置等

—**Util**：一些重要且独立的工具类，包括了 **MQTT** 平台的封装与使用，邮件系统的封装与使用，标准 **JSON** 返回格式的规定等

—**Exception**：全局异常的封装与规定

在数据方面，我们除了基本的 **Mysql** 外还搭建了 **redis** 作为我们的缓存数据库，对于我们本次一共近百万的数据量来说，设置一定的缓存可以较大幅度提升我们的整体性能。

5.项目亮点

5.1 核心算法

主要使用到的函数是：

`numpy.polyfit(x,y,deg,rcond=None,full=False,w=None,cov=False)`

用到的原理是最小二乘多项式拟合。最小二乘法多项式曲线拟合，就是根据给定的 m 个点,并不要求这条曲线精确地经过这些点，而是曲线 $y=f(x)$ 的近似曲线 $y = \phi(x)$ 。

常见的曲线拟合方法有以下三种：

- 1.使偏差绝对值之和最小

$$\min_{\phi} \sum_{i=1}^m |\delta_i| = \sum_{i=1}^m |\phi(x_i) - y_i|$$

- 2.使偏差绝对值最大的最小

$$\min_{\phi} \max_i |\delta_i| = \max_i |\phi(x_i) - y_i|$$

- 3.使偏差平方和最小

$$\min_{\phi} \sum_{i=1}^m \delta_i^2 = \sum_{i=1}^m (\phi(x_i) - y_i)^2$$

其中按偏差平方和最小的原则选取拟合曲线，并且采取二项式方程为拟合曲线的方法,称为最小二乘法。

在此次项目中获得预测模型，就是利用了最小二乘多项式回归，通过获得以往的时间-确诊拟合曲线，来预测未来的确诊数据。具体步骤如下：

1. 对总数据进行处理，分割每个国家的数据
2. 对每个国家的数据进行细微的数据清洗，将一些错误数据筛选掉
3. 对每个国家的数据进行最小二乘多项式拟合，获取多项式系数
4. 通过多项式系数，预测未来几天的确诊人数

5.2 MQTT

5.2.1 MQTT 原理

MQTT 是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT 协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。在很多情况下，包括受限的环境中，如：机器与机器（M2M）通信和物联网（IoT）。其在，通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。

我们本次仿照物联网应用中对于 MQTT 的使用，实现了用户订阅并自动获取通知邮件的功能。

5.2.2 MQTT 实现

我们采用的是一个免费的 MQTT 平台——环信即时通讯云作为我们的 MQTT 服务器。

环信即时通讯云

应用名称: 1112211219116365#demo 试用 创建

服务概览 当前状态: 已开通 功能介绍

MQTT服务是基于MQTT协议的实时消息服务平台。使用MQTT服务可实现信令、应用内消息实时推送、实时数据流传输等场景。

服务套餐

当前套餐: 免费版 套餐介绍 联系商务

消息总量/月	消息流量/月	峰值会话数
300万条	5.00/GB	100个

服务配置

AppID: 7n36i0 复制

连接地址: 7n36i0.cn1.mqtt.chat 复制

连接端口: 1883(mqtt), 1884(mqtts), 80(ws), 443(wss)

REST API地址: https://api.cn1.mqtt.chat/app/7n36i0 复制

注: clientID组织形式为"deviceID@appID", 其中deviceID由用户自定义快速连接指南

今日统计数据

DAU 消息总数 峰值会话数 峰值订阅数

0 0 0 0

我们在后端的 Util 中对 MQTT 进行了封装与函数的定义

```
CountryInformationService 62
ProvinceCovidInformationService 63
UserService 64
Util 65
  AjaxJson 168
  ConnectUtil 176
  MQTTUtil 177
  SendEmailUtil 208
  TimeUtil 201
  VerifyEmailUtil 205
DemoApplication 206
  resources 215
    static 216
    templates 225
      emailTemplate.html 226
      application.yml 235
test 236
Persistence 240

public void getToken() { ... }
public MqttClient setMqttClient() throws MqttException { ... }
public AjaxJson MqttClientConnect() throws MqttException, InterruptedException { ... }
public AjaxJson MqttClientDisconnect() throws MqttException { ... }
public AjaxJson sendMessage(String information) throws MqttException { ... }
public AjaxJson sendMessage(String information, String topic) throws MqttException { ... }
public AjaxJson sendMessage(String information, String topic, int qosLevel) throws MqttException { ... }
public AjaxJson subscribe(String[] topFilters) throws MqttException { ... }
```

用户通过订阅特定的主题，管理员向该主题发送信息，通过我们后端的处理，将信息转化为邮件发送到用户的邮箱当中。

5.3 Redis

5.3.1 Redis 原理

Redis 是一个高性能的 key-value 数据库。redis 的出现，很大程度补偿了 memcached 这类 key/value 存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了 Java, C/C++, C#, PHP, JavaScript, Perl, Object-C, Python, Ruby, Erlang 等客户端，使用很方便。

Redis 支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得 Redis 可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。

我们将一些大量且常用的数据放到我们的 redis 中方便前端的二次查阅，成功做到了数据的高效处理。

5.3.2 Redis 实现

我们在放数据库的阿里云 Linux 服务器上又建立了 redis 服务器，并在后端的 config 中全局配置了 redis 服务器的使用，之后在需要缓存的数据的函数上使用@注释来进行缓存与其他操作。

```
29  */
30  @Override
31  @Bean
32  public KeyGenerator keyGenerator() {...}
33
34  @Bean
35  public RedisTemplate<Object, Object> redisTemplate() {...}
36
37  @Bean
38  @Override
39  public CacheResolver cacheResolver() { return new SimpleCacheResolver(Objects.requireNonNull(cacheManager())); }
40
41  @Bean
42  @Override
43  public CacheErrorHandler errorHandler() {...}
44
45  @Bean
46
47  @Override
48  public CacheManager cacheManager() {...}
49
```