|  |  |
| --- | --- |
| **民大校徽** | **中央民族大学课程设计报告** |
| **Course Lab Report in MUC** |

**新冠疫情分析系统**

**详细设计说明书**

|  |  |
| --- | --- |
| 小组成员： | 20011441 黄影 |
| 小组成员： | 20150019 李承骏 |
| 院 系 ： | 信息工程学院 |
| 专 业： | 软件工程 |
| 任课教师： | 孙娜 |
|  |  |

目录

[1、 引言 1](#_Toc11736)

[1.1编写目的 2](#_Toc19496)

[1.2大数据技术发展历程及研究现状 2](#_Toc15645)

[1.3参考文献 2](#_Toc25156)

[2、 设计概述 2](#_Toc14250)

[2.1 java爬虫 3](#_Toc8723)

[2.2JSoup 4](#_Toc14201)

[3.系统可行性分析 4](#_Toc22475)

[3.1技术可行性 4](#_Toc24639)

[4. 系统功能模块设计 5](#_Toc27534)

[5. 数据模型设计 7](#_Toc13649)

[5.1数据库表设计 7](#_Toc18407)

[6.应用原型设计 11](#_Toc26550)

[6.1 数据爬取功能模块 11](#_Toc19465)

[6.2数据功能模块 13](#_Toc13439)

[6.3用户功能模块 19](#_Toc22719)

[7.项目GitHub地址 23](#_Toc14660)

1. **引言**

**1.1编写目的**

本文档旨在介绍基于Spark的新冠疫情分析系统的设计与实现，利用新冠肺炎疫情监测平台，将全国疫情信息、基本物资、病例信息等信息进行即时采集，更能直观地掌握疫情形势,利用 Java进行数据收集、数据处理、数据分析、数据处理、数据分析等。

**1.2大数据技术发展历程及研究现状**

大数据，即需要快速获取、处理、分析以提取价值的大量交易数据、多样化的交易数据、交互数据和传感器数据，通常以pb (1024TB)的规模进行。不同的组织对大数据也有不同的定义。

麦肯锡对大数据的定义：一个足够大的数据集合是超越传统database-

softwaretools的可能性在访问、存储、管理和分析方面,有四个特点:一个巨大的dataschaal,快速的数据流,多种数据类型和低密度的值。

定义大数据通过移动信息技术研究中心:大数据是帮助企业利用庞大的数据量assasses,实时、准确地了解未知领域的动态逻辑,和快速开发新的数据管理技术为业务流程、组织和行业。

IDC认为大数据有四个特点:体积、异质性、高速和价值。

按照数据的处理速度，可以将其划分成批量的海量数据和流式的海量数据。大量的海量数据也被称作“历史海量”，而流大数据则被称作“即时大数据”。

大规模的大数据系统，在大量的预处理和高效的实时查询之后，需要对大量的数据进行大量的合并和装载。尽管此类软件可以有效地实现对海量海量信息的需求查询，但是无法及时获取当前的数据，并具有一定的延迟。

大型的流式数据处理技术是以流式方式对数据进行数据的处理，然后把这些数据逐级装入到一个高效的存储数据库中进行查询。该方法能够有效地对当前的实时数据进行预测和分析。

随着因特网和计算机产业的飞速发展，企业对数据的使用也日益关注，商业软件的开发也逐步从批量的数据平台过渡到了实时的数据流。Spark是卡夫卡大数据的典型，它的主要应用是数据流，最近几年已经占据了绝对的优势。

**1.3参考文献**

1. 李庆民。基于java的软件agent开发环境的分析[J].数字技术与应用，2017,01:189.
2. 籍慧文。Web应用开发中JAVA编程语言的应用探讨[J].科技创新与应用，2017,07:90.

[3] 王宏玉，徐步步。基于Java的BBS开发[J].电脑知识与技术，2016,28:81-82.

[4] 生力军。基于JavaServlet的微信公众平台开发实训环境搭建[J.电脑知识与技术，2017,01:79-81

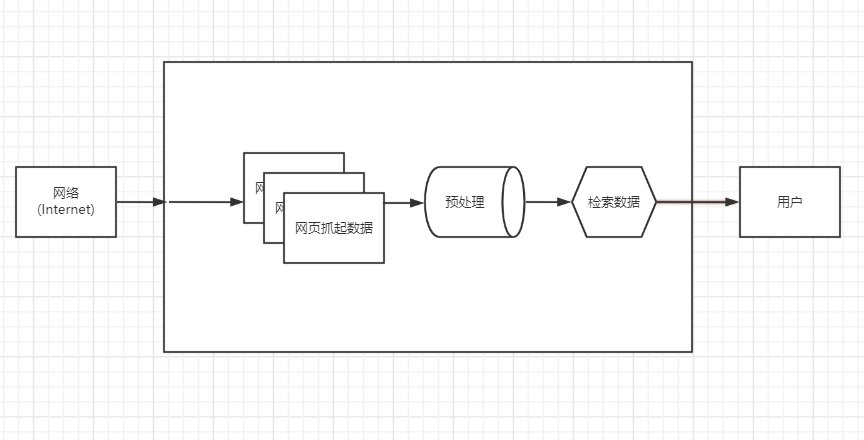
[5] 曹英.大数据环境下Hadoop 性能优化的研究[D].大连海事大学,2013.

[6] 童明.基于HDFS的分布式存储研究与应用[D].华中科技大学,2012.

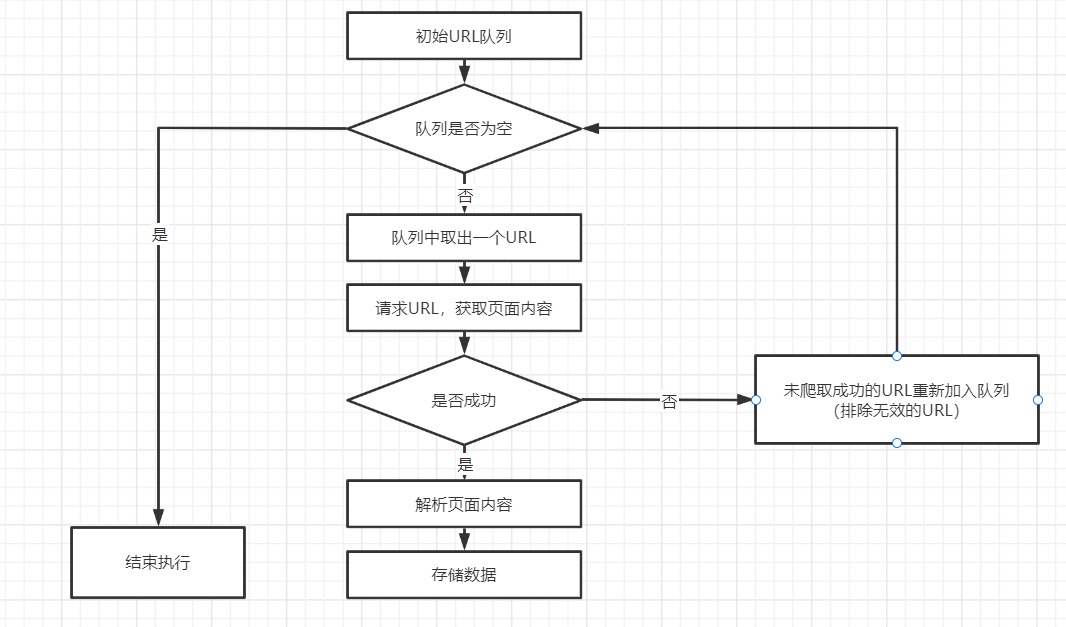
1. **设计概述**

**2.1 java爬虫**

访问拥有大量数据的资源的最有效方法是网络上的爬虫技术。爬虫技术和技术用于搜索引擎，如100度和100度搜索引擎、谷歌搜索引擎等。而爬虫科技能够满足在互联网上搜索所需的要求和网页包含捕获。



在本系统中运用java进行网络爬虫的流程如下：



**2.2JSoup**

Jsoup是一款Java的html解析工具，主要是对html和xml文件进行解析

在写爬虫的时候，当我们用HttpClient之类的框架，得到目标网页的源码后，需要从网页源码中取得我们想要的内容。就可以使用jsoup轻松获取想要的内容。

jsoup 是一款基于 Java 语言的 HTML 请求及解析器，可直接请求某个 URL 地址、解析 HTML 文本内容。它提供了一套非常省力的 API，可通过 DOM、CSS 以及类似于 jQuery 的操作方法来取出和操作数据。

JSoup的主要功能如下：

1.从一个URL，文件或字符串中解析HTML；

2.使用DOM或CSS选择器来查找、取出数据；

3.可操作HTML元素、属性、文本；

**3.系统可行性分析**

**3.1技术可行性**

本项目采用了几种方法：Java Respark、Java集成SPARK、数据分析代码实现、本地数据存储、通过Echarts实现接口等。该技术的实现是可行的，技术、体系结构、整个项目的开发思路在现有技术的支持下是完全可以实现的，所以技术可行性是完全可能的。

**3.2经济可行性**

对电脑等硬件设备的要求不高，对于数据源的选则，是从公开的免费的丁香园网站获取，服务器的选则也是是用自己的电脑。综上，本系统的实现满足经济可行性的原则。

**3.3操作可行性**

整体来说，项目启动后便可以实时获取数据，并进行数据的实时分析，操作起来还是很方便的。操作可行性很好。

**3.4法律可行性**

这个项目的分析仅供学习使用，且数据源公开透明，不涉及法律纠纷。最终的分析结果也仅用于学习，斌不会侵害他人或机构的利益。法律可行性很好

1. **系统功能模块设计**

### **4.1爬取疫情数据功能分析**

在项目中创建crawler模块，专门用于负责数据的爬取部分。

* 通过HttpUtils.getHtml()方法来爬取指定页面，方法中传入String类型的url参数，在此次项目中，这个url的值为丁香园官网的一期数据查询网站链接，此网站，通过专业权威的内容分享互动、丰富全面的医疗数据积累、高质量的数字医疗服务，连接医生、科研人士、患者、医院、生物医药企业和保险企业。此次的url为 ： https://ncov.dxy.cn/ncovh5/view/pneumonia。
* 解析页面中的指定内容 ： 以上爬取方法获取到的内容为丁香园疫情统包含HTML的第2页上的所有数据都被删除了，包括标签：文档中提到了当前版本的网页，HTML标签，没有直接出现在网页上的“头”标签，标签用来放置网站的原始数据，这里是Meta用来创建页面行集的地方，以避免混淆，标题中的内容将显示在页面标题中，搜索引擎根据标题中的内容对页面的主要内容进行评估，身体是HTML的一个子元素，代表页面的主要对象、所有可见页面和其他结构，其中包含定义BODI权重所需的标签。此时，必须使用JSoup解析器方法来分析所取的整个页面。包括JSoup选择：基本选择器、组合选择器、伪选择器。调用正确的数据选择方法获取响应。
* 使用正则表达式获取 json 格式的疫情数据

定义正则表达式String pattern = "\\[(.\*)\\]"，编译成正则对象：Pattern reg = Pattern.compile(pattern);调用 reg.matcher 来匹配 text 中符合正则表达式的内容：Matcher matcher = reg.matcher(text);如果 text 中的内容和正则规则匹配上就取出来，对 json 数据进行更进一步的解析。

* 将第一层json(省份数据)解析成为JavaBean。
* 将第二层json(城市数据)解析成为JavaBean，将 JavaBean 转为 Json字符串再发送给 kafka。
* 获取第一层json(省份数据)中的每一天的统计数据；获取 statisticsDataStr 中 data(每一天的统计数据) 字段对应的数据，将解析出来的每一天的数据设置回 pBean 中的 statisticsData 字段中，之前该字段是一个 url，将 JavaBean 转为 Json字符串再发送给 kafka。
* 在整个大方法前加注解@Scheduled(cron="0 0 8 \* \* ?")，表示当启动服务后，程序会在每天的8点定时执行爬取数据的工作。

**4.2物资信息生成分析**

通过JSON.toJSONString()方法可以将JSON格式的数据转换成字符串类型，方便我们将生成的疫情物资数据转换为jsonStr再发送到Kafka集群来进行数据分析。

**4.3数据处理分析**

* 创建StructuredStreaming执行环境：

StructuredStreaming支持使用SQL来处理实时流数据,数据抽象和SparkSQL一样,也是DataFrame和DataSet

所以这里创建StructuredStreamin执行环境就直接创建SparkSession即可

导入隐式转换方便后续使用

通过jsonStrDS对象的writeStream方法的format、outputMode、trigger、option等API，来实现信息的设置。

* 连接Kafka：

从kafka接收消息；

取出消息中的value；

输出目的地；

输出模式,默认就是append表示显示新增行；

触发间隔,0表示尽可能快的执行；

表示如果列名过长不进行截断；

**4.4数据存储分析**

该项目采用爬虫、数据库、大数据分析、爬虫理解、使用、数据库（MySQL）和大数据知识等方法进行。

在数据设计方面，首先应建立基本信息材料数据库，以存储有关材料数量及其变化的信息。具体的数据库字段设计如下：

数据存储器可以记录在获取新数据时发生的变化，并使我们能够对数据进行更多的分析，同时也可以更安全地长期存储这些变化，而这些变化并不那么容易丢失。二次甚至三次数据分析可能更有效。同时，我们将更方便地获得信息。

1. **数据模型设计**

**5.1数据库表设计**

表5.1-1 北京疫情数据表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否不为null | 是否是主键 | 注释 |
| id | int | 4 | 是 | 是 | 主键ID |
| confirmed\_count | int | 4 | 否 | 否 | 累计确诊 |
| confirmed\_incr | int | 4 | 否 | 否 | 当天确诊 |
| cured\_count | int | 4 | 否 | 否 | 累计治愈 |
| cured\_incr | int | 4 | 否 | 否 | 当天治愈 |
| dead\_count | int | 4 | 否 | 否 | 累计死亡 |
| dead\_incr | int | 4 | 否 | 否 | 当天死亡 |
| date\_id | varchar | 255 | 否 | 否 | 日期 |

图5.1-2北京疫情数据库设计图

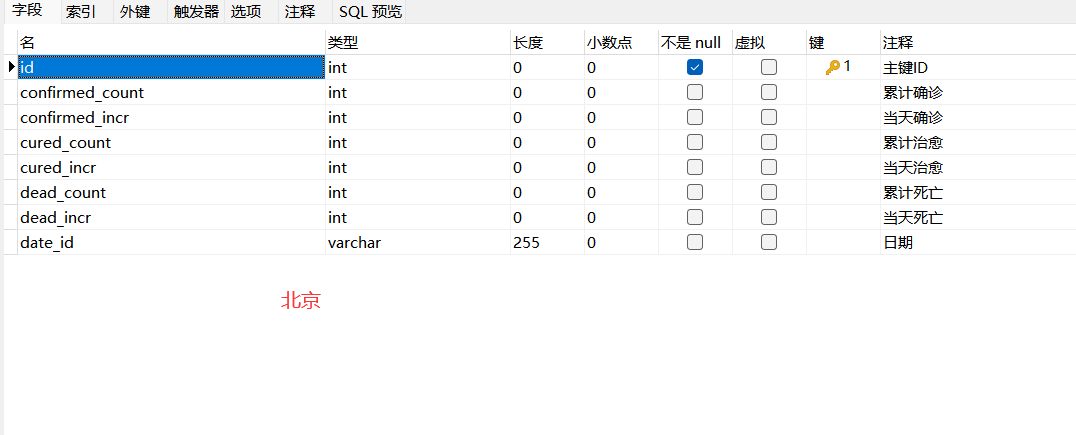


表5.1-3 上海疫情数据表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否不为null | 是否是主键 | 注释 |
| id | int | 4 | 是 | 是 | 主键ID |
| confirmed\_count | int | 4 | 否 | 否 | 累计确诊 |
| confirmed\_incr | int | 4 | 否 | 否 | 当天确诊 |
| cured\_count | int | 4 | 否 | 否 | 累计治愈 |
| cured\_incr | int | 4 | 否 | 否 | 当天治愈 |
| dead\_count | int | 4 | 否 | 否 | 累计死亡 |
| dead\_incr | int | 4 | 否 | 否 | 当天死亡 |
| date\_id | varchar | 255 | 否 | 否 | 日期 |

图5.1-4 北京疫情数据库设计图

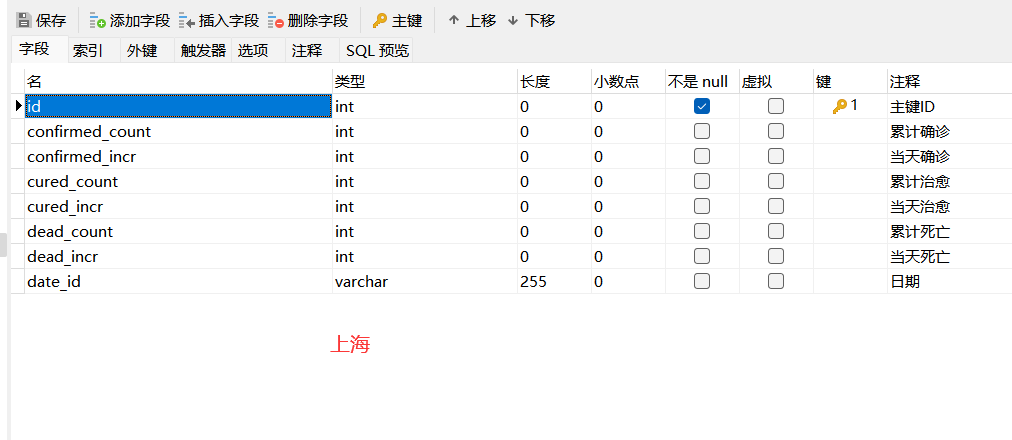


表5.1-5疫情数据表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否不为null | 是否是主键 | 注释 |
| id | int | 4 | 是 | 是 | 主键ID |
| province\_name | varchar | 255 | 否 | 否 | 省份名称 |
| province\_short\_name | varchar | 255 | 否 | 否 | 省份简称 |
| city\_name | varchar | 255 | 否 | 否 | 城市 |
| current\_confirmed\_count | int | 4 | 否 | 否 | 当天确诊 |
| confirmed\_count | int | 4 | 否 | 否 | 累计确诊 |
| suspected\_count | int | 4 | 否 | 否 | 疑似病例 |
| cured\_count | int | 4 | 否 | 否 | 累计治愈 |
| dead\_count | int | 4 | 否 | 否 | 累计死亡 |
| location\_id | int | 4 | 否 | 否 | 位置id |
| pid | int | 4 | 否 | 否 | 省份id |
| datetime | varchar | 255 | 否 | 否 | 日期 |

图5.l-6疫情数据库设计图

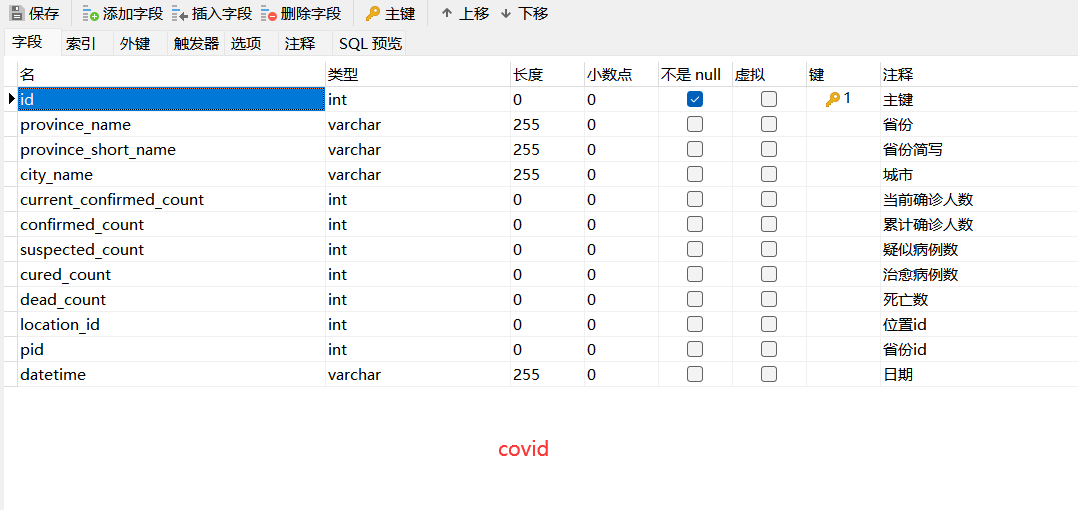


表5.1-7 湖北疫情数据表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否不为null | 是否是主键 | 注释 |
| id | int | 4 | 是 | 是 | 主键ID |
| confirmed\_count | int | 4 | 否 | 否 | 累计确诊 |
| confirmed\_incr | int | 4 | 否 | 否 | 当天确诊 |
| cured\_count | int | 4 | 否 | 否 | 累计治愈 |
| cured\_incr | int | 4 | 否 | 否 | 当天治愈 |
| dead\_count | int | 4 | 否 | 否 | 累计死亡 |
| dead\_incr | int | 4 | 否 | 否 | 当天死亡 |
| date\_id | varchar | 255 | 否 | 否 | 日期 |

图5.1-8湖北数据库设计图



表5.1-9 用户数据表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否不为null | 是否是主键 | 注释 |
| uid | int | 4 | 是 | 是 | 主键ID |
| username | varchar | 20 | 是 | 否 | 用户名 |
| password | char | 32 | 是 | 否 | 密码 |
| salt | char | 36 | 否 | 否 | 盐值 |
| gender | int | 0 | 否 | 否 | 性别 |
| phone | varcahr | 50 | 否 | 否 | 电话 |
| email | varchar | 50 | 否 | 否 | 邮箱 |
| avatar | varchar | 50 | 否 | 否 | 图片 |
| Created\_user | varchar | 20 | 否 | 否 | 创建人 |
| Created\_time | datetime | 0 | 否 | 否 | 创建时间 |
| Modified\_user | varchar | 20 | 否 | 否 | 修改人 |
| Modified\_time | datetime | 0 | 否 | 否 | 修改时间 |

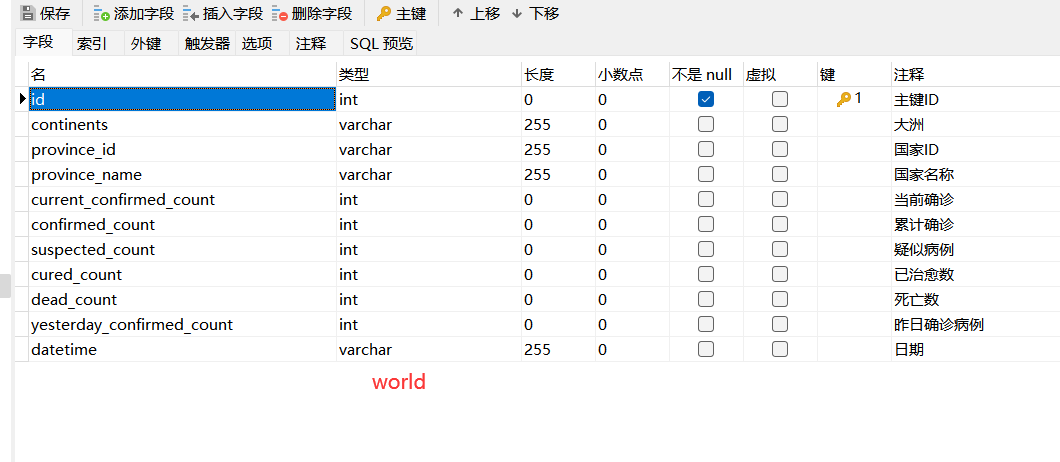
图5.1-10用户据库设计图



表5.1-11 世界疫情数据表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 长度 | 是否不为null | 是否是主键 | 注释 |
| id | int | 0 | 是 | 是 | 主键ID |
| continents | varchar | 255 | 否 | 否 | 大洲 |
| province\_id | varchar | 255 | 否 | 否 | 国家ID |
| province\_name | varchar | 255 | 否 | 否 | 国家名称 |
| current\_confirmed\_count | int | 0 | 否 | 否 | 当前确诊 |
| confirmed\_count | int | 0 | 否 | 否 | 累计确诊 |
| suspected\_count | int | 0 | 否 | 否 | 疑似病例 |
| cured\_count | int | 0 | 否 | 否 | 已治愈数 |
| dead\_count | int | 0 | 否 | 否 | 死亡数 |
| yesterday\_confirmed\_count | int | 0 | 否 | 否 | 昨日确诊 |
| datetime | varchar | 255 | 否 | 否 | 日期 |

图5.1-12世界疫情据库设计图



# **6.应用原型设计**

## **6.1 数据爬取功能模块**

**6.1.1 Java爬虫的代码实现**

* 爬取指定页面：

HttpUtils.getHtml("https://ncov.dxy.cn/ncovh5/view/pneumonia");

解析页面中的指定内容Jsoup.parse(html);

doc.select("script[id=getAreaStat]").toString();

使用正则表达式获取 json 格式的疫情数据

定义正则表达式String pattern = "\\[(.\*)\\]";

编译成正则对象Pattern.compile(pattern);

调用 reg.matcher 来匹配 text 中符合正则表达式的内容reg.matcher(text);

定义一个空字符串jsonStr，如果 text 中的内容和正则规则匹配上就取出来，并将匹配的值赋值给空字符串jsonStr，否则在控制台打印错误信息 ： 没有匹配的内容可以爬取；

在成功的条件下，对 json 数据进行更进一步的解析；

将第一层json(省份数据)解析成为JavaBean

List<CovidBean> pCovidBeans = JSON.parseArray(jsonStr, CovidBean.class);

遍历整个省份的数据，并对每个省份数据对象设置基本的属性的信息，创建好时间，获取城市属性，并开始对城市信息进行第二轮解析

pBean 为省份，设置时间字段pBean.setDatetime(datetime); 获取城市:pBean.getCities();

* 将第二层json(城市数据)解析成为JavaBean；

List<CovidBean> cCovidBeans = JSON.parseArray(citiesStr, CovidBean.class);

遍历所有的城市信息，并对每个城市的数据对象设置基本的属性信息 ： 创建并设置设置时间对象，将省份id作为城市的pid设置上去，以及城市的简称，城市的所属省份的省份的名称。

cBean 为城市cBean.setDatetime(datetime);

setPid(pBean.getLocationId()); // 将省份id作为城市的pid

etProvinceShortName(pBean.getProvinceShortName());

setProvinceName(pBean.getProvinceName());

最后输出城市的数据。

后续需要将城市数据发送给 Kafka

将 JavaBean 转为 Json字符串再发送给 kafka

* 这里的爬虫模块中，是将kafka与SpringBoot整合在了一起，所以在这个爬虫模块中，我们只需要引入kafka相关的依赖，和SringBoot的相关依赖，通过 注解的方式将kafkaTemplate注入到项目中即可使用卡夫卡模块的相关API，用以连接Linux的卡夫卡集群，向集群中发送数据。具体实现如下 ：

kafkaTemplate.send("covid19", cBean.getPid(), cBeanStr);

获取第一层json(省份数据)中的每一天的统计数据

pBean.getStatisticsData(); // 获取每一天的具体数据

HttpUtils.getHtml(statisticsDataUrl);// 获取每一天的具体数据的url，因为每天的具体数据对应的是一个网络url地址，所以我们需要先获取到这个url地址，然后进行访问，才可以拿到每一天的具体的数据。

获取 statisticsDataStr 中 data(每一天的统计数据) 字段对应的数据

JSON.parseObject(statisticsDataStr);

获取每一天的具体的data数据内容 ： jsonObject.getString("data");

将解析出来的每一天的数据设置回 pBean 中的 statisticsData 字段中，之前该字段是一个setStatisticsData(dataStr);

setCities(null); // 设置城市属性信息

输出对象信息，后续需要将省份数据发送给 Kafka

将 JavaBean 转为 Json字符串再发送给 kafka

JSON.toJSONString(pBean);

卡夫卡将具体数据发送到kafk集群中。

kafkaTemplate.send("covid19", pBean.getLocationId(), pBeanStr);

## **6.2数据功能模块**

新冠疫情管理：

以数字形式，在不同区域或密度图中着色，标明经证实、可疑、死亡、愈合、国外输入的总数，以及国外输入的总数，相对昨天增加和按地区分布

其次，该表显示了外向输入趋势、新的流行趋势、综合诊断趋势、出院趋势、患者类型趋势、男女比例、年龄分布等。D。水平轴-日期，纵向轴-数字

第三，该表显示了输入的外部趋势、流行病的新趋势、综合诊断趋势、出院趋势、患者类型趋势、男女比例、年龄分布等。D。水平轴是区域名称和纵向轴数。

### **6.2.1 使用Spark分析数据**

创建 Spark 控制器层、业务层、以及SparkUtil类来实现java集成spark进行数据分析。

* 控制器层：

@Autowired

private SparkService sparkService;

@GetMapping("/getNumByProvince")

public List<CovidBean> getNumByProvince() {

List<CovidBean> list = sparkService.getNumByProvince();

return list;

}

* 业务层：

// 通过Spring装配 SparkSession 对象

@Resource

private SparkSession sparkSession;

// 具体的数据分析方法

public List<CovidBean> getNumByProvince(){

// Sql语句

String sql = "SELECT province\_name,sum(confirmed\_count) FROM covid\_data group by province\_name";

// 调用 SparkUtil 中对应的方法进行数据统计

return SparkSqlUtil.getNumByProvince(sparkSession, sql);

}

SparkUtil工具类：

// 指定要读取的 json 文件路径

private static final String tempPath1 =

System.getProperty("user.dir") +

"\\src\\main\\resources\\getNumOfProvince.json";

// 具体的数据统计方法

public static List<CovidBean> getNumByProvince

(SparkSession sparkSession, String sql) {

// 通过 SparkSession 对象的 .read().json() 方法读取文件，获取数据集对象

Dataset<Row> df = sparkSession.read().json(tempPath1);

// 创建一个临时表，并进行算子操作

df.createOrReplaceTempView("covid\_data");

df = sparkSession.sql(sql);

// 通过读取的数据集获取数据列表

List<Row> rows = df.collectAsList();

// 遍历并将属性值设置到CovidBean对象实例中

List<CovidBean> list = Lists.newArrayList();

for (Row row : rows) {

CovidBean vo = new CovidBean();

vo.setProvinceName(row.getAs(0).toString()); vo.setConfirmedCount(Integer.valueOf(row.getAs(1).toString()));

list.add(vo);

}

return list;

}

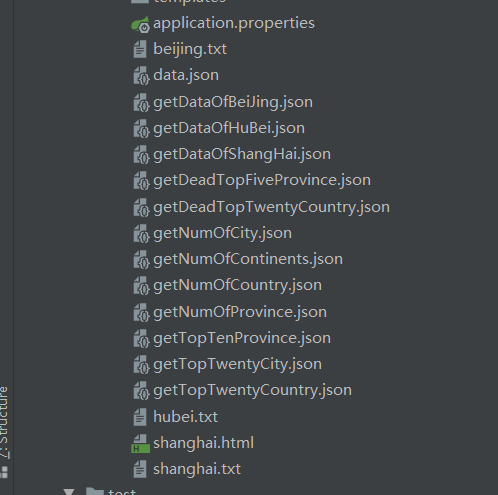


图 6.2.1-1 json文件

分析好的json文件，通过 java 集成的 Spark 技术对数据进行统计。

### **6.2.2 使用Spark处理数据**

在本次的疫情项目数据统计分析中，重点进行了以下的几个维度的统计并作展示：

* 全国各省份确诊人数统计扇形统计表图

public static List<CovidBean> getNumByProvince

(SparkSession sparkSession, String sql)

* 全国确诊人数最多的十个地区柱状图展示

public static List<CovidBean> getTopTenProvince

(SparkSession sparkSession, String sql)

* 上海确诊、治愈、死亡折线图展示

public static List<ResultVOPlus> getDataOfShangHai

(SparkSession sparkSession, String sql)

* 湖北确诊、治愈、死亡折线图展示

public static List<ResultVOPlus> getDataOfHuBei

(SparkSession sparkSession, String sql)

* 北京确诊、治愈、死亡折线图展示

public static List<ResultVOPlus> getDataOfBeiJing(SparkSession sparkSession, String sql)

* 全国死亡人数最多的五个省份扇形统计表图

public static List<ResultVO> getDeadTopFiveProvince

(SparkSession sparkSession, String sql)

* 全球国家确诊人数统计扇形表图

public static List<ResultVO> getNumOfCountry

(SparkSession sparkSession, String sql)

* 七大洲确诊人数统计扇形表图

public static List<ResultVO> getNumOfContinents

(SparkSession sparkSession, String sql)

object SparkSQL\_Demo {

def main(args: Array[String]): Unit = {

//创建上下文环境配置对象

val conf: SparkConf = new SparkConf().setMaster("local[\*]").setAppName("SparkSQL01\_Demo")

//创建SparkSession对象

val spark: SparkSession = SparkSession.builder().config(conf).getOrCreate()

//RDD=>DataFrame=>DataSet转换需要引入隐式转换规则，否则无法转换

//spark不是包名，是上下文环境对象名

import spark.implicits.\_

//读取json文件 创建DataFrame {"username": "lisi","age": 18}

val df: DataFrame = spark.read.json("D:\\dev\\workspace\\spark-bak\\spark-bak-00\\input\\test.json")

//df.show()

//SQL风格语法

df.createOrReplaceTempView("user")

//spark.sql("select avg(age) from user").show

//DSL风格语法

//df.select("username","age").show()

//\*\*\*\*\*RDD=>DataFrame=>DataSet\*\*\*\*\*

//RDD

val rdd1: RDD[(Int, String, Int)] = spark.sparkContext.makeRDD(List((1,"qiaofeng",30),(2,"xuzhu",28),(3,"duanyu",20)))

val df1: DataFrame = rdd1.toDF("id","name","age")

//df1.show()

//DateSet

val ds1: Dataset[User] = df1.as[User]

//ds1.show()

//\*\*\*\*\*DataSet=>DataFrame=>RDD\*\*\*\*\*

//DataFrame

val df2: DataFrame = ds1.toDF()

//RDD 返回的RDD类型为Row，里面提供的getXXX方法可以获取字段值，类似jdbc处理结果集，但是索引从0开始

val rdd2: RDD[Row] = df2.rdd

//rdd2.foreach(a=>println(a.getString(1)))

//\*\*\*\*\*RDD=>DataSe\*\*\*\*\*

rdd1.map{

case (id,name,age)=>User(id,name,age)

}.toDS()

//\*\*\*\*\*DataSet=>=>RDD\*\*\*\*\*

ds1.rdd

spark.stop()

}

}

case class CovidBean(key:String,value:Int)

**`b**

这部分使用Springboot技术进行实现

## **6.3用户功能模块**

相关功能有：注册、登录、修改密码、修改资料。

此外，还需要确定上述功能的开发顺序，通常按“添加>chess>>编辑”的顺序进行。

加密算法不适合密码加密，因为所有的加密算法都可以重新计算，如果使用的算法和加密时使用的参数都是已知的，那么就可以很容易地计算原始数据了。密码的安全性能做的最好的事情就是把它加密，在加密的情况下，所有的加密算法都不使用，只有通过算法。该算法的特点如下：

* 电文正文同时收到了同样的摘要；
* 在使用的摘要算法没有变化的情况下，无论原文长度如何，都会进行固定；
* 电文的文字与原文不同，其摘要不太可能相同。
* 在信息摘要领域，必须有n种不同的原稿可以计算得到完全相同的摘要。
* 当使用Message Digest 5算法执行摘要操作时，所有结果都是在32位六位系统中实现的。转换成二进制代码需要128位，这样MD5算法也变成了128位。

### **6.3.1 用户注册**

前端输入框接收用户手动输入的用户名和密码的文字信息，然后通过提交注册按钮，有ajax技术，发送到指定的控制器中，从而进入java代码实现注册操作。具体地注册并登陆Hoch，以便使用Ajax JSON传递信息，说明该登录是否注册，注册后，直接返回用户说明已经注册了，您没有使用它，并且可能会意外地将数字放在它的位置。

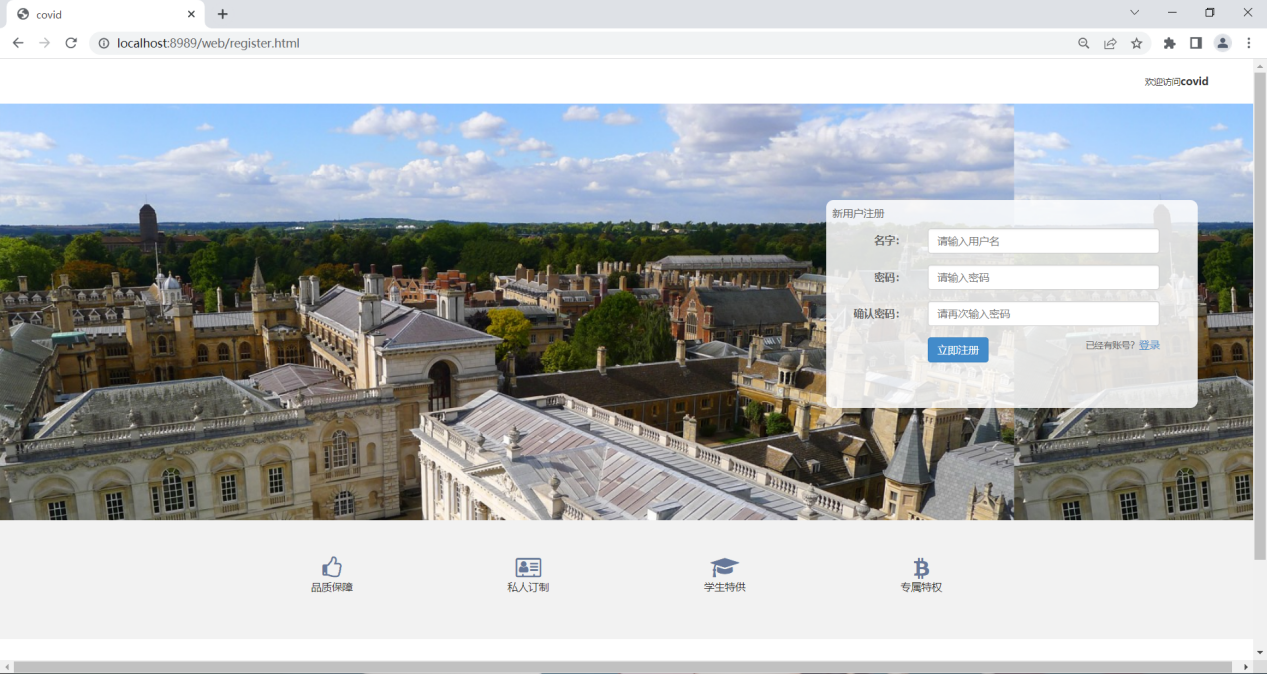


图 6.3.1-1 用户注册界面

### **6.3.2 用户登录**

前端输入框接收用户手动输入的用户名和密码的文字信息，然后通过提交注册按钮，有ajax技术，发送到指定的控制器中，从而进入java代码实现注册操作。

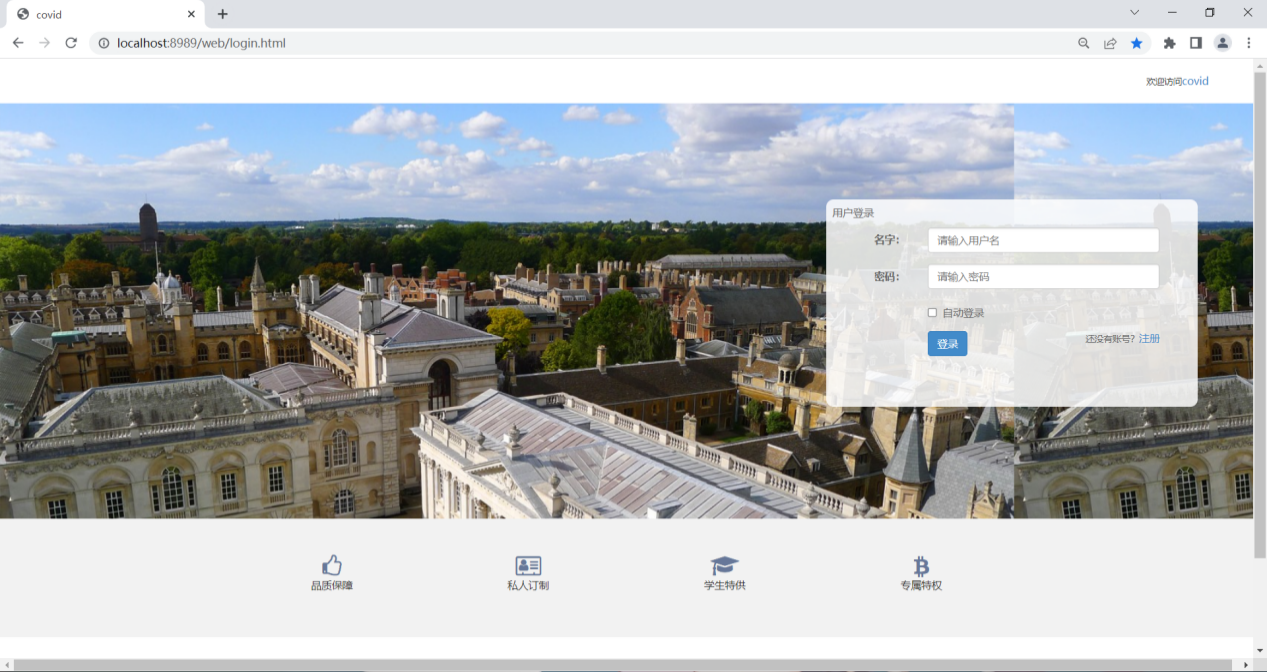


图 6.3.2-1 用户登录界面

### **6.3.3 用户信息编辑**

前端输入框接收用户手动输入的用户名和密码的文字信息，然后通过提交注册按钮，通过使用ajax技术，发送到指定的控制器中，从而进入java代码实现注册操作。

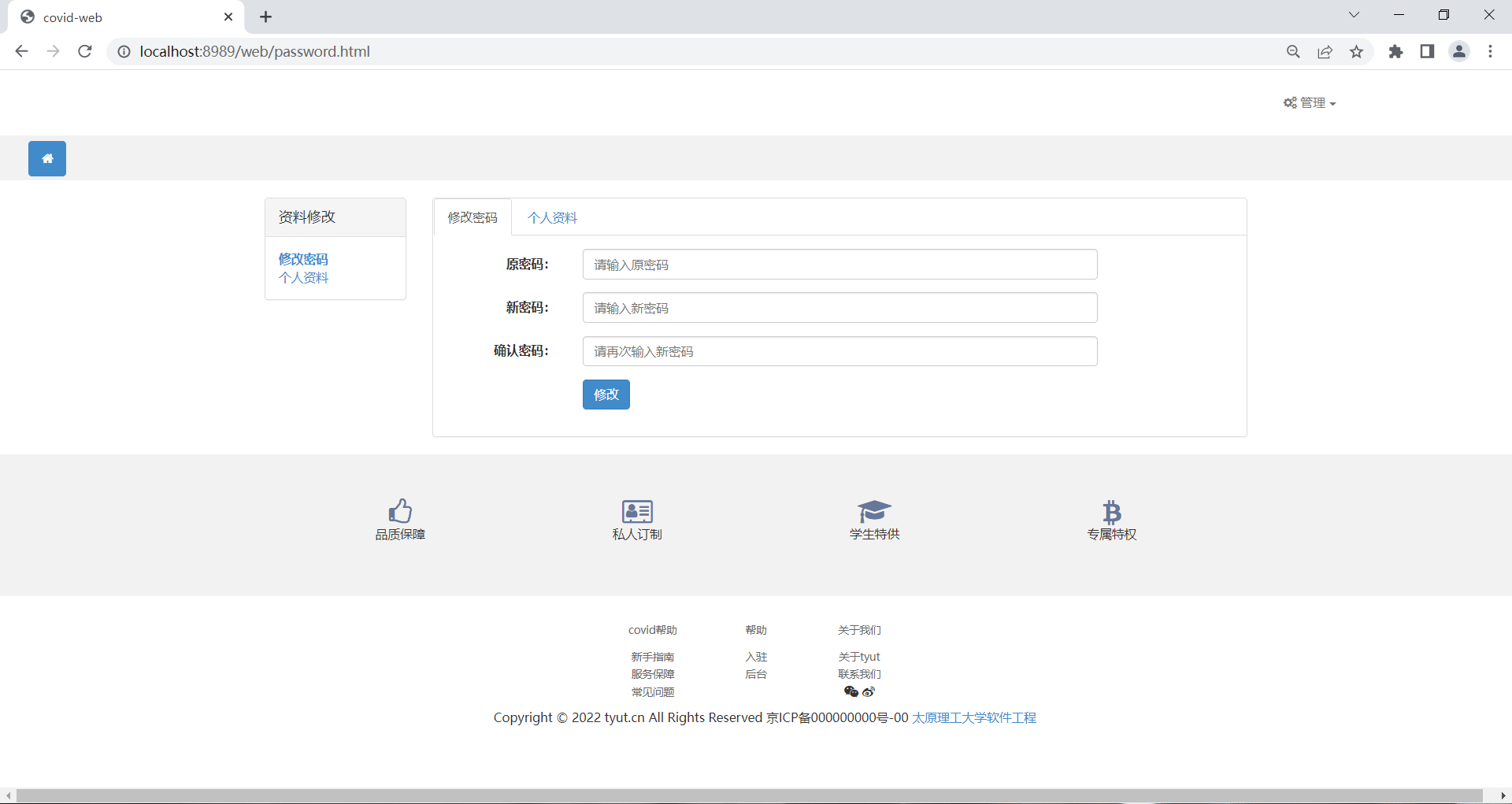


图 6.3-1 用户修改密码界面

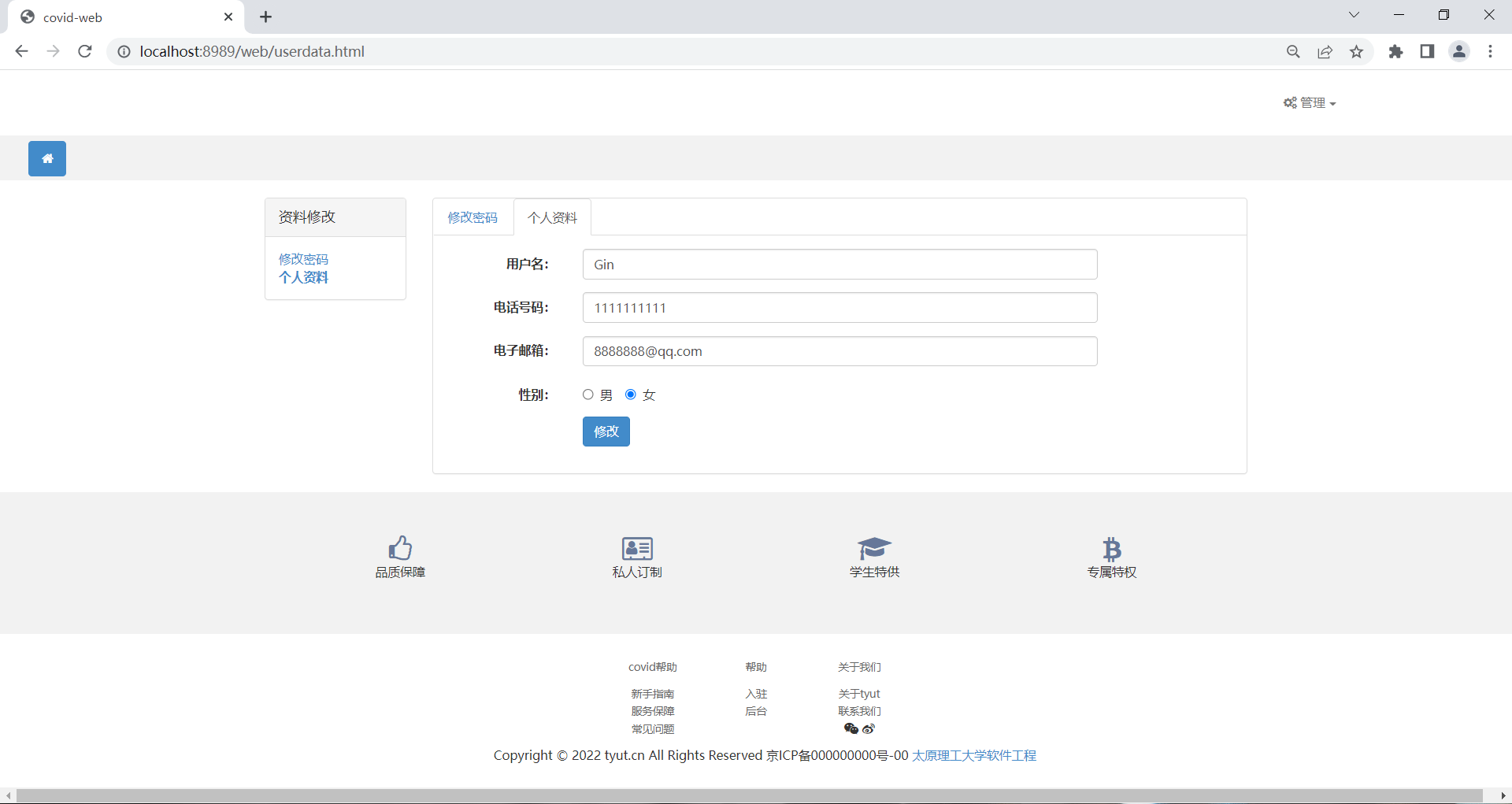


图 6.3.3-3 用户修改个人资料界面图