项目实现文档

指导老师：李宇

组长：王磊 组员：申淳元、燕怡楠、韦永剑、吴明昊

2020

目录

[1.数据处理 1](#_Toc45497782)

[1.1获取数据 1](#_Toc45497783)

[1.2数据清洗 1](#_Toc45497784)

[1.3日期分离 1](#_Toc45497785)

[2.数据建模 2](#_Toc45497786)

[2.1 ARIMA模型 2](#_Toc45497787)

[2.2 Prophet模型 2](#_Toc45497788)

[2.3 GRU模型 2](#_Toc45497789)

[2.4 SARIMA模型 2](#_Toc45497790)

[3.应用输出 3](#_Toc45497791)

[3.1微服务架构 3](#_Toc45497792)

[3.2 Java 程序调用 3](#_Toc45497793)

[3.3用户登录 3](#_Toc45497794)

[3.4数据展示 3](#_Toc45497795)

[3.5形成预测数据 3](#_Toc45497796)

[3.6用户管理 3](#_Toc45497797)

[3.7权限管理 3](#_Toc45497798)

[3.8项目部署 3](#_Toc45497799)

# 1.数据处理

## 1.1获取数据

我们从NOAA网站获取需要的数据。首先找到每个省份的省会，挑选数据较为完整的气象站下载数据，包括最高气温，最低气温，平均气温等数据。将数据以csv文件的形式下载到本地。

## 1.2数据清洗相关方法

引用python中pandas包，通过pandas读取csv文件中的数据，将需要的数据列通过表头元素选中输出到数组中，然后利用pandas.query来选择所需要的清洗数据的时间长短，最后输出到指定文件完成清洗。输出到指定文件时可以选择index=true以及header的默认，以保留表头与时间序列，方便后续运算。

同时，通过datetime包将数据中的日期由字符串形式转为date形式，便于进行日期的计算。

模块涉及代码文件列表如表1.1所示

|  |  |
| --- | --- |
| **文件名** | **文件说明** |
| last.py | 根据已下载数据集进行数据清洗，将最大最小值结果输出到文件。 |

表1.1模块涉及代码文件列表

## 1.3日期分离

根据后续模型中SARIMA模型的需要，需要将一年中的时间按照每个月的相同日期提取出来，分别储存到不同的文件中去，命名为每月中的日期，方便调用。在这里采用循环清洗，一次性就可以将数据拆分成31个不同的文件，而且利用文件的二次读取与储存，消除了文件中的日期信息，并将日期信息保存成文件名。则文件所代表的年月存储在文件内，具体日期为文件名。

## 1.4数据清洗流程图

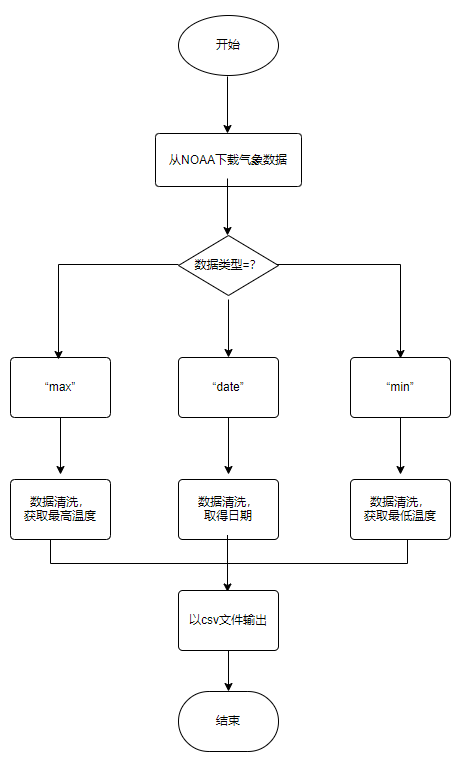
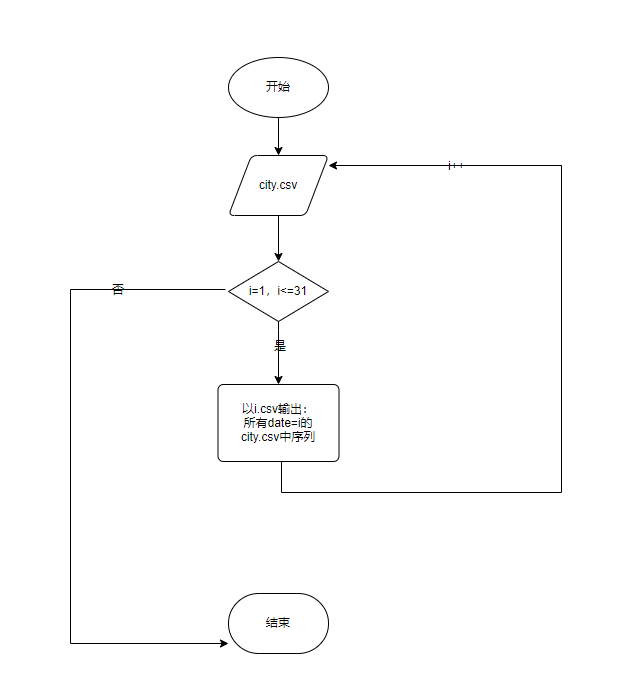


图1.1第一步数据清洗



图表1.2 第二步数据清洗

# 2.数据建模

## 2.1 ARIMA模型

用于时间序列分析和预测的ARIMA模型参数很难配置。因为有3个参数需要通过迭代试验来评估。

我们可以使用网格搜索程序自动化为ARIMA模型评估大量超参数的过程。采用超参数的网格搜索ARIMA模型。第一步，先对现有ARIMA模型利用AIC和BIC准则进行评估，评估ARIMA的误差值，与已经评估过的模型进行对比，保留结果更好的参数。第二步，在已有参数的基础上迭代枚举p,q,d参数的值，继续进行评估，搜索误差最小的参数。速度较慢，而且mse高达50。

## 2.2 fbProphet模型

Facebook 所提供的 prophet 算法不仅可以处理时间序列存在一些异常值的情况，也可以处理部分缺失值的情形，还能够几乎全自动地预测时间序列未来的走势。prophet 算法是基于时间序列分解和机器学习的拟合来做的，其中在拟合模型的时候使用了 pyStan 这个开源工具，因此能够在较快的时间内得到需要预测的结果。使用fbprophet 包含四个主要步骤：

1．输入已知的时间序列的时间戳和相应的值；

2 . 输入需要预测的时间序列的长度；

3 . 输出未来的时间序列走势。

4 . 输出结果可以提供必要的统计指标，包括拟合曲线，上界和下界等。

使用fbprophet模型后速度大幅增加，同时mse误差值也缩小至23左右。

## 2.3 GRU模型

在GRU模型中，主要调用了keras中的相关函数，因此除了制定训练集、测试集需要自行处理数据格式，只需要简单配置GRU运行环境参数，就可以利用model.fit直接输出预测误差mse，而利用model.predict就可以将预测结果输出。但是在具体实践操作中，由于数据格式传输混乱，导致耽误了许多时间。模型经过数万数据的学习，可以使得mse值保持在0.1以下，但是因为model.predict不能正常输出反归一化的结果且长时间未解决，为了不影响项目进度，暂时搁置。

## 2.4 SARIMA模型

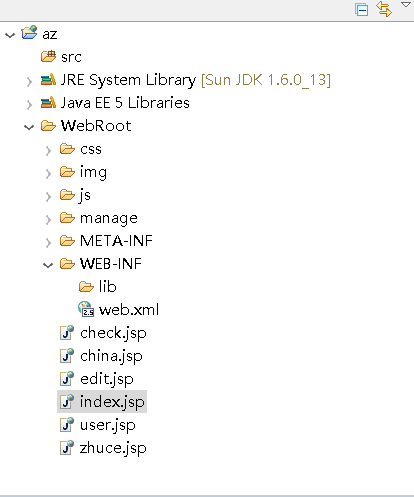
在季节性变化中，通过个人创意优化，将每个月相同的一天（二月三号与五月三号便是同一天）作为每个月的代表，然后通过12个月的周期性气温变化建立模型预测气温，通过日期分离操作中的31个文件的循环读取。因为在循环中采用了freq=’MS’作为固定频率，即每月第一天，所以在不同日期预测结束后会被追加-1,。由于在日期分离工作中已经将具体日期保存在文件名中，只需要将-1抹去再追加文件名，就可以还原时间序列。然后挨个处理并将预测结果以追加的方式添加到城市相应的气温预测csv中，并利用sort\_values根据时间序列排序，就可以得到按顺序排列的气温预测表。

# 3.Java web前端模块

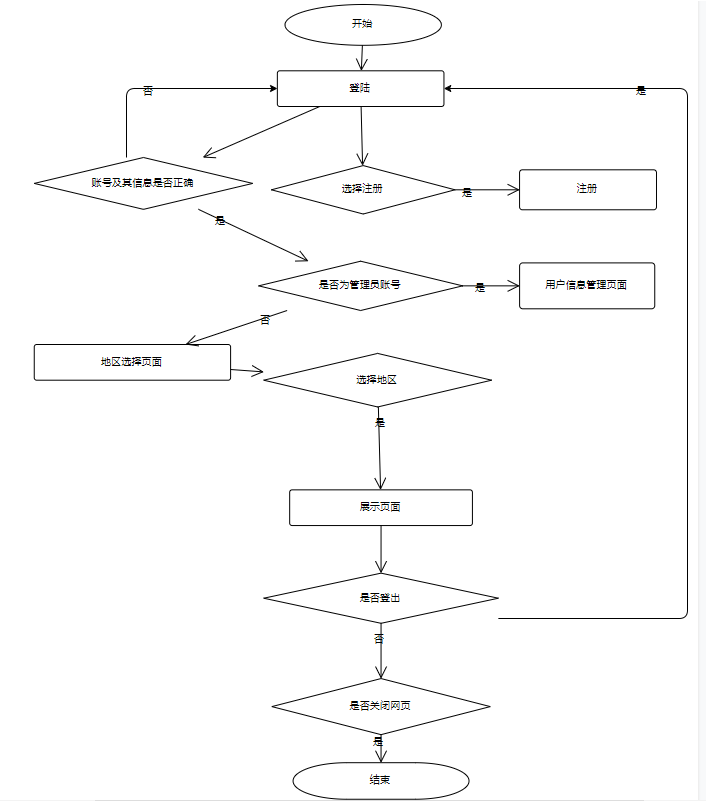
## 3.1 Java Web前端实现简介

前端以Java web项目为框架，使用css,js,html等Web前端语言实现可视化的用户操作界面,美观的界面设计和合理的功能安排不仅为用户带来良好的使用体验,也方便管理员进行后台用户权限等管理。

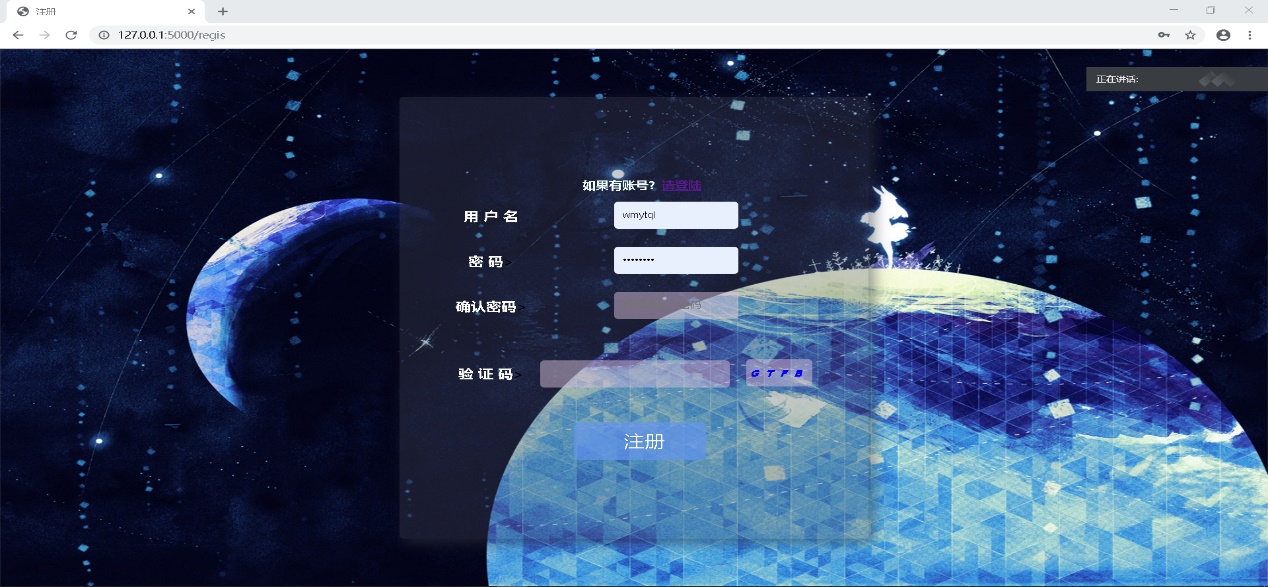
## 3.2 JavaWeb前端文件结构

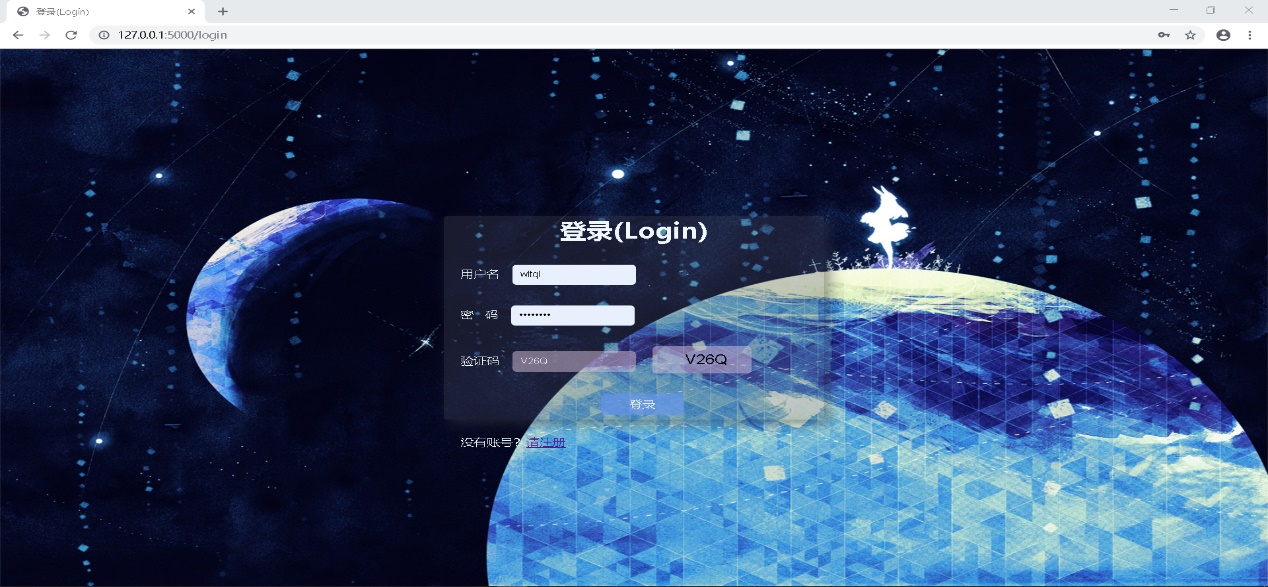


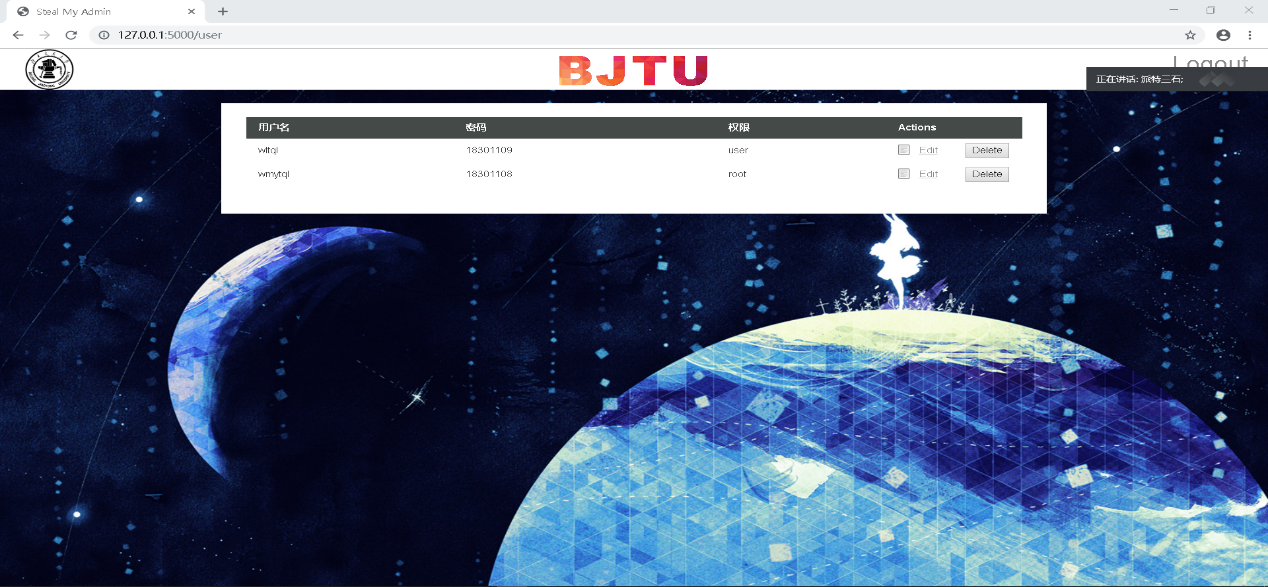
## 3.3 JavaWeb前端流程图

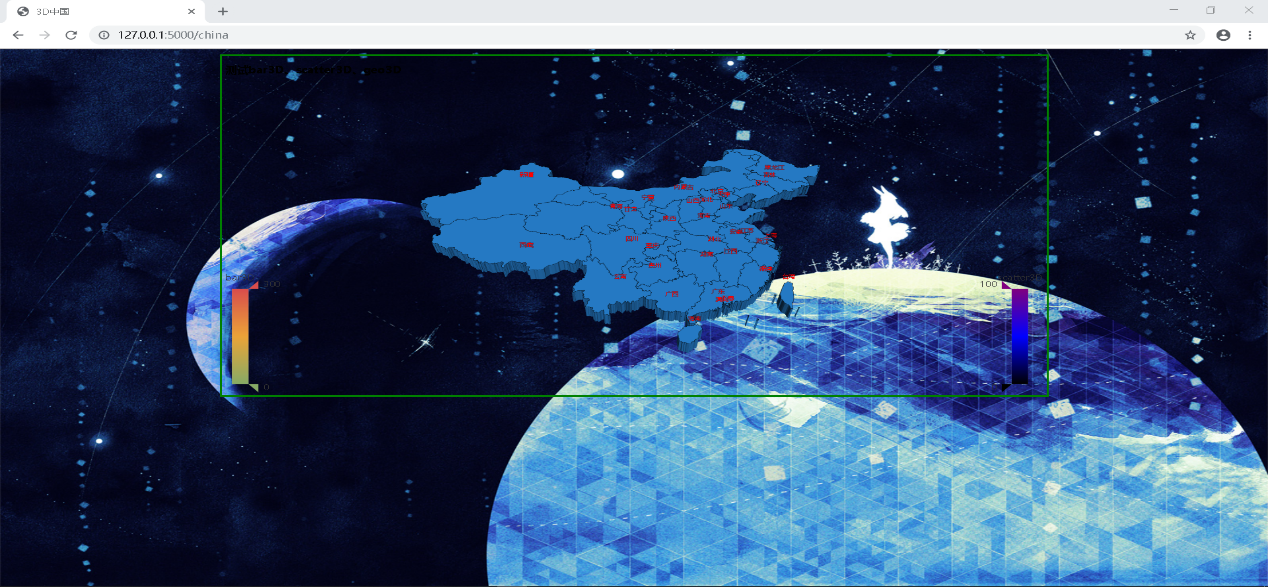


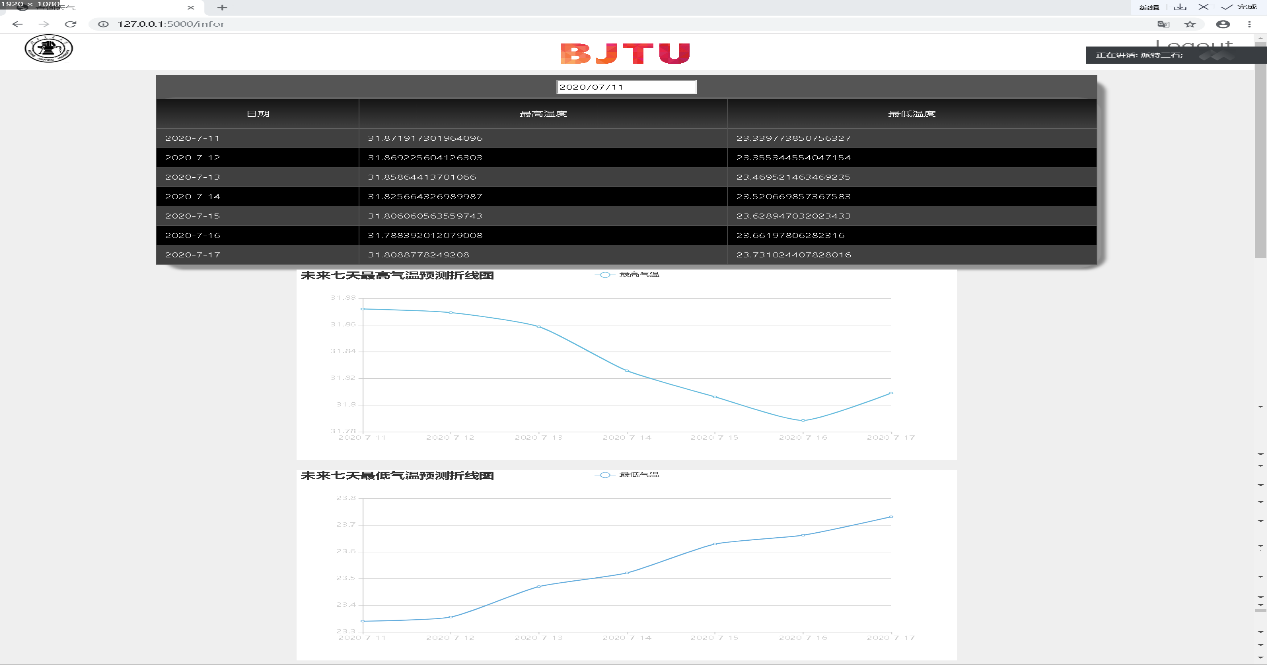
## 3.4 JavaWeb前端实现图











# 4.flask后端模块

## 4.1 Flask Web后端模块实现简介

该模块作为后端负责向Java前端发送和接受请求,与Mysql数据库交互信息。根据前端通过websocket发来的请求， 从算法模型输出的预测结果文件中读取对应日期及之后7天的天气信息，并通过websocket再发给前端。除此之外,服务器还实现了用户管理功能，对数据库中的用户信息进行增删改查。

## 4.2 Flask Web后端模块相关方法实现

Webindex.py

文件说明:flask整体框架，搭建路由，执行对数据库操作，与前端进行交互

|  |  |
| --- | --- |
| 方法名称 | 方法功能 |
| User | 数据库中用户类，包含用户名，密码，用户角色。 |
| Index | 返回登录界面url |
| test\_login(data) | 判断登录用户名密码是否匹配，返回对应权限的界面url |
| test\_register(data) | 判断用户名是否已存在，返回对应的信息 |
| test\_deleteuser(data) | 在服务器中删除用户信息 |
| test\_getusers(data) | 根据不同权限，返回不同用户信息 |
| test\_edit(data) | 修改服务器中用户信息 |
| test\_getdate(data) | 根据用户选择日期返回对应的天气数据 |

表4.1dataCleaning类方法及功能说明

Config.py

文件说明:实现后端与mysql数据库的通信

|  |  |
| --- | --- |
| 属性名称 | 属性说明 |
| SECRET\_KEY | 数据库scret key |
| SQLALCHEMY\_DATABASE\_URI | Mysql uri |
| SQLALCHEMY\_TRACK\_MODIFICATIONS | Sqlite track modifications |
| USERNAME | Mysql 用户名 |
| PASSWORD | Mysql 密码 |
| HOST | Mysql host |
| DATABASE | Mysql所用数据库 |

表4.2Config类属性说明

|  |  |
| --- | --- |
| 方法名称 | 方法功能 |
| create\_all() | 执行配置文件,数据库初始化 |

表4.3Config类方法及功能说明

User类

类说明:初始化用户信息

|  |  |
| --- | --- |
| 属性名称 | 属性说明 |
| Username | 用户名 |
| password | 密码 |
| kind | 用户角色 |

|  |  |
| --- | --- |
| 方法名称 | 方法功能 |
| \_\_repr\_\_(self) | 返回用户信息 |
| To\_json(model) | 数据库对象转为json |
| To\_json\_list() | 数据库对象列表转为json |

表4.5User方法说明

## 4.3 Flask Web后端模块文件

1. 模块涉及代码文件列表如表2.4所示

|  |  |
| --- | --- |
| **文件名** | **文件说明** |
| Webindex.py | Flask框架 |
| Config.py | 数据库配置信息 |

表4.8Flask Web模块涉及代码文件列表

# 5.云端服务器部署模块

## 5.1 云端服务器

项目选择部署到阿里云，通过xshell和xftp连接到阿里云，更新了apt后使用pip3进行依赖的管理、安装。

## 5.2 云端服务器上前端

通过virtualenv和virtualenvwrapper建立了一个基于python3的虚拟环境，安装后端所需全部依赖，由于和大多数的web框架兼容，并具有实现简单，轻量级，高性能等特点，不用像uwgsi那样手动配置，故选择用Gunicorn做Python WSGI UNIX HTTP Server，并用nginx进行反代，用Supervisor进行监视，防止异常退出。在全局下安装mysql提供数据库支持。最后上传代码，调试后运行正常

## 5.3 云端服务器上后端

安装jdk和tomcat，设置相关变量与配置，上传前端war包，运行tomcat，完成前端的部署与调试