

**气温预测分析系统**

**系统实现文档**

组长：徐子涵

组员：周子涵，李俊泽，徐紫程，李东昂

目录

[1. 引言 2](#_Toc45535883)

[1) 编写目的 2](#_Toc45535884)

[2) 项目背景 2](#_Toc45535885)

[2. 总体设计 2](#_Toc45535886)

[1) 需求概述 2](#_Toc45535887)

[2) 软件结构 2](#_Toc45535888)

[3. 程序描述 3](#_Toc45535889)

[1) 界面 3](#_Toc45535890)

[2) 功能 3](#_Toc45535891)

[3) 主要功能算法 4](#_Toc45535892)

[4) 接口 6](#_Toc45535893)

[5) 数据库设计 6](#_Toc45535894)

[6) 限制条件 6](#_Toc45535895)

# 引言

## 编写目的

本文档用于说明本项目开发的详细过程，使委托方了解软件的内部结构，从而对气温预测系统有更深入的了解，便于后期的维护和修改工作，本文档从该项目的总体设计和各个模块的功能、性能、输入输出、算法、接口、程序逻辑、储存分配及其他给出了详细的设计方案，为软件开发制定详细的计划，同时也提交系统分析员，由其给出意见。

## 项目背景

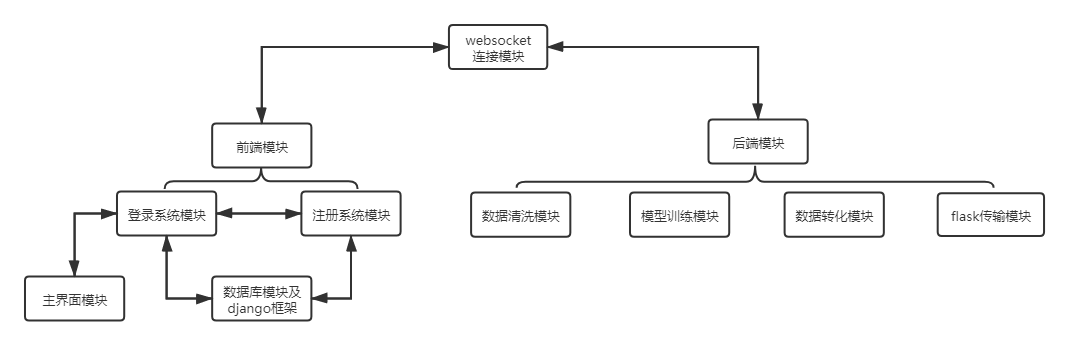
* 项目名称： 气温预测分析系统
* 开发工具：IDEA、Pycharm
* 开发语言：Python、HTML、CSS、JavaScript
* 项目的任务提出者：北京交通大学软件学院
* 开发者：徐子涵、周子涵、李俊泽、徐紫程、李东昂
* 用户：全体

# 总体设计

## 需求概述

* 界面：简洁美观可以良好交互的界面能为用户带来舒服的体验
* 功能：基本实现根据已知的气温数据预测未来七天的气温，并将预测数据传给客户端经过框架简单处理后交由前端渲染展示数据
* 登录注册：保障用户信息的安全，并简单进行权限管理
* 多城市搜索：使系统受众群体更加广泛

## 软件结构



# 程序描述

从本节开始，将逐个给出各个层次中每个模块的设计考虑。

主要功能：该天气预测系统旨在通过对过去的气象数据进行学习，生成预测模型，从而基于当前时刻的气象数据预测未来的气温，并通过网页授权给用户进行预测结果展示

可扩展性：能够适应应用要求进行相应的变化和修改，具有灵活的可扩充性。

## 界面

系统界面主框架主要包括登录界面，注册界面，系统主界面、未登录界面，界面内主要有数据表格展示，数据图表展示、城市查询窗口以及注销，同时如查询窗口，注册，登录等表单数据提交窗口都有丰富的浏览器提示信息

## 功能

* 数据清洗：对获得的数据进行清洗，滤除缺失数据，修正错误数据
* 模型训练：使用清洗后的数据对模型进行训练，从而使其对数据具有预测能力
* 数据预测：输入近日的数据，通过训练好的模型对未来的气象进行预测
* 数据转化：将预测的数据进行格式修正，使其成为可以通过网络进行传播且方便前端以及Django框架使用的json格式数据
* 数据传输：通过网络，将后台预测数据通过websocket远程传输至网页端
* 网页展示：网页端将获取到的气象数据进行可视化展示，生成表格和折线图柱状图等
* 用户注册：新用户进入网页浏览数据前，需要注册账号用于身份验证，新注册的用户默认为初级用户权限，注册后则自动跳转至登录界面
* 用户登录：用户输入系统账号密码后自动跳转至网页主界面
* 城市搜索：在输入框中输入想要检索的城市，如输入正确则显示城市天气信息，输入有误则会提示用户输入错误
* 权限管理：不同级别的用户有不同的访问权限，不同的访问权限允许检索的数据内容不同。

## 主要功能算法

* 数据清洗：通过调用pandas库对数据进行读取，去除其中缺失数据，生成可用数据
* 模型训练：通过调用keras库来构建模型，将可用数据按照11:1划分训练集和验证集，通过学习训练集学习模型，通过验证集来验证模型准确性
* 数据预测：通过调用keras库，加载之前的训练模型，读取今日数据生成新的预测数据
* 数据转化：调用json包，将预测的数据进行格式修正，使其成为可以通过网络进行传播且方便前端以及Django框架使用的json格式数据
* 数据传输：调用websockets包，asyncio包，json包，通过websockets库作为服务端接受客户端发送的请求以及发送对应数据，同时在接受不到客户端的有效信息时将会不断的发送与接受特定的无效信息保持心跳，防止websocket连接断开导致系统不能正常使用
* 网页展示：网页展示主要有以下几个view视图函数，
  + - Weather()：处理“weather/”url请求，在得到高级用户的登录请求后将执行此视图函数，先执行登录状态检查，如果未登录将进入提示未登录的页面，反之则通过维护一个配置文件来与websocke客户端进行状态更新，发送相应请求，flask端收到请求后进行相应处理并返回对应的json资源文件数据流，Django收到数据流后将通过json.loads()函数生成json对象并检验数据的合理和正确性，如果检验核对正确后将返回给js前端进行数据渲染，
    - low\_weather():处理“low\_weather/”url请求，在得到低级用户的登录请求后将执行此函数，随后同上，将数据渲染显示在网页上
    - loginWeb() registerWeb()：处理相应url请求，进入到注册和登录页面
    - logout()：清除用户登录状态，并跳转至登录页面
* 用户注册登录：新用户进入网页浏览数据前，需要注册账号用于身份验证，新注册的用户默认为初级用户权限，注册后则自动跳转至登录界面，用户输入系统账号密码后自动跳转至网页主界面
  + - checkLogin() register() ：首先读取get请求的参数，注册函数将读取数据库数据条数并以此分配唯一ID，连同表单数据一起录入到数据库中，并将用户权限默认分配为初级用户，登录函数将根据ID读取数据库该行数据，并与密码，权限进行比对，如果错误将返回信息，前端进行相应提示信息，然后根据权限分配url请求进入到不同的网页主页面
* 城市搜索：在输入框中输入想要检索的城市，如输入正确则向框架发送信息请求，框架得到请求后转而向模型预测flask端发送ws请求，并得到相应的json数据，经过合理性和城市ID的检验后返回至前端刷新数据
* 权限管理：初始设想采用RBAC模型，并成功创建了数据库迁移文件，建立了用户表，权限表，角色表，用户角色关系表，角色权限关系表，注册到了Django官方的admin管理页面，并初始添加了高级用户和低级用户的权限，创建了VIP角色和普通角色，并创建了用户和角色关系，后期打算实现中间件进行权限验证，以及对视图函数加上权限注解去实现权限验证部分，但因时间人力等因素，最终无法按期实现中间件部分，于是在用户表下添加了一个名为用户权限的列，有此权限的用户为VIP角色可以进行查询操作，无此权限的用户只能查看一个城市的天气数据，并同时有一个superUser可以对所有用户权限角色表进行操作
* websocket实现：实现了长连接，系统开始运行时就将django端和flask端通过websocket进行连接，并在无有效信息传送时不断发送ping保证连接顺畅（此方法解决了一个至今百思不得其解的bug，我直接登录页面websocket可以正常使用，我注册后先登录已有的账号可以正常使用注销后登录新注册账号可以正常使用，我注册后直接登录现有账号后websocket发送信息就会断开产生报错，通过添加调试信息打印发现数据库操作并未出现问题，后来考虑两次请求间隔时间太长导致连接被毙（但是我直接登陆就是挂一下午还是可以用中间也不会发送请求）就加入了ping，成功解决了问题，果然解决bug要用魔法打败魔法）
* Ajax实现：前端使用原生js生成xmlhttp请求，发送get请求，是异步请求，并开启一个监听器监听请求返回状态的变化并在监听到正确响应后进行相应操作，同时前端实现了轮询，当返回信息中的标识符‘token’不为1时，就认为不是应该得到的数据，发送下一次请求
* websocket和http请求完全分离：通过维护一个配置文件和资源文件来实现两种请求的完全分离，当框架收到前端发送的信息后写入配置文件，并开始监听资源文件，直到资源文件里出现了正确的数据后返回给前端（通过‘token’和‘cityID’来检验数据合理性和正确性），同时框架的websocket端通过监听配置文件，当获取到新的请求信息后就向flask端发送请求，当接收到应有数据后写入资源文件（最开始在两种协议混在一起写的时候因为异步同步不统一的问题，产生了很多问题，后来经过构思实现了这种完全分离的写法）

## 接口

* 外部接口：本软件的屏幕格式、按钮格式、输入输出方法等都是完全按照人性化的设计，方便用户执行各种操作。按钮简单明了，互动性强，本软件产品的用户需要通过键盘以及鼠标进行操作，随后在界面上有相应的动作反应
* 内部接口：内部模块之间的接口是通过内部函数以及websocket或ajax网络传输完成，在程序设计中用设计软件本身所带的调用函数以及第三方库实现

## 数据库设计

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| UserID | varchar(8) | N | null |  | 用户ID |
| name | varchar(20) | N | null |  | 用户名 |
| password | varchar(20) | N | null |  | 用户密码 |
| userPermission | varchar(20) | N | null |  | 用户权限 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| name | varchar(64) | N | null |  | 权限表名称 |
| url | varchar(255) | N | null |  | url名称 |
| per\_method | smallint | N | null |  | 权限方法名称 |
| argument\_list | varchar(255) | N | null |  | 参数列表 |
| describe | varchar(255) | N | null |  | 权限表描述 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 允许空 | 默认值 | 约束 | 说明 |
| name | varchar(32) | N | null |  | 角色名称 |

同时我们使用Django框架的modles的to参数生成用户角色关系表和角色权限关系表

## 限制条件

本系统操作简单，用户交互提示信息完善，浏览器需要支持JavaScript，Ajax请求未考虑ie6以下以及非主流浏览器，请使用chrome Firefox 或者ie6以上版本。推荐使用Google chrome浏览器。