

**基于 spark 技术的气温预测分析系统设计与实现**

**系统实现文档**

组长：王磊

组员：吴介豪、宋廷泽、杨翔越、余福

目录

[1. 引言 2](#_Toc42596663)

[1) 编写目的 2](#_Toc42596664)

[2) 项目背景 2](#_Toc42596665)

[2. 任务概述 2](#_Toc42596666)

[1) 目标 2](#_Toc42596667)

[2) 运行环境 2](#_Toc42596668)

[3) 需求概述 2](#_Toc42596669)

[4) 条件与限制 3](#_Toc42596670)

[3. 总体设计 3](#_Toc42596671)

[1) 运行流程 3](#_Toc42596672)

[2) 主要功能函数定义 4](#_Toc42596674)

# 引言

## 编写目的

概要设计是为了对用户的需求进行设计，进而实现整个项目，主要利用相对抽象的语言进行整体需求概括，确定软件体系结构，并解决“设计什么？”和“怎样设计？”两大问题。本文档是为了说明项目的设计思路，主体结构，模块构成，模块结构以及信息接口，并以此为设计人员提供详细的设计依据，为管理人员提供管理依据，同时也为测试人员提供测试依据。

## 项目背景

* 项目名称：基于 spark 技术的气温预测分析系统设计与实现
* 开发工具与语言：Pycharm、MySQL、Eclipse、Java、Python、JavaScript
* 项目的任务提出者：北京交通大学软件学院
* 开发者：王磊、吴介豪、宋廷泽、杨翔越、余福
* 用户：全体
* 实现项目的单位：北京交通大学软件学院小学期实践小组

# 任务概述

## 目标

对项目进行概括性设计，说明项目设计思路、整体结构、模块构成。

## 运行环境

* 操作系统：Microsoft Windows 10

## 需求概述

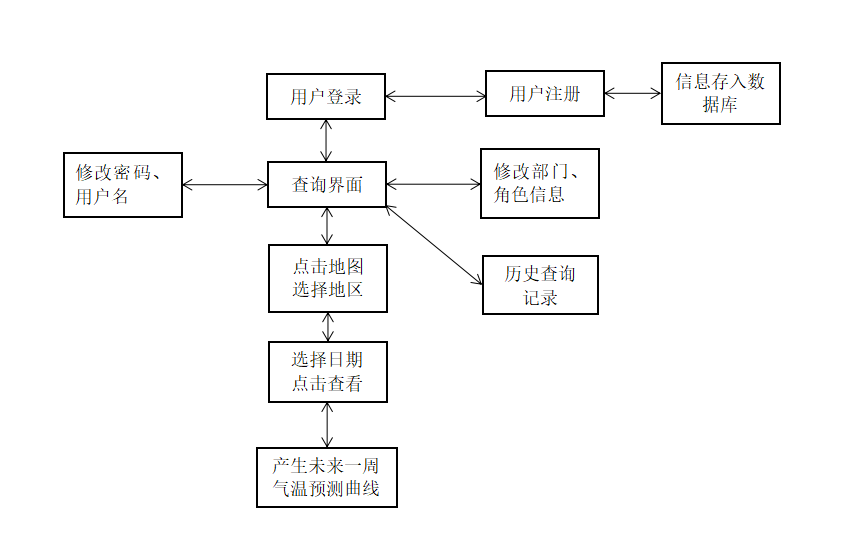
* 主要完成的功能需求有：登陆注册、气温查询、部门管理。项目运行流程为：首先玩家进行注册登录，登录后进入主界面，可以选择地图上的城市，然后选择具体日期，获取未来一周气温预测。然后，部门管理可以进行增加删除，修改用户。

## 条件与限制

* 项目应当具备的条件：相应软、硬件配置，相应开发环境以及技术人员管理人员支持。
* 工作完成期限：要求在2周内完成项目整体开发。

# 总体设计

## 运行流程



## 主要功能函数定义

**（1）前端**

* 1.处理天气：
* **public** **class** LoginController **extends** AbstractController{
* @Override
* **protected** ModelAndView handleRequestInternal(HttpServletRequest request,
* HttpServletResponse response) **throws** Exception {
* }
* 2.该函数负责接收前端页面的地区选择和日期选择 并将这两个参数传输至SocketTest函数，得到后七天的预测数据并返回到前端。
* Socket连接python函数:

**public** **static** String test(**int** city,**int** month,**int** day) {}

该函数将通过socket连接已经启动的python服务器程序，并将日期和地区传输给python服务器，并得到后期天的预测数据。

* 3.登陆页面传输函数:

@Controller

@RequestMapping("")

**public** **class** IndexController **extends** BaseController {

@RequestMapping("/index")

**public** String hello(){

**return** "index";

}

@RequestMapping("")

**public** String index() {

**return** "login";

}

}

* 4.密码加密函数:

**public** **static** String encode(String text) {}

该函数讲接收的密码通过MD5方式加密后，并返回加密过的字符串密码。

* 5.添加用户函数:
* @ResponseBody
* @RequestMapping("/addUser")
* **public** BaseModel addUser(HttpSession session, UserModel user) {}
* 6.更新用户数据函数:
* @ResponseBody
* @RequestMapping("/updateUser")
* **public** BaseModel updateUser(HttpSession session, UserModel user) {}
* 7.得到用户信息函数:
* @ResponseBody
* @RequestMapping("/getUser")
* **public** BaseModel getUser(HttpSession session, String userId) {}
* 8.删除用户函数:
* @ResponseBody
* @RequestMapping("/deleteUser")
* **public** BaseModel deleteUser(HttpSession session, String userId) {}
* 9.负责处理用户登录的函数:
* @ResponseBody
* @RequestMapping("/login")
* **public** BaseModel login(HttpSession session, String username, String password) {}

* 10.负责处理用户登出的函数:
* @ResponseBody
* @RequestMapping("/loginout")
* **public** BaseModel loginout(HttpSession session) {}

**（2）后端**

1．将预测数据转换为Json格式.

def transfer\_date(old\_value, new\_month, new\_day):

2.模型处理类和初始化、处理数据的函数

class ProcessData(object):  
 def \_\_init\_\_(self,data,predict\_year,data\_type):

def process\_minmax(self):

3. 测试Websocket的函数

async def hello(uri):

async def echo(websocket, path):

4. 设置路由以及相关函数，通过参数month和day来返回预测数据

@app.route('/data/<int:month>/<int:day>')

def tempForcast(month,day):