商贸学院

Business College of Southwest University

**害虫识别平台软件设计说明**

队伍： KKSK队

队员：

完成时间：2022年6月1日

害虫识别平台软件设计说明

目录

[Business College of Southwest University 1](#_Toc105516850)

**[害虫识别平台软件设计说明](#_Toc105516851)** [1](#_Toc105516851)

[害虫识别平台软件设计说明 2](#_Toc105516852)

[1引言 3](#_Toc105516853)

[1.1标识 3](#_Toc105516854)

[1.2系统概述 3](#_Toc105516855)

[1.3文档概述 3](#_Toc105516856)

[2 界面 3](#_Toc105516857)

[2.1 PC端 3](#_Toc105516858)

[2.2移动端网页 5](#_Toc105516859)

[2.3安卓APP 6](#_Toc105516860)

[3系统体系结构设计 6](#_Toc105516861)

[3.1系统总体设计 6](#_Toc105516862)

[3.1.1概述 6](#_Toc105516863)

[3.1.2设计思想 7](#_Toc105516864)

[3.1.3基本处理流程 8](#_Toc105516865)

[3.1.4系统体系结构 9](#_Toc105516866)

[3.2系统部件 10](#_Toc105516867)

[3.3接口设计 11](#_Toc105516868)

[3.3.1接口标识和图表 11](#_Toc105516869)

[4运行设计 13](#_Toc105516870)

[4.1系统初始化 13](#_Toc105516871)

[4.2运行结束 13](#_Toc105516872)

[5系统出错处理设计 14](#_Toc105516873)

[5.1出错信息 14](#_Toc105516874)

[5.2补救措施 14](#_Toc105516875)

[6系统维护设计 14](#_Toc105516876)

[7尚待解决的问题 14](#_Toc105516877)

# 1引言

## 1.1标识

本软件为西南大学商贸学院软件设计大赛作品。

版本号：v1.0.0 beta

## 1.2系统概述

此软件设计旨在通过用户上传害虫图片，系统使用VGG16模型识别返回给用户对应结果以及相关信息。此外，用户也可以将害虫数据上传到系统数据库中，系统会根据IP地址判断其地理位置，利用高德地图组件对害虫分布数据进行可视化处理。

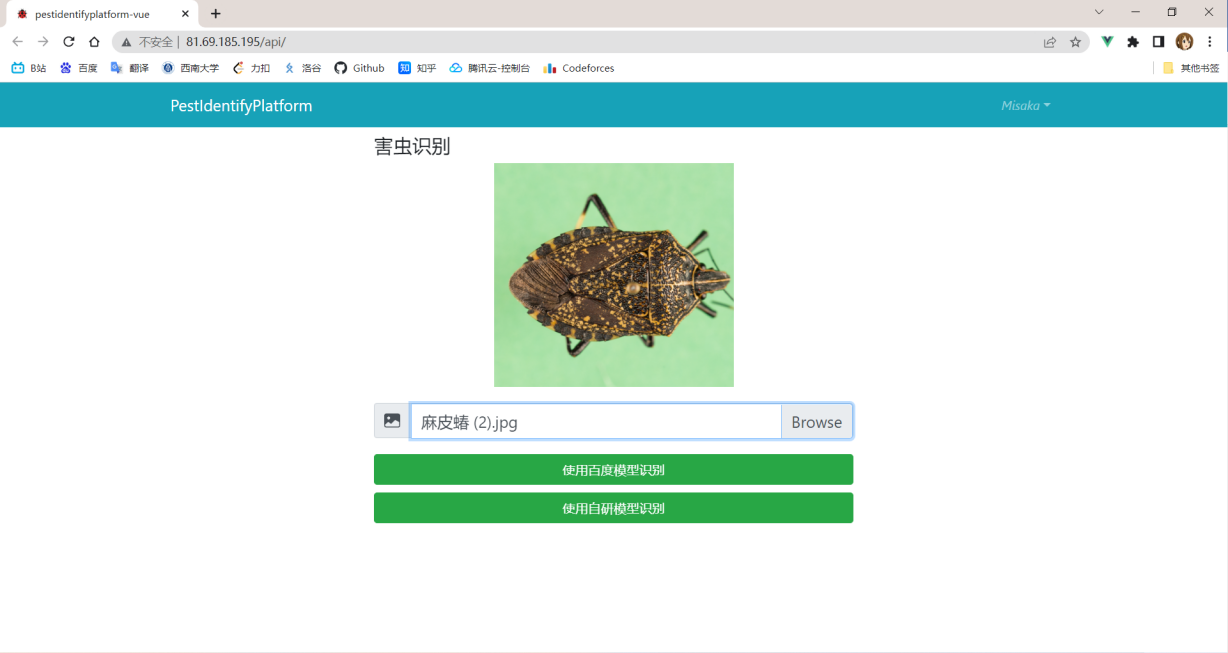
## 1.3文档概述

此软件设计文档是为了方便评委老师了解此软件，用于评价打分等用途。

# 2 界面

## 2.1 PC端

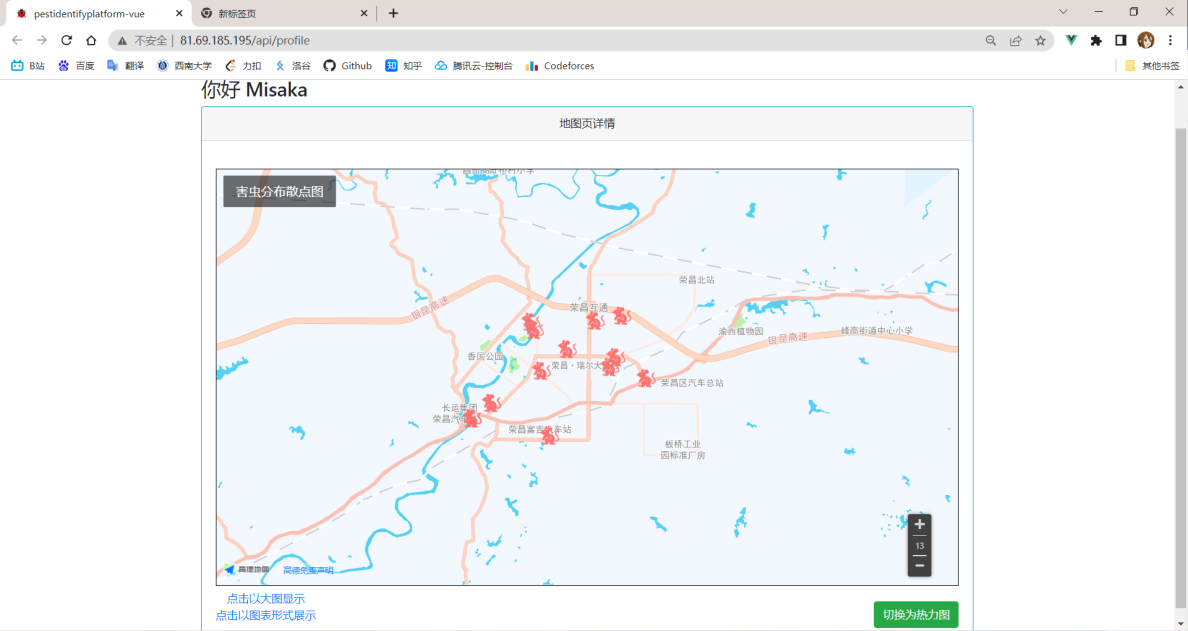
首页



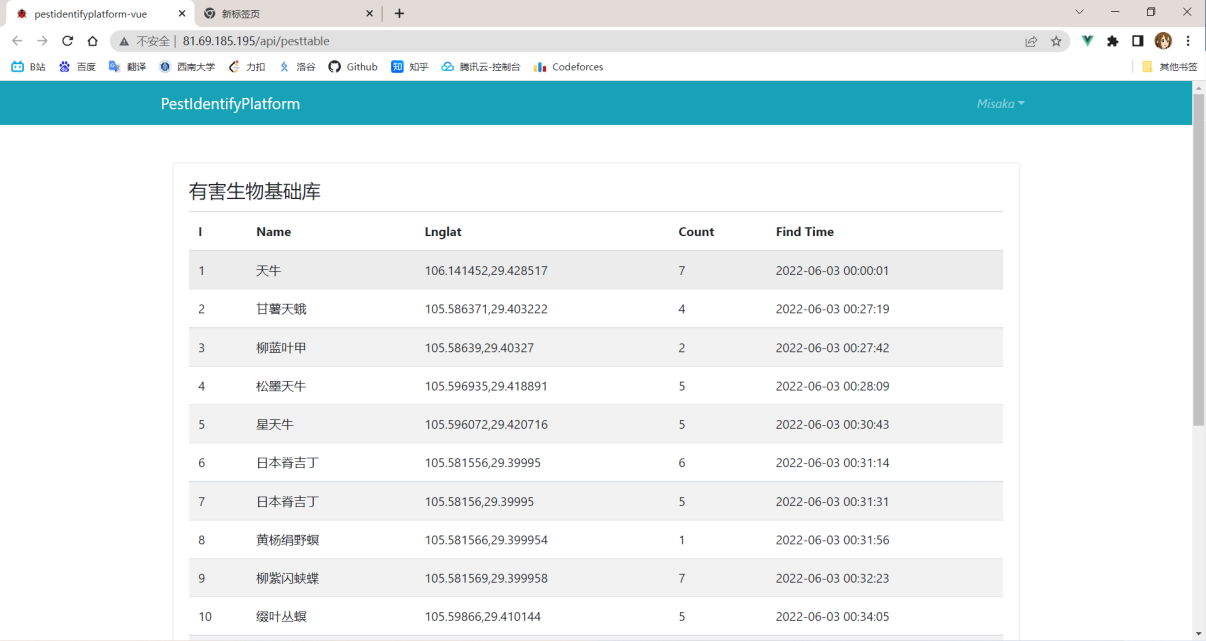
害虫数据上传页

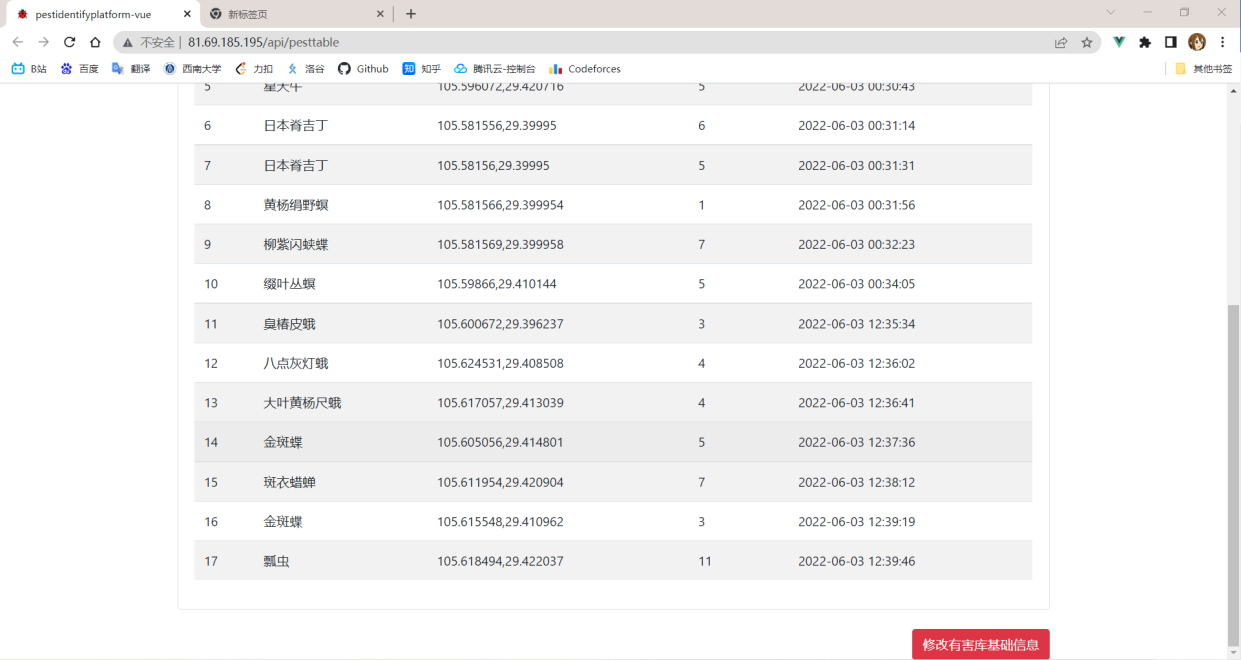


数据可视化界面



害虫数据基础库维护





## 2.2移动端网页

注册 登录 识别首页 识别结果

## 2.3安卓APP

拍照识别 实时识别

# 3系统体系结构设计

## 3.1系统总体设计

### 3.1.1概述

#### 3.1.1.1功能描述

1. 用户上传害虫图片，识别害虫相关信息并展示。
2. 在任意位置上传害虫数据，并保存到害虫数据库中。
3. 对现有害虫数据以散点图、热力图等方式进行可视化呈现。
4. **离线的情况**下利用安卓APP识别图片或视频中的害虫。

#### 3.1.1.2运行环境

后端二进制文件“go\_build\_PestIdentifyPlatform\_linux”部署于腾讯服务器Linux Centos7.0上，理应可以部署在任何Linux发行版上。

所用的数据库为MySQL 5.7，部署在腾讯服务器Linux Centos7.0的Docker容器中。前端代码部署在Nginx代理服务器。

### 3.1.2设计思想

#### 3.1.2.1系统构思

本系统设计的思想是前后端分离，前端页面通过 Ajax 调用后端的 RestFul API 并使用 JSON数据进行交互。分离业务、分离开发、分离部署，前后端之间只有HTTP(S)上的JSON通信，做到松耦合。

#### 3.1.2.2关键技术与算法

VGG16：

VGG16网络通过将图片转为输入，提取图像特征，学习网络参数，最终实现图像分类的目的 。

Nginx：

突破限制，可以配置内网中拥有公网 IP 的机器作为反向代理，从而实现对内网服务的访问。

可以部署多台服务器，配置代理将请求转发给不同的服务器。这样，当负载增大的时候，客户端无需做任何修改，服务端通过简单的增加服务器便可以应对。访问加速，对于某些大型服务，通过使用反向代理，当请求到来时，代理可以将请求转发给最近的服务器，从而让用户在最短时间内获得响应。

Gin：

基于 Radix 树的路由，小内存占用，无反射，可预测的 API 性能。

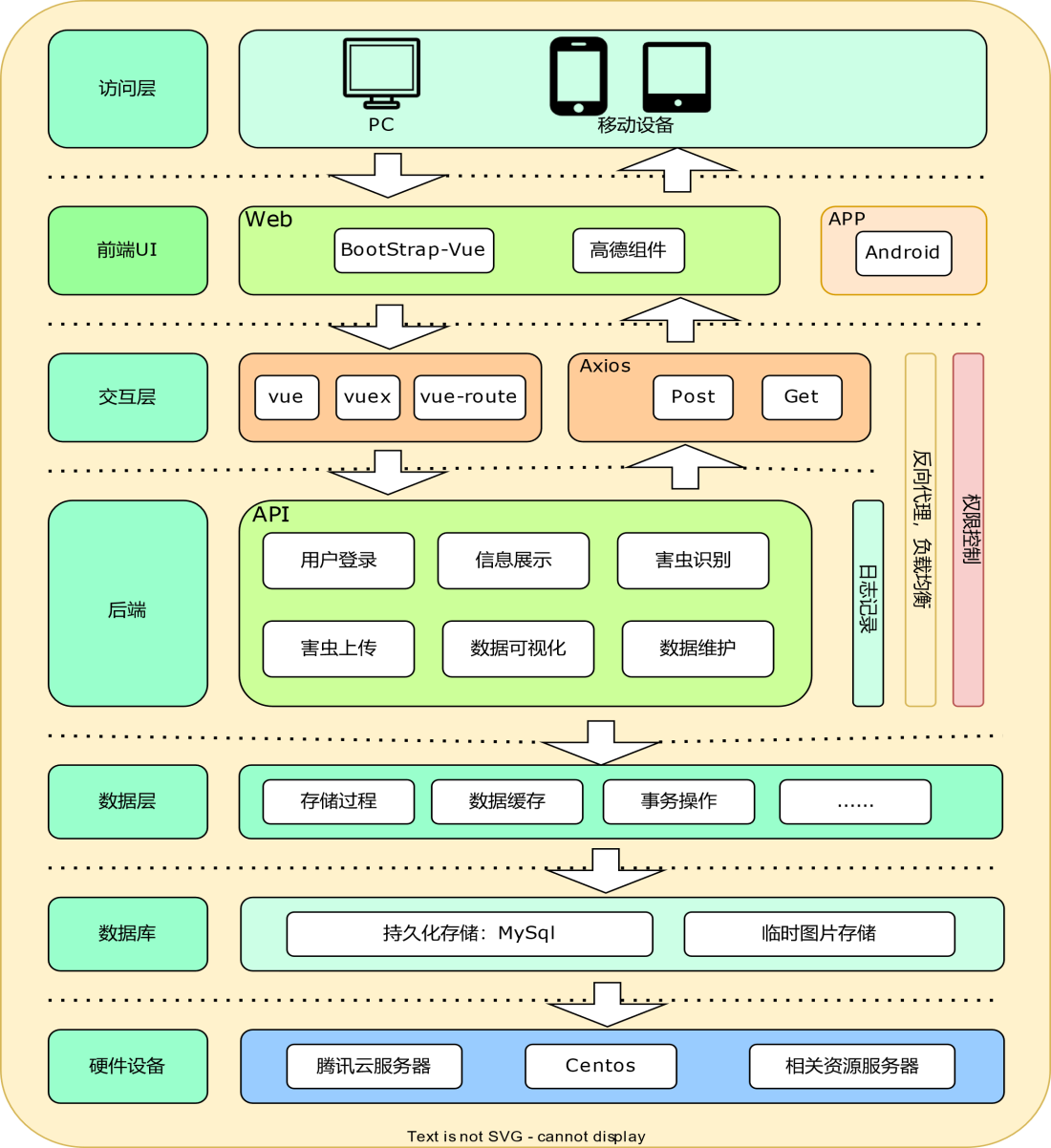
传入的 HTTP 请求可以由一系列中间件和最终操作来处理。 例如：Logger，Authorization，GZIP，最终操作 DB。提供了一种方便的方法来收集 HTTP 请求期间发生的所有错误。最终，中间件可以将它们写入日志文件或数据库。

Docker

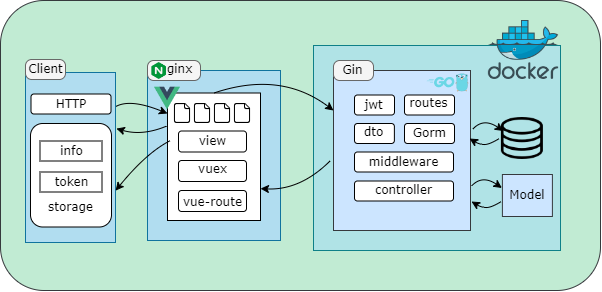
可以帮助开发人员快速构建轻量级和可移植的软件容器，从而简化应用程序开发、测试和部署等诸多环节。具体来说，它是一个用于构建应用程序的软件平台，可以提供一种小型和轻量级的执行环境用来共享操作系统内核，但在其他方面是独立运行的。

### 3.1.3基本处理流程

#### 3.1.3.1系统架构图



#### 3.1.3.2数据流程图



### 3.1.4系统体系结构

#### 3.1.4.1系统配置项

|  |  |
| --- | --- |
| 模块 | 功能 |
| main.go | 整个程序的入口 |
| routes.go | 定义路由，并实现对路由的监听 |
| common | 处理程序与其他程序的接口，如连接数据库，调用百度api接口，生成token等功能 |
| config | 定义配置文件，如运行端口、数据库用户密码、字符集、时区等 |
| controller | 定义不同的控制函数，处理用户请求 |
| dto | 做表示层(做为展示给用户数据) |
| middleware | 作为中间件，用于登录的信息获取以及鉴权以及解决同源策略 |
| model | 定义模型，如用户模型，害虫数据模型，百度接口模型 |
| resopnse | 定义统一的response接口以及标识码 |
| util | 定义功能组件，如给用户随机分配用户名，做正则等等。 |

前端模块划分

|  |  |
| --- | --- |
| 模块 | 功能 |
| assets | 注册scss文件，以实现对bootstrap-vue的使用 |
| components | Vue默认的helloworld组件 |
| router | 定义页面的路由组，以及模块化用户请求路由 |
| service | 封装用户的登录、注册请求，以及定义本地缓存服务 |
| store | 根据用户模型，同步更改用户的token数据和更新本地缓存 |
| utils | 封装功能组件，封装请求方法 |
| views | 定义页面，如登录、注册、害虫信息展示、害虫数据上传等等。 |
| App.vue | 项目的主组件，页面入口文件 ，所有页面都在App.vue下进行切换 |
| main.js | vue框架的main文件，用于引入项目所需要的npm包 |
| .eslintrc.js | 定义语法规范，如首行缩进，是否空行等。 |
| .env.develpoment | vue项目的配置文件，定义api基础地址 |

#### 3.1.4.2 系统配置项设计

后端配置项文件：

|  |  |
| --- | --- |
| server.port | 后端运行的端口 |
| database.diverName | 数据库驱动的类型 |
| database.host | 数据库运行的ip |
| database.port | 数据库运行的端口 |
| database.database | 数据库的数据库名 |
| database.username | 数据库的用户名 |
| database.password | 该用户的密码 |
| database.charset | 数据库使用的字符集 |
| database.loc | 数据库所用的时区 |
| baidu.clint\_id | 百度api的id |
| baidu.clint\_secret | 百度api的密钥 |

## 3.2系统部件

* 采用 Intel® Xeon® Cascade Lake 处理器，主频2.5GHz，睿频3.1GHz
* 使用 Intel Advanced Vector Extension (AVX-512) 指令集
* 处理器为1核
* 运行内存为2G
* 用块存储设备映射（device-mapping）将存储设备映射为自身可以识别的位置
* 最高支持29Gbps内网带宽

## 3.3接口设计

接口为URL路径的restful风格，后端提供接口，前端负责调用。接口分为四个部分组成：接口请求地址（url）、接口请求方式（get/post）、请求数据（request）、响应数据（response）

### 3.3.1接口标识和图表

#### api接口列表

|  |  |
| --- | --- |
| 接口 | 说明 |
| /api/auth/register | 用户注册 |
| /api/auth/login | 用户登录 |
| /api/auth/info | 用户信息展示 |
| /api/uploadimg | 用户上传图片 |
| /api/uploadlocation | 用户上传害虫数据 |
| /api/sendlocation | 发送害虫分布数据 |

#### 接口详情

* 用户注册
* 接口地址：/api/auth/register
* 返回格式：JSON
* 请求方式：POST
* 接口备注：用户提交注册信息，后端验证后注册成功，发放token
* 请求参数说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **必填** | **说明** |
| name | string | false | 用户名 |
| email | string | true | 邮箱 |
| telephone | string | true | 手机号 |
| password | string | true | 密码 |

* 返回参数说明：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| code | int | 状态码 |
| data | string | token |
| msg | string | 注册成功 |

* 用户登录
* 接口地址：/api/auth/login
* 返回格式：JSON
* 请求方式：POST
* 接口备注：用户提交登录信息，后端验证后登录成功，发放token
* 请求参数说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **必填** | **说明** |
| email | string | true | 邮箱 |
| password | string | true | 密码 |

* 返回参数说明：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| code | int | 状态码 |
| data | string | token |
| msg | string | 登录成功 |

* 查看用户信息
* 接口地址：/api/auth/info
* 返回格式：JSON
* 请求方式：GET
* 接口备注：用户在用户页面查看用户信息
* 请求参数说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **必填** | **说明** |
| null | null | false | 无 |

* 返回参数说明：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| code | int | 状态码 |
| data | Object | dto用户模型 |
| msg | string | 空 |

* 上传图片
* 接口地址：/api/uploadimg
* 返回格式：JSON
* 请求方式：POST
* 接口备注：用户在主页上传图片到服务器
* 请求参数说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **必填** | **说明** |
| imgBase64 | string | true | base64格式 |

* 返回参数说明：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| code | int | 状态码 |
| data | Object | responseModel |
| msg | string | 成功获得数据 |

* 上传害虫信息和经纬度
* 接口地址：/api/uplocation
* 返回格式：JSON
* 请求方式：POST
* 接口备注：用户在主页上传害虫信息和经纬度到服务器
* 请求参数说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **必填** | **说明** |
| name | string | true | 害虫学名 |
| lng | string | true | 经度 |
| lat | string | true | 纬度 |

* 返回参数说明：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| code | int | 状态码 |
| data | null | 空 |
| msg | string | 上传数据成功 |

* 发送害虫数据信息
* 接口地址：/api/uplocation
* 返回格式：JSONP
* 请求方式：GET
* 接口备注：用户在主页显示害虫分布地图
* 请求参数说明：

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **必填** | **说明** |
| null | null | false | 空 |

* 返回参数说明：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| code | int | 状态码 |
| data | SendLocationModel | 定义的返回数据模型 |
| msg | string | 空 |

# 4运行设计

## 4.1系统初始化

docker start [容器id]

nohup ./go\_build\_PestIdentifyPlatform\_linux $

cd /usr/sbin

./nginx

## 4.2运行结束

此软件一直挂载在远程云服务器上，除非异常状况或人为结束，软件不会停止运行 。

# 5系统出错处理设计

## 5.1出错信息

后端程序会把错误信息以log形式输入到nohup.out文件中，方便检查。

## 5.2补救措施

对前后端进行重启

docker restart [容器id]

./nginx -s load

# 6系统维护设计

后端程序会把日志文件输入到nohup.out文件中，以便维护系统。

# 7尚待解决的问题

问题一、

在系统设计时只考虑到用JWT(Json Web Token)，来判断用户登录。但是没有对已登录的用户的权限（如上传数据和修改数据）做进一步的划分。

解决方案：

使用基于角色的权限访问控制RBAC，把权限与角色相关联。只要让用户成为适当角色，从而得到这些角色的权限，方便指派与回收。

问题二、

害虫数据可视化是基于用户上传数据实现的，但是很难保证用户是否在正确的地点上传正确的数据。虽然可以管理员事后审查能够修改数据，但时效性低。

解决方案：

1. 用户首先要获得上传数据的权限,才能上传数据
2. 在用户上传数据时，先将其提交给审核人员审核，审核结果正确再将其放入数据库中。

问题三、

1. 大多数害虫都是变态发育，幼虫和成虫有很大的差距，但是训练模型中没有把幼虫和成虫标记为两个标签，导致准确率下降。
2. 大多数害虫具有拟态特性，极好的与环境融为一体，很难突出主体。但是训练集里的数据没有去除背景，导致准确率下降。

解决方案：

1. 手动标记同一种害虫的幼虫和成虫。
2. 在进行卷积前，进行图像主体检测提取主体或手动对图片进行剪裁处理，提高数据集的可靠性。