2021年7月13日

离了树的叶属于地上的世界

林业有害生物智能识别小程序

详细设计说明书

目录

[1. 引言 2](#_Toc77238066)

[1.1 编写目的 2](#_Toc77238067)

[1.2 范围 2](#_Toc77238068)

[1.3 参考资料 2](#_Toc77238069)

[2. 总体设计 3](#_Toc77238070)

[2.1 需求规定 3](#_Toc77238071)

[2.2 运行环境 3](#_Toc77238072)

[2.3 详细设计 3](#_Toc77238073)

[2.3.1 页面介绍 3](#_Toc77238074)

[2.3.2 java后台架构： 6](#_Toc77238075)

[2.3.3 分析端架构： 13](#_Toc77238076)

[2.3.4 运行设计 15](#_Toc77238077)

[3. 尚未解决的问题 16](#_Toc77238078)

# 1. 引言

## 1.1 编写目的

在我国林业发展迈入快车道的背景下，森林病虫害的问题日渐凸显。而林业有害生物的多样性和防治环境的复杂性则给防控工作带来了一定的困难，为了满足防治需求，现结合微信小程序平台与人工智能领域中的目标检测开发出一套采用了Client/Server结构的林业有害生物识别系统， 为方便用户理解，本说明书对该系统的技术细节与应用场景进行了相关阐述。

## 1.2 范围

软件系统名称：林业有害生物智能识别小程序

开发者：郑浩然、张权巍、陈智勇

## 1.3 参考资料

[微信小程序官方文档](https://developers.weixin.qq.com/miniprogram/dev/framework/)

[YOLOv5-P6](https://github.com/ultralytics/yolov5/releases)

Java程序设计基础

[detect.py注释解析](https://blog.csdn.net/weixin_44152895/article/details/110009680)

[spring-boot官方文档](https://spring.io/projects/spring-boot)

# 2. 总体设计

## 2.1 需求规定

系统旨在满足用户在害虫防治时的辨识需要，于使用者而言主要分为两类：一类是小程序使用者，用户进入小程序后可采用拍照或从相册内获取图片两种方式取得害虫数据并获知分析结果，还可在搜索栏搜索相关害虫获取介绍数据；另一类是程序维护人员，其可在PC端登录后台管理系统进行害虫数据的维护。

## 2.2 运行环境

操作系统：windows 10

数据库管理系统：MySQL V5.0.22

相关软件工具：Spring-Tools-Suite 3.9.6

开发平台：微信开发者工具

环境：Python 3.7.10、PyTorch1.8.1+CUDA10.2、Flask1.1.2

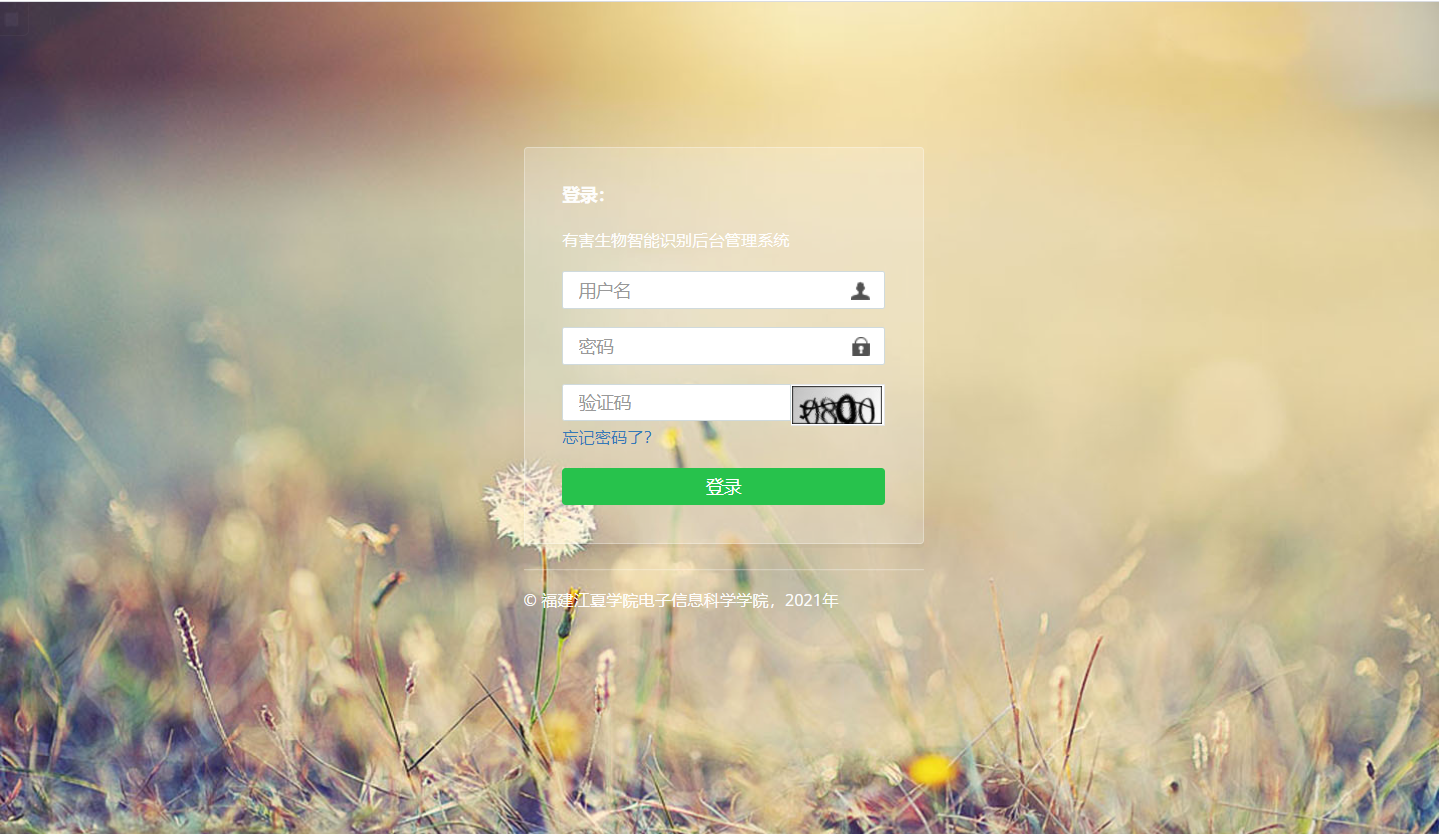
## 2.3 详细设计

### 2.3.1 页面介绍

小程序前台页面：

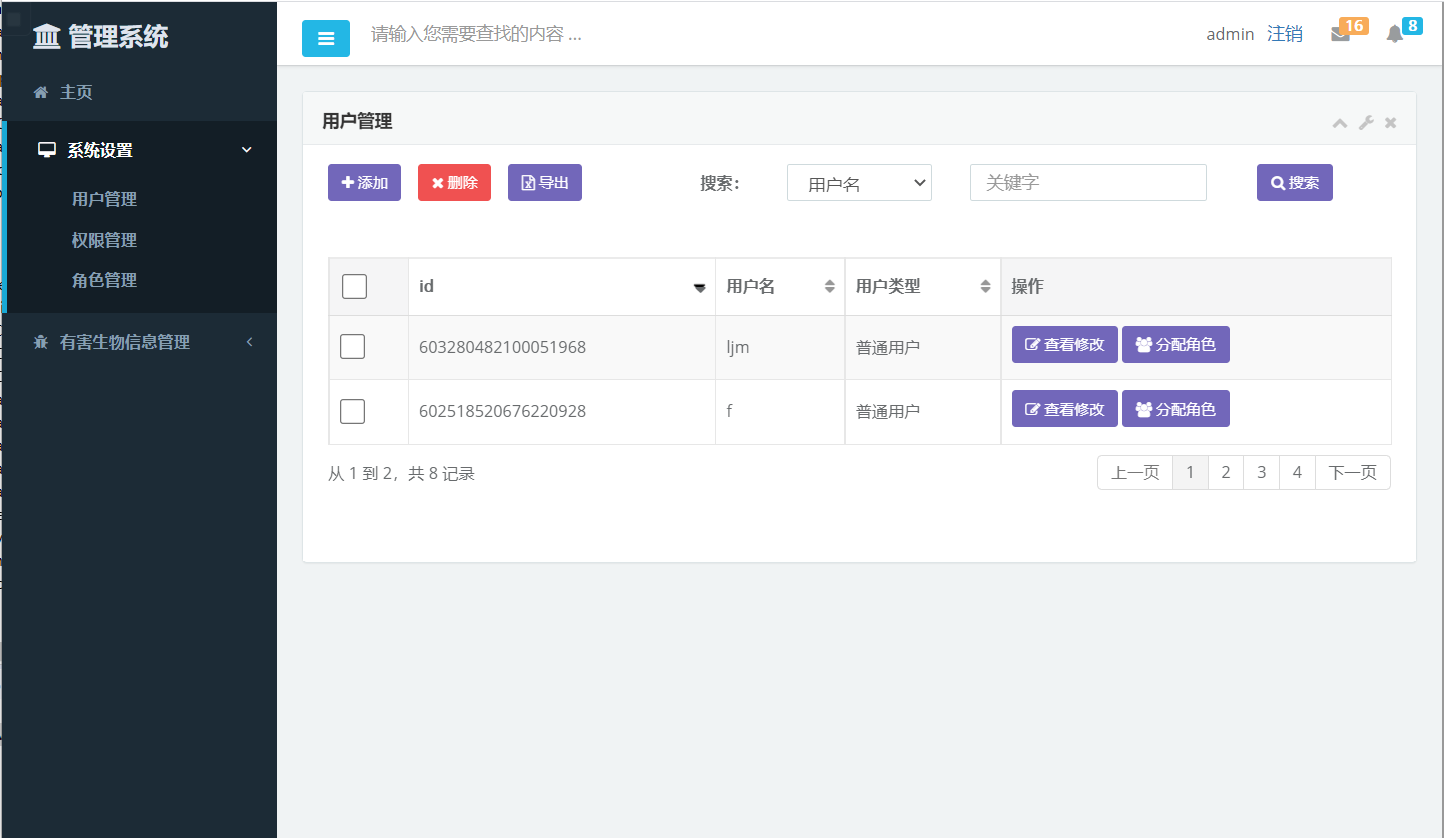


后台管理登录页面

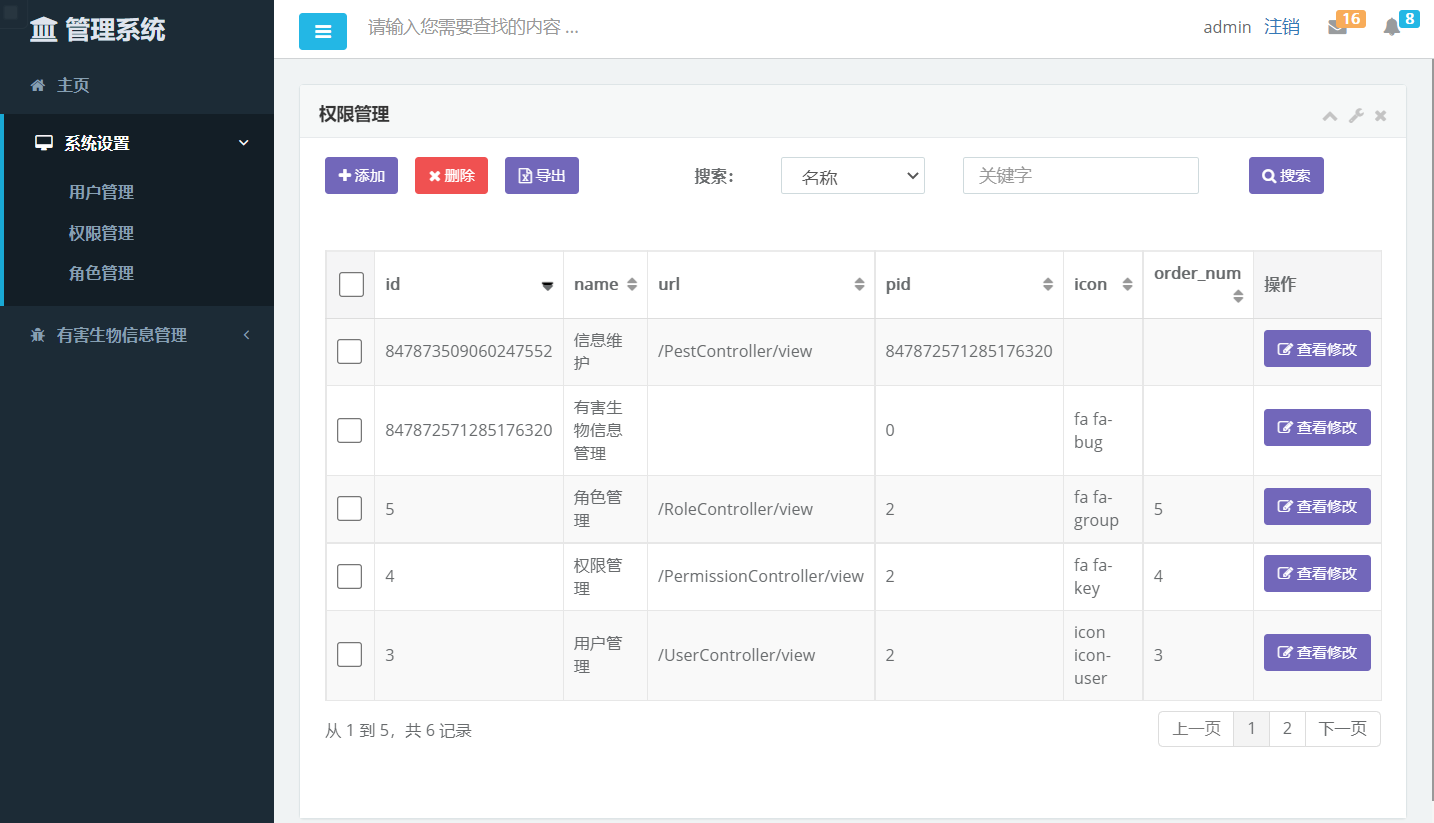


后台管理页面

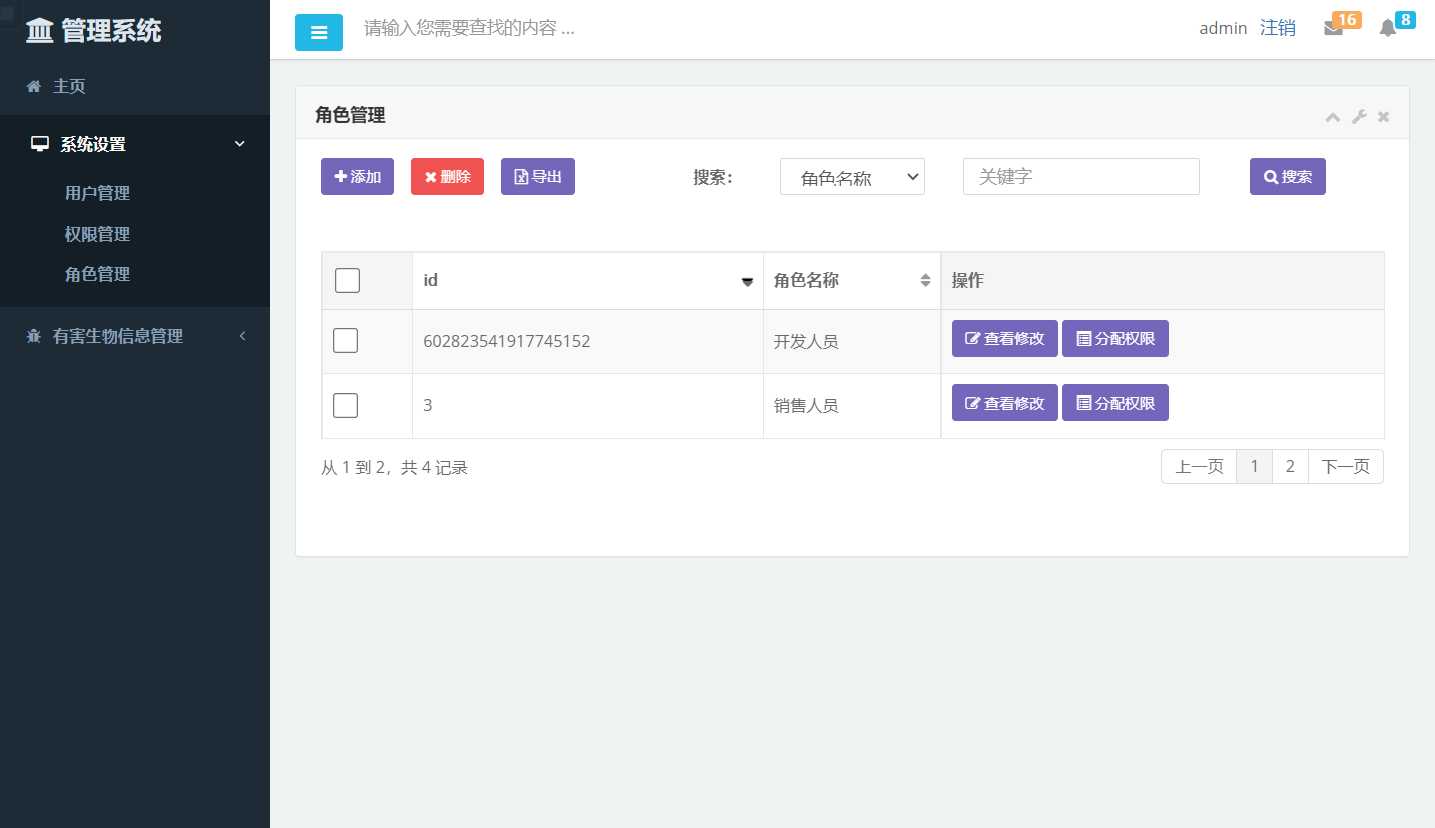
用户管理界面



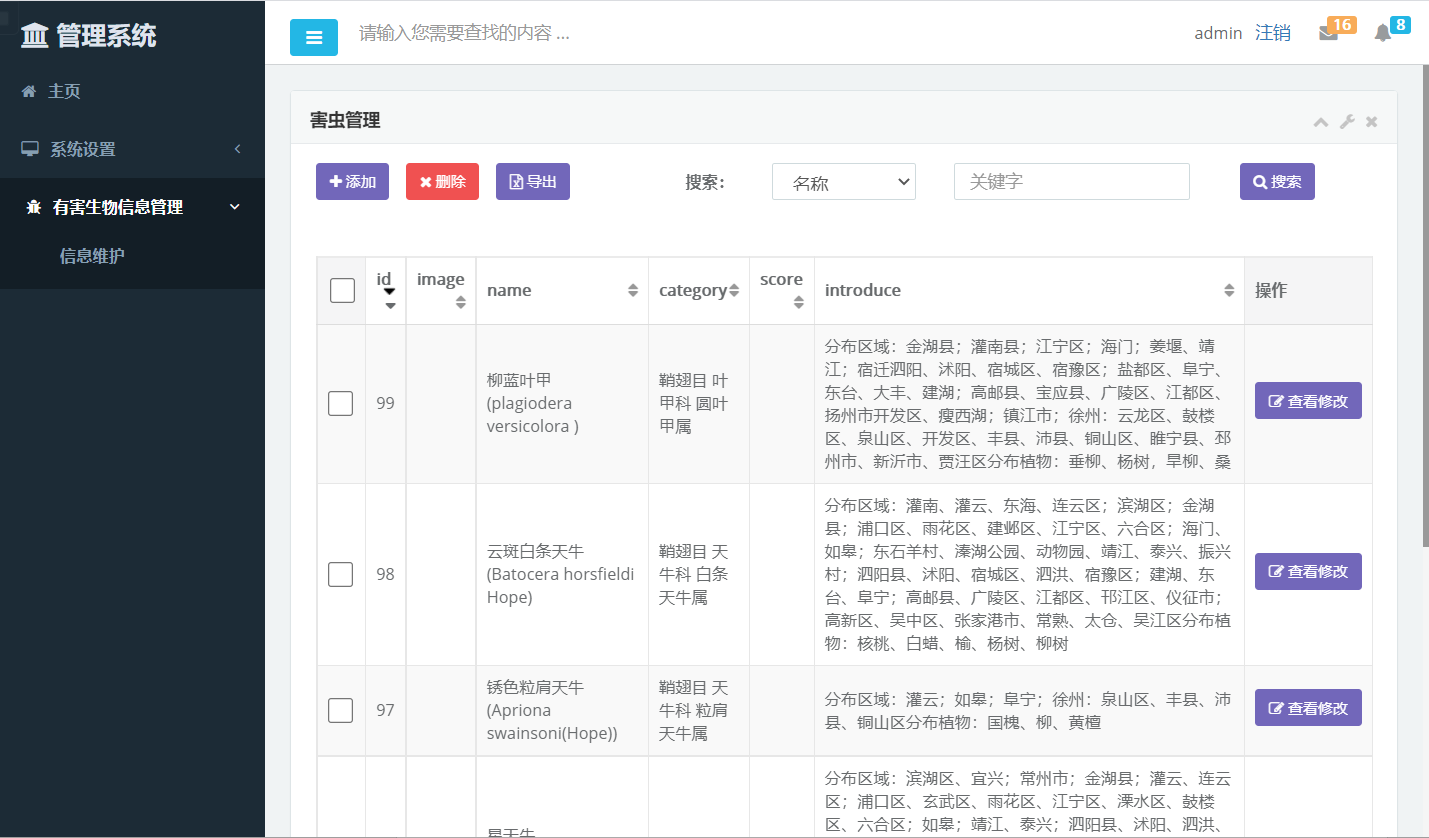
权限管理界面



角色管理界面



有害生物信息维护界面



### 2.3.2 java后台架构：

总体基于java SpringBoot框架，应用雪花主键、MD5加密、Thymeleaf模板等多种技术构建。

#### 2.3.2.1 功能需求：

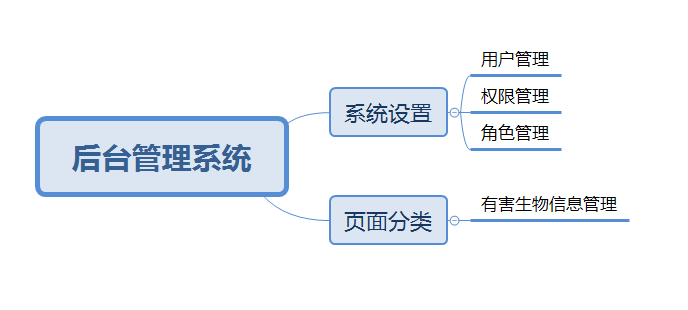
（1）用户管理：实现用户的增删改查和用户的角色分配。

（2）权限管理：实现权限的增删改查。

（3）角色管理：实现角色的增删改查和角色的权限分配。

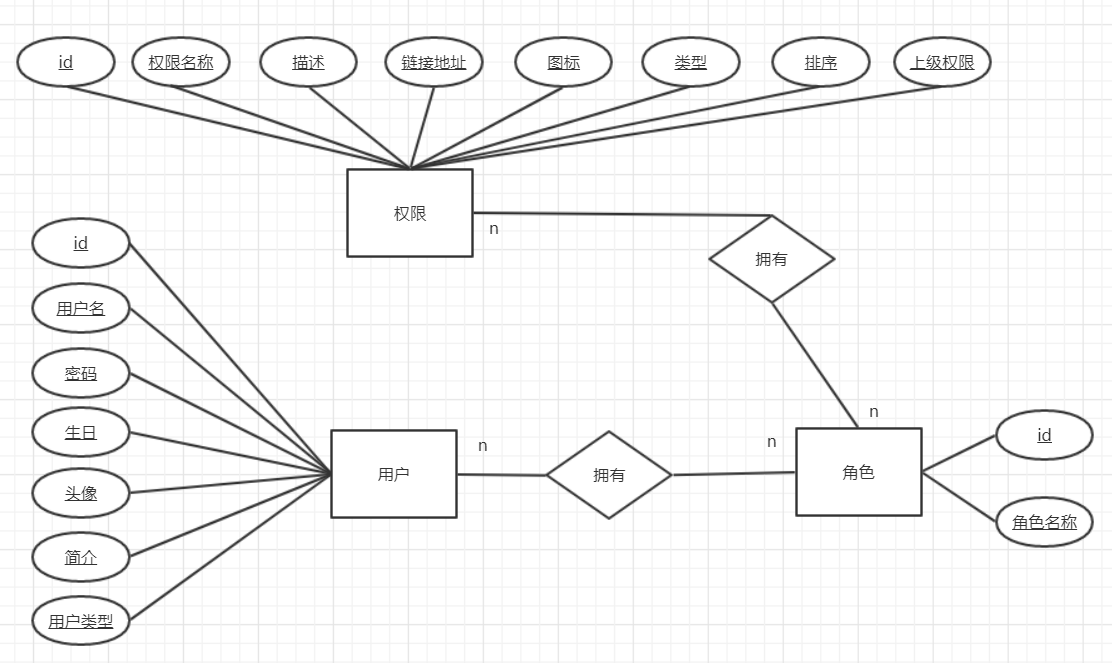
（4）用户登录：实现用户登录验证、动态菜单加载、登录拦截。

#### 2.3.2.2 模块设计：



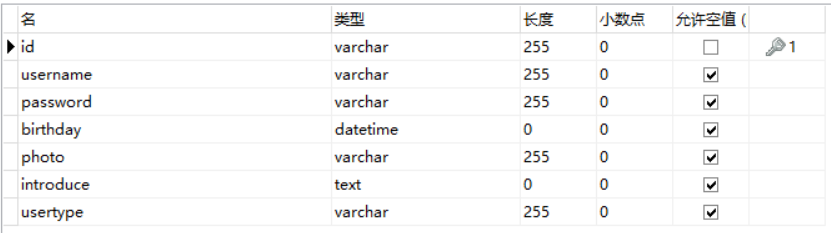
#### 2.3.2.3 数据库设计：

2.3.2.3.1 E-R图：



2.3.2.3.2表设计：

（1）用户表：



CREATE TABLE `t\_sys\_user` (

`id` varchar(255) COLLATE utf8\_bin NOT NULL,

`username` varchar(255) CHARACTER SET utf8 DEFAULT NULL COMMENT '用户账号',

`password` varchar(255) CHARACTER SET utf8 DEFAULT NULL COMMENT '用户密码',

`birthday` datetime DEFAULT NULL,

`photo` varchar(255) COLLATE utf8\_bin DEFAULT NULL,

`introduce` text COLLATE utf8\_bin,

`usertype` varchar(255) COLLATE utf8\_bin DEFAULT '普通用户' COMMENT '1.普通用户；2.超级管理员（不用授权）',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_bin COMMENT='用户表';

（2）权限表：



CREATE TABLE `t\_sys\_permission` (

`id` varchar(255) NOT NULL COMMENT 'id',

`name` varchar(255) DEFAULT NULL COMMENT '权限名称',

`descripion` varchar(255) DEFAULT NULL COMMENT '权限描述',

`url` varchar(255) DEFAULT NULL COMMENT '授权链接',

`pid` varchar(255) DEFAULT NULL COMMENT '父节点id',

`perms` varchar(255) DEFAULT NULL COMMENT '权限标识',

`type` int(11) DEFAULT '1' COMMENT '类型:1.菜单;2.按钮',

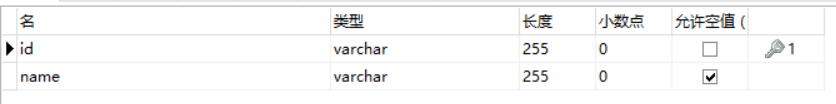
`icon` varchar(255) DEFAULT NULL COMMENT '菜单图标',

`order\_num` int(11) DEFAULT NULL COMMENT '排序',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='权限表';

（3）角色表：



CREATE TABLE `t\_sys\_role` (

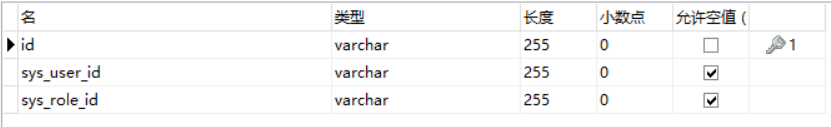
`id` varchar(255) NOT NULL COMMENT 'id',

`name` varchar(255) DEFAULT NULL COMMENT '角色名称',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='角色表';

（4）用户角色表：



CREATE TABLE `t\_sys\_role\_user` (

`id` varchar(255) NOT NULL,

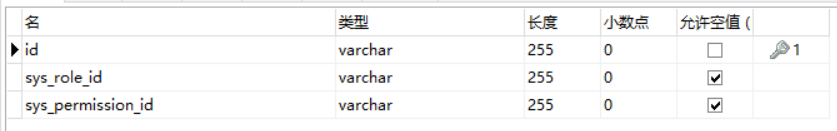
`sys\_user\_id` varchar(255) DEFAULT NULL COMMENT '用户id',

`sys\_role\_id` varchar(255) DEFAULT NULL COMMENT '角色id',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='用户角色中间表';

（5）角色权限表：



CREATE TABLE `t\_sys\_permission\_role` (

`id` varchar(255) NOT NULL,

`sys\_role\_id` varchar(255) DEFAULT NULL COMMENT '角色id',

`sys\_permission\_id` varchar(255) DEFAULT NULL COMMENT '权限id',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='角色权限中间表';

#### 2.3.2.4 接口设计

2.3.2.4.1 内部接口设计

**需求说明：**

小程序端通过访问java后台的api接口，实现向分析端传输分析图片，在经由后台转发分析段结果给小程序端。

**相关代码：**

@Controller

@RequestMapping("/ApiController")

public class ApiController {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

@Autowired

private PestService pestService;// 注入业务层的service

/\*\*

\* 接收前端传来的害虫的Json数据，并将其响应给分析端

\* @return

\*/

@RequestMapping("/pestInfor")

@ResponseBody

public Json pestInfor(@RequestBody String data) {

Pest pest=new Pest();

Json j = new Json();

JSONObject json;

//创建客户端

CloseableHttpClient httpClient = HttpClientBuilder.create().build();

//创建请求

HttpPost httpPost = new HttpPost("http://192.168.137.1:5000/get\_json");

data = "{\"base64\":\""+data+"\"}";

StringEntity entity = new StringEntity(data, "UTF-8");

httpPost.setEntity(entity);

httpPost.setHeader("Content-Type", "application/json;charset=utf8");

// 响应模型

CloseableHttpResponse response = null;

try {

//执行post请求

response = httpClient.execute(httpPost);

//获取响应实体

HttpEntity responseEntity = response.getEntity();

System.out.println("响应状态为:" + response.getStatusLine());

if (responseEntity != null) {

j.setSuccess(true);

System.out.println("响应内容长度为:" + responseEntity.getContentLength());

json=JSON.parseObject(EntityUtils.toString(responseEntity));

System.out.println("12345");

String name = json.getString("name");

String category = json.getString("category");

String introduce = json.getString("introduce");

String score = json.getString("score");

pest.setScore(score);

pest.setName(name);

pest.setCategory(category);

pest.setIntroduce(introduce);

j.setObj(pest);

}

} catch (ClientProtocolException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

try {

// 释放资源

if (httpClient != null) {

httpClient.close();

}

if (response != null) {

response.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

return j;

}

}

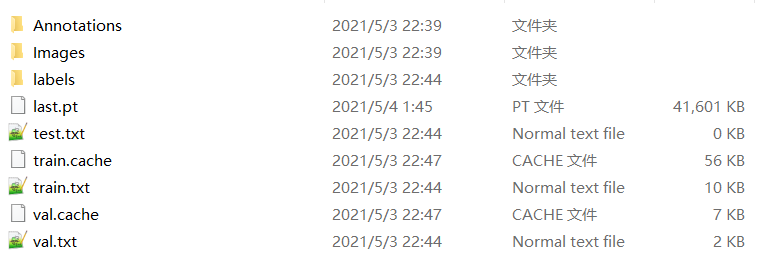
### 2.3.3 分析端架构：

#### 2.3.3.1 总体架构

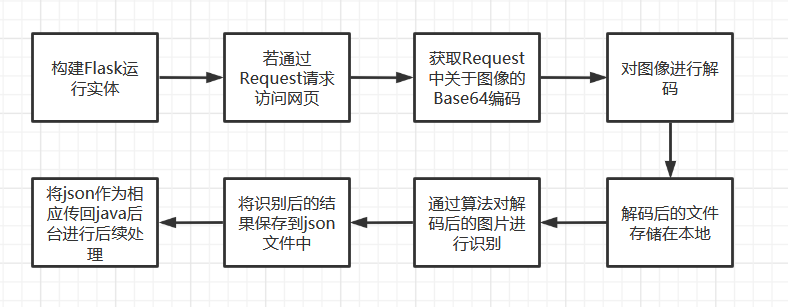
请求接收和发送基于Python的Flask框架，识别图像基于PyTorch框架下YoloV5的图像识别的算法

#### 2.3.3.2 设计框图

模型通过labelimg手动框出识别目标，并采用YOLOV5算法搭建，为了节省空间，搭建完的模型需要进行压缩，下图我们可以看到压缩后模型的大小与官方提供的模型大小几乎一致



#### 2.3.3.3 设计框图



#### 2.3.3.4 算法修改

通过增加和修改原算法中的代码，得到所需要的识别后的json文件

if save\_json:

file\_name = save\_path.split('\\')

content\_dic = {

"name": (names[int(cls)]),

"score": conf.tolist(),

"number": total[(names[int(cls)])]

}

content\_json.append(content\_dic)

#### 2.3.3.5 部分代码

# 启动flask程序

app = Flask(\_\_name\_\_)

@app.route('/get\_json', methods=['POST', 'GET'])

def get\_jsonData():

# 获取request的json数据

if request.json:

for key, value in request.json.items():

if key == "base64":

print("Detect vary base64, start debase.")

deBase64(value)

break

# 对解码后的图片进行检测

print("Start Detect...")

goDetect()

# 获取输出的json

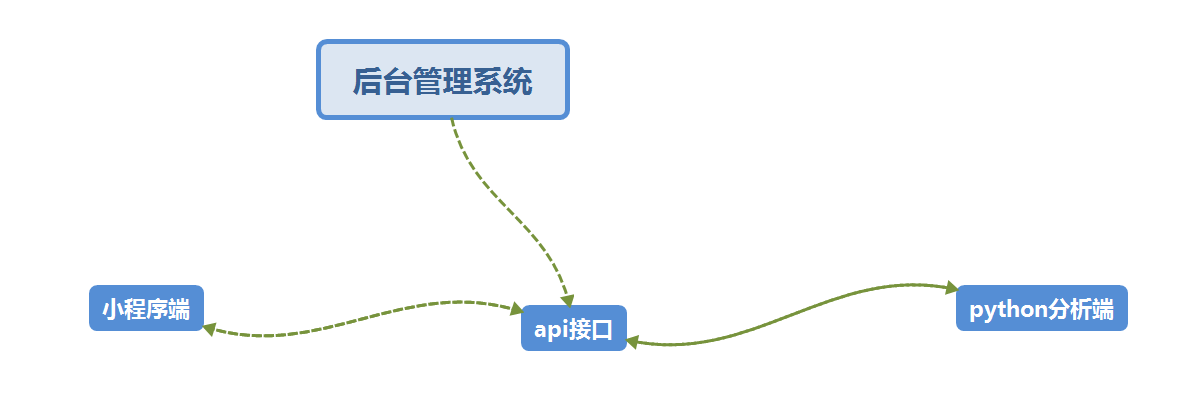
# data = getJsonFile()

# 将json数据传回网页

return toBase64()

### 2.3.4 运行设计

#### 2.3.4.1 运行模块组合



#### 2.3.4.2 运行时间

# 3. 尚未解决的问题

预添加功能：

（1）后期可根据识别搜索记录在后台管理系统中显示害虫搜索排行榜，以整理统计出当前的热门害虫。

（2）对接相关库、在搜索出相应害虫后为用户推荐适应农药并提供商城直达服务。