

摸鱼遥感平台

[概要设计文档]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 日期 | 版本号 | 修订说明 | 修订人 | 审核人 | 批准人 |
| 2022-07-04 | V1.0 | 创建 | 文皓 |  |  |
|  |  |  |  |  |  |

目录

[1. 概述 1](#_Toc107951523)

[2. 项目设计 1](#_Toc107951524)

[2.1. 总体架构设计 1](#_Toc107951525)

[2.2. 总体业务流程 2](#_Toc107951526)

[2.3. 功能模块图 3](#_Toc107951527)

[2.4. 设计视图 4](#_Toc107951528)

[2.5. 数据库设计图 5](#_Toc107951529)

[2.6. 接口设计 6](#_Toc107951530)

[2.7. 错误处理 6](#_Toc107951531)

# 概述

“摸鱼遥感平台”是一款面向遥感研究人员的Web 平台。 相比于使用人工和传统方法手动特征，本产品实现了一个基于互联网的，专门分析遥感数据的算法平台。其充分利用了最新的人工智能技术，同时给研究人员提供了便捷的项目管理，数据集管理等功能。

# 项目设计

## 总体架构设计

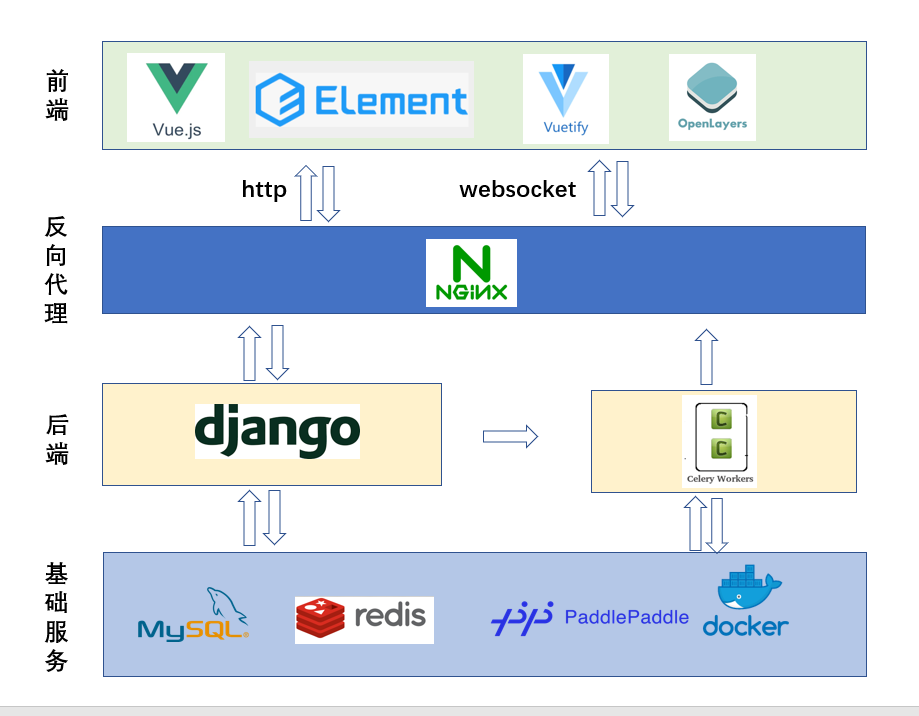


图 2‑1 总体架构图

项目整体采用前后端分离和服务器代理的模式进行架构设计与开发。其中，前端使用 vue.js 作为响应式界面的主页开发技术, 以@vue/cli（vue 脚手架）为架构辅助工具进行开发。使用 Axios 作为跨域接口访问的核心技术，以定制级 ElementUi 以及Vuetify 为UI组件, 图片显示部分使用Openlayers组件。后端使用 Django 架构进行基础的功能模块编写, 推理任务调度部分使用Celery分布式系统, 并以Redis作为消息队列 算法部分使用Paddle Inference进行推理部署，实现整体的人工智能模块。此外，基本信息数据库使用 Mysql，并根据功能进行分表。同时使用http和websocket作为前后端通信协议。最终后端以及nginx都运行在docker之中。

## 总体业务流程

用户可以通过网站的注册功能注册称为正式用户，注册页面需要用户填写用户名、密码和确认密码，只有用户名未被使用且两次输入密码一致时方可注册成功。注册成功后会跳转到登录页面。用户在登录页面需要输入正确的用户名和密码，方可跳转到网站首页。

在进入首页后，默认呈现的是“我的项目”页面，该页面罗列了用户创建的遥感项目。对于每一个项目，可点击“编辑”或“删除”按钮实现对应功能。同时，用户可点击页面左侧的“新建项目”按钮创建新的项目，网站提供四类功能：目标提取、变化检测、目标检测、地物分类。用户可以任选其中一种创建项目，并从网站图像数据中选取相应数量的图像，点击“提交任务”后服务器开始运行对应的处理程序，并返回结果，以对应形式显示在原图上。

在网站首页，用户可以通过点击导航栏中“我的数据”一栏查看和编辑图像数据。图片分为总共图片、公共图片、个人图片和分解结果四类。对于每一类中的图片，用户可以进行编辑名称、下载、删除、创建（查看）项目操作。点击左侧的“选取文件”按钮，用户可以选择上传本机的遥感图片。

点击首页导航栏中的“检索数据”一栏，可以在地图中框选图片，保存为自己的数据。本网站地图数据来源于高德地图，同时用户可以输入经纬度来快速定位到某一区域。点击鼠标两次确定框选矩形的两个顶点，为图片编辑名称后点击“上传”保存至网站服务器，可供新的项目使用。

## 功能模块图

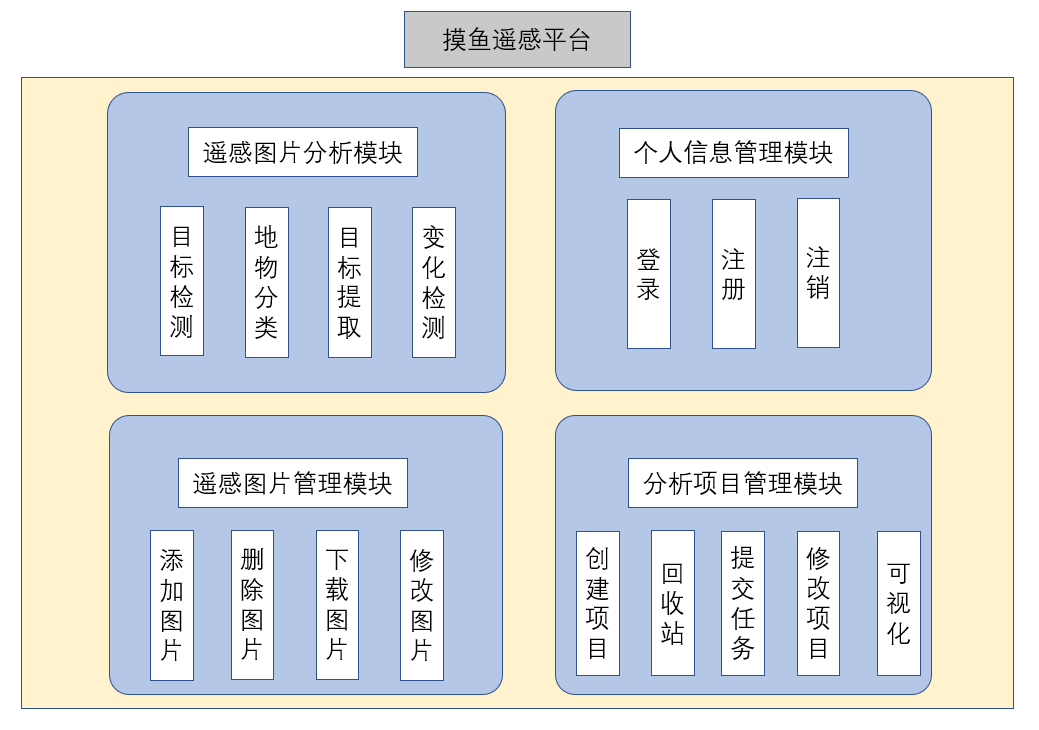


图 2‑2 功能模块图

应用整体可分为四大模块: 遥感图片分析模块, 遥感图片管理模块, 个人信息管理模块, 分析项目管理模块。 各个模块的详细功能见上图所示。其中具体功能的说明详见《用户使用说明书》

## 设计视图

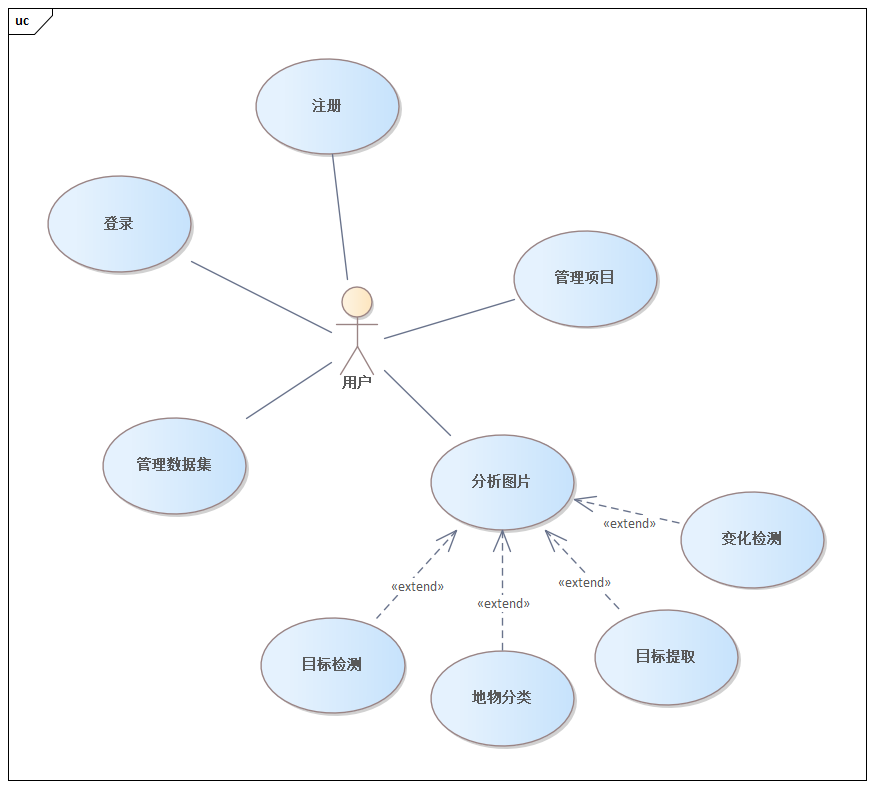


图 2‑3 用例图

用户作为 actor，具有登录、注册、管理数据集、管理项目、分析图片等用例。在分析图片之中，用户可以选择目标检测，地物分类，目标提取，变化检测四种不同的分析方式。

## 数据库设计图



图 2‑4 数据库概念设计图



图 2‑5 数据库逻辑设计图



图 2‑6 数据库物理模型图

数据库的设计图见上图所示，在实际之中我们则是使用了Django自带的ORM进行数据库设计的实现，其中的User类的实现由于继承了Django自带的部分属性。其具体实现有所不同。

## 接口设计

目前项目并未开放对外接口, 只有后端提供的大量的HTTP Restful 风格的

接口。其具体接口描述详见《接口描述文档》。

## 错误处理

为了便于用户使用和开发人员调试, 前后端的状态码传递基于HTTP的状态码含义, 每一个响应之中都将会有一个message字段对错误信息进行解释。主要使用的HTTP状态码如下:

|  |  |
| --- | --- |
| **状态码** | **含义** |
| 200 | 请求成功 |
| 400 | 无效的请求 |
| 401 | 尚未登录 |

表格 1 状态码表