基于百度飞桨的遥感图像智能解译平台

详细设计说明书



二〇二二年六月

目录

目录

[1 引言 7](#_Toc108009679)

[1.1 编写目的 7](#_Toc108009680)

[1.2 背景 7](#_Toc108009681)

[1.3 定义 7](#_Toc108009682)

[1.4 参考资料 8](#_Toc108009683)

[2 设计概述 8](#_Toc108009684)

[2.1 任务和目标 8](#_Toc108009685)

[2.2 需求概述 8](#_Toc108009686)

[2.3 运行环境 9](#_Toc108009687)

[2.3.1 硬件环境 9](#_Toc108009688)

[2.3.2 软件环境 10](#_Toc108009689)

[2.4 条件与限制 10](#_Toc108009690)

[2.5 项目设计整体设计介绍 10](#_Toc108009691)

[3 系统结构 10](#_Toc108009692)

[3.1 系统架构 10](#_Toc108009693)

[3.1.1 用户层： 11](#_Toc108009694)

[3.1.2 业务层： 11](#_Toc108009695)

[3.1.3 数据库层： 11](#_Toc108009696)

[3.2 系统功能模块 12](#_Toc108009697)

[4 模块详细设计 12](#_Toc108009698)

[4.1 用户注册模块 12](#_Toc108009699)

[4.1.1 功能 12](#_Toc108009700)

[4.1.2 性能 12](#_Toc108009701)

[4.1.3 输入项 12](#_Toc108009702)

[4.1.4 输出项 12](#_Toc108009703)

[4.1.5 设计方法 12](#_Toc108009704)

[4.1.6 流程逻辑 13](#_Toc108009705)

[4.1.7 接口 13](#_Toc108009706)

[4.1.8 限制条件 13](#_Toc108009707)

[4.1.9 测试计划 13](#_Toc108009708)

[4.2 用户登录模块 13](#_Toc108009709)

[4.2.1 功能 14](#_Toc108009710)

[4.2.2 性能 14](#_Toc108009711)

[4.2.3 输入项 14](#_Toc108009712)

[4.2.4 输出项 14](#_Toc108009713)

[4.2.5 设计方法 14](#_Toc108009714)

[4.2.6 流程逻辑 14](#_Toc108009715)

[4.2.7 接口 15](#_Toc108009716)

[4.2.8 限制条件 15](#_Toc108009717)

[4.2.9 测试计划 15](#_Toc108009718)

[4.3 上传图片模块 15](#_Toc108009719)

[4.3.1 功能 15](#_Toc108009720)

[4.3.2 性能 15](#_Toc108009721)

[4.3.3 输入项 15](#_Toc108009722)

[4.3.4 输出项 15](#_Toc108009723)

[4.3.5 设计方法 16](#_Toc108009724)

[4.3.6 流程逻辑 16](#_Toc108009725)

[4.3.7 接口 16](#_Toc108009726)

[4.3.8 限制条件 17](#_Toc108009727)

[4.3.9 测试计划 17](#_Toc108009728)

[4.4 下载图片模块 17](#_Toc108009729)

[4.4.1 功能 17](#_Toc108009730)

[4.4.2 性能 17](#_Toc108009731)

[4.4.3 输入项 17](#_Toc108009732)

[4.4.4 输出项 17](#_Toc108009733)

[4.4.5 设计方法 17](#_Toc108009734)

[4.4.6 流程逻辑 18](#_Toc108009735)

[4.4.7 接口 18](#_Toc108009736)

[4.4.8 限制条件 18](#_Toc108009737)

[4.4.9 测试计划 18](#_Toc108009738)

[4.5 目标检测模块 18](#_Toc108009739)

[4.5.1 功能 18](#_Toc108009740)

[4.5.2 性能 18](#_Toc108009741)

[4.5.3 输入项 19](#_Toc108009742)

[4.5.4 输出项 19](#_Toc108009743)

[4.5.5 设计方法 19](#_Toc108009744)

[4.5.6 流程逻辑 19](#_Toc108009745)

[4.5.7 接口 19](#_Toc108009746)

[4.5.8 限制条件 19](#_Toc108009747)

[4.5.9 测试计划 19](#_Toc108009748)

[4.6 目标提取模块 20](#_Toc108009749)

[4.6.1 功能 20](#_Toc108009750)

[4.6.2 性能 20](#_Toc108009751)

[4.6.3 输入项 20](#_Toc108009752)

[4.6.4 输出项 20](#_Toc108009753)

[4.6.5 设计方法 20](#_Toc108009754)

[4.6.6 流程逻辑 20](#_Toc108009755)

[4.6.7 接口 21](#_Toc108009756)

[4.6.8 限制条件 21](#_Toc108009757)

[4.6.9 测试计划 21](#_Toc108009758)

[4.7 变化检测模块 21](#_Toc108009759)

[4.7.1 功能 21](#_Toc108009760)

[4.7.2 性能 22](#_Toc108009761)

[4.7.3 输入项 22](#_Toc108009762)

[4.7.4 输出项 22](#_Toc108009763)

[4.7.5设计方法 22](#_Toc108009764)

[4.7.6 流程逻辑 22](#_Toc108009765)

[4.7.7 接口 23](#_Toc108009766)

[4.7.8 限制条件 23](#_Toc108009767)

[4.7.9 测试计划 23](#_Toc108009768)

[4.8 地物分类模块 23](#_Toc108009769)

[4.8.1 功能 23](#_Toc108009770)

[4.8.2 性能 23](#_Toc108009771)

[4.8.3 输入项 23](#_Toc108009772)

[4.8.4 输出项 23](#_Toc108009773)

[4.8.5 设计方法 23](#_Toc108009774)

[4.8.6 流程逻辑 23](#_Toc108009775)

[4.8.7 接口 24](#_Toc108009776)

[4.8.8 限制条件 24](#_Toc108009777)

[4.8.9 测试计划 24](#_Toc108009778)

[4.9 遥感图像分析结果展示模块 24](#_Toc108009779)

[4.9.1 功能 24](#_Toc108009780)

[4.9.2 性能 25](#_Toc108009781)

[4.9.3 输入项 25](#_Toc108009782)

[4.9.4 输出项 25](#_Toc108009783)

[4.9.5 设计方法 25](#_Toc108009784)

[4.9.6 接口 25](#_Toc108009785)

[4.9.7 限制条件 25](#_Toc108009786)

[4.9.8 测试计划 26](#_Toc108009787)

[5 前端详细设计 26](#_Toc108009788)

[5.1 前端样式详细设计 29](#_Toc108009789)

[5.1.1 登录/注册/密码找回： 29](#_Toc108009790)

[5.2 前端引导详细设计 36](#_Toc108009791)

[5.3 前端反馈详细设计 37](#_Toc108009792)

[5.4 前端数据存储调度详细设计 38](#_Toc108009793)

[5.5 前端拦截器详细设计 38](#_Toc108009794)

[5.6 前端路由详细设计 39](#_Toc108009795)

[5.7 前端代理详细设计 39](#_Toc108009796)

[6 后端详细设计 40](#_Toc108009797)

[6.1 后端接口详细设计 43](#_Toc108009798)

[6.1.1 数据库访问接口-serverAPI.py 43](#_Toc108009799)

[6.1.2 变化检测访问接口-serverCD.py 44](#_Toc108009800)

[6.1.3 地物分类访问接口-serverGOD.py 46](#_Toc108009801)

[6.1.4 目标检测访问接口-serverOD.py 47](#_Toc108009802)

[6.1.5 目标提取访问接口-serverTE.py 48](#_Toc108009803)

[6.2 数据库详细设计 49](#_Toc108009804)

[6.2.1 数据库E-R实体图 49](#_Toc108009805)

[6.2.2 数据库创建代码 49](#_Toc108009806)

[6.2.3 数据库连接代码截图 50](#_Toc108009807)

# 1 引言

## 1.1 编写目的

此详细说明书对遥感图像智能解译平台进行了实现层面上的要求与说明，对系统设计大的算法的原理与具体实现进行了详细说明，为进行之后的实现、优化与测试提供参考。

该详细设计说明书的预期读者为本项目开发成员以及对该项目感兴趣，拓展和维护该项目的相关人员。

## 1.2 背景

遥感数据具有覆盖范围及时域广、获取迅速、动态信息强等特点，已广泛应用于城市规划、气象预测、环境保护、防灾减灾、农林业监测等领域并取得了良好的经济和社会效益。

近年来，随着遥感技术的进一步发展和新一代高分辨率卫星系统相继投入应用，我国遥感领域已步入了高分辨率影像的快车道，对遥感数据的分析应用服务需求与日俱增。传统方式对高分辨率卫星遥感图像的特征刻画能力差且人工成本高。随着人工智能及深度学习技术快速发展，通过应用深度学习技术可以加速遥感领域智能化应用，促进遥感数据处理走向智能化，使我国遥感事业更好地服务国计民生。该软件主要用于处理遥感图片。

## 1.3 定义

* + 1. **飞桨(PaddlePaddle)**：飞桨(PaddlePaddle)以百度多年的深度学习技术研究和业务应用为基础，是中国首个开源开放、技术领先、功能完备的产业级深度学习平台，集深度学习核心训练和推理框架、基础模型库、端到端开发套件和丰富的工具组件于一体。拥有较为活跃的开发者社区生态。提供大量官方支持模型集合，并推出全类型的高性能部署和集成方案供开发者使用。能帮助开发者快速实现 AI 想法，快速上线 AI 业务。帮助越来越多的行业完成 AI 赋能，实现产业智能化升级。
    2. **卷积神经网络(CNN)**：卷积神经网络（Convolutional Neural Networks, CNN）是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习的代表

算法之一。卷积神经网络仿造生物的视知觉机制构建，可以进行监督学习和非监督学习，其隐含层内的卷积核参数共享和层间连接的稀疏性使得卷积神经网络能够以较小的计算量对格点化特征，因此在图像特征识别领域应用较为广泛。

## 1.4 参考资料

[1] 需求规格说明书-基于百度飞桨的遥感图像智能解译平台.

[2] 可行性分析报告-基于百度飞桨的遥感图像智能解译平台.

[3] 项目概要设计-基于百度飞桨的遥感图像智能解译平台.

[4] [什么是遥感影像变化检测](https://baike.baidu.com/item/%E5%8F%98%E5%8C%96%E6%A3%80%E6%B5%8B/8636264)-百度百科

[5]Y. Long, Y. Gong, Z. Xiao and Q. Liu, "Accurate Object Localization in Remote Sensing Images Based on Convolutional Neural Networks," in IEEE Transactions on Geoscience and Remote Sensing, vol. 55, no. 5, pp. 2486-2498, May 2017. doi: 10.1109/TGRS.2016.2645610

[6]Z Xiao, Q Liu, G Tang, X Zhai, "Elliptic Fourier transformation-based histograms of oriented gradients for rotationally invariant object detection in remote-sensing images", International Journal of Remote Sensing, vol. 36, no. 2, 2015

[7] Mnih, V. (2013). Machine Learning for Aerial Image Labeling [Phdthesis]. University of Toronto.

# 2 设计概述

## 2.1 任务和目标

该系统的详细设计目标是使软件的整个设计与实现过程必须在保证软件质量的前提下通过生产能力的提高、员工工作效率的提高等等使软件开发成本最小化。

本项目系统目标是设计出符合中软杯比赛要求的程序，主要包括四个功能，目标提取，变化检测，目标检测，地物分类；并将软件与运行在WEB平台上。还需注重用户界面的视觉、交互及分析结果的创新展示。

## 2.2 需求概述

本系统是一个遥感图像智能解译平台，平台运行在Web服务器上，可通过网络与该平台进行连接，用户异地也可使用该平台。通过图像分割技术对卫星图像中指定对象完成分割，将分割结果提取出来，展示到页面上供用户查看；通过图像分割技术对同区域的卫星图像变化情况完成检测，标记到对应的图像上；通过目标检测技术对卫星图像中指定对象完成检测，检测结果同样显示到图像上，供检测人员查看；利用图像分割技术对卫星图像每个像素完成分类，完成地物分类

以下来自需求分析说明书：

表1 需求对应表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 需求编号 | 需求 | 章节号 | 备注 |
| 1 | PaddleRS-0010 | 注册 | 3.1注册/登录 |  |
| 2 | PaddleRS-0020 | 登录 | 3.1注册/登录 |  |
| 3 | PaddleRS-0030 | 导航界面 | 3.2导航界面 |  |
| 4 | PaddleRS-0040 | 图像上传 | 3.3 图像上传/下载 |  |
| 5 | PaddleRS-0050 | 图像下载 | 3.3 图像上传/下载 |  |
| 6 | PaddleRS-0060 | 目标提取 | 3.4 目标提取 |  |
| 7 | PaddleRS-0070 | 变化检测 | 3.5 变化检测 |  |
| 8 | PaddleRS-0080 | 目标检测 | 3.6 目标检测 |  |
| 9 | PaddleRS-0090 | 地物分类 | 3.7 地物分类 |  |
| 10 | PaddleRS-0100 | 系统性能指标 | 4.系统性能指标 |  |
| 11 | PaddleRS-0110 | 系统接口需求 | 5系统接口需求 |  |

## 2.3 运行环境

### 2.3.1 硬件环境

环境设备：

Web服务器：内存大于等于8GB

数据库服务器：mysql5.7, 8G RAM, 1000GB磁盘系统

深度学习服务器：GPU: Tesla V100. Video Mem: 32GB；

CPU: 4 Cores. RAM: 32GB. Disk: 100GB

使用设备：

手机、笔记本电脑/台式电脑/平板电脑，并能连接 WLAN 或者蜂窝网络；

其他设备：鼠标、键盘。

### 2.3.2 软件环境

运行环境：Linux 操作系统、Windows 操作系统；

使用环境：Windows 操作系统、Android 操作系统、IOS 操作系统、Chrome。

## 2.4 条件与限制

开发该系统的主要任务由小组成员共同承担，无主要资金来源，故在设计开发中不能使用过多的资金成本投入，但由于飞桨平台能够免费提供 GPU 算力，所以项目在设计开发方面有足够的算力支持。在交付使用后，应保证软件的运行寿命在 1 年以上；系统运行顺畅无卡顿，无闪退等严重BUG；UI界面美观、逻辑合理、交互友好、可视化美观。进行系统方案选择比较的期限为 1 周，且系统开发时间应基本控制在 2 个月内。

## 2.5 项目设计整体设计介绍

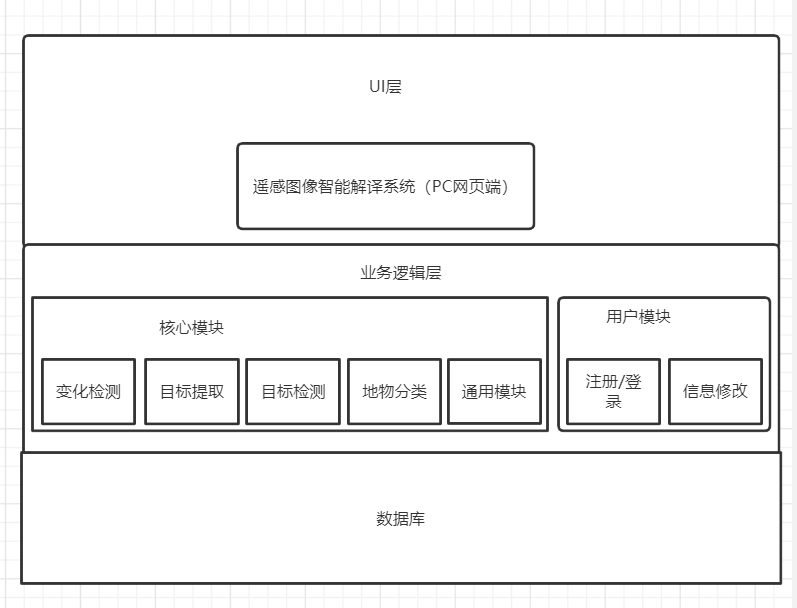
项目采用客户端/服务器模式（即：C/S架构），分前后端设计和开发，采用前后端分离技术，及提前定义好接口和数据传输及解析标准，前后端交互由外部接口实现，内部接口用于内部数据的处理和调用。

前端采用Vue开发，调用了Element-UI组件进行样式的美化，以便给用户更好的体验，同时使用了Cookie存储用户的数据，用户可以免密登录。在用户层面上，还考虑到了如何便捷和引导用户的操作，设计编写了大量的判断和引导语句，提示用户如何进行各项的操作，对用户的错误输入也能给予反馈，页面加载动画等细节部分都考虑到位，只为给用户的提供更加舒适的体验。为了平台的安全，前端还采用了拦截器技术，对非法访问进行拦截。

后端采用Python的Flask库搭建服务器框架，能对前端的请求数据进行解析和响应，采用Mysql数据库存储用户的数据，并有内部模型调用接口，用于内部程序处理用户的遥感数据，对外提供模型的单个处理接口和批量处理接口，以便高效满足用户的需求，且模型结果能返回到前端用于展示，保证用户能对结果有更加直观的了解。

# 3 系统结构

## 3.1 系统架构



**图3-1 系统架构图**

根据项目需求，本系统面向PC端用户采取C/S(客户端/服务端)架构设计，软件体系结构为分层结构，一共分为三层，用户界面层，业务逻辑层，数据库。每个层次功能与作用介绍如下：

### 3.1.1 用户层：

该板块为用户与系统进行人机交互的PC端Web界面，使用Vue实现。Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。（官方介绍）

Vue 只关注视图层， 采用自底向上增量开发的设计。

Vue 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

### 3.1.2 业务层：

可以从逻辑上分为三个个板块：

1、遥感图像分析模块：

（1）目标提取（使用图像分割技术对卫星图像中指定对象完成分割）

（2）变化检测（使用图像分割技术对同区域两个时期的卫星图像变化情况完成分析）

（3）目标检测（使用目标检测技术对卫星图像中指定对象完成检测）

（4）地物分类（使用图像分割技术对卫星图像每个像素完成分类）

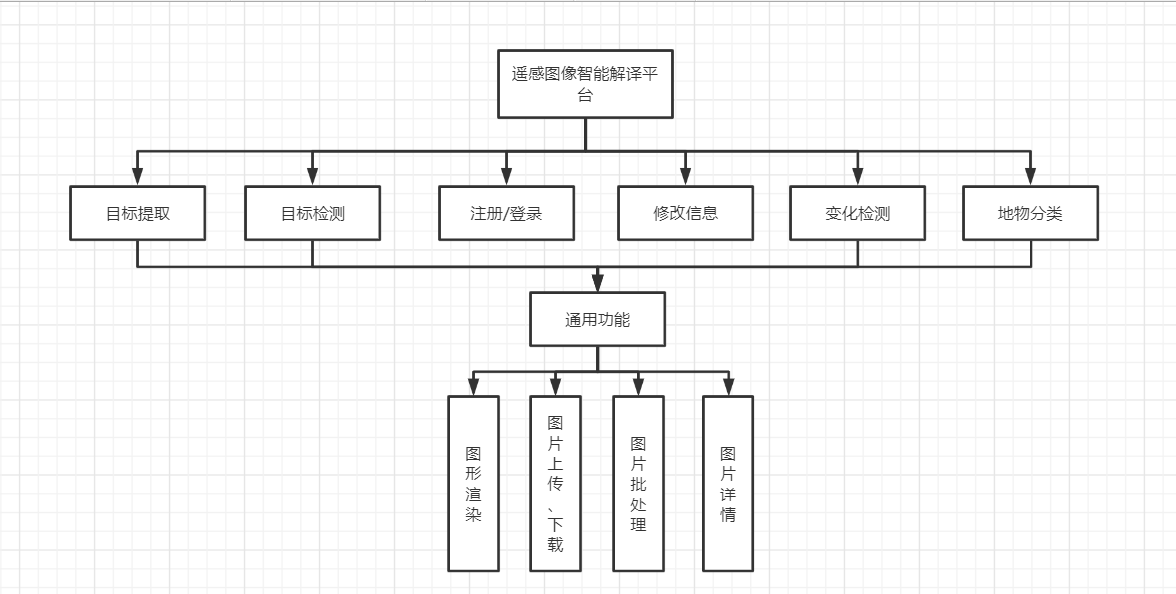
2、用户管理模块：用户登录和用户注册

3、图片上传/下载模块

### 3.1.3 数据库层：

该层使用MySql数据库产品进行构建，同时增加一个备份子库。因为业务不是很复杂无需添加redis等充当缓存。

## 3.2 系统功能模块

****

**图3-2 遥感图像智能解译平台功能模块图**

# 4 模块详细设计

## 4.1 用户注册模块

### 4.1.1 功能

平台系统需要用户登录后才能使用，注册模块能帮助没有账号的用户，快速注册一个登录账号。

### 4.1.2 性能

该模块属于建立在 Web 客户端的实现用户交互的功能性模块，对响应速度的要求较高。

当用户填写完成注册信息，并提交注册申请后，系统需要在3秒内将注册成功或失败信息告知用户。

### 4.1.3 输入项

输入项为用户的注册信息，如电话号码、邮箱、用户名和密码等个人验证信息。

### 4.1.4 输出项

输出项为注册成功或失败提示（失败提示需包含失败原因信息）。

### 4.1.5 设计方法

1. 在 Web 页面中构建了一个 HTML Format 组件，然后在JavaScript代码中将该表单中的用户注册信息与“注册”按钮的点击响应事件进行绑定。
2. 数据成功发送至后台后，需将数据存入对应的数据库中，以保证用户后续能登录验证通过。
3. 数据成功保存至数据库后，返回“成功注册”提示。反之，提示什么原因导致的注册失败，方便用户进行更正后重新提交。
4. 用户的表单数据以POST请求方式提交到后台。

### 4.1.6 流程逻辑



开始

结束

后端返回注册成功或失败信息

前端显示注册成功或失败信息

后台保存注册信息

点击注册按钮发起注册请求

填写注册信息

**图 4-1 用户注册模块流程图**

用户在填写完注册信息，点击提交按钮后，注册信息将以POST方式提交到后台，后台将尝试保存注册信息，并返回注册成功或失败信息到前端，前端将这些信息展示告知给用户。

### 4.1.7 接口

用户注册接口，后端接收前端发起的POST请求，请求数据为对应的注册表单数据。接口返回注册成功或失败数据。

### 4.1.8 限制条件

用户成功填表单。

### 4.1.9 测试计划

编写测试用例，之后在注册页面填写对应的注册表单数据测试用例，然后点击“注册”，查看前端相应结果以及后台数据库是否正确保存了表单中的所有注册信息。

接口测试：使用 POSTMAN 工具，采用POST请求方式，构建好对应的表单数据后，点击“Send”发起请求，对注册接口的测试。

## 4.2 用户登录模块

### 4.2.1 功能

用于验证用户身份，保证用户账户等数据的安全。

### 4.2.2 性能

该模块属于建立在 Web 客户端的实现用户交互的功能性模块，对响应速度的要求较高。

用户填写完登录信息并点击“登录”按钮后，需在3秒内对给于用户反馈响应。

### 4.2.3 输入项

输入项为用户的账号和密码。

### 4.2.4 输出项

输出项为登录验证成功或失败信息，成功则跳转到系统主页面，失败则提示用户失败原因。

### 4.2.5 设计方法

1. 在 Web 页面中构建了一个 HTML Format 组件，然后在JavaScript代码中将该表单中的用户登录验证信息与“登录”按钮的点击响应事件进行绑定。
2. 数据发送至后台进行登录信息的验证，验证成功或失败信息将返回至前端页面。
3. 用户的登录表单数据以POST请求方式提交到后台。

### 4.2.6 流程逻辑



开始

结束

后端返回验证成功或失败信息

前端对成功或失败响应作出对应的反馈

对数据进行登录验证

点击登录按钮

填写登录验证

**图 4-2 用户登录模块流程图**

用户首先填写登录验证信息，点击“登录”按钮后，验证信息将上传至后台进行验证，后台会返回验证结果数据，前端会对验证结果数据作出相应的反馈。

### 4.2.7 接口

用户登录接口，后端对前端的POST登录请求，给于相应的响应。

### 4.2.8 限制条件

用户已填写完登录验证表单，前后端正常运行中。

### 4.2.9 测试计划

对于该模块编写登录测试用例，对后台数据库中已有和没有的账户进行登录验证操作，查看登录响应是否符合预期结果。

接口测试：使用 POSTMAN 工具对设置输出结果置信度的接口进行接口测试。

## 4.3 上传图片模块

上传图片模块用于实现用户选择待处理的本地图片，然后将其上传至服务端的功能。通过此模块，系统可以获取到要进行分析的遥感图像数据。该模块使用Ajax实现浏览器与 Web 服务器之间数据的异步交互。

### 4.3.1 功能

上传图像模块实现用户从本地选择待处理图片（图片格式为PNG类型），然后前端页面读取图片数据，最后通过 jQuery 框架中的 ajax()函数建立的 POST Request 将其上传至服务端。

### 4.3.2 性能

该模块属于建立在 Web 客户端的实现用户交互的功能性模块，对响应速度的要求较高。

点击“上传图片”按钮后从本地选择图片的响应速度取决于用户电脑的硬件性能；图片发送至服务端的速度取决于用户的网络性能。

### 4.3.3 输入项

输入项为从客户端读取的本地图片流数据。

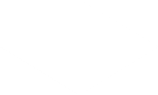
### 4.3.4 输出项

输出项为发送到服务端后，经过处理后得到的图片数据。

### 4.3.5 设计方法

1. 首先在 Web 页面中构建了一个 HTML Button 组件，创建对该Button 按钮的点击响应事件
2. 通过该按钮绑定的 HTML File 标签获得图片文件的存储路径。
3. 然后利用 jQuery 框架中的文件读取函数对用户选择的文件进行数据读取和格式检查（主要是文件大小的检查：小于 100MB；文件类型的检查：是否是对应格式的遥感图片文件）。
4. 对于通过格式检查的遥感图片流数据，通过 jQuery 中封装完善的 ajax()函数将图片流数据通过 POST Request 命令发送至指定的 URL 地址。
5. 服务端接收此 POST Request 命令并最终接受到其中包含的图片流数据。

### 4.3.6 流程逻辑



N

开始

图片格式是否

正确

前端根据获取的文件路径读取数据

选择待处理的图片文件

用户点击“上传图片”按钮

Y



结束

前端发送数据

服务端接收、处理并返回数据

**图 4-3 上传图像模块流程图**

用户点击“上传图片”按钮，在弹出的文件选择页面中选择待处理的图片文件。前端根据获取到的文件路径读取文件数据，然后系统对该文件进行数据的格式检查：若通过检查，则前端发送数据，由服务端接收、处理并返回数据；若未通过，用户需要重新选择相应的图片文件。

### 4.3.7 接口

前后端图片数据交互接口，通过 Web 客户端获取图片数据，作为后续服务端图片分析操作的输入项。

### 4.3.8 限制条件

1. 限制上传的图片文件必须为PNG或JPG(JPEG)格式；
2. 限制图片文件大小不大于 10MB。

### 4.3.9 测试计划

对于该模块的测试预期对不同文件类型、不同文件大小建立对应的有效等价类和无效等价类的测试用例，用于进行上传图片模块的“通过测试”和“失败测试”。

接口测试：使用 POSTMAN 工具对上传图片的接口进行接口测试。

## 4.4 下载图片模块

### 4.4.1 功能

用户可以将服务器端处理后的图片数据保存到本地，方便用户对结果数据进行管理和存储。

### 4.4.2 性能

该模块属于建立在 Web 客户端的实现用户交互的功能性模块，对响应速度的要求较高。

点击“下载图片”按钮后从本地下载图片的响应速度取决于用户电脑的硬件性能；图片下载完毕的速度取决于用户的网络性能。

### 4.4.3 输入项

输入项为用户的点击下载事件。

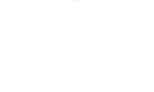
### 4.4.4 输出项

输出项为服务器传送到本地的图片流数据。

### 4.4.5 设计方法

1. 首先在 Web 页面中构建了一个 HTML Button 组件，并创建对该Button 按钮的点击响应事件。
2. 通过该按钮绑定的 HTML File 标签获得图片文件的在服务器的存储路径。
3. 在用户选择完图像保存路径并点击确认下载后，传输对应的图像数据流响应数据到客户端。

### 4.4.6 流程逻辑



N

开始

路径是否合理

用户选择图像保存路径

用户点击下载按钮

已有处理好的图像结果数据

Y



结束

服务端发送数据

客户端接收数据

**图 4-4 下载图像模块流程图**

用户在图像结果数据以出来的基础上，点击“下载按钮”，选择保存路径，路径合理则服务器返回请求图像数据，客户端将会接收这些图像数据并将数据保存到指定的文件夹中，若路径不合理（如：读写权限不够）则提示用户重新现在保存路径。

### 4.4.7 接口

前后端图片数据交互接口，通过 Web 客户端发起下载请求，作为后续服务端图片下载操作的输入项。

### 4.4.8 限制条件

1. 图像数据保存地址要合理。
2. 网络连接要通畅。

### 4.4.9 测试计划

对于该模块的测试预期通过点击下载按钮，下载对应的图片并查看下载结果是否正确，已进行“失败测试”和“通过测试”。

接口测试：使用 POSTMAN 工具对图片下载接口进行接口测试。

## 4.5 目标检测模块

### 4.5.1 功能

使用目标检测技术对卫星图像中指定对象完成检测。

### 4.5.2 性能

处理速度：每幅图像15秒以内；

模型精度：0.80以上。

### 4.5.3 输入项

用户上传的遥感图像文件（或压缩包数据）。

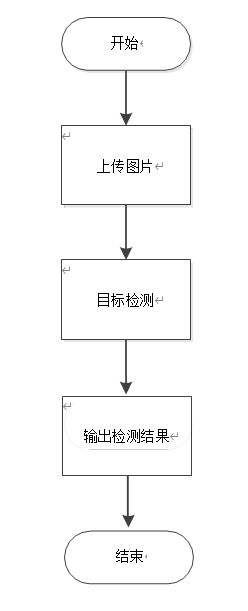
### 4.5.4 输出项

目标检测后得到的包含检测数据的图像。

### 4.5.5 设计方法

1. 后台调用目标检测接口，对待处理的遥感图像进行目标检测。
2. 检测完毕后，将检测结果返回到前端进行显示。

### 4.5.6 流程逻辑



**图 4-5 目标检测模块流程图**

### 4.5.7 接口

目标检测接口：接口接收传入的遥感图像数据，返回检测后的结果图像数据。

### 4.5.8 限制条件

遥感图像为RGB三通道的png格式，尺寸为1024×1024像素

### 4.5.9 测试计划

首先测试训练得到的模型是否符合条件，是否能正常进行目标检测，其次检测接口是否能正常被调用，返回的图像数据是否能被前端接收并展示。

## 4.6 目标提取模块

### 4.6.1 功能

使用图像分割技术对卫星图像中指定对象完成分割，并输出提取之后的图片。

### 4.6.2 性能

处理速度：每幅图像15秒以内；

模型精度：0.80以上。

### 4.6.3 输入项

本地的遥感图片，图片格式为PNG格式，图片大小为1500×1500。

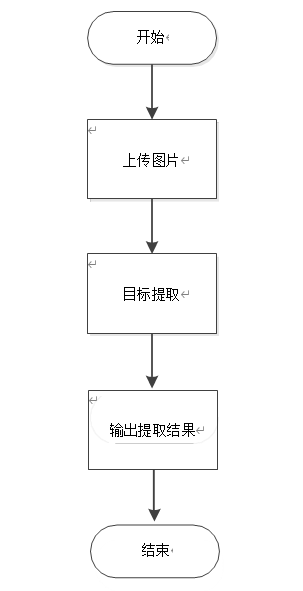
### 4.6.4 输出项

分割之后的图片，图片格式为PNG格式，大小以分割后的实际情况为准。

### 设计方法

1. 借助官方提供的模型，加载部署模型。
2. 对上传的图像进行预处理，然后使用模型预测结果，并进行对比，计算指标，达到预设要求返回分析结果

### 4.6.6 流程逻辑



**图 4-6 目标提取模块流程图**

### 4.6.7 接口

目标提取接口：接收遥感图像数据，返回目标提取后的结果图片。

### 4.6.8 限制条件

输入数据：遥感图片格式为PNG格式，图片大小为1044×915。

### 4.6.9 测试计划

测试目标提取模块功能是否能正常提取出对应的目标数据，之后进行接口测试，检验接口是否能正常调用和返回结果数据，返回时间是否在预期范围内，图像结果数据是否能在前端正常展示。

## 4.7 变化检测模块

### 4.7.1 功能

使用图像分割技术对同区域两个时期的卫星图像变化情况完成分析。返回变化检测后的8bit结果图像。

### 4.7.2 性能

处理速度：每幅图像处理结果在5秒以内。

### 4.7.3 输入项

本地的遥感图片，且为一对前后时相遥感图像，要求模型能够定位出其中建筑变化的区域。输入图像为RGB三通道的png格式，尺寸为1024×1024像素。

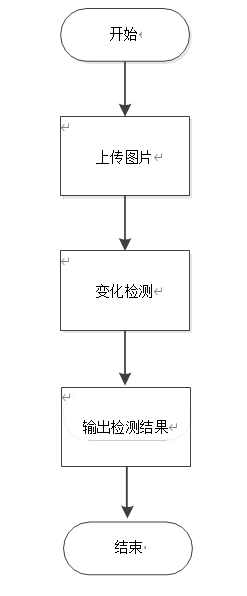
### 4.7.4 输出项

输出图像为单通道png格式的灰度图（8bit），其中255代表变化的建筑区域，0代表未变化建筑区域。输出图像尺寸与输入图相同

### 4.7.5设计方法

1. 为了处理接收外部输入、返回预测结果，并做一些业务逻辑相关的处理，需要引入一个处理中心，这个处理中心通常是web框架 如 flask、tornado、Django等，其作用是搭建http服务，将外部输入传给模型，将模型预测结果返回。
2. 通过web框架如flask等搭建好对外服务，初始化时加载好模型。当外部应用服务搜索应用的输入时，如上传了一张图片，对外服务就将输入传给预处理函数，将输入处理成可以给模型预测的样子，例如做标准化、去噪等等。
3. 随后模型进行预测，并将结果返回给变化检测界面。

### 4.7.6 流程逻辑



**图 4-7 变化检测模块流程图**

### 4.7.7 接口

变化检测接口：向接口传入符合条件的两幅遥感图像数据，接口返回变化检测后的结果数据。

### 4.7.8 限制条件

输入图像为RGB三通道的png格式，尺寸为1024×1024像素

### 4.7.9 测试计划

变化检测算法模型的准确率要达到0.85以上，接口测试时，数据能通过接口正常传输到后台，并能成功返回变化检测后的图像结果数据到前端进行展示。

## 4.8 地物分类模块

### 4.8.1 功能

使用图像分割技术对卫星图像每个像素完成分类。

### 4.8.2 性能

处理速度：每幅图像处理事件在5秒以内。

### 4.8.3 输入项

本地的遥感图片，图像为RGB三通道的png格式，尺寸为1024×1024像素

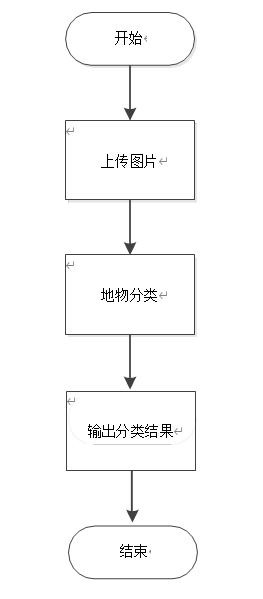
### 4.8.4 输出项

分类之后的图片，图片上有分类信息，图像为RGB三通道的png格式，尺寸为1024×1024像素。

### 4.8.5 设计方法

1. 借助官方提供的模型，加载部署模型。
2. 模型对上传的图像进行预处理，预测结果，并进行对比，计算指标，达到预设要求返回分析结果。

### 4.8.6 流程逻辑



**图 4-8 地物分类模块流程图**

### 4.8.7 接口

地物分类接口：对出入的遥感图像数据进行地物分类，分类结果返回至前端进行展示；

### 4.8.8 限制条件

图像为RGB三通道的png格式，尺寸为1024×1024像素。

### 4.8.9 测试计划

测试地物分类模型是否能正常运行，接口能否被正常调用，返回结果是否符合要求，前端能否正常接收并显示地物分类后的结果图像数据。

## 4.9 遥感图像分析结果展示模块

### 4.9.1 功能

该模块主要实现从服务端获取到系统分析处理好的遥感图像数据，然后在 Web客户端进行展示的功能。遥感图像分析结果显示模块是在 Web 客户端上构建的另一 UI界面、功能界面，该模块使得用户能够在 Web页面中直接观察遥感图像数据的处理结果。

处理结果将以图片信息、文字信息等多种类型进行显示。

### 4.9.2 性能

该模块属于建立在 Web 客户端的实现用户交互的功能性模块，对响应速度的要求较高。

首先是获取分析结果的速度，首要限制是上传的图像的大小，其次是服务端硬件设备的处理性能，最后Web客户端的实际网络情况也将影响接收图像结果的速度。

### 4.9.3 输入项

输入项为服务端处理完成的图像数据。

### 4.9.4 输出项

输出项为Web客户端接收到的图像数据。

### 4.9.5 设计方法

分析结果显示模块主要分为两个子模块：Web 客户端的图像数据显示子模块、前端数据接收子模块。

1. 首先，在 Web 客户端的图像数据显示子模块中：在Web页面中构建了Image信息显示的HTML DIV区域，Web客户端接受到的图像数据会在该区域进行展示。其中Image数据的显示使用到了HTML Img标签。
2. 其次，是前端数据接收子模块：该模块主要负责接收服务端处理好的图像数据。当用户成功上传图像后，服务端首先对图像进行处理，并返回指定处理后的图像数据，前端页面通过jQuery中的ajax()库函数修改Img标签图片的 URL 地址，然后显示在上述图像数据显示子模块中的 HTML Img 标签中。

### 4.9.6 接口

前后端分析结果显示接口，Web 客户端通过此接口向用户展示系统处理好的图片数据信息。

### 4.9.7 限制条件

网络状态是否通畅，浏览器是否能正常显示UI界面。

### 4.9.8 测试计划

对于该模块将主要进行 Web UI 界面测试，确保各个组件能够准确无误的显示在 Web 页面中。测试每个组件实际显示的信息是否与系统预先设定好的信息相同；测试服务端返回的图像数据是否与用户上传的待处理图像文件相匹配。

# 5 前端详细设计

**前端目录结构：**

│ .babelrc

│ .gitignore

│ .postcssrc.js

│ index.html //首页页面

│ package-lock.json //npm包信息

│ package.json //npm包信息

│ README.md //程序说明和指导文件

│

├─.idea

│ .gitignore

│ misc.xml

│ modules.xml

│ vcs.xml

│ vueweb.iml

│ workspace.xml

│

├─build //预编译目录

│ build.js

│ check-versions.js

│ logo.png

│ utils.js

│ vue-loader.conf.js

│ webpack.base.conf.js

│ webpack.dev.conf.js

│ webpack.prod.conf.js

│

├─config

│ dev.env.js

│ index.js //代理配置文件

│ prod.env.js

│

├─src

│ │ App.vue

│ │ main.js //主函数

│ │

│ ├─api

│ │ client.js //自定义内部接口文件

│ │

│ ├─assets //样式目录

│ │ ├─css

│ │ │ common.less

│ │ │ var.less

│ │ │

│ │ ├─font

│ │ │ iconfont.css

│ │ │ iconfont.eot

│ │ │ iconfont.svg

│ │ │ iconfont.ttf

│ │ │ iconfont.woff

│ │ │

│ │ └─img

│ │ banner1.jpg

│ │ banner2.jpg

│ │ banner3.jpg

│ │ banner4.jpg

│ │ banner5.jpg

│ │ code.png

│ │ index1.gif

│ │ index2.gif

│ │ market.JPG

│ │ placeholder.png

│ │ po.jpg

│ │ return.png

│ │

│ ├─components //组件目录

│ │ TextInput.vue //文本框组件

│ │

│ ├─config //配置目录

│ │ axios-client.js //axios拦截器配置文件

│ │

│ ├─pages

│ │ │ ErrorPage.vue //错误页面

│ │ │

│ │ └─client

│ │ Backstage.vue //基本导航页面

│ │ ChangeDetection.vue //变化检测页面

│ │ EditInfo.vue //信息编辑页面

│ │ GroundObjectsClassification.vue //地物分类页面

│ │ Login.vue //登录页面

│ │ ObjectDetection.vue //目标检测页面

│ │ TargetExtraction.vue //目标提取页面

│ │

│ ├─router

│ │ index.js //路由器配置文件

│ │

│ ├─store //数据存储配置文件

│ │ actions.js

│ │ getters.js

│ │ index.js

│ │ mutation-types.js

│ │ mutations.js

│ │ state.js

│ │

│ └─util //自定义工具(函数)

│ Global.js

│ util.js

│

└─static //静态数据目录

│ .gitkeep

│

└─img

store.jpg

stored.jpg

## 5.1 前端样式详细设计

### 5.1.1 登录/注册/密码找回：

登录界面由标题、选择栏、账号输入框、复选框、超链接和登录按钮组成。简洁美观，易于用户选择和操作。



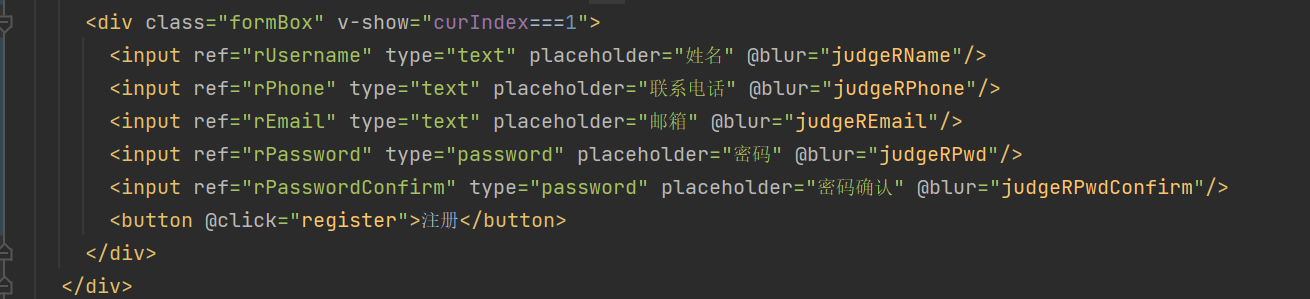
核心代码如图所示：



注册界面同样简洁明了，用户填写完成数据并点击注册按钮后即可注册成功。



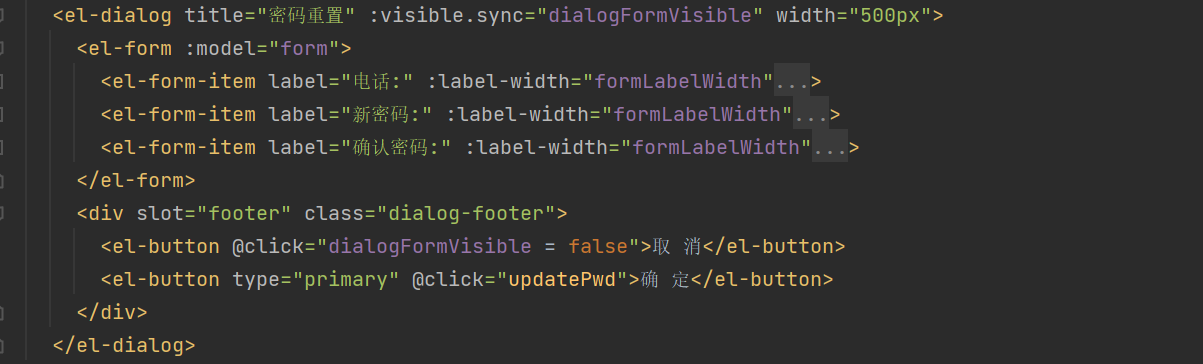
核心代码如图所示：



密码找回页面由模态框构成，用户输入正确的信息即可重置密码。



核心代码如图所示：

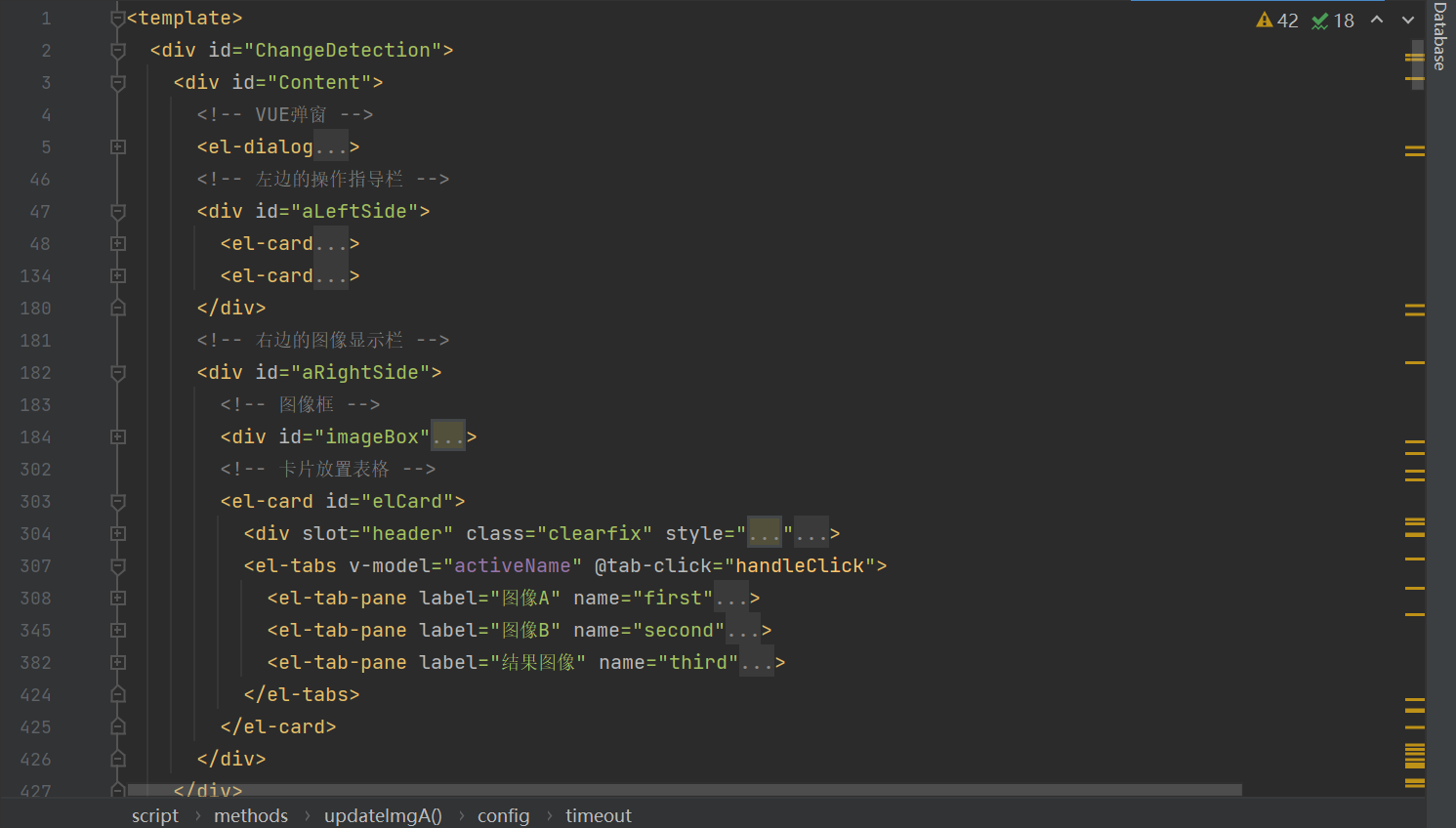


5.1.2 变化检测页面



变化检测页面样式由Element-UI设计实现，直观明了，同时有自适应功能，在不同分辨率的电脑上都能显示，易于操作。

核心框架如图所示：



5.1.3 地物分类页面



地物分类页面样式由Element-UI设计实现，直观明了，同时有自适应功能，在不同分辨率的电脑上都能显示，易于操作。

核心代码如图所示：

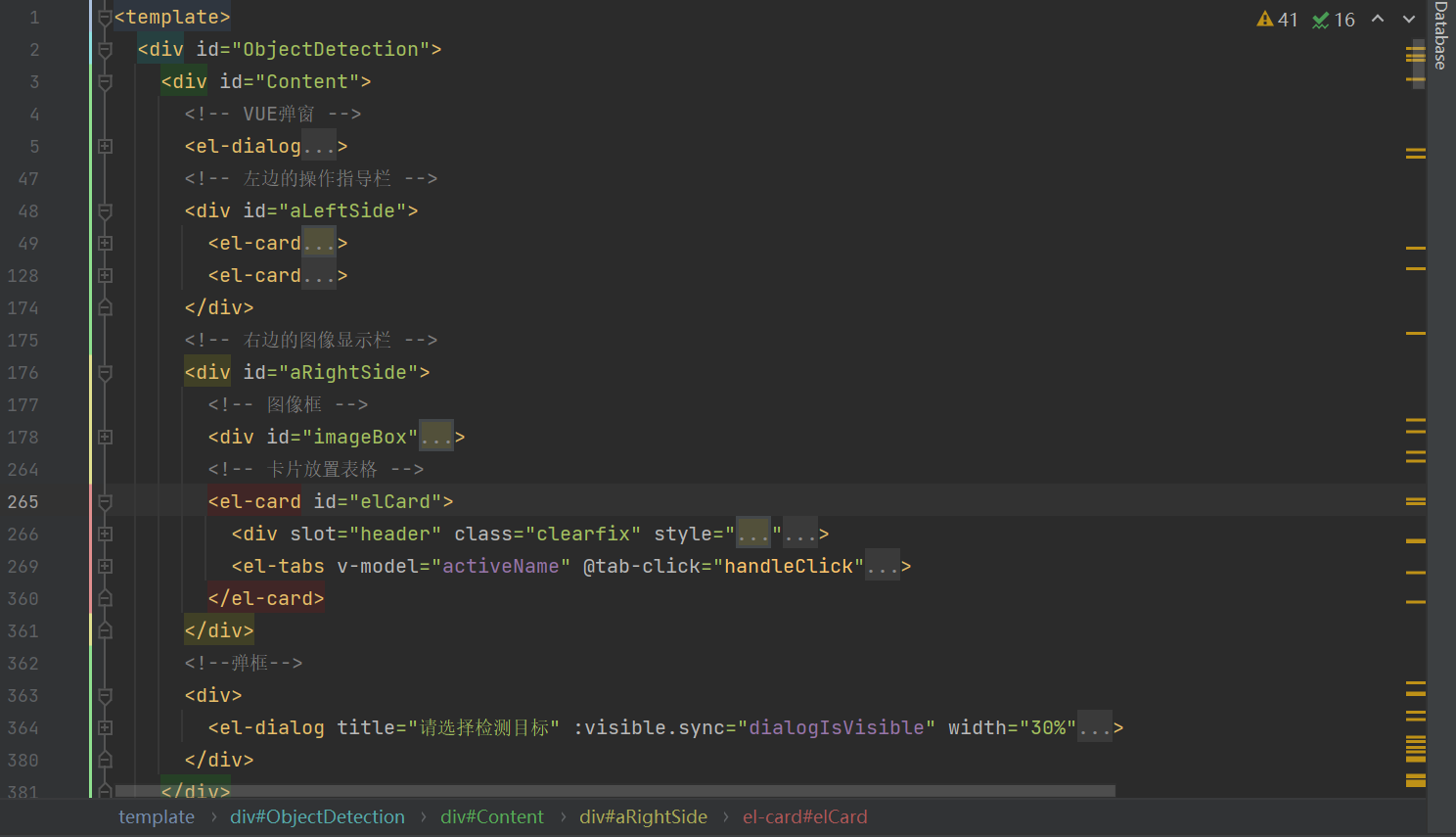


5.1.4 目标检测页面



目标检测页面样式由Element-UI设计实现，直观明了，同时有自适应功能，在不同分辨率的电脑上都能显示，易于操作。

核心代码如图所示：



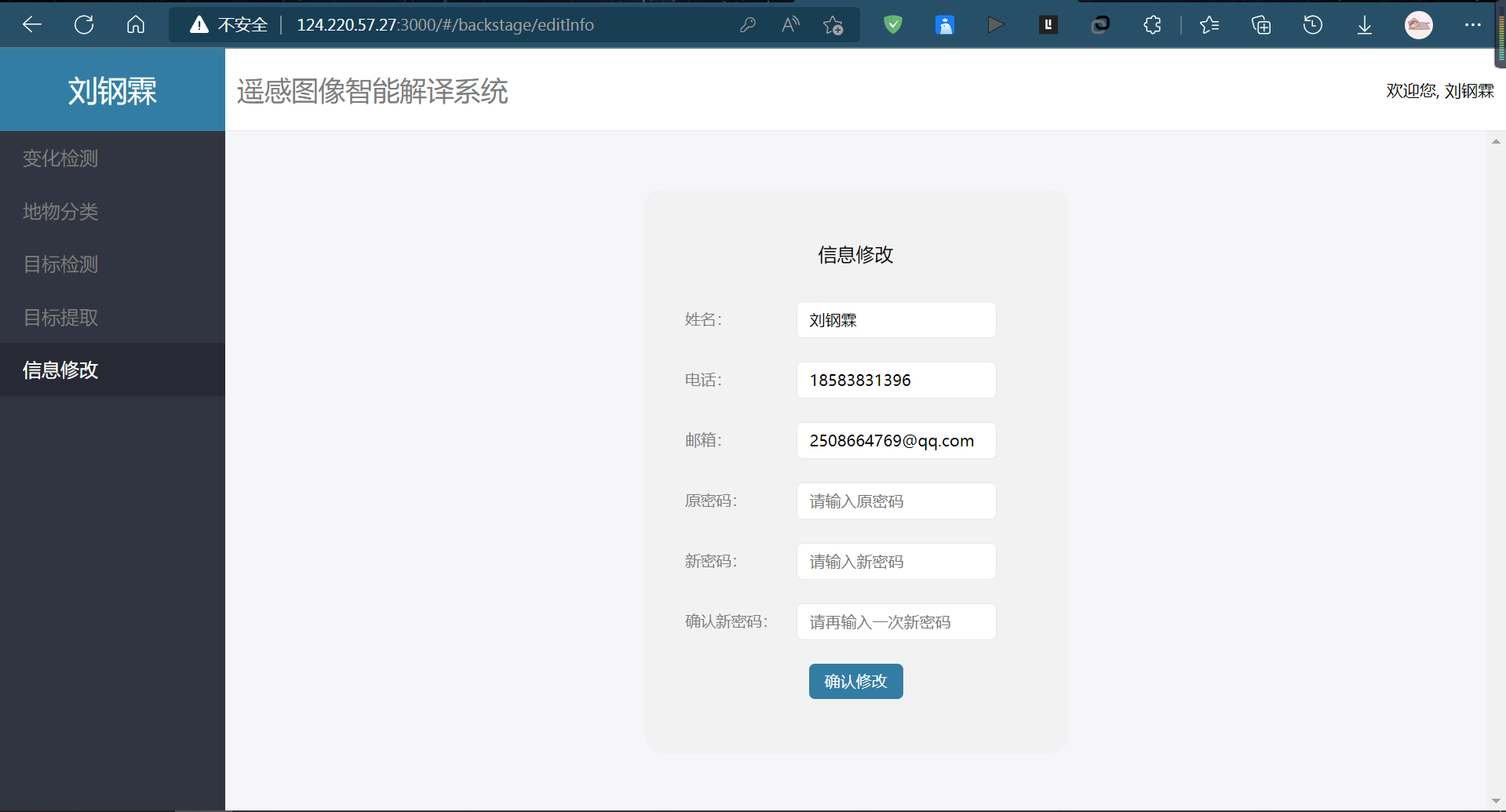
5.1.5 目标提取页面



目标提取页面样式由Element-UI设计实现，直观明了，同时有自适应功能，在不同分辨率的电脑上都能显示，易于操作。

核心代码如图所示：

5.1.6 信息修改页面



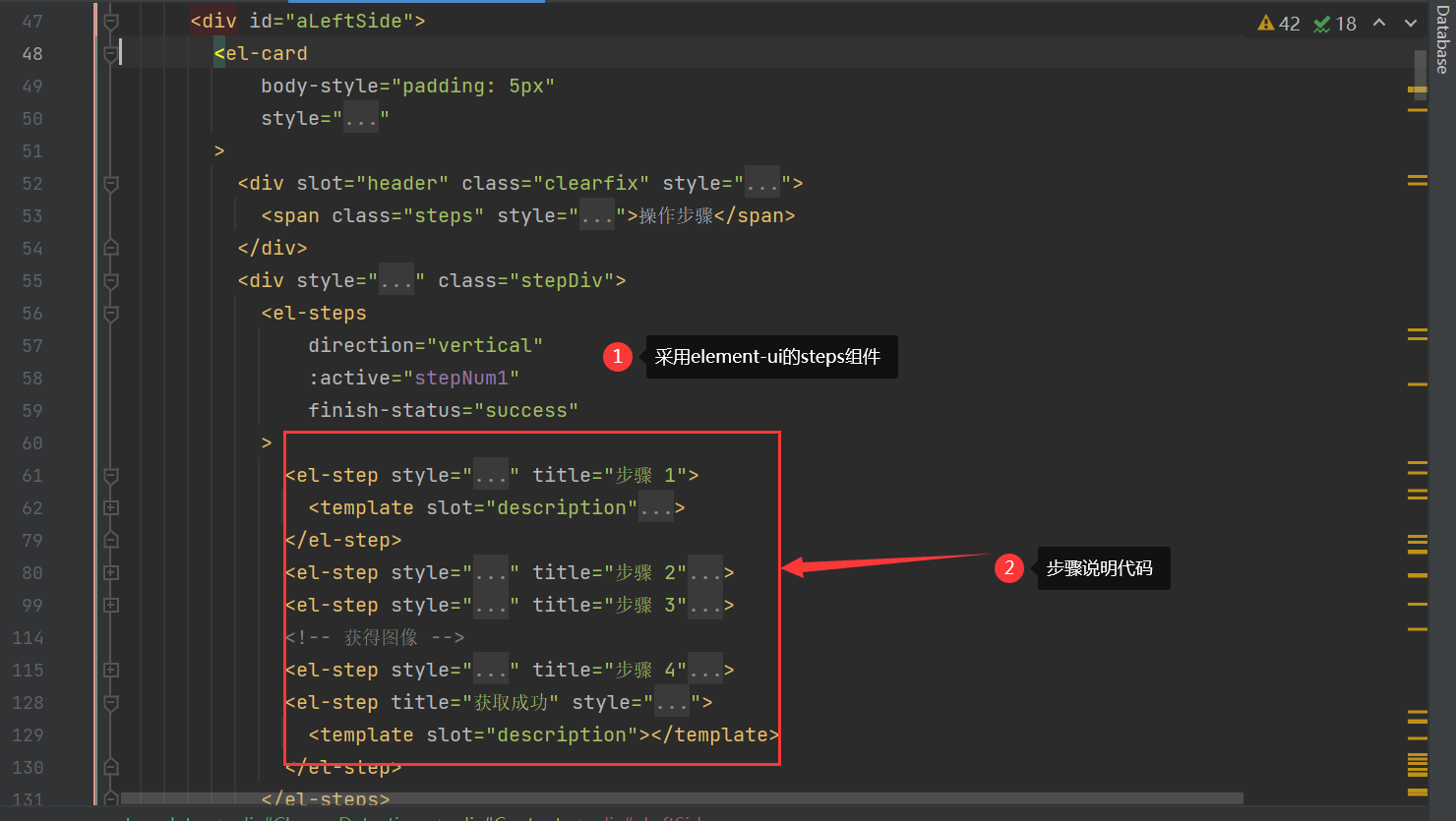
信息修改页面主要由Form表单组成，自适应功能，简洁明了易于操作。

## 5.2 前端引导详细设计

为了使用户快速上手变化检测、地物分类等功能的用户，页面中嵌入了引导组件，帮助用户一步步进行操作，让用户不会为如何操作而烦恼。

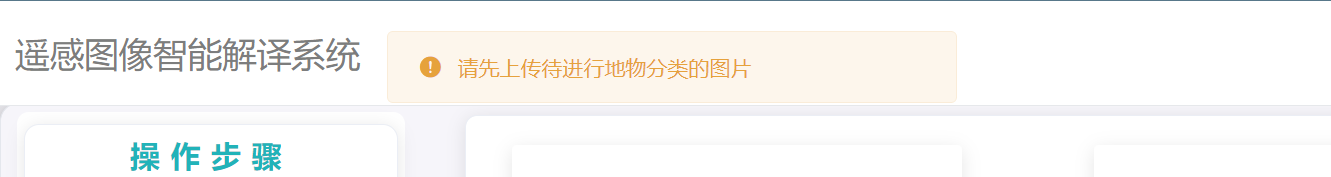


部分代码如下图所示：



## 5.3 前端反馈详细设计

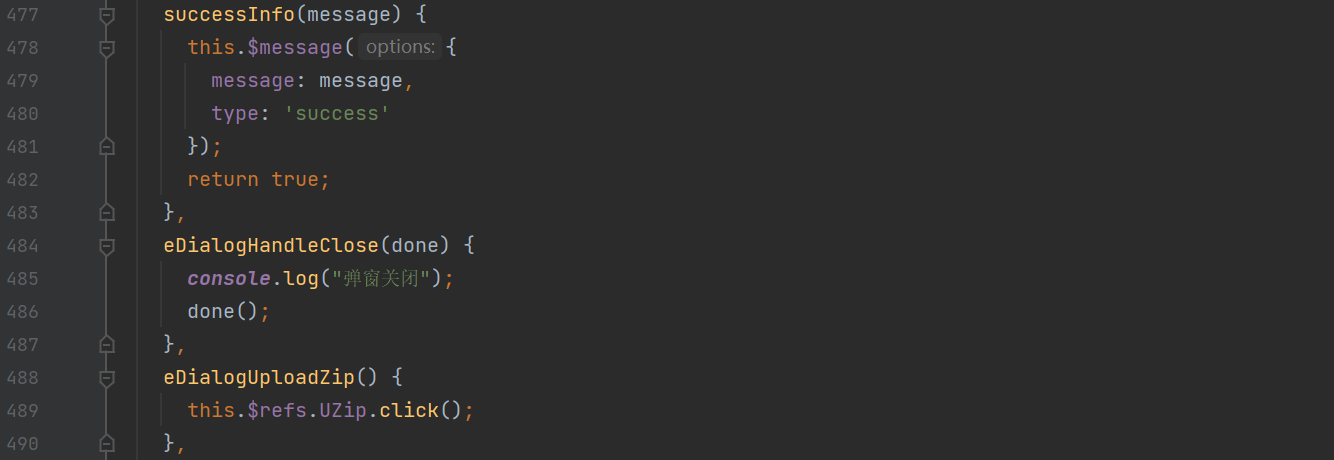
为了保证用户输入的数据和操作顺序的正确性，以及将网络或者后端的异常信息反馈给用户，前端框架中，加入了大量的信息反馈提示组件。告诉用户各种错误提示。如图所示：





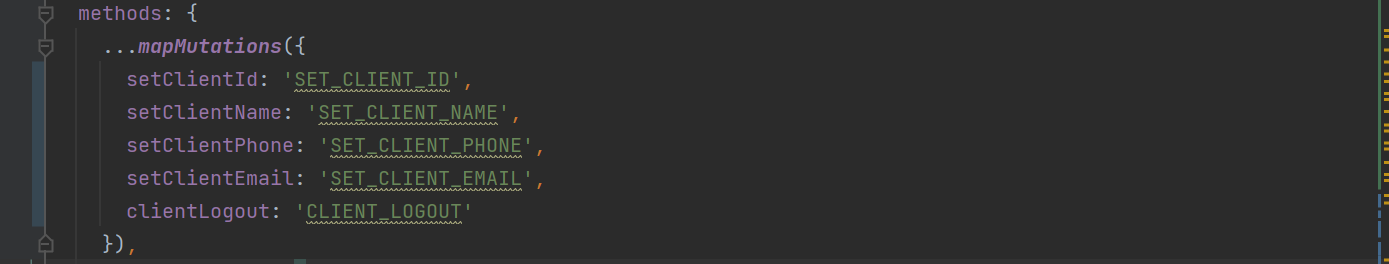


部分代码展示：



## 5.4 前端数据存储调度详细设计

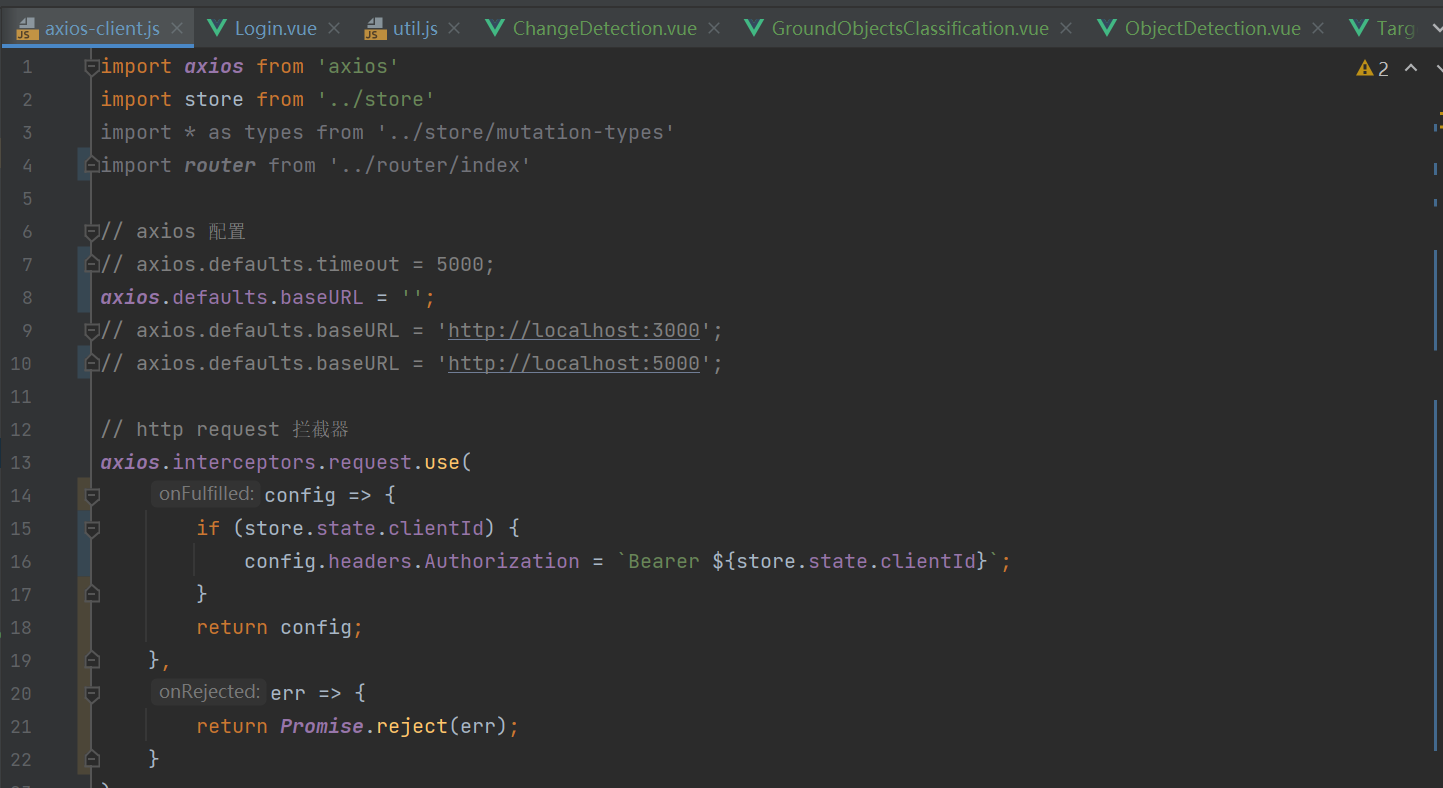
用户的数据使用mapMutation插件实现，登录信息由Cookie进行存储，相关代码如图所示：





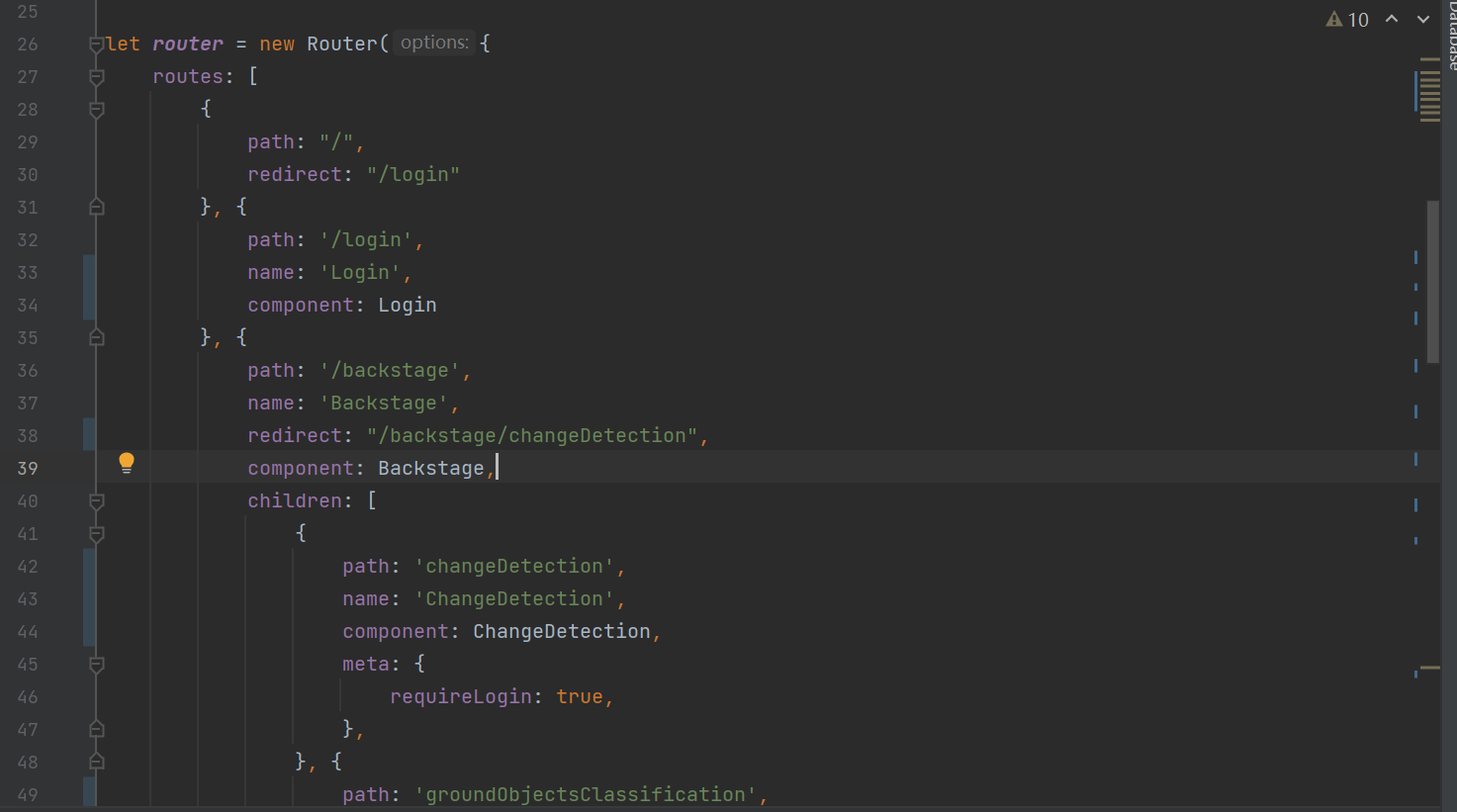
## 5.5 前端拦截器详细设计

通过判断用户的ID是否存在对非法请求进行拦截。核心代码如图：



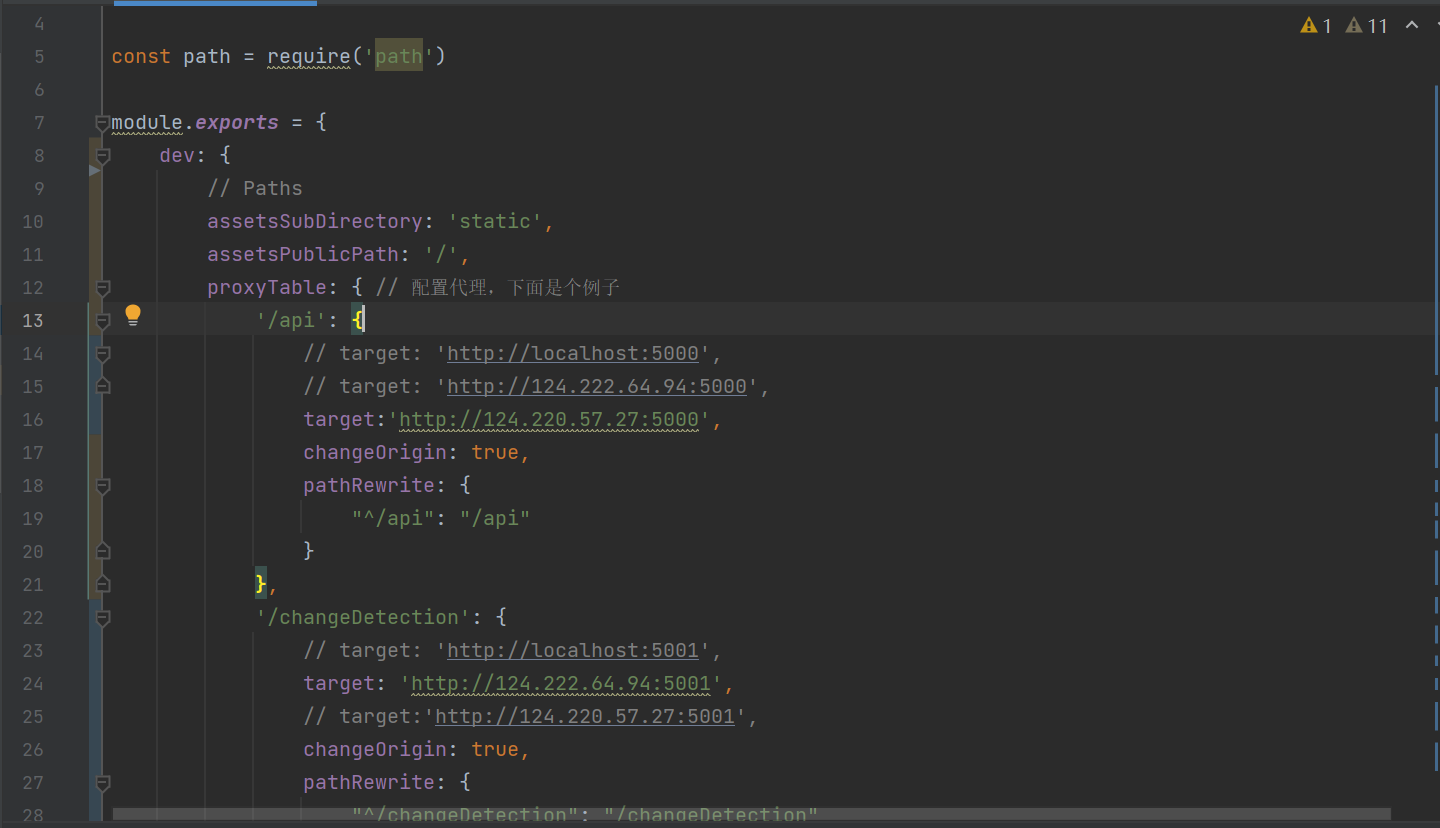
## 5.6 前端路由详细设计

Vue页面的跳转由路由实现，在router文件夹下的index.js文件中可配置路由跳转逻辑。如下图所示：



## 5.7 前端代理详细设计

为了访问服务器上的后端接口，需要进行跨域访问配置，可在config文件夹中的index.js文件中配置代理，部分代码如图所示：



# 6 后端详细设计

**后端目录结构：**

D:.

│ .gitignore

│ readme.md //介绍文件

│ serverAPI.py //数据库访问服务代码文件

│ serverCD.py //变化检测服务代码文件

│ serverGOD.py //地物分类服务代码文件

│ serverOD.py //目标检测服务代码文件

│ serverTE.py //目标提取服务代码文件

│

├─.idea

│ │ .gitignore

│ │ misc.xml

│ │ modules.xml

│ │ RSIIISystem.iml

│ │ vcs.xml

│ │ workspace.xml

│ │

│ └─inspectionProfiles

│ profiles\_settings.xml

│ Project\_Default.xml

│

├─model

│ └─exportedModel //该目录存放导出各个的模型

│ ├─changeDetection

│ │ .success

│ │ model.pdiparams

│ │ model.pdiparams.info

│ │ model.pdmodel

│ │ model.yml

│ │ pipeline.yml

│ │

│ ├─groundObjectsClassification

│ │ model.pdiparams

│ │ model.pdiparams.info

│ │ model.pdmodel

│ │ model.yml

│ │ pipeline.yml

│ │

│ ├─objectDetection

│ │ │ aircraft.zip

│ │ │

│ │ ├─aircraft

│ │ │ model.pdiparams

│ │ │ model.pdiparams.info

│ │ │ model.pdmodel

│ │ │ model.yml

│ │ │ pipeline.yml

│ │ │

│ │ ├─oiltank

│ │ │ model.pdiparams

│ │ │ model.pdiparams.info

│ │ │ model.pdmodel

│ │ │ model.yml

│ │ │ pipeline.yml

│ │ │

│ │ ├─overpass

│ │ │ model.pdiparams

│ │ │ model.pdiparams.info

│ │ │ model.pdmodel

│ │ │ model.yml

│ │ │ pipeline.yml

│ │ │

│ │ └─playground

│ │ model.pdiparams

│ │ model.pdiparams.info

│ │ model.pdmodel

│ │ model.yml

│ │ pipeline.yml

│ │

│ └─targetExtraction

│ │ inference\_model.zip

│ │

│ ├─inference\_model

│ │ model.pdiparams

│ │ model.pdiparams.info

│ │ model.pdmodel

│ │ model.yml

│ │ pipeline.yml

│ │

│ └─overpass

│ model.pdiparams

│ model.pdiparams.info

│ model.pdmodel

│ model.yml

│ pipeline.yml

│

├─utils //工具目录，功能代码的封装

│ │ aUtils.py //自定义基础工具库

│ │ database.py //数据库连接库

│ │ model.py //模型加载库

│ │

│ └─\_\_pycache\_\_

│ aUtils.cpython-37.pyc

│ database.cpython-37.pyc

│ model.cpython-37.pyc

│

└─\_\_pycache\_\_

## 6.1 后端接口详细设计

### 6.1.1 数据库访问接口-serverAPI.py

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路由 | 端口号 | 功能 | 请求 | 相应 |
| /api | 5000 | 测试服务器是否正常运行 | 请求:  GET  数据:无 | Welcome to RSIIISystem's API Server! |
| /api/client/login | 5000 | 登录 | 请求：  POST  数据：  电话号和密码json数据 | 登录成功或失败信息 |
| /api/client/register | 5000 | 注册 | 请求：  POST  数据：  用户个人信息json数据 | 注册成功或失败信息 |
| /api/client/updatePwd | 5000 | 密码重置 | 请求：  POST  数据：  用户验证json数据 | 密码重置成功或失败信息 |
| /api/client/update | 5000 | 信息更改 | 请求：  POST  数据：  用户信息json数据 | 更新成功或失败信息 |

### 6.1.2 变化检测访问接口-serverCD.py

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路由 | 端口 | 功能 | 请求 | 响应 |
| /changeDetection/  uploadImgA | 5001 | 接收并保存变化前的图片A | 请求：  POST  数据：  图像文件数据 | 上传成功或失败信息以及图像信息 |
| /changeDetection/  uploadImgB | 5001 | 接收并保存变化后的图片B | 请求：  POST  数据：  图像文件数据 | 上传成功说失败信息以及图像信息 |
| /changeDetection/  uploadZip | 5001 | 接收并保存压缩包数据 | 请求：  POST  数据：  文件数据 | 结果文件的压缩包连接 |
| /changeDetection/  getProgressNum | 5001 | 获取处理进度 | 请求：  POST  数据：  无 | 处理进度百分数 |
| /changeDetection/  handle | 5001 | 触发变化检测 | 请求：  POST  数据：  图像名json数据 | 结果图像的连接以及相关信息 |
| /changeDetection/  download/<string:filename> | 5001 | 下载处理结果文件 | 请求：  GET  数据：  文件名 | 后台文件数据 |
| /changeDetection/  getImage/<string:filename> | 5001 | 图片获取 | 请求：  GET  数据：  文件名 | 后台文件数据 |

### 6.1.3 地物分类访问接口-serverGOD.py

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路由 | 端口 | 功能 | 请求 | 响应 |
| /groundObjectsClassification/  uploadImg | 5002 | 接收并保存图片数据 | 请求：  POST  数据：  图像文件数据 | 上传成功或失败信息以及图像信息 |
| /groundObjectsClassification/  uploadZip | 5002 | 接收并保存压缩包数据 | 请求：  POST  数据：  文件数据 | 结果文件的压缩包连接 |
| /groundObjectsClassification/  getProgressNum | 5002 | 获取处理进度 | 请求：  POST  数据：  无 | 处理进度百分数 |
| /groundObjectsClassification/  handle | 5002 | 触发地物分类 | 请求：  POST  数据：  图像名json数据 | 结果图像的连接以及相关信息 |
| /groundObjectsClassification/  download/<string:filename> | 5002 | 下载处理结果文件 | 请求：  GET  数据：  文件名 | 后台文件数据 |
| /groundObjectsClassification/  getImage/<string:filename> | 5002 | 图片获取 | 请求：  GET  数据：  文件名 | 后台文件数据 |

### 6.1.4 目标检测访问接口-serverOD.py

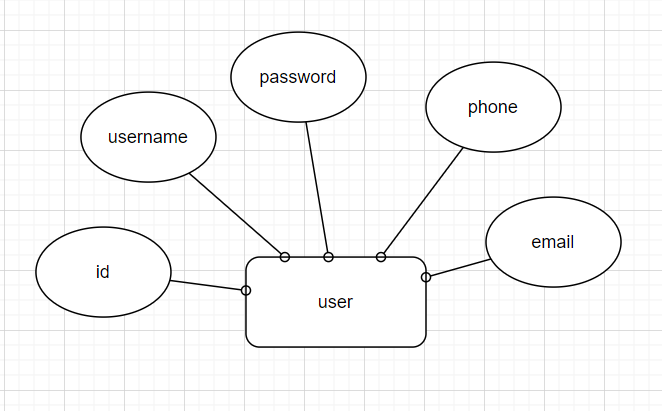
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路由 | 端口 | 功能 | 请求 | 响应 |
| /objectDetection/  uploadImg | 5003 | 接收并保存图片数据 | 请求：  POST  数据：  图像文件数据 | 上传成功或失败信息以及图像信息 |
| /objectDetection/  uploadZip | 5003 | 接收并保存压缩包数据 | 请求：  POST  数据：  文件数据 | 结果文件的压缩包连接 |
| /objectDetection/  getProgressNum | 5003 | 获取处理进度 | 请求：  POST  数据：  无 | 处理进度百分数 |
| /objectDetection/  handle | 5003 | 触发目标检测 | 请求：  POST  数据：  图像名json数据 | 结果图像的连接以及相关信息 |
| /objectDetection/  download/<string:filename> | 5003 | 下载处理结果文件 | 请求：  GET  数据：  文件名 | 后台文件数据 |
| /objectDetection/  getImage/<string:filename> | 5003 | 图片获取 | 请求：  GET  数据：  文件名 | 后台文件数据 |

### 6.1.5 目标提取访问接口-serverTE.py

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 路由 | 端口 | 功能 | 请求 | 响应 |
| /targetExtraction/  uploadImg | 5004 | 接收并保存图片数据 | 请求：  POST  数据：  图像文件数据 | 上传成功或失败信息以及图像信息 |
| /targetExtraction/  uploadZip | 5004 | 接收并保存压缩包数据 | 请求：  POST  数据：  文件数据 | 结果文件的压缩包连接 |
| /targetExtraction/  getProgressNum | 5004 | 获取处理进度 | 请求：  POST  数据：  无 | 处理进度百分数 |
| /targetExtraction/  handle | 5004 | 触发目标提取 | 请求：  POST  数据：  图像名json数据 | 结果图像的连接以及相关信息 |
| /targetExtraction/  download/<string:filename> | 5004 | 下载处理结果文件 | 请求：  GET  数据：  文件名 | 后台文件数据 |
| /targetExtraction/  getImage/<string:filename> | 5004 | 图片获取 | 请求：  GET  数据：  文件名 | 后台文件数据 |

## 6.2 数据库详细设计

### 6.2.1 数据库E-R实体图



### 6.2.2 数据库创建代码

CREATE TABLE `user` (

`id` bigint UNSIGNED NOT NULL AUTO\_INCREMENT,

`username` varchar(255) CHARACTER SET utf8mb4 NULL DEFAULT NULL,

`phone` varchar(255) CHARACTER SET utf8mb4 NULL DEFAULT NULL,

`email` varchar(255) CHARACTER SET utf8mb4 NULL DEFAULT NULL,

`password` varchar(255) CHARACTER SET utf8mb4 NULL DEFAULT NULL,

PRIMARY KEY (`id`) USING BTREE

) ENGINE = InnoDB AUTO\_INCREMENT = 11 CHARACTER SET = utf8mb4 ROW\_FORMAT = Dynamic;

### 6.2.3 数据库连接代码截图

