# 项目简介

用户上传一张图片，系统自动进行表单填写。

实现的主要功能应该有：
　　图片上传、获取图片OCR结果，标记图片文字识别区域，图片类型识别，表单定义，训练定制模型，表单自动填写等等。

# 软件设计方案

## 1.软件架构

本系统拟采用B/S架构，客户代码通过请求和应答的方式访问或者调用服务代码。

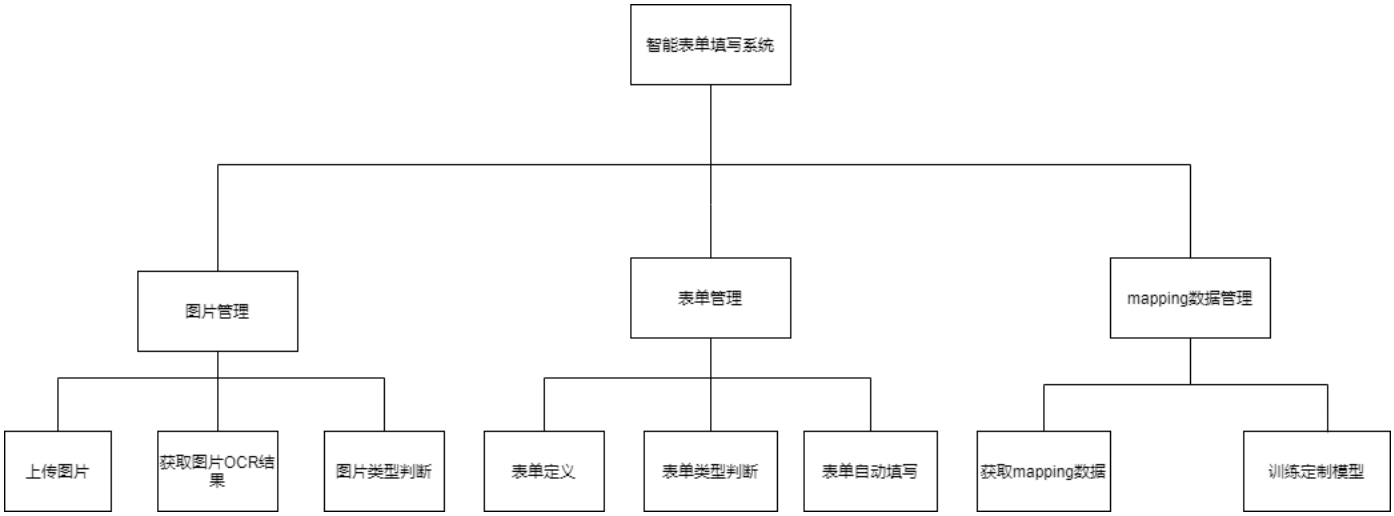
为了让前后端彻底独立开发，我们在开发上选择MVC风格，即模型-视图-控制器架构。Model（模型）代表一个存取数据的对象及其数据模型。View（视图）代表模型包含的数据的表达方式，一般表达为可视化的界面接口。Controller（控制器）作用于模型和视图上，控制数据流向模型对象，并在数据变化时更新视图。控制器可以使视图与模型分离开解耦合。

## 2.系统API接口

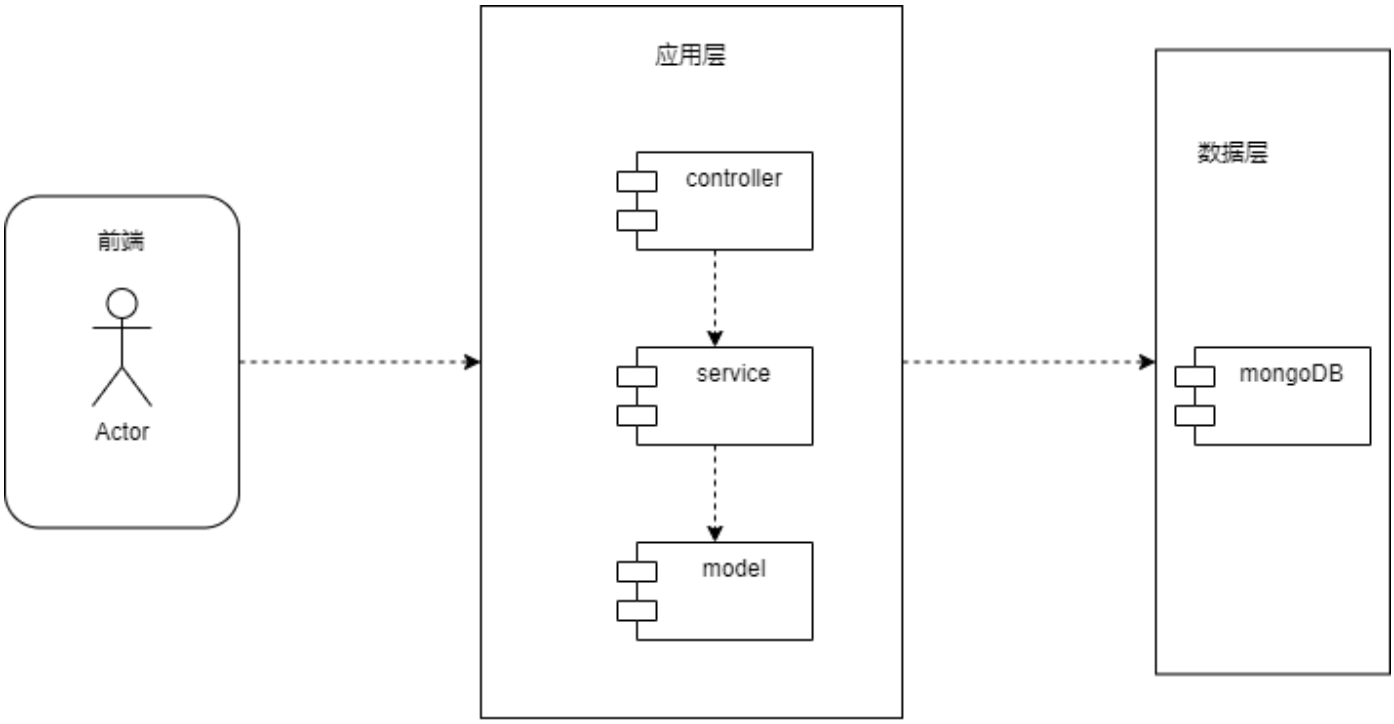
- 表单定义接口：
  输入：表单各输入框关键字
  输出：表单类别id
  功能：动态形成表单并获取表单类别
- 获取mapping数据接口：
  输入：图片，表单类型id
  输出：mapping数据，图片OCR数据，图片类型id
  功能：调用OCR接口获取图片OCR数据，并进行图片类型判断，根据图片类型以及表单类型找到相应的mapping数据
- OCR接口
  输入：图片
  输出：OCR数据
  功能：返回图片OCR结果
- 测试接口
  输入：mapping数据
  输出：true
  功能：存储图片类型id，表单类型id以及mapping数据为一条数据

## 3.视图分析

1.根据功能划分，可以得到功能分解视图

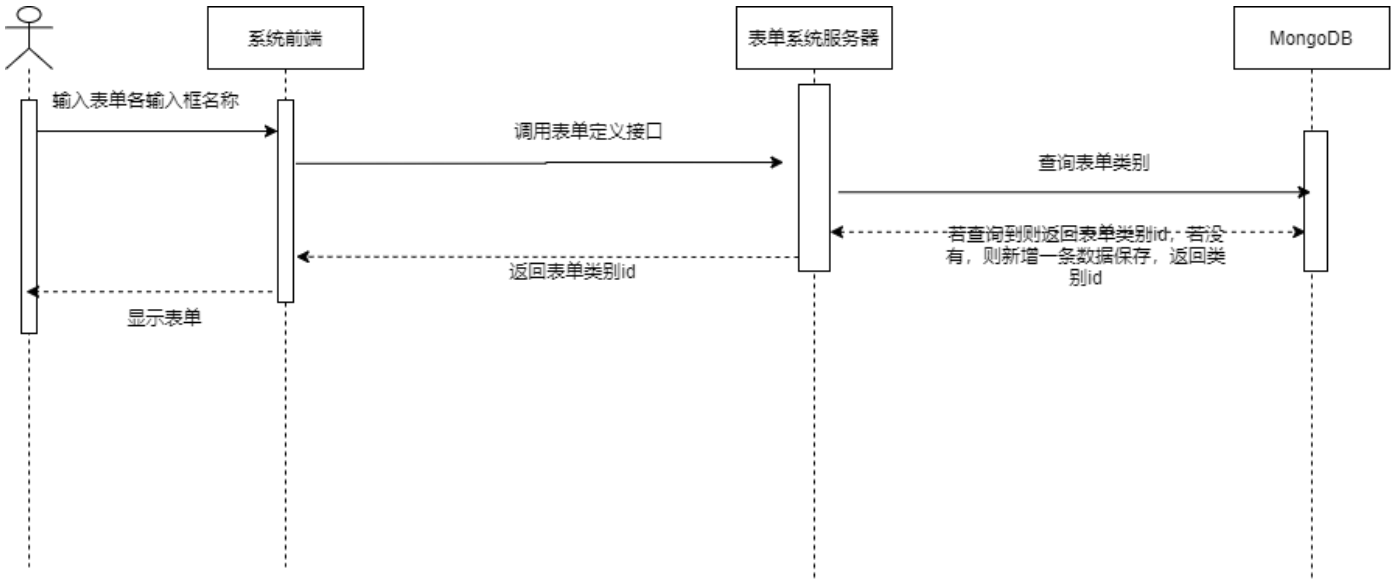


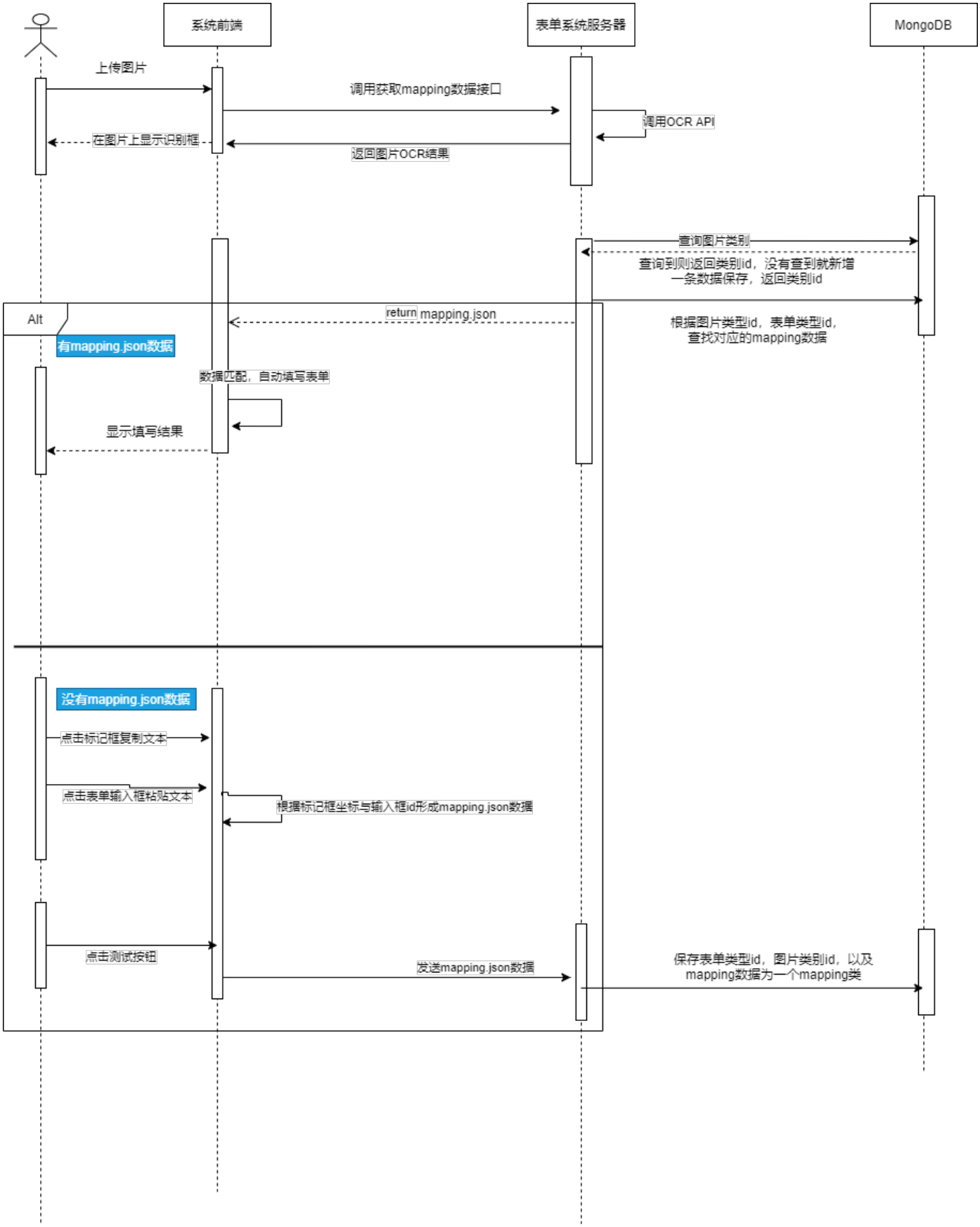

2.依赖视图




3.执行视图

用户使用该系统的步骤应为：先定义表单，后上传图片。


系统前端
表单系统服务器
MongoDB
输入表单各输入框名称
调用表单定义接口
查询表单类别
若查询到则返回表单类别id，若没有，则新增一条数据保存，返回类别id
返回表单类别id
显示表单




4.实现视图

```
src                                源代码目录
——server：后端相关代码
    config.py：定义可接受图片格式、配置端口
```

```
    main.py：响应前端请求，将图片以及识别结果（JSON）传入数据库
    ocr.py：调用有道OCR API，相关配置信息

——web：前端相关代码
    dist：生成打包后文件
    node_modules：安装的依赖包
    node_modules：安装的依赖包
    bottom.vue：copyrightDataModel.vue：定义运行界面右侧，数据模板相关组件，实现
saveDataModel()、addItem()、deleteItem()等方法
    DataModelManager.vue：定义DataModel相关管理操作，例如删除模板
    OcrCanvas.vue：定义运行界面左侧，“识别结果”界面相关vue组件，实现initDraw()、clickCanvas()等
方法
    DataManager.vue：是DataModelManager.vue、OcrCanvas.vue的父组件，实现运行界面左侧界面，定义
handleSuccess()、setOcrData()等方法。
    index.js：导出vuex所有配置
    utils：封装getOverlapbox()、findBox()、transform()等函数
    main.js：vue入口函数
```