图片管理网站 — 设计文档

1. 项目概述

本项目旨在实现一个图片管理网站，支持用户注册登录、图片上传、图片元数据（EXIF）读取、图片分类与标注、图片检索、图片编辑（缩放/裁剪/色调调整）、图片展示（轮播、缩略图）、AI 自动生成图片描述，以及移动端/微信内置浏览器友好展示。

系统使用 **Vue 3 + Element Plus** 作为前端框架，**Django + Django REST Framework** 作为后端 API 服务，数据库采用 **SQLite**，异步任务使用 **Celery + Redis**。

2. 需求梳理

2.1 基本功能

序号	功能
1	用户注册与登录（用户名、密码、email 验证、唯一性检查）
2	支持通过 PC 或手机浏览器上传图片并保存
3	自动读取图片 EXIF 信息并创建分类标签（拍摄时间、地点、分辨率等）
4	用户可为图片添加自定义标签，便于检索
5	生成缩略图用于列表与展示
6	图片元信息持久化存储于数据库，便于后续查询
7	提供多条件检索（时间、标签、上传者、分辨率等）
8	提供友好的展示界面（轮播、画廊）
9	提供简单图片编辑（裁剪、旋转、色彩微调）
10	删除与批量操作（批量删除、批量打标签）
11	移动端适配（响应式布局）与微信内置浏览器兼容

2.2 增强功能

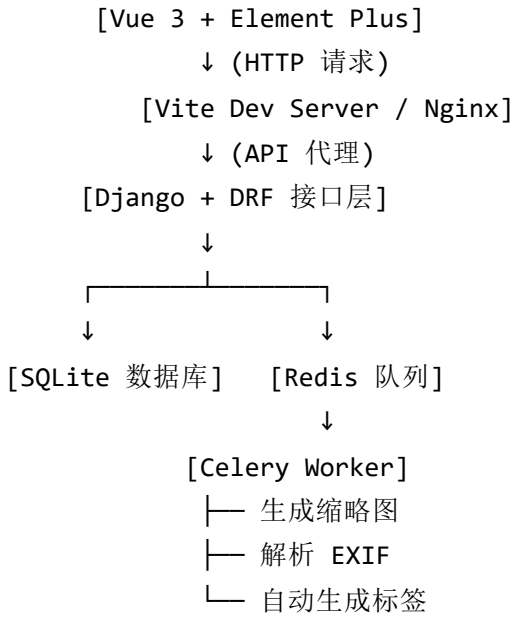
序号	功能
1	AI 模型自动分析图片并生成描述

3. 非功能性需求

- **可用性**：界面友好，交互清晰，支持移动端自适应
- **扩展性**：支持接入外部 AI 服务、对象存储
- **安全性**：权限控制、文件校验、API Key 加密存储、Token 鉴权
- **性能**：异步处理耗时任务，避免阻塞用户请求

4. 系统总体架构

采用前后端分离架构：



技术栈

层级	技术
前端	Vue 3 + Vite + Element Plus + Pinia
后端	Django 3.2 + Django REST Framework + djoser

层级	技术
异步任务	Celery + Redis
图片处理	Pillow + ExifRead
AI 服务	SiliconFlow API (deepseek-vl2)
数据库	SQLite
部署	Docker

5. 数据库设计

5.1 User（用户表）

字段名	类型	说明
id	Integer	主键
email	String	登录邮箱（唯一）
username	String	用户名
password	String	哈希密码
is_active	Boolean	是否激活
is_staff	Boolean	是否管理员
date_joined	DateTime	注册时间

5.2 UserProfile（用户资料表）

字段名	类型	说明
id	Integer	主键
user	ForeignKey	关联用户
avatar	ImageField	头像
bio	TextField	个人简介
_vision_api_key	BinaryField	加密的 API Key

5.3 Image（图片表）

字段名	类型	说明
id	Integer	主键
owner	ForeignKey	上传者
title	String	图片标题
description	TextField	图片描述
file	ImageField	原图路径
thumbnail	ImageField	缩略图路径
width	Integer	宽度
height	Integer	高度
file_size	Integer	文件大小
is_public	Boolean	是否公开
upload_time	DateTime	上传时间
processing_status	String	处理状态

5.4 Tag（标签表）

字段名	类型	说明
id	Integer	主键
name	String	标签名称（唯一）
tag_type	String	标签类型（user/auto/ai）
use_count	Integer	使用次数

5.5 ImageTag（图片-标签关联表）

字段名	类型	说明
id	Integer	主键
image	ForeignKey	关联图片
tag	ForeignKey	关联标签

字段名	类型	说明
created_at	DateTime	创建时间

5.6 ExifData（EXIF 数据表）

字段名	类型	说明
id	Integer	主键
image	OneToOne	关联图片
camera_make	String	相机制造商
camera_model	String	相机型号
taken_at	DateTime	拍摄时间
exposure_time	String	曝光时间
f_number	String	光圈值
iso_speed	Integer	ISO
focal_length	String	焦距
gps_latitude	Float	GPS 纬度
gps_longitude	Float	GPS 经度

6. API 接口设计

6.1 用户认证

方法	路径	说明
POST	`/api/auth/users/`	用户注册
POST	`/api/auth/token/login/`	用户登录
POST	`/api/auth/token/logout/`	用户登出
GET	`/api/auth/users/me/`	获取当前用户信息
PATCH	`/api/auth/users/me/`	更新用户信息

6.2 用户资料

方法	路径	说明
GET	`/api/users/profile/`	获取用户资料
PATCH	`/api/users/profile/`	更新用户资料
POST	`/api/users/profile/avatar/`	上传头像
GET	`/api/users/vision-api-key/`	获取 API Key
POST	`/api/users/vision-api-key/`	设置 API Key
DELETE	`/api/users/vision-api-key/`	删除 API Key

6.3 图片管理

方法	路径	说明
GET	`/api/images/`	获取图片列表
POST	`/api/images/`	上传图片
GET	`/api/images/{id}/`	获取图片详情
PATCH	`/api/images/{id}/`	更新图片信息
DELETE	`/api/images/{id}/`	删除图片
POST	`/api/images/{id}/edit/`	编辑图片
POST	`/api/images/{id}/generate_ai_description/`	AI 生成描述
POST	`/api/images/batch_delete/`	批量删除
POST	`/api/images/batch_update/`	批量更新
GET	`/api/images/random/`	获取随机图片

6.4 标签管理

方法	路径	说明
GET	`/api/tags/`	获取标签列表
GET	`/api/tags/hot/`	获取热门标签
GET	`/api/tags/{id}/images/`	获取标签下的图片

6.5 管理员接口

方法	路径	说明
GET	`/api/admin/images/`	管理所有图片
GET	`/api/admin/users/`	管理所有用户
GET	`/api/admin/stats/`	获取系统统计

7. 前端设计

7.1 页面结构

```
/                   # 首页（轮播展示）
/login              # 登录页
/register           # 注册页
/images             # 我的图片（列表/网格）
/images/:id         # 图片详情
/upload             # 上传页面
/profile            # 个人资料
/admin              # 管理后台
  /admin/images     # 图片管理
  /admin/users      # 用户管理
```

7.2 核心组件

组件	功能
ImageCard	图片卡片（缩略图 + 基本信息）
ImageGrid	网格布局展示
ImageList	列表布局展示
UploadPanel	拖拽上传面板
ImageEditor	图片编辑器（裁剪/旋转/调色）
TagEditor	标签编辑器
FilterPanel	筛选面板
Carousel	轮播组件

7.3 状态管理

使用 Pinia 管理全局状态：

- `userStore`：用户登录状态、Token 管理

8. 异步处理架构

```
上传请求 → Django 接收 → 保存原图 → 返回响应（即时）
                              ↓
                        触发 Celery 任务
                              ↓
                         Redis 消息队列
                              ↓
                        Celery Worker 处理
                         ├── 生成缩略图
                         ├── 解析 EXIF
                         └── 自动生成标签
                              ↓
                        更新数据库状态
```

9. AI 自动描述生成

9.1 功能说明

- 使用 SiliconFlow API 调用 deepseek-vl2 视觉模型
- 分析图片内容生成中文描述
- 用户需在个人设置中配置 API Key

9.2 API Key 安全存储

- 使用 Fernet 对称加密算法
- 加密密钥由 Django SECRET_KEY 派生
- 数据库中存储加密后的密文
- 前端仅显示脱敏后的 Key（前4位 + ... + 后4位）

9.3 调用流程

```
前端请求 → 验证用户身份 → 获取用户 API Key → 解密
    ↓
读取图片 → Base64 编码 → 调用 SiliconFlow API
    ↓
解析响应 → 更新图片描述 → 返回结果
```

10. 安全设计

措施	说明
Token 鉴权	使用 DRF Token 认证
文件类型校验	仅允许 jpg/png/gif/webp/bmp
权限控制	私有图片仅所有者可访问
API Key 加密	敏感信息加密存储
CORS 配置	限制跨域请求来源

11. 部署方案

使用 docker-compose 编排：

服务	说明
backend	Django + Gunicorn
frontend	Nginx 静态服务
redis	消息队列
celery-worker	异步任务处理
celery-beat	定时任务调度

12. 开发计划

阶段	内容
1	项目环境与基础工程结构
2	用户系统（注册/登录/鉴权）
3	图片上传与基础存储
4	图片展示与检索
5	EXIF 解析与标签系统
6	异步处理（缩略图/EXIF）
7	图片编辑与前端优化
8	AI 自动描述生成
9	安全、测试与交付

附录：项目目录结构

```
photo-manager/
├── backend/                    # Django 后端
│   ├── apps/
│   │   ├── users/              # 用户管理
│   │   │   ├── models.py       # User, UserProfile
│   │   │   ├── views.py        # 用户相关视图
│   │   │   └── serializers.py
│   │   ├── images/             # 图片管理
│   │   │   ├── models.py       # Image, ExifData
│   │   │   ├── views.py        # 图片 CRUD, AI 生成
│   │   │   └── tasks.py        # Celery 异步任务
│   │   └── tags/               # 标签管理
│   │       ├── models.py       # Tag, ImageTag
│   │       └── views.py
│   ├── config/                 # Django 配置
│   │   ├── settings.py
│   │   ├── celery.py
│   │   └── urls.py
│   ├── Dockerfile
│   └── requirements.txt
├── frontend/                   # Vue 前端
│   ├── src/
│   │   ├── views/              # 页面组件
│   │   ├── components/         # 公共组件
│   │   ├── utils/              # API 封装
│   │   └── stores/             # Pinia 状态
│   ├── Dockerfile
│   └── nginx.conf
├── docs/                       # 文档
├── docker-compose.yml
└── README.md
```