武汉大学

计算机学院实训

PhotoX软件系统设计说明书

项目成员：武星烨，廖文佳，吕天白，钱逸文，马郴敏

撰 写 人：武星烨，廖文佳，吕天白，钱逸文，马郴敏

2025 年 6 月 24 日

评 审 人：_____ _____年___月__日

武汉凡诺软件技术有限公司

2025 年

修订记录

日期	修订版本	修改章节	修改描述	修订人

目　录

关键词：

摘　要：

缩略语清单：〈对本文所用缩略语进行说明。〉

缩略语	英文全名	中文解释

1.简介

1.1目的

本文档旨在详细描述 Photox 图像管理平台的系统设计方案，涵盖其系统架构、模块划分、功能设计、数据流程、安全策略及部署方案。该文档面向项目开发人员、测试人员、产品经理以及未来的系统维护者，提供系统开发和后续演进的技术依据与指导。通过本设计文档，各相关人员能够准确理解 Photox 平台的系统目标、关键模块功能以及整体技术架构，从而保证系统开发的一致性和高质量交付。

1.2 范围

一、软件名称

photox

二、软件功能

Photox 图像智能管理平台旨在为用户提供一站式的图片管理与共享服务。该系统将实现如下核心功能：

图片上传与管理：支持多张图片批量上传，自动识别图片元数据（拍摄时间、地点等），并按时间线归档。

AI 智能标签与分类：基于深度学习模型，对图片内容进行识别，自动生成标签，如人物、场景、物体等，并按标签进行分类管理。

相册自动整理：根据图片时间、地点或标签自动生成相册，同时支持用户自定义相册、手动调整内容。

社区分享功能：支持图片与相册在平台内共享，可设置公开/私密权限，用户可对内容点赞、评论、收藏。

搜索与筛选：提供关键词搜索、标签筛选等多种图片检索手段，提升查找效率。

统计与推荐：通过用户使用行为数据进行智能推荐（如热图、标签云等），并提供用户上传/浏览/互动数据统计。

多终端访问：支持网页端访问，适配不同屏幕设备；系统后端提供 RESTful API 接口，支持未来扩展至移动端应用。

权限与角色管理：设有普通用户与管理员权限，管理员可管理用户、审核共享内容、配置系统参数。

不实现的功能（当前版本）：

图像编辑功能（如滤镜、美颜等）暂未包含；

视频文件的上传与处理暂不支持；

第三方社交平台（如微博、微信）直接分享暂未集成；

三、软件应用

Photox 作为一个基于人工智能的图像智能管理平台，适用于以下应用场景和用户群体：

个人用户：希望方便地存储、分类、查找和回忆生活中的照片，尤其是对照片内容自动分析和智能标签有需求的用户；

摄影爱好者与内容创作者：需上传大量照片并按照主题、场景进行高效管理与分享；

小型企业/组织：用于组织内部图片资料管理，例如项目文档、活动相册、团队展示等；

教育与研究机构：用于图像数据整理、教学内容素材库建设等；

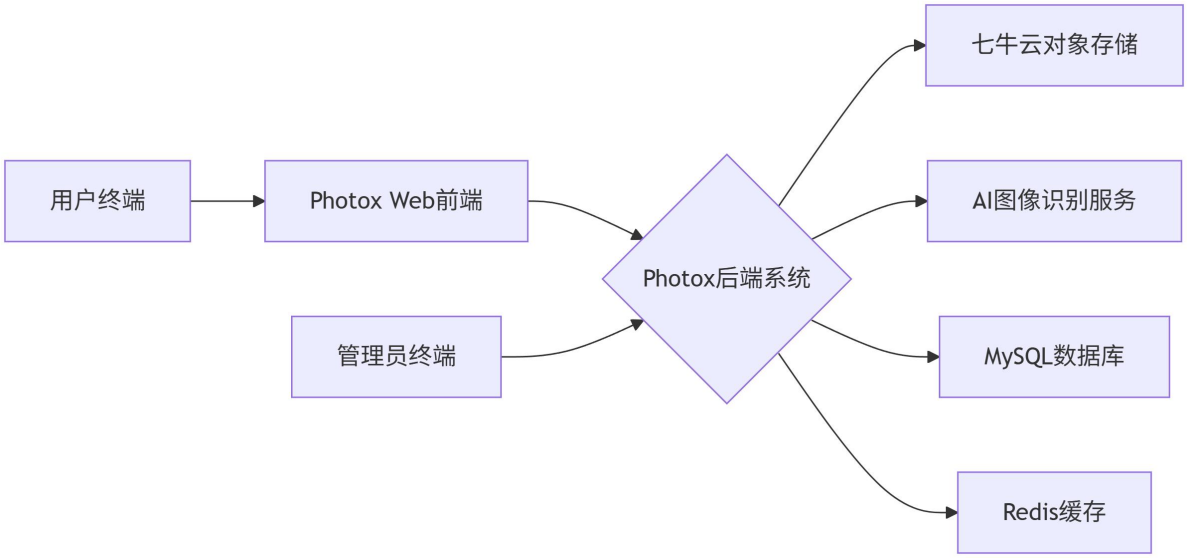
社交平台/图片社区的技术基础平台：可作为支持图像内容上传、管理与推荐功能的底层系统；

AI图像处理相关产品：可作为图像识别、内容挖掘的测试平台或前端可视化接口。

该系统结合“图像管理 + 智能分析 + 社区分享”三大功能于一体，具备良好的可扩展性，适用于现代互联网图像应用的广泛场景。

2.第0层设计描述

2.1 软件系统上下文定义



外部实体说明：

用户终端

属性：浏览器（Chrome/Firefox/Safari）、操作系统（Windows/macOS/移动端）、网络环境（≥10Mbps 带宽）

交互：通过 HTTPS 访问前端 SPA，发送 API 请求，接收 JSON 响应及静态资源。

七牛云对象存储

属性：RESTful API 接口、CDN 加速、SSD 存储、Access Key/Secret Key 认证

交互：后端通过 SDK 上传/删除图片，返回资源 URL。

AI 图像识别服务

属性：预训练深度学习模型（ResNet/YOLO）、GPU 推理支持、JSON 格式结果

交互：后端异步调用，输入图片 URL，输出标签/分类置信度。

MySQL 数据库

属性：关系型数据存储、InnoDB 引擎、主从复制支持

交互：后端 ORM 操作，存储用户、图片元数据、相册关系。

Redis 缓存

属性：内存键值存储、支持过期机制

交互：缓存 JWT 黑名单、热点图片元数据、搜索索引。

2.2 设计思路

一、设计可选方案

方案	优点	缺点	选择结果
单体架构（Django 全栈）	开发简单，部署便捷	前后端耦合，难以扩展	✗ 否决
微服务架构	模块解耦，独立伸缩	运维复杂，网络延迟高	✗ 过度设计
前后端分离	前端灵活迭代，后端专注 API	需协调接口规范	✓ 采用

选定方案理由：
契合团队技术栈（Vue3+Django），降低学习成本
支持 Web/移动端多平台适配

二、设计约束

技术栈强制

后端：Python 3.10+ / Django 4.x / DRF

前端：Vue 3 / Vite / ElementPlus

数据库：MySQL 8.0

存储：七牛云 SDK（不可更换）

安全合规

数据加密：TLS 1.3 传输，bcrypt 密码哈希

权限控制：RBAC 模型 + JWT 认证

合规性：遵循《个人信息保护法》，提供数据删除接口

三、遵循标准

类型	标准/规范
API 设计	RESTful，OpenAPI 3.0
前端	ES2022，Vue 风格指南
代码质量	PEP8（Python），ESLint（JS）
部署	Docker Compose v2，12Factor

四、硬件限制

最低生产配置：CPU：4 核心以上

内存：8 GB 及以上

存储：支持 SSD 磁盘，最低 50GB，可挂载对象存储

网络：公网 IP，带宽 ≥ 10 Mbps，支持 HTTPS 与防火墙策略

GPU（可选）：用于后续扩展 AI 模型推理优化

五、技术限制

文件格式与大小：图片上传仅支持 JPEG 和 PNG 格式，且单张图片大小不能超过 50MB。

AI 分析性能：AI 分析处理时间被要求不超过 5 秒，这可能限制了分析模型的复杂度和深度。

第三方服务依赖：系统的稳定性依赖于第三方服务，如云存储（阿里云 OSS, AWS S3）和 AI 分析服务（OpenAI API 等）。这些服务的稳定性、可用性或策略变更将直接影响本软件的运行。

浏览器支持：仅支持现代主流浏览器，无法兼容过时或非主流的浏览器版本。

六、其他

1.弹性扩展

水平扩展：无状态 API 层（Django + Gunicorn + Nginx LB）

异步任务：Celery + Redis 处理图片分析/备份等长时操作

2.降级策略

AI 服务不可用时，跳过自动标签，允许手动补标

七牛云故障时，启用本地临时存储（报警通知管理员）

3.成本优化

图片压缩：上传时生成 WebP 格式缩略图（原图保留）

CDN 缓存：热点相册资源预加载，减少回源流量

3. 第一层设计描述

3.1 系统结构

一、系统结构描述

photox 系统采用分层架构设计，主要分为前端显示层、后端业务逻辑层、数据库存储层和 AI 模块服务层。

前端显示层：基于 Vue.js 实现 SPA 页面，负责用户界面渲染与用户交互操作；

后端逻辑层：基于 Django 框架构建，提供 RESTful API 接口，处理用户请求、业务逻辑判断、

安全认证与权限控制；

数据库层：使用 MySQL 数据库管理系统，存储用户数据、图片元数据、相册信息等；

AI 模块服务层：集成深度学习模型（如图像识别模型）服务，实现标签生成、人脸识别、场景识别等功能。

系统结构设计遵循高内聚低耦合原则，模块之间通过标准接口进行通信，便于维护和扩展。二、业务流程说明

二、业务流程说明

典型用例：用户上传图片后系统完成自动分类和智能标注。

业务流程如下：

用户通过前端上传图片

前端将图片通过 HTTP POST 请求发送至 Django 后端接口

后端保存图片，并将图片传送至 AI 模型模块处理

AI 模型识别图片内容，返回标签信息

后端将标签与图片关联存储到数据库中

前端异步刷新图片库并展示标签结果

3.2 分解描述

本节描述系统中的子系统和模块。

一、模块1/子系统1描述

1.Overview 简介：

负责用户的注册、登录、身份认证、权限控制、角色分配。

2.Functions 功能列表：

用户注册/登录/注销

权限验证中间件

不同角色访问控制（普通用户/管理员）

二、模块2/子系统2描述

1.Overview 简介：

对用户上传的图片进行内容分析、标签生成、特征提取等。

2.Functions 功能列表：

图像预处理

调用模型生成智能标签

提取人物、人脸、场景等信息

3.3 依赖性描述

本节描述系统中的子系统，数据结构，模块，进程等设计实体间的关系。

依赖关系描述可以使用文字，结构图，（交互）事务图。

用户管理模块依赖于 Django 自带的 auth 系统

图像处理模块依赖 AI 模型服务（可部署为 Flask REST 接口或微服务）

后端所有业务逻辑模块都依赖数据库操作模块

所有接口服务通过统一的 API 路由映射并提供接口文档

3.4 接口描述

本节描述软件系统中设计实体(如子系统，模块，进程)的接口.

接口描述可以使用接口文件，参数表。

对于外部实体只有同被描述软件相关的接口才需描述。

接口可以是函数调用、事件、消息、信号等。

一、社区模块接口描述

1. 评论接口

名称：创建评论

说明：用户对相册或图片发表评论，支持回复评论。

定义：

URL：`POST /api/v1/community/comments/`

参数：

参数名	类型	必填	说明
album	int	是	相册 ID
content	string	是	评论内容
parent	int	否	父评论 ID（回复时用）

返回：评论对象 JSON

名称：获取评论列表

说明：获取某相册下的所有评论。

定义：

URL：`GET /api/v1/community/comments/?album=<album_id>`

参数：album_id（URL 参数）

返回：评论列表 JSON

2. 点赞接口

名称：切换点赞状态

说明：对相册、评论或图片进行点赞/取消点赞。

定义：

URL： `POST /api/v1/community/likes/toggle/`

参数：

参数名	类型	必填	说明
like_type	string	是	点赞类型（album/comment/image）
object_id	int	是	被点赞对象 ID

返回： `{ liked: bool, like_count: int, ... }`

名称：检查点赞

说明：查询当前用户是否已点赞某对象及总点赞数

- 定义
  - URL： `GET /api/v1/community/likes/check/?like_type=xxx&object_id=xx`
  - 返回： `{ liked: bool, like_count: int`
  -

3 关注接

名称：切换关注状

说明：关注/取消关注某用户

定义：

URL： `POST /api/v1/community/follows/<user_id>/toggle/`

返回： `{ following: bool, message: string }`

名称：获取粉丝列

说明：获取某用户的粉丝。

定义：

URL：`GET /api/v1/community/follows/<user_id>/followers/`

返回：粉丝用户列表

名称：获取关注列表

说明：获取某用户关注的人。

定义：

URL：`GET /api/v1/community/follows/<user_id>/following/`

返回：关注用户列表

4. 通知接口

名称：获取通知列表

说明：获取当前用户的所有通知。

定义：

URL：`GET /api/v1/community/notifications/`

返回：通知列表

名称：获取未读通知数量

说明：获取当前用户未读通知数量。

定义：

URL：`GET /api/v1/community/notifications/unread_count/`

返回：`{ unread_count: int }`

5. 用户接口

名称：获取用户信息

说明：获取指定用户或当前用户信息，包括关注数和粉丝数

定义：

URL：`GET /api/v1/community/users/<user_id>/` 或 `/me/`

返回：用户信息（含 follower_count, following_count

三、用户模块接口描述

1. 用户注册

名称：用户注册

说明：新用户注册账号。

定义：

URL：`POST /api/v1/users/register/`

参数：

参数名	类型	必填	说明
username	string	是	用户名
email	string	是	邮箱
password	string	是	密码

返回：用户对象 JSON

2. 用户登录

名称：用户登录

说明：用户登录获取 Token

定义：

URL：`POST /api/v1/users/login/` 或 `/api/token/`

参数：

参数名	类型	必填	说明
username	string	是	用户名
password	string	是	密码

返回：`{ token: string }` 或 `{ access: string, refresh: string }`

3. 名称：获取用户信息

说明：获取指定用户或当前用户信息。

定义：

URL：`GET /api/v1/community/users/<user_id>/` 或 `/me/`

返回：用户信息（含关注数、粉丝数等）

四、图片模块接口描述

1. 上传图片

名称：上传图片

说明：用户上传图片到相册。

定义：

URL：`POST /api/v1/images/upload`

参数：

参数名	类型	必填	说明
album	int	是	相册 ID

参数名	类型	必填	说明
image	file	是	图片文件
title	string	否	图片标题

返回：图片对象 JSON

2. 获取图片列表

名称：获取图片列表

说明：获取某相册下的所有图片。

定义：

URL：`GET /api/v1/images/?album=<album_id>`

返回：图片列表 JSON

3. 获取图片详情

名称：获取图片详情

说明：获取单张图片的详细信息。

定义：

URL：`GET /api/v1/images/<image_id>/`

返回：图片对象 JSON

4. 删除图片

名称：删除图片

说明：用户删除自己上传的图片。

定义：

URL：`DELETE /api/v1/images/<image_id>/`

返回：`204 No Content`

五、相册模块接口描述

1. 创建相册

名称：创建相册

说明：用户新建相册。

定义：

URL：`POST /api/v1/albums/`

参数：

参数名	类型	必填	说明
title	string	是	相册标题
description	string	否	相册描述

返回：相册对象 JSON

2. 获取相册列表

名称：获取相册列表

说明：获取用户的所有相册。

定义：

URL：`GET /api/v1/albums/?user=<user_id>`

返回：相册列表 JSON

3. 获取相册详情

名称：获取相册详情

说明：获取单个相册的详细信息。

定义：

URL：`GET /api/v1/albums/<album_id>/`

返回：相册对象 JSON

4. 删除相册

名称：删除相册

说明用户删除自己创建的相册。

定义：

URL：`DELETE /api/v1/albums/<album_id>/`

返回：`204 No Content`

六、AI 图片识别模块接口描述

1. 图片智能分类

名称：图片智能分类

说明：对上传的图片进行 AI 自动分类。

定义：

URL：`POST /api/v1/ai/classify/`

参数：

参数名	类型	必填	说明
image	file	是	图片文件

返回：`{ class: string, confidence: float }`

2. 多模型图片识别

名称：多模型图片识别

说明：对图片进行多模型识别，返回多种标签。

定义：

URL：`POST /api/v1/ai/multimodel/`

参数：

参数名	类型	必填	说明
image	file	是	图片文件

返回：`{ results: [ { model: string, class: string, confidence: float }, ... ] }`

七、系统管理模块接口描述

1. 管理员登录

名称：管理员登录

说明：管理员后台登录。

定义：

URL：`/admin/login/`

参数：用户名、密码（表单）

返回：重定向到后台首页

2. 数据统计

名称：获取系统统计数据

说明：管理员获取用户数、图片数、相册数等统计信息。

定义：

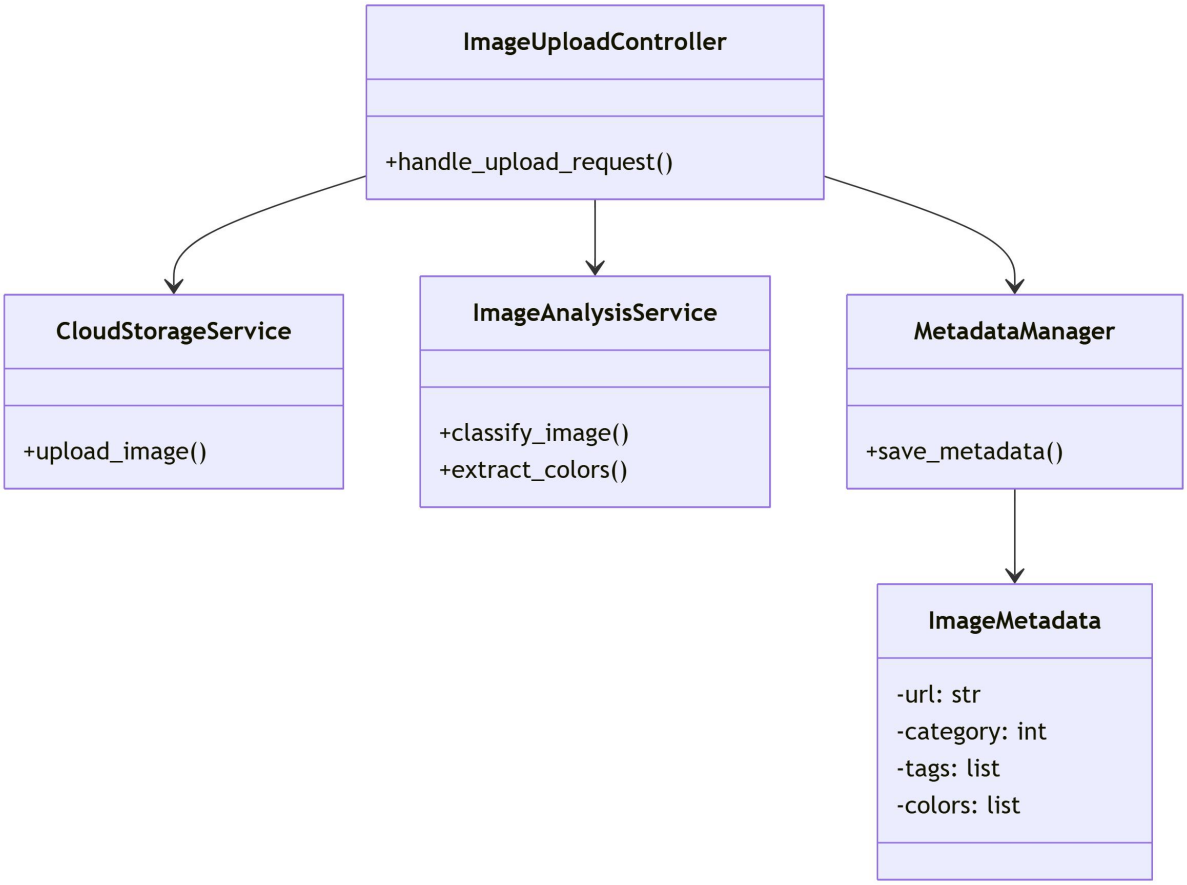
URL： GET /api/v1/admin/stats/

返回： { user_count: int, album_count: int, image_count: int, ... }

4. 第二层设计描述

4.1 图片上传模块

一、模块设计描述



1、类名：ImageUploadController

（1）标识

PhotoX_Upload_ImageUploadController

（2）简介

处理 HTTP 上传请求的控制器，协调上传流程

（3）类定义（Optional）

```
class ImageUploadController:
    def handle_upload_request(self, request):
        """
        处理图片上传请求
        参数：request - HTTP 请求对象（包含用户 ID 和图片文件）
        返回：上传结果（成功/失败）和图片 URL
        """
```

2、类名：CloudStorageService

（1）标识

PhotoX_Upload_CloudStorageService

（2）简介

封装七牛云存储的上传功能

（3）类定义（Optional）

```
class CloudStorageService:

    def __init__(self, access_key, secret_key):

        self.access_key = access_key

        self.secret_key = secret_key

    def upload_image(self, file_path, bucket_name, key):

        """

        上传图片到云存储

        参数：

            file_path: 本地图片路径

            bucket_name: 存储桶名称

            key: 云存储中的文件名

        返回：图片的公开访问 URL
```

```
    """
```

3、类名：ImageAnalysisService

（1）标识：PhotoX_Upload_CloudStorageService

（2）简介：封装七牛云存储的上传功能

（3）类定义：

```
class CloudStorageService:
    def __init__(self, access_key, secret_key):
        self.access_key = access_key
        self.secret_key = secret_key

    def upload_image(self, file_path, bucket_name, key):
        """
        上传图片到云存储
        参数：
            file_path: 本地图片路径
            bucket_name: 存储桶名称
            key: 云存储中的文件名
        返回：图片的公开访问 URL
        """
```

4.类名：MetadataManager

（1）标识：PhotoX_Upload_MetadataManager

（2）简介：管理图片元数据的存储

（3）类定义：

```
class MetadataManager:
    def save_metadata(self, image_url, user_id, category, tags, colors):
        """
        保存图片元数据到数据库
        参数：
            image_url: 图片 URL
```

user_id: 用户 ID

category: 分类 ID

tags: 标签列表

colors: 颜色列表

返回：操作结果

"""

5.类名：ImageMetadata

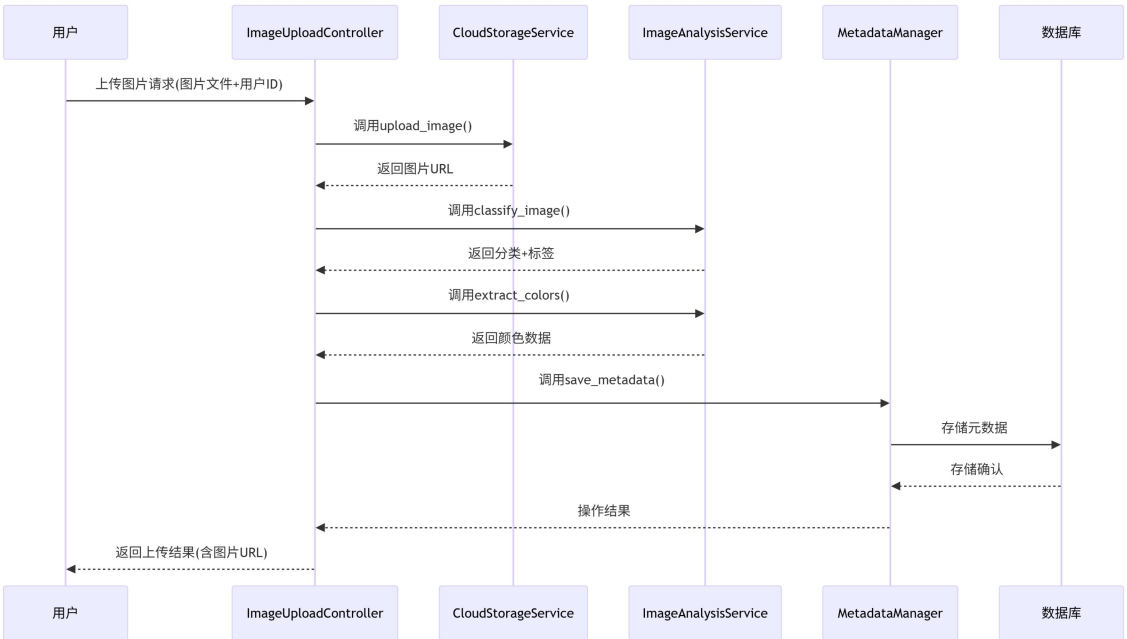
（1）标识：PhotoX_Upload_ImageMetadata

（2）简介：图片元数据模型（数据库实体）

（3）类定义：

```
class ImageMetadata(models.Model):
    url = models.URLField(max_length=255)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    category = models.IntegerField()
    tags = models.JSONField()
    colors = models.JSONField()
    created_at = models.DateTimeField(auto_now_add=True)
```

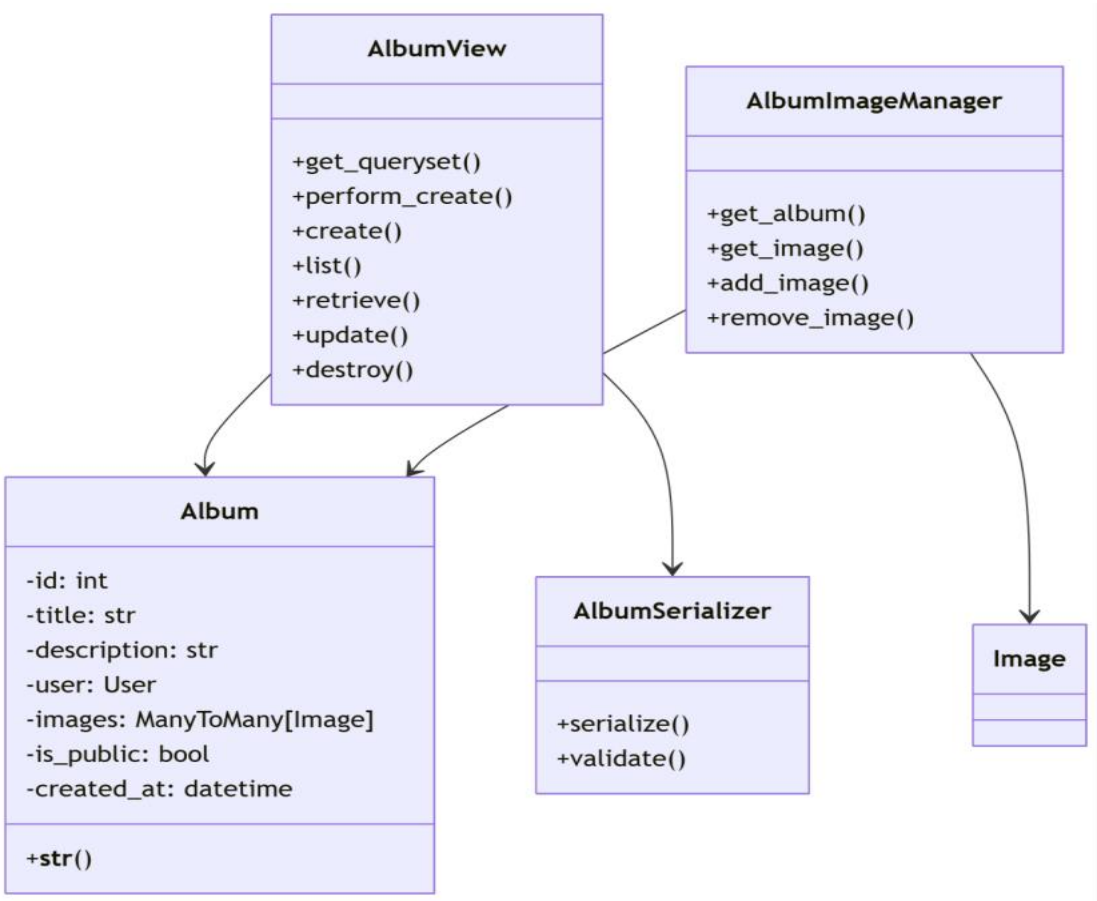
二、功能实现说明



4.2 相册管理模块

模块设计描述

相册管理模块负责处理相册的创建、管理、图片组织和权限控制...



1. 类名：Album

（1）标识：PhotoX_Album_Album

（2）简介：相册数据模型，表示用户创建的相册实体

（3）类定义：

```
class Album(models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True, null=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    images = models.ManyToManyField(Image, blank=True)
    is_public = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
```

2. 类名：AlbumView

（1）标识：PhotoX_Album_AlbumView

（2）简介：处理相册的创建、列表、详情、更新和删除

3. 类名：AlbumSerializer

（1）标识：PhotoX_Album_AlbumSerializer

（2）简介：序列化相册数据，处理数据验证

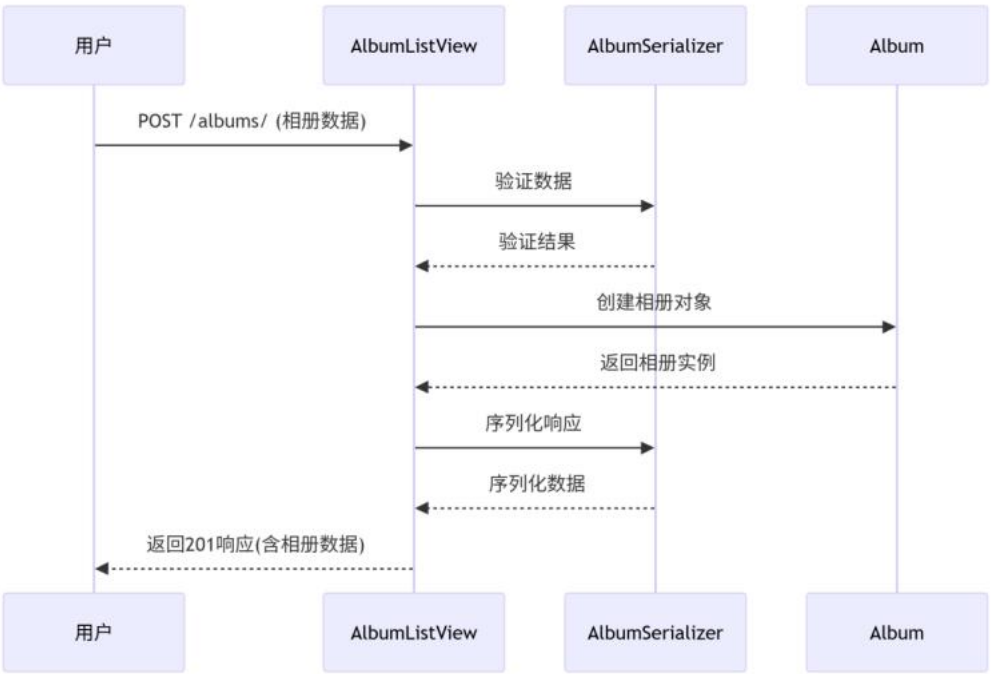
4. 类名：AlbumImageManager

（1）标识：PhotoX_Album_AlbumImageManager

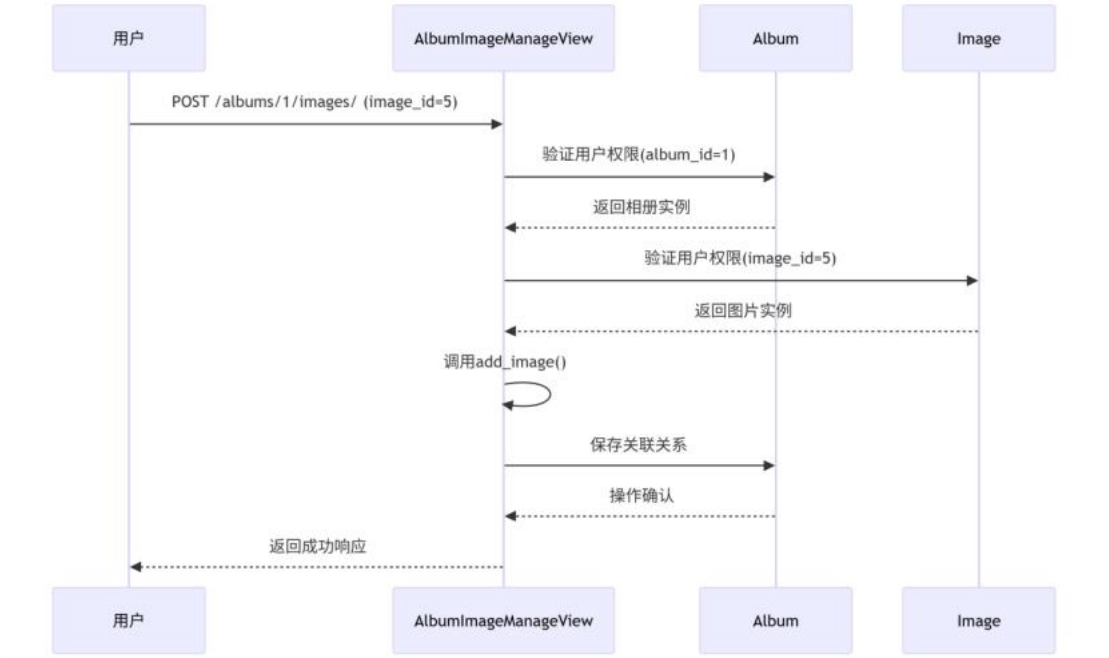
（2）简介：管理相册内图片的添加和移除

二、功能实现说明

1. 创建相册序列图



2. 添加图片到相册序列图



5. 数据库设计

5.1 实体定义

规范化设计：采用第三范式（3NF）进行设计，减少数据冗余，保证数据一致性。

实体关系映射：使用外键（Foreign Key）明确实体间的关联关系，并设置级联操作（如 ON DELETE CASCADE）来维护数据完整性 。

多对多关系处理：对于图片与标签、相册与图片之间的多对多关系，将创建独立的中间关联表进行管理。

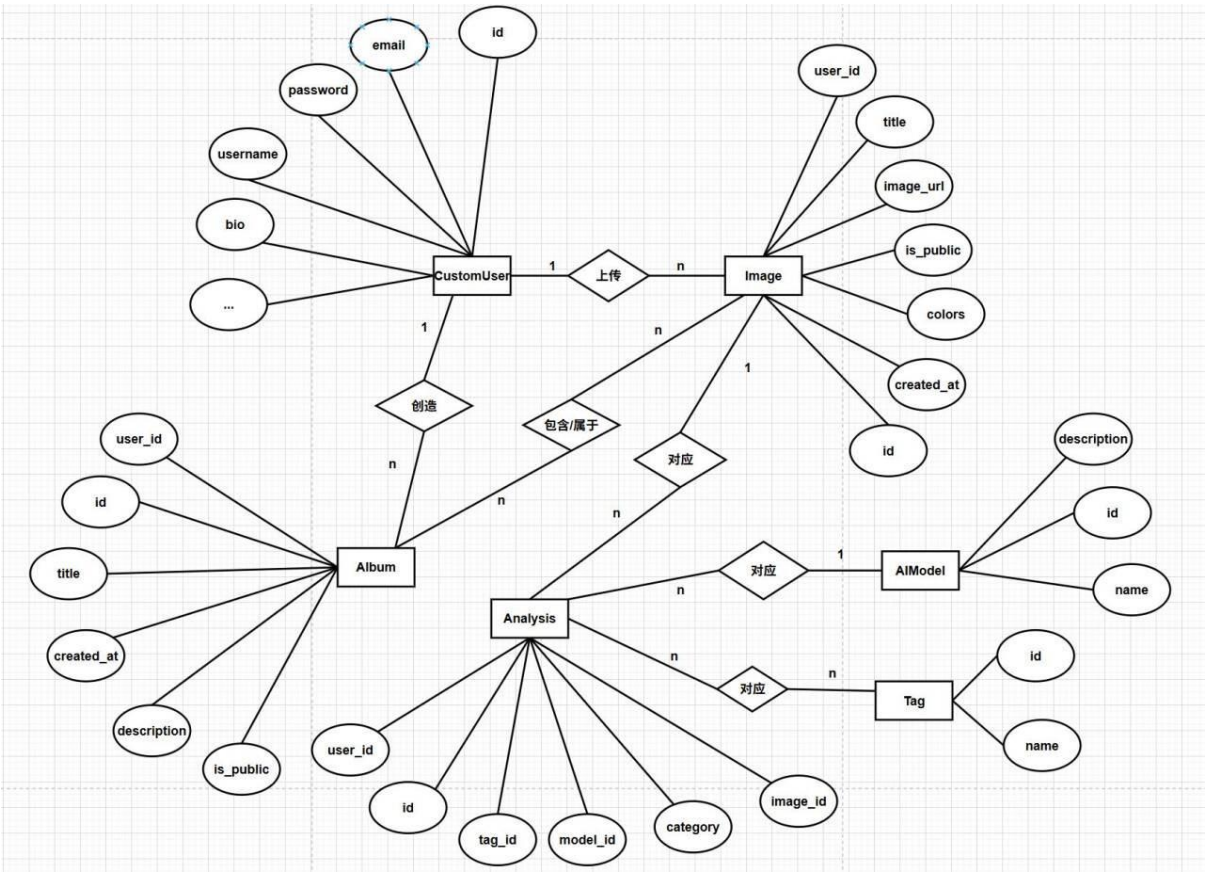
性能考量：在频繁查询的字段（如外键、标签名、用户 ID 等）上建立索引，以提高检索效率，满足图片搜索响应时间小于等于 2 秒的要求 。

数据与文件分离：图片、头像等二进制大文件将存储在云存储服务（如七牛云）中 ，数据库仅保存其访问 URL，以减小数据库体积，提升存取性能。

逻辑抽象：业务逻辑（如权限判断、社区分享逻辑）主要在应用层（Django）实现，数据库层面专注于数据的高效、安全存储。

二、内部依赖性描述

使用ER图描述实体间的关联依赖关系，分析对存取空间、性能、完整性的要求。



5.2 行为定义

一、存储过程/触发器分类与分解

按功能模块归类，详细定义如下：

类别	存储过程/触发器名称	功能描述	关键参数	返回值/记录集	依赖对象	特殊要求
用户管理	sp_user_register	处理用户注册逻辑	输入：@username, @password_hash, @email 输出：@user_id	注册状态码 (0:成功, 1:用户名重复, 2:邮箱重复)	Users 表	启用事务，原子性写入
	sp_user_login	验证登	输入：@identifier (用户名/邮箱), @password_hash	登录状态码 (0:成功, 1:凭证错误, 2:账户禁用)	Users, Sessions 表	JWT To

		录凭证并生成会话	输出：@auth_token，@user_id			ken 生成
	trg_user_audit	记录用户关键操作日志	触发事件：Users 表 UPDATE/DELETE	无	User_Audit_Logs 表	记录操作前/后数据
图片管理	sp_image_upload	写入图片元数据并触发 AI 分析	输入：@user_id，@image_url，@file_size 输出：@image_id	图片 ID	Images，AI_Tasks 表	事务内写入图片+AI 任务记录
	trg_image_delete	删除图片时清理云端文件	触发事件：Images 表 DELETE	无	七牛云存储 API	异步调用云存储删除接口

AI分析	sp_ai_tag_generation	调用AI服务存储标签结果	输入：@image_id 输出：@confidence_score	标签JSON数组([{"tag":"cat","score":0.92}])	Image_Tags, Images表	依赖外部AI服务API
相册管理	sp_album_create	创建相册并关联图片	输入：@user_id, @title, @image_ids（图片ID数组) 输出：@album_id	相册ID	Albums, Album_Images表	启用事务
	sp_album_update_cover	自动更新相册封面（首张图片）	触发事件：Album_Images表INSERT（排序首位变更）	无	Albums表	仅更新封面URL
社区模块	sp_get_public_albums	分页查询公开相册	输入：@page, @page_size, @sort_by 输出：相册ID/标题/封面/作者	分页的相册记录集	Albums, Users视图(v_public_albums)	响应时间≤1s

列
表

二、外部依赖性描述

云存储服务

依赖七牛云 SDK 的 delete_object 接口（触发器 trg_image_delete 调用）

图片上传由应用层直接调用云存储 API，不通过存储过程

AI 分析服务

存储过程 sp_ai_tag_generation 通过 HTTP 调用外部 AI 服务的 /analyze 接口

前端 API 约束

所有存储过程需返回结构化状态码（如 {code: 0, data: {...}}）供 RESTful API 封装

三、内部依赖性描述

数据表依赖

核心表 Images 被 sp_image_upload, trg_image_delete, sp_ai_tag_generation 依赖

相册操作依赖 Albums 与 Album_Images 的关联表

存储过程间依赖

sp_image_upload 执行后自动调用 sp_ai_tag_generation（通过事件队列）

sp_album_create 调用 sp_validate_image_ownership（验证图片归属）

视图依赖

v_public_albums 视图为社区模块提供预聚合数据（依赖 Albums + Users + Images 表）

关键约束说明

事务完整性

涉及多表写入的操作（如注册、相册创建）必须启用事务（例如 sp_user_register）。

性能优化

高频查询（如 sp_get_public_albums）需利用覆盖索引和视图缓存。

安全要求

所有存储过程启用最小权限原则，禁止动态 SQL 拼接（防 SQL 注入）。

错误处理

返回明确错误码（示例）：

0 ：成功

1xx ： 用户模块错误

2xx ： 图片模块错误

5xx ： 系统级错误

6. 原型或界面设计