目录

一、 引言

1.1 编写目的

本设计文档旨在基于需求分析，对系统的总体结构、功能模块划分、数据流与控制流、接口设计、数据库设计以及关键算法进行详细说明。文档的目的是为系统的开发、测试和维护提供统一的技术依据，确保开发团队对系统实现有一致的理解和明确的指导。

本设计文档面向系统开发人员、测试人员和项目管理人员，供其在后续开发阶段参考使用。

通过本设计文档，可以实现以下目标：

① 明确系统架构与模块划分，保证实现过程的结构化与可扩展性；
② 为数据库与接口设计提供依据，确保系统各部分协同一致；
③ 指导开发人员根据设计方案进行编码与集成；
④ 为测试阶段提供验证系统功能和性能的设计参考；
⑤ 为系统后期维护与功能扩展提供文档支持。

1.2 背景

中国生成式 AI 产业正处于高速发展阶段，生成式 AI 在多个行业中的应用正在迅速扩展。根据中国互联网络信息中心（CNNIC）发布的《生成式人工智能应用发展报告（2025）》显示，截至 2025 年 6 月，中国生成式 AI 用户规模已达 5.15 亿人，较半年前增加 2.66 亿，增幅高达 106.6%，普及率达到 36.5%。这一趋势表明，生成式 AI 技术已经不再是实验阶段，而是进入了大规模应用期，涵盖了内容创作、问答、办公、娱乐、教育等多个领域，正在逐步重塑人机交互和内容生产方式。

在此背景下，本项目的开发聚焦于生成式 AI 的核心能力——高效、精准的图片生成。通过调用领先的 AI 生成图片 API，系统能够帮助用户轻松创建图像内容，无论是创意设计、艺术创作还是商业用途，均能快速高效地完成。此外，本项目还将 NFT 功能作为前瞻性实验性功能进行集成，探索 AI 生成内容与数字资产的结合，旨在为用户提供一个创新的平台，让他们能够将生成的内容转化为独一无二的数字资产，进一步推动创作者与消费者之间的互动与价值转化。

当前，AIGC（AI Generated Content，生成式人工智能创作内容）生态环境日益多样化。国内外企业在基础 AI 模型上竞争激烈，像 OpenAI、Google、Runway 等国际公司持续引领技术发展，字节跳动、腾讯、快手等国内企业也通过平台创新实现生态竞争力。在这一竞争激烈的市场环境中，本项目将以“简洁易用”作为核心竞争力，优化用户体验，降低技术门槛，使普通用户无需专业知识即可通过简单的操作生成高质量的视觉内容。与此同时，结合 NFT 功能，系统将为专业创作者提供更具价值的创作工具，探索内容创作与数字资产的无缝连接。

随着用户需求从“技术尝鲜”转向“日常应用”，便捷性、操作体验以及创新性将成为产品竞争力的核心因素。本项目旨在打造一个操作直观、体验友好的 AI 内容创作工具，不仅满足日常内容创作的需求，还通过 NFT 功能的创新引领行业前沿，形成差异化优势，为用户提供更全面的创作与资产管理体验。

1.3 定义

在本系统设计文档中，以下术语和定义用于阐明系统中的关键概念与功能：

- 生成式 AI（Generative AI）：一种利用机器学习模型自动生成内容的技术，能够基于文本、图像或视频生成新的创意作品。本系统主要通过调用 AI 生成图片 API 来实现图像内容的自动生成。
- AIGC（AI Generated Content）：指由人工智能生成的内容，包括文本、图像、音频、视频等形式。AIGC 被广泛应用于创意内容的生成，极大地提升了内容创作的效率和质量。
- NFT（Non-Fungible Token，非同质化代币）：一种基于区块链技术的数字资产，具有唯一性和不可替代性。本项目中，NFT 用于将生成的内容（如图像和视频）转化为独一无二的数字资产，使用户能够购买、销售和收藏这些创作内容。
- API 接口（Application Programming Interface）：应用程序编程接口。系统通过 API 与外部 AI 生成模型（如腾讯混元、阿里千问等）进行交互，完成图像、视频等内容的生成任务。API 接口用于传递用户请求、处理生成任务以及返回生成结果。
- 用户管理模块：负责用户身份的验证与权限控制，包括用户注册、登录、密码找回和个人信息管理等功能，保障系统的安全性与用户数据隐私。
- 内容生成模块：系统的核心功能模块，负责接入各种 AI 生成模型的 API，处理用户输入的文本或图像生成请求，并将结果呈现给用户。该模块支持生成参数调整、进度监控和多模型结果对比。
- 结果管理模块：用于保存和管理用户生成的内容，包括作品分类、收藏、分享、导出等功能，帮助用户高效地组织和利用创作成果。
- 后台管理模块：为系统管理员提供全面的运维和监管能力，包括用户管理、内容审核、系统监控、API 使用统计等功能，确保系统的安全、稳定和合规运行。
- 任务 ID：每个生成任务在系统中都会分配一个唯一的任务标识符，用于标识和追踪生成任务的执行状态和结果。
- 生成任务：用户发起的内容生成请求，包括文本生成图像（T2I）、图像生成图像（I2I）、文本生成视频（T2V）等任务类型。
- 生成结果：由 AI 生成模型返回的图像或视频文件，用户可以在平台上查看、下载，或将其转换为 NFT 等数字资产。
- 用户权限：系统针对不同用户角色（如普通用户、专业创作者、管理员）设置的权限控制，确保各类用户只能访问与其身份匹配的功能和数据。
- 批量生成：针对专业创作者的需求，系统支持批量提交和管理多个生成任务的功能。
- 任务进度监控：系统提供实时的任务进度更新，用户可以查看每个生成任务的当前状态（如“处理中”、“完成”、“失败”）以及生成的结果。

1.4 参考资料

[1] 中国互联网络信息中心. 生成式人工智能应用发展报告（2025）[R]. 2025.

[2] Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach [M]. 9th ed. New York: McGraw-Hill, 2020.

二、任务概述

2.1 目标

本项目旨在开发一款面向普通用户与内容创作者的轻量级 AI 视觉内容生成客户端，核心功能是通过调用先进的生成式 AI 模型 API，提供一站式的高质量视觉内容生成服务。系统将以生成图像和视频为主，结合文本提示（Text-to-Image，Text-to-Video）或参考图像（Image-to-Image）等技术，帮助用户创作艺术作品或进行创意设计。

除了 AI 图像生成功能外，项目还在生成内容的基础上，探索将作品转化为 NFT 的创新功能。用户不仅可以生成个性化的内容，还能够将这些作品铸造成 NFT，进行数字收藏、交易和认证，进一步扩展创作的经济价值和社交价值。

项目的技术架构将采用高度模块化的设计，以保证系统的可扩展性和后续功能的灵活加入。客户端将支持接入多种主流 AI 生成模型 API，确保平台能够兼容并调用国内外先进的 AI 服务，如 OpenAI 的 DALL-E、字节跳动的 “即梦” 等，同时具备快速启动、低资源占用和流畅操作的特点。

系统的总体目标包括：

- 降低 AI 创作门槛：通过极简的交互界面、智能提示词辅助手段和模板化生成，帮助没有专业背景的用户快速上手并生成高质量的视觉内容。
- 满足多层次创作需求：为专业创作者提供批量生成、任务管理、作品分类、导出等高级功能，支持更高效的创作与管理。
- NFT 集成：为用户提供一个创新的平台，将生成的图片与视频作品转化为 NFT，支持作品的铸造、交易和数字所有权认证，探索数字创作与区块链技术的结合，提升平台的创新性与前瞻性。

项目的长期目标是以基础的 AI 生成功能为核心，逐步构建集素材管理、模板共享、创作者社区和 NFT 交易市场于一体的创新平台，持续提升产品的用户粘性与市场竞争力。

2.2 运行环境

- 客户端平台：Windows（基于 Python Qt 的桌面应用）
- 后端服务器：Windows（基于 Python Flask 的服务器应用）
- 数据库：PostgreSQL
- 接口环境：HTTPS（采用 SSL 加密的 HTTP 接口）
- 外部依赖：第三方 API（DALL-E、即梦等）

三、总体设计

3.1 软件总体设计

3.1.1 系统架构

软件总体设计方面，系统采用客户端-服务器（C/S）架构并按照功能划分为多个模块。客户端为 Windows 平台的 PC 应用（通过浏览器界面与用户交互），服务端为 Flask 架构的应用服务器，统一对外提供 RESTful API 服务。系统顶层的软件结构如图 7 所示，各主要功能模块通过明确的接口进行通信与协作。

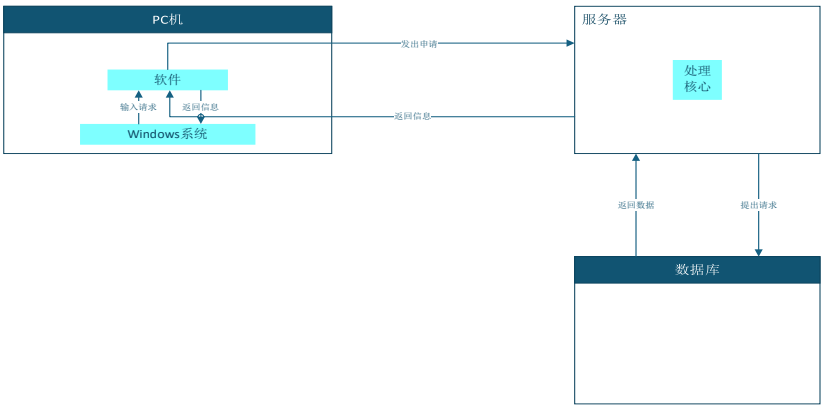

图 3.1 系统架构图

3.1.2 软件模块结构

软件系统将分为以下五大核心模块：

① 用户管理模块：负责用户身份验证、权限控制、账户管理等，确保系统的安全性和数据隐私。

② 内容生成模块：核心功能模块，负责 AI 模型 API 的集成和内容生成任务的执行，支持生成参数的调整、任务进度的实时监控及多模型结果对比。

③ 结果管理模块：用户生成内容的管理模块，以画廊形式支持作品的分类、收藏、搜索、分享与导出等功能，帮助用户高效地组织和复用创作成果。

④ 后台管理模块：为管理员提供用户管理、内容审核、系统监控和 API 使用统计等功能，保障系统的稳定性与运营安全。

⑤ NFT 管理模块：为用户提供生成内容的 NFT 铸造和管理功能，具体包括：

（1）NFT 铸造：将生成的图片、视频或创意内容转化为独一无二的 NFT，并进行区块链上注册，赋予其数字所有权和版权认证。

（2）NFT 交易：用户可以通过平台直接展示、买卖、拍卖自己的 NFT 作品，拓宽创作收益渠道。

（3）NFT 展示与社交功能：用户可以在平台内展示自己创作的 NFT 作品，并与其他创作者互动，参与虚拟展览、收藏活动等，增强平台的社交与互动性。

（4）数字资产管理：为用户提供 NFT 资产的管理功能，支持查询、转账、分享和存储等操作，提升用户的资产管理体验。

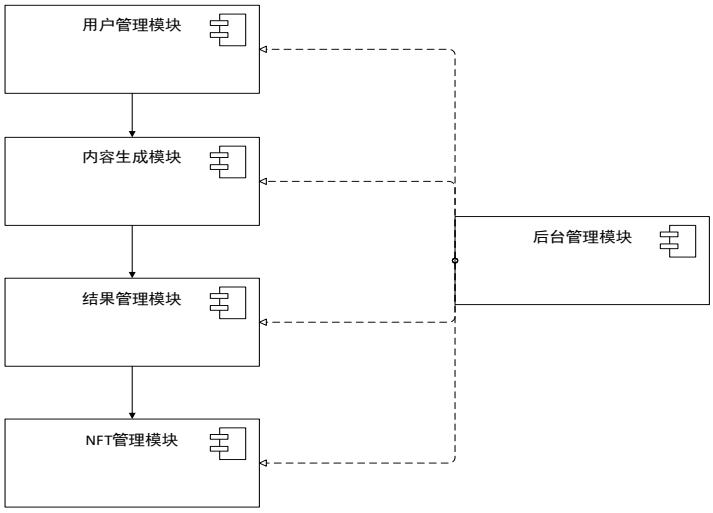

图 3.2 软件核心模块图

通过以上各模块的协同工作，本系统不仅能够为普通用户提供简便易用的 AI 创作工具，还能够为专业创作者提供强大的内容管理和 NFT 功能，推动平台向集数字艺术创作与交易于一体的综合平台发展。项目的长期目标是以核心的 AI 内容生成功能为基础，逐步构建集素材管理、模板共享、创作者社区和 NFT 交易市场于一体的创新平台，持续提升产品的用户黏性与市场竞争力。

3.2 硬件总体设计

在本系统大体可分为 2 部分，即客户端部分和云端部分。用户通过本地安装的客户端访问服务，请求经由互联网发送到云端的服务器，完成请求的任务，将相关记录存储入集中的云端的数据库中，并返回响应结果。部署描述如下表

节点	组件
用户 PC	桌面客户端应用程序(提供用户交互接口)
远程服务器	应用服务器（执行身份验证、内容生成等服务）
中心数据库	数据库管理系统

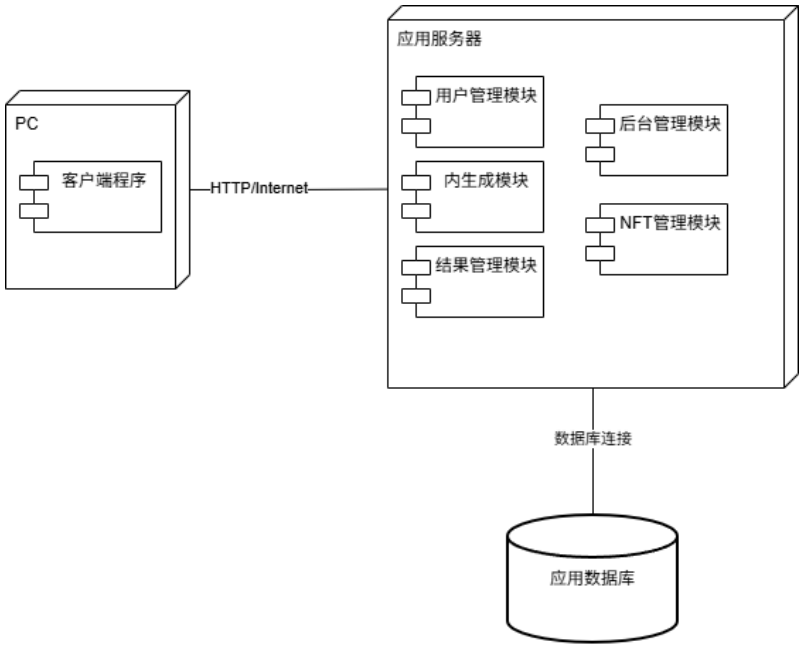

图 3.3 系统部署图

3.3 用户管理模块

3.3.1 结构描述

用户管理模块由用户接口、认证服务、用户仓储、令牌服务和邮件服务组成，它们共同支持注册、登录、身份验证、验证码发送和令牌管理等功能，外部依赖包括数据库和邮件服务器。

用户接口作为客户端的统一入口，负责接收请求并进行基础校验，再将处理交给内部核心服务，屏蔽了模块内部的复杂性。

认证服务承担主要业务逻辑，负责认证流程、注册管理、密码校验和验证码处理，并通过用户仓储、令牌服务和邮件服务完成数据存取、令牌管理和邮件发送。

用户仓储抽象了数据库访问，用于读取与保存用户信息。令牌服务负责生成与校验令牌，维护有效期与刷新策略。邮件服务负责与邮件服务器交互，发送验证码与通知邮件。

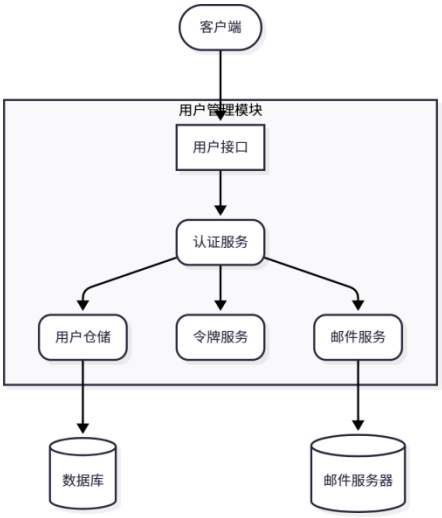

图 3.4 用户管理模块

3.3.2 功能描述

用户管理模块是系统的安全门户，负责处理所有与用户身份认证和账户管理相关的操作。其核心功能包括：

- 用户注册：提供新用户注册功能。用户填写用户名、邮箱和密码等信息创建账户，系统验证所填信息的唯一性和合规性。
- 用户登录：已注册用户通过用户名和密码进行身份验证，验证通过后系统建立授权会话（颁发 Token 凭证）。
- 忘记密码：用户忘记密码时，可通过注册邮箱接收验证码并重置密码，找回账户访问权限。
- 资料修改：用户登录后可以进入个人资料管理，修改账户邮箱、密码等信息，提交后系统验证并更新保存新的资料。

3.3.3 流程描述

3.3.3.1 用户注册流程描述： 注册流程如图 4 所示。用户在系统登录主界面点击“注册”按钮，系统跳转至注册界面。用户输入用户名、邮箱、密码等信息后提交注册请求。后端对提交信息进行唯一性和格式验证，验证通过则在数据库中新建用户记录，返回注册成功结果给前端界面。

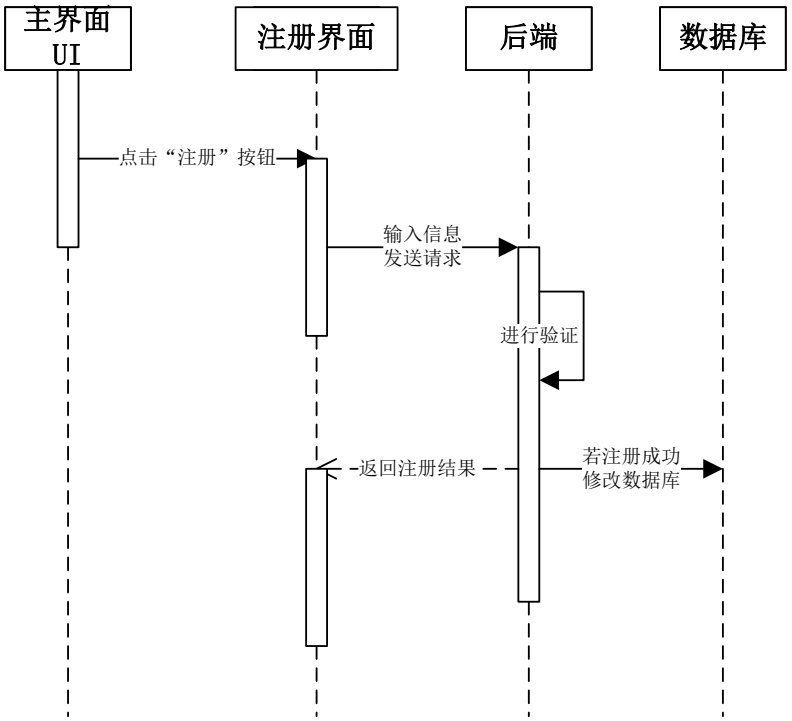

图 3.5 用户注册流程

3.3.3.2 用户登录流程描述： 登录流程如图 5 所示。用户在主界面的登录表单中输入用户名和密码后提交请求，后端读取数据库验证用户名和密码是否匹配。若验证通过则登录成功，进入系统内主界面；若失败则返回登录页面并提示错误。

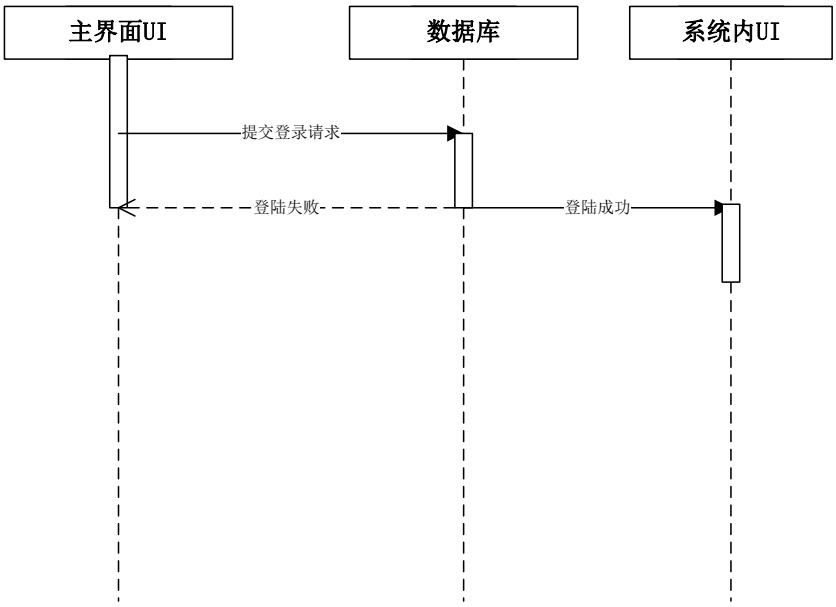

图 3.6 用户登录流程

3.3.3.3 找回密码流程描述： 找回密码流程如图 6 所示。用户在登录页面点击“忘记密码”按钮，进入找回密码界面并提交注册邮箱以获取验证码。系统发送验证码邮件后，用户填写验证码并设置新密码提交请求。后端验证验证码有效且匹配后，在数据库更新该用户的密码。找回密码成功后，系统返回结果，用户可返回登录页面使用新密码登录。

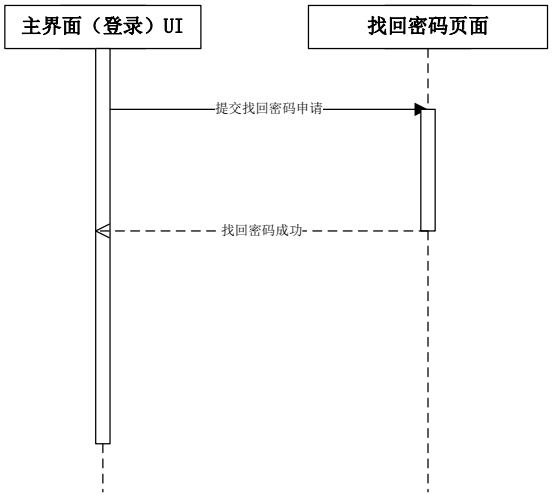

图 3.7 找回密码流程

3.3.3.4 资料修改流程描述： 资料修改流程如图 11 所示。用户登录系统后进入个人资料管理界面，编辑需要修改的信息（如邮箱或密码），点击“保存”提交修改请求。后端收到请求后对提交信息进行验证，验证通过则更新数据库中的用户信息，并返回修改结果。用户资料修改成功后，前端提示更新成功；如验证失败则返回错误提示。

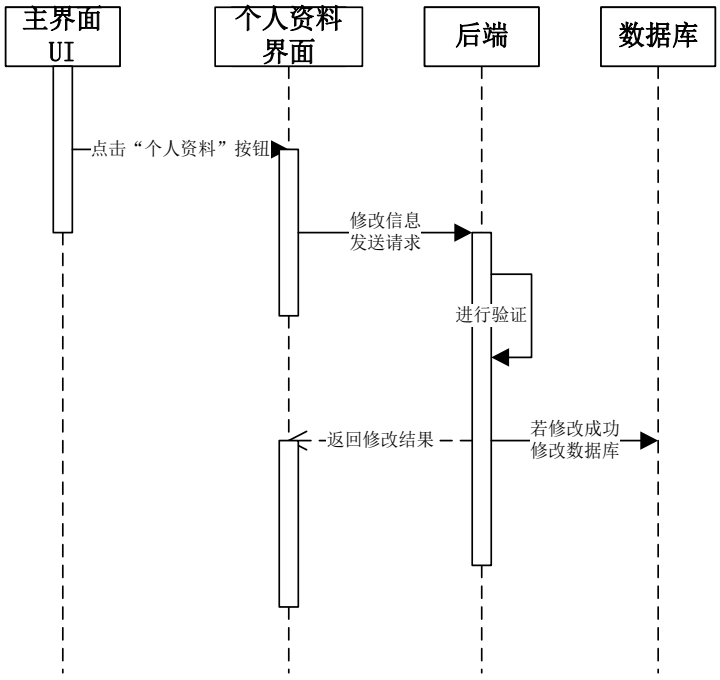

图 3.8 资料修改流程

3.4 内容生成模块

3.4.1 结构描述

内容生成模块由多个组件协作完成任务接收、调度、生成与结果记录。客户端的生成请求首先进入外部接口并触发生成流程，内部的任务流转、模型调用和数据存储等细节则在模块内部自动处理。

请求进入生成接口后由生成服务解析并构建任务，再将任务推入任务队列。任务队列负责统一调度，从各客户端接收任务并按照策略提供给后续流程。核心生成由工作进程执行，它从队列取出任务，根据任务类型依次完成生成阶段。

生成过程中，工作进程会使用多项内部能力。对象存储负责持久化生成过程或结果中需要保存的数据。AI 模型适配器根据任务需求选择合适的大模型并完成模型调用。生成完成后，记录仓储将相关过程与最终结果保存至数据库，便于展示、查询和审计。

通过任务队列、工作进程、模型适配与记录存储的协同，内容生成模块形成清晰的任务生命周期管理，使系统在并发任务和大模型调用场景下保持稳定可控。

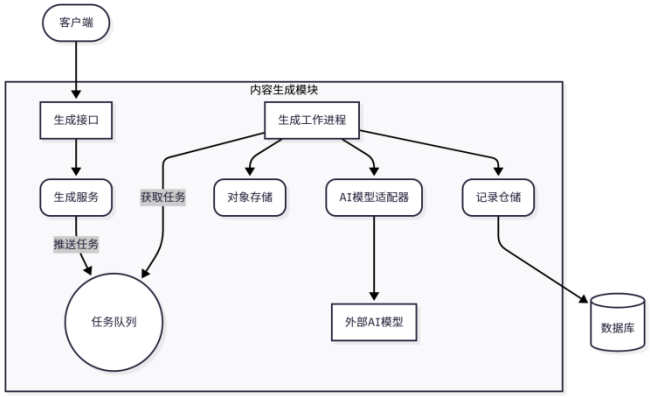

图 3.9 内容生成模块

3.4.2 功能描述

内容生成模块是系统的核心业务引擎，负责处理 AI 图片与视频的生成任务。其主要功能包括：

- 依据文字生成图片：

（1）任务发起：用户输入文本提示词（Prompt）描述期望的图像内容。

（2）异步处理：系统将生成任务提交至后台队列，避免前端界面阻塞，用户可实时查看任务排队进度。

（3）多模型集成：系统通过可扩展的框架接入并调用第三方大模型的图像生成 API，以支持不同风格或模型的生成。

（4）状态监控：对生成任务进行全生命周期管理（状态包括排队中、生成中、完成、失败），提供任务进度更新和结果预览。

- 依据参考图片生成图片：

（1）参考图上传与处理：用户上传一张参考图片，系统对该图片进行预处理（格式转换、尺寸调整、特征提取等），为后续生成提供视觉特征基础。

（2）混合提示生成：系统结合用户输入的文本提示词和参考图片提取的视觉特征，共同指导图像生成过程。

（3）风格迁移与内容重绘：支持基于参考图的风格迁移和内容重绘（在参考图的构图基础上进行修改或增强），生成与源图相关又有所变化的新图像。

- 依据文字生成视频：

（1）任务发起：用户输入文本，描述期望生成的视频场景或内容。

（2）异步处理：系统将视频生成任务加入后台异步队列，避免界面卡顿。

（3）模型调用：调用已集成的 AI 视频生成模型 API，将用户的文本描述作为参数传递给模型进行内容生成。

（4）结果获取：模型生成视频后，系统将视频文件保存至存储服务器，并更新任务状态为完成，同时提供生成视频的预览或下载链接。

- 依据关键帧生成视频：

（1）任务发起：用户输入文本和关键帧来描述期望生成的视频场景或内容。

（2）异步处理：系统将视频生成任务加入后台异步队列，避免界面卡顿。

（3）模型调用：调用已集成的 AI 视频生成模型 API，将用户的文本描述与关键帧作为参数传递给模型进行内容生成。

（4）结果获取：模型生成视频后，系统将视频文件保存至存储服务器，并更新任务状态为完成，同时提供生成视频的预览或下载链接。

3.4.3 流程描述

3.4.3.1 文生图流程描述： 文生图是指用户仅通过输入文本描述来驱动 AI 模型生成相应图像的过程，其流程如图 12 所示。用户在内容生成界面选择“文字生成图片”模式，输入描述希望生成内容的文本提示词，选择相关参数后提交任务。后端收到生成请求后，首先验证用户身份和使用配额，然后创建生成任务记录，状态置为“排队中”，并将任务加入异步队列等待处理。任务队列的工作进程取出该任务后，状态更新为“生成中”，随后调用选定的图像生成模型 API，将用户的提示词和参数发送给模型处理。模型返回生成的图片后，系统将图片文件保存至存储服务器，更新任务状态为“完成”，记录生成的图片访问地址，并将结果通过查询接口提供给前端显示预览。

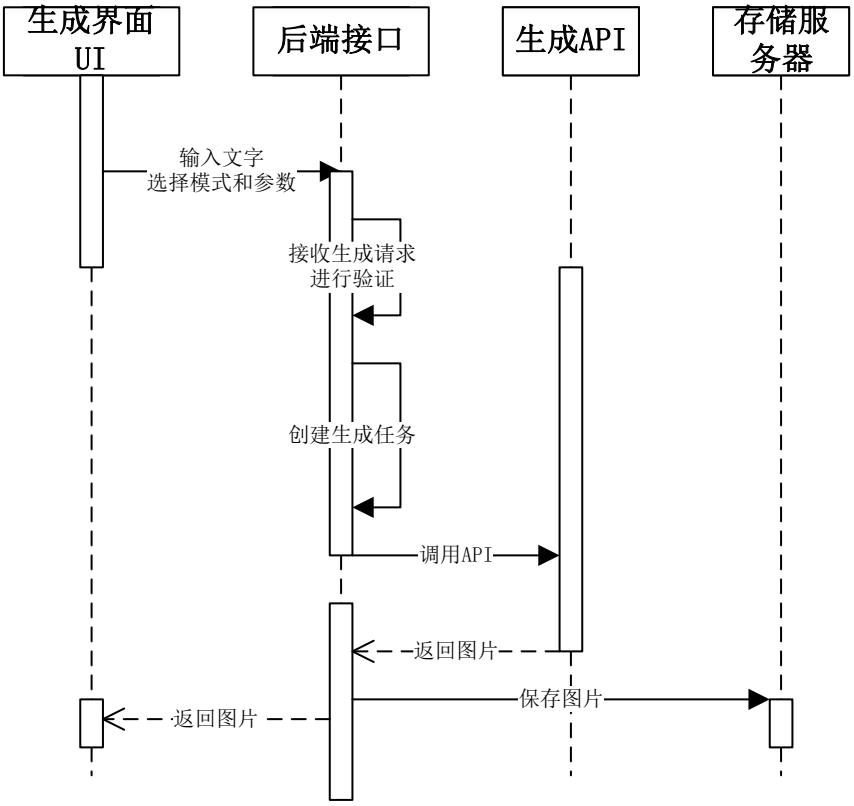

图 3.10 文生图流程

3.4.3.2 图生图流程描述： 图生图是指用户上传一张参考图片并结合文本描述，引导 AI 模型在参考图的基础上进行风格或内容的转换和再创造，其流程如图 13 所示。用户在内容生成界面选择“图生图”模式，上传参考图片，在文本输入框中描述希望生成的效果，选择参数后提交任务。后端收到请求后，进行用户身份和配额验证，通过后创建生成任务（状态“排队中”），并加入后台队列。队列处理进程取出任务后将状态改为“生成中”，然后调用选定的图像生成模型 API，发送用户的文本提示和参考图参数执行生成。模型返回生成的图片后，系统保存结果至存储服务器，将任务状态更新为“完成”，记录生成图片的访问地址，前端随后可获取并展示新图片。

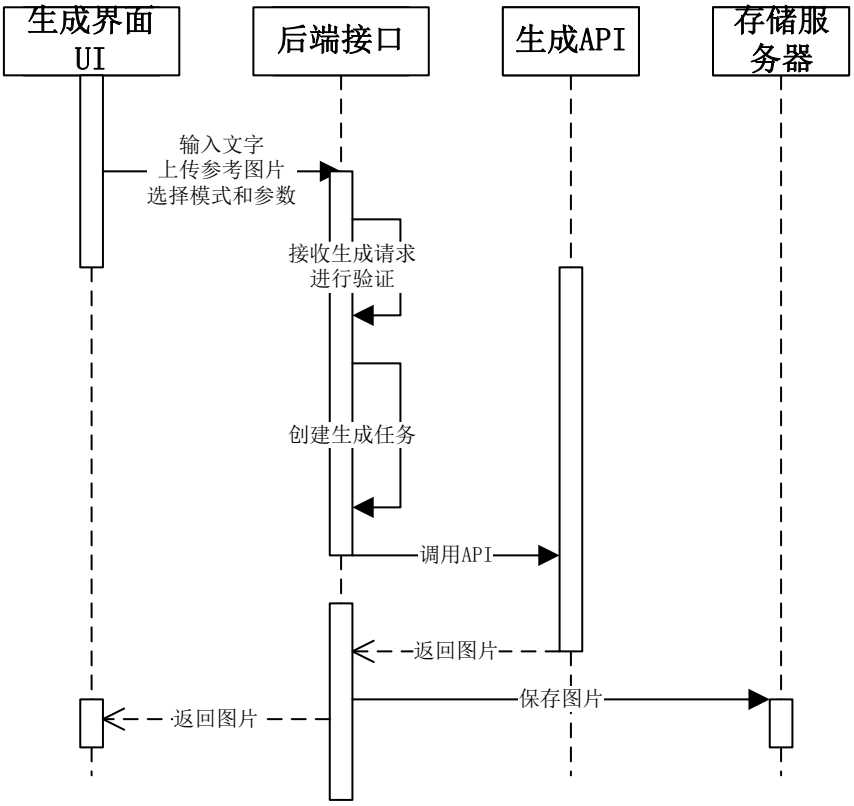

图 3.11 图生图流程

3.4.3.3 文生视频流程描述： 文生视频是指用户通过输入文本描述来生成对应视频内容的过程，其流程与文生图类似，如图 14 所示。用户在内容生成界面选择“文字生成视频”模式，输入希望生成的视频场景描述并设置相关参数（如期望时长、分辨率等）后提交任务。后端验证用户权限和配额后创建任务（状态“排队中”），将其加入异步任务队列。队列取出任务后置状态为“生成中”，调用对接的 AI 视频生成模型 API，将文本描述和参数发送至模型进行视频渲染生成。模型完成后返回生成的视频文件，系统将视频保存到存储，并更新任务状态为“完成”，记录视频文件的访问路径。用户随后可以通过前端提供的链接预览或下载生成的视频内容。

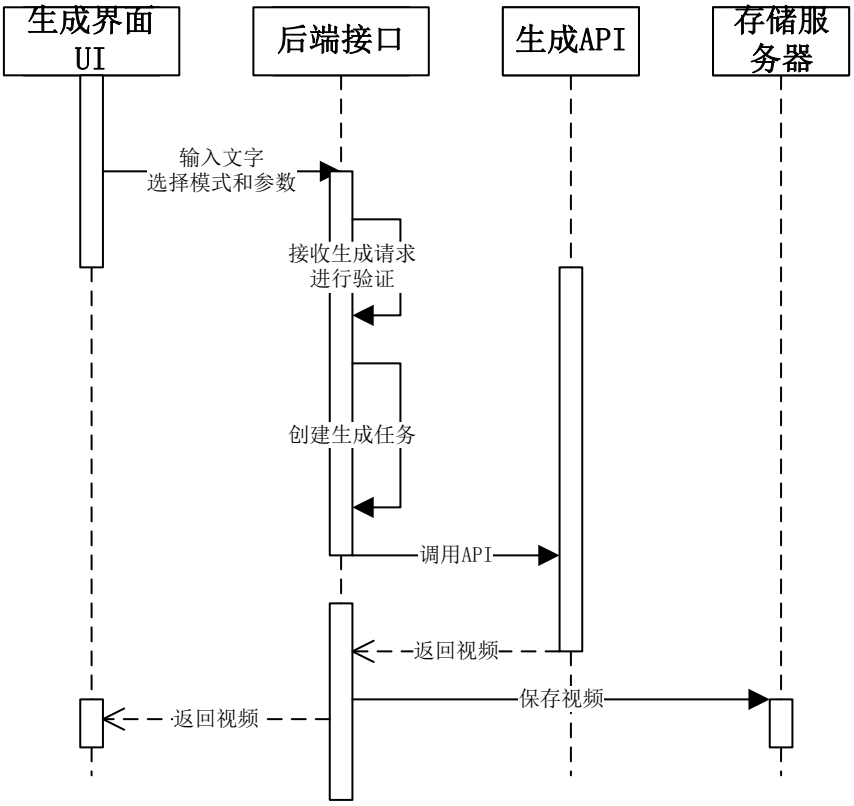

图 3.12 文生视频流程

3.5 结果管理模块

3.5.1 结构描述

结果管理模块由负责结果访问、权限校验和数据持久化的组件组成，客户端通过该模块获取生成任务的内容或结构化信息。流程从权限中间件开始，用于在请求进入模块前完成访问控制，只允许已验证用户继续操作。通过校验后，请求进入结果接口，由其将外部请求转化为内部可处理的形式。

核心处理由结果服务承担，它负责读取、整理并返回结果数据，并根据请求参数执行结果筛选、分页或目录组织等逻辑。结果服务不直接访问数据库，而依赖下层存储组件完成数据获取与保存。

结果数据存放在两个仓储中。记录仓储保存可展示或下载的生成内容及相关记录；目录仓储保存结果的组织结构与元数据，如分组与层级信息。二者都与数据库相连，为结果服务提供持久化数据支持。

通过权限校验、结果接口、结果服务与底层仓储的配合，结果管理模块完成从请求进入到结果返回的完整流程，在确保访问控制的同时持续提供稳定、清晰的结果访问能力。

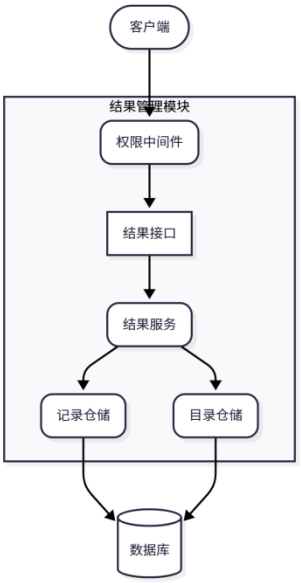

图 3.13 结果管理模块

3.5.2 功能描述

结果管理模块负责对用户生成的内容进行持久化存储、组织和管理，旨在提升用户的素材管理效率。其核心功能包括：

- 生成记录查看：以时间线或画廊形式展示用户的所有历史生成结果，支持按时间、模型、关键词等进行筛选和搜索，方便用户浏览检索。
- 删除生成结果：允许用户删除不再需要的生成结果，以释放存储空间并保持生成记录列表的整洁。
- 树状文件夹管理：用户可创建、删除多级文件夹，并将生成结果移动至目标文件夹或移出文件夹，从而实现灵活的个性化分类管理。

3.5.3 流程描述

3.5.3.1 生成记录查看流程描述： 生成记录查看是用户管理其 AI 创作成果的核心入口，其流程如图 15 所示。用户从系统主界面的导航菜单进入“生成记录”页面，前端通过调用获取记录列表接口拉取该用户的所有历史生成结果数据。系统以时间线或缩略图画廊的方式集中展示结果列表，用户可以滚动浏览并通过筛选条件快速查找特定记录。用户点击某一条生成结果，可查看其详细信息及相关操作选项。

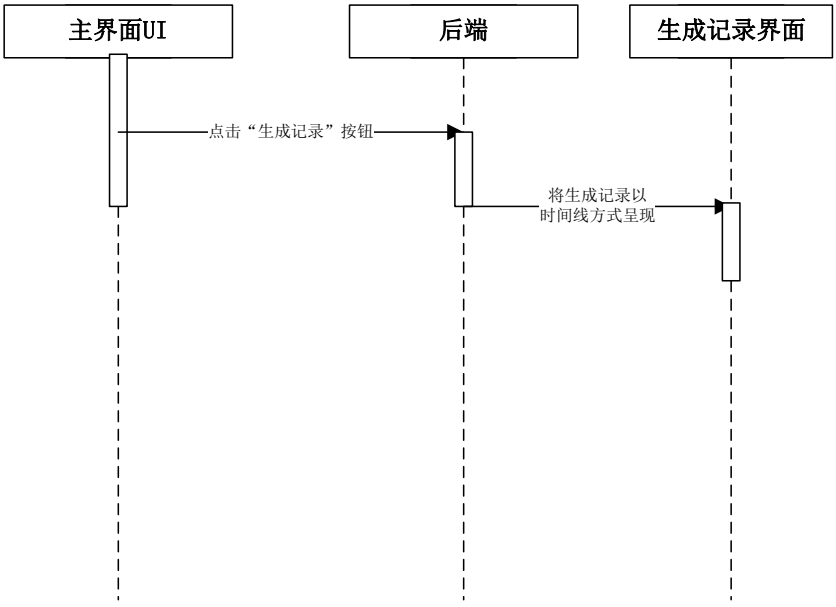

图 3.14 生成记录查看流程

3.5.3.2 删除生成记录流程描述： 删除生成结果的流程如图 16 所示。用户在生成记录列表或结果详情页面中勾选一条或多条想要删除的记录，点击“删除”按钮。前端弹出确认对话框提示此操作不可恢复，要求用户再次确认。用户确认后，前端调用删除结果接口请求后端执行删除。后端首先校验当前请求者对这些记录是否有删除权限（例如记录属于该用户本人），验证通过后在数据库将对应记录状态标记为已删除或移除记录。删除成功后，后端返回操作成功的响应。前端接收到响应后，从当前列表中移除被删除的记录项，并提示“删除成功”。

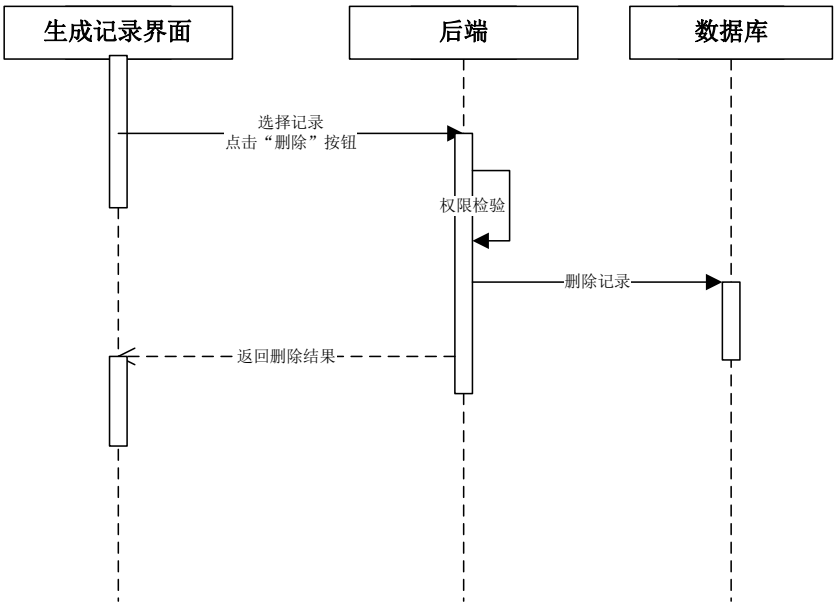

图 3.15 删除生成记录流程

3.5.3.3 树状文件夹管理功能： 树状文件夹管理为用户提供强大的内容组织工具，允许用户通过创建多级文件夹结构来自定义分类和收藏生成结果，实现条理清晰、易于检索的个人作品库。其主要子流程包括：

- A. 创建文件夹流程： 用户在生成记录管理界面点击“新建文件夹”按钮，系统弹出输入框让用户输入新文件夹名称。提交后，后端校验名称合法性（非空且当前用户下不重复），然后在数据库中创建新的文件夹记录。创建成功后，新文件夹立即出现在前端文件夹树中，用户可在其中组织作品。

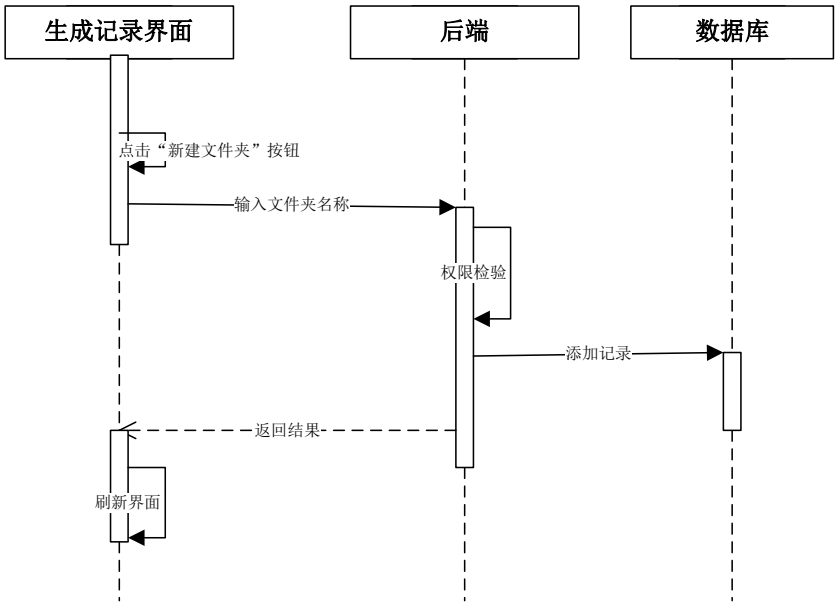

图 3.16 创建文件夹流程

- B. 将生成结果移入文件夹流程： 用户在生成记录列表中勾选一个或多个结果，然后选择“移动到...”操作。系统弹出当前用户的文件夹树供其选择目标文件夹。用户选择目标文件夹并确认后，前端调用更新记录接口请求后端执行移动操作。后端校验用户权限且目标文件夹存在后，更新数据库中对应记录的文件夹关联关系。操作成功后，后端返回结果，前端刷新界面，在目标文件夹下显示这些移动后的结果。

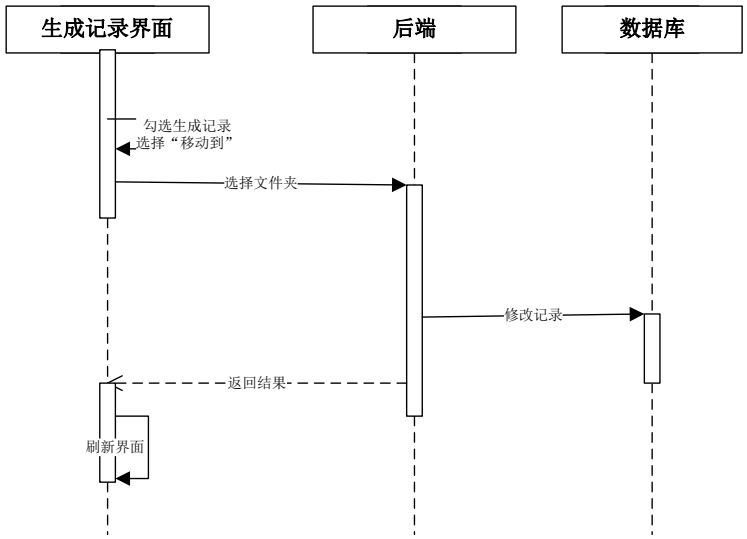

图 3.17 生成结果移入文件夹流程

- C. 删除文件夹流程： 用户在文件夹树上右键点击某个自建文件夹，选择“删除”操作。系统弹出确认对话框提示将删除该文件夹以及将其中作品移出。用户确认后，前端调用删除文件夹接口请求后端执行删除。后端验证用户权限后，在数据库删除该文件夹记录，并将该文件夹内所有生成结果的归属关系移除（通常移到默认根目录或未分类状态）。删除成功后，后端返回结果，前端从文件夹树中移除该文件夹节点。

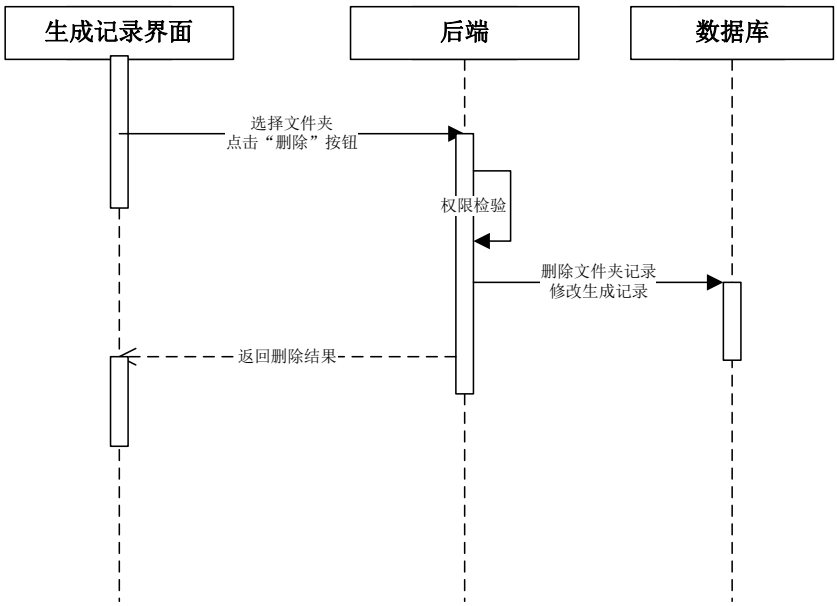

图 3.18 删除文件夹流程

3.6 后台管理模块

3.6.1 结构描述

后台管理模块由负责系统管理、数据维护与运行监控的组件组成，为管理端提供对系统状态与用户数据的操作能力。所有管理请求首先经过管理员中间件进行身份与权限校验，通过后由管理接口接收并转化为内部可执行的指令。

模块内部主要依赖两类服务。管理服务处理用户相关的管理操作，如信息维护与状态调整，并通过用户仓储读写数据库以保持数据一致性。统计服务则面向系统运行情况，通过访问日志库获取数据，用于生成各类统计结果与运行概况；日志库持久化系统运行中产生的日志与统计数据，为统计服务提供基础。

通过管理员中间件、管理接口、管理服务、统计服务及相应仓储的配合，后台管理模块构建了完整的管理流程，使系统能够在安全控制下持续提供数据维护与运行监控能力，并保持结构清晰可扩展。

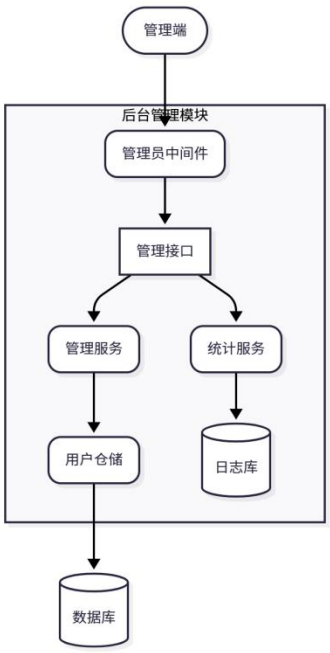

图 3.19 后台管理模块

3.6.2 功能描述

后台管理模块为系统管理员提供全面的运维管理功能，确保系统安全、稳定、合规运行。核心功能包括：

- 用户管理：管理员可以查看所有注册用户列表，并对违规用户执行封禁或解封等操作。

- 内容审核：对用户生成的内容进行审核，过滤违规或不当的图片/视频，确保平台内容合规。
- 系统监控与统计：监控系统性能指标，统计各模型 API 的调用次数、成功率、失败率等关键数据，为容量规划和异常预警提供依据。
- API 使用统计：汇总统计系统各功能模块的 API 使用情况，帮助管理员了解系统负载和用户使用偏好。

3.6.3 流程描述

3.6.3.1 管理员封禁用户： 管理员封禁用户的流程如图 21 所示。管理员登录后台后进入用户管理界面，选中目标用户并点击“封禁”按钮。前端调用封禁用户接口请求后端。后端校验请求者的管理员权限，通过后在数据库将所选用户的状态字段更新为“封禁”，然后返回操作结果。前端收到成功响应后，刷新用户列表状态，并提示封禁成功。

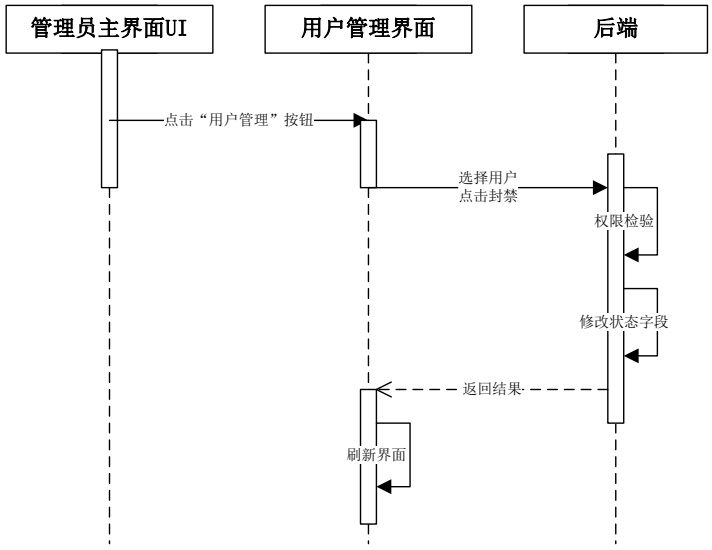

图 3.20 管理员封禁用户流程

3.6.3.2 管理员解封用户： 管理员为用户解封的流程如图 22 所示。管理员在后台用户管理界面选中已封禁的用户，点击“解封”按钮。系统流程与封禁类似：前端调用解封接口 -> 后端验证管理员权限 -> 更新数据库中用户状态为“正常” -> 返回成功响应。前端随后更新列表显示并提示用户状态已解封。

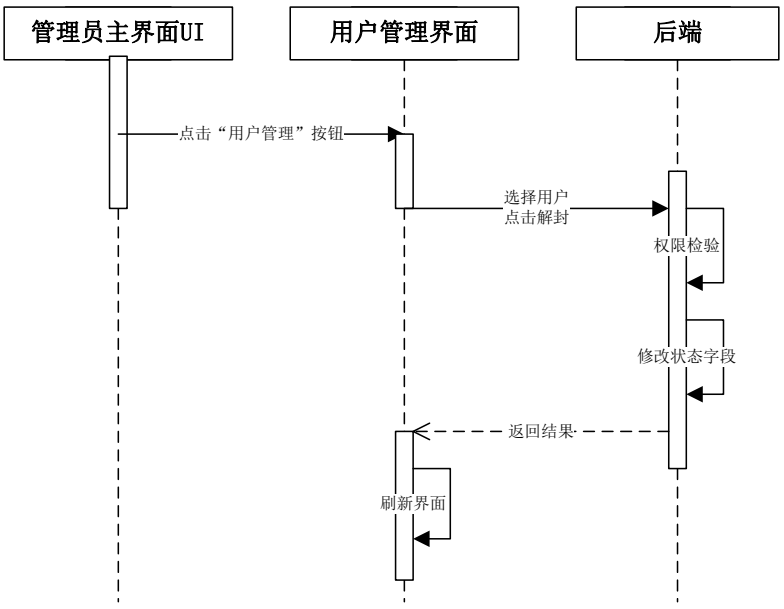

图 3.21 管理员解封用户流程

3.6.3.3 系统监控与统计： 系统监控与统计功能流程如图 23 所示。管理员在后台选择“系统监控”功能后，系统展示统计查询面板。管理员设置查询条件（如时间范围）后点击“查询”按钮，前端调用获取统计数据接口请求后端。后端服务根据条件从日志数据库或监控存储中检索原始 API 调用记录，并进行聚合计算得到统计结果数据，返回给前端。前端将统计结果在系统监控界面上以图表或表格形式展示，包括系统运行性能指标和各模型 API 的使用情况统计，帮助管理员直观了解系统负载和性能。管理员据此进行性能优化、资源规划或异常情况预警。

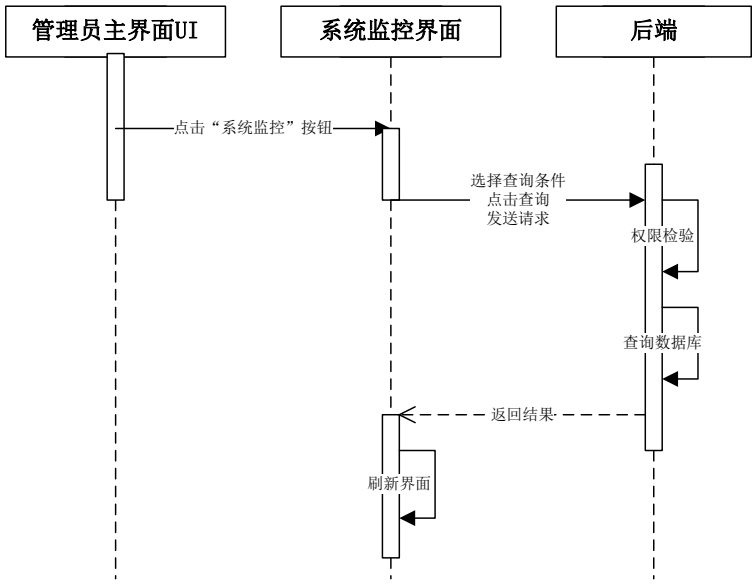

图 3.22 系统监控与统计流程

3.7 NFT 管理模块

3.7.1 结构描述

NFT 管理模块由多组组件协作完成客户端的 NFT 操作，包括内容存储、链上交互及资产记录管理。客户端请求首先进入 NFT 接口，由其解析参数并引导进入内部业务流程。

核心处理由铸造服务负责，它统筹从内容上传到最终上链的全过程，并调用多个子服务完成具体任务。IPFS 服务负责将内容存入 IPFS 网络并返回哈希。区块链服务通过智能合约实现 NFT 创建与元数据绑定等链上操作。NFT 仓储则记录链上标识、元数据位置及用户关联信息，并将其持久化到数据库中。

依托 IPFS 网络、区块链网络与本地数据库，NFT 管理模块构建了从内容上传到链上铸造再到资产记录的完整流程，使系统能够以稳定、规范的方式为客户端提供 NFT 服务。

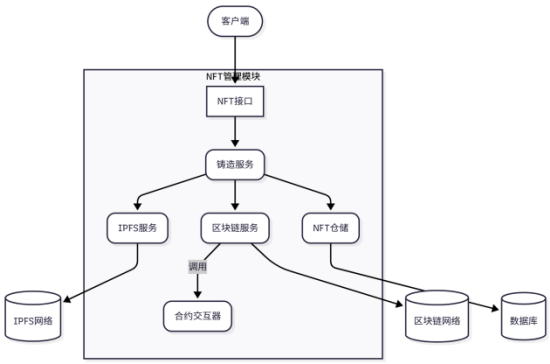

图 3.23 NFT 管理模块

3.7.1 功能描述

NFT 管理模块是系统的增值服务模块，为用户提供 AI 生成内容的区块链确权与资产化能力。主要功能包括：

- 生成结果上链确权：用户可将选中的生成结果（图片或视频）通过区块链进行上链登记，获得不可篡改的数字版权证明。
- 去中心化存储：系统将文件上传至去中心化存储网络（如 IPFS），获取唯一的内容标识符（CID），确保原始作品内容的长期保存和不可篡改。
- 区块链记录：通过智能合约将文件 CID、作者、生成时间等元数据记录在区块链（如 Polygon 等）上，生成对应 NFT 的唯一标识（Token ID）和交易记录。
- NFT 交易：支持将上链确权的作品与外部 NFT 市场对接，用户可以在第三方平台（如 OpenSea 测试网）查看和交易自己的 NFT 作品。

3.7.2 流程描述

3.7.2.1 上链确权流程描述： NFT 上链确权流程较为复杂，涉及用户界面、应用服务器、去中心化存储网络和区块链节点的交互，其流程如图 20 所示。用户在“生成记录”界面选择希

望确权的作品（图片或视频），点击“上链确权”按钮。系统弹出确认对话框，展示本次上链操作的摘要信息，包括作品预览及文件名、预估 Gas 费用（使用区块链原生代币计价，如 MATIC）、以及即将写入链上的元数据摘要（如部分提示词、使用模型名称、生成时间等）。用户核对信息无误后点击“确认”。

随后，后端服务接收到用户确认指令，首先计算生成结果文件的哈希值，并将文件内容上传至 IPFS 去中心化存储网络。上传成功后，IPFS 返回该文件的唯一内容标识符 CID。接着，系统自动构建包含作品名称、描述、以及指向该文件 CID 的引用等信息的 NFT 元数据文件（JSON 格式），并再次上传至 IPFS，获得元数据文件的 CID。

然后，系统通过区块链 API 在 Polygon 的 Mumbai 测试网络上发起 NFT 铸造交易，将作品的 CID（内容地址）、作者地址、生成时间等信息通过智能合约写入区块链，完成 NFT 的创建。交易提交后返回交易哈希、合约地址和生成的 Token ID 等信息。当区块链网络确认交易后，作品的链上所有权记录被永久保存且不可篡改。

最后，系统将整个流程涉及的关键数据（包括作品文件 CID、元数据 CID、交易哈希、合约地址、Token ID、所属用户等）写入本地数据库的 NFT 资产记录，并在前端画廊界面对该作品标注已确权的标识。用户可以通过提供的 Polygon 区块链浏览器（PolygonScan）链接及 OpenSea 测试网链接查询验证该 NFT 资产的详情。至此，实现了从内容生成、哈希存证、上链确权到展示交易的完整闭环。

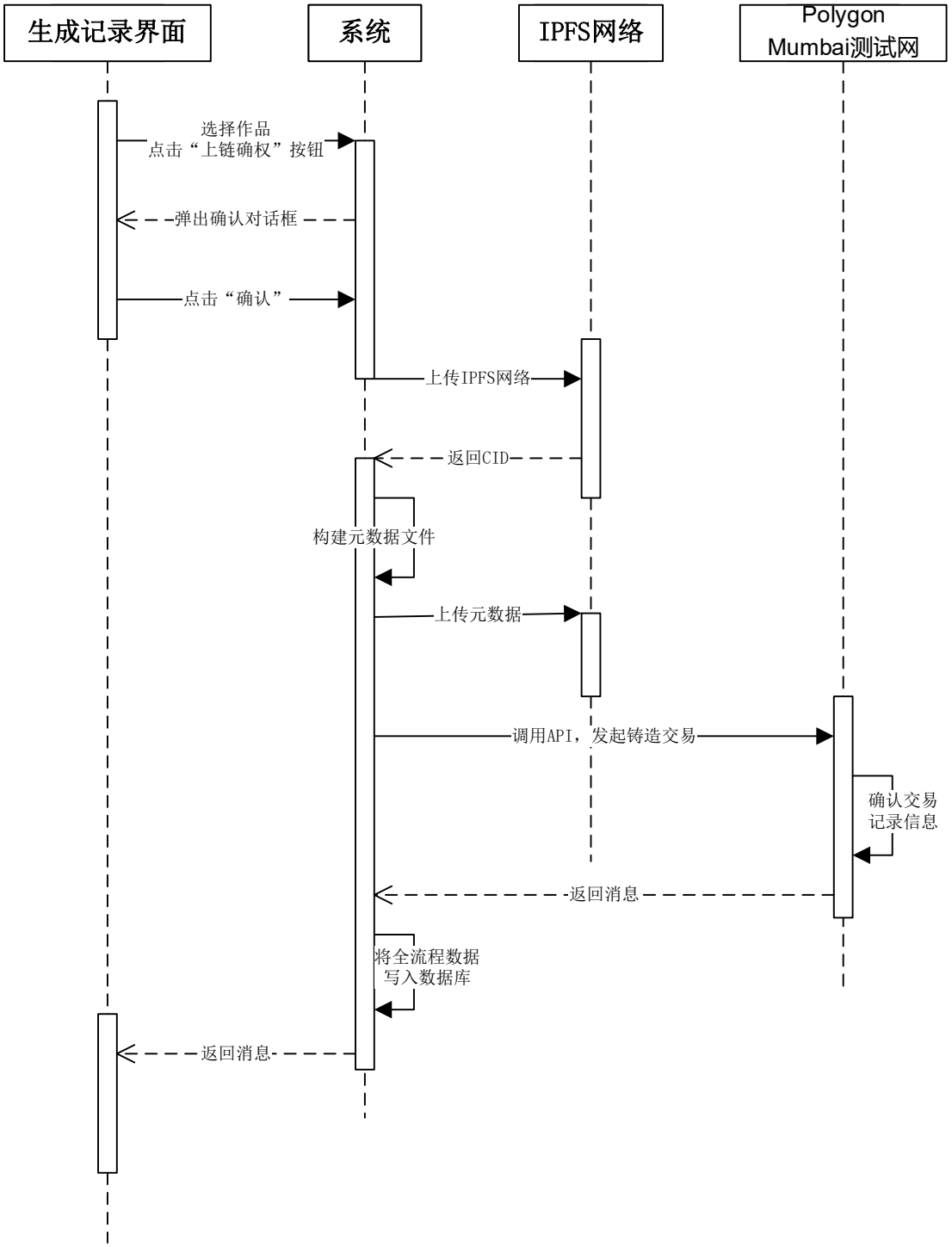
生成记录界面
系统
IPFS网络
Polygon
Mumbai测试网
选择作品
点击“上链确权”按钮
弹出确认对话框
点击“确认”
上传IPFS网络
返回CID
构建元数据文件
上传元数据
调用API，发起铸造交易
确认交易
记录信息
返回消息
将全流程数据
写入数据库
返回消息

图 3.24 上链确权流程

四、数据设计

4.1 设计概述与核心原则

系统的数据设计概况参考 ER 图。

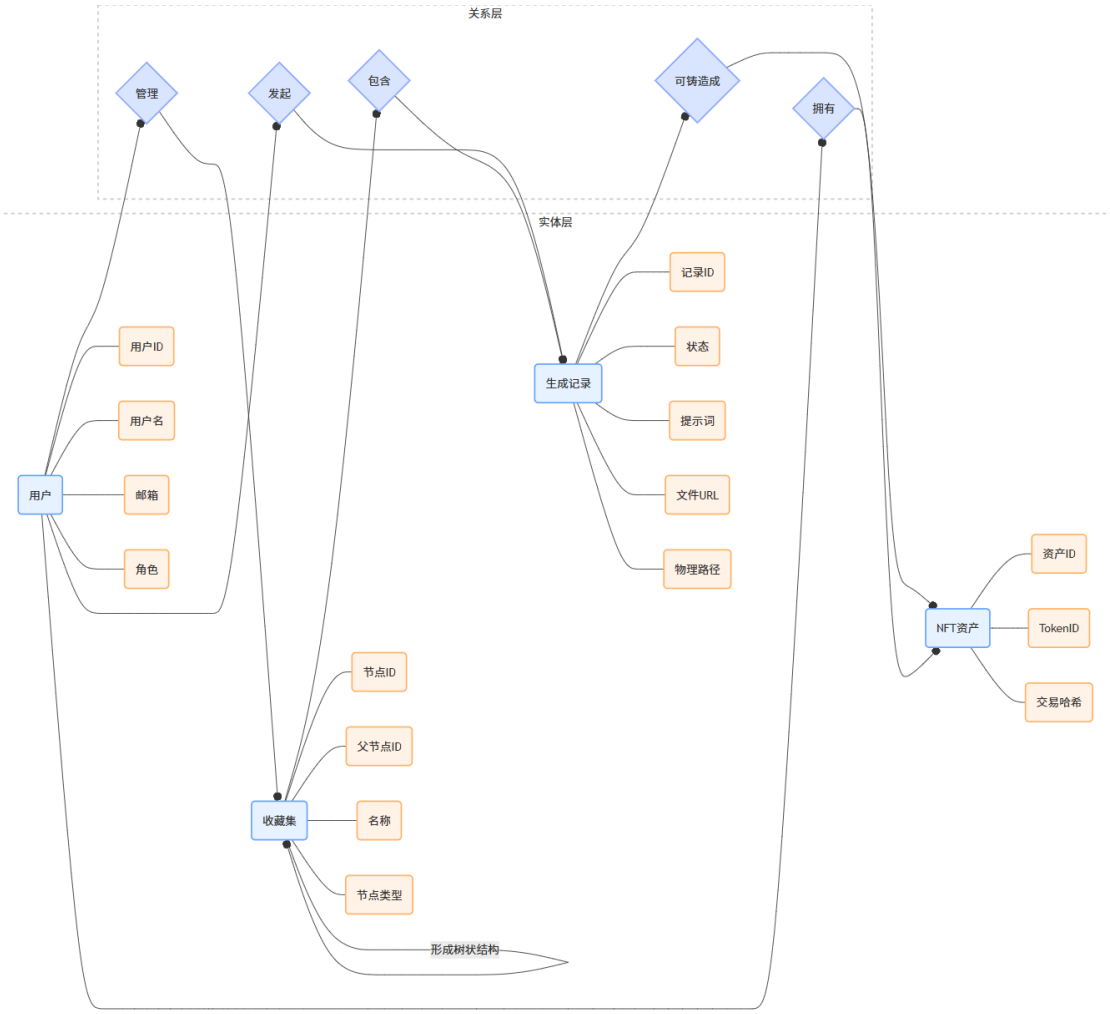

图 4.1 数据 ER 图

本数据库设计遵循以下核心原则：

① 过程与结果统一：将“生成任务”（一个动态过程）与“生成作品”（一个静态结果）合并到单一的 generations 表中进行管理。通过 status 字段追踪其完整的生命周期，简化了查询逻辑并减少了表的数量。

② 物理与逻辑分离：generations 表负责记录生成结果的物理路径（在服务器上的实际位置）和网络 URL。而 collections 表则负责构建用户可见的逻辑树形结构（文件夹/文件），用户对文件的移动、重命名等操作仅影响逻辑层，与物理存储解耦，极大提升了操作性能和灵活性。

③ 数据安全：用户密码等敏感信息采用不可逆的哈希算法加盐存储。所有与用户资产相关的操作均通过外键约束保证数据完整性。

④ 标识符规范：对外暴露的业务 ID（如生成记录 ID）采用 UUID 等无序字符串，隐藏数据库内部的自增主键，增强系统的安全性与扩展性。

4.2 实体表结构定义

本系统采用关系型数据库存储和管理应用数据。数据库设计遵循第三范式，确保数据的一致性和完整性。主要数据表包括用户表(users)、生成记录表(generations)、收藏集表(collections)和 NFT 资产表(nfts)。

参照完整性简介：

① 一个用户(users)可以拥有多条生成记录(generations)和多个收藏集节点(collections)。

② 一个收藏集节点(collections)如果是“文件”类型，则指向一条唯一的生成记录(generations)。

③ 一条生成记录(generations)可以被铸造成一个 NFT 资产(nfts)。

4.2.1 用户表 (Users)

说明：存储平台用户的核心账户信息、认证凭证、角色和状态。这是系统中所有与用户相关联数据的根源。

字段名	类型	说明	约束
id	BIGINT UNSIGNED	用户唯一 ID	主键，自增
username	VARCHAR(50)	用户登录名	不为空，唯一
email	VARCHAR(255)	用户邮箱，用于通知和找回密码	不为空，唯一
password_hash	VARCHAR(255)	使用 bcrypt 或 Argon2 哈希加盐后的密码	不为空
role	ENUM('user', 'admin')	用户角色	不为空，默认 'user'
status	ENUM('active', 'banned')	账户状态	不为空，默认 'active'
created_at	TIMESTAMP	注册时间	不为空，默认当前时间
updated_at	TIMESTAMP	账户信息最后更新时间	不为空，默认当前时间，自动更新

表 4.1 用户表

4.2.2 生成记录表 (generations)

说明：核心业务表，记录用户发起的每一次内容生成请求。它追踪任务从排队到完成的完整生命周期，并存储最终生成作品的访问路径。

字段名	类型	说明	约束
id	BIGINT UNSIGNED	内部唯一 ID	主键，自增
uuid	VARCHAR(36)	业务唯一 ID (UUID)，用于前后端通信	不为空，唯一
user_id	BIGINT UNSIGNED	发起任务的用户 ID	外键，引用 users(id)
status	ENUM('queued', 'processing', 'completed', 'failed')	任务状态	不为空，默认 'queued'
generation_type	ENUM('t2i', 'i2i', 't2v', 'i2v')	生成类型：文生图、图生图等	不为空
prompt	TEXT	用户输入的文本提示词	可为空
parameters	JSON	其他生成参数，如风格、分辨率等	可为空
result_url	VARCHAR(512)	生成结果的网络 URL	可为空 (成功后填写)
physical_path	VARCHAR(512)	生成结果的服务器物理路径	可为空 (成功后填写)

表 4.2 生成记录表

4.2.3 收藏集表 (collections)

说明：用于构建用户专属的、可视化的树形文件/文件夹结构。此表完全是逻辑层，与文件的物理存储位置无关。

字段名	类型	说明	约束
id	BIGINT UNSIGNED	文件夹唯一 ID	主键，自增
user_id	BIGINT UNSIGNED	节点所属的用户 ID	外键，引用 users(id)
parent_id	BIGINT UNSIGNED	父节点 ID，自引用 collections(id)	可为空（根节点为 NULL）
name	VARCHAR(100)	节点名称（文件夹名或文件名）	不为空
node_type	ENUM('folder', 'file')	节点类型	不为空

表 4.3 收藏集表

4.2.4 NFT 资产表（NFTs）

说明：记录已在区块链上完成铸造(Mint)的作品信息，将用户的链下数字作品与链上资产进行关联。

字段名	类型	说明	约束
id	BIGINT UNSIGNED	资产记录唯一 ID	主键，自增
generation_id	BIGINT UNSIGNED	关联的站内生成记录 ID	外键，引用 generations(id)，唯一
owner_user_id	BIGINT UNSIGNED	当前站内拥有者用户 ID	外键，引用 users(id)
token_id	VARCHAR(255)	NFT 在链上的唯一 Token ID	铸造成功后记录
contract_address	VARCHAR(255)	NFT 智能合约的链上地址	不为空
transaction_hash	VARCHAR(255)	铸造交易的哈希值	交易提交后记录，唯一
status	ENUM('pending', 'confirmed', 'failed')	铸造状态	不为空，默认 'pending'

created_at	TIMESTAMP	提交铸造请求的时间	不为空，默认当前时间
confirmed_at	TIMESTAMP	链上确认时间	交易确认后记录

表 4.4 NFT 资产表

4.3 索引策略

为了保证高频查询操作的性能，除了各表的主键和唯一键会自动创建索引外，还需额外建立以下复合索引：

① generations 表:

索引：在 (user_id, status) 字段上建立复合索引。

理由：这是最常见的查询场景，用于快速拉取某用户所有“已完成”的作品以在画廊中展示，或查询“处理中”的任务列表。

② collections 表:

索引：在 (user_id, parent_id) 字段上建立复合索引。

理由：用于高效地构建和查询用户的文件夹树。当用户打开一个文件夹时，需要根据 user_id 和 parent_id 快速查找其下的所有子节点（文件和子文件夹）。

五、接口设计

5.1 用户界面

系统客户端界面中主要包含三类组件：界面层、控制层、服务层。

界面层由多个窗口组成，负责界面展示与用户交互，不包含网络逻辑、不负责业务流程。界面层通过信号将用户操作传递给控制层。

控制层负责业务流程管理，触发服务层请求、处理服务层结果，并调用界面层的方法完成界面跳转或提示。控制层不直接访问网络，也不负责长期数据存储，而是承担界面层和服务层之间的协调作用。

服务层不处理界面，也不管理窗口，只专注于网络与数据的交互。

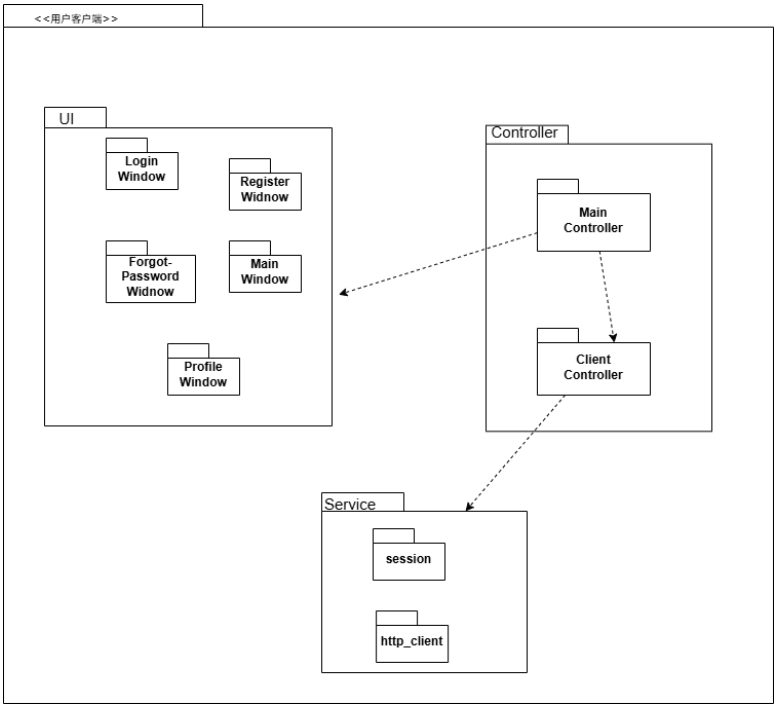

图 5.1 客户端包图

系统的用户界面采用简洁直观的图形界面（GUI），以下列出了主要界面及功能：

1）**登录界面：**提供用户账号登录入口。界面包含输入用户名和密码的文本框，以及“登录”按钮。同时提供跳转至注册页面和找回密码功能的链接按钮。用户输入凭据点击“登录”后，系统验证身份并反馈登录结果。

图 5.2 登录界面

2）**注册界面：**提供新用户注册表单。界面包含用户名、邮箱、密码等字段的输入框和“注册”按钮。用户填写完信息点击注册后，系统校验输入合法性并创建新账户，然后返回登录界面或提示注册成功。

图 5.3 注册界面

3）忘记密码界面：提供密码重置流程入口。界面让用户输入注册邮箱并点击“获取验证码”，然后输入收到的验证码和新密码提交重置请求。系统验证验证码正确后更新密码，并跳转回登录界面。

图 5.4 忘记密码界面

4）主界面：这是用户登录成功后看到的主要工作界面，如图所示。左侧为功能菜单或工具栏，用户可在此选择“内容生成”、“生成记录”、“个人资料”等功能入口；右侧为内容区域。默认显示内容生成功能的操作区：左半部分为输入区域，允许用户选择生成模式（文生图/图生图/文生视频），输入文本提示词或上传参考图片，设置参数后点击相应生成按钮；右半部分为结果预览区，以列表或缩略图形式按时间顺序显示用户近期生成的图片/视频结果，并提供导航至“收藏夹”和“个人资料”等部分。

图 5.5 主界面

5）收藏夹界面：从主界面右侧打开的收藏夹侧栏，用于浏览和管理已收藏的生成结果。用户可以在此创建文件夹、查看各文件夹下的内容缩略图列表、移动或移出收藏内容等。界面支持折叠/展开多级文件夹树结构。

图 5.6 收藏夹界面

6）个人资料界面：提供用户账户信息管理的界面。界面左侧是个人资料菜单栏，包括“账号信息”、“修改邮箱”、“修改密码”等选项卡。默认显示“账号信息”选项卡，展示当前账户的用户名和邮箱等基本信息；切换到“修改邮箱”时，在右侧显示输入新邮箱和提交的界面；选择“修改密码”时，则显示输入旧密码、新密码及确认的新密码的表单界面。用户填写后点击“保存”提交修改请求。整个界面风格与登录窗口保持一致，操作直观，便于用户管理个人账户信息。

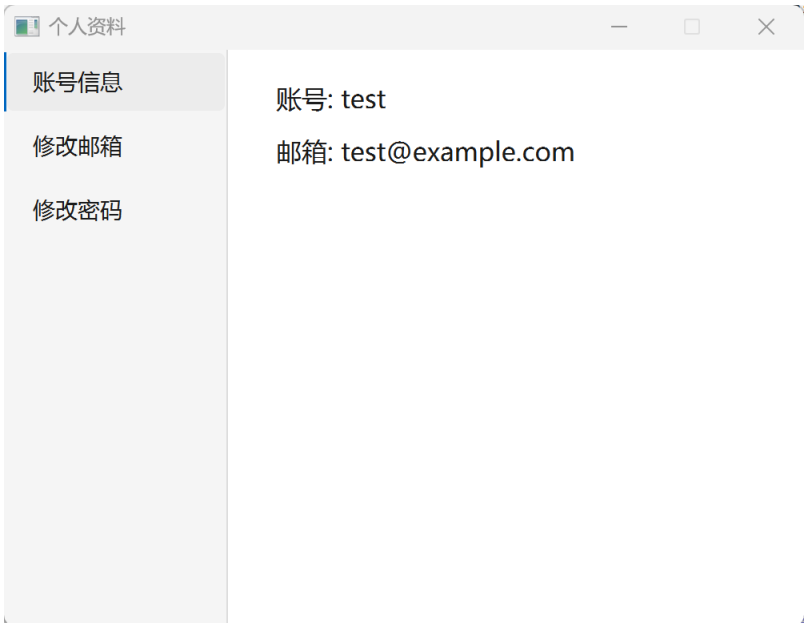

图 5.7 个人资料界面 - 账号信息

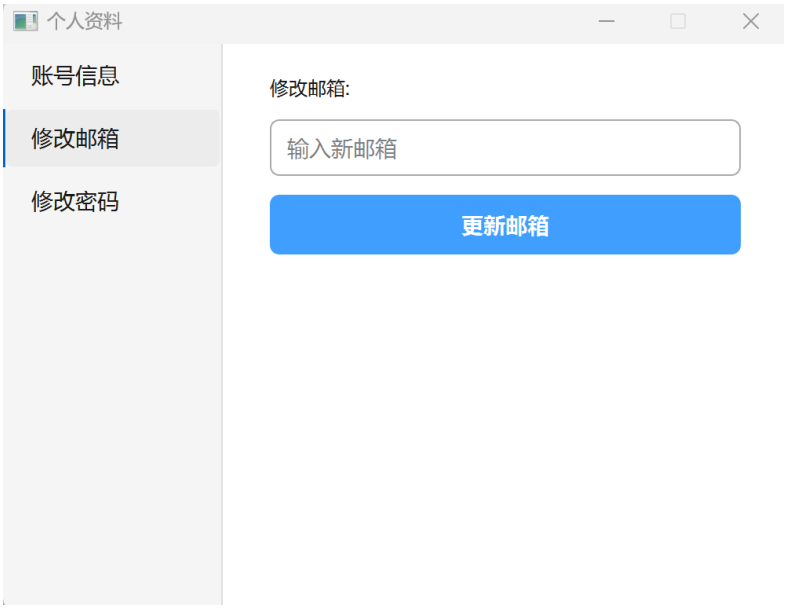

图 5.8 个人资料界面 - 修改邮箱

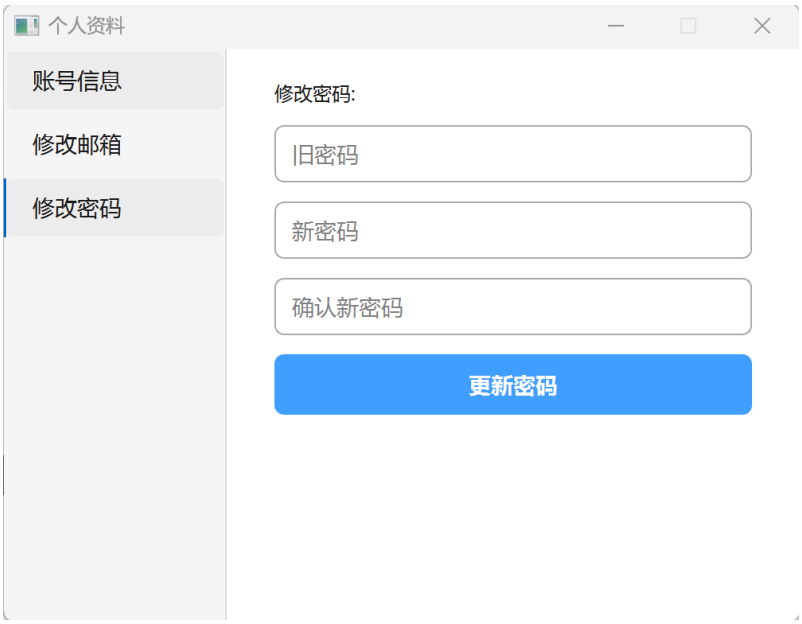

图 5.9 个人资料界面 - 修改密码

7）在线画廊界面：画廊页面由顶部导航栏和作品展示区组成。顶部导航包含平台标识“PICTUREDECK”、主要功能选项（如“画廊”、“市场”）以及搜索框。主体展示区采用卡片式网格布局显示平台内公开的作品列表，每个作品卡片包含缩略图、标题、生成日期、使用模型类型及若干标签。若作品已铸造成 NFT，则卡片右上角显示 “NFT” 标识及定价信息；非 NFT 作品则标注“非 NFT”。界面整体采用深色背景搭配浅色卡片（卡片带圆角和阴影效果），标签以圆角矩形高亮显示在卡片底部。用户可以通过搜索或筛选浏览作品列表，点击某一作品卡片可打开作品详情页面。

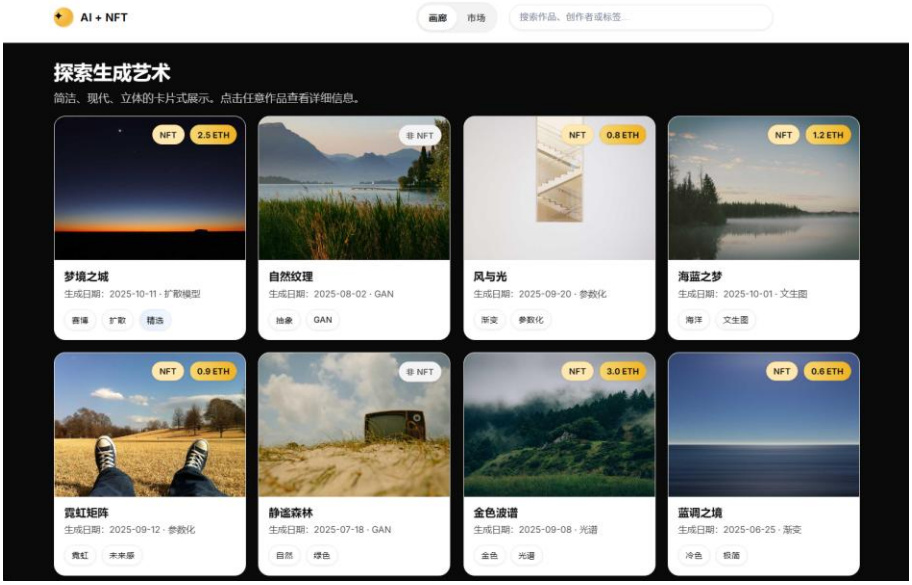

图 5.10 在线画廊界面

8）作品详情界面：作品详情页面展示选定作品的完整大图或视频，以及作品的标题、描述、创作信息（作者、生成时间、使用模型参数等）和标签。如果作品已铸造成 NFT，还会显示其区块链相关信息（如 Token ID、交易价格等）和交易按钮。用户在该页面可以对作品执行收藏、点赞、分享等互动操作。

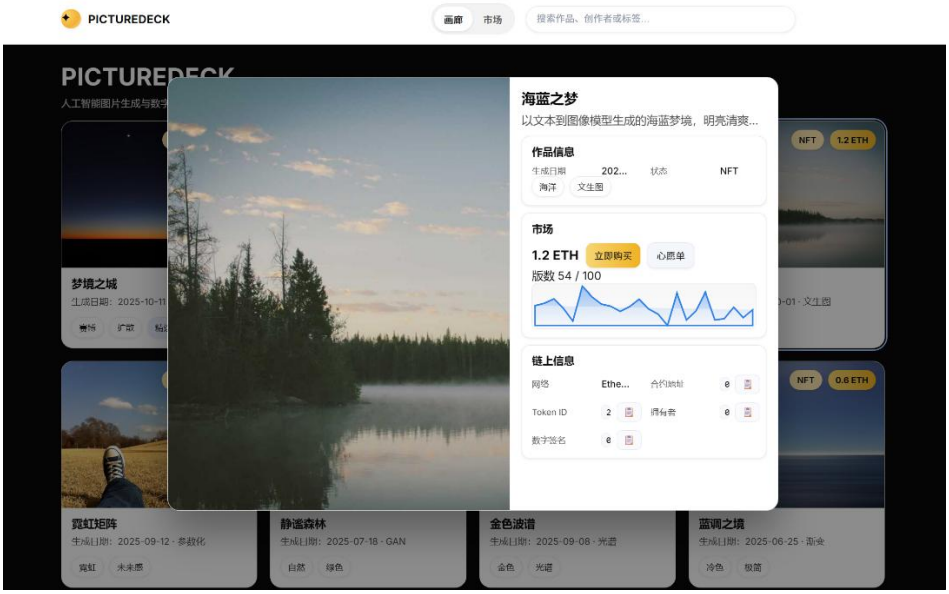

图 5.11 作品详情界面

5.2 外部接口

本系统通过调用多个外部 API 服务实现 AI 内容生成与 NFT 铸造的全流程集成，包括图像/视频生成、去中心化存储、元数据上传、NFT 铸造上链及区块链查询等操作。所有外部 API 调用均采用 HTTPS 加密传输，遵循 RESTful 规范，以保证交互的安全性和可维护性。

外部接口的概览如下表所示：

模块	URL	方法	功能
AI 内容生成	https://api.openai.com/v1/images/generations	POST	文生图任务（调用 OpenAI 模型生成图片）
AI 内容生成	https://api.stability.ai/v2beta/stable-image/generate/sd3	POST	图生图任务（调用 Stability AI 模型生成图片）
去中心化存储	https://api.nft.storage/upload	POST	上传图片至 IPFS
元数据生成	https://api.nft.storage/upload	POST	上传元数据 JSON 至 IPFS

NFT 铸造	https://api.nftport.xyz/v0/mints/easy/urls	POST	铸造 NFT（在区块链上创建 NFT）
区块链查询	https://api.nftport.xyz/v0/transactions/{tx_hash}?chain=polygon	GET	查询上链交易状态
展示与外链	https://mumbai.polygonscan.com/tx/{tx_hash}	GET	区块链浏览器交易详情链接
展示与外链	https://testnets.opensea.io/assets/mumbai/{contract}/{token_id}	GET	OpenSea 测试网 NFT 展示页

表 5.1 外部接口一览

上述外部接口中，大模型 API（如 OpenAI 和 Stability AI）用于内容生成，NFT Storage API 用于去中心化存储文件和元数据，NFTport API 用于铸造 NFT 及查询交易状态，Polygonscan 和 OpenSea 链接用于浏览链上交易记录和作品展示。通过对接这些外部服务，系统实现了从内容生成到链上确权的一系列自动化操作。其中部分第三方 API 服务和参数可能根据实际选型有所调整，但整体调用流程保持一致。接口的详细调用和集成配置在实现阶段将根据各服务提供的文档进行对接和测试，以确保功能正确实现和安全可靠。

5.3 内部接口

1）前后端交互接口

本系统前后端通过标准化的 RESTful API 进行通信，所有接口均使用 HTTPS 协议加密传输，以保证数据在网络传输过程中的安全性和完整性。前端通过发送 HTTP 请求与后端 API 进行交互，包括用户管理、内容生成功能模块。

通用 API 设计原则：

- URL 结构：所有 API 均以 /api/v1/ 为前缀，遵循 主版本号/资源名 的格式。
- 认证机制：除登录/注册等公开接口外，所有需要授权的接口均需在 HTTP 请求头中携带 Authorization: Bearer <JWT_TOKEN>。
- 请求/响应格式：所有请求体和响应体均使用 application/json 格式。

相关接口的概览参考下表。接口详述见后文。

模块	URI	方法	功能
用户管理	/api/v1/auth/register	POST	注册
用户管理	/api/v1/auth/login	POST	登录
用户管理	/api/v1/auth/forgot-password	POST	忘记密码，请求发送邮件

用户管理	/api/v1/auth/reset-password	POST	忘记密码，请求重置密码
用户管理	/api/v1/user/reset-password	POST	修改密码
用户管理	/api/v1/user/reset-email	POST	修改邮箱
内容生成	/api/v1/generation	POST	发起生成任务
内容生成	/api/v1/upload	POST	上传参考图片
内容生成	/api/v1/generation/{taskId}	GET	查询生成结果
内容生成	result_url（由查询生成结果的响应指定）	GET	获得生成结果

表 5.1 前后端接口一览

① 注册接口

请求体示例:

```
{
        "username":"test_user",
        "email":"test@example.com",
        "password": "User@1234"
}
```

请求参数说明：

参数名	类型	说明	约束
username	string	用户名	唯一
email	string	用户邮箱（用于找回密码）	唯一
password	string	用户密码	

响应体示例：

```
{
       "code": 201,
    "message": "注册成功",
       "data": {
        "username": "test_user",
        "email": "test@example.com",
        "created_at": "2025-10-30T14:10:00Z"
       }
}
```


响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 401（失败，用户名重复）； 402（失败，邮箱重复）； 500（失败，服务器内部错误）
data	object	用户注册信息	
data.created_at	string	注册时间	

② 登录接口

请求体示例:

```
{
  "username": "test_user",
  "password": "User@1234"
}
```

请求参数说明：

参数名	类型	说明	约束
username	string	用户名	
password	string	用户密码	

响应体示例：

```
{
  "code": 200,
  "message": "登录成功",
  "data": {
    "username": "test_user",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "token_expire_at": "2025-11-30T14:30:00Z"
  }
}
```

响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 401（失败，账号或密码错误）；

			500（失败，服务器内部错误）
data	object	用户信息	
data.token	string	令牌，用于后续接口鉴权	
data.token_expire_at	string	令牌过期时间（UTC 时间）	

③ 忘记密码、请求发送邮件接口

请求体示例:

```
{
  "email": "test@example.com"
}
```

请求参数说明：

参数名	类型	说明	约束
email	string	用户注册时使用的邮箱	

响应体示例：

```
{
  "code": 200,
  "message": "邮件已发送"
}
```

响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 401（失败，邮箱未注册）； 500（失败，服务器内部错误）

④ 忘记密码、重置密码接口

请求体示例:

```
{
  "email": "test@example.com",
  "verification_code": "123456",
  "new_password": "User@5678",
}
```

请求参数说明：

参数名	类型	说明	约束
email	string	用户注册时使用的邮箱	
verification_code	string	系统发送到邮箱的验证码	
new_password	string	新密码	

响应体示例：

```
{
  "code": 200,
```

```
  "message": "成功重置密码"
}
```

响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 401（失败，验证码错误或过期）； 500（失败，服务器内部错误）

⑤ 修改密码接口

请求头包含：

字段名	类型	说明
Authorization	string	值为 Bearer <Token>，用于身份认证与鉴权

请求体示例:

```
{
  "old_password": "User@1234",
  "new_password": "User@5678",
}
```

请求参数说明：

参数名	类型	说明	约束
old_password	string	旧密码	

new_password	string	新密码	

响应体示例：

```
{
  "code": 200,
  "message": "成功修改密码"
}
```

响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 401（失败，旧密码不正确）； 500（失败，服务器内部错误）

⑥ 修改邮箱接口

请求头包含：

字段名	类型	说明
Authorization	string	值为 Bearer <Token>，用于身份认证与鉴权

请求体示例:

```
{
  "new_email": "newtest@example.com",
}
```

请求参数说明：

参数名	类型	说明	约束
new_email	string	新密码	

响应体示例：

```
{
  "code": 200,
  "message": "成功修改邮箱"
  "data": {
    "email": "newtest@example.com"
  }
}
```

响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 401（失败，邮箱重复）； 500（失败，服务器内部错误）

⑦ 发起生成任务接口

请求头包含：

字段名	类型	说明
Authorization	string	值为 Bearer <Token>，用于身份认证与鉴权

请求体示例:

```
{
```

```
  "type": "i2i",
  "prompt": "让上传的照片呈现出油画效果",
  "images": ["https://example.com/uploads/ref_10231.png"],
}
```

请求参数说明：

参数名	类型	说明	约束
type	string	任务类型	取值为： “t2i”（文生图）； “i2i”（参考图生图）； “t2v”（文生视频）； “i2v”（关键帧生视频）；
prompt	string	文本提示词	
images	array<string>	一组参考图片URL，可为空	

响应体示例：

```
{
  "code": 200,
  "message": "生成任务已创建"
  "data": {
    "task_id": "task_10001",
    "created_at": "2025-11-04T17:00:00Z"
  }
}
```

响应体参数说明：

参数名	类型	说明	约束

code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 500（失败，服务器内部错误）
data.task_id	string	系统生成的任务 ID	唯一

⑧ 上传参考图片

请求头包含：

字段名	类型	说明
Authorization	string	值为 Bearer <Token>，用于身份认证与鉴权

请求体为上传的图片

响应体示例：

```
{
  "code": 200,
  "message": "上传成功"
  "data": {
    "file_id": "ref_10231",
    "file_url": "https://cdn.aigcapp.com/uploads/ref_10231.png"
  }
}
```

响应体参数说明：

参数名	类型	说明	约束

code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 500（失败，服务器内部错误）
data.file_id	string	图片在系统中的临时 id	唯一
data.file_url	string	图片在系统中的临时 url	唯一

⑨ 查询生成结果

请求头包含：

字段名	类型	说明
Authorization	string	值为 Bearer <Token>，用于身份认证与鉴权

GET 方法无请求体

响应体示例：

```
{
  "code": 200,
  "message": "任务完成",
  "data": {
    "task_id": "task_10001",
    "status": "completed",
    "result_url": "https://example.com/results/task_10001.png",
    "created_at": "2025-11-04T17:00:00Z",
    "completed_at": "2025-11-04T17:00:15Z"
  }
```

}

响应体参数说明：

参数名	类型	说明	约束
code	int	状态码	取值为： 200（成功）； 400（失败，参数不合规）； 500（失败，服务器内部错误）
data.status	string	任务的状态	取值为： processing; completed; failed
data.file_url	string	可选，指定完成的生成结果的 url	

⑩ 获取生成结果

请求头包含：

字段名	类型	说明
Authorization	string	值为 Bearer <Token>，用于身份认证与鉴权

GET 方法无请求体

响应体为生成的图片或视频

六、其他

6.1 安全保密设计

系统采用多层安全策略以保障用户数据和生成内容的安全性。所有外部通信均使用 HTTPS 加密通道，防止流量被窃听或篡改。用户密码使用行业标准的哈希算法存储，如 bcrypt 或 Argon2，避免明文存储风险。用户身份认证基于 JWT Token，实现短周期会话管理与权限校验。

在涉及敏感操作（如修改邮箱、重置密码）时，系统要求用户通过邮箱验证码再次验证身份。所有生成内容在存储前会经过内容审核服务过滤，以防止非法使用。生成结果的访问链接支持配置为带签名的限时 URL，有效防止外链泄露和未授权访问。后台管理功能仅限管理员角色使用，后台接口采用严格的权限验证机制，避免越权访问。同时，系统对所有关键操作进行日志记录，并定期清理敏感日志，防止不必要的暴露。

6.2 维护设计

系统在设计上注重可维护性，采用模块化、层次化结构，使功能组件之间保持低耦合。代码仓库使用 GIT 版本控制系统进行管理，以便多人协作开发与回滚操作。数据库模型遵循清晰的字段命名和外键引用规则，便于长期维护和扩展。

系统通过配置文件管理不同运行环境（开发、测试、生产）的参数，减少硬编码。后台设有日志中心，用于查看运行记录、错误信息和性能数据，方便开发人员定位问题。系统提供监控平台，监测 CPU、内存、任务队列长度、数据库连接数等关键资源，并在出现异常时触发告警。生成模块采用可插拔设计，使新增 API 模型或替换第三方服务成本较低。

在数据层面，定期进行备份并支持数据恢复流程，保障内容的长期安全与可用性。系统文档包括接口说明、数据库设计、部署流程及运维指南，确保后续开发者能够快速理解并维护整个系统。