智慧幕墙系统架构运维与数据集管理平台：系统设计规约 (SDS) 文档

💻 **小组成员：**2250758 林继申、2251730 刘淑仪、2256225 中谷天音

1 引言

1.1 项目背景

- **项目名称**：智慧幕墙系统架构运维与数据集管理平台
- **项目目标**：
  - 搭建智慧幕墙后端集成系统架构；
  - 构建一个自动化集成与部署的 CI/CD 流水线；
  - 进行智慧幕墙后端 GPU 算力服务器的运维工作；
  - 构建一个高效、安全、可扩展的数据集管理平台，支持数据集的上传、下载、管理、权限控制等功能。

1.2 设计目标

本系统设计文档（SDS）旨在为开发团队提供详细的设计指导，确保系统能够按照需求规格说明书（SRS）中的要求实现。设计目标包括：

1. **模块化设计**：通过高内聚低耦合的设计原则，确保系统的可维护性和可扩展性。
2. **高效数据管理**：通过阿里云对象存储（OSS）实现大规模数据的高效存储和访问。
3. **安全性设计**：通过权限控制、数据加密和日志监控，确保数据的安全性和隐私性。
4. **自动化运维**：通过 Docker、Docker Compose 和 CI/CD 流水线，实现系统的自动化部署和持续集成。
5. **标准化接口**：遵循 RESTful API 设计原则，提供标准化的数据上传、下载和查询接口，确保前后端交互的清晰和高效。

1.3 设计原则

1. **高内聚低耦合**：模块内部功能紧密相关，模块之间依赖关系尽量减少。
2. **模块化设计**：按照功能或业务领域划分模块，便于独立开发和维护。
3. **错误处理机制**：定义明确的错误码和错误信息，提升系统的稳定性和用户体验。
4. **可扩展性和可维护性**：设计时考虑未来的扩展需求，确保系统能够随着业务增长而灵活扩展。

1.4 设计依据

本系统设计规约（SDS）的制定基于以下关键依据和标准，确保系统设计符合项目需求、技术规范和客户化定制的要求：

1.4.1 基于 SRS 文档的系统设计

- 本 SDS 文档是基于《智慧幕墙系统架构运维与数据集管理平台：系统需求规约 (SRS) 文档》的需求分析结果进行设计的。
- SRS 文档中明确的功能需求和非功能需求（如数据集管理、权限控制、备份与恢复、日志管理等）是系统设计的核心依据。
- 设计过程中严格遵循 SRS 文档中的模块划分、接口定义和安全性要求，确保系统功能与需求一致。

1.4.2 基于 Docker 开发部署的容器化标准

- 系统设计采用 Docker 容器化技术，确保开发、测试和生产环境的一致性。
- 基于 Docker Compose 的多容器部署方案，支持快速构建和扩展系统服务。
- 遵循 Docker 的最佳实践，包括镜像优化、容器编排和资源管理，确保系统的高效运行和可维护性。

1.4.3 基于客户化定制的考量

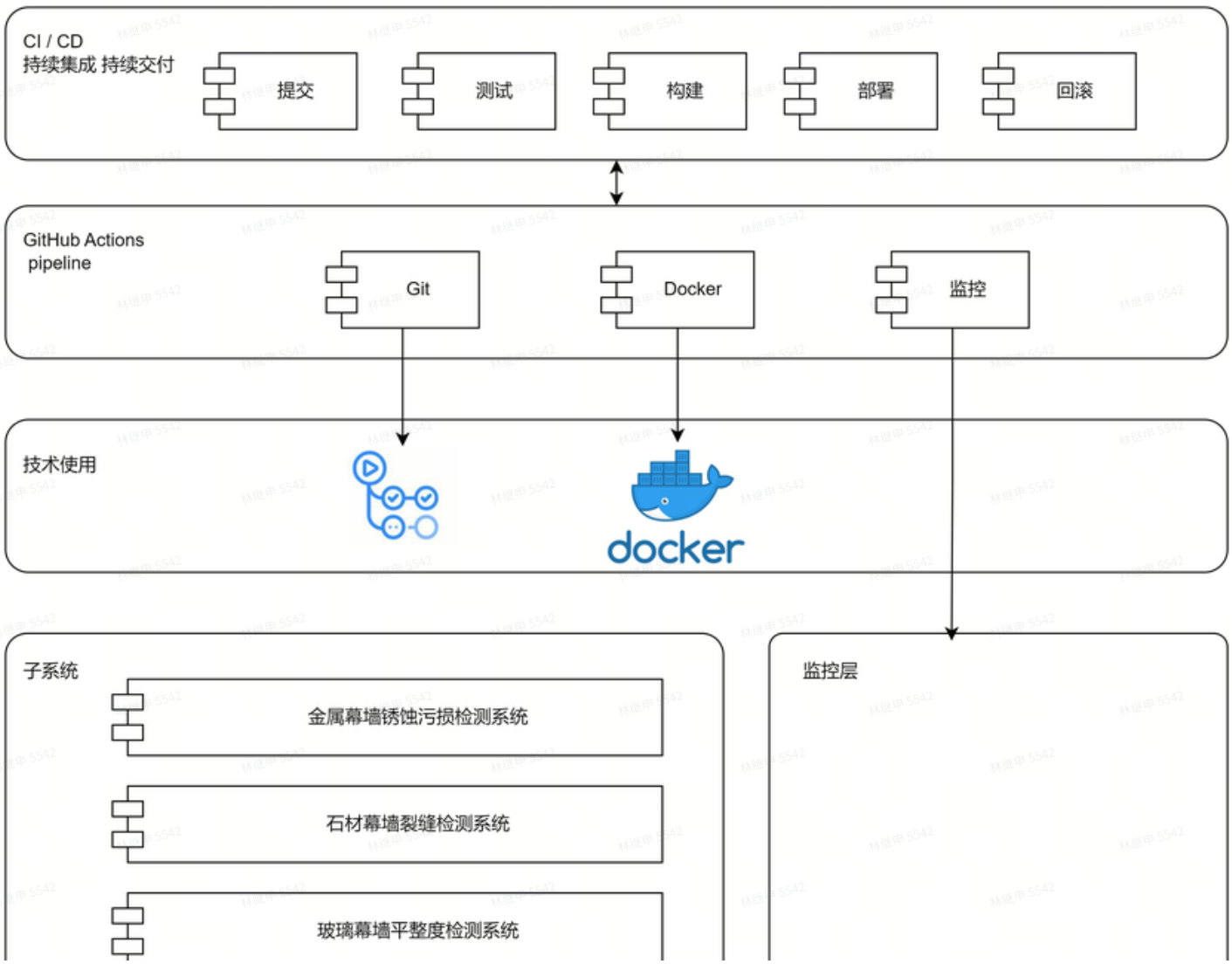
- 系统设计充分考虑了客户的特定需求，包括数据集的分类管理、权限控制的细粒度分配、以及日志审计的定制化功能。
- 针对客户的实际业务场景，设计了灵活的数据备份与恢复策略，支持按需备份和部分恢复功能。
- 在用户界面设计上，采用了客户偏好的交互风格和布局，确保用户体验的流畅性和直观性。

1.4.4 脚本与自动化工具

- 系统设计中使用了大量脚本和自动化工具，以提高开发、部署和运维效率。这些脚本包括：
  - **Docker 镜像构建脚本**：用于快速构建和发布系统服务的 Docker 镜像。
  - **CI/CD 流水线脚本**：通过 GitHub Actions 实现的持续集成和持续部署脚本，支持自动化构建、测试和部署。
  - **备份与恢复脚本**：用于定期备份数据集和元数据，并支持从备份中恢复数据的脚本。
  - **日志清理脚本**：用于定期清理过期日志，释放存储资源的脚本。

2 系统概述

2.1 系统架构图



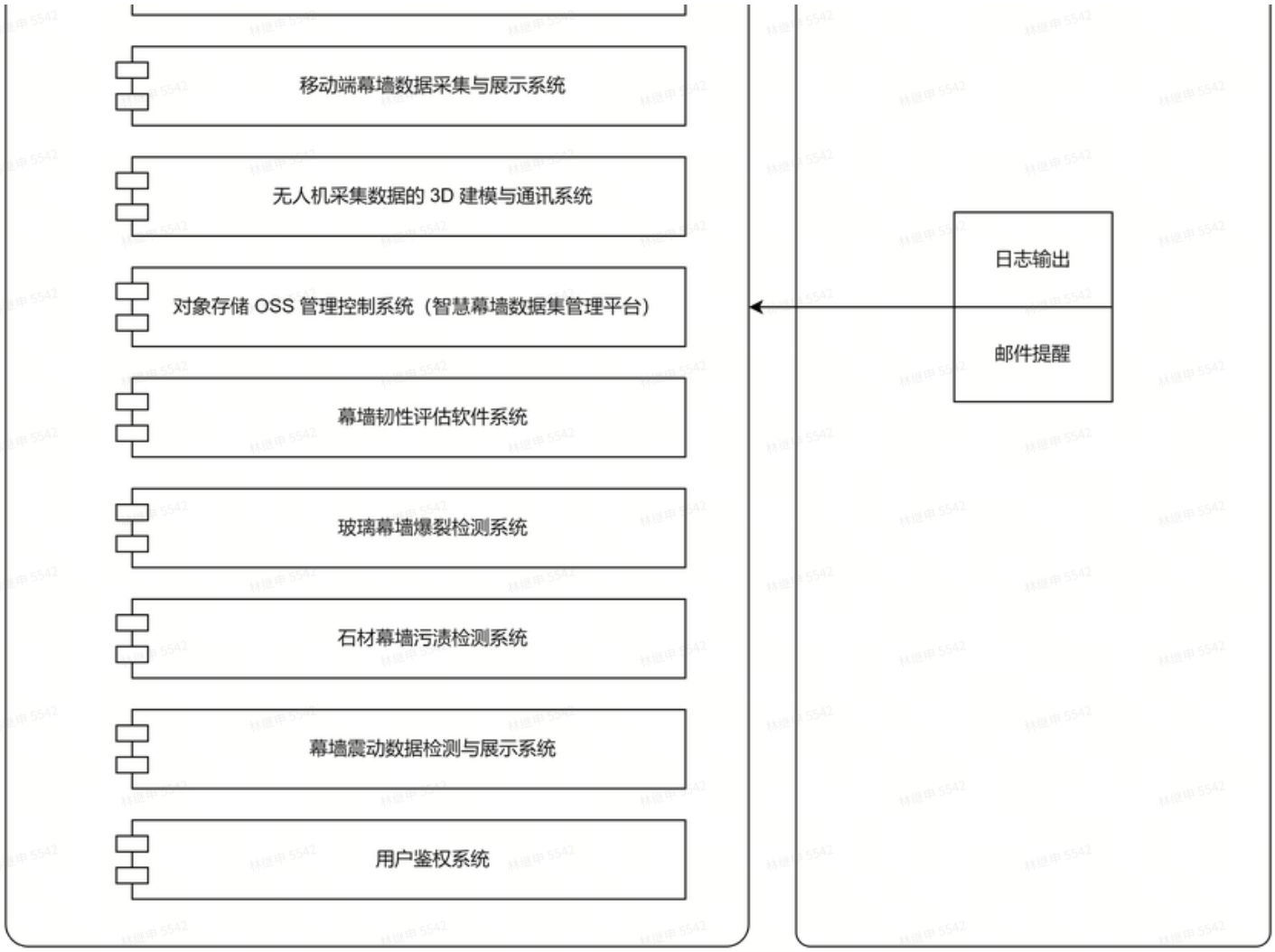


图 2-1 智慧幕墙系统架构与部署运维架构图

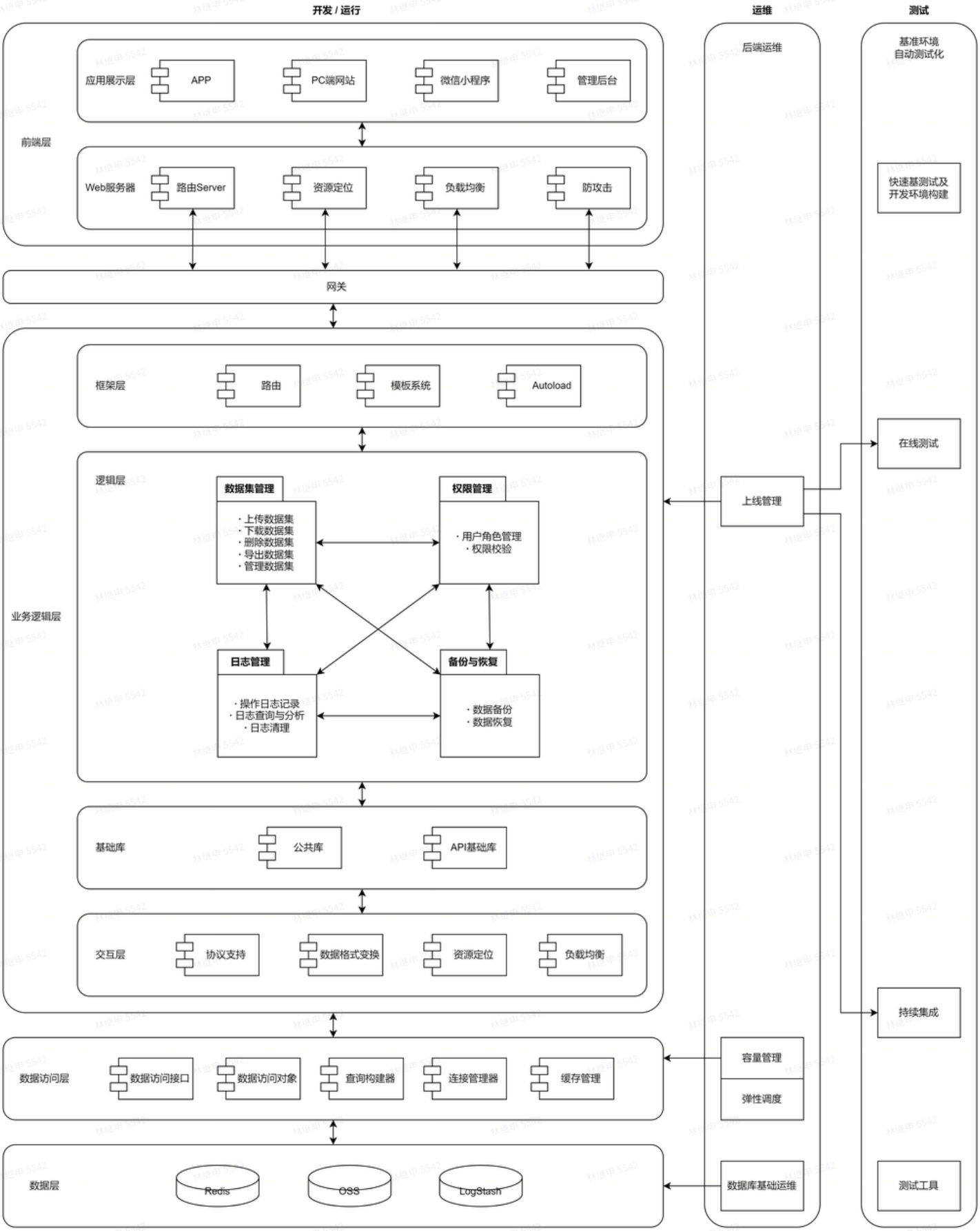


图 2-2 智慧幕墙数据集管理平台系统架构图

2.2 系统架构设计

智慧幕墙数据集管理与运维系统采用分层架构设计，主要分为前端层、后端层和持久化层，具体架构如下：

2.1.1 前端层

- **功能**：负责用户界面的展示和用户操作的交互。
- **技术栈**：基于 Vue.js 框架，采用组件化设计，确保界面的模块化和可复用性。
- **主要模块**：
  - 文件列表展示
  - 文件上传与下载操作
  - 用户登录与权限管理
  - 系统监控与日志查看

2.1.2 后端层

- **功能**：处理业务逻辑、数据管理和权限控制，为前端提供统一的 API 接口。
- **技术栈**：基于 Spring Boot 框架，采用模块化设计，确保业务逻辑的独立性和可维护性。
- **主要模块**：
  - 用户管理：负责用户认证和权限控制。
  - 数据集管理：处理数据的上传、下载、查询等操作。
  - 日志管理：记录系统运行日志和部署日志。
  - API 网关：提供统一的 RESTful API 接口，供前端调用。

2.1.3 持久化层

- **功能**：负责数据集的存储和管理，确保数据的高效访问和安全性。
- **技术栈**：使用阿里云对象存储（OSS）作为数据存储服务，支持大规模数据的高效存储和快速访问。
- **主要功能**：
  - 数据存储：通过 OSS 实现数据集的持久化存储。
  - 数据备份：定期对数据集进行自动化备份，确保数据安全。
  - 数据恢复：在数据丢失或损坏时，能够快速恢复。

2.1.4 安全性设计

- **访问控制**：通过阿里云 RAM 实现严格的权限管理，确保只有授权用户或角色能够访问特定数据集。
- **数据加密**：对敏感数据进行加密存储，确保数据的安全性。

2.1.5 运维支持

- **环境一致性**：使用 Docker 和 Docker Compose 技术，确保开发、测试和生产环境的一致性。
- **自动化部署**：通过 CI/CD 流水线实现代码的自动化构建、测试和部署。
- **资源监控**：实时监控服务器资源使用情况，确保系统的高效运行。

2.1.6 目标系统环境

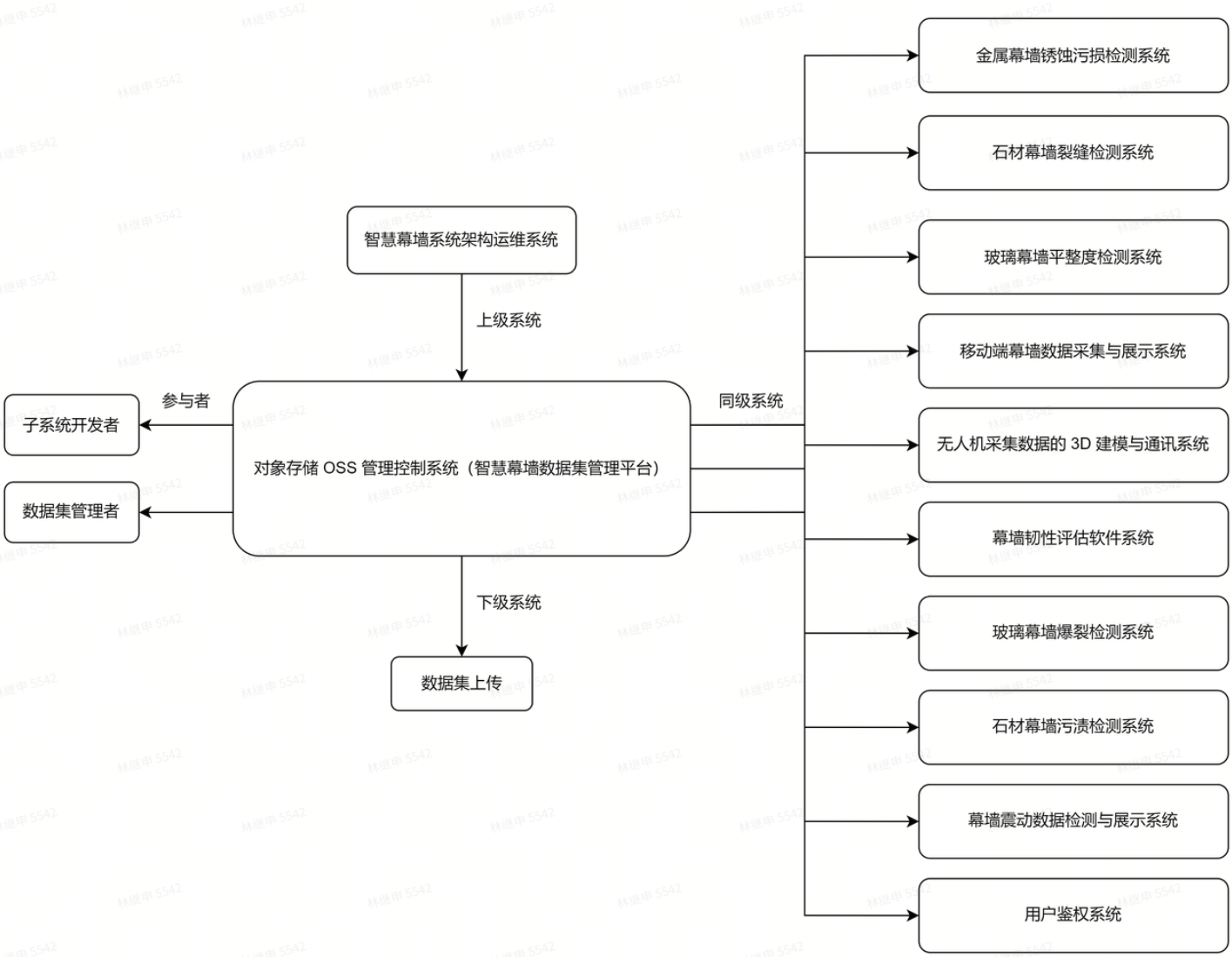


图 2-3 智慧幕墙系统 ACD 图

3 详细设计

3.1 数据集管理模块

3.1.1 上传数据集

- **功能描述**：用户通过前端界面选择本地文件并上传至阿里云 OSS（对象存储服务）。
- **流程**：
  a. 用户选择文件并触发上传操作。
  b. 前端将文件及相关信息（如文件路径、用户信息）通过 RESTful API 发送至后端。

c. 后端对文件路径进行合法性校验，确保路径符合命名规则且不包含非法字符。
d. 后端调用阿里云 OSS 的 SDK，将文件上传至指定的 OSS 存储桶。
e. 上传成功后，后端生成文件的下载链接并返回给前端。
f. 前端展示下载链接，用户可通过该链接直接下载文件。

- **错误处理**：
  - 如果文件路径非法，返回 400 Bad Request 错误。
  - 如果上传过程中出现网络中断或 OSS 服务不可用，返回 500 Internal Server Error。

3.1.2 下载数据集

- **功能描述**：用户通过文件路径从 OSS 下载指定文件。
- **流程**：
  a. 用户通过前端界面输入文件路径并触发下载操作。
  b. 前端将文件路径通过 RESTful API 发送至后端。
  c. 后端对文件路径进行校验，确保文件存在且用户有权限下载。
  d. 后端调用阿里云 OSS 的 SDK，从 OSS 存储桶中下载文件。
  e. 下载成功后，后端将文件的二进制数据返回给前端。
  f. 前端将文件保存到用户本地，并提供下载完成的提示。
- **错误处理**：
  - 如果文件不存在，返回 404 Not Found 错误。
  - 如果用户无权限下载，返回 401 Unauthorized 错误。

3.1.3 删除数据集

- **功能描述**：用户可以删除指定的数据集。
- **流程**：
  a. 用户通过前端界面选择要删除的文件并触发删除操作。
  b. 前端将文件路径及相关信息通过 RESTful API 发送至后端。
  c. 后端验证用户是否有权限删除该文件。
  d. 后端调用阿里云 OSS 的 SDK，从 OSS 存储桶中删除文件及其元数据。
  e. 删除成功后，后端返回 200 OK 状态码，并提示用户删除成功。
  f. 前端更新文件列表，移除已删除的文件。
- **错误处理**：
  - 如果文件不存在，返回 404 Not Found 错误。

- 如果用户无权限删除，返回 401 Unauthorized 错误。

3.1.4 导出数据集

- **功能描述**：用户可以选择单个或多个数据集，导出其对应的 URL 链接。
- **流程**：
  a. 用户选择单个数据集或批量选择多个数据集。
  b. 系统生成可访问的 URL 链接，用户可以通过该链接直接访问或下载数据集。
  c. 导出的 URL 链接具有时效性，确保数据的安全性。
  d. 每次导出操作都会被记录到操作日志中，便于后续审计和追踪。
- **权限校验**：在导出操作时，系统会进行权限校验，确保只有符合权限要求的用户才能执行导出操作。

3.1.5 管理数据集

- **功能描述**：用户可以对数据集进行复制、剪切、删除和批量粘贴等管理操作。
- **流程**：
  a. **文件复制**：
    - 用户通过前端界面选择文件并触发复制操作。
    - 前端将文件路径及相关信息通过 RESTful API 发送至后端。
    - 后端调用阿里云 OSS 的 SDK，复制指定文件到目标路径。
    - 复制成功后，后端返回 200 OK 状态码，并提示用户复制成功。
  b. **文件剪切**
    - 用户通过前端界面选择文件并触发剪切操作。
    - 前端将文件路径及相关信息通过 RESTful API 发送至后端。
    - 后端调用阿里云 OSS 的 SDK，将文件从原路径移动到目标路径。
    - 剪切成功后，后端返回 200 OK 状态码，并提示用户剪切成功。
  c. **文件粘贴**
    - 用户通过前端界面选择一个或多个文件并触发粘贴操作。
    - 前端将文件路径及相关信息通过 RESTful API 发送至后端。
    - 后端调用阿里云 OSS 的 SDK，批量粘贴文件到目标路径。
    - 粘贴成功后，后端返回 200 OK 状态码，并提示用户粘贴成功。
- **错误处理**：
  - 如果文件不存在，返回 404 Not Found 错误。

- 如果操作过程中出现错误，返回 500 Internal Server Error，并记录错误日志

3.1.6 权限控制模块

3.1.6.1 用户角色管理

- **功能描述**：系统支持多种用户角色，如管理员、普通用户等，不同角色拥有不同的权限。
- **角色定义**：
  - **管理员**：拥有最高权限，可以管理所有数据集，包括上传、下载、删除、查询等操作，同时可以管理其他用户角色和权限。
  - **普通用户**：拥有部分权限，通常只能上传、下载和查询自己上传的数据集，无法删除他人上传的数据集。
- **权限分配**：每个角色在系统中都有明确的权限定义，权限包括上传、下载、删除、查询等。

3.1.6.2 权限校验

- **功能描述**：在用户进行上传、下载、删除等操作时，系统会进行权限校验，确保只有符合权限要求的用户才能执行相应操作。
- **校验流程**：
  a. 系统根据用户的角色，查询其拥有的权限列表。
  b. 检查当前操作是否在用户的权限范围内。
  c. 如果用户拥有相应权限，则允许执行操作；否则，返回 401 Unauthorized 错误。
- **实现方式**：权限校验逻辑通过后端 Spring Boot 的拦截器（Interceptor）实现。

3.2 备份与恢复模块

3.2.1 数据备份

- **功能描述**：系统会定期对数据集进行备份，确保数据的安全性和业务连续性。
- **备份策略**：
  - **备份频率**：每周一 00:00 自动执行一次全量备份。
  - **保留时间**：每个备份保留 30 天，过期备份自动删除。
  - **备份内容**：包括所有数据集及其元数据（如文件路径、上传时间、用户信息等）。
- **备份实现**：
  a. 系统在备份时间点自动触发备份任务。
  b. 将数据集及其元数据打包为压缩文件。
  c. 将压缩文件上传至阿里云 OSS 的指定存储桶。

d. 记录备份任务的执行状态和日志。

3.2.2 数据恢复

- **功能描述**：在数据丢失或损坏时，系统可以通过备份进行数据恢复，确保数据安全与业务连续性。
- **恢复流程**：
  a. 系统管理员从备份列表中选择需要恢复的备份点。
  b. 系统管理员通过界面或命令行工具触发恢复操作。
  c. 系统从阿里云 OSS 下载备份文件，并通过脚本或工具实现数据恢复。
  d. 系统检查恢复后的数据是否完整和可用，并记录恢复日志。
- **恢复权限**：只有系统管理员可以触发数据恢复操作，确保恢复过程的安全性。

3.3 日志管理模块

3.3.1 操作日志记录

- **功能描述**：系统记录每次部署和系统运行的关键事件，确保运维人员和开发人员能够监控系统状态、排查问题，并支持系统调试和优化。
- **日志内容**：
  - **部署日志**
    - 记录每次部署的详细信息，包括：
      - 部署时间
      - 部署人员
      - 部署版本
      - 部署状态（成功/失败）
      - 部署过程中执行的命令
      - 部署失败时的错误信息
  - **系统日志**
    - 记录系统运行过程中的关键事件，如：
      - 服务启动/停止
      - 资源使用情况（CPU、内存、磁盘）
      - 异常错误（如网络中断、服务崩溃）
- **日志格式**
  - 日志采用结构化格式（如 JSON），便于解析和查询。

- 每条日志包含以下字段：
  - 时间戳
  - 日志级别（如 INFO、WARN、ERROR）
  - 操作类型（如部署、启动、停止）
  - 操作详情
  - 操作结果
- **日志存储**
  - 日志存储在专门的日志服务器或云存储服务（如阿里云日志服务）中，确保日志的安全性和可扩展性。
  - 日志按日期或项目进行分片存储，便于管理和查询。
- **日志记录实现**
  - 通过日志框架在代码中嵌入日志记录逻辑。
  - 部署脚本中增加日志记录功能，确保每次部署的详细操作都被记录。

3.3.2 日志查询与分析

- **功能描述**：运维人员和系统管理员可以通过日志查询界面或命令行工具查询日志，并对日志数据进行分析，生成报告或告警，帮助监控系统健康状态。
- **日志查询**：
  - 支持按时间范围、日志级别、操作类型等条件进行筛选和查询。
  - 查询结果以结构化格式返回，便于进一步分析。
- **日志分析**：
  - 对日志数据进行分析，生成报告或告警，帮助运维人员监控系统健康状态。
  - 支持对异常错误进行统计和分析，帮助快速定位问题。
  - 分析结果可以通过可视化工具展示，便于运维人员直观了解系统状态。

3.3.3 日志清理

- **功能描述**：系统定期清理过期日志，释放存储资源，确保日志存储的高效性和可扩展性。
- **清理策略**：
  - 根据日志的存储时间和存储空间进行配置。
  - 默认策略为保留最近 30 天的日志，过期日志自动删除。
  - 清理操作记录在系统日志中，便于审计和追踪。

3.4 类设计

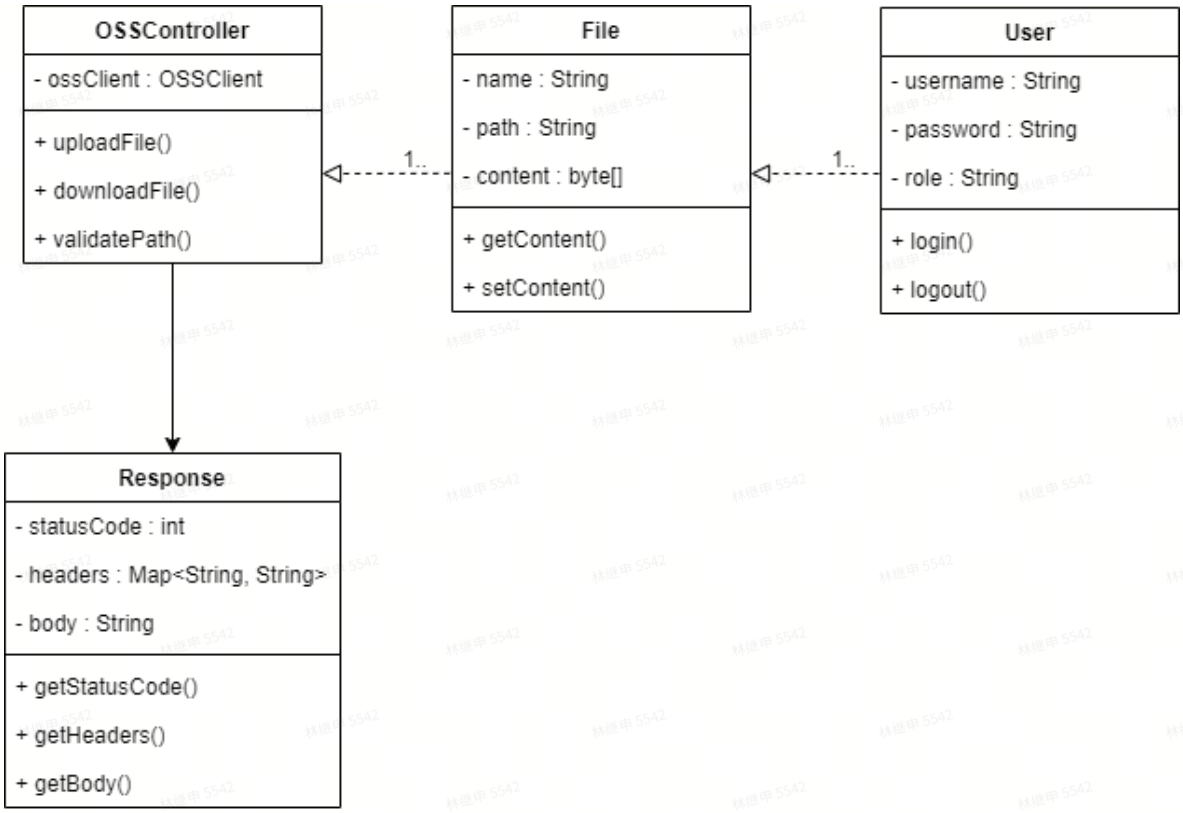


图 3-1 智慧幕墙系统 UML 类图

1. **OSSController**
   - 这是系统中的一个控制器类，负责处理与阿里云对象存储（OSS）相关的操作。
   - 它包含一个 `ossClient` 属性，类型为 `OSSClient`，用于与阿里云 OSS 进行交互。
   - 主要方法包括 `uploadFile()`、`downloadFile()` 和 `validatePath()`，分别用于文件上传、下载和路径验证。
2. **File**
   - 这个类代表系统中的文件对象。
   - 因为 `OSSController` 负责处理文件的上传和下载操作，故该类与 `OSSController` 类有关系。
3. **User**
   - 这个类代表系统中的用户对象。
   - 它包含 `username`、`password` 和 `role` 属性，分别用于存储用户的用户名、密码和角色信息。
   - 主要方法包括 `login()` 和 `logout()`，用于用户的登录和登出操作。

类之间的关系：

- **OSSController** 和 **File** 之间的关系：`OSSController` 类通过 `uploadFile()` 和 `downloadFile()` 方法与 `File` 类进行交互，处理文件的上传和下载操作。

- **OSSController 和 User 之间的关系：** `OSSController` 类需要与 `User` 类进行交互，以验证用户的权限。

3.5 开发视图

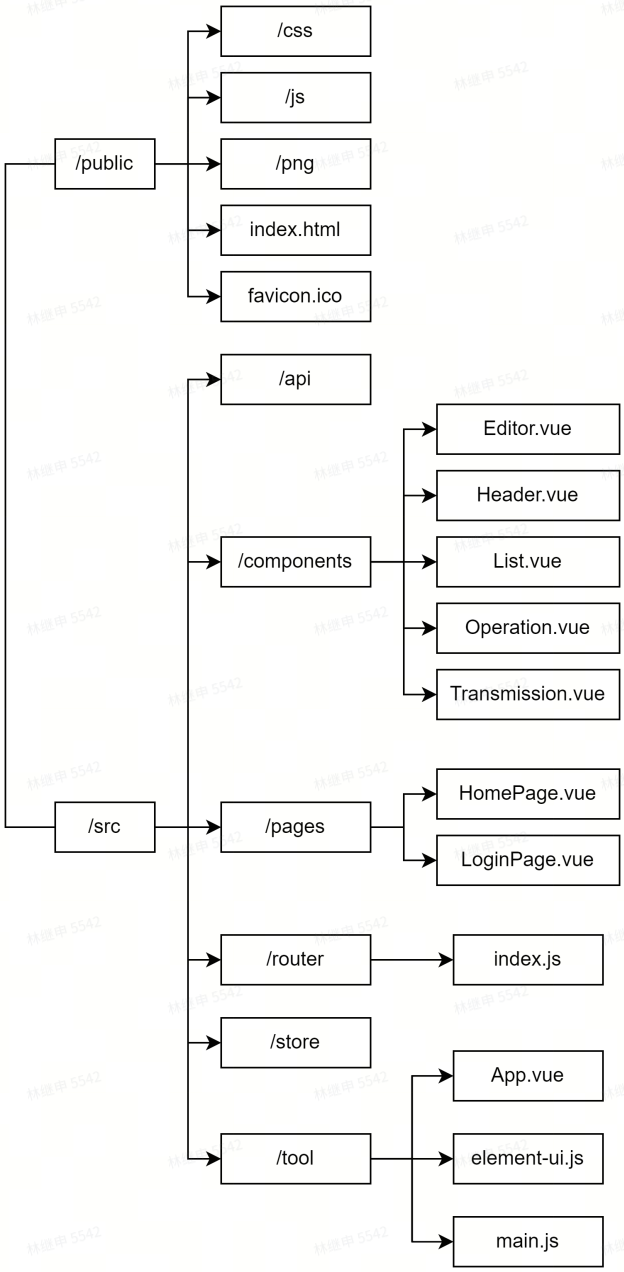


图 3-2 智慧幕墙前端应用程序开发视图

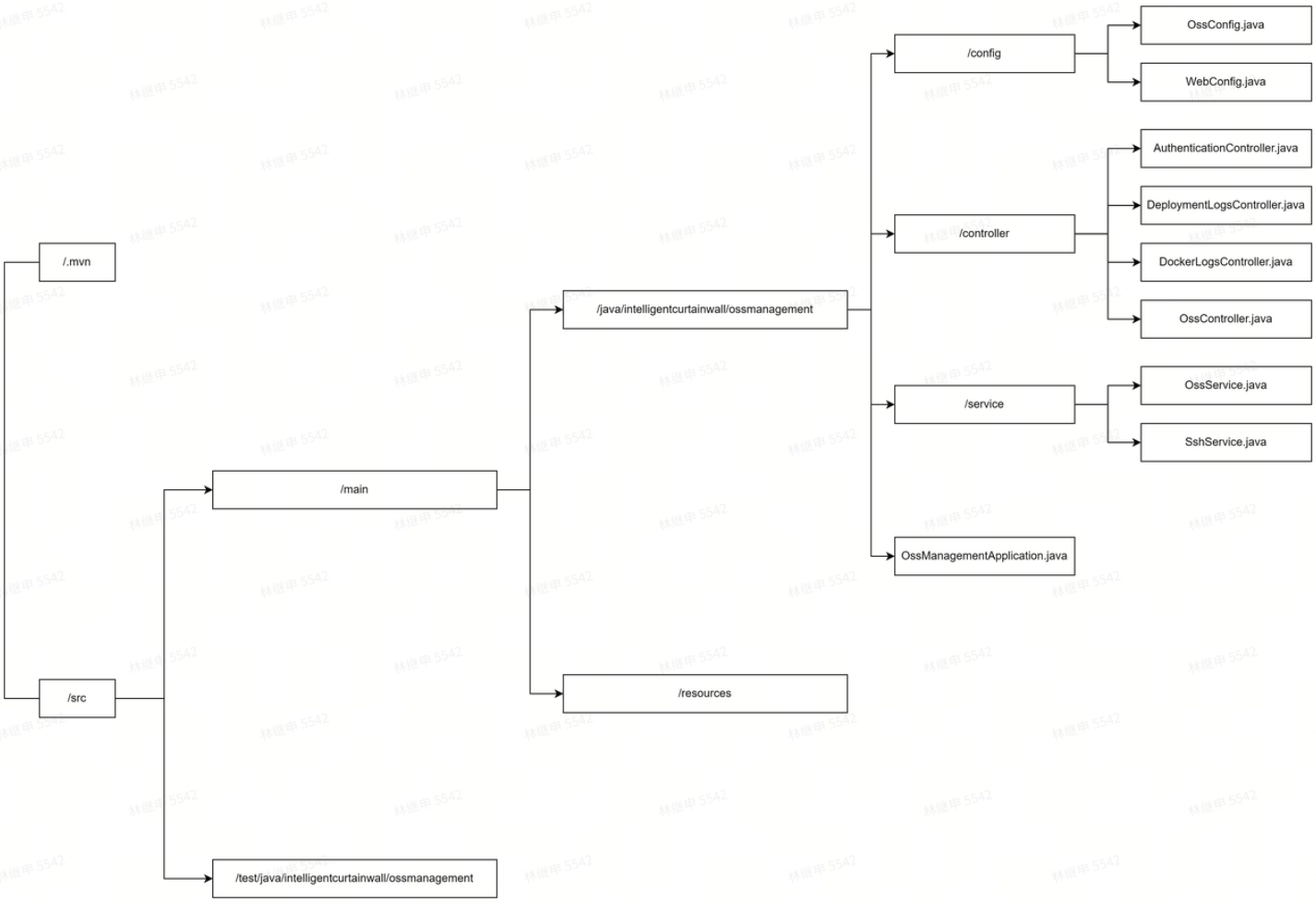


图 3-3 智慧幕墙后端应用程序开发视图

4 接口设计

本目标系统接口分为内部接口、外部接口和人机交互的接口。

4.1 外部接口

子系统名称	接口名称	请求参数	RESTful 调用方式	响应信息
用户管理系统	登录	username, password	POST /login	token, authentication
	发送验证码	email	POST /sendCode	发送成功信息
	校验验证码并注册	email, password, code	POST /validate	注册成功信息
	获取子系统全部用户	sysname	GET /getSubsysUser	email[]
	更新用户权限	username, user	POST /super/updatePermission	更新成功信息

	获取全部用户权限信息	authentication	GET /super/getAllPermissions	user
	获取当前登录用户权限	authentication	GET /custom/getPermissions	user
玻璃平整度检测系统	检测图片平整度	url, username	POST /detect	result
	查询历史记录	username	GET /history	flatnessHistoryDto[]
玻璃爆裂检测系统	上传图片	file	POST /upload	downloadUrl
	上传图片检测	url, username	POST /classify	result
	获得爆裂情况	url, username	POST /showDefect	downloadUrl
	查询历史记录	username	GET /history	defectHistoryDto[]
幕墙韧性评估系统	上传数据	file	POST /data/	数据上传成功信息
	获取所有批次	无	GET /data/batch	data[]
	获取某批次数据	batch	GET /data/batch_data	data[]
	熵权法评估	batch	POST /evalu/entropy	data[]
	获取熵权法评估权重	batch	GET /evalu/entropy	data[]
	获取熵权法评估结果	batch	GET /evalu/eresult	data[]
	粗糙集评估	batch	POST /evalu/roughset	data[]
	获取粗糙集评估权重	batch	GET /evalu/roughset	data[]
	获取粗糙集评估结果	batch	GET /evalu/result	data[]
	获取粗糙集评估的决策规则	batch, input, columns	GET /evalu/rs-deci-rules	data[]
幕墙污渍检测系统	污渍分析	username, input_url	POST /predict	results[]
	污渍历史记录查询	username, start_time, end_time	POST /history	data[]

移动端幕墙数据采集与展示系统	获取建筑名与对应图片	无	GET /buildi	data[]
	获取传感器状态	无	GET /sens	data[]
	检测图片平整度	building	GET /sens	data[]
	查询时程曲线（X/Y/Z方向）	fData, device, table	GET /timeSeries/X GET /timeSeries/Y GET /timeSeries/Z	data[]
	查询频谱曲线（X/Y/Z方向）	fData, device	GET /frequency/X GET /frequency/Y GET /frequency/Z	data[]
	存储时程曲线异常数据	TimeSeriesAnomal	POST /TimeAn	存储成功消息
	获取时程曲线异常数据	无	GET /TimeAnt	data[]
	存储频谱曲线异常数据	FrequencyAnomaly	POST /SpectrumAnomaly	存储成功消息
	获取频谱曲线异常数据	无	GET /SpectrumAnomaly	data[]
幕墙振动数据监测与展示系统	异常数据查询	device, direction, start_time, end_time	GET /abnormalData	data[]
	异常数据下载	device, direction, start_time, end_time	GET /abnormalData/download	文件下载
	设备列表查询	无	GET /deviceList	data[]
	秒级/分钟级/小时级/天级/月级/年级数据查询	device, num	GET /data/{level}	data[]
	振动状态检测	device	GET /vibrationStatus	状态信息
	获取阈值或偏移量	device, type	GET /threshold	阈值或偏移量
	更新阈值或偏移量	device, type, value	POST /threshold	更新成功信息
	告警记录查询		GET /alertRecords	告警记录

		device, start_time, end_time, limit		
	插入告警记录	device, content, call_function, severity, phone, email	POST /alertRecords	插入成功信息
	邮件告警发送	device, alertMessage, email	POST /sendEmailAlert	发送成功信息
	短信告警发送	device, phoneNumber, signName, templateCode, templateParam	POST /sendSmsAlert	发送成功信息
无人机采集数据的 3D 建模与通讯系统	添加异常图片	imagePath	POST /imageData	data[]
	获取所有异常图片坐标	无	GET /imageData/allCoordinates	data[]
	根据点击坐标获取异常图片	clickX, clickY, clickZ	GET /imageData/closest	data[]
	添加交互日志	clickX, clickY, clickZ	POST /interactionLog	data[]
	删除异常图片	imageId	DELETE /imageData/{imageId}	删除成功信息
石材幕墙裂缝检测系统	发送验证码	code, destination	POST /sendemail	发送成功信息
	上传图片	file	POST /upload	downloadUrl
	上传图片检测	url	POST /segment	result[]
	Segformer 检测裂缝	url	POST /segformer-detect	result[]
	CrackSegmentation 检测裂缝	url	POST /crack-detect	result[]
	查询历史记录	无	GET /getProject	ProjectDto
	删除历史记录	id	DELETE /deleteProject	删除成功信息

幕墙锈蚀检测系统	图片上传检测	file, filename	POST /detect	detection_image_url, Inst_url, history, message_url, original_image_url

4.2 内部接口

OSS 控制器 API 提供文件的上传与下载功能，通过 HTTP 请求与服务端交互。所有接口均以 `/oss` 为基础路径。

4.2.1.1 下载文件接口

1. **接口描述**：通过提供文件的路径，从 OSS 存储中下载指定文件。
2. **URL**：`GET /oss/download/{文件路径}`
3. **请求参数**：无直接请求参数。

> ⚠ 文件的最终存储路径（包括文件名）需要在上传请求的 URL 中明确指定。举例来说，如果您上传的原始文件名是 `upload.txt`，并在请求URL中指定存储路径为 `/oss/upload/user/documents/file.txt`，那么该文件会以指定的文件名 `file.txt` 存储于 `user/documents/` 目录下，即最终存储路径为 `user/documents/file.txt`。请注意，这里的 `file.txt` 是需要由您在上传请求中明确指定的新文件名（可以与原文件名重合）。

4. **响应**：
   - **成功**：返回文件的二进制数据，同时响应头包含文件下载的必要信息。
     - **HTTP 状态码**：`200 OK`
     - **响应头**：`Content-Disposition: attachment; filename=<文件名>`
     - **响应体**：文件的二进制内容
   - **失败**：返回 HTTP 错误状态码。
     - `404 Not Found`：文件不存在。

4.2.1.2 上传文件接口

1. **接口描述**：通过提供文件及相关用户信息，将文件上传至 OSS 存储。
2. **URL**：`POST /oss/upload/{文件路径}`
3. **请求参数**：

参数名	类型	必填	描述

file	MultipartFile	是	要上传的文件
userName	String	是	账号（用于身份验证）
password	String	是	密码（用于身份验证）

> ⚠ 文件路径通过 URL 动态指定。例如，上传的文件为 `upload.txt`，上传路径为 `/oss/upload/user/documents/file.txt`，则 `upload.txt` 会存储为 `user/documents/file.txt`。

4. **响应**：
   - **成功**：返回文件的下载链接。
     - **HTTP 状态码**：200 OK
     - **响应体**：`{ "downloadUrl": "http://<服务域名>/oss/download/<文件路径>" }`
   - **失败**：返回 HTTP 错误状态码及错误信息。
     - `401 Unauthorized`：身份验证失败。
     - `400 Bad Request`：对象键格式无效（如路径或文件名非法）。
     - `500 Internal Server Error`：文件上传失败。
5. **验证规则**：
   - 目录部分（路径中的每一层目录）只能包含字母、数字和 `-`。
   - 文件名部分只能包含字母、数字、`.` 和 `-`。
   - 不允许使用连续的斜杠（`//`）。

4.3 人机交互接口

使用图形用户界面（GUI）。以下是一些具体的特征和功能：

1. **可视化元素**：平台使用了图标、按钮、菜单和列表等图形元素，使用户能够通过点击和选择来操作。
2. **文件管理**：用户可以通过界面浏览、复制、剪切、删除和粘贴文件，这些操作通常通过图形按钮或右键菜单完成。
3. **数据展示**：平台以表格或列表的形式展示文件信息，如文件名、大小、日期等，用户可以直观地查看和管理数据。
4. **导航和操作**：用户可以通过图形界面导航到不同的目录或功能模块，如数据采集、3D建模、可视化环境等。
5. **交互反馈**：平台可能提供操作反馈，如确认对话框、进度条或状态提示，以增强用户体验。

6. **多任务处理**：用户可以在同一界面中同时进行多项操作，如查看数据、上传文件、下载文件等。

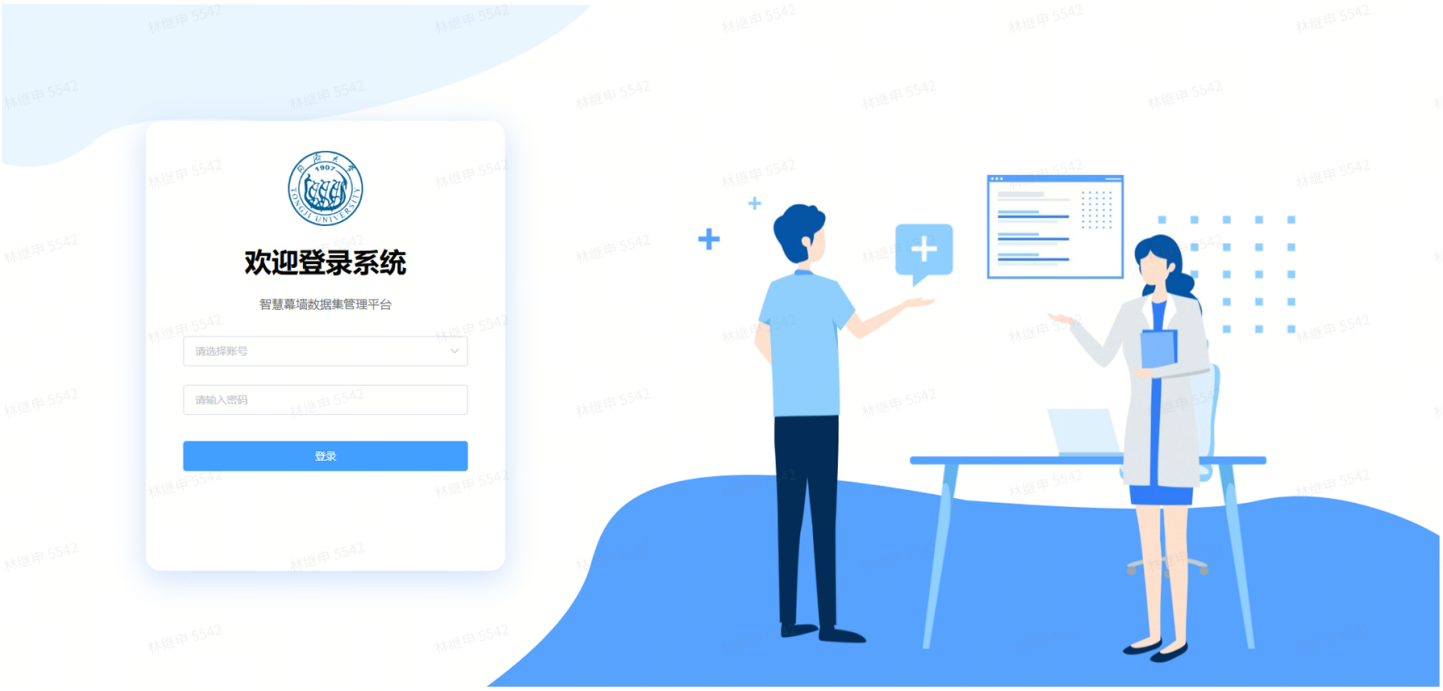


图 4-1 数据集管理平台登录页

图 4-2 数据集管理平台管理页

图 4-3 数据集管理平台控制台

图 4-4 数据集管理平台登录页图像预览示例 1

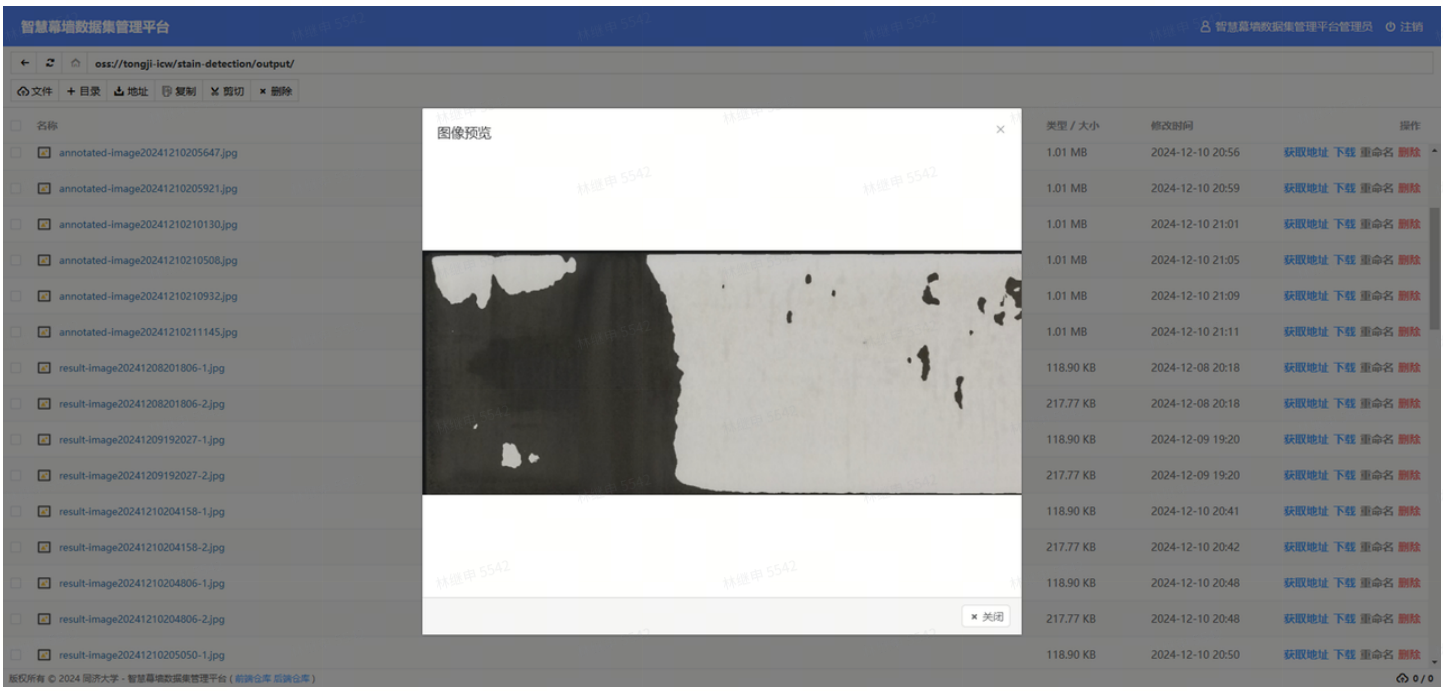


图 4-5 数据集管理平台登录页图像预览示例 2

5 安全性设计

5.1 权限控制

- **资源访问控制**：使用阿里云 RAM（Resource Access Management）进行细粒度的权限管理，确保只有授权用户或角色能够访问特定数据集和 API。
- **最小权限原则**：基于最小权限原则，系统仅为用户分配完成其任务所需的最低权限，避免权限过度分配。
- **角色管理**：系统支持多种用户角色（如管理员、普通用户等），每个角色拥有明确的权限定义，管理员可以通过后台管理系统为不同角色分配或修改权限。

- **权限校验**：在用户进行上传、下载、删除等操作时，系统会进行权限校验，确保只有符合权限要求的用户才能执行相应操作。

5.2 数据加密

- **传输加密**：数据在传输过程中使用 HTTPS 协议进行加密，确保数据在传输过程中不被窃取或篡改。
- **存储加密**：数据在存储过程中使用阿里云 OSS 提供的加密功能，确保数据在存储过程中的安全性，防止未授权访问。
- **敏感数据加密**：对用户信息、数据集元数据等敏感数据进行加密存储，确保数据的安全性。

5.3 日志审计

- **操作日志记录**：系统会记录用户的所有关键操作（如上传、下载、删除、权限变更等），日志内容包括操作时间、操作用户、操作类型、操作结果等。
- **日志存储**：日志存储在专门的日志服务器或云存储服务（如阿里云日志服务）中，确保日志的安全性和可扩展性。
- **日志查询与审计**：管理员可以通过日志管理系统查询系统中的操作日志，确保系统的可追溯性，便于审计和问题排查。
- **日志备份**：定期备份日志数据，确保日志的完整性和可恢复性。

6 性能设计

6.1 高并发处理

- **高并发支持**：系统应支持大规模数据集的上传和下载，确保在高并发情况下的性能稳定。
- **分布式存储**：使用阿里云 OSS 进行大规模数据存储，确保数据存储的高效性和可扩展性。
- **负载均衡**：通过负载均衡技术，将请求分发到多个服务器节点，确保系统在高并发情况下的稳定性。

6.2 响应时间优化

- **上传和下载优化**：上传和下载操作的响应时间应控制在合理范围内，确保用户体验。对于小于 1GB 的文件，响应时间不超过 3 秒；对于大于 1GB 的文件，响应时间不超过 10 秒。
- **缓存机制**：使用缓存机制（如 Redis）优化查询操作的响应时间，减少数据库查询压力，提升系统性能。
- **异步处理**：对于耗时较长的操作（如大规模数据上传等），系统采用异步处理机制，避免阻塞主线程，提升系统的响应速度。

7 可扩展性设计

7.1 模块化设计

- **前端模块化**：前端采用 Vue.js 框架，利用其组件化特性，将界面拆分为独立、可复用的组件，提升开发效率和代码可维护性。
- **后端模块化**：后端采用 Spring Boot 框架，按照功能或业务领域划分模块（如用户管理、数据集管理、权限控制等），每个模块独立开发和维护，确保系统的可扩展性和可维护性。
- **接口标准化**：前后端通过 RESTful API 进行交互，接口设计遵循清晰、高效的原则，便于后续功能的扩展和集成。

7.2 插件化设计

- **插件化架构**：系统支持插件化设计，便于后续功能的扩展和升级。新增功能模块时，尽量减少对现有系统的影响，确保系统的灵活性和可扩展性。
- **动态加载**：系统支持动态加载插件，用户可以根据需求启用或禁用特定功能模块，提升系统的灵活性和可定制性。

8 测试计划

8.1 单元测试

- **测试目标**：对各个模块进行单元测试，确保模块功能的正确性。
- **测试工具**：使用 JUnit、Mockito 等测试工具进行单元测试，覆盖核心业务逻辑。
- **测试内容**：包括数据集管理、权限控制、日志管理、备份与恢复等模块的功能测试。

8.2 集成测试

- **测试目标**：对前后端接口进行集成测试，确保接口的正确性和稳定性。
- **测试工具**：使用 Postman、Swagger 等工具进行接口测试，确保前后端交互的顺畅。
- **测试内容**：包括上传、下载、删除、权限校验等接口的功能测试和性能测试。

8.3 性能测试

- **测试目标**：对系统进行性能测试，确保系统在高并发情况下的性能稳定。
- **测试工具**：使用 JMeter、LoadRunner 等性能测试工具，模拟高并发场景下的系统表现。
- **测试内容**：包括上传、下载、查询等操作在高并发情况下的响应时间、吞吐量、资源利用率等性能指标。

9 部署与运维

9.1 部署方案

- **容器化部署**：系统部署在阿里云服务器上，使用 Docker 进行容器化部署，确保开发、测试和生产环境的一致性。
- **自动化部署**：通过 CI/CD 流水线实现代码的自动化构建、测试和部署，减少人工操作，提高部署效率。
- **环境一致性**：使用 Docker Compose 技术，确保开发、测试和生产环境的一致性，避免因环境差异导致的运行错误。

9.2 运维方案

- **定期维护**：定期进行系统维护，修复已知问题和优化系统性能，确保系统的稳定运行。
- **数据备份**：定期备份数据，确保数据安全与业务连续性。备份策略为每周一 00:00 自动执行一次全量备份，每个备份保留 30 天。
- **监控与告警**：实时监控系统运行状态，包括 CPU、内存、磁盘等资源使用情况。如果系统出现异常，及时发送告警通知给管理员。
- **版本管理**：使用 Git 进行版本管理，确保代码的可追溯性和可回滚性。每次发布新版本时，记录版本变更内容，便于后续维护和问题排查。

9.3 部署视图

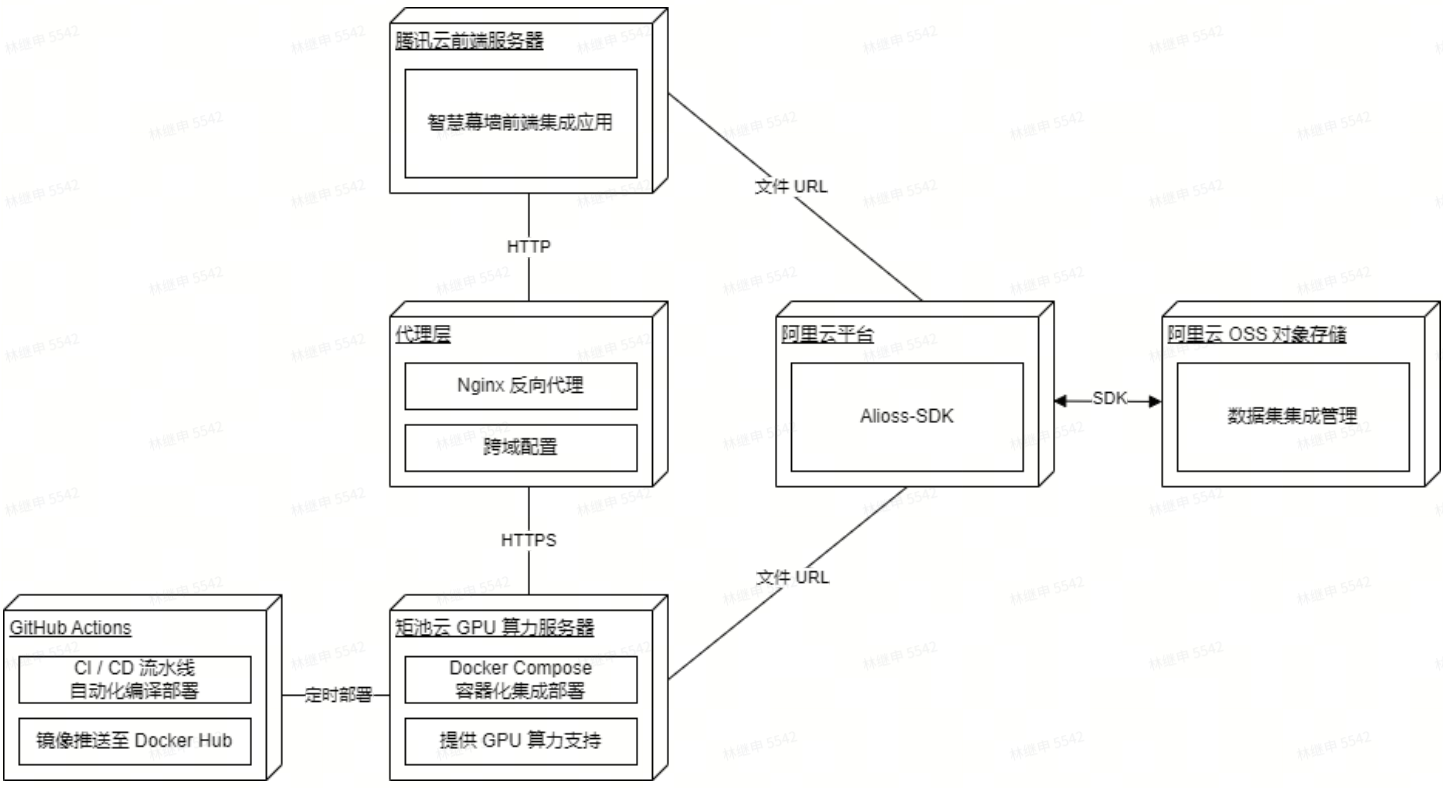


图 9-1 智慧幕墙系统部署视图

- **腾讯云前端服务器**
  - 前端应用的部署服务器，托管在腾讯云上。
  - 负责运行前端应用（如 Vue.js 构建的静态文件），用户通过浏览器访问该服务器来使用前端界面。
- **代理层（Nginx 反向代理）**
  - Nginx 作为反向代理服务器，负责处理前端请求的转发和负载均衡。
  - Nginx 还负责处理跨域请求，确保前端应用能够与后端服务进行安全的通信。
- **短池云 GPU 算力服务器**
  - 这是一个提供 GPU 算力的服务器，用于处理计算密集型任务（如模型训练、图像处理等）。
  - 通过 Docker Compose 进行容器化集成部署，确保开发、测试和生产环境的一致性。
  - 该服务器提供 GPU 加速，适合处理需要高性能计算的任务。
- **阿里云平台**
  - 使用阿里云的对象存储 SDK（Alioss-SDK）与阿里云 OSS 进行交互，实现文件的上传、下载和管理。
  - 阿里云 OSS 对象存储用于存储大规模数据集，确保数据的高效访问和持久化。
- **GitHub Actions**
  - 通过 GitHub Actions 实现持续集成和持续部署（CI/CD），自动化代码的构建、测试和部署流程。
  - 构建好的 Docker 镜像会被自动推送到 Docker Hub，便于后续的部署和使用。

10 总结

10.1 设计规约总结

本系统设计规约文档（SDS）详细描述了智慧幕墙数据集管理与运维系统的架构设计、模块设计、接口设计、安全性设计、性能设计、可扩展性设计以及测试与部署方案。通过对系统的全面设计分析，确保了系统能够按照需求规格说明书（SRS）中的要求实现，并具备高效、安全、可扩展的特性。

- **系统架构设计**
  - **前端层**：基于 Vue.js 框架，采用组件化设计，确保界面的模块化和可复用性。使用 Vue Router 进行页面路由管理，Vuex 进行状态管理。
  - **后端层**：基于 Spring Boot 框架，采用模块化设计，确保业务逻辑的独立性和可维护性。使用 Spring Security 实现权限控制，Spring AOP 实现日志记录。
  - **持久化层**：使用阿里云对象存储（OSS）作为数据存储服务，支持大规模数据的高效存储和快速访问。定期进行数据备份和恢复，确保数据安全。

- **模块设计**
  - **数据集管理模块**：支持数据集的上传、下载、删除、导出和管理操作，确保数据的高效管理和安全访问。
  - **权限控制模块**：通过阿里云 RAM 实现细粒度的权限管理，确保只有授权用户能够访问特定数据集。
  - **备份与恢复模块**：定期对数据集进行自动化备份，确保数据的安全性和业务连续性。支持数据恢复操作，确保在数据丢失或损坏时能够快速恢复。
  - **日志管理模块**：记录系统运行和部署的关键事件，支持日志查询、分析和清理，确保系统的可追溯性和可维护性。
- **接口设计**
  - **RESTful API**：提供标准化的数据上传、下载、查询接口，确保前后端交互的清晰和高效。接口设计遵循 RESTful 原则，使用标准的 HTTP 方法和状态码。
  - **安全性设计**：通过权限控制、数据加密和日志审计，确保数据的安全性和隐私性。使用 HTTPS 协议进行数据传输加密，阿里云 OSS 进行数据存储加密。
- **性能设计**
  - **高并发处理**：支持大规模数据集的上传和下载，确保在高并发情况下的性能稳定。使用分布式存储和负载均衡技术，提升系统的处理能力。
  - **响应时间优化**：优化上传和下载操作的响应时间，确保用户体验。使用缓存机制和异步处理，提升系统的响应速度。
- **可扩展性设计**
  - **模块化设计**：前端和后端均采用模块化设计，确保系统的可扩展性和可维护性。支持插件化架构，便于后续功能的扩展和升级。
  - **接口标准化**：前后端通过 RESTful API 进行交互，接口设计遵循清晰、高效的原则，便于后续功能的扩展和集成。
- **测试与部署**
  - **测试方案**：包括单元测试、集成测试和性能测试，确保系统的功能正确性和性能稳定性。使用 JUnit、Postman、JMeter 等工具进行测试。
  - **部署方案**：使用 Docker 进行容器化部署，确保开发、测试和生产环境的一致性。通过 CI/CD 流水线实现自动化部署，提高部署效率。

10.2 未来开发计划

- **数据集管理**
  - **增强数据分类与索引功能**：进一步优化数据分类和索引机制，提升数据查询效率。
  - **支持更多数据格式**：扩展系统支持的数据格式，满足更多业务需求。

- **数据生命周期管理**：实现数据的自动清理、归档和删除，优化数据生命周期管理。

- **项目运维**
  - **自动化运维工具**：引入更多自动化运维工具，提升运维效率。
  - **智能监控与告警**：实现智能监控和告警功能，及时发现和解决系统问题。
  - **资源动态调整**：支持服务器资源的动态调整，根据负载情况自动分配资源。
- **安全性**
  - **多因素认证**：引入多因素认证机制，提升系统安全性。
  - **安全审计**：定期进行安全审计，确保系统的安全性。
- **用户体验**
  - **用户界面优化**：持续优化用户界面，提升用户体验。
  - **用户反馈机制**：建立用户反馈机制，及时收集和处理用户意见。
- **技术升级**
  - **新技术引入**：持续关注新技术发展，适时引入新技术，提升系统性能。
  - **系统性能优化**：持续优化系统性能，确保系统在高负载情况下的稳定运行。