系统设计文档

知识图谱构建子系统设计文档

一、引言

1.1 项目背景与意义

中国拥有五千多年连续不断的文明历史，留下了丰富的历史文化遗产。然而，近代以来，因战争、掠夺、非法交易等原因，大量珍贵的中国文物流失海外，现分布于全球众多博物馆、图书馆、拍卖行及私人收藏中。这些文物作为中华文明的重要载体，不仅在历史研究、艺术鉴赏等领域具有重要价值，也在文化传播、国际交流与国家文化软实力建设中发挥着不可替代的作用。

为了实现海外中国文物的系统化管理、语义整合与知识服务，构建“海外藏中国文物知识管理与服务平台”成为亟需推进的数字工程。其中，知识图谱作为连接异构数据、表达语义关系、支持智能查询与推理的核心技术手段，是平台建设的关键支撑。

在此背景下，开发面向海外藏中国文物的知识图谱子系统，能够有效整合文物分布信息、文物本体属性及其历史文化背景，提升平台的数据组织能力和服务智能化水平，对于加强中华文化认同、促进文物回流保护、支持数字人文研究均具有重要意义。

1.2 子系统目标与功能定位

本子系统是“海外藏中国文物知识管理与服务平台”的重要组成部分，承担海外中国文物知识图谱的构建与管理任务。其核心目标在于通过自动化的信息抽取与语义建模技术，从权威数据源中采集和处理文物相关信息，建立面向语义关联、便于查询与分析的知识图谱，为上层管理系统、用户服务接口及数据可视化等模块提供知识支持。

本子系统主要包含以下功能：

- **文物数据采集**：从海外文博机构的官方网站或开放数据接口中获取结构化与非结构化的文物信息；
- **实体与关系抽取**：对文物描述信息进行语义分析，识别文物本体及其相关实体（如产地、年代、材质、现藏地等）与关系；

- **本体建模与图谱构建**：构建适用于海外中国文物领域的本体模型，并基于抽取得到的三元组数据构建知识图谱；
- **图数据库管理与查询接口**：实现知识图谱的持久化存储与多维查询功能；
- **图谱服务支撑**：为平台的知识展示、智能推荐、可视化分析等功能模块提供数据接口和知识支撑。

通过该子系统的建设，将显著提升平台的语义组织与知识服务能力，为实现海外文物“可知、可管、可用”的总体目标提供坚实的技术基础。

二、需求分析

本知识图谱子系统是“海外藏中国文物知识管理与服务平台”的一个子系统，主要负责从多个数据来源中提取文物信息，并构建结构化的知识图谱，方便后续的查询、展示和管理。

2.1 功能性需求

1. **数据采集功能**
    - 能够从海外博物馆的网站（如大英博物馆等）获取与中国文物相关的信息。
    - 支持自动化的数据采集，尽量减少人工操作。
2. **实体与关系抽取功能**
    - 能够识别文物的基本信息，如名称、朝代、材质、用途、藏馆等。
    - 能够提取文物之间的关系，例如“某文物属于某朝代”、“某文物藏于某博物馆”等。
3. **知识图谱构建功能**
    - 将抽取到的信息以图谱的形式保存下来，用“实体-关系-实体”的三元组方式表达。
    - 支持图谱的添加、修改和更新。
4. **查询与接口功能**
    - 用户可以通过关键词或条件查询文物信息。

- 支持通过接口把图谱数据提供给其他模块或系统使用。

5. **图谱服务功能**
   - 为平台提供可视化展示和后续推荐服务的数据支持。

2.2 非功能性需求

1. **操作简单**
   - 系统界面尽量简洁，使用方便，适合非技术人员使用。
2. **稳定性好**
   - 系统运行应尽量稳定，不容易出错，数据处理过程能自动记录日志。
3. **可扩展性**
   - 后期可以添加更多数据来源或扩展新的实体类型和关系。
4. **响应速度快**
   - 查询结果能在几秒内返回，不影响用户体验。
5. **安全性基本保障**
   - 系统接口应有限制，防止数据被恶意滥用。

三、系统架构设计

3.1 总体架构图

3.2 模块划分说明

本系统共划分为以下五个主要功能模块，各模块之间相互协作，共同完成从数据采集到知识图谱构建的全过程。

（1）数据爬取模块

该模块负责从指定的海外博物馆官方网站（如大英博物馆等）自动抓取与中国文物相关的信息。采集内容包括但不限于文物名称、图片、制作年代、描述信息、所属朝代等。为了提升数据的覆盖面和准确性，系统设计了灵活的爬虫策略，支持结构化页面和动态加载页面的抓取，确保数据采集的全面性。

（2）数据清洗模块

该模块对原始抓取数据进行清洗和规范化处理，使其符合数据库存储格式要求。主要操作包括：

- 筛选所需数据列并进行字段重命名；
- 清理重复值和空白数据；
- 统一数据格式（如时间格式、编号、单位等）；
- 填补缺省值或进行默认处理。该模块保证了数据的整洁性和一致性，为后续建模打下基础。

（3）数据补充模块

考虑到部分博物馆网站信息不完整，系统设计了数据补充机制。通过自动方式（调用 deepseek 接口）的方式，对缺失字段进行完善，如文物出处、用途、材质等。该模块提高了数据的完整性，确保每条文物记录都具备较高的信息质量。

（4）数据建模模块

该模块将清洗并补充后的文物信息转化为标准的三元组形式，即“实体-关系-实体”的结构表示（如“唐三彩马”-属于-“唐代”）。在建模过程中，系统会进一步对数据进行规范化处理，如统一命名规则、对实体进行消歧、规范关系类型等。最终形成可用于构建知识图谱的结构化数据集。

（5）数据存储模块

本模块负责将处理后的数据进行持久化保存。存储方式包括：

- 使用 MySQL 数据库存储清洗后的结构化数据，便于数据管理和备份；

- 使用 Neo4j 图数据库存储知识图谱三元组数据，支持复杂关系的高效查询与可视化展示。图数据库为知识图谱的检索和分析提供了强有力的支持，是系统的核心数据支撑平台。

3.3 技术选型

为了实现海外中国文物知识图谱构建子系统的各项功能，系统采用了主流且易于开发维护的开源技术。以下是各模块对应的技术选型说明：

模块名称	技术选型	说明
数据爬取模块	Python+Selenium	Selenium 用于抓取动态页面，
数据清洗模块		支持高效的数据处理、清洗、去重、填补缺失等操作
数据补充模块		可通过 API 补全缺失信息，也支持人工填充
数据建模模块		结合实体识别与规则抽取，构造“实体-关系-实体”三元组
数据存储模块	MySQL + Neo4j	MySQL 用于结构化数据存储，Neo4j 用于图谱三元组存储
接口服务（扩展）		
可视化展示（调用端）		

知识服务子系统系统设计文档

一、引言

1.1 目的

本项目旨在开发一个海外文物知识服务子系统。本说明文档用于明确用户对该系统的实际需求，并确立一套完整、准确、清晰、具体的系统需求及设计方案。此文档不仅为开发人员实现各模块及功能提供指导，也帮助用户全面了解系统的设计与功能。该文档详细记录了用户对系统的功能和性能等方面的具体需求，可作为用户需求确认和系统总体设计的依据，同时也用于后续系统的验证和维护。

1.2 背景

建立海外藏中国文物信息数据库及海外文物知识服务子系统，有助于全面掌握并直观展示海外藏中国文物的信息，并与国内文物信息结合，形成较为完整的中国历史文物资料体系。这不仅能够更好地支持中国历史研究，也为未来的文物追索、征集和保护工作提供有力支持。

二、任务概述

2.1 目标

开发一个基于 Web 的系统，利用知识图谱构建子系统所采集的数据，提供数据的浏览、查询与可视化等功能服务。

2.2 用户特点

系统面向广大海外藏文物爱好者及志愿者用户。目标用户为普通电脑用户，不需要专业技术背景。系统提供图形化操作界面，使用便捷，只需具备基本的上网知识即可顺利使用。

三、总体设计

3.1 功能

1. **数据浏览**：支持多种形式展示的浏览功能。①提供基本的筛选、排序功能，可按照文物类型、文物年代、博物馆等多种基础信息进行索引、筛选、排序方式浏览文物信息以方便用户的使用。②提供查看文物详情功能，显示文物的详细数据，如文本、图像等信息，点击文物图片，可以进行放大缩小。③相关文物推荐功能，在该文物页面显示相关文物，相关规则自定，如相似主题、同一作者、图像内容相似等。

2. **数据查询**：支持文物的简单查询功能和高级查询功能。简单查询根据输入的关键字，如文物名称、博物馆名称、文物年代等进行查询。高级查询可以对文物的多个字段进行限定查询。

3. **数据可视化显示**：支持两种可视化。①包含结点、边的力导向图知识图谱展示。②文物时间轴：按照时间轴的方式、展示各个时段的文物信息、时间等信息。

4. **用户个人信息管理**：用户可以注册录该系统，设置用户名密码、邮箱等个人信息，并且可以看到点赞、评论、收藏过的展品。

3.2 技术架构

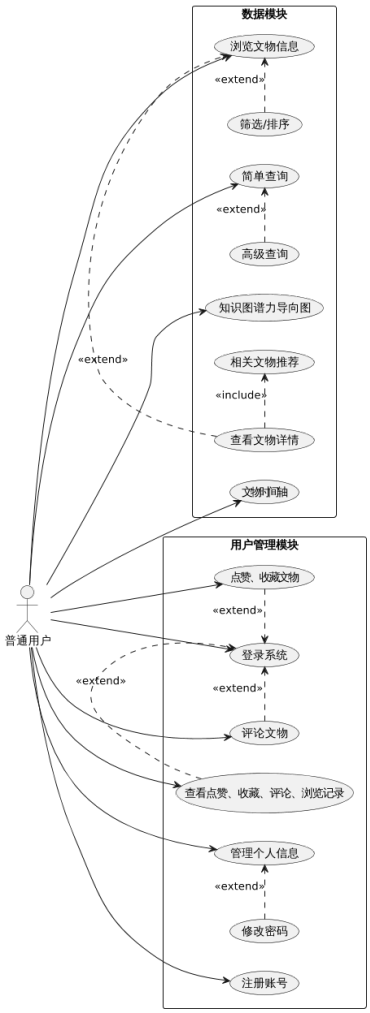
组件	技术选型
后端框架	Flask 2.0.1
数据库	MySQL 8.0 + Neo4j 4.4
缓存	Flask-Caching
跨域处理	Flask-CORS

图像处理	Pillow 9.5.0
驱动	pymysql + neo4j-driver

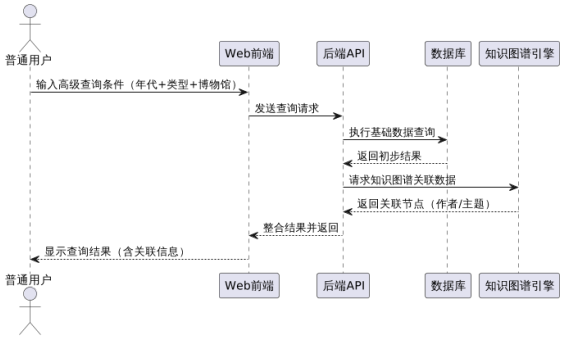
环境要求：

- Python 3.8+
- MySQL 8.0+
- Neo4j 4.4+
- 内存：4GB+（建议 8GB）

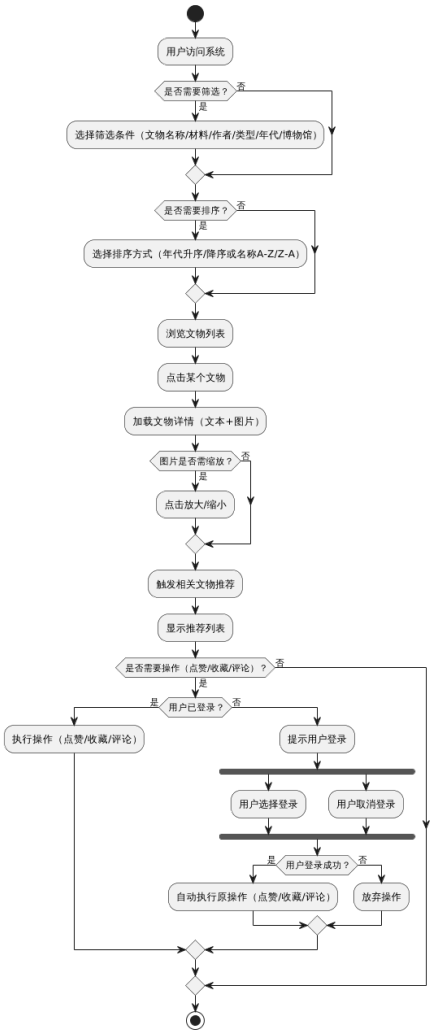
3.3 子系统用例图



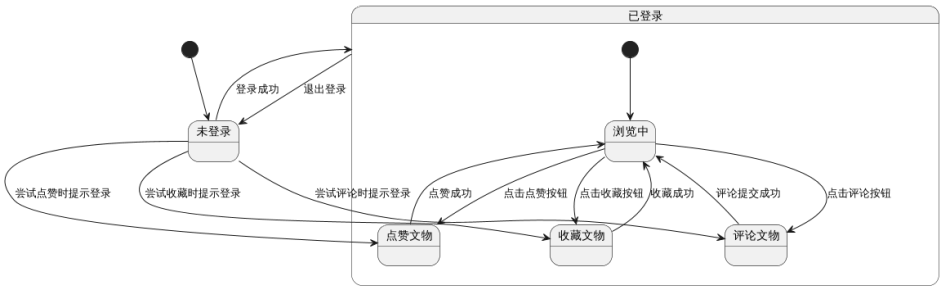
3.4 子系统搜索功能交互序列图



3.5 子系统文物展示、详情功能活动图



3.6 点赞、收藏、评论功能状态图



四、接口设计

4.1 用户管理模块接口设计

4.1.1 用户登录接口

接口说明

- 接口地址：`/login`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON
- 认证方式：无（登录后返回用户信息）

请求参数

参数名	类型	是否必须	说明
phone_number	string	是	用户手机号码
password	string	是	用户密码

成功响应

```
{
  "status": "success",
  "message": "登录成功",
  "username": "张三",
  "user_id": "123456"
}
```

4.1.2 用户注册接口

接口说明

- 接口地址：`/register`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：用户提交注册信息，完成注册流程。

请求参数

参数名	类型	是否必须	描述	格式或规则
username	string	是	用户名	非空
password	string	是	密码	长度不少于 6 位
confirmpass	string	否	确认密码（前端校验）	与 password 一致，前端校验
id_number	string	是	身份证号	中国大陆 18 位身份证格式

phone_number	string	是	手机号	以 13-19 开头的 11 位手机号码

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "注册成功"
}
```

4.1.3 个人信息获取接口

接口说明

- 接口地址：`/user_info/<int:user_id>`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：根据用户 ID 获取用户的基本信息及相关数据（收藏、点赞、评论、浏览记录）。

请求参数

参数名	类型	必填	说明
user_id	string	是	用户唯一标识

成功响应

- 返回示例

```
{
  "user_info": {
    "name": "张三",
    "phone_number": "18112340000",
    "id_number": "320112********1234",
    "gender": 1,
    "age": 28,
    "address": "北京市朝阳区",
    "wechat": "zhangsan_wechat",
    "qq": "12345678",
    "description": "这是简介"
  },
  "favorites": {...},        // 用户收藏信息
  "likes": {...},            // 用户点赞信息
  "comments": {...},         // 用户评论信息
  "browsing_history": {...} // 用户浏览记录
}
```

4.1.4 用户头像上传接口

接口说明

- 接口地址：`/upload_avatar/<int:user_id>`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON

请求参数

参数名	类型	必填	说明

user_id	string	是	用户唯一标识
file	file	是	上传的头像文件

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "头像上传成功"
}
```

4.1.5 用户密码修改接口

接口说明

- 接口地址：`/update_password`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：修改用户密码

请求参数

参数名	类型	必填	说明
user_id	string	是	用户唯一标识
new_password	string	是	新密码（长度≥6）

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "密码修改成功"
}
```

4.1.6 用户信息编辑接口

接口说明

- 接口地址：`/update_user_info`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：更新用户的个人信息

请求参数

参数名	类型	必填	说明
user_id	string	是	用户唯一标识
name	string	是	用户名
description	string	否	用户简介
gender	integer	否	性别（0 女，1 男）

address	string	否	地址
age	integer	否	年龄
wechat	string	否	微信号
qq	string	否	QQ 号

响应示例：

```
{
  "status": "success",
  "message": "用户信息更新成功"
}
```

4.2 文物展示模块接口设计

4.2.1 文物搜索接口

接口说明

- 接口地址：`/search`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：根据用户选择的标签和搜索条件，分页返回符合条件的文物数据列表，支持排序、过滤和高级搜索。

请求参数

参数名	类型	必填	说明
q	string	否	普通搜索关键字（来自“搜索：xxx”标签）
sort	string	否	排序方式，支持选项： - 时间：新-旧 - 时间：旧-新 - 名称：A-Z - 名称：Z-A
condition	string	否	限定搜索字段，支持选项：仅限作者、仅限标题、仅限描述、仅限类型、仅限朝代、仅限材料、仅限尺寸
author	string	否	按作者筛选，来自“作者：xxx”标签
name	string	否	按名称筛选，来自“名称：xxx”标签
museum	string	否	按博物馆筛选，来自“博物馆：xxx”标签
type	string	否	按类型筛选，来自“类型：xxx”标签

dynasty	string	否	按朝代筛选，来自“朝代：xxx”标签
matrials	string	否	按材料筛选，来自“材料：xxx”标签
after	string (时间)	否	起始时间筛选，格式示例：2020、-500 (BCE 表示为负数)
before	string (时间)	否	结束时间筛选，格式示例同上
popular	array[string]	否	热门标签数组，例如：["仰韶文化", "作者不详"]
page	integer	是	页码，分页参数
page_size	integer	是	每页返回数据数量

请求示例

```
GET /search?q=青铜器&sort=时间：新-旧&author=李四&popular=仰韶文化&popular=纸本水墨
&page=1&page_size=20
Host: localhost:5000
```

成功响应

- 返回参数

参数名	类型	说明
results	array	符合搜索条件的文物列表

- 单个文物对象结构示例

```
{
  "id": 123,
  "name": "青铜鼎",
  "author": "作者不详",
  "museum": "故宫博物院",
  "type": "青铜器",
  "dynasty": "商朝",
  "materials": "青铜",
  "description": "商代青铜器，纹饰精美",
  "date": "公元前 1200 年",
  "image_url": "http://example.com/image.jpg",
  "video_url": null
}
```

- 错误码示例

HTTP 状态码	错误说明	说明
500	服务器内部错误	服务器处理请求出错
400	请求参数错误	参数格式或内容不合法
404	路径不存在	请求的接口地址错误

4.2.2 搜索建议接口

接口说明

- 接口地址：`/search-suggestions`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：获取搜索框自动填充的建议词（朝代、材质、博物馆等维度）

请求参数：无

成功响应

- 返回示例

```
{
  "朝代": ["唐朝", "宋朝", ...],
  "材质": ["青铜", "陶瓷", ...],
  "博物馆": ["故宫博物院", "上海博物馆", ...],
  "类型": ["青铜器", "书画", ...],
  "名称": ["四羊方尊", "清明上河图", ...],
  "作者": ["张择端", "王羲之", ...]
}
```

4.2.3 文物详情及相关推荐接口

接口说明

- 接口地址：`/api/detail_inform`
- 请求方式：GET

- 请求格式：JSON
- 响应格式：JSON
- 功能说明：获取指定文物的详细信息（基础属性、关联视频、相关推荐等）

请求参数

参数名	类型	必传	描述
relic_id	Number	是	文物唯一标识 ID

成功响应

- 返回示例

```
{
  "status": "success",
  "img_url": "http://localhost:5000/static/relic/123.jpg",
  "relic_id": 123,
  "relic_inform": {
    "name": "青花瓷瓶",
    "author": "明代官窑",
    "dynasty": "明朝",
    "matrials": "陶瓷",
    "type": "瓷器",
    "size": "高 30cm",
    "description": "明代典型青花瓷瓶...",
    "likes_count": 120,
    "views_count": 5000
  },
  "namelist": [/* 主题相关文物列表 */],
  "authorlist": [/* 作者相关文物列表 */],
  "dynastylist": [/* 朝代相关文物列表 */],
  "rand_list": [/* 随机推荐文物列表 */],
  "video_data": [
```

```
    {
      "video_id": 1,
      "video_url": "http://localhost:5000/video/123.mp4",
      "title": "文物修复纪录片"
    }
  ]
}
```

4.2.4 收藏点赞记录获取接口

接口说明

- 接口地址：`/api/get_thumsbup`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：查询当前用户对指定文物的点赞、收藏状态

请求参数

参数名	类型	必传	描述
relic_id	Number	是	文物 ID
user_id	String	是	用户 ID（登录后获取）

成功响应

- 返回示例

```
{
  "islike": true,  // 是否已点赞
  "isfav": false   // 是否已收藏（与收藏接口共用）
}
```

4.3 文物交互模块接口设计

4.3.1 文物评论获取接口

接口说明

- 接口地址：`/get/comments`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：获取指定文物的评论列表（含子评论）

请求参数

参数名	类型	必传	描述
relic_id	Number	是	文物 ID

成功响应

- 返回示例

```
[
  {
    "comment_id": 1,
    "user_id": "u123",
    "name": "用户 A",
```

```
    "content": "很有历史感！",
    "images": ["http://localhost:5000/comment/1.jpg"],
    "reply_count": 2,
    "children": [
      {
        "comment_id": 101,
        "user_id": "u456",
        "name": "用户 B",
        "content": "同意！"
      }
    ]
  }
]
```

4.3.2 文物评论提交接口

接口说明

- 接口地址：`/user/comment`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：用户发表评论（支持文本+图片）

请求参数

参数名	类型	必传	描述
text	String	是	评论文本内容
relic_id	Number	是	文物 ID

user_id	String	是	用户 ID（登录后获取）
images	File Array	否	评论图片（最多 9 张）

成功响应

```
{
  "status": "success",
  "message": "评论提交成功，请等待审核"
}
```

4.3.3 文物评论回复提交接口

接口说明

- 接口地址：`/user/reply`
- 请求方式：POST
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：用户回复某条评论（仅文本）

请求参数

参数名	类型	必传	描述
parent_id	Number	是	被回复的评论 ID
user_id	String	是	用户 ID

content	String	是	回复内容

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "回复成功，请等待审核"
}
```

4.3.4 点赞提交接口

接口说明

- 接口地址：`/api/put_like/<int:relic_id>`
- 请求方式：PUT
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：更新文物点赞计数并记录用户点赞状态

请求参数

参数名	类型	必传	描述
likes_count	Number	是	最新点赞数（前端计算后传递）
user_id	String	是	用户 ID

islike	Boolean	是	操作后状态（true: 点赞/false: 取消）

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "点赞状态更新成功"
}
```

4.3.5 收藏提交接口

接口说明

- 接口地址：`/api/put_Fav/<int:relic_id>`
- 请求方式：PUT
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：更新用户对文物的收藏状态

请求参数

参数名	类型	必传	描述
user_id	String	是	用户 ID
museum_id	Number	是	文物所属博物馆 ID

is_fav	Boolean	是	操作后状态（true: 收藏/false: 取消）

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "收藏状态更新成功"
}
```

4.3.6 浏览记录更新接口

接口说明

- 接口地址：`/api/put_view/<int:relic_id>`
- 请求方式：PUT
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：用户访问文物详情时增加浏览计数

请求参数

参数名	类型	必传	描述
views_count	Number	是	最新浏览数（前端计算后传递）
user_id	String	否	用户 ID（未登录时

			可不传）

成功响应

- 返回示例

```
{
  "status": "success",
  "message": "浏览计数更新成功"
}
```

4.4 知识图谱可视化模块接口设计

4.4.1 知识图谱数据获取接口

接口说明

- 接口地址：`/graph-data`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：根据关键词返回知识图谱的节点和连边数据，前端使用 D3 进行可视化展示，支持关键词模糊搜索。

请求参数

参数名	类型	必填	说明
keyword	string	否	搜索关键词，模糊匹配图谱内容

请求示例

```
GET /graph-data?keyword=古代建筑
Host: localhost:5000
```

成功响应

- 返回参数

参数名	类型	说明
nodes	array	节点数组，每个节点表示图谱中的一个实体
links	array	连边数组，表示节点之间的关系

- nodes 结构说明

字段名	类型	说明
id	string	节点唯一标识
label	string	节点标签类别（类型）
name	string	节点名称
properties	object	节点属性键值对，包含详细信息

- links 结构说明

字段名	类型	说明

source	string	起始节点 ID
target	string	目标节点 ID
type	string	关系类型（连边名称）

- 返回示例


```
"nodes": [
  {
    "id": "1",
    "label": "建筑",
    "name": "故宫",
    "properties": {
      "位置": "北京",
      "年代": "明清",
      "简介": "中国古代宫殿建筑群"
    }
  },
  {
    "id": "2",
    "label": "朝代",
    "name": "明朝",
    "properties": {
      "开始时间": "1368",
      "结束时间": "1644"
    }
  }
],
"links": [
  {
    "source": "1",
    "target": "2",
    "type": "所属朝代"
  }
```

```
]
}
```

4.5 时间线可视化模块接口设计

4.5.1 时间线数据获取接口

接口说明

- 接口地址：`/timeline-data`
- 请求方式：GET
- 请求格式：JSON
- 响应格式：JSON
- 功能说明：查询文物入库时间（ entry_time ）非空的文物数据，返回时间线事件所需的结构化信息。

请求参数：无

成功响应

- 返回示例

```
[
  {
    "name": "后母戊鼎",
    "description": "商后期（约前十四世纪至前十
    一世纪）铸品，原器 1939 年 3 月出土于河南安阳",
    "year": "1939",
    "image": "/static/relic_images/1.jpg",
    "museum": "中国国家博物馆",
    "type": "青铜器",
    "dynasty": "商代",
```

```
    "likes_count": 1234,
    "views_count": 5678,
    "size": "高 133 厘米、口长 110 厘米、口宽 79 厘
    米",
    "matrials": "青铜",
    "author": "商王室"
  },
  {
    "name": "《清明上河图》",
    "description": "北宋风俗画，中国十大传世名
    画之一",
    "year": "1950",
    "image": null,
    "museum": "北京故宫博物院",
    "type": "书画",
    "dynasty": "北宋",
    "likes_count": 5678,
    "views_count": 9012,
    "size": "宽 24.8 厘米、长 528.7 厘米",
    "matrials": "纸本水墨",
    "author": "张择端"
  }
]
```


五、运行设计

5.1 核心功能模块

5.1.1 用户认证模块

- MD5 密码加密
- 头像上传处理（PNG 格式转换）

5.1.2 知识图谱模块

- 节点关系可视化
- 关键词模糊搜索
- 缓存优化策略

5.1.3 文物管理模块

- 多维复合搜索（支持 9 个维度）
- 时间线展示
- 热度统计（浏览/点赞）

5.1.4 交互功能模块

- 浏览历史记录
- 点赞/收藏联动更新
- 评论系统

5.2 安全设计

1. **密码存储**：MD5 哈希加密
2. **文件上传限制**：

- 仅允许图像格式
- 强制格式转换
- 独立存储目录

5.3 运行模块组合

5.3.1 请求处理流程

5.3.2 注册

5.3.3 浏览

5.3.4 搜索

5.3.5 修改个人信息

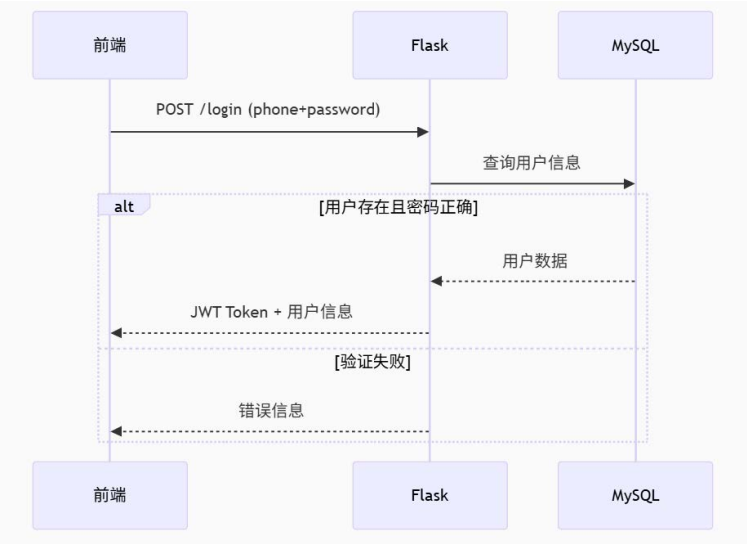
5.4 系统架构

5.4.1 系统架构图

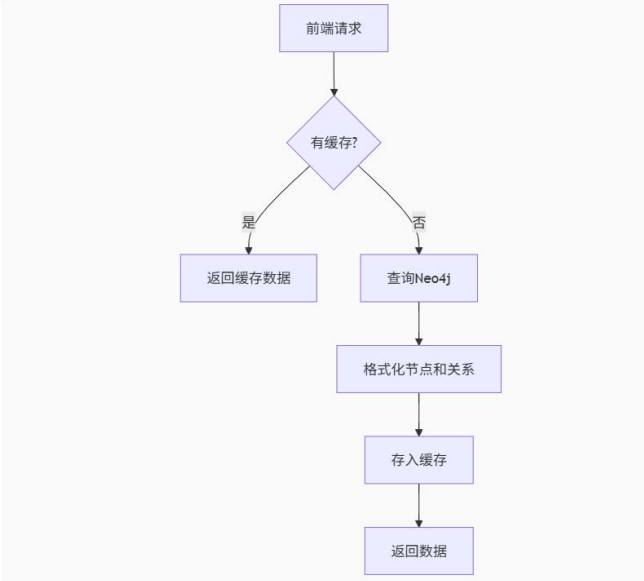
[MISSING IMAGE: ,]

5.4.2 核心操作流程图

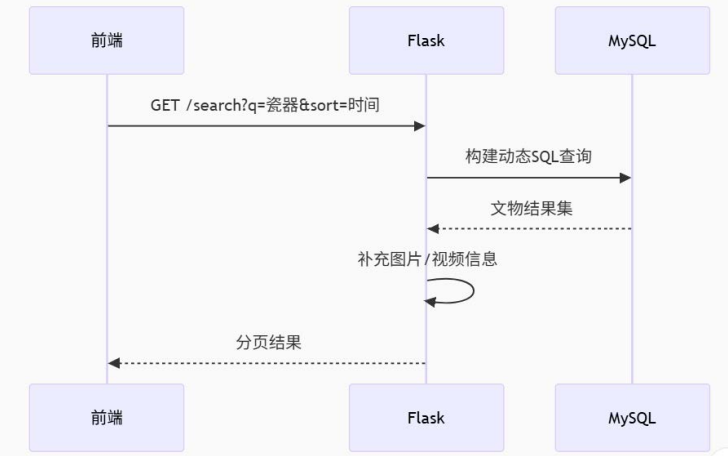
用户认证流程



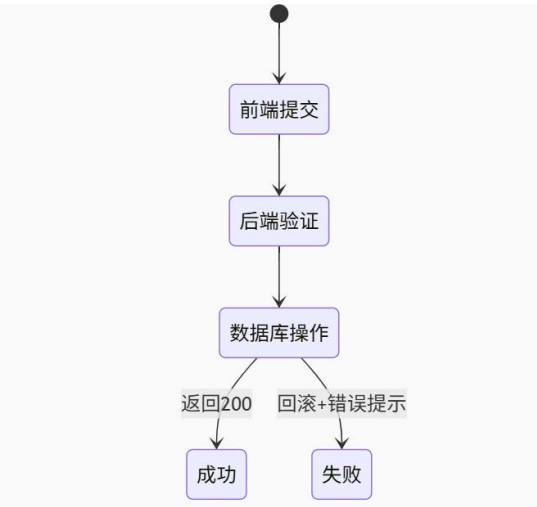
知识图谱查询流程



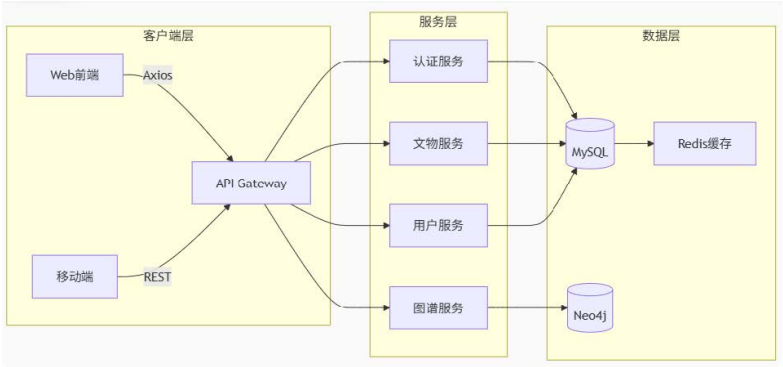
文物搜索流程



数据更新流程



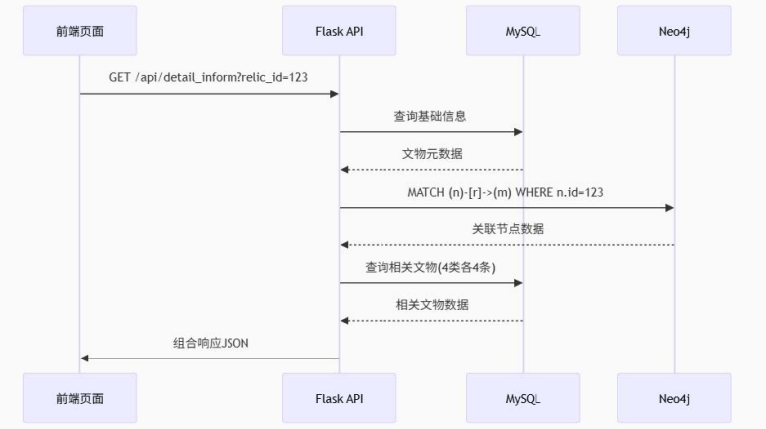
5.4.3 分层架构详细设计



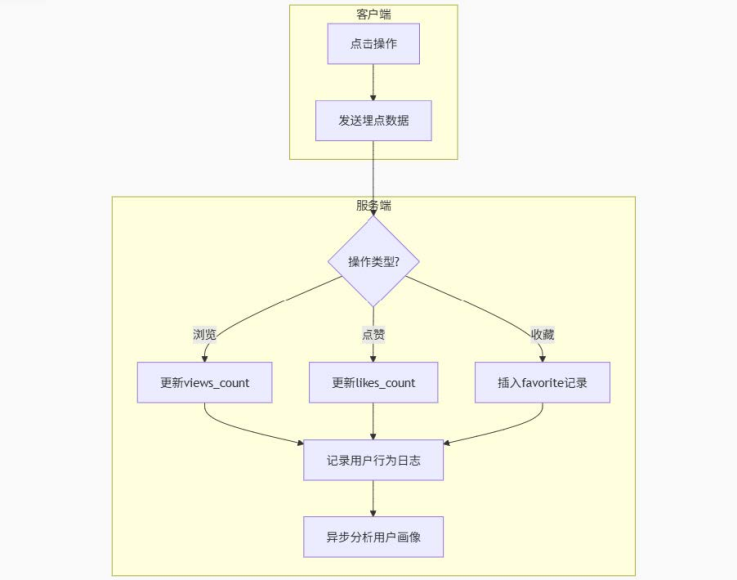
1. **业务服务：**
    - 认证服务：处理`/login`、`/register`
    - 图谱服务：封装 Neo4j 的 Cypher 查询
    - 文物服务：处理文物和搜索
    - 用户服务：管理用户数据和行为
2. **数据访问层：**
    - MySQL：通过 PyMySQL 连接池管理
    - Neo4j：官方 Driver 实现会话池
    - 缓存：使用 Flask-Caching 模式

5.4.4 关键业务流程时序图

文物详情页加载流程



用户行为数据流



六、性能优化设计

采用缓存策略：

- 热点数据：`@cache.memoize(timeout=120)`
- 文物详情页：静态化处理
- 图谱数据：预生成常用查询结果

七、运行控制

7.1 注册以及登录

1. 注册时用户会提供用户名、密码。用户的所有个人信息存于网站数据库中，以用于登陆，查询等。
2. 用户名任意。
3. 注册时需设定任意密码，用户自行设置。

7.2 修改客户注册信息和忘记密码

1. 用户注册成功后，可以修改信息。
2. 如果用户密码遗失，无法进行登陆操作，可以重新注册账号

7.3 搜索和查询文物

1. 搜索和查询文物详情在用户登陆状态或者游客状态均有此权限进行操作。
2. 搜索文物时，可以根据关键字搜索，也可以根据文物的年限搜索

7.4 运行时间

1. PC 端访问使用浏览器，网页代码不占用客户端时间，客户端和服务端通信占用网络传输时间，服务端对数据库操作占用服务器 CPU 时间。
2. 添加展示页面懒加载，提供更快的页面响应。

八、系统错误处理设计

8.1. 错误分类

8.1.1 数据库错误

- **连接错误**：MySQL/Neo4j 连接失败
- **查询错误**：SQL 语法错误、查询超时
- **事务错误**：提交/回滚失败
- **数据不存在**：查询结果为空

8.1.2 业务逻辑错误

- **参数验证错误**：缺少必要参数、参数格式错误
- **权限错误**：未授权访问
- **资源冲突**：重复注册、重复操作

8.1.3 外部服务错误

- 文件系统错误（头像上传）
- 图片处理错误
- 缓存服务错误

8.1.4 系统级错误

- 内存溢出
- 未知异常

8.2 错误响应规范

8.2.1 标准错误格式

```
{
  "status": "error",
  "message": "简明错误信息",
  "error_code": "可选错误代码",
  "details": "可选详细信息"
}
```

8.2.2 HTTP 状态码使用

- `400 Bad Request`：客户端参数错误
- `401 Unauthorized`：未授权
- `403 Forbidden`：禁止访问
- `404 Not Found`：资源不存在
- `500 Internal Server Error`：服务器内部错误

知识问答子系统系统设计文档

一、需求分析

1.1 项目需求背景

随着博物馆数字化转型的深入推进，访客期望获取更加便捷、个性化的知识服务。传统的参观方式（如讲解员讲解、展板阅读）存在时间和空间限制，难以满足现代参观者对深度内容和交互体验的需求。博物馆知识问答子系统旨在利用人工智能技术，打破这些限制，为参观者和研究人员提供高效、准确的知识获取渠道。

1.2 用户需求分析

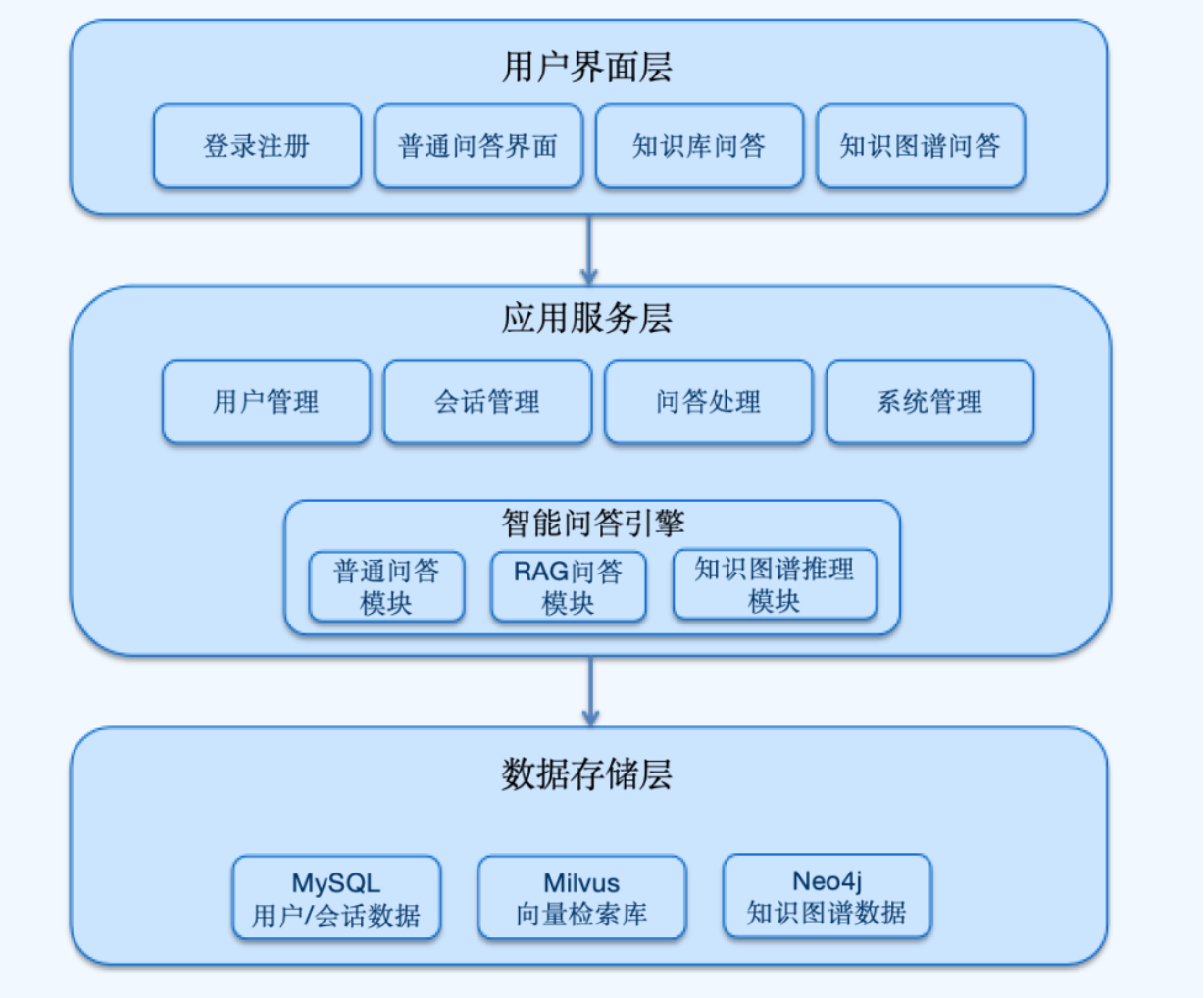
调研显示，用户希望通过系统便捷获取展品背景、历史文化信息，获得个性化推荐，并支持深入查询和实体关联。同时系统应具备准确答疑、内容可信、界面友好、使用简单等特点，可辅助讲解、减少重复工作，并通过数据分析优化内容与服务。

1.3 功能需求列表

系统需具备用户登录、会话管理和通用问答等基础功能，支持基于知识库和知识图谱的精准问答、多轮对话与流式响应，同时拓展多媒体融合、个性化知识推荐、多语言支持和数据分析能力，以满足多类用户的智能交互需求。

二、系统架构设计

2.1 总体架构图



2.2 模块划分说明

系统采用三层架构，包括用户界面层（提供登录注册、通用问答、知识库和图谱问答等交互界面）、应用服务层（负责用户管理、会话处理、问答生成），以及数据存储层（整合 MySQL、Milvus、Neo4j 多种数据库，支持结构化、向量化与图谱数据存储与检索）。

2.3 技术选型

系统前端采用 Vue 3 搭配 Vue Router 和 Axios，后端基于 FastAPI 与 SQLAlchemy，支持 JWT 认证与异步流式问答。数据层使用 MySQL、Milvus 与 Neo4j，部署方面通过 Docker 实现容器化。

三、数据库设计

3.1 MySQL 数据库设计

3.1.1 用户表（user）

字段名	数据类型	约束	说明
user_id	INT	PK, AUTO_INCREMENT	用户 ID
phone_number	VARCHAR(20)	NOT NULL	电话号码
password	VARCHAR(32)	NOT NULL	密码（加密存储）
id_number	VARCHAR(18)	NOT NULL	身份证号
name	VARCHAR(20)	NOT NULL	用户名
description	VARCHAR(255)	NULL	用户简介
gender	SMALLINT	NULL	用户性别，0：女，1：男
age	SMALLINT	NULL	年龄
address	VARCHAR(255)	NULL	地址
wechat	VARCHAR(50)	NULL	微信号
qq	VARCHAR(50)	NULL	QQ 号
account_status	SMALLINT	NOT NULL, DEFAULT 1	用户状态，0:

			禁用,1:启用
role_type	SMALLINT	NOT NULL, DEFAULT 0	用户角色, 0:用户,1:管理员
create_time	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间
update_time	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE	更新时间

3.1.2 聊天会话表（chat_sessions）

字段名	数据类型	约束	说明
id	INT	PK, AUTO_INCREMENT	会话 ID
title	VARCHAR(100)	DEFAULT '新对话'	会话标题
user_id	INT	NOT NULL	所属用户 ID
type	SMALLINT	DEFAULT 1	会话类型：1(普通)、2(知识库)、3(知识图谱)
is_active	BOOLEAN	DEFAULT TRUE	是否活跃
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE	更新时间

3.1.3 聊天消息表（chat_messages）

字段名	数据类型	约束	说明
id	INT	PK, AUTO_INCREMENT	消息 ID
session_id	INT	FK, NOT NULL	所属会话 ID
content	TEXT	NOT NULL	消息内容
is_user	BOOLEAN	DEFAULT TRUE	是否为用户消息
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间

3.1.4 知识库表（knowledge_bases）

字段名	数据类型	约束	说明
id	INT	PK, AUTO_INCREMENT	知识库 ID
name	VARCHAR(100)	NOT NULL	知识库名称
description	TEXT	NULL	知识库描述
status	SMALLINT	DEFAULT 1	状态：0(禁用)，1(启用)

created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE	更新时间

3.2 Milvus 向量数据库设计

3.2.1 博物馆知识向量集合（museum_knowledge）

字段名	数据类型	说明
id	INT64	主键 ID
content	VARCHAR	知识内容文本
embedding	FLOAT_VECTOR	文本向量表示（维度由模型决定，如 768 或 1536）
source	VARCHAR	知识来源
category	VARCHAR	知识类别
metadata	JSON	额外元数据信息

索引设计：

- 向量索引类型：HNSW（层次可导航小世界图）
- 距离度量：余弦相似度（Cosine Similarity）
- 搜索参数：ef=64, nprobe=16（可根据性能调整）

3.3 Neo4j 图数据库设计

3.3.1 节点类型

1. **展品（Exhibit）**
    - 属性：id, name, creation_date, description, location, material, dimension, image_url
2. **人物（Person）**
    - 属性：id, name, birth_date, death_date, nationality, biography, era
3. **朝代/时期（Era）**
    - 属性：id, name, start_year, end_year, description
4. **地点（Place）**
    - 属性：id, name, location, description, modern_name
5. **文化（Culture）**
    - 属性：id, name, time_period, region, description
6. **分类（Category）**
    - 属性：id, name, description, parent_category

3.3.2 关系类型

1. **创作（CREATED_BY）**
    - 连接：Exhibit → Person
    - 属性：year, attribution_certainty
2. **出土于（EXCAVATED_FROM）**

- 连接：Exhibit → Place
- 属性：excavation_date, excavation_team

3. **属于（BELONGS_TO）**
   - 连接：Exhibit → Category
   - 属性：certainty
4. **创作于（CREATED_DURING）**
   - 连接：Exhibit → Era
   - 属性：certainty
5. **来源于（ORIGINATED_FROM）**
   - 连接：Exhibit → Culture
   - 属性：certainty
6. **居住于（LIVED_IN）**
   - 连接：Person → Place
   - 属性：start_year, end_year
7. **生活于（LIVED_DURING）**
   - 连接：Person → Era
   - 属性：certainty

3.4 ER 图

3.5 数据流说明

1. **用户注册/登录流程**
   - 用户提交注册信息 → 验证信息有效性 → 加密密码 → 存入 user 表 → 生成 JWT 令牌 → 返回给用户
   - 用户提交登录信息 → 验证凭证 → 生成 JWT 令牌 → 返回给用户
2. **聊天会话创建流程**
   - 用户请求创建会话 → 验证用户身份 → 创建会话记录 → 返回会话信息
   - 会话类型确定使用的问答模式（普通/知识库/图谱）
3. **问答流程**
   - 普通问答：用户提问 → 保存至消息表 → 组织上下文 → LLM 生成回答 → 保存回答 → 返回用户
   - 知识库问答：用户提问 → 向量化 → Milvus 检索相关知识 → 增强提示 → LLM 生成回答 → 保存回答 → 返回用户

图谱问答：用户提问 → 解析意图和实体 → Neo4j 图查询 → 组织查询结果 → 生成回答 → 保存回答 → 返回用户

四、接口设计

本系统采用 RESTful API 设计风格，所有接口返回统一使用以下 JSON 格式：

```
{
  "code": 200,          // 状态码，200 表示成功，其他值表示错误
  "message": "操作成功", // 操作结果描述
  "data": {}            // 返回的数据，可能是对象或数组
}
```

4.1 用户认证接口

4.1.1 用户注册

- **接口路径：**`/api/account/register`
- **请求方式：**POST
- **请求参数：**

```
{
  "user_id": 1,                    // 用户 id
  "phone_number": "13812345678",   // 手机号码，用作登录账号
  "password": "password123",       // 密码，长度 8-20 位
  "id_number": "110101199001011234", // 身份证号码
  "name": "张三",                  // 用户姓名
  "gender": 1,                     // 性别：0(女)，1(男)
  "age": 30,                       // 年龄
  "address": "北京市朝阳区",        // 地址
  "wechat": "zhangsan123",         // 微信号
  "qq": "123456789"                // QQ 号
}
```

- **响应结果：**

```
{
  "code": 200,
  "message": "注册成功",
  "data": {
    "user_id": 1,
    "name": "张三",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
}
```

- **错误码：**

错误码	说明
400	请求参数错误
409	电话号码已被注册
500	服务器内部错误

- **示例请求：**

```
curl -X POST http://api.example.com/api/account/register \
  -H "Content-Type: application/json" \
  -d '{
    "phone_number": "13812345678",
    "password": "password123",
    "id_number": "110101199001011234",
    "name": "张三",
    "gender": 1,
    "age": 30
  }'
```

4.1.2 用户登录

- **接口路径：**`/api/account/login`
- **请求方式：**POST
- **请求参数：**

```
{
  "phone_number": "13812345678",    // 手机号码
  "password": "password123"         // 密码
}
```

- **响应结果：**

```
{
  "code": 200,
  "message": "登录成功",
  "data": {
    "user_id": 1,
    "name": "张三",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "role_type": 0
  }
}
```

- **错误码：**

错误码	说明
400	请求参数错误
401	用户名或密码错误

403	账户已被禁用
500	服务器内部错误

- **示例请求：**

```
curl -X POST http://api.example.com/api/account/login \
  -H "Content-Type: application/json" \
  -d '{
    "phone_number": "13812345678",
    "password": "password123"
  }'
```

4.1.3 忘记密码

- **接口路径：**`/api/account/forgot-password`
- **请求方式：**POST
- **请求参数：**

```
{
  "phone_number": "13812345678",   // 手机号码
  "id_number": "110101199001011234" // 身份证号码
}
```

- **响应结果：**

```
{
  "code": 200,
  "message": "验证成功，可以重置密码",
  "data": {
    "reset_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
```

```
}
```

- **错误码：**

错误码	说明
400	请求参数错误
404	用户不存在
403	身份证号码与手机号不匹配
500	服务器内部错误

4.1.4 重置密码

- **接口路径：**`/api/account/reset-password`
- **请求方式：**POST
- **请求参数：**

```
{
  "reset_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...", // 重置令牌
  "new_password": "newpassword123"                          // 新密码
}
```

- **响应结果：**

```
{
  "code": 200,
  "message": "密码重置成功",
  "data": null
```

```
}
```

4.2 聊天会话接口

4.2.1 创建新聊天会话

- **接口路径：**`/api/chat/sessions`
- **请求方式：**POST
- **请求头：**
  - Authorization: Bearer {token}
- **请求参数：**

```
{
  "title": "关于青铜器的问答",      // 会话标题，默认为"新对话"
  "type": 2                        // 会话类型：1(普通)、2(知识库)、3(知识图谱)
}
```

- **响应结果：**

```
{
  "code": 200,
  "message": "创建成功",
  "data": {
    "id": 1,
    "title": "关于青铜器的问答",
    "type": 2,
    "created_at": "2023-05-20T10:30:00",
    "updated_at": "2023-05-20T10:30:00"
  }
```

```
}
```

- **示例请求：**

```
curl -X POST http://api.example.com/api/chat/sessions \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..." \
  -H "Content-Type: application/json" \
  -d '{
    "title": "关于青铜器的问答",
    "type": 2
  }'
```

4.2.2 获取用户聊天会话列表

- **接口路径：**`/api/chat/sessions`
- **请求方式：**GET
- **请求头：**
  - Authorization: Bearer {token}
- **响应结果：**

```
{
  "code": 200,
  "message": "获取成功",
  "data": [
    {
      "id": 1,
      "title": "关于青铜器的问答",
      "type": 2,
      "created_at": "2023-05-20T10:30:00",
      "updated_at": "2023-05-20T10:30:00"
    },
```

```
    {
      "id": 2,
      "title": "中国古代绘画",
      "type": 3,
      "created_at": "2023-05-21T14:15:00",
      "updated_at": "2023-05-21T15:45:00"
    }
  ]
}
```

4.2.3 获取单个聊天会话及其消息

- **接口路径：** `/api/chat/sessions/{session_id}`
- **请求方式：** GET
- **请求头：**
  - Authorization: Bearer {token}
- **路径参数：**
  - session_id: 会话 ID
- **响应结果：**

```
{
  "code": 200,
  "message": "获取成功",
  "data": {
    "session": {
      "id": 1,
      "title": "关于青铜器的问答",
      "type": 2,
      "created_at": "2023-05-20T10:30:00",
      "updated_at": "2023-05-20T10:30:00"
```

```
    },
    "messages": [
      {
        "id": 1,
        "content": "商代青铜器有哪些特点？",
        "is_user": true,
        "created_at": "2023-05-20T10:31:00"
      },
      {
        "id": 2,
        "content": "商代青铜器主要有以下特点：\n1. 器型多样，包括礼器、乐器、兵器和生活用具等...",
        "is_user": false,
        "created_at": "2023-05-20T10:31:05"
      }
    ]
  }
}
```

4.2.4 删除聊天会话

- **接口路径：** `/api/chat/sessions/{session_id}`
- **请求方式：** DELETE
- **请求头：**
  - Authorization: Bearer {token}
- **路径参数：**
  - session_id: 会话 ID
- **返回数据：**

```
{
  "code": 200,
  "message": "删除成功",
  "data": null
}
```

4.2.5 更新聊天会话标题

- **接口路径：** `/api/chat/sessions/{session_id}`
- **请求方式：** PUT
- **请求头：**
  - Authorization: Bearer {token}
- **路径参数：**
  - session_id: 会话 ID
- **响应结果：**

```
{
  "title": "商代青铜器特点研究"  // 新会话标题
}
```

- **返回数据：**

```
{
  "code": 200,
  "message": "更新成功",
  "data": {
    "id": 1,
    "title": "商代青铜器特点研究",
    "created_at": "2023-05-20T10:30:00",
```

```
    "updated_at": "2023-05-20T11:15:00"
  }
}
```

4.3 问答接口

4.3.1 发送消息并获取回答

- **接口路径：** `/api/chat/sessions/{session_id}/messages`
- **请求方式：** POST
- **请求头：**
  - Authorization: Bearer {token}
- **路径参数：**
  - session_id: 会话 ID
- **请求参数：**

```
{
  "content": "商代青铜器与西周青铜器有什么区别？"  // 消息内容
}
```

- **响应结果：**

```
{
  "code": 200,
  "message": "发送成功",
  "data": {
    "user_message": {
      "id": 3,
      "content": "商代青铜器与西周青铜器有什么区别？",
```

```
      "is_user": true,
      "created_at": "2023-05-20T11:20:00"
    },
    "ai_message": {
      "id": 4,
      "content": "商代青铜器与西周青铜器的主要区别体现在以下几个方面：\n1. 纹饰：商代青铜器多以兽面纹（饕餮纹）为主...",
      "is_user": false,
      "created_at": "2023-05-20T11:20:10"
    }
  }
}
```

- **示例请求：**

```
curl -X POST http://api.example.com/api/chat/sessions/1/messages \
  -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..." \
  -H "Content-Type: application/json" \
  -d '{
    "content": "商代青铜器与西周青铜器有什么区别？"
  }'
```

4.3.2 发送消息并获取流式回复

- **接口路径：**`/api/chat/sessions/{session_id}/stream`
- **请求方式：**POST
- **请求头：**
  - Authorization: Bearer {token}
- **路径参数：**
  - session_id: 会话 ID

- **请求参数：**

```
{
  "content": "商代青铜器与西周青铜器有什么区别？"  // 消息内容
}
```

- **响应结果：**
  - 返回类型：text/event-stream
  - 每个事件包含生成的文本片段
  - 最后一个事件带有 [DONE] 标记表示结束
- **示例事件流：**

```
data: {"text": "商代青铜器与西周青铜器的主要区别"}
data: {"text": "体现在以下几个方面：\n1. 纹饰："}
data: {"text": "商代青铜器多以兽面纹（饕餮纹）为主..."}
...
data: {"text": "[DONE]", "message_id": 4}
```

4.4 知识库管理接口

4.4.1 获取知识库列表

- **接口路径：**`/api/knowledgebase`
- **请求方式：**GET
- **请求头：**
  - Authorization: Bearer {token}
- **响应结果：**

```
{
  "code": 200,
  "message": "获取成功",
  "data": [
    {
      "id": 1,
      "name": "中国古代青铜器",
      "description": "包含商周时期主要青铜器的信息",
      "status": 1,
      "created_at": "2023-04-10T09:00:00",
      "updated_at": "2023-04-10T09:00:00"
    },
    {
      "id": 2,
      "name": "中国古代陶瓷",
      "description": "从新石器时代到明清的陶瓷艺术发展",
      "status": 1,
      "created_at": "2023-04-12T14:30:00",
      "updated_at": "2023-04-12T14:30:00"
    }
  ]
}
```


4.5 统一状态码

状态码	说明
200	成功
400	请求参数错误
401	未授权/认证失败

403	禁止访问/无权限
404	资源不存在
409	资源冲突
422	请求实体无法处理
429	请求过于频繁
500	服务器内部错误

掌上博物馆系统设计文档

一、需求分析

1.1 项目需求背景

随着博物馆数字化转型的深入推进，访客期望获取更加便捷、个性化的知识服务。传统的参观方式（如讲解员讲解、展板阅读）存在时间和空间限制，难以满足现代参观者对深度内容和交互体验的需求。掌上博物馆系统旨在利用移动互联网和人工智能技术，打破这些限制，为参观者和文化爱好者提供高效、便捷的博物馆体验渠道。

1.2 用户需求分析

调研显示，用户希望通过系统便捷获取展品背景、历史文化信息，获得个性化推荐，并支持深入查询和互动体验。同时系统应具备内容准确、界面友好、使用简单等特点，可辅助参观体验、减少信息获取障碍，并通过数据分析优化内容与服务。

1.3 功能需求列表

系统需具备以下核心功能：

1. 文物浏览：显示文物的基本信息、图片，支持按关键字搜索
2. 用户交互：支持用户对文物点赞、评论、收藏，上传相关照片
3. 以图搜图：支持上传图片或拍摄照片，根据图片特征搜索相关文物
4. 用户个人信息管理：支持用户注册、登录，个人信息设置
5. 用户个人动态：支持用户发表动态，上传文字和图片，其他用户可点赞、评论
6. 博物馆浏览：提供博物馆信息展示、排行榜、公告等功能

二、系统架构设计

2.1 总体架构图

2.2 模块划分说明

系统采用三层架构，包括：

1. **前端应用层**：基于 HarmonyOS 的移动应用，提供用户界面和交互功能
2. **应用服务层**：使用 Node.js 和 Flask 构建的后端服务，提供业务逻辑处理
3. **数据存储层**：使用 MySQL 存储结构化数据，Milvus 存储图像特征向量，文件系统存储图片资源

2.3 技术选型

- **前端**：ArkTS 语言、HarmonyOS SDK、Stage 模型应用程序包结构
- **后端**：

- 主要服务：Node.js + Express.js
- 图像处理服务：Python + Flask + CLIP 模型
- **数据库：**
- 关系型数据库：MySQL
- 向量数据库：Milvus
- **开发工具：**DevEco Studio
- **代码管理：**GitHub

三、数据库设计

3.1 MySQL 数据库设计

3.1.1 用户表（user）

字段名	数据类型	约束	说明
user_id	INT	PK, AUTO_INCREMENT	用户 ID
phone_number	VARCHAR(20)	NOT NULL	电话号码
password	VARCHAR(32)	NOT NULL	密码（加密存储）
id_number	VARCHAR(18)	NOT NULL	身份证号
name	VARCHAR(20)	NOT NULL	用户名
description	VARCHAR(255)	NULL	用户简介

gender	SMALLINT	NULL	用户性别，0：女，1：男
age	SMALLINT	NULL	年龄
address	VARCHAR(255)	NULL	地址
wechat	VARCHAR(50)	NULL	微信号
qq	VARCHAR(50)	NULL	QQ 号
account_status	SMALLINT	NOT NULL, DEFAULT 1	用户状态, 0:禁用,1:启用
role_type	SMALLINT	NOT NULL, DEFAULT 0	用户角色, 0:用户,1:管理员
create_time	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	创建时间
update_time	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE	更新时间

3.1.2 文物表（cultural_relic）

字段名	数据类型	约束	说明
relic_id	INT	NOT NULL	文物 ID
museum_id	INT	NOT NULL	博物馆 ID

name	VARCHAR(50)	NOT NULL	文物名称
type	VARCHAR(50)	NULL	文物类型(标签)
description	TEXT	NULL	文物介绍
size	TEXT	NULL	文物尺寸
matrials	VARCHAR(50)	NULL	文物材料
dynasty	VARCHAR(32)	NULL	文物年代
likes_count	INT	NOT NULL, DEFAULT 0	点赞数
views_count	INT	NOT NULL, DEFAULT 0	阅读量
author	VARCHAR(50)	NOT NULL	作者
entry_time	NULL	NULL	入馆时间（年份，负数为公元前）
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间

3.1.3 博物馆表（museum）

字段名	数据类型	约束	说明
museum_id	INT	NOT NULL	博物馆 ID
museum_name	VARCHAR(100)	NOT NULL	博物馆名字
description	TEXT	NULL	博物馆介绍
address	VARCHAR(100)	NULL	博物馆线下地址
website_url	VARCHAR(255)	NULL	博物馆网站链接
author	VARCHAR(50)	NOT NULL	作者
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间

3.1.4 博物馆图片表（museum_image）

字段名	数据类型	约束	说明
image_id	INT	NOT NULL	图片 ID
img_url	TEXT	NOT NULL	博物馆图片地址

museum_id	INT	NOT NULL	博物馆 ID

3.1.5 文物图片表（relic_image）

字段名	数据类型	约束	说明
image_id	INT	NOT NULL	图片 ID
img_url	TEXT	NOT NULL	博物馆图片地址
relic_id	INT	NOT NULL	文物 ID

3.1.6 文物视频表（relic_video）

字段名	数据类型	约束	说明
video_id	INT	NOT NULL	视频 ID（自增主键）
relic_id	INT	NOT NULL	文物 ID
video_url	VARCHAR(255)	NULL	视频地址（不为 null 则是第三方 url，为 null 表示本地存储）
is_official	TINYINT	NOT NULL DEFAULT 0	官方标记（0：用户上传；1：官方发布）
title	VARCHAR(50)	NOT NULL	视频标题

duration	INT	NULL	duration
views_count	INT	NOT NULL, DEFAULT 0	播放次数
status	TINYINT	NOT NULL DEFAULT 2	状态（0：下架；1：正常；2：审核中）
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间，也作为上传时间
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间

3.1.7 文物评论表（relic_comment）

字段名	数据类型	约束	说明
relic_id	INT	NOT NULL	文物 ID
comment_id	INT	NOT NULL	评论 id
user_id	INT	PK, AUTO_INCREMENT	用户 ID
parent_id	INT	DEFAULT NULL	父评论 ID（为 NULL 表示顶级评论）
likes_count	INT	NOT NULL, DEFAULT 0	点赞数
reply_count	INT	NOT NULL, DEFAULT 0	回复数

is_deleted	TINYINT	NOT NULL, DEFAULT 0	删除标记（0：未删除，1：已删除）
status	TINYINT	NOT NULL DEFAULT 2	状态（0：下架；1：正常；2：审核中）
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录修改时间，如果审核状态修改可作为状态修改时间

3.1.8 用户图片表（user_image）

字段名	数据类型	约束	说明
image_id	INT	NOT NULL	图片 ID
user_id	INT	NOT NULL	用户 ID
image_suffix	VARCHAR(10)	NULL	图片后缀：.jpg | .png | ...
comment_id	INT	NOT NULL	评论 id
status	TINYINT	NOT NULL DEFAULT 2	状态（0：下架；1：正常；2：审核中）
create_time	NULL	DEFAULT	记录创建时间

		CURRENT_TIMESTAMP	
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间

3.1.9 文物点赞表（relic_like ）

字段名	数据类型	约束	说明
relic_id	INT	NOT NULL	文物 ID
user_id	INT	NOT NULL	用户 ID

3.1.10 评论点赞表（comment_like）

字段名	数据类型	约束	说明
comment_id	INT	NOT NULL	评论 ID
user_id	INT	NOT NULL	用户 ID

3.1.11 用户收藏表（user_favorite）

字段名	数据类型	约束	说明
relic_id	INT	NOT NULL	文物 ID
museum_id	INT	NOT NULL	博物馆 ID
user_id	INT	NOT NULL	用户 ID
id	INT	NOT NULL	自增主键

favorite_type	TINYINT	NOT NULL	收藏类型：0：博物馆，1：文物
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间

3.1.12 用户浏览历史表（user_browsing_history）

字段名	数据类型	约束	说明
relic_id	INT	NOT NULL	文物 ID
image_id	INT	NOT NULL	图片 ID
user_id	INT	NOT NULL	用户 ID
id	INT	NOT NULL	自增主键
search_contentfavorite_type	TEXT	NOT NULL	搜索关键词或内容
search_time	NULL	DEFAULT CURRENT_TIMESTAMP	搜索时间

3.1.13 博物馆公告表（museum_notice）

字段名	数据类型	约束	说明
notice_id	INT	NOT NULL	公告 ID

museum_id	INT	NOT NULL	博物馆 ID
notice_title	VARCHAR(100)	NULL	公告标题
notice_author	VARCHAR(50)	NULL	公告作者
notice_content	TEXT	NULL	公告内容
notice_time	NULL	DEFAULT CURRENT_TIMESTAMP	公告时间
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间

3.1.14 用户反馈表（complaint_feedback）

字段名	数据类型	约束	说明
feedback_id	INT	NOT NULL	反馈 ID
user_id	INT	NOT NULL	用户 ID
feedback_content	TEXT	NULL	反馈内容

contact_info	VARCHAR(50)	(NULL)	联系方式（邮箱/电话）
feedback_type	TINYINT	NULL	反馈类型，1：投诉，2：建议，3：错误报告，4：其他
process_status	TINYINT	NOT NULL DEFAULT 0	处理状态，0：未处理，1：处理中，2：已处理，3：已关闭
process_remark	TEXT	NULL	处理备注
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间|反馈时间
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间|处理时间

3.1.15 日志表（admin_log）

字段名	数据类型	约束	说明
log_id	INT	NOT NULL	日志 ID

admin_name	VARCHAR(20)	NOT NULL	管理员名称
operation	VARCHAR(50)	NOT NULL	操作内容
operation_time	DATETIME	DEFAULT CURRENT_TIMESTAM	操作时间

3.1.16 会话消息表（chat_messages）

字段名	数据类型	约束	说明
id	INT	NOT NULL	消息 ID，自增主键
session_id	INT	NOT NULL	所属会话 ID，关联 chat_sessions 表
content	TEXT	NOT NULL	消息内容
is_user	TINYINT	NOT NULL DEFAULT 1	是否用户发送的消息：TRUE(用户消息)、FALSE(系统回复)
created_at	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间

3.1.17 聊天会话表（chat_sessions）

字段名	数据类型	约束	说明
id	INT	NOT NULL	消息 ID，自增

			主键
user_id	INT	NOT NULL	用户 ID
title	VARCHAR(100)	DEFAULT '新对话'	会话标题
type	TINYINT	NOT NULL DEFAULT 1	会话类型：1(普通问答)、2(知识库问答)、3(知识图谱问答)
is_active	TINYINT	NOT NULL DEFAULT 1	会话是否活跃
created_at	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间
updated_at	NULL	DEFAULT CURRENT_TIMESTAMP	会话最后更新时间

3.1.18 数据库备份表（database_backup_recover）

字段名	数据类型	约束	说明
id	INT	NOT NULL	主键
admin_name	VARCHAR(20)	NOT NULL	管理员名称
comment	VARCHAR(50)	NULL	备份注释

path	VARCHAR(255)	NOT NULL	备份文件路径
create_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录创建时间
update_time	NULL	DEFAULT CURRENT_TIMESTAMP	记录更新时间

3.1.19 用户搜索历史表（user_search_history）

字段名	数据类型	约束	说明
relic_id	INT	NOT NULL	文物 ID
image_id	INT	NOT NULL	图片 ID
user_id	INT	NOT NULL	用户 ID
id	INT	NOT NULL	自增主键
search_content	TEXT	NOT NULL	搜索关键词或内容
search_time	NULL	DEFAULT CURRENT_TIMESTAMP	搜索时间

3.2 Milvus 向量数据库设计

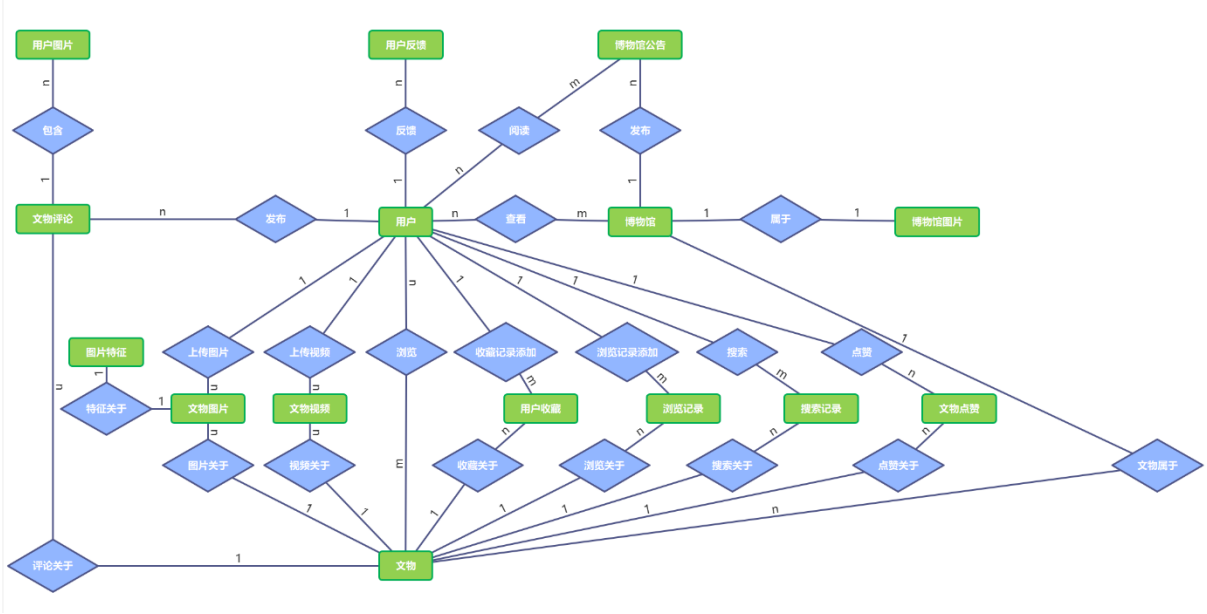
3.2.1 文物图像特征集合（antique_features）

字段名	数据类型	说明
id	INT	主键 ID（对应文物 ID）
image_vector	FLOAT_VECTOR	图像特征向量（512 维）

索引设计：

- 向量索引类型：IVF_FLAT
- 距离度量：余弦相似度（Cosine Similarity）
- 搜索参数：nlist=128

3.3 ER 图



3.4 数据流说明

- **文物浏览流程**
- 用户请求文物列表 → 应用服务层查询文物数据 → 返回文物基本信息和图片
- 用户点击文物详情 → 查询详细信息和相关图片/视频 → 更新浏览记录 → 返回完整详情
- **用户交互流程**
- 用户点赞文物 → 更新文物点赞数据 → 返回最新点赞状态
- 用户评论文物 → 保存评论内容 → 更新评论列表 → 返回成功状态
- 用户收藏文物 → 更新收藏记录 → 返回最新收藏状态
- **以图搜图流程**
- 用户上传图片 → 预处理图片 → 提取特征向量 → 向量数据库检索 → 返回相似文物列表
- **用户个人动态流程**
- 用户发布动态 → 保存文字内容 → 上传图片 → 关联评论和图片 → 返回发布结果
- 用户查看动态 → 获取动态列表和相关评论 → 返回完整动态内容
- **博物馆浏览流程**
- 用户查看博物馆列表 → 返回博物馆基本信息和排行
- 用户查看博物馆详情 → 获取博物馆详细信息和馆藏文物 → 返回完整信息

四、接口设计

4.1 用户管理模块

4.1.1 用户注册

- **接口路径：** `/api/user/register`
- **请求方式：** POST
- **请求参数**:

```
{
    "phone_number": "13812345678",    // 手机号码，用作登录账号
    "password": "password123",        // 密码，长度 8-20 位
    "id_number": "110101199001011234", // 身份证号码
    "name": "张三"                     // 用户姓名
}
```

- **响应结果：**

```
{
    "code": 0,
    "message": "注册成功",
    "data": {
        "user_id": 1
    }
}
```

4.1.2 用户登录

- **接口路径：** `/api/user/login`
- **请求方式：** POST

- **请求参数**:

```
{
    "phone_number": "13812345678",    // 手机号码
    "password": "password123"         // 密码
}
```

- **响应结果**：

```
{
    "code": 0,
    "message": "登录成功",
    "data": {
        "user_id": 1
    }
}
```

4.1.3 获取用户信息

- **接口路径**： `/api/user/login`
- **请求方式**： GET
- **路径参数**：userId - 用户 ID
- **响应结果**：

```
{
    "code": 0,
    "message": "获取成功",
    "data": {
        "user_id": 1,
        "phone_number": "13812345678",
        "name": "张三",
        "description": "个人简介",
```

```
        "gender": 1,                // 性别，1-男，0-女
        "age": 30,
        "address": "北京市朝阳区",
        "wechat": "wechat123",
        "qq": "123456",
        "account_status": 1,        // 账号状态，1-正常，0-禁用
        "comment_status": 1,        // 评论状态，1-允许评论，0-禁止评论
        "role_type": 0,             // 角色类型，0-普通用户，1-管理员
        "create_time": "2023-01-01T12:00:00Z",
        "update_time": "2023-01-01T12:00:00Z"
    }
}
```

4.1.4 更新用户信息

- **接口路径**: /api/user/info/:userId
- **请求方式**: PUT
- **路径参数**: userId - 用户 ID
- **请求参数**:

```
{
    "name": "张三",
    "phone_number": "13812345678",
    "gender": 1,
    "age": 30,
    "description": "新的个人简介",
    "address": "北京市海淀区",
    "wechat": "new_wechat",
    "qq": "654321"
}
```

- **响应结果：**

```
{
    "code": 0,
    "message": "更新成功"
}
```

4.2 文物管理模块

4.2.1 获取文物列表

- **接口路径**: /api/antique/list
- **请求方式**: GET
- **响应结果**:

```
[
    {
        "id": 1,
        "name": "青花瓷花瓶",
        "image": "http://example.com/images/vase.jpg",
        "category": "清朝",
        "like_count": 120,
        "views_count": 1500
    },
    {
        "id": 2,
        "name": "唐三彩骆驼",
        "image": "http://example.com/images/camel.jpg",
        "category": "唐朝",
        "like_count": 95,
        "views_count": 1200
    }
```

```
]
```

4.2.2 获取文物详情

- **接口路径**: /api/antique/detail/:id
- **请求方式**: GET
- **路径参数**: id - 文物 ID
- **响应结果**:

```
{
    "relic_id": 1,
    "name": "青花瓷花瓶",
    "type": "瓷器",
    "size": "高 30cm，直径 15cm",
    "matrials": "瓷土",
    "dynasty": "清朝",
    "author": "匿名",
    "entry_time": "2020-01-15",
    "description": "这是一件精美的青花瓷花瓶，制作于清朝乾隆年间...",
    "images": [
        "http://example.com/images/vase_1.jpg",
        "http://example.com/images/vase_2.jpg"
    ],
    "videos": [
        "http://example.com/videos/vase_intro.mp4"
    ]
}
```

4.2.3 文物点赞

- **接口路径**: /api/antique/like/:id

- **请求方式**: POST
- **路径参数**: id - 文物 ID
- **请求参数**:

```
{
    "user_id": 1
}
```

- **响应结果**:

```
{
    "like_count": 121
}
```

4.2.4 获取文物点赞状态

- **接口路径**: /api/antique/status/:id
- **请求方式**: GET
- **路径参数**: id - 文物 ID
- **查询参数**: user_id - 用户 ID
- **响应结果**:

```
{
    "like_count": 121,
    "is_liked": true
}
```

4.2.5 文物收藏

- **接口路径**: /api/antique/favorite/:id
- **请求方式**: POST
- **路径参数**: id - 文物 ID
- **请求参数**:

```
{
    "user_id": 1
}
```

- **响应结果**:

```
{
    "favorite_count": 56
}
```

4.2.6 获取文物收藏状态

- **接口路径**: /api/antique/fstatus/:id
- **请求方式**: GET
- **路径参数**: id - 文物 ID
- **查询参数**: user_id - 用户 ID
- **响应结果**:

```
{
    "favorite_count": 56,
    "is_favorited": true
```

```
}
```

4.2.7 更新文物浏览量

- **接口路径**: /api/antique/views/:id
- **请求方式**: POST
- **路径参数**: id - 文物 ID
- **请求参数**:

```
{
    "user_id": 1
}
```

- **响应结果**:

```
{
    "views_count": 1501
}
```

4.2.8 获取文物浏览量

- **接口路径**: /api/antique/views/:id
- **请求方式**: GET
- **路径参数**: id - 文物 ID
- **响应结果**:

```
{
    "views_count": 1501
```

```
}
```

4.2.9 获取用户浏览历史

- **接口路径**: /api/antique/history/:user_id
- **请求方式**: GET
- **路径参数**: user_id - 用户 ID
- **响应结果**:

```
[
    {
        "id": 1,
        "name": "青花瓷花瓶",
        "imageUrl": "http://example.com/images/vase.jpg",
        "browseTime": "2023-05-20T14:30:00Z"
    },
    {
        "id": 3,
        "name": "明代青铜器",
        "imageUrl": "http://example.com/images/bronze.jpg",
        "browseTime": "2023-05-19T10:15:00Z"
    }
]
```

4.2.10 相似文物搜索

- **接口路径**: /search_similar
- **请求方式**: POST
- **请求参数**: 图片文件（使用 multipart/form-data 格式）

- **响应结果**:


```
{
    "code": 200,
    "message": "搜索成功",
    "data": {
        "similar_ids": [1, 5, 12, 18],
        "count": 4
    }
}
```


4.3 评论管理模块

4.3.1 获取文物评论列表

- **接口路径**: /api/antique/comments/:id
- **请求方式**: GET
- **路径参数**: id - 文物 ID
- **响应结果**:


```
[
    {
        "comment_id": 1,
        "content": "这件文物真是精美绝伦！",
        "like_count": 15,
        "reply_count": 2,
        "create_time": "2023-04-10T09:30:00Z",
        "parent_id": null,
        "status": 1,
        "is_deleted": 0,
        "user_name": "文物爱好者"
    },
```

```
    {
        "comment_id": 2,
        "content": "这是我见过的最好的藏品之一",
        "like_count": 8,
        "reply_count": 0,
        "create_time": "2023-04-11T14:20:00Z",
        "parent_id": null,
        "status": 1,
        "is_deleted": 0,
        "user_name": "历史研究者"
    }
]
```

4.3.2 提交评论

- **接口路径**: /api/antique/upload_comments/:id
- **请求方式**: POST
- **路径参数**: id - 文物 ID
- **请求参数**:

```
{
    "user_id": 1,
    "content": "这件文物非常精美，尤其是那些精细的雕刻！",
    "parent_id": null   // 如果是回复评论，则填写父评论 ID
}
```

- **响应结果**:

```
{
    "comment_id": 3
```

```
}
```

4.3.3 点赞评论

- **接口路径**: /api/antique/comments/like/:comment_id
- **请求方式**: POST
- **路径参数**: comment_id - 评论 ID
- **请求参数**:

```
json


{
    "user_id": 1
}
```

- **响应结果**:

```
json


{
    "like_count": 16
}
```

4.3.4 检查评论状态

- **接口路径**: /api/antique/comment_status/:user_id
- **请求方式**: GET
- **路径参数**: user_id - 用户 ID

- **响应结果**:

```
json


{
    "comment_status": 1,  // 1-允许评论，0-禁止评论
    "name": "张三"
}
```

4.3.5 上传评论图片

- **接口路径**: /upload_image
- **请求方式**: POST
- **请求头**:
    - user_id: 用户 ID
    - comment_id: 评论 ID
- **请求参数**: 图片文件（使用 multipart/form-data 格式）
- **响应结果**:

```
json


{
    "code": 200,
    "message": "图片上传成功",
    "data": {
        "image_id": 12,
        "image_name": "12.jpg"
    }
}
```

4.4. 动态管理模块

4.4.1 获取动态列表

- **接口路径**: /api/dynamic/list
- **请求方式**: GET
- **响应结果**:

```
json


[
    {
        "dynamic_id": 1,
        "content": "今天参观了故宫博物院，收获颇丰！",
        "create_time": "2023-05-15T10:00:00Z",
        "comment_status": 1,
        "like_count": 25,
        "reply_count": 3,
        "username": "文化爱好者",
        "images": [
            {
                "image_id": 1,
                "suffix": "jpg",
                "status": 1
            },
            {
                "image_id": 2,
                "suffix": "jpg",
                "status": 1
            }
        ]
    },
    {
        "dynamic_id": 2,
```

```
        "content": "刚刚看到了一件稀有的明代文物，太震撼了！",
        "create_time": "2023-05-16T14:30:00Z",
        "comment_status": 1,
        "like_count": 18,
        "reply_count": 2,
        "username": "历史研究者",
        "images": [
            {
                "image_id": 3,
                "suffix": "jpg",
                "status": 1
            }
        ]
    }
]
```

4.4.2 发布动态

- **接口路径**: /api/dynamic/publish
- **请求方式**: POST
- **请求参数**:

```
json


{
    "user_id": 1,
    "content": "今天参观了国家博物馆，看到了很多精美的文物！",
    "image_ids": [1, 2, 3]  // 已上传的图片 ID 数组
}
```

- **响应结果**:

```
json


{
    "comment_id": 5,
    "message": "动态发布成功，等待审核"
}
```

4.4.3 获取动态评论列表

- **接口路径**: /api/dynamic/comments/:dynamicId
- **请求方式**: GET
- **路径参数**: dynamicId - 动态 ID
- **响应结果**:

```
json


[
    {
        "comment_id": 1,
        "content": "太棒了，我也想去参观！",
        "create_time": "2023-05-15T11:30:00Z",
        "like_count": 5,
        "reply_count": 1,
        "parent_id": null,
        "username": "文物爱好者",
        "replies": [
            {
                "comment_id": 3,
                "content": "周末人会很多，建议提前预约",
                "create_time": "2023-05-15T12:15:00Z",
                "like_count": 2,
                "reply_count": 0,
```

```
                "parent_id": 1,
                "username": "博物馆志愿者"
            }
        ]
    },
    {
        "comment_id": 2,
        "content": "分享一下具体位置吧",
        "create_time": "2023-05-15T13:45:00Z",
        "like_count": 3,
        "reply_count": 0,
        "parent_id": null,
        "username": "旅行达人",
        "replies": []
    }
]
```

4.4.4 发表动态评论

- **接口路径**: /api/dynamic/comment
- **请求方式**: POST
- **请求参数**:

```
json


{
    "dynamic_id": 1,
    "user_id": 2,
    "content": "非常感谢分享，这些照片拍得真好！",
    "parent_id": null  // 如果是回复评论，则填写父评论 ID
}
```

- **响应结果**:

```
json


{
    "comment_id": 4,
    "message": "评论发布成功"
}
```

4.4.5 动态点赞/取消点赞

- **接口路径**: /api/dynamic/like
- **请求方式**: POST
- **请求参数**:

```
json


{
    "dynamic_id": 1,
    "action": "like"  // "like"-点赞，"unlike"-取消点赞
}
```

- **响应结果**:

```
json


{
    "newLikeCount": 26,
    "isLiked": true
}
```

4.4.6 删除评论

- **接口路径**: /api/dynamic/comment/:commentId
- **请求方式**: DELETE
- **路径参数**: commentId - 评论 ID
- **请求参数**:

```
json


{
    "user_id": 1
}
```

- **响应结果**:

```
json


{
    "message": "评论删除成功"
}
```

4.4.7 获取子评论

- **接口路径**: /api/dynamic/comments/children/:parentId
- **请求方式**: GET
- **路径参数**: parentId - 父评论 ID
- **响应结果**:

```
json


[
    {
        "comment_id": 3,
        "content": "周末人会很多，建议提前预约",
        "create_time": "2023-05-15T12:15:00Z",
        "like_count": 2,
        "reply_count": 0,
        "parent_id": 1,
        "username": "博物馆志愿者",
        "replies": []
    },
    {
        "comment_id": 5,
        "content": "我上周去的，确实需要预约",
        "create_time": "2023-05-16T09:20:00Z",
        "like_count": 1,
        "reply_count": 0,
        "parent_id": 1,
        "username": "文化爱好者",
        "replies": []
    }
]
```

4.4.8 上传动态图片

- **接口路径**: /api/dynamic/upload/image
- **请求方式**: POST
- **请求参数**:
    - user_id: 用户 ID

- 图片文件（使用 multipart/form-data 格式）

- **响应结果**:

```
json


{
    "image_id": 4,
    "filename": "4.jpg",
    "status": 0  // 初始状态为审核中
}
```

4.5 博物馆模块

4.5.1 获取博物馆公告

- **接口路径**: /api/museum/list

- **请求方式**: GET

- **响应结果**:

```
json


[
    {
        "notice_id": 1,
        "museum_id": 1,
        "notice_title": "五一假期开放时间调整公告",
        "notice_time": "2023-04-25T00:00:00Z",
        "notice_author": "博物馆管理员",
        "notice_content": "尊敬的观众朋友们，五一假期期间，本馆开放时间调整为9:00-20:00，请合理安排参观时间。",
        "create_time": "2023-04-25T10:30:00Z",
        "update_time": "2023-04-25T10:30:00Z"
```

```
    },
    {
        "notice_id": 2,
        "museum_id": 1,
        "notice_title": "新展览开幕通知",
        "notice_time": "2023-04-28T00:00:00Z",
        "notice_author": "展览部主任",
        "notice_content": "我馆将于 5 月 10 日开幕"丝路文明"特展，欢迎广大观众前来参观。",
        "create_time": "2023-04-28T14:15:00Z",
        "update_time": "2023-04-28T14:15:00Z"
    }
]
```

4.5.2 获取博物馆信息

- **接口路径**: /api/museum/info
- **请求方式**: GET
- **响应结果**:

```
json

[
    {
        "museum_id": 1,
        "museum_name": "故宫博物院",
        "museum_image": "http://example.com/images/forbidden_city.jpg"
    },
    {
        "museum_id": 2,
        "museum_name": "国家博物馆",
        "museum_image": "http://example.com/images/national_museum.jpg"
    },
    {
```

```
        "museum_id": 3,
        "museum_name": "上海博物馆",
        "museum_image": "http://example.com/images/shanghai_museum.jpg"
    }
]
```

4.5.3 获取博物馆详情

- **接口路径**: /api/museum/detail/:id
- **请求方式**: GET
- **路径参数**: id - 博物馆 ID
- **响应结果**:

```
json


{
    "museum_id": 1,
    "museum_name": "故宫博物院",
    "description": "故宫博物院是中国最大的古代文化艺术博物馆，建立于 1925 年 10 月 10 日，位于北京故宫紫禁城内。是在明朝、清朝两代皇宫及其收藏的基础上建立起来的中国综合性博物馆，也是中国最大的古代文化艺术博物馆，其文物收藏主要来源于清代宫中旧藏，是第一批全国爱国主义教育示范基地。",
    "address": "北京市东城区景山前街 4 号",
    "website_url": "https://www.dpm.org.cn/",
    "booking_url": "https://www.dpm.org.cn/visit/guide.html"
}
```

4.5.4 获取博物馆藏品

- **接口路径**: /api/museum/relics/:id

- **请求方式**: GET
- **路径参数**: id - 博物馆 ID
- **响应结果**:

```
json


[
    {
        "relic_id": 1,
        "relic_name": "清乾隆粉彩瓷瓶",
        "relic_image": "http://example.com/images/qing_vase.jpg"
    },
    {
        "relic_id": 2,
        "relic_name": "宋代青瓷",
        "relic_image": "http://example.com/images/song_ceramics.jpg"
    },
    {
        "relic_id": 3,
        "relic_name": "明代龙纹玉带",
        "relic_image": "http://example.com/images/ming_jade.jpg"
    }
]
```

4.5.5 获取博物馆排行榜

- **接口路径**: /api/museum/ranking
- **请求方式**: GET
- **响应结果**:

```
json
```

```
[
    {
        "museum_id": 1,
        "museum_name": "故宫博物院",
        "address": "北京市东城区景山前街 4 号",
        "museum_image": "http://example.com/images/forbidden_city.jpg",
        "relic_count": 1862431
    },
    {
        "museum_id": 2,
        "museum_name": "国家博物馆",
        "address": "北京市东城区天安门广场东侧",
        "museum_image": "http://example.com/images/national_museum.jpg",
        "relic_count": 1050000
    },
    {
        "museum_id": 3,
        "museum_name": "上海博物馆",
        "address": "上海市黄浦区人民大道 201 号",
        "museum_image": "http://example.com/images/shanghai_museum.jpg",
        "relic_count": 120000
    }
]
```


五、系统实现细节

5.1 前端实现

系统前端基于 HarmonyOS 平台，使用 ArkTS 语言实现，采用 Stage 模型应用程序包结构。主要页面包括：

1. **首页**：显示博物馆信息和公告

2. **文物浏览页**：展示文物列表，支持分类和搜索
3. **文物详情页**：展示文物详细信息，支持点赞、评论、收藏
4. **博物馆页面**：展示博物馆信息和排行
5. **用户动态页**：展示和发布用户动态
6. **个人中心页**：管理用户信息、收藏、浏览历史等

前端实现了以下关键功能：

- 使用 LazyForEach 优化大量数据加载
- 实现流畅的图片预览和图库浏览
- 支持评论嵌套显示和交互
- 实现动态点赞和状态同步

5.2 后端实现

系统后端主要基于 Node.js，部分功能使用 Python 实现：

1. **Node.js 服务**：
    - 使用 Express 框架处理 HTTP 请求
    - 实现用户认证、文物管理、博物馆信息等核心功能
    - 使用 MySQL 存储结构化数据
2. **Python 服务**：
    - 使用 Flask 框架提供图像处理 API
    - 基于 CLIP 模型实现图像特征提取
    - 使用 Milvus 向量数据库进行相似图像检索

后端关键实现包括：

- 使用事务保证数据一致性
- 实现图像上传和处理流程
- 支持评论树状结构和嵌套回复
- 实现文物点赞和收藏状态管理

5.3 以图搜图实现

以图搜图功能基于 CLIP 多模态模型实现，流程包括：

1. **离线处理阶段**：
    - 文物图像预处理和特征提取
    - 使用 CLIP 的图像编码器将每个文物图像编码成特征向量
    - 将特征向量存储在 Milvus 向量数据库中
2. **在线查询阶段**：
    - 处理用户上传的查询图像
    - 使用相同的 CLIP 图像编码器提取特征向量
    - 在 Milvus 中检索相似向量
    - 按相似度排序返回文物信息

六、系统部署

6.1 部署架构

系统采用前后端分离架构部署：

- 移动应用：通过应用商店分发
- Node.js 后端：部署在云服务器上
- Python 图像服务：独立部署在专用服务器上
- 数据库：MySQL 和 Milvus 部署在独立服务器上

6.2 运行环境要求

1. **前端：**
    - HarmonyOS 3.0 及以上版本
    - 设备：支持 HarmonyOS 的手机或平板设备
2. **后端：**
    - Node.js v14.0 及以上
    - Python 3.8 及以上
    - MySQL 8.0 及以上
    - Milvus 2.0 及以上
3. **服务器：**
    - CPU: 4 核及以上
    - 内存: 8GB 及以上
    - 存储: 100GB 及以上
    - 操作系统: Ubuntu 20.04 或 CentOS 8

七、总结与展望

7.1 系统特点

掌上博物馆系统具有以下特点：

1. 提供全面的文物浏览和博物馆信息展示功能
2. 支持丰富的用户交互，包括点赞、评论、收藏
3. 创新性地引入以图搜图功能，增强用户体验
4. 支持用户个人动态分享，促进用户社交互动
5. 具备完善的用户个人信息管理功能

7.2 未来展望

系统未来可以在以下方面进行扩展：

1. 引入 AR/VR 技术，提供沉浸式文物展示体验
2. 集成语音导览功能，支持多语言讲解
3. 增加文物知识图谱，提供更深入的关联知识探索
4. 开发个性化推荐功能，基于用户兴趣推荐文物和展览
5. 拓展社交功能，支持用户组织线下活动和文化交流

掌上博物馆系统通过现代移动互联网技术，为用户提供了便捷、丰富的博物馆体验，有效促进了文化传播和知识分享，为文化遗产的数字化保护和传承提供了有力支持。

后台管理子系统系统设计文档

一、需求分析

1.1 项目需求背景

在博物馆数字化转型的背景下，多个子系统协同工作共同构建了完整的海外文物馆藏数字化服务平台。随着文物展示、知识问答、图片上传、信息检索等功能的逐步完善，系统产生了大量的多源异构数据，包括用户行为数据、文物信息、展览记录、评论互动等。这些数据的分散存储和管理给系统维护、安全管理和决策支持带来了巨大挑战。

后台管理子系统作为整个平台的"中枢神经系统"，承担着数据集成、监控管理和安全保障的核心职责。它需要将现有各子系统的数据进行统一汇总、管理和调度，为平台管理员提供全面、可靠的运维工具。通过集中化管理，后台系统不仅能够提高运营效率、保障系统安全，还能为后续的数据分析和决策支持奠定坚实基础。

1.2 用户需求分析

后台管理系统面临以下核心需求：

1. **数据集中化管理需求**：团队需要一个统一的平台查看和管理来自各子系统的数据，避免在多个系统间频繁切换，提高工作效率。
2. **安全管理需求**：平台需要保障用户信息安全、文物数据安全和系统运行安全，需要完善的权限管理和数据备份恢复机制。
3. **内容治理需求**：随着用户互动的增加，需要有效管理用户发布的评论和图片，防止不良信息传播。
4. **资源管理需求**：对系统内的博物馆、文物等资源进行统一管理，支持新增、修改、查询和删除操作。
5. **信息发布需求**：管理员需要发布平台公告，向用户传达重要信息。
6. **系统维护需求**：提供日志查询功能和数据库管理工具，便于故障排查和数据维护。

1.3 功能需求列表

基于上述分析，后台管理子系统需要实现以下核心功能模块：

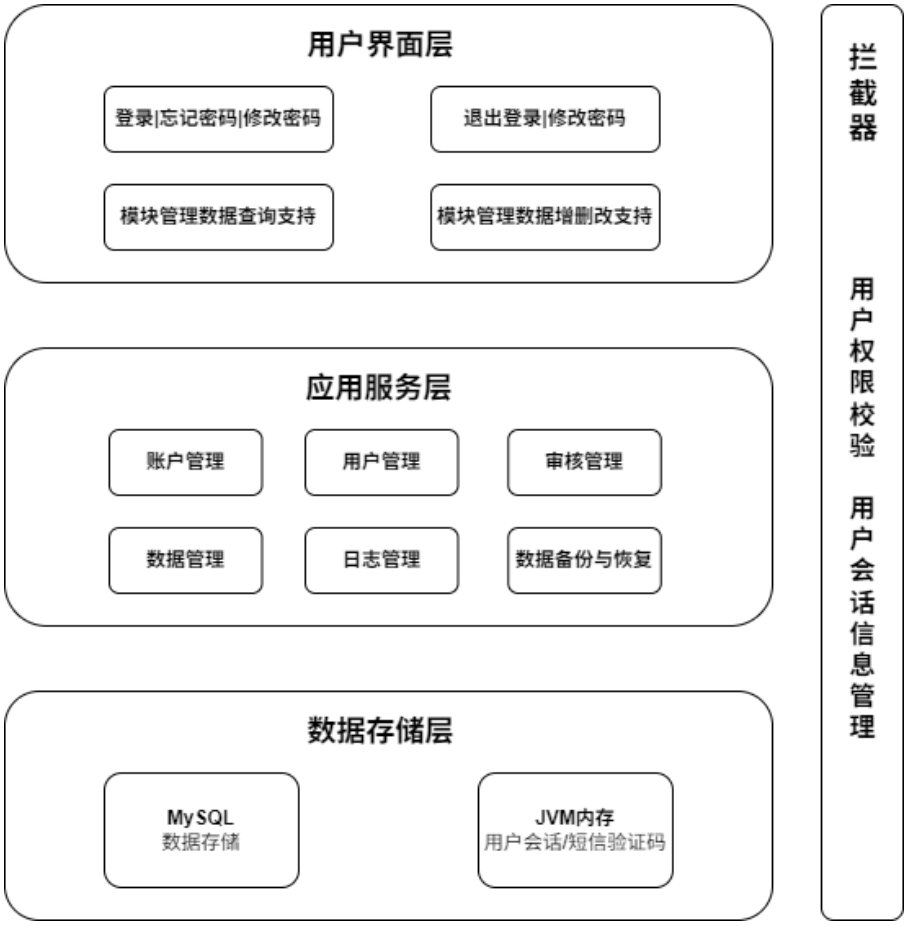
1. **用户管理**
    - 用户账户创建、修改与删除
    - 用户账号状态与评论状态管理（激活/禁用）
    - 安全风险用户识别与处理
2. **审核管理**
    - 用户评论内容审核
    - 用户上传图片内容审核
    - 审核记录存储
    - 用户评论状态管理
3. **博物馆管理**
    - 博物馆基本信息维护（名称、介绍、地址等）
    - 博物馆信息查询
4. **文物管理**
    - 文物基本信息维护（名称、尺寸、年代等）
    - 文物信息查询
5. **公告管理**
    - 博物馆公告创建与发布
    - 公告信息查询
6. **日志管理**

- 系统操作日志记录与存储

7. **数据库备份与恢复管理**
   - 数据库手动备份路径配置
   - 数据库历史备份记录查询
   - 数据恢复版本管理与回滚

二、系统架构设计

2.1 总体架构图



2.2 模块划分说明

系统采用三层架构，包括用户界面层（提供登录页面、各模块数据管理页面等）、应用服务层（负责实现各模块数据增删查改逻辑），以及数据存储层（使用 MySQL 实现数据持久化存储，使用 JVM 内存保存用户会话信息和短信验证码等即时数据）。

2.3 技术选型

系统前端使用 Vue 3 搭配 Element-Plus、 Vue Router 和 Axios，后端使用 SpringBoot3 。数据层使用 MySQL 和 JVM 内存，部署方面通过 Docker 实现容器化。

三、数据库设计

3.1 用户表

```
create table user
(
    user_id        bigint auto_increment comment '用户id'
        primary key,
    phone_number   varchar(20)                         not null comment '电话号码',
    password       varchar(32)                         not null comment '密码',
    id_number      char(18)                            not null comment '身份证号',
    name           varchar(20)                         not null comment '用户名',
    description    varchar(255)                        null comment '用户简介',
    gender         tinyint(1)                          null comment '用户性别，0：女，1：男',
    age            tinyint unsigned                    null comment '年龄',
    address        varchar(255)                        null comment '地址',
    wechat         varchar(50)                         null comment '微信号',
    qq             varchar(50)                         null comment 'qq号',
    account_status tinyint  default 1                  not null comment '用户状态，0:禁用,1:启用',
    comment_status tinyint  default 1                  not null comment '用户评论状态（0-禁止评论，1-允许评论）',
    role_type      tinyint  default 0                  not null comment '用户角色，0:用户,1:管理员',
    create_time    datetime default CURRENT_TIMESTAMP  null comment '创建时间',
    update_time    datetime default CURRENT_TIMESTAMP  null on update CURRENT_TIMESTAMP comment '更新时间'
)
    comment '用户表';
```

3.2 用户评论表

```sql
create table relic_comment
(
    comment_id  bigint unsigned auto_increment comment '评论id'
        primary key,
    relic_id    bigint unsigned                        not null comment '文物id',
    user_id     bigint unsigned                        not null comment '用户id',
    content     text                                   not null comment '评论内容',
    parent_id   bigint unsigned                        null comment '父评论ID（为NULL表示顶级评论）',
    like_count  int unsigned     default '0'           null comment '点赞数',
    reply_count int unsigned     default '0'           null comment '回复数',
    is_deleted  tinyint unsigned default '0'           null comment '删除标记（0：未删除，1：已删除）',
    status      tinyint          default 0             null comment '评论状态（0-审核中，1-过审，2-未过审）',
    create_time datetime         default CURRENT_TIMESTAMP null comment '记录创建时间，也作为评论时间',
    update_time datetime         default CURRENT_TIMESTAMP null on update CURRENT_TIMESTAMP comment '记录修改时间，如果审核状态修改可作为状态修改时间'
)
    comment '文物评论表' collate = utf8mb4_unicode_ci;
```

3.3 用户图片表

```sql
create table user_image
(
    image_id     bigint auto_increment comment '图片id'
        primary key,
    image_suffix varchar(10)                        null comment '图片后缀：.jpg | .png | ...',
    user_id      bigint                             not null comment '用户id',
    comment_id   bigint                             not null comment '关联的评论id',
    status       tinyint  default 0                 not null comment '状态，0-审核中，1-过审，2-未过审',
    create_time  datetime default CURRENT_TIMESTAMP null comment '创建时间',
    update_time  datetime default CURRENT_TIMESTAMP null on update CURRENT_TIMESTAMP comment '更新时间'
)
    comment '用户上传图片表' charset = utf8mb4;
```

3.4 博物馆数据表

```sql
create table museum
(
    museum_id   bigint unsigned auto_increment comment '博物馆Id'
        primary key,
    museum_name varchar(100)                        not null comment '博物馆名字',
    description text                                null comment '博物馆介绍',
    address     varchar(100)                        null comment '博物馆线下地址',
    website_url varchar(255)                        null comment '博物馆网站链接',
    booking_url varchar(255)                        null comment '博物馆预约链接',
    create_time datetime default (now())            null comment '创建时间',
    update_time datetime default CURRENT_TIMESTAMP null on update CURRENT_TIMESTAMP comment '修改时间'
)
    comment '博物馆表' charset = utf8mb4;
```

3.5 文物表

```sql
create table cultural_relic
(
    relic_id    bigint unsigned auto_increment comment '文物id'
        primary key,
    museum_id   bigint unsigned                          not null comment '博物馆id',
    name        varchar(50)                              not null comment '文物名称',
    type        varchar(50)                              null comment '文物类型(标签)',
    description text                                     null comment '文物介绍',
    size        text                                     null comment '文物尺寸',
    matrials    varchar(50)                              null comment '文物材料',
    dynasty     varchar(32)                              null comment '文物年代',
    likes_count bigint unsigned default '0'              null comment '点赞数',
    views_count bigint unsigned default '0'              null comment '阅读量',
    author      varchar(50)                              null,
    entry_time  int                                      null comment '入馆时间（年份，负数为公元前）',
    create_time datetime        default CURRENT_TIMESTAMP null comment '记录创建时间',
    update_time datetime        default CURRENT_TIMESTAMP null on update CURRENT_TIMESTAMP comment '记录更新时间'
)
    comment '文物表' charset = utf8mb4;
```

3.6 公告表

```sql
create table museum_notice
(
    notice_id      bigint unsigned auto_increment comment '公告id'
        primary key,
    museum_id      bigint unsigned                   not null comment '博物馆id',
    notice_title   varchar(100)                      not null comment '公告标题',
    notice_author  varchar(50)                       null comment '公告作者',
    notice_content text                              null comment '公告内容',
    notice_time    datetime                          null comment '公告时间',
    create_time    datetime default (now())          null comment '创建时间',
    update_time    datetime default CURRENT_TIMESTAMP null on update CURRENT_TIMESTAMP comment '修改时间'
)
    comment '博物馆公告表' charset = utf8mb4;
```

3.7 管理员日志表

```sql
create table admin_log
(
    log_id         bigint unsigned auto_increment comment '日志id'
        primary key,
    admin_name     varchar(20)                       not null comment '管理员名称',
    operation      varchar(50)                       not null comment '操作内容',
    operation_time datetime default CURRENT_TIMESTAMP not null comment '操作时间'
)
    comment '管理员操作日志表' charset = utf8mb4;
```

3.8 数据库备份与恢复记录表

```sql
create table database_backup_recover
(
    id          int unsigned auto_increment comment '主键'
        primary key,
    admin_name  varchar(20)  not null comment '管理员名称',
    comment     varchar(50)  null comment '备份注释',
    path        varchar(255) not null comment '备份文件路径',
    create_time datetime     not null comment '创建时间',
    update_time datetime     null comment '修改时间'
)
    comment '数据库备份与恢复表';
```

四、接口设计

4.1 接口基本信息说明

本系统采用 RESTful API 设计风格，所有接口返回统一使用以下 JSON 格式：

```
{
  "code": 200,          // 状态码，200 表示成功，其他值表示错误
  "info": "请求成功", // 操作结果描述
  "data": {}            // 返回的数据，可能是对象或数组
}
```

错误码：

错误码	说明
200	请求成功
400	请求错误

401	登录超时
404	请求地址不存在
500	服务器错误
600	请求参数错误

4.2 账户管理接口

4.1.1 用户登录

- 接口路径：/api/account/login- 请求方式：POST- 请求参数：

```
{
  "phoneNumber": "13812345678",    // 手机号码，用作登录账号
  "password": "password123"        // 密码
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "userId": 1,
    "name": "张三",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
}
```

4.1.2 获取短信验证码

- 接口路径：/api/account/getCheckCode● 请求方式：POST● 请求参数：

```
{
  "phoneNumber": "13812345678"   // 手机号码
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
}
```

4.1.3 验证码校验

- 接口路径：/api/account/submitCheckCode● 请求方式：POST● 请求参数：

```
{
  "phoneNumber": "13812345678",    // 手机号码
  "checkCode": "123456"           // 短信验证码
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.1.4 忘记密码并修改密码

- 接口路径：/api/account/forgetAndChangePassword- 请求方式：POST- 请求参数：

```
{
  "phoneNumber": "13812345678",   // 手机号码
  "newPassword": "newpassword123" // 新密码
}
```

- 响应结果：

```
{
  "code": 200,
 "info": "请求成功"
}
```

- 接口路径：/api/account/logout- 请求方式：POST- 请求参数：

```
{
  "userId": 1   // 用户 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.1.6 修改密码

- 接口路径：/api/account/changePassword- 请求方式：POST- 请求参数：

```
{
  "user_id": 1,                    // 用户 ID
  "oldPassword": "oldpassword123", // 旧密码
  "newPassword": "newpassword123"  // 新密码
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.3 用户管理接口

4.2.1 加载用户列表

- 接口路径：/api/user/loadUserList- 请求方式：POST- 请求参数：

```
{
  "pageNum": 1,                    // 当前页码
  "pageSize": 10,                  // 每页条数
  "userId": "1",                       // 用户 id
  "phoneNumber": "135",            // 电话号码（模糊查询）
  "idNumber": 4411,                // 身份证号（模糊查询）
  "name": "测试"                   // 用户名称（模糊查询）
  "accountStatus": 1                   // 账户状态
  "roleType": 1                            // 角色类型
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                 // 总记录数
    "list": [
      {
        "userId": 1,
        "username": "张三",
        "phone": "13812345678",
        "email": "zhangsan@example.com",
        "status": 1,
        "role": "admin",
        "createTime": "2023-01-01 10:00:00"
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.2.2 添加或修改用户

- 接口路径：/api/user/addOrUpdateUser- 请求方式：POST- 请求参数：

```
{
  "userId": 1,                    // 用户 ID(新增时可不传)
  "username": "张三",             // 用户名
  "password": "password123",      // 密码(新增时必填，修改时可不填)
  "phone": "13812345678",         // 手机号码
  "email": "zhangsan@example.com",// 邮箱
  "status": 1,                    // 状态
  "role": 1                               // 角色类型
```

```
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.2.3 删除用户

- 接口路径：/api/user/deleteUser● 请求方式：POST● 请求参数：

```
{
  "userId": 1                    // 用户 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.2.4 修改用户评论状态

- 接口路径：/api/user/updateUserCommentStatus● 请求方式：POST● 请求参数：

```
{
  "userId": 1,                   // 用户 ID
  "commentStatus": 0             // 评论状态(0-禁用，1-启用)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.4 评论审核管理接口

4.3.1 加载评论列表

- 接口路径：/api/comment/loadCommentList- 请求方式：POST- 请求参数：

```
{
  "pageNum": 1,                    // 当前页码
  "pageSize": 10,                  // 每页条数
  "commentId": 1001,               // 评论 ID(可选)
  "relicId": 2001,                 // 文物 ID(可选)
  "userId": 3001,                  // 用户 ID(可选)
  "parentId": null,                // 父评论 ID(可选)
  "status": 1,                     // 审核状态(0-审核中，1-已通过，2-未通过)
  "commentTimeStart": "2023-01-01 00:00:00", // 评论时间起始(可选)
  "commentTimeEnd": "2023-12-31 23:59:59"    // 评论时间结束(可选)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                  // 总记录数
    "list": [
      {
```

```
        "commentId": 1,
        "relicId": 1001,
        "userId": 1,
        "userName": "张三",
        "content": "这件文物很有历史价值",
        "parentId": null,          // 父评论 ID，顶级评论为 null
        "status": 1,               // 0-审核中，1-已通过，2-未通过
        "commentTime": "2023-05-15 14:30:00",
        "replyContent": "感谢您的评论", // 管理员回复内容(如有)
        "replyTime": "2023-05-15 15:00:00" // 管理员回复时间(如有)
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.3.2 修改评论状态

- 接口路径：/api/comment/updateCommentStatus● 请求方式：POST● 请求参数：

```
{
  "commentId": 1,                  // 评论 ID(必填)
  "parentId": null,                // 父评论 ID(可选)
  "userId": 1,                     // 用户 ID(可选)
  "status": 1                      // 要修改的评论状态(0-审核中，1-已通过，2-未通过)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
```

```
}
```

4.3.3 删除评论

- 接口路径：/api/comment/deleteComment● 请求方式：POST● 请求参数：

```
{
  "commentId": 1                    // 评论 ID(必填)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.5 图片管审核理接口

4.4.1 分页查询图片列表

- 接口路径：/api/image/loadImageList● 请求方式：POST● 请求参数：

```
{
  "pageNum": 1,                     // 当前页码
  "pageSize": 10,                   // 每页条数
  "imageId": 1001,                  // 图片 ID(可选)
  "userId": 2001,                   // 用户 ID(可选)
  "commentId": 3001,                // 关联评论 ID(可选)
  "status": 1,                      // 图片状态(0-待审核，1-已通过，2-已拒绝)
  "uploadTimeStart": "2023-01-01 00:00:00", // 上传时间起始(可选)
  "uploadTimeEnd": "2023-12-31 23:59:59"    // 上传时间结束(可选)
```

```
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                  // 总记录数
    "list": [
      {
        "imageId": 1,
        "imageSuffix": "jpg",      // 图片后缀
        "userId": 1,
        "userName": "张三",        // 用户名(可根据实际需求添加)
        "commentId": 101,
        "status": 1,               // 0-待审核，1-已通过，2-已拒绝
        "uploadTime": "2023-05-15 14:30:00",
        "updateTime": "2023-05-15 15:00:00", // 状态修改时间
        "imageUrl": "/images/ancient_pottery.jpg" // 图片访问 URL(可根据实际需求添加)
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.4.2 修改图片审核状态

- 接口路径：/api/image/updateImageStatus● 请求方式：POST● 请求参数：

```
{
  "imageId": 1,                    // 图片 ID(必填)
```

```
  "commentId": 101,                // 关联评论 ID(必填)
  "status": 1                      // 要修改的状态(0-待审核，1-已通过，2-已拒绝)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.4.3 加载图片

- 接口路径：/api/image/loadImage● 请求方式：GET● 请求参数：

```
imageName=ancient_pottery.jpg
```

(注：此接口通过 URL 参数传递 imageName，而非 JSON 请求体)

- 响应结果：

- 直接返回图片二进制流

4.6 博物馆管理接口

4.5.1 加载博物馆列表

- 接口路径：/api/museum/loadMuseumList● 请求方式：POST● 请求参数：

```
{
  "pageNum": 1,                    // 当前页码
  "pageSize": 10,                  // 每页条数
  "museumId": 1001,                // 博物馆 ID(可选)
  "museumName": "故宫博物院"        // 博物馆名称(模糊查询，可选)
```

```
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                  // 总记录数
    "list": [
      {
        "museumId": 1,
        "museumName": "故宫博物院",
        "description": "中国最大的古代文化艺术博物馆",
        "address": "北京市东城区景山前街 4 号",
        "websiteUrl": "https://www.dpm.org.cn/",
        "bookingUrl": "https://ticket.dpm.org.cn/"
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.5.2 添加或修改博物馆

- 接口路径：/api/museum/addOrUpdateMuseum● 请求方式：POST● 请求参数：

```
{
  "museumId": 1,                   // 博物馆 ID(新增时可不填)
  "museumName": "故宫博物院",      // 博物馆名称
  "description": "中国最大的古代文化艺术博物馆", // 博物馆介绍
  "address": "北京市东城区景山前街 4 号",          // 博物馆地址
```

```
  "websiteUrl": "https://www.dpm.org.cn/",       // 博物馆网址
  "bookingUrl": "https://ticket.dpm.org.cn/"     // 博物馆购票网址
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.5.3 删除博物馆

- 接口路径：/api/museum/deleteMuseum● 请求方式：POST● 请求参数：

```
{
  "museumId": 1                    // 博物馆 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.7 文物管理接口

4.6.1 加载文物列表

- 接口路径：/api/relic/loadRelicList● 请求方式：POST● 请求参数：

```
{
  "pageNum": 1,                    // 当前页码
  "pageSize": 10,                  // 每页条数
  "relicId": 1001,                 // 文物 ID(可选)
  "museumId": 2001,                // 博物馆 ID(可选)
  "name": "青铜器",                 // 文物名称(模糊查询，可选)
  "type": "瓷器",                   // 文物种类(可选)
  "matrials": "陶瓷",               // 文物材质(可选)
  "dynasty": "唐朝",                // 文物年代(可选)
  "author": "张三",                 // 文物作者(可选)
  "entryTime": 2020                // 文物入库年份(可选)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                  // 总记录数
    "list": [
      {
        "relicId": 1,
        "museumId": 101,
        "name": "唐三彩马",
        "type": "陶俑",
        "description": "唐代三彩釉陶器代表作",
        "size": "高 30cm，长 40cm",
        "matrials": "陶土",
        "dynasty": "唐朝",
        "likesCount": 120,
        "viewsCount": 560,
        "author": "佚名",
        "entryTime": 2018
      }
```

```
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.6.2 添加或修改文物

- 接口路径：/api/relic/addOrUpdateRelic- 请求方式：POST- 请求参数：

```
{
  "relicId": 1,                    // 文物 ID(新增时可不填)
  "museumId": 101,                 // 博物馆 ID
  "name": "唐三彩马",               // 文物名称
  "type": "陶俑",                  // 文物种类
  "description": "唐代三彩釉陶器代表作", // 文物描述
  "size": "高 30cm，长 40cm",        // 文物尺寸
  "matrials": "陶土",              // 文物材质
  "dynasty": "唐朝",               // 文物年代
  "likesCount": 120,               // 点赞数(新增时可填 0)
  "viewsCount": 560,               // 浏览次数(新增时可填 0)
  "author": "佚名",                // 文物作者
  "entryTime": 2018                // 文物入库年份
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.6.3 删除文物

- 接口路径：/api/relic/deleteRelic- 请求方式：POST- 请求参数：

```
{
  "relicId": 1                    // 文物 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.8 公告管理接口

4.7.1 加载公告列表

- 接口路径：/api/notice/loadNoticeList- 请求方式：POST- 请求参数：

```
{
  "pageNum": 1,                   // 当前页码
  "pageSize": 10,                 // 每页条数
  "noticeId": 1001,               // 公告 ID(可选)
  "museumId": 2001,               // 博物馆 ID(可选)
  "noticeTitle": "紧急通知",       // 公告标题(模糊查询，可选)
  "noticeAuthor": "张三",          // 发布人(可选)
  "noticeTimeStart": "2023-01-01 00:00:00", // 发布时间起始(可选)
  "noticeTimeEnd": "2023-12-31 23:59:59"    // 发布时间结束(可选)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                 // 总记录数
    "list": [
      {
        "noticeId": 1,
        "museumId": 101,
        "museumName": "故宫博物院",
        "noticeTitle": "紧急闭馆通知",
        "noticeAuthor": "管理员 A",
        "noticeContent": "因特殊原因，故宫博物院将于明日闭馆一天，请游客提前安排行程。",
        "noticeTime": "2023-05-15 14:30:00"
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.7.2 添加或修改公告

● 接口路径：/api/notice/addOrUpdateNotice● 请求方式：POST● 请求参数：

```
{
  "noticeId": 1,                  // 公告 ID(新增时可不填)
  "museumId": 101,                // 博物馆 ID
  "noticeTitle": "紧急闭馆通知",    // 公告标题
  "noticeAuthor": "管理员 A",       // 发布人
  "noticeContent": "因特殊原因，故宫博物院将于明日闭馆一天，请游客提前安排行程。" // 发布内容
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.7.3 删除公告

- 接口路径：/api/notice/deleteNotice- 请求方式：POST- 请求参数：

```
{
  "noticeId": 1                    // 公告 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.9 日志管理接口

4.8.1 加载日志列表

- 接口路径：/api/log/loadLogList- 请求方式：POST- 请求参数：

```
{
  "pageNum": 1,                    // 当前页码
  "pageSize": 10,                  // 每页条数
  "adminName": "张三",             // 管理员姓名(模糊查询，可选)
  "operationTimeStart": "2023-01-01 00:00:00", // 操作时间起始(可选)
```

```
  "operationTimeEnd": "2023-12-31 23:59:59"    // 操作时间结束(可选)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 100,                  // 总记录数
    "list": [
      {
        "logId": 1,
        "adminName": "张三",
        "operation": "用户登录",
        "operationTime": "2023-05-15 14:30:00"
      },
      {
        "logId": 2,
        "adminName": "李四",
        "operation": "添加文物",
        "operationTime": "2023-05-15 15:00:00"
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 10
  }
}
```

4.8.2 删除日志

- 接口路径：/api/log/deleteLog● 请求方式：POST● 请求参数：

```
{
  "logId": 1                    // 日志 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.8.3 增加日志

- 接口路径：/api/log/addLog● 请求方式：POST● 请求参数：

```
{
  "adminName": "张三",             // 管理员姓名
  "phoneNumber": "13812345678",    // 管理员电话号码
  "password": "password123",       // 管理员密码(通常不需要传递)
  "operation": "新增文物记录"      // 日志内容
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.10 数据库备份与恢复管理接口

4.9.1 加载备份列表

- 接口路径：/api/database/loadDataBaseList- 请求方式：POST- 请求参数：

```
{
  "pageNum": 1,                    // 当前页码
  "pageSize": 10,                  // 每页条数
  "id": 1,                         // 备份记录 ID(可选)
  "adminName": "admin",            // 管理员用户名(模糊查询，可选)
  "comment": "2023-05",            // 备份注释(模糊查询，可选)
  "backupTimeStart": "2023-01-01 00:00:00", // 备份时间起始(可选)
  "backupTimeEnd": "2023-12-31 23:59:59"    // 备份时间结束(可选)
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": {
    "total": 5,
    "list": [
      {
        "id": 1,
        "adminName": "admin",
        "comment": "2023 年 5 月全量备份",
        "path": "/backup/db_20230501.sql",
        "createTime": "2023-05-01 02:30:00"
      },
      {
        "id": 2,
        "adminName": "admin",
        "comment": "2023 年 6 月增量备份",
```

```
        "path": "/backup/db_20230601.sql",
        "createTime": "2023-06-01 03:15:00"
      }
    ],
    "pageNum": 1,
    "pageSize": 10,
    "pages": 1
  }
}
```


4.9.2 添加或修改数据库备份

- 接口路径：/api/database/addOrUpdateBackup● 请求方式：POST● 请求参数：


```
{
  "id": 1,                          // 备份记录 ID(新增时可不填)
  "adminName": "admin",             // 管理员用户名
  "comment": "2023 年 7 月全量备份", // 备份注释
  "path": "/backup/db_20230701.sql" // 备份路径
}
```


- 响应结果：


```
{
  "code": 200,
  "info": "请求成功"
}
```


4.9.3 删除备份记录

- 接口路径：/api/database/deleteDataBaseBackup● 请求方式：POST● 请求参数：

```
{
  "id": 1                          // 备份记录 ID
}
```

- 响应结果：

```
{
  "code": 200,
  "info": "请求成功"
}
```

4.9.4 获取默认备份路径

- 接口路径：/api/database/getDefaultPath● 请求方式：POST● 请求参数：无● 响应结果：

```
{
  "code": 200,
  "info": "请求成功",
  "data": "/backup/default"
}
```

4.9.5 恢复数据库

- 接口路径：/api/database/recover● 请求方式：POST● 请求参数：

```
{
  "id": 1                          // 要恢复的备份记录 ID
}
```

- 响应结果：

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

```
  "info": "请求成功"
}
```