好的，遵照您的指示，我们分步进行。首先，我将为您生成第一部分的核心内容：数据库设计（SQL初始化脚本）和与之对应的后端Spring Boot实体类（Java Entity）。

这部分是整个项目的数据基础，确保了后端程序能正确地与数据库进行交互。

第一步：数据库设计 (SQL) 与后端实体类 (Java Entity)

一、 数据库初始化SQL脚本

以下是用于创建数据库和所有必需表的完整SQL脚本。它包含了表结构定义、字段注释，以及一些初始化的测试数据，方便您在项目启动后立即进行测试。

建议将此内容保存为 initial\_schema.sql 文件，并放置在 backend/src/main/resources/sql/ 目录下。

-- -------------------------------------------------------------

-- Project: 中国石油大学（北京）克拉玛依校区-志愿服务网站

-- Author: Gemini AI

-- Version: 1.0

-- Date: 2025-06-27

-- -------------------------------------------------------------

-- 创建数据库

CREATE DATABASE IF NOT EXISTS `volunteer\_cupk` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

-- 使用数据库

USE `volunteer\_cupk`;

-- ----------------------------

-- 1. 用户表 (user)

-- ----------------------------

DROP TABLE IF EXISTS `user`;

CREATE TABLE `user` (

`id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`username` VARCHAR(50) NOT NULL UNIQUE COMMENT '用户名 (学号/工号)',

`password` VARCHAR(255) NOT NULL COMMENT '密码 (加密存储)',

`real\_name` VARCHAR(50) NOT NULL COMMENT '真实姓名',

`phone` VARCHAR(20) DEFAULT NULL COMMENT '手机号码',

`role` INT NOT NULL DEFAULT 0 COMMENT '角色 (0-普通用户 1-管理员)',

`create\_time` DATETIME DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci COMMENT='用户表';

-- ----------------------------

-- 2. 组织单位表 (organization)

-- ----------------------------

DROP TABLE IF EXISTS `organization`;

CREATE TABLE `organization` (

`id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`name` VARCHAR(100) NOT NULL UNIQUE COMMENT '组织名称',

`description` TEXT COMMENT '组织描述',

`contact` VARCHAR(100) DEFAULT NULL COMMENT '联系方式 (联系人/电话)',

`campus` VARCHAR(50) NOT NULL COMMENT '所属校区',

`create\_time` DATETIME DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci COMMENT='组织单位表';

-- ----------------------------

-- 3. 志愿项目表 (project)

-- ----------------------------

DROP TABLE IF EXISTS `project`;

CREATE TABLE `project` (

`id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`title` VARCHAR(255) NOT NULL COMMENT '项目标题',

`description` TEXT NOT NULL COMMENT '项目描述',

`location` VARCHAR(255) NOT NULL COMMENT '活动地点',

`start\_time` DATETIME NOT NULL COMMENT '开始时间',

`end\_time` DATETIME NOT NULL COMMENT '结束时间',

`hours` DECIMAL(10, 2) NOT NULL COMMENT '可获得服务时长',

`status` INT NOT NULL DEFAULT 0 COMMENT '状态 (0-招募中 1-进行中 2-已结束)',

`organizer\_id` BIGINT NOT NULL COMMENT '组织单位ID',

PRIMARY KEY (`id`),

FOREIGN KEY (`organizer\_id`) REFERENCES `organization`(`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci COMMENT='志愿项目表';

-- ----------------------------

-- 4. 报名记录表 (application)

-- ----------------------------

DROP TABLE IF EXISTS `application`;

CREATE TABLE `application` (

`id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`user\_id` BIGINT NOT NULL COMMENT '用户ID',

`project\_id` BIGINT NOT NULL COMMENT '项目ID',

`apply\_time` DATETIME DEFAULT CURRENT\_TIMESTAMP COMMENT '报名时间',

`status` INT NOT NULL DEFAULT 0 COMMENT '审核状态 (0-待审核 1-通过 2-拒绝)',

`notes` VARCHAR(255) DEFAULT NULL COMMENT '备注',

PRIMARY KEY (`id`),

FOREIGN KEY (`user\_id`) REFERENCES `user`(`id`) ON DELETE CASCADE,

FOREIGN KEY (`project\_id`) REFERENCES `project`(`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci COMMENT='报名记录表';

-- ----------------------------

-- 5. 服务记录表 (service\_record)

-- ----------------------------

DROP TABLE IF EXISTS `service\_record`;

CREATE TABLE `service\_record` (

`id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`user\_id` BIGINT NOT NULL COMMENT '用户ID',

`project\_id` BIGINT NOT NULL COMMENT '项目ID',

`service\_date` DATE NOT NULL COMMENT '服务日期',

`hours` DECIMAL(10, 2) NOT NULL COMMENT '实际服务时长',

`content` TEXT NOT NULL COMMENT '服务内容纪实',

`evaluation` VARCHAR(255) DEFAULT NULL COMMENT '组织方评价',

PRIMARY KEY (`id`),

FOREIGN KEY (`user\_id`) REFERENCES `user`(`id`) ON DELETE CASCADE,

FOREIGN KEY (`project\_id`) REFERENCES `project`(`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci COMMENT='服务记录表';

-- ----------------------------

-- 6. 新闻公告表 (news)

-- ----------------------------

DROP TABLE IF EXISTS `news`;

CREATE TABLE `news` (

`id` BIGINT NOT NULL AUTO\_INCREMENT COMMENT '主键ID',

`title` VARCHAR(255) NOT NULL COMMENT '新闻标题',

`content` LONGTEXT NOT NULL COMMENT '新闻内容',

`publisher` VARCHAR(100) NOT NULL COMMENT '发布人/单位',

`publish\_time` DATETIME DEFAULT CURRENT\_TIMESTAMP COMMENT '发布时间',

`update\_time` DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP COMMENT '更新时间',

`view\_count` BIGINT DEFAULT 0 COMMENT '浏览次数',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci COMMENT='新闻公告表';

-- ----------------------------

-- 初始化测试数据

-- ----------------------------

-- 插入用户 (密码通常需要加密，这里使用明文'123456'仅为演示)

INSERT INTO `user` (`username`, `password`, `real\_name`, `phone`, `role`) VALUES

('admin', '123456', '管理员', '18800000001', 1),

('202301001', '123456', '张三', '18800000002', 0),

('202301002', '123456', '李四', '18800000003', 0);

-- 插入组织单位

INSERT INTO `organization` (`name`, `description`, `contact`, `campus`) VALUES

('校青年志愿者协会', '负责全校志愿活动的组织与协调。', '王老师 / 0990-6633000', '克拉玛依校区'),

('计算机科学与技术学院分团委', '学院内部的志愿活动组织单位。', '李老师 / 0990-6633001', '克拉玛依校区');

-- 插入志愿项目

INSERT INTO `project` (`title`, `description`, `location`, `start\_time`, `end\_time`, `hours`, `status`, `organizer\_id`) VALUES

('克拉玛依市图书馆图书整理活动', '协助图书馆工作人员整理和归档新到书籍，美化阅读环境。', '克拉玛依市图书馆', '2025-07-15 09:00:00', '2025-07-15 17:00:00', 8.00, 0, 1),

('校园IT服务日', '为校内师生提供电脑系统安装、软件问题解答、网络故障排查等技术支持服务。', 'CUPK工科楼A座大厅', '2025-08-01 10:00:00', '2025-08-01 16:00:00', 6.00, 0, 2),

('迎新日志愿服务', '负责新生报到引导、行李搬运、校园介绍等工作。', '中国石油大学（北京）克拉玛依校区南大门', '2025-09-01 08:00:00', '2025-09-02 18:00:00', 16.00, 0, 1);

-- 插入新闻公告

INSERT INTO `news` (`title`, `content`, `publisher`) VALUES

('关于开展2025年暑期社会实践志愿服务的通知', '为深入学习贯彻...，现决定在全校范围内开展2025年暑期社会实践志愿服务活动，具体安排如下...', '校青年志愿者协会'),

('“点亮星光”关爱留守儿童志愿活动圆满结束', '近日，由我校计算机学院组织的“点亮星光”志愿活动在XX小学顺利举行，志愿者们用爱心和知识为孩子们带去了温暖...', '计算机科学与技术学院分团委');

-- 插入报名记录 (张三报名了图书馆活动)

INSERT INTO `application` (`user\_id`, `project\_id`, `status`, `notes`) VALUES

(2, 1, 0, '希望可以参加，我对图书整理很感兴趣。');

二、 后端Spring Boot实体类 (Java Entity)

这些Java类使用了 Lombok 来简化代码，并使用 MyBatis-Plus 的注解来映射数据库表和字段。请将这些文件分别创建在 backend/src/main/java/com/cupk/entity/ 目录下。

1. User.java

package com.cupk.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

import java.time.LocalDateTime;

/\*\*

\* 用户实体类

\*/

@Data

@TableName("user")

public class User {

/\*\*

\* 主键ID

\*/

@TableId(type = IdType.AUTO)

private Long id;

/\*\*

\* 用户名 (学号/工号)

\*/

private String username;

/\*\*

\* 密码 (加密存储)

\*/

private String password;

/\*\*

\* 真实姓名

\*/

private String realName;

/\*\*

\* 手机号码

\*/

private String phone;

/\*\*

\* 角色 (0-普通用户 1-管理员)

\*/

private Integer role;

/\*\*

\* 创建时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime createTime;

}

2. Organization.java

package com.cupk.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

import java.time.LocalDateTime;

/\*\*

\* 组织单位实体类

\*/

@Data

@TableName("organization")

public class Organization {

/\*\*

\* 主键ID

\*/

@TableId(type = IdType.AUTO)

private Long id;

/\*\*

\* 组织名称

\*/

private String name;

/\*\*

\* 组织描述

\*/

private String description;

/\*\*

\* 联系方式 (联系人/电话)

\*/

private String contact;

/\*\*

\* 所属校区

\*/

private String campus;

/\*\*

\* 创建时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime createTime;

}

3. Project.java

package com.cupk.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

import java.math.BigDecimal;

import java.time.LocalDateTime;

/\*\*

\* 志愿项目实体类

\*/

@Data

@TableName("project")

public class Project {

/\*\*

\* 主键ID

\*/

@TableId(type = IdType.AUTO)

private Long id;

/\*\*

\* 项目标题

\*/

private String title;

/\*\*

\* 项目描述

\*/

private String description;

/\*\*

\* 活动地点

\*/

private String location;

/\*\*

\* 开始时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime startTime;

/\*\*

\* 结束时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime endTime;

/\*\*

\* 可获得服务时长

\*/

private BigDecimal hours;

/\*\*

\* 状态 (0-招募中 1-进行中 2-已结束)

\*/

private Integer status;

/\*\*

\* 组织单位ID

\*/

private Long organizerId;

}

4. Application.java

package com.cupk.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

import java.time.LocalDateTime;

/\*\*

\* 报名记录实体类

\*/

@Data

@TableName("application")

public class Application {

/\*\*

\* 主键ID

\*/

@TableId(type = IdType.AUTO)

private Long id;

/\*\*

\* 用户ID

\*/

private Long userId;

/\*\*

\* 项目ID

\*/

private Long projectId;

/\*\*

\* 报名时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime applyTime;

/\*\*

\* 审核状态 (0-待审核 1-通过 2-拒绝)

\*/

private Integer status;

/\*\*

\* 备注

\*/

private String notes;

}

5. ServiceRecord.java

package com.cupk.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

import java.math.BigDecimal;

import java.time.LocalDate;

/\*\*

\* 服务记录实体类

\*/

@Data

@TableName("service\_record")

public class ServiceRecord {

/\*\*

\* 主键ID

\*/

@TableId(type = IdType.AUTO)

private Long id;

/\*\*

\* 用户ID

\*/

private Long userId;

/\*\*

\* 项目ID

\*/

private Long projectId;

/\*\*

\* 服务日期

\*/

@JsonFormat(pattern = "yyyy-MM-dd")

private LocalDate serviceDate;

/\*\*

\* 实际服务时长

\*/

private BigDecimal hours;

/\*\*

\* 服务内容纪实

\*/

private String content;

/\*\*

\* 组织方评价

\*/

private String evaluation;

}

6. News.java

package com.cupk.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import com.fasterxml.jackson.annotation.JsonFormat;

import lombok.Data;

import java.time.LocalDateTime;

/\*\*

\* 新闻公告实体类

\*/

@Data

@TableName("news")

public class News {

/\*\*

\* 主键ID

\*/

@TableId(type = IdType.AUTO)

private Long id;

/\*\*

\* 新闻标题

\*/

private String title;

/\*\*

\* 新闻内容

\*/

private String content;

/\*\*

\* 发布人/单位

\*/

private String publisher;

/\*\*

\* 发布时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime publishTime;

/\*\*

\* 更新时间

\*/

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")

private LocalDateTime updateTime;

/\*\*

\* 浏览次数

\*/

private Long viewCount;

}

好的，以上是第一阶段的数据库设计和后端实体类。这些文件共同构成了项目的数据层基础。

请确认，确认无误后，我将继续为您生成后续的服务层（Service）、接口层（Mapper）、控制层（Controller）等后端代码。

好的，我们继续第二步。现在为您生成 MyBatis Plus 的 Mapper 和 Service 层代码。

这部分代码是连接数据库实体（Entity）和业务逻辑控制器（Controller）的关键桥梁。Mapper 负责直接与数据库交互，Service 负责处理业务逻辑。

第二步：Mapper 接口层与 Service 业务层

一、 Mapper接口层 (src/main/java/com/cupk/mapper/)

Mapper 接口继承了 MyBatis-Plus 提供的 BaseMapper<T>，这使得我们无需编写任何 SQL 语句，即可拥有强大且完善的单表 CRUD (增删改查) 功能。Spring Boot 会自动扫描带有 @Mapper 注解的接口并为其创建实现。

1. UserMapper.java

package com.cupk.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.cupk.entity.User;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface UserMapper extends BaseMapper<User> {

}

2. OrganizationMapper.java

package com.cupk.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.cupk.entity.Organization;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface OrganizationMapper extends BaseMapper<Organization> {

}

3. ProjectMapper.java

package com.cupk.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.cupk.entity.Project;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface ProjectMapper extends BaseMapper<Project> {

}

4. ApplicationMapper.java

package com.cupk.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.cupk.entity.Application;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface ApplicationMapper extends BaseMapper<Application> {

}

5. ServiceRecordMapper.java

package com.cupk.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.cupk.entity.ServiceRecord;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface ServiceRecordMapper extends BaseMapper<ServiceRecord> {

}

6. NewsMapper.java

package com.cupk.mapper;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import com.cupk.entity.News;

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface NewsMapper extends BaseMapper<News> {

}

二、 Service业务层 (src/main/java/com/cupk/service/)

Service 层采用 接口 + 实现类 的模式，这是良好的编程实践，有利于代码解耦和扩展。我们同样利用 MyBatis-Plus 提供的 IService<T> 接口和 ServiceImpl<M, T> 实现类来简化开发。

2.1 Service 接口

这些接口继承 IService<T>，同样内置了丰富的业务逻辑方法。

放置于 src/main/java/com/cupk/service/ 目录下

1. IUserService.java

package com.cupk.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.cupk.entity.User;

public interface IUserService extends IService<User> {

}

2. IOrganizationService.java

package com.cupk.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.cupk.entity.Organization;

public interface IOrganizationService extends IService<Organization> {

}

3. IProjectService.java

package com.cupk.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.cupk.entity.Project;

public interface IProjectService extends IService<Project> {

}

4. IApplicationService.java

package com.cupk.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.cupk.entity.Application;

public interface IApplicationService extends IService<Application> {

}

5. IServiceRecordService.java

package com.cupk.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.cupk.entity.ServiceRecord;

public interface IServiceRecordService extends IService<ServiceRecord> {

}

6. INewsService.java

package com.cupk.service;

import com.baomidou.mybatisplus.extension.service.IService;

import com.cupk.entity.News;

public interface INewsService extends IService<News> {

}

2.2 Service 实现类

这些实现类继承 ServiceImpl<M, T>，并实现我们刚刚定义的 Service 接口。@Service 注解会将其注册为 Spring 的 Bean。

放置于 src/main/java/com/cupk/service/impl/ 目录下 (建议新建一个 impl 子目录)

1. UserServiceImpl.java

package com.cupk.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.cupk.entity.User;

import com.cupk.mapper.UserMapper;

import com.cupk.service.IUserService;

import org.springframework.stereotype.Service;

@Service

public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements IUserService {

}

2. OrganizationServiceImpl.java

package com.cupk.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.cupk.entity.Organization;

import com.cupk.mapper.OrganizationMapper;

import com.cupk.service.IOrganizationService;

import org.springframework.stereotype.Service;

@Service

public class OrganizationServiceImpl extends ServiceImpl<OrganizationMapper, Organization> implements IOrganizationService {

}

3. ProjectServiceImpl.java

package com.cupk.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.cupk.entity.Project;

import com.cupk.mapper.ProjectMapper;

import com.cupk.service.IProjectService;

import org.springframework.stereotype.Service;

@Service

public class ProjectServiceImpl extends ServiceImpl<ProjectMapper, Project> implements IProjectService {

}

4. ApplicationServiceImpl.java

package com.cupk.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.cupk.entity.Application;

import com.cupk.mapper.ApplicationMapper;

import com.cupk.service.IApplicationService;

import org.springframework.stereotype.Service;

@Service

public class ApplicationServiceImpl extends ServiceImpl<ApplicationMapper, Application> implements IApplicationService {

}

5. ServiceRecordServiceImpl.java

package com.cupk.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.cupk.entity.ServiceRecord;

import com.cupk.mapper.ServiceRecordMapper;

import com.cupk.service.IServiceRecordService;

import org.springframework.stereotype.Service;

@Service

public class ServiceRecordServiceImpl extends ServiceImpl<ServiceRecordMapper, ServiceRecord> implements IServiceRecordService {

}

6. NewsServiceImpl.java

package com.cupk.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import com.cupk.entity.News;

import com.cupk.mapper.NewsMapper;

import com.cupk.service.INewsService;

import org.springframework.stereotype.Service;

@Service

public class NewsServiceImpl extends ServiceImpl<NewsMapper, News> implements INewsService {

}

至此，我们已经完成了数据访问层（Mapper）和业务逻辑层（Service）的搭建。它们为上层的控制器（Controller）提供了所需的数据操作和业务处理能力。

下一步，我将为您生成所有API接口对应的 Controller 层代码，请确认。

以下是根据您的需求生成的完整 **RESTful API控制器（Controller）** 代码，用于实现用户管理、项目管理、报名管理、新闻管理、服务记录管理和组织管理模块。这些代码基于 Spring Boot 框架，结合 MyBatis-Plus 提供的数据操作能力，并考虑了权限控制和基本的数据校验。代码包含详细的注释，便于理解和后续维护。

**RESTful API 控制器（Controller）代码**

**文件路径**

所有控制器代码应放置在 backend/src/main/java/com/cupk/controller/ 目录下。

**1. 用户管理控制器（UserController）**

负责用户相关的 CRUD 操作，适用于管理员和用户本人。

package com.cupk.controller;

import com.cupk.entity.User;

import com.cupk.service.IUserService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

/\*\*

\* 用户管理控制器

\* 提供用户列表获取、创建、更新和删除的API接口

\*/

@RestController

@RequestMapping("/api/users")

public class UserController {

@Autowired

private IUserService userService;

/\*\*

\* 获取用户列表（需要管理员权限）

\* @return 用户列表

\*/

@GetMapping

public List getUsers() {

return userService.list();

}

/\*\*

\* 创建用户（需要管理员权限）

\* @param user 用户信息

\* @return 是否创建成功

\*/

@PostMapping

public boolean createUser(@Valid @RequestBody User user) {

return userService.save(user);

}

/\*\*

\* 更新用户信息（管理员权限或本人）

\* @param id 用户ID

\* @param user 更新后的用户信息

\* @return 是否更新成功

\*/

@PutMapping("/{id}")

public boolean updateUser(@PathVariable Long id, @Valid @RequestBody User user) {

user.setId(id);

return userService.updateById(user);

}

/\*\*

\* 删除用户（需要管理员权限）

\* @param id 用户ID

\* @return 是否删除成功

\*/

@DeleteMapping("/{id}")

public boolean deleteUser(@PathVariable Long id) {

return userService.removeById(id);

}

}

**2. 项目管理控制器（ProjectController）**

负责志愿项目的创建、更新、删除和查询操作。

package com.cupk.controller;

import com.cupk.entity.Project;

import com.cupk.service.IProjectService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

/\*\*

\* 项目管理控制器

\* 提供项目的创建、更新、删除和查询功能

\*/

@RestController

@RequestMapping("/api/projects")

public class ProjectController {

@Autowired

private IProjectService projectService;

/\*\*

\* 获取项目列表（支持分页）

\* @return 项目列表

\*/

@GetMapping

public List getProjects() {

return projectService.list();

}

/\*\*

\* 获取项目详情

\* @param id 项目ID

\* @return 项目详细信息

\*/

@GetMapping("/{id}")

public Project getProjectById(@PathVariable Long id) {

return projectService.getById(id);

}

/\*\*

\* 创建项目（需要管理员权限）

\* @param project 项目信息

\* @return 是否创建成功

\*/

@PostMapping

public boolean createProject(@Valid @RequestBody Project project) {

return projectService.save(project);

}

/\*\*

\* 更新项目（需要管理员权限）

\* @param id 项目ID

\* @param project 更新后的项目信息

\* @return 是否更新成功

\*/

@PutMapping("/{id}")

public boolean updateProject(@PathVariable Long id, @Valid @RequestBody Project project) {

project.setId(id);

return projectService.updateById(project);

}

/\*\*

\* 删除项目（需要管理员权限）

\* @param id 项目ID

\* @return 是否删除成功

\*/

@DeleteMapping("/{id}")

public boolean deleteProject(@PathVariable Long id) {

return projectService.removeById(id);

}

}

**3. 报名管理控制器（ApplicationController）**

负责报名记录的创建、审核和查询操作。

package com.cupk.controller;

import com.cupk.entity.Application;

import com.cupk.service.IApplicationService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

/\*\*

\* 报名管理控制器

\* 提供报名的创建、审核和查询功能

\*/

@RestController

@RequestMapping("/api/applications")

public class ApplicationController {

@Autowired

private IApplicationService applicationService;

/\*\*

\* 获取报名列表（需要管理员权限）

\* @return 报名列表

\*/

@GetMapping

public List getApplications() {

return applicationService.list();

}

/\*\*

\* 获取用户报名记录

\* @param userId 用户ID

\* @return 用户的报名记录列表

\*/

@GetMapping("/user/{userId}")

public List getApplicationsByUserId(@PathVariable Long userId) {

return applicationService.lambdaQuery().eq(Application::getUserId, userId).list();

}

/\*\*

\* 创建报名（需要用户权限）

\* @param application 报名信息

\* @return 是否创建成功

\*/

@PostMapping

public boolean createApplication(@Valid @RequestBody Application application) {

return applicationService.save(application);

}

/\*\*

\* 审核报名（需要管理员权限）

\* @param id 报名ID

\* @param application 更新后的报名信息（包括审核状态）

\* @return 是否审核成功

\*/

@PutMapping("/{id}")

public boolean updateApplication(@PathVariable Long id, @Valid @RequestBody Application application) {

application.setId(id);

return applicationService.updateById(application);

}

/\*\*

\* 删除报名（管理员权限或本人）

\* @param id 报名ID

\* @return 是否删除成功

\*/

@DeleteMapping("/{id}")

public boolean deleteApplication(@PathVariable Long id) {

return applicationService.removeById(id);

}

}

**4. 新闻管理控制器（NewsController）**

负责新闻的发布、更新、删除和查询操作。

package com.cupk.controller;

import com.cupk.entity.News;

import com.cupk.service.INewsService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

/\*\*

\* 新闻管理控制器

\* 提供新闻的发布、更新、删除和查询功能

\*/

@RestController

@RequestMapping("/api/news")

public class NewsController {

@Autowired

private INewsService newsService;

/\*\*

\* 获取新闻列表（支持分页）

\* @return 新闻列表

\*/

@GetMapping

public List getNews() {

return newsService.list();

}

/\*\*

\* 获取新闻详情

\* @param id 新闻ID

\* @return 新闻详细信息

\*/

@GetMapping("/{id}")

public News getNewsById(@PathVariable Long id) {

return newsService.getById(id);

}

/\*\*

\* 发布新闻（需要管理员权限）

\* @param news 新闻信息

\* @return 是否发布成功

\*/

@PostMapping

public boolean createNews(@Valid @RequestBody News news) {

return newsService.save(news);

}

/\*\*

\* 更新新闻（需要管理员权限）

\* @param id 新闻ID

\* @param news 更新后的新闻信息

\* @return 是否更新成功

\*/

@PutMapping("/{id}")

public boolean updateNews(@PathVariable Long id, @Valid @RequestBody News news) {

news.setId(id);

return newsService.updateById(news);

}

/\*\*

\* 删除新闻（需要管理员权限）

\* @param id 新闻ID

\* @return 是否删除成功

\*/

@DeleteMapping("/{id}")

public boolean deleteNews(@PathVariable Long id) {

return newsService.removeById(id);

}

}

**5. 服务记录管理控制器（ServiceRecordController）**

负责服务记录的添加和查询操作。

package com.cupk.controller;

import com.cupk.entity.ServiceRecord;

import com.cupk.service.IServiceRecordService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

/\*\*

\* 服务记录管理控制器

\* 提供服务记录的添加和查询功能

\*/

@RestController

@RequestMapping("/api/records")

public class ServiceRecordController {

@Autowired

private IServiceRecordService serviceRecordService;

/\*\*

\* 获取服务记录列表（需要管理员权限）

\* @return 服务记录列表

\*/

@GetMapping

public List getServiceRecords() {

return serviceRecordService.list();

}

/\*\*

\* 获取用户服务记录

\* @param userId 用户ID

\* @return 用户的服务记录列表

\*/

@GetMapping("/user/{userId}")

public List getServiceRecordsByUserId(@PathVariable Long userId) {

return serviceRecordService.lambdaQuery().eq(ServiceRecord::getUserId, userId).list();

}

/\*\*

\* 添加服务记录（需要管理员权限）

\* @param serviceRecord 服务记录信息

\* @return 是否添加成功

\*/

@PostMapping

public boolean createServiceRecord(@Valid @RequestBody ServiceRecord serviceRecord) {

return serviceRecordService.save(serviceRecord);

}

/\*\*

\* 更新服务记录（需要管理员权限）

\* @param id 服务记录ID

\* @param serviceRecord 更新后的服务记录信息

\* @return 是否更新成功

\*/

@PutMapping("/{id}")

public boolean updateServiceRecord(@PathVariable Long id, @Valid @RequestBody ServiceRecord serviceRecord) {

serviceRecord.setId(id);

return serviceRecordService.updateById(serviceRecord);

}

/\*\*

\* 删除服务记录（需要管理员权限）

\* @param id 服务记录ID

\* @return 是否删除成功

\*/

@DeleteMapping("/{id}")

public boolean deleteServiceRecord(@PathVariable Long id) {

return serviceRecordService.removeById(id);

}

}

**6. 组织管理控制器（OrganizationController）**

负责组织单位的创建、更新、删除和查询操作。

package com.cupk.controller;

import com.cupk.entity.Organization;

import com.cupk.service.IOrganizationService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.validation.Valid;

import java.util.List;

/\*\*

\* 组织管理控制器

\* 提供组织单位的创建、更新、删除和查询功能

\*/

@RestController

@RequestMapping("/api/organizations")

public class OrganizationController {

@Autowired

private IOrganizationService organizationService;

/\*\*

\* 获取组织列表

\* @return 组织列表

\*/

@GetMapping

public List getOrganizations() {

return organizationService.list();

}

/\*\*

\* 获取组织详情

\* @param id 组织ID

\* @return 组织详细信息

\*/

@GetMapping("/{id}")

public Organization getOrganizationById(@PathVariable Long id) {

return organizationService.getById(id);

}

/\*\*

\* 创建组织（需要管理员权限）

\* @param organization 组织信息

\* @return 是否创建成功

\*/

@PostMapping

public boolean createOrganization(@Valid @RequestBody Organization organization) {

return organizationService.save(organization);

}

/\*\*

\* 更新组织（需要管理员权限）

\* @param id 组织ID

\* @param organization 更新后的组织信息

\* @return 是否更新成功

\*/

@PutMapping("/{id}")

public boolean updateOrganization(@PathVariable Long id, @Valid @RequestBody Organization organization) {

organization.setId(id);

return organizationService.updateById(organization);

}

/\*\*

\* 删除组织（需要管理员权限）

\* @param id 组织ID

\* @return 是否删除成功

\*/

@DeleteMapping("/{id}")

public boolean deleteOrganization(@PathVariable Long id) {

return organizationService.removeById(id);

}

}

**实现说明**

**1. 功能覆盖**

以上控制器完全实现了您提供的 API 接口设计：

* **项目管理**：GET /api/projects, GET /api/projects/{id}, POST /api/projects, PUT /api/projects/{id}, DELETE /api/projects/{id}
* **报名管理**：GET /api/applications, GET /api/applications/user/{userId}, POST /api/applications, PUT /api/applications/{id}, DELETE /api/applications/{id}
* **新闻管理**：GET /api/news, GET /api/news/{id}, POST /api/news, PUT /api/news/{id}, DELETE /api/news/{id}
* **服务记录管理**：GET /api/records, GET /api/records/user/{userId}, POST /api/records, PUT /api/records/{id}, DELETE /api/records/{id}
* **组织管理**：GET /api/organizations, GET /api/organizations/{id}, POST /api/organizations, PUT /api/organizations/{id}, DELETE /api/organizations/{id}
* **用户管理**：GET /api/users, POST /api/users, PUT /api/users/{id}, DELETE /api/users/{id}

**2. 权限控制**

* 当前代码仅展示了基本的 CRUD 操作，实际应用中需添加权限控制逻辑。例如：
  + 使用 **Spring Security** 或 **JWT** 验证用户身份和角色。
  + 在方法上添加注解（如 @PreAuthorize("hasRole('ADMIN')")）限制管理员权限。
* 权限说明已在注释中标明，您可根据需求进一步实现。

**3. 数据校验**

* 使用 @Valid 注解对 @RequestBody 参数进行后端校验。
* 需在实体类中添加具体校验规则（如 @NotNull, @Size 等），以确保输入数据合法性。
* 前端校验需在 Vue 页面中结合 Element Plus 的表单组件实现。

**4. 分页功能**

* 当前 GET 接口返回全量数据，实际应用中可通过添加 page 和 size 参数，使用 MyBatis-Plus 的 Page 类实现分页。

**5. 异常处理**

* 当前代码返回简单的布尔值或实体对象，建议添加全局异常处理器，返回统一的响应格式（如 { "code": 200, "message": "成功", "data": {} }）。

**下一步建议**

* **权限控制**：集成 JWT 和 Spring Security。
* **分页支持**：在 GET 接口中添加分页参数。
* **响应封装**：定义统一响应类 ResponseResult，优化返回格式。
* **异常处理**：添加 @RestControllerAdvice 处理全局异常。

如果您需要进一步扩展代码（如添加上述功能）或生成其他部分（如前端代码、README 部署指南），请随时告知！