

# 大作业：HNU校园交流平台 实验报告

## 大作业：HNU校园交流平台 实验报告

### 一、数据库设计部分

1. 需求分析
  - 1.1 背景概述
  - 1.2 功能需求
  - 1.3 数据需求
2. 概念结构设计
  - 2.1 实体
  - 2.2 E-R图
3. 逻辑结构设计
  - 3.1 模型选择
  - 3.2 主题划分
  - 3.3 依赖分析
  - 3.4 联系
  - 3.5 用户子模式
4. 物理结构设计
  - 4.1 具体设计
  - 4.2 关系实体
  - 4.3 存取方法
  - 4.4 存储结构
5. 运行测试
  - 5.1 验证实体间联系
  - 5.2 验证实体完整性
  - 5.3 验证参照完整性
  - 5.4 验证用户定义的完整性
  - 5.5 视图查看信息

### 二、软件设计部分

1. Web技术栈
2. 概要设计（功能）
  - 2.1 面向群体
  - 2.2 基本要求
  - 2.3 功能需求
3. 详细设计（代码）
  - 3.1 服务器
  - 3.2 客户端
4. 界面设计（用户说明）
  - 4.1 学生端
  - 4.2 管理员端

### 三、实验总结

## 成员分工

- 李超程：需求分析、软件设计部分
- 王君悦：概念结构设计、逻辑结构设计
- 张晓博：物理结构设计
- 毛惠兰：系统架构设计、软件设计部分、实验报告撰写
- 柳曾斌：界面原型设计

# 一、数据库设计部分

## 1. 需求分析

### 1.1 背景概述

随着湖南大学在校学生规模不断扩大，目前全日制学生人数已超过四万人，校园内每天都会产生大量与学习、生活相关的信息。这些信息类型多样、更新频繁，主要依赖微信群、QQ群、朋友圈等非结构化渠道进行传播，存在信息分散、时效性差、重复发布、难以检索等问题。对于新入学学生而言，获取考试安排、课程信息、二手交易、租房、校园生活经验等内容往往成本较高，适应校园环境存在一定困难。

基于上述背景，本项目拟设计并实现一个面向湖南大学学生的校园信息交流平台。平台以浏览器—服务器（BS）架构为基础，通过统一的信息发布与管理机制，对校园内各类信息进行集中展示与持久化存储，提升信息获取效率，促进学生之间的交流互动。同时，通过引入管理员审核机制，保证平台内容的真实性与合规性，营造健康、有序的校园交流环境。

### 1.2 功能需求

#### 学生端

- 注册与身份认证：**用户需通过手机号验证码注册，并提交学号、校园卡照片进行实名审核，确保用户均为本校学生。
- 信息发布：**支持“二手闲置、打听求助、恋爱交友、校园趣事、考试信息”五大类别的帖子发布。系统要求标题为4-20字，并计划接入大模型API进行内容合规性检测。
- 搜索：**支持按标题或内容全文检索。
- 互动：**用户可以在帖子下留言评论、进行深入探讨，并能点赞感兴趣的内容。
- 热榜浏览：**系统根据热度算法自动排序，展示热度前10名的“热搜”帖子。
- 个人管理：**学生可以维护个人资料（昵称、联系方式等），查看或删除自己发布的历史帖子。

#### 管理员端

- 注册审核：**人工核验用户提交的学号及校园证件照片，拒绝不合法的注册申请。
- 帖子管理：**对全站帖子进行管理，有权删除违规或不合法的帖子。
- 用户状态管理：**针对散布违规信息的用户，管理员可对其禁言，以维护良好的校园社区生态。

### 1.3 数据需求

#### 1. 数据结构：用户数据（`users`）

- 含义说明：**用于存储平台中所有用户的基本信息、登录认证信息及用户角色状态，是系统实现身份认证、权限控制和用户管理的核心数据表。平台通过该表区分学生用户与管理员用户，并支持对用户进行禁言、审核等管理操作。
- 组成：**包括用户唯一标识、手机号、昵称、登录密码、学号、校园卡照片地址、认证状态、用户角色、禁言状态、创建时间及更新时间等数据字段。

#### 2. 数据结构：帖子分类数据（`post_categories`）

- 含义说明：**用于统一管理平台中帖子的分类信息，避免分类字段散乱，提升系统的可维护性和扩展性。所有帖子均通过外键方式关联具体分类。
- 组成：**包括分类唯一标识、分类代码、分类名称、分类描述、排序顺序、启用状态、创建时间及更新时间等字段。

#### 3. 数据结构：帖子数据（`posts`）

- **含义说明：**用于存储学生在平台发布的各类信息帖，是校园交流平台的核心业务数据。支持分类管理、全文搜索、热度统计及状态控制。
- **组成：**包括帖子唯一标识、标题、正文内容、所属分类标识、发布者标识、联系方式、浏览量、点赞数、热度值、帖子状态、全文检索向量、创建时间及更新时间等字段。

#### 4. 数据结构：评论数据 (`comments`)

- **含义说明：**用于记录用户在帖子下的评论及评论回复信息，实现多层级互动交流功能，支持对评论进行管理与状态控制。
- **组成：**包括评论唯一标识、所属帖子标识、评论用户标识、评论内容、父评论标识（用于回复功能）、点赞数、评论状态、创建时间及更新时间等字段。

#### 5. 数据结构：帖子点赞数据 (`post_likes`)

- **含义说明：**用于记录用户对帖子的点赞行为，防止同一用户对同一帖子重复点赞，同时为帖子点赞数与热度计算提供数据支持。
- **组成：**包括点赞记录唯一标识、帖子标识、用户标识及点赞时间等字段，并通过唯一性约束保证点赞行为的唯一性。

#### 6. 数据结构：评论点赞数据 (`comment_likes`)

- **含义说明：**用于记录用户对评论的点赞行为，支持评论热度展示，并防止重复点赞。
- **组成：**包括点赞记录唯一标识、评论标识、用户标识及点赞时间等字段，并设置联合唯一约束限制重复操作。

## 2. 概念结构设计

包含用户表 `users`、帖子分类表 `post_categories`、帖子表 `posts`、评论表 `comments`、帖子点赞表 `post_likes`、评论点赞表 `comment_likes` 6个实体。每个实体的属性、码如下：

### 2.1 实体

#### 1. 用户表 (`users`)

字段名	数据类型	备注
<code>id</code>	BIGSERIAL	用户ID，自增主键
<code>phone</code>	VARCHAR(11)	手机号，唯一，用于登录
<code>nickname</code>	VARCHAR(50)	用户昵称
<code>password</code>	VARCHAR(255)	登录密码，加密存储
<code>student_id</code>	VARCHAR(20)	学号
<code>campus_card_url</code>	VARCHAR(500)	校园卡照片URL
<code>auth_status</code>	VARCHAR(20)	认证状态: pending / approved / rejected
<code>role</code>	VARCHAR(20)	用户角色: STUDENT / ADMIN
<code>is_muted</code>	BOOLEAN	是否被禁言
<code>create_time</code>	TIMESTAMP	创建时间
<code>update_time</code>	TIMESTAMP	更新时间

## 2. 帖子分类表 (post\_categories)

字段名	数据类型	备注
id	SERIAL	分类ID, 自增主键
category_code	VARCHAR(20)	分类代码 (唯一)
category_name	VARCHAR(50)	分类名称
description	VARCHAR(200)	分类描述
sort_order	INT	排序顺序
is_active	BOOLEAN	是否启用
create_time	TIMESTAMP	创建时间
update_time	TIMESTAMP	更新时间

## 3. 帖子表 (posts)

字段名	数据类型	备注
id	BIGSERIAL	帖子ID, 自增主键
title	VARCHAR(20)	帖子标题 (4-20字)
content	TEXT	帖子正文内容
category_id	INT	分类ID, 外键
author_id	BIGINT	发布者ID, 外键
contact_info	VARCHAR(200)	联系方式 (可选)
view_count	INT	浏览量
like_count	INT	点赞数
hot_score	DECIMAL(10,2)	热度值, 用于排序
status	VARCHAR(20)	状态: normal / deleted
search_vector	TSVECTOR	全文检索向量
create_time	TIMESTAMP	创建时间
update_time	TIMESTAMP	更新时间

## 4. 评论表 (comments)

字段名	数据类型	备注
id	BIGSERIAL	评论ID, 自增主键
post_id	BIGINT	帖子ID, 外键
user_id	BIGINT	评论用户ID, 外键
content	TEXT	评论内容
parent_id	BIGINT	父评论ID (回复用)
like_count	INT	点赞数
status	VARCHAR(20)	状态: normal / deleted
create_time	TIMESTAMP	创建时间
update_time	TIMESTAMP	更新时间

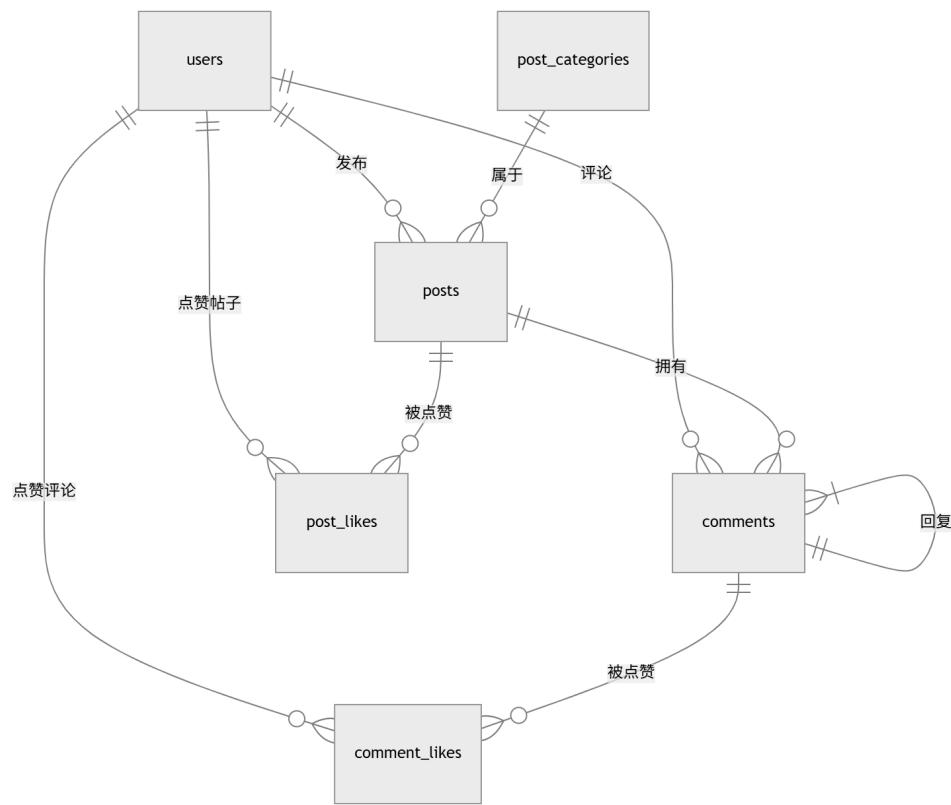
#### 5. 帖子点赞表 (post\_likes)

字段名	数据类型	备注
id	BIGSERIAL	点赞记录ID
post_id	BIGINT	帖子ID, 外键
user_id	BIGINT	用户ID, 外键
create_time	TIMESTAMP	点赞时间

#### 6. 评论点赞表 (comment\_likes)

字段名	数据类型	备注
id	BIGSERIAL	点赞记录ID
comment_id	BIGINT	评论ID, 外键
user_id	BIGINT	用户ID, 外键
create_time	TIMESTAMP	点赞时间

## 2.2 E-R图



## 3. 逻辑结构设计

### 3.1 模型选择

选用关系数据模型来实现相应的设计。

### 3.2 主题划分

分成学生主题模块和管理员主题模块，具体如下：

#### 学生主题模块

学生主题模块面向普通学生用户，主要支持信息发布、浏览与互动等功能。该模块由以下实体构成：

- 用户**：有用户ID、手机号、昵称、登录密码、学号、校园卡照片地址、认证状态、角色标识、禁言状态、创建时间、更新时间11个属性。
- 帖子分类**：有分类ID、分类代码、分类名称、分类描述、排序顺序、是否启用、创建时间、更新时间8个属性。
- 帖子**：有帖子ID、标题、正文内容、分类ID、发布者ID、联系方式、浏览量、点赞数、热度值、帖子状态、全文检索向量、创建时间、更新时间13个属性。
- 评论**：有评论ID、帖子ID、评论用户ID、评论内容、父评论ID、点赞数、评论状态、创建时间、更新时间9个属性。
- 帖子点赞**：有点赞记录ID、帖子ID、用户ID、点赞时间4个属性。
- 评论点赞**：有点赞记录ID、评论ID、用户ID、点赞时间4个属性。

#### 管理员主题模块

管理员主题模块用于平台运行管理和内容审核，保障平台秩序与信息安全。该模块由以下实体构成：

- 用户**: 有用户ID、手机号、昵称、登录密码、学号、校园卡照片地址、认证状态、角色标识、禁言状态、创建时间、更新时间11个属性。
- 帖子**: 有帖子ID、标题、正文内容、分类ID、发布者ID、联系方式、浏览量、点赞数、热度值、帖子状态、全文检索向量、创建时间、更新时间13个属性。
- 评论**: 有评论ID、帖子ID、评论用户ID、评论内容、父评论ID、点赞数、评论状态、创建时间、更新时间9个属性。

### 3.3 依赖分析

#### 1. 用户 (用户ID, 手机号, 昵称, 登录密码, 学号, 校园卡照片地址, 认证状态, 角色标识, 禁言状态, 创建时间, 更新时间)

由于用户实体中用户ID是候选码，能够唯一标识一个用户，因此不存在非主属性对码的部分函数依赖；另外，除候选码用户ID以外的所有非主属性，如手机号、昵称、登录密码、学号、校园卡照片地址、认证状态、角色标识、禁言状态以及时间属性，均只能通过用户ID唯一推出，不存在非主属性之间的传递依赖，而且由于用户ID是唯一的决定因素，因此该关系模式不仅满足第三范式，同时也满足 BCNF。

#### 2. 帖子分类 (分类ID, 分类代码, 分类名称, 分类描述, 排序顺序, 是否启用, 创建时间, 更新时间)

由于帖子分类实体中分类ID是候选码，能够唯一确定一条分类记录，因此不存在非主属性对码的部分函数依赖；同时，分类名称、分类描述、排序顺序、是否启用及时间属性等均完全依赖于候选码分类ID，不存在非主属性经由其他非主属性间接决定的传递依赖，而且分类ID作为唯一决定因素，该关系模式满足第三范式，并符合 BCNF 的要求。

#### 3. 帖子 (帖子ID, 标题, 正文内容, 分类ID, 发布者ID, 联系方式, 浏览量, 点赞数, 热度值, 帖子状态, 全文检索向量, 创建时间, 更新时间)

由于帖子实体中帖子ID是候选码，能够唯一标识一条帖子记录，因此不存在非主属性对码的部分函数依赖；此外，标题、正文内容、分类ID、发布者ID、联系方式、浏览量、点赞数、热度值、帖子状态、全文检索向量以及时间属性等均只能由帖子ID唯一决定，外键属性不会引入传递依赖，因此该关系模式不存在传递依赖，且帖子ID是唯一的决定因素，从而该关系模式满足第三范式并满足 BCNF。

#### 4. 评论 (评论ID, 帖子ID, 评论用户ID, 评论内容, 父评论ID, 点赞数, 评论状态, 创建时间, 更新时间)

由于评论实体中评论ID是候选码，能够唯一确定一条评论，因此不存在非主属性对码的部分函数依赖；同时，帖子ID、评论用户ID、评论内容、父评论ID、点赞数、评论状态及时间属性等均完全依赖于候选码评论ID，不存在通过其他非主属性间接决定的传递依赖，而且评论ID作为唯一决定因素，该关系模式满足第三范式，并符合 BCNF。

#### 5. 帖子点赞 (点赞记录ID, 帖子ID, 用户ID, 点赞时间)

由于帖子点赞实体中点赞记录ID是候选码，能够唯一标识一条点赞记录，因此不存在非主属性对码的部分函数依赖；另外，帖子ID、用户ID以及点赞时间等属性均只能通过候选码点赞记录ID推出，不存在非主属性之间的传递依赖，而且候选码点赞记录ID是唯一的决定因素，因此该关系模式满足第三范式，并满足 BCNF。

#### 6. 评论点赞 (点赞记录ID, 评论ID, 用户ID, 点赞时间)

由于评论点赞实体中点赞记录ID是候选码，能够唯一确定一条点赞记录，因此不存在非主属性对码的部分函数依赖；同时，评论ID、用户ID和点赞时间等属性均完全依赖于候选码点赞记录ID，不存在传递依赖，而且由于候选码点赞记录ID是唯一的决定因素，所以该关系模式满足第三范式，并满足 BCNF。

## 3.4 联系

实体间的联系如下：

### 一对多 (1:N)

1. **用户-帖子**：一名用户可以发布多篇帖子，但每篇帖子仅属于一个作者。
2. **用户-评论**：一名用户可以发表多条评论，但每条评论仅由一个用户撰写。
3. **用户-帖子点赞**：一名用户可以产生多条点赞记录，但每条记录仅对应一个操作者。
4. **用户-评论点赞**：一名用户可以为多条评论点赞，每条记录仅记录一个点赞者。
5. **帖子分类-帖子**：一个分类下可以包含多个帖子，但每个帖子只能归属于一个分类。
6. **帖子-评论**：一个帖子可以拥有多条评论，但每条评论仅属于一个特定的帖子。
7. **帖子-帖子点赞**：一个帖子可以获得多次点赞，每条记录对应一次具体的点赞行为。
8. **评论-评论点赞**：一条评论可以获得多次点赞，每条记录对应一次具体的点赞行为。
9. **父评论-子评论（回复）**：一条评论（作为父级）可以拥有多条回复（作为子级），每条回复仅指向一个父级评论。

## 3.5 用户子模式

创建用户信息视图 `users_info`，是因为平台中大量业务场景需要频繁查询用户的基础信息，如帖子展示、评论展示、后台管理等，为简化查询操作，并同时保证用户密码等敏感隐私信息不被暴露，将只包含必要字段的用户信息独立出来，既提高了使用便利性，也增强了系统的数据安全性。

### 视图定义

```
5  CREATE OR REPLACE VIEW users_info AS
6  SELECT
7      id,
8      phone,
9      nickname,
10     student_id,
11     campus_card_url,
12     create_time
13  FROM users;
```

将创建视图的语句导入数据库：

The screenshot shows a PostgreSQL database interface with a query editor window. The title bar of the window is labeled 'users\_info\_create.sql'. The query itself is:

```
-- =====
-- 用户信息视图 (不包含敏感信息)
-- 仅展示: 用户ID、手机号、昵称、学号、校园卡、注册时间
CREATE OR REPLACE VIEW users_info AS
SELECT
    id,
    phone,
    nickname,
    student_id,
    campus_card_url,
    create_time
FROM users;
```

Below the query window, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the message 'CREATE VIEW'. At the bottom of the 'Messages' tab, a red-bordered box highlights the message 'Query returned successfully in 224 msec.'

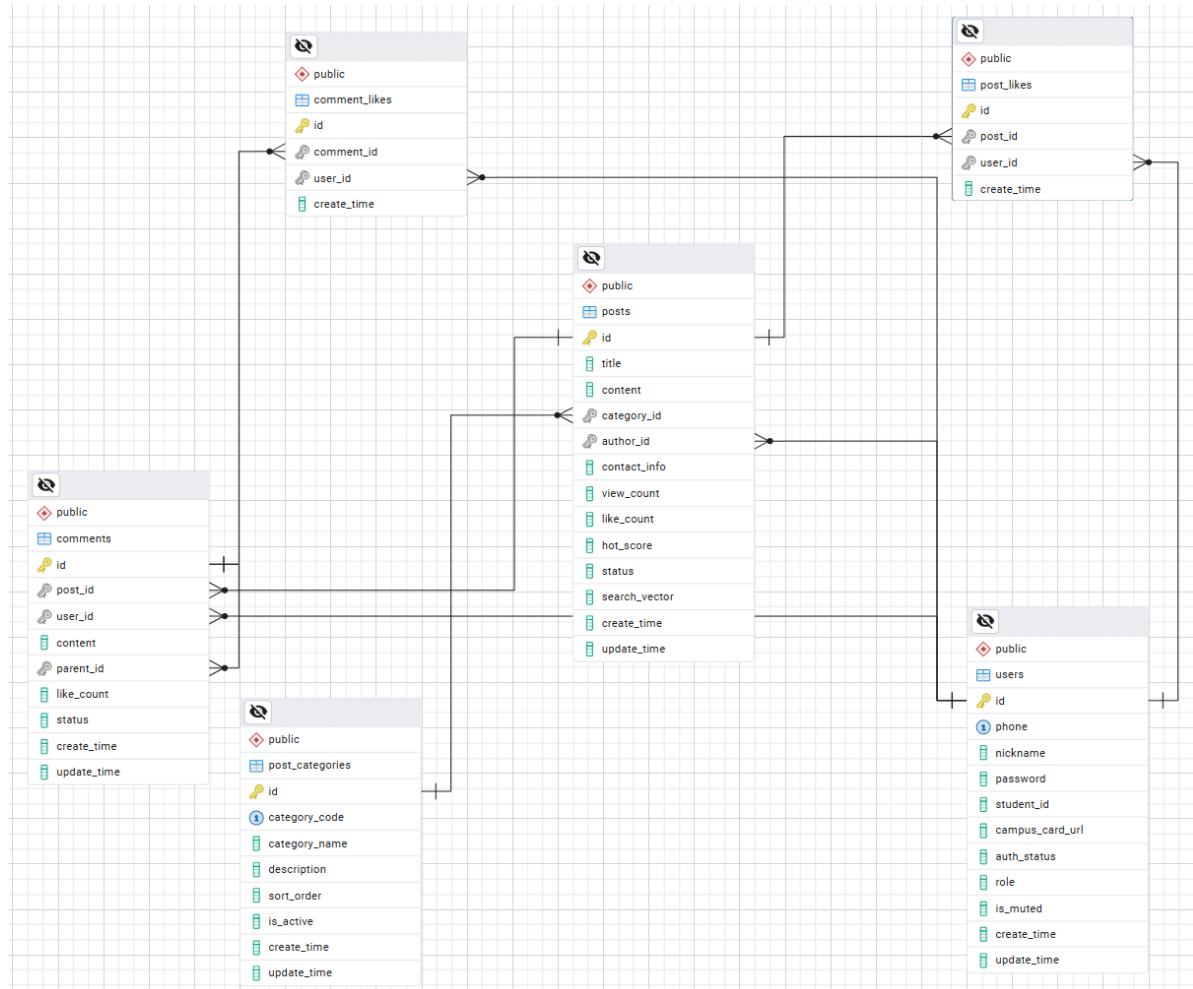
Views (1)  
users\_info

创建视图成功。

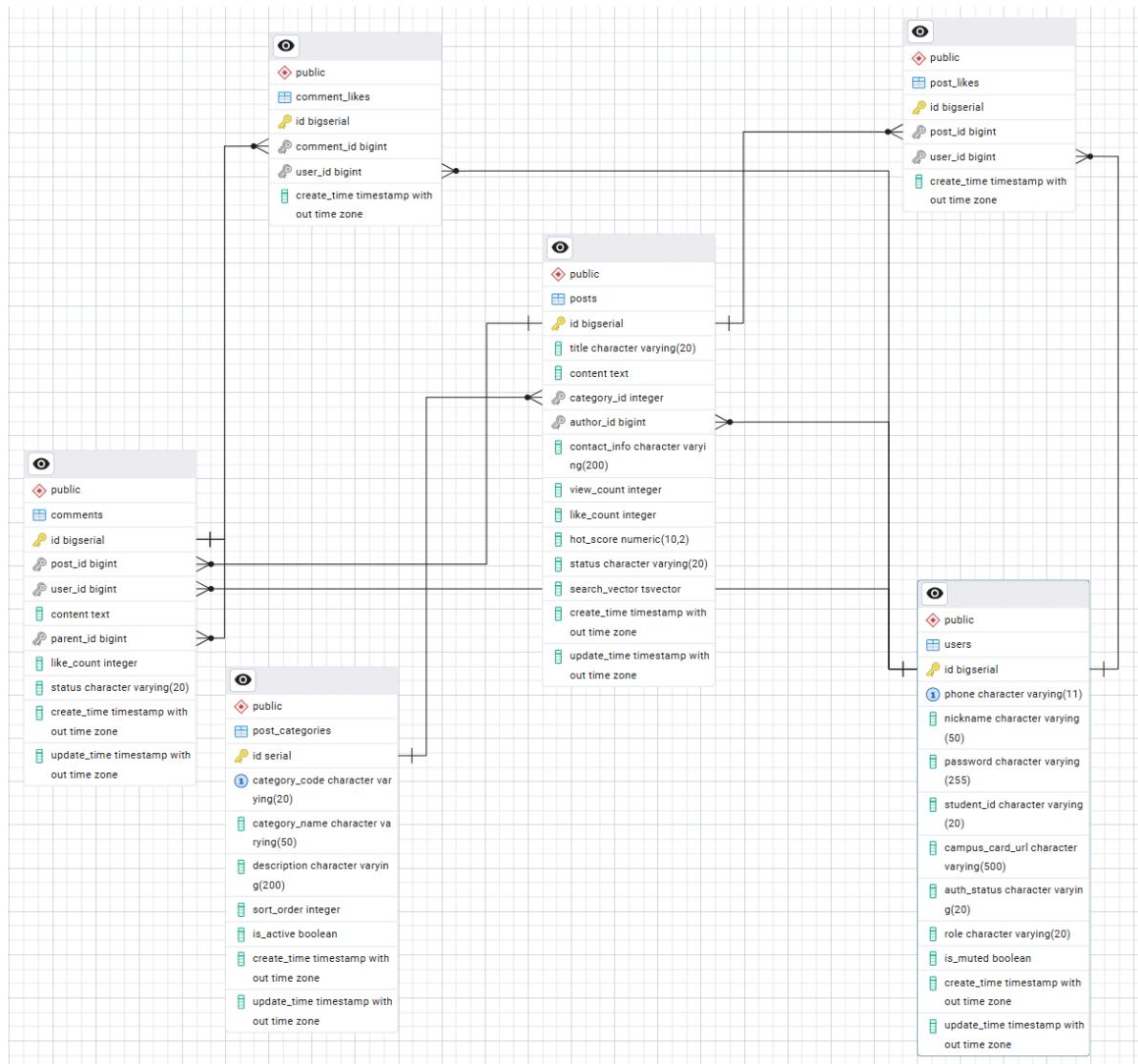
## 4. 物理结构设计

### 4.1 具体设计

根据优化和调整，最后的逻辑模型如下：



最后的物理模型如下：



## 4.2 关系实体

一个关系对应一张表，根据最后的物理模型，我们需要6张表来实现关系数据模型。

### 1. 用户实体

各属性类型长度定义：

users								
	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default	
<input type="button" value="edit"/>	<input type="button" value="trash"/>	id	bigint		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('1')	
<input type="button" value="edit"/>	<input type="button" value="trash"/>	phone	character varying	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
<input type="button" value="edit"/>	<input type="button" value="trash"/>	nickname	character varying	50	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
<input type="button" value="edit"/>	<input type="button" value="trash"/>	password	character varying	255	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
<input type="button" value="edit"/>	<input type="button" value="trash"/>	student_id	character varying	20	<input type="checkbox"/>	<input type="checkbox"/>		
<input type="button" value="edit"/>	<input type="button" value="trash"/>	campus_card_u	character varying	500	<input type="checkbox"/>	<input type="checkbox"/>		
<input type="button" value="edit"/>	<input type="button" value="trash"/>	auth_status	character varying	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'pending'	
<input type="button" value="edit"/>	<input type="button" value="trash"/>	role	character varying	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'STUDEN	
<input type="button" value="edit"/>	<input type="button" value="trash"/>	is_muted	boolean		<input checked="" type="checkbox"/>	<input type="checkbox"/>	false	
<input type="button" value="edit"/>	<input type="button" value="trash"/>	create_time	timestamp without time zone		<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT	
<input type="button" value="edit"/>	<input type="button" value="trash"/>	update_time	timestamp without time zone		<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT	

- 主键：用户ID `id`
- 各属性值约束：
  - 认证状态 `auth_status` 只能取以下三个值：`'pending'`、`'approved'`、`'rejected'`
  - 角色 `role` 只能取以下两个值：`'STUDENT'`、`'ADMIN'`
  - 手机号 `phone` 为11位手机号且取值唯一

The screenshot shows two tables from the PostgreSQL pg\_constraint system catalog:

Primary Key		Foreign Key		Check		Unique		Exclude	
		Name		Check					
		chk_auth_status		auth_status::text = ANY (ARRAY['pending'::character varying,					
		chk_role		role::text = ANY (ARRAY['STUDENT'::character varying, 'ADM					
		chk_phone_format		phone::text ~ '^1[3-9]\d{9}\$'::text					

Primary Key		Foreign Key		Check		Unique		Exclude	
		Name		Columns					
		users_phone_key		phone					

## 2. 帖子分类实体

各属性类型长度定义：

The screenshot shows the pg\_attribute table for the `post_categories` table:

post_categories									
General		Columns		Advanced		Constraints		Partitions	
Inherited from table(s)		Select to inherit from...							
Columns									
	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default		
		id	integer		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('1')		
		category_code	character varying	20	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
		category_name	character varying	50	<input checked="" type="checkbox"/>	<input type="checkbox"/>			
		description	character varying	200	<input type="checkbox"/>	<input type="checkbox"/>			
		sort_order	integer		<input checked="" type="checkbox"/>	<input type="checkbox"/>	0		
		is_active	boolean		<input checked="" type="checkbox"/>	<input type="checkbox"/>	true		
		create_time	timestamp without time zone		<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT		
		update_time	timestamp without time zone		<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT		

- 主键：分类ID `id`
- 各属性值约束：
  - 帖子分类代码 `category_code` 取值唯一

The screenshot shows the pg\_constraint table for the `post_categories` table:

Primary Key		Foreign Key		Check		Unique		Exclude	
		Name		Columns					
		post_categories_category_code_key		category_code					

## 3. 帖子实体

各属性类型长度定义：

posts

General Columns Advanced Constraints Partitions Parameters Security SQL

**Columns**

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	id	bigint			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('')
	title	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	content	text			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	category_id	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	author_id	bigint			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	contact_info	character varying	200		<input type="checkbox"/>	<input type="checkbox"/>	
	view_count	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	0
	like_count	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	0
	hot_score	numeric	10	2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0.00
	status	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	'normal'
	search_vector	tsvector			<input type="checkbox"/>	<input type="checkbox"/>	
	create_time	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
	update_time	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP

- 主键：帖子ID `id`
- 外键：
  - 分类ID `category_id` → `post_categories(id)` (关联帖子分类)
  - 作者ID `author_id` → `users(id)` (关联发帖用户)
- 各属性值约束：
  - 帖子状态 `status` 的值只能是 `'normal'` 或 `'deleted'`
  - 帖子标题 `title` 的长度在4到20个字符之间

Primary Key Foreign Key Check Unique Exclude

	Name	Columns	Referenced Table
	fk_posts_author	(author_id) -> (id)	public.users
	fk_posts_category	(category_id) -> (id)	public.post_categories

Primary Key Foreign Key Check Unique Exclude

	Name	Check
	chk_posts_status	status::text = ANY (ARRAY['normal'::character varying, 'delet'])
	chk_posts_title_length	char_length(title::text) >= 4 AND char_length(title::text) <= 20

#### 4. 评论实体

各属性类型长度定义：

comments

General Columns Advanced Constraints Partitions Parameters Security SQL

**Columns**

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	id	bigint			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('')
	post_id	bigint			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	user_id	bigint			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	content	text			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	parent_id	bigint			<input type="checkbox"/>	<input type="checkbox"/>	
	like_count	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	0
	status	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	'normal'::character varying
	create_time	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP
	update_time	timestamp without time zone			<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP

- 主键：评论ID `id`
- 外键：
  - 帖子ID `post_id` → `posts(id)` ON DELETE CASCADE (关联所属帖子)
  - 用户ID `user_id` → `users(id)` (关联评论用户)
  - 父评论ID `parent_id` → `comments(id)` ON DELETE CASCADE (关联父评论，支持嵌套回复)
- 各属性值约束：
  - 评论状态 `status` 的值只能是 `'normal'` 或 `'deleted'`

Foreign Key

Name		Columns	Referenced Table
	fk_comments_parent	(parent_id) -> (id)	public.comments
	fk_comments_post	(post_id) -> (id)	public.posts
	fk_comments_user	(user_id) -> (id)	public.users

Check

Name	Check	
	chk_comments_status	status::text = ANY (ARRAY['normal'::character varying, 'deleted'::character varying])

## 5. 帖子点赞实体

各属性类型长度定义：

**post\_likes**

General Columns Advanced Constraints Partitions Parameters Security SQL

Inherited from table(s) Select to inherit from...

**Columns**

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	id	bigint	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('f')
	post_id	bigint	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	user_id	bigint	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	create_time	timestamp without time zone	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP

- 主键: 点赞记录ID `id`
- 外键:
  - 帖子ID `post_id` → `posts(id)` ON DELETE CASCADE
  - 用户ID `user_id` → `users(id)` ON DELETE CASCADE
- 各属性值约束:
  - `(post_id, user_id)` 复合键取值唯一

Primary Key Foreign Key Check Unique Exclude

	Name	Columns	Referenced Table
	fk_post_likes_post	<code>(post_id) -&gt; (id)</code>	public.posts
	fk_post_likes_user	<code>(user_id) -&gt; (id)</code>	public.users

Primary Key Foreign Key Check Unique Exclude

	Name	Columns
	uk_post_likes	<code>post_id, user_id</code>

## 6. 评论点赞实体

各属性类型长度定义:

**comment\_likes**

General Columns Advanced Constraints Partitions Parameters Security SQL

Inherited from table(s) Select to inherit from...

**Columns**

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	id	bigint	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	nextval('c')
	comment_id	bigint	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	user_id	bigint	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	create_time	timestamp without time zone	<input type="button" value="▼"/>		<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP

- 主键: 点赞记录ID `id`
- 外键:
  - 评论ID `comment_id` → `comments(id)` ON DELETE CASCADE
  - 用户ID `user_id` → `users(id)` ON DELETE CASCADE
- 各属性值约束:

## ■ (comment\_id,user\_id) 复合键取值唯一

The screenshot shows the pgAdmin interface with two tables displayed. The top table is for 'Foreign Key' and contains two entries:

Name	Columns	Referenced Table
fk_comment_likes_comment	(comment_id) -> (id)	public.comments
fk_comment_likes_user	(user_id) -> (id)	public.users

The bottom table is for 'Unique' and contains one entry:

Name	Columns
uk_comment_likes	comment_id,user_id

## 4.3 存取方法

在我们的设计中，主要采用 B-Tree 索引来优化存取性能。B-Tree 索引是关系型数据库中最常用的索引类型，适合等值查询、范围查询以及排序操作。平台中用户、帖子、评论及点赞等核心数据表，查询频率普遍高于更新频率，因此 B-Tree 索引能够快速定位记录、支持分页排序，并有效提升按条件筛选和统计分析的效率。例如，用户按时间排序浏览帖子、统计点赞数和评论数量等场景，B-Tree 索引都能提供高效的响应性能，满足校园交流平台读多写少、查询操作频繁的特点。

对于帖子表 (posts) 中的标题和内容字段，我们单独设置了 search\_vector 字段并采用 GIN 索引。原因在于标题和内容属于大文本字段，用户需要支持全文搜索和模糊匹配。传统的 B-Tree 索引不适合大文本的匹配查询，而 GIN 索引能够高效地对文本进行全文检索，支持多词匹配、模糊查询以及排名排序，极大提升了搜索功能的性能，使平台用户能够快速找到感兴趣的帖子，实现高效的信息检索体验。

代码示例：

```
1 -- 帖子表索引
2 CREATE INDEX idx_posts_category_id ON posts(category_id);
3 CREATE INDEX idx_posts_author_id ON posts(author_id);
4 CREATE INDEX idx_posts_status ON posts(status);
5 CREATE INDEX idx_posts_hot_score ON posts(hot_score DESC, create_time DESC);
6 CREATE INDEX idx_posts_create_time ON posts(create_time DESC);
7 CREATE INDEX idx_posts_view_count ON posts(view_count DESC);
8 CREATE INDEX idx_posts_like_count ON posts(like_count DESC);
9 -- 全文检索索引 (GIN索引, 支持快速全文搜索)
10 CREATE INDEX idx_posts_search_vector ON posts USING GIN(search_vector);
```

## 4.4 存储结构

在数据库中，数据表的存储结构是由存储引擎决定的。PostgreSQL 并不支持多种存储引擎，因此我们的数据库采用 PostgreSQL 默认的 Heap 存储，数据以无序堆方式存放，支持 MVCC 多版本并发控制，保证事务安全和数据一致性。结合 B-Tree 索引加速常规查询、排序和统计操作，posts 表的 search\_vector 字段使用 GIN 索引支持高效全文检索，从而满足平台对高频查询、分页显示和搜索功能的需求。

## 5. 运行测试

执行逻辑模型生成的SQL文件，运行成功，结果如下：

The screenshot shows the pgAdmin interface with the following details:

- Toolbar:** Dashboard, SQL, Statistics, Dependencies, Dependents, Processes, baseSQL.sql.
- Session:** HNU\_Campus/postgres@PostgreSQL 16.
- Query History:** Shows the executed SQL code for creating the 'users' table.

```
11 -- =====
12 -- 1. 用户表 (users)
13 --
14 CREATE TABLE users (
15     id BIGSERIAL PRIMARY KEY,
16     phone VARCHAR(11) NOT NULL UNIQUE,
17     nickname VARCHAR(50) NOT NULL,
18     password VARCHAR(255) NOT NULL,
19     student_id VARCHAR(20),
20     campus_card_url VARCHAR(500),
21     auth_status VARCHAR(20) NOT NULL DEFAULT 'pending',
22     role VARCHAR(20) NOT NULL DEFAULT 'STUDENT',
23     is_muted BOOLEAN NOT NULL DEFAULT FALSE,
24     create_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
25     update_time TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
26     CONSTRAINT chk_auth_status CHECK (auth_status IN ('pending', 'approved', 'rejected'))
```

- Data Output:** Shows the message "Query returned successfully in 163 msec."
- Tables:** Shows a list of tables: comment\_likes, comments, post\_categories, post\_likes, posts, users.

向数据库中插入 `users` 数据：

The screenshot shows the pgAdmin interface with the following details:

- Toolbar:** Standard pgAdmin icons.
- Table View:** Shows the `users` table with 10 rows of data.

	<code>id</code> [PK] bigint	<code>phone</code> character varying (11)	<code>nickname</code> character varying (50)	<code>password</code> character varying (255)	<code>student_id</code> character varying (20)	<code>campus_card_url</code> character varying (500)
1	1	18800000001	user01	\$2a\$06\$TZkpAQx0bkWjyG8ArL6Duh/vuoq2PyQHzMMDRjiZKAzbP...	2021001001	[null]
2	2	18800000002	user02	\$2a\$06\$QJjzV.zebwTNpXHJLn0ewWu8yP7Jofy9IfpyGba19wMQ...	2021001002	[null]
3	3	18800000003	user03	\$2a\$06\$SWh5Mbqant0OUyCHiNBRPD05z13vHWjWkhk7zE8xAkz...	2021001003	[null]
4	4	18800000004	user04	\$2a\$06\$ufYFMj54tL.pSwMLmeNGpue3acNu8c3hGovMN04.VhJ0...	2021001004	[null]
5	5	18800000005	user05	\$2a\$06\$ShzaT69Fbjv5m3A/6jrXwOnMcjZN79MQfki.y9r6EYCB1q...	2021001005	[null]
6	6	18800000006	user06	\$2a\$06\$kqmHNrvOsMSNfiXxQ4x9DuBaQW9tEEr1G8SJTWwS2zpt...	2021001006	[null]
7	7	18800000007	user07	\$2a\$06\$SuolLXEsgp2ZTM.y7sd5le.gD070XuNORWlZUqTxflfX70...	2021001007	[null]
8	8	18800000008	user08	\$2a\$06\$eqsgnG/qMxmITE4ZHmT0DOJwL4nwNUSzhdn/HVwRZqS...	2021001008	[null]
9	9	18800000009	user09	\$2a\$06\$9lczOfYmn1kptsgUVn03dOu.VKWDZQw7nhl0HyADIAQsH/...	2021001009	[null]
10	10	18800000010	user10	\$2a\$06\$3BzsZWSJwCEXXuV/YolfwOPD0lfQfJKxoDXHSiDaSdjfcjt...	2021001010	[null]

### 5.1 验证实体间联系

#### (1) 用户-帖子

	<b>id</b> [PK] bigint	<b>title</b> character varying (20)	<b>content</b> text	<b>category_id</b> integer	<b>author_id</b> bigint	<b>contact_info</b> character varying (20)
13	13	求助NS2安装	Ubuntu装NS2总遇到依赖与编译报错，nam也打不开，求踩坑经验与检查清单。	2	18	手机: 18888880018
14	14	求帮看401问题	Spring Security + JWT登录能拿到token，但接口一直401，怀疑过滤器链顺序问题，求思路。	2	16	手机: 18888880016
15	15	求推荐组件库	React项目做后台管理，纠结Ant Design和MUI，尤其关心React 19兼容性与表格体验。	2	10	QQ: 10200010
16	16	求助Docker配置	Windows下VSCode连Docker经常找不到daemon或权限不足，求从安装到跑通compose的步...	2	23	手机: 18888880023
17	17	求推荐Markdown转Wo...	报告要交Word但我习惯写Markdown，公式代码目录崩就行，求好用的转换方案。	2	21	QQ: 10200021
18	18	求助依赖冲突	React 19装拖拽库提示peer依赖不支持，有没有替代库或解决办法，最好给最小示例。	2	3	手机: 18888880003
19	19	求推荐云服务器	准备部署Spring Boot并绑定二级域名，需要稳定入门云服务器，预算有限求避坑建议。	2	11	邮箱: user11@example.com
20	20	求助Excel排图	需要把(1)-(100)图片按顺序放进xlsx，每张图一行且无空白行，求脚本或Power Query方案。	2	23	手机: 18888880023

多条帖子对应同一个用户，满足一对多关系，插入成功。

## (2) 帖子-帖子分类

	<b>id</b> [PK] bigint	<b>title</b> character varying (20)	<b>content</b> text	<b>category_id</b> integer	<b>author_id</b> bigint	<b>contact_info</b> character varying (20)
5	5	出人体工学椅	人体工学椅可升降带腰托，坐垫不塌，适合久坐学习，校内自提。	1	23	手机: 18888880023
6	6	出二手平板	平板成色良好屏幕无划痕，适合记笔记看课件，送保护壳与充电器。	1	11	邮箱: user11@example.com
7	7	出网线水晶头	网线若干加一袋水晶头，适合做网络实验和练压线，打包出更便宜。	1	26	邮箱: user26@example.com
8	8	出计算机网络书	九成新教材，基本无笔记，附赠我整理的重点页码清单，期末复习友好。	1	15	QQ: 10200015
9	9	出二手单反	入门单反相机，功能正常，配基础镜头，适合摄影社新手练习，支持面验机。	1	17	微信: wx_17
10	10	出宿舍台灯	护眼台灯三档亮度，使用正常无频闪，适合夜间学习，便宜出。	1	5	微信: wx_05

多条帖子对应同一个帖子分类，满足多对一关系，插入成功。

## (3) 用户-评论

	<b>id</b> [PK] bigint	<b>post_id</b> bigint	<b>user_id</b> bigint	<b>content</b> text	<b>parent_id</b> bigint	<b>like_count</b> integer	<b>status</b> character varying (20)	<b>create_time</b> timestamp without time zone	<b>update_time</b> timestamp without time zone
19	19	10	5	台灯亮度够不够？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
20	20	10	12	有没有频闪？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
21	21	11	7	南门那家打印挺快的。	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
22	22	11	21	校内打印一般偏贵。	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
23	23	12	21	git add 文件名 就行。	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
24	24	12	3	别直接 git add .	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211

多条评论对应同一个用户，满足多对一关系，插入成功。

## (4) 用户-帖子点赞

	<b>id</b> [PK] bigint	<b>post_id</b> bigint	<b>user_id</b> bigint	<b>create_time</b> timestamp without time zone
37	37	13	7	2025-12-30 09:36:03.160969
38	38	11	16	2025-12-30 09:36:03.160969
39	39	14	16	2025-12-30 09:36:03.160969
40	40	15	16	2025-12-30 09:36:03.160969
41	41	12	21	2025-12-30 09:36:03.160969

一个用户可以点赞多条帖子，满足一对多关系，插入成功。

## (5) 用户-评论点赞

	<b>id</b> [PK] bigint	<b>comment_id</b> bigint	<b>user_id</b> bigint	<b>create_time</b> timestamp without time zone
33	33	10	13	2025-12-30 09:30:18.274227
34	34	10	18	2025-12-30 09:30:18.274227
35	35	11	7	2025-12-30 09:30:18.274227
36	36	12	7	2025-12-30 09:30:18.274227
37	37	13	7	2025-12-30 09:30:18.274227
38	38	11	16	2025-12-30 09:30:18.274227

一个用户可以点赞多条评论，满足一对多关系，插入成功。

## (6) 帖子-评论

	<code>id</code> [PK] bigint	<code>post_id</code> bigint	<code>user_id</code> bigint	<code>content</code> text	<code>parent_id</code> bigint	<code>like_count</code> integer	<code>status</code> character varying (20)	<code>create_time</code> timestamp without time zone	<code>update_time</code> timestamp without time zone
1	1	1	2	显示器用了多久？色彩还准吗？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
2	2	1	5	支持当面验货的话我可以看看。	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
3	3	2	9	路由器对宿舍信号覆盖还行吗？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
4	4	2	11	预算大概多少，可以参考下。	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
5	5	3	20	键盘是什么轴体？声音大不大？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
6	6	3	14	有原包装吗？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211
7	7	4	15	小冰箱功率多少？宿舍能用吗？	[null]	0	normal	2025-12-30 09:16:41.672211	2025-12-30 09:16:41.672211

多条评论对应同一个帖子，满足多对一关系，插入成功。

## (7) 帖子-帖子点赞

	<code>id</code> [PK] bigint	<code>post_id</code> bigint	<code>user_id</code> bigint	<code>create_time</code> timestamp without time zone
1		1	1	2025-12-30 09:36:03.160969
2		2	1	2025-12-30 09:36:03.160969
3		3	1	2025-12-30 09:36:03.160969
4		4	1	2025-12-30 09:36:03.160969
5		5	2	2025-12-30 09:36:03.160969

一条帖子可以有多个点赞，满足一对多关系，插入成功。

## (8) 评论-评论点赞

	<code>id</code> [PK] bigint	<code>comment_id</code> bigint	<code>user_id</code> bigint	<code>create_time</code> timestamp without time zone
1		1	1	2025-12-30 09:30:18.274227
2		2	1	2025-12-30 09:30:18.274227
3		3	1	2025-12-30 09:30:18.274227
4		4	1	2025-12-30 09:30:18.274227
5		5	2	2025-12-30 09:30:18.274227

一条评论可以有多个点赞，满足一对多关系，插入成功。

## (9) 父评论-子评论

	<code>id</code> [PK] bigint	<code>post_id</code> bigint	<code>user_id</code> bigint	<code>content</code> text	<code>parent_id</code> bigint	<code>like_count</code> integer	<code>status</code> character varying (20)	<code>create_time</code> timestamp without time zone	<code>update_time</code> timestamp without time zone
38	38	1	20	可以周末当面验。	2	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
39	39	12	16	对，新手最容易 add 多了。	23	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
40	40	12	25	我之前就踩过这个坑。	24	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
41	41	13	18	虚拟机确实会事。	26	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
42	42	14	16	我也是调整过滤器解决的。	27	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
43	43	15	10	后台项目我选 AntD。	29	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
44	44	16	23	WSL 网络模式也要注意。	31	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
45	45	17	21	Pandoc 配合模板很好用。	33	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
46	46	18	3	等生态稳定点再升级。	35	0	normal	2025-12-30 09:20:42.221818	2025-12-30 09:20:42.221818
47	47	1	9	我上次就是当面验的。	2	0	normal	2025-12-30 09:23:51.124765	2025-12-30 09:23:51.124765

多条评论对应同一条父评论，满足多对一关系，插入成功。

## 5.2 验证实体完整性

### (1) 主键重复

```
1  INSERT INTO users (id, phone, nickname, password)
2  VALUES (1, '13900000000', 'test2', '123456');
```

```
ERROR: duplicate key value violates unique constraint "users_pkey"
Key (id)=(1) already exists.
```

```
SQL state: 23505
Detail: Key (id)=(1) already exists.
```

可以看到，数据库拒绝插入，因为数据库里已经存在ID为1的用户了。

## (2) 主键为空

```
1  INSERT INTO users (id, phone, nickname, password)
2  VALUES (NULL, '13700000000', 'test3', '123456');
```

```
ERROR: null value in column "id" of relation "users" violates not-null constraint
Failing row contains (null, 13700000000, test3, 123456, null, null, pending, STUDENT, f, 2025-12-30 15:58:41.439464, 2025-12-30 15:58:41.439464).

SQL state: 23502
Detail: Failing row contains (null, 13700000000, test3, 123456, null, null, pending, STUDENT, f, 2025-12-30 15:58:41.439464, 2025-12-30 15:58:41.439464).
```

数据库同样拒绝插入，因为主键不允许为空。

## 5.3 验证参照完整性

### 被参照属性值不存在

```
1  INSERT INTO posts (title, content, category_id, author_id)
2  VALUES ('测试帖子', '测试内容', 1, 9999);
```

```
ERROR: insert or update on table "posts" violates foreign key constraint "fk_posts_author"
Key (author_id)=(9999) is not present in table "users".

SQL state: 23503
Detail: Key (author_id)=(9999) is not present in table "users".
```

可以看到，数据库拒绝插入，因为 `users` 表里没有ID为9999的用户。

## 5.4 验证用户定义的完整性

用户定义的完整性约束有很多，这里只放了一个示例：

### 认证状态不合法

```
1  INSERT INTO users (phone, nickname, password, auth_status)
2  VALUES ('13800000001', 'test_user2', '123456', 'waiting');
```

```
ERROR: new row for relation "users" violates check constraint "chk_auth_status"
Failing row contains (32, 13800000001, test_user2, 123456, null, null, waiting, STUDENT, f, 2025-12-30 16:06:59.048516, 2025-12-30 16:06:59.048516).

SQL state: 23514
Detail: Failing row contains (32, 13800000001, test_user2, 123456, null, null, waiting, STUDENT, f, 2025-12-30 16:06:59.048516, 2025-12-30 16:06:59.048516).
```

可以看到，数据库拒绝插入，因为认证状态 `auth_status` 只能取以下三个值：`'pending'`、`'approved'`、`'rejected'`。

## 5.5 视图查看信息

	<code>id</code> bigint	<code>phone</code> character varying (11)	<code>nickname</code> character varying (50)	<code>student_id</code> character varying (20)	<code>campus_card_url</code> character varying (500)	<code>create_time</code> timestamp without time zone
1	3	18800000003	user03	2021001003	[null]	2025-12-30 08:44:25.714856
2	4	18800000004	user04	2021001004	[null]	2025-12-30 08:44:25.714856
3	5	18800000005	user05	2021001005	[null]	2025-12-30 08:44:25.714856
4	6	18800000006	user06	2021001006	[null]	2025-12-30 08:44:25.714856
5	7	18800000007	user07	2021001007	[null]	2025-12-30 08:44:25.714856
6	8	18800000008	user08	2021001008	[null]	2025-12-30 08:44:25.714856
7	9	18800000009	user09	2021001009	[null]	2025-12-30 08:44:25.714856
8	10	18800000010	user10	2021001010	[null]	2025-12-30 08:44:25.714856

可以看到，视图成功返回用户的基本信息，而不返回密码等私密信息。起到了安全性保护的作用。

## 二、软件设计部分

### 1. Web技术栈

本项目采用前后端分离的 Web 架构进行设计与实现。

**后端：**基于 Spring Boot 框架构建，负责业务逻辑处理、接口服务及权限控制；数据层采用 PostgreSQL 作为关系型数据库，用于存储用户信息、帖子、评论及管理数据，并结合 Redis 实现验证码缓存、热点数据缓存和访问性能优化。

**前端：**基于 React 框架进行开发，采用 Ant Design 组件库构建用户界面，实现良好的交互体验与界面一致性。前后端通过 RESTful API 进行通信，实现用户注册与登录、帖子发布与浏览、评论互动、管理员审核与禁言等核心功能。

### 2. 概要设计（功能）

#### 2.1 面向群体

在校学生和平台管理员

#### 2.2 基本要求

1. 用户注册和登录
2. 查看和修改个人信息
3. 发布、浏览、搜索帖子
4. 评论、回复及点赞互动
5. 管理员审核用户及管理权限
6. 删除违规帖子及评论
7. 支持按热度排序帖子

#### 2.3 功能需求

##### 1. 学生

- 注册和登录账号
- 查看和修改个人信息（昵称、手机号、校园卡照片等）
- 发布帖子
- 浏览帖子列表
- 查看帖子详情
- 对帖子发表评论
- 回复他人评论
- 点赞帖子
- 点赞评论
- 搜索帖子（按标题或内容全文检索）

##### 2. 管理员

- 注册和登录账号
- 查看和修改个人信息（昵称、手机号等）
- 审核学生用户的认证状态

- 禁言或恢复用户发帖权限
- 删 除违规帖子
- 删 除违规评论

## 3. 详细设计（代码）

### 3.1 服务器

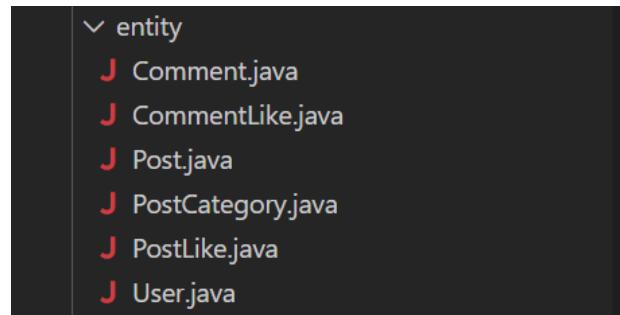
后端采用 Java Spring Boot 框架开发，按照分层架构，从上到下主要划分为 Controller（表示层）、Service（业务层）、Mapper（数据访问层）和 Entity / DTO（数据模型层）。Entity 用于映射数据库表结构，DTO 用于前后端数据传输，二者分离避免直接暴露数据库结构。业务流水：Controller --> Service --> Mapper（Mapper 接口）--> PostgreSQL

- 前端请求首先到达 Controller 层，Controller 使用 DTO 接收和封装请求参数，随后调用 Service 层执行业务逻辑；
- Service 层通过 Mapper 接口完成对数据库的访问。
- 数据库返回结果后，数据逐层向上返回，Service 层根据业务需要将 Entity 转换为 DTO，最终由 Controller 将处理结果返回给前端。

此外，系统还包括 Security（安全层），用于权限控制，为核心业务提供基础支撑。使用 JWT 鉴权，识别是否是 ADMIN。

#### 1. 实体层（entity）

定义数据库表对应的实体类，包含字段映射、数据类型和约束，直接对应 PostgreSQL 数据库表结构。



#### 2. 数据传输对象（DTO）

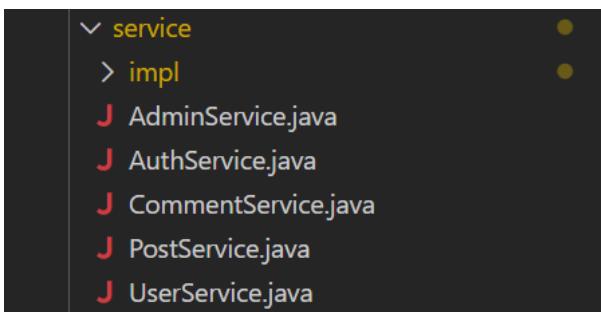
用于在前端与后端之间传递数据，解耦实体类与接口请求。按照功能模块进一步划分为：

- admin：管理员相关请求数据
- auth：认证与登录请求数据
- comment：评论相关请求数据
- common：通用请求数据
- post：帖子相关请求数据
- user：用户相关请求数据



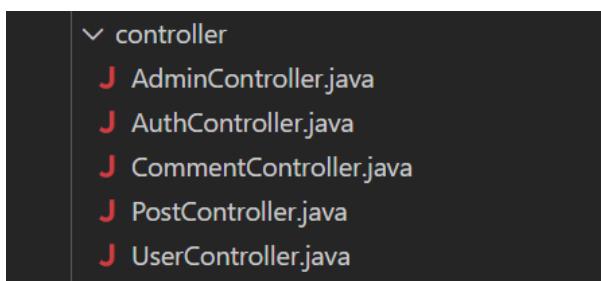
### 3. 服务层 (service)

封装具体业务逻辑，实现对实体、Mapper 的调用。service 接口定义业务操作规范，impl 实现具体逻辑。包括用户管理、帖子管理、评论管理、点赞操作等核心功能。



### 4. 控制器层 (controller)

负责处理客户端请求，接收前端传入的参数，调用服务层 (service) 完成业务逻辑处理，并将结果返回给前端。



#### ◦ AdminController

负责处理管理员相关的审核与管理请求。提供用户认证审核、强制删除帖子、禁言或解禁用户、获取待审核用户列表、获取全部用户列表以及删除评论等操作的接口。

##### ▪ 审核注册信息

```
1 @PostMapping("/auth/review")
2 @Operation(summary = "审核注册信息", description = "审核用户注册信息，通过
或拒绝")
3 public ApiResponse<Void> reviewAuth(@Valid @RequestBody
AuthReviewDTO reviewDTO) {
4     Long adminId = CurrentUserContext.getUserId();
5     adminService.reviewAuth(adminId, reviewDTO);
6     return ApiResponse.success("审核成功");
7 }
```

##### ▪ 强制删除帖子

```
1 @DeleteMapping("/posts/{id}")
2 @Operation(summary = "强制删除帖子", description = "管理员强制删除帖子（逻辑删除）")
3 public ApiResponse<Void> forceDeletePost(
4     @Parameter(description = "帖子ID", example = "1", required =
true)
5     @PathVariable Long id) {
6     Long adminId = CurrentUserContext.getUserId();
7     adminService.forceDeletePost(adminId, id);
8     return ApiResponse.success("删除成功");
9 }
```

## ■ 禁言用户

```
1 @PostMapping("/users/mute")
2 @Operation(summary = "禁言用户", description = "管理员禁言或解除禁言用户")
3 public ApiResponse<Void> muteUser(@Valid @RequestBody UserMuteDTO
muteDTO) {
4     Long adminId = CurrentUserContext.getUserId();
5     adminService.muteUser(adminId, muteDTO);
6     return ApiResponse.success("操作成功");
7 }
```

## ■ 获取待审核用户列表

```
1 @GetMapping("/users/pending")
2 @Operation(summary = "获取待审核用户列表", description = "获取所有待审核
的用户列表，支持分页")
3 @Parameter(name = "page", description = "页码，从1开始", example =
"1")
4 @Parameter(name = "size", description = "每页数量", example = "10")
5 public ApiResponse<List<UserInfoDTO>> getPendingUsers(
6     @RequestParam(defaultValue = "1") Integer page,
7     @RequestParam(defaultValue = "10") Integer size) {
8     Long adminId = CurrentUserContext.getUserId();
9     return
10    ApiResponse.success(adminService.getPendingUsers(adminId, page,
size));
11 }
```

## ■ 获取全部用户列表

```
1 @GetMapping("/users")
2 @Operation(summary = "获取全部用户列表", description = "获取所有用户列
表, 支持分页")
3 @Parameter(name = "page", description = "页码, 从1开始", example =
"1")
4 @Parameter(name = "size", description = "每页数量", example = "10")
5 public ApiResponse<List<UserInfoDTO>> getAllUsers(
6     @RequestParam(defaultValue = "1") Integer page,
7     @RequestParam(defaultValue = "10") Integer size) {
8     Long adminId = CurrentUserContext.getUserId();
9     return ApiResponse.success(adminService.getAllUsers(adminId,
page, size));
10 }
```

## ■ 删除评论

```
1 @DeleteMapping("/comments/{id}")
2 @Operation(summary = "删除评论", description = "管理员删除用户评论（逻辑删
除）")
3 public ApiResponse<Void> deleteComment(
4     @Parameter(description = "评论ID", example = "1", required =
true)
5     @PathVariable Long id) {
6     Long adminId = CurrentUserContext.getUserId();
7     adminService.deleteComment(adminId, id);
8     return ApiResponse.success("删除成功");
9 }
```

### o AuthController

负责处理用户注册、登录以及验证码发送等认证相关请求。提供用户注册接口，支持手机号注册并结合验证码校验机制；提供用户登录接口，登录成功后返回 JWT Token，用于后续接口的身份认证；同时提供验证码发送接口，将验证码存储至 Redis 并设置有效期。

## ■ 用户注册

```
1 @PostMapping("/register")
2 @Operation(summary = "用户注册", description = "用户注册接口, 包含手机验证
码校验逻辑")
3 public ApiResponse<Long> register(@Valid @RequestBody RegisterDTO
registerDTO) {
4     Long userId = authService.register(registerDTO);
5     return ApiResponse.success("注册成功, 等待审核", userId);
6 }
```

## ■ 用户登录

```
1 @PostMapping("/login")
2 @Operation(summary = "用户登录", description = "用户登录接口, 返回JWT Token")
3 public ApiResponse<LoginResponseDTO> login(@Valid @RequestBody
4 LoginDTO loginDTO) {
5     LoginResponseDTO response = authService.login(loginDTO);
6     return ApiResponse.success(response);
}
```

#### ■ 发送验证码

```
1 @PostMapping("/send-verify-code")
2 @Operation(summary = "发送验证码", description = "发送手机验证码到Redis, 有效期5分钟")
3 public ApiResponse<Void> sendVerifyCode(@RequestParam String phone)
{
4     authService.sendVerifyCode(phone);
5     return ApiResponse.success("验证码已发送");
6 }
```

#### o CommentController

负责评论相关请求，提供评论发布、评论点赞与取消点赞以及评论删除等操作的接口。支持用户对帖子进行评论或回复他人评论，同时采用逻辑删除方式管理评论数据，并限制仅评论发布者本人可执行删除操作。

#### ■ 发布评论

```
1 @PostMapping
2 @Operation(summary = "发布评论", description = "对帖子发布评论或回复评论")
3 public ApiResponse<Long> createComment(@Valid @RequestBody
CommentCreateDTO createDTO) {
4     Long userId = CurrentUserContext.getUserId();
5     Long commentId = commentService.createComment(userId,
createDTO);
6     return ApiResponse.success("评论成功", commentId);
7 }
```

#### ■ 点赞/取消点赞评论

```
1 @PostMapping("/{id}/like")
2 @Operation(summary = "点赞/取消点赞评论", description = "点赞或取消点赞评论")
3 public ApiResponse<Boolean> toggleLike(
4     @Parameter(description = "评论ID", example = "1", required =
true)
5     @PathVariable Long id) {
6     Long userId = CurrentUserContext.getUserId();
7     boolean liked = commentService.toggleLike(id, userId);
8     return ApiResponse.success(liked);
9 }
```

## ■ 删除评论

```
1 @DeleteMapping("/{id}")
2 @Operation(summary = "删除评论", description = "逻辑删除评论, 只有评论者本人可以删除")
3 public ApiResponse<Void> deleteComment(
4     @Parameter(description = "评论ID", example = "1", required = true)
5     @PathVariable Long id) {
6     Long userId = CurrentUserContext.getUserId();
7     commentService.deleteComment(id, userId);
8     return ApiResponse.success("删除成功");
9 }
```

### ◦ PostController

负责帖子相关请求，提供帖子发布、删除、列表查询、详情查看、搜索以及点赞与取消点赞等操作的接口。支持用户发布和管理个人帖子，通过分页与分类筛选获取帖子列表，并提供基于关键词的模糊搜索能力；同时结合缓存机制返回热门帖子，提高访问效率。在帖子详情展示中整合评论信息，并通过权限控制保证仅作者本人可删除帖子。

## ■ 发布帖子

```
1 @PostMapping
2 @Operation(summary = "发布帖子", description = "发布新帖子, 标题限制4-20字, 分类必须合法")
3 @SecurityRequirement(name = "Bearer Authentication")
4 public ApiResponse<Long> createPost(@Valid @RequestBody
5 PostCreateDTO createDTO) {
6     Long userId = CurrentUserContext.getUserId();
7     Long postId = postService.createPost(userId, createDTO);
8     return ApiResponse.success("发布成功", postId);
9 }
```

## ■ 删除帖子

```
1 @DeleteMapping("/{id}")
2 @Operation(summary = "删除帖子", description = "逻辑删除帖子, 只有作者本人可以删除")
3 @SecurityRequirement(name = "Bearer Authentication")
4 public ApiResponse<Void> deletePost(
5     @Parameter(description = "帖子ID", example = "1", required = true)
6     @PathVariable Long id) {
7     Long userId = CurrentUserContext.getUserId();
8     postService.deletePost(userId, id);
9     return ApiResponse.success("删除成功");
10 }
```

## ■ 获取帖子列表

```
1 @GetMapping
2 @Operation(summary = "获取帖子列表", description = "获取帖子列表, 支持分
3 页和按分类筛选")
4 public ApiResponse<List<PostListDTO>> getPostList(
5     @Parameter(description = "分类ID", example = "1")
6     @RequestParam(required = false) Integer categoryId,
7     @Parameter(description = "页码, 从1开始", example = "1")
8     @RequestParam(defaultValue = "1") Integer page,
9     @Parameter(description = "每页数量", example = "10")
10    @RequestParam(defaultValue = "10") Integer size) {
11
12     return ApiResponse.success(postService.getPostList(categoryId,
13 page, size));
14 }
```

## ■ 搜索帖子

```
1 @GetMapping("/search")
2 @Operation(summary = "搜索帖子", description = "支持模糊搜索 (SQL LIKE
实现), 可预留ES接口")
3 public ApiResponse<List<PostListDTO>> searchPosts(
4     @Parameter(description = "搜索关键词", example = "自行车",
required = true)
5     @RequestParam String keyword,
6     @Parameter(description = "分类ID", example = "1")
7     @RequestParam(required = false) Integer categoryId,
8     @Parameter(description = "页码, 从1开始", example = "1")
9     @RequestParam(defaultValue = "1") Integer page,
10    @Parameter(description = "每页数量", example = "10")
11    @RequestParam(defaultValue = "10") Integer size) {
12
13     return ApiResponse.success(postService.searchPosts(keyword,
categoryId, page, size));
14 }
```

## ■ 获取热搜帖子

```
1 @GetMapping("/hot")
2 @Operation(summary = "获取热搜帖子", description = "返回热度前10的帖子, 使
用Redis缓存")
3 public ApiResponse<List<PostListDTO>> getHotPosts() {
4
5     return ApiResponse.success(postService.getHotPosts());
6 }
```

## ■ 获取帖子详情

```
1 @GetMapping("/{id}")
2 @Operation(summary = "获取帖子详情", description = "获取帖子详情，包含评论
列表")
3 public ApiResponse<PostDetailDTO> getPostDetail(
4     @Parameter(description = "帖子ID", example = "1", required =
true)
5     @PathVariable Long id) {
6     Long currentUserId = CurrentUserContext.getUserId();
7     return ApiResponse.success(postService.getPostDetail(id,
currentUser));
8 }
```

#### ■ 点赞/取消点赞帖子

```
1 @PostMapping("/{id}/like")
2 @Operation(summary = "点赞/取消点赞帖子", description = "点赞或取消点赞帖
子")
3 @SecurityRequirement(name = "Bearer Authentication")
4 public ApiResponse<Boolean> toggleLike(
5     @Parameter(description = "帖子ID", example = "1", required =
true)
6     @PathVariable Long id) {
7     Long userId = CurrentUserContext.getUserId();
8     boolean liked = postService.toggleLike(id, userId);
9     return ApiResponse.success(liked);
10 }
```

#### ○ UserController

负责用户个人信息相关请求，提供用户信息查询与维护操作的接口。支持当前登录用户查看和修改个人资料，并提供查看本人已发布帖子列表的接口，通过分页方式返回数据；结合统一的身份认证机制，确保所有用户相关操作仅在合法登录状态下进行。

#### ■ 获取个人信息

```
1 @GetMapping("/me")
2 @Operation(summary = "获取个人信息", description = "获取当前登录用户的个人
信息")
3 public ApiResponse<UserInfoDTO> getMyInfo() {
4     Long userId = CurrentUserContext.getUserId();
5     return ApiResponse.success(userInfoService.getUserInfo(userId));
6 }
```

#### ■ 修改个人信息

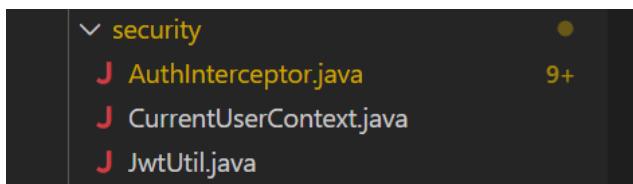
```
1 @PutMapping("/me")
2 @Operation(summary = "修改个人信息", description = "修改当前登录用户的个人信息")
3 public ApiResponse<Void> updateMyInfo(@valid @RequestBody
4 UserUpdateDTO updateDTO) {
5     Long userId = CurrentUserContext.getUserId();
6     userService.updateUserInfo(userId, updateDTO);
7     return ApiResponse.success("修改成功");
}
```

### ■ 查看我的发帖

```
1 @GetMapping("/me/posts")
2 @Operation(summary = "查看我的发帖", description = "获取当前用户发布的帖子列表, 支持分页")
3 @Parameter(name = "page", description = "页码, 从1开始", example = "1")
4 @Parameter(name = "size", description = "每页数量", example = "10")
5 public ApiResponse<List<PostListDTO>> getMyPosts(
6     @RequestParam(defaultValue = "1") Integer page,
7     @RequestParam(defaultValue = "10") Integer size) {
8     Long userId = CurrentUserContext.getUserId();
9     return ApiResponse.success(userService.getUserPosts(userId,
10 page, size));
}
```

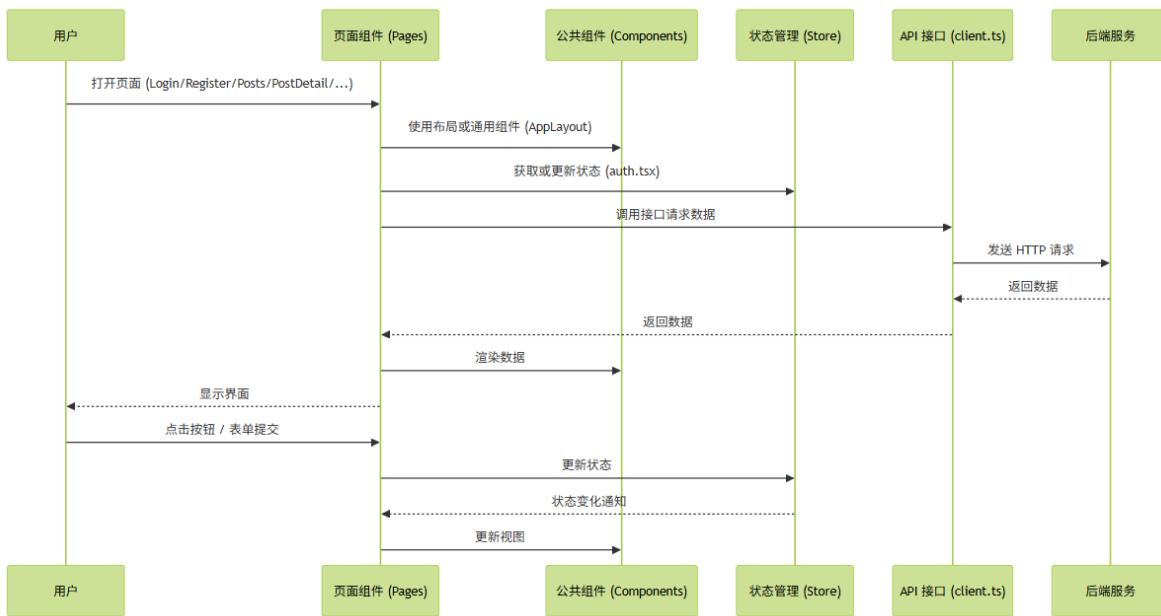
## 5. 安全层 (security)

负责用户身份认证和权限控制，包括 JWT 令牌解析、角色权限验证和接口访问控制，保障平台数据安全。



## 3.2 客户端

### 前端业务流程图



## 目录结构

```

1 src/
2   └── api/          # API接口层
3     |   └── client.ts      # HTTP客户端
4     |   └── types.ts      # 类型定义
5   └── assets/        # 静态资源
6   └── components/    # 可复用组件
7     |   └── AppLayout.tsx # 布局组件
8   └── pages/         # 页面组件 (7个业务页面)
9   └── store/         # 状态管理
10  |   └── auth.tsx      # 认证状态
11  └── mock/          # Mock数据

```

## 公用组件 (components)

- **AppLayout.tsx:** 统一页面布局，包括顶部导航栏、侧边菜单、主体内容区域。

## 页面 (pages)

- **学生端**

主要包括注册、登录、发帖、浏览帖子、评论点赞和个人信息管理。

页面	功能描述
<b>Login.tsx</b>	用户登录页面
<b>Register.tsx</b>	用户注册页面
<b>Posts.tsx</b>	帖子列表页, 浏览和搜索帖子
<b>PostDetail.tsx</b>	帖子详情页, 查看帖子和评论, 发表评论/点赞
<b>CreatePost.tsx</b>	发布新帖
<b>MyPosts.tsx</b>	查看和管理自己的帖子 (删除帖子)
<b>Profile.tsx</b>	查看和修改个人信息

- **管理员端**

主要包括用户审核和管理操作。

页面	功能描述
<b>AdminPending.tsx</b>	待审核用户列表, 审核注册用户、禁言用户等管理操作

## 4. 界面设计 (用户说明)

### 游客主界面 (默认)

The screenshot shows the homepage of the HNU Campus Exchange Platform. At the top, there is a blue header bar with the text "HNU 校园交流平台" and "首页". On the right side of the header are "登录" (Login) and "注册" (Register) buttons. Below the header, there is a search bar with a dropdown menu for "选择分类" (Select Category) and a search input field with a magnifying glass icon. To the right of the search bar is a "热搜榜" (Hot Search List) sidebar containing ten items numbered 1 to 10, each with a small colored circle and text. The main content area displays several posts in a grid format. Each post includes a title, author, category, and two numerical values representing views and likes.

帖子标题	作者	分类	发布时间	浏览量	点赞数
离散数学重点	user07	考试信息	2025/12/28 04:25:42	239	63
离散数学期末求重点：命题逻辑、集合关系、图论最短路与生成树、逆推与组合计数。					
求推荐组件库	user10	打听求助	2025/12/27 15:37:42	283	38
React项目做后台管理，纠结Ant Design和MUI，尤其关心React 19兼容性与表格体验。					
求助NS2安装	user18	打听求助	2025/12/26 10:20:42	226	31
Ubuntu装NS2总遇到依赖与编译报错，nam也打不开，求踩坑经验与检查清单。					
求推荐打印店	user07	打听求助	2025/12/26 01:19:42	449	11
需要打印装订60页左右报告，最好当天可取，价格透明不乱加项，求校内外推荐。					
计网复习资料	user28	考试信息	2025/12/25 10:00:42	341	77
计算机网络求资料：TCP三次握手四次挥手、拥塞控制、DNS/HTTP、子网划分与路由协议。					

游客只能浏览, 不能使用其他功能。

### 4.1 学生端

#### 注册

## 用户注册

\* 手机号

\* 验证码  
 获取验证码

\* 昵称

\* 密码  
 

学号

校园卡照片 URL

[提交注册](#)

[已有账号? 去登录](#)

## 登录

## 用户登录

\* 手机号

\* 验证码  
 获取验证码

\* 密码  
 

[登 录](#)

[还没有账号? 去注册](#)

- 未注册用户登录或密码错误会提示“手机号或密码不正确”：

 登录失败

手机号或密码错误

- 验证码错误会提示“验证码错误或已过期”：

 登录失败

验证码错误或已过期

- 未通过审核用户登录会提示“当前账号未通过申请”：

 无法登录

当前帐号未通过申请, 请稍等。

## 获取验证码

The screenshot shows the interface of Another Redis Desktop Manager. On the left, there's a tree view of keys under '127.0.0.1@6379'. Under the 'verify\_code' key, three sub-keys are listed: 'verify\_code:18666959731', 'verify\_code:18800000001', and 'verify\_code:18817153931'. On the right, a detailed view of the 'verify\_code:18666959731' key is shown, with its value being '18666959731' and TTL set to 291.

验证码的有效期为5分钟。

## 主页面

The screenshot shows the homepage of the HNU Campus Exchange Platform. It features a top navigation bar with links for 首页, 发帖, 我的帖子, and 个人中心. A user profile is shown on the right with the message '你好, user01' and a '退出登录' button. The main content area displays several posts:

- 离散数学重点** (浏览 252 点赞 63)  
作者: user07 分类: 考试信息 发布时间: 2025/12/28 04:25:42  
离散数学期末求重点: 命题逻辑、集合关系、图论最短路与生成树、递推与组合计数。
- 求推荐组件库** (浏览 283 点赞 38)  
作者: user10 分类: 打听求助 发布时间: 2025/12/27 15:37:42  
React项目做后台管理, 纠结Ant Design和MUI, 尤其关心React 19兼容性与表格体验。
- 求助NS2安装** (浏览 226 点赞 31)  
作者: user18 分类: 打听求助 发布时间: 2025/12/26 10:20:42  
Ubuntu装NS2总遇到依赖与编译报错, nam也打不开, 求踩坑经验与检查清单。
- 求推荐打印店** (浏览 453 点赞 11)  
作者: user07 分类: 打听求助 发布时间: 2025/12/26 01:19:42  
需要打印装订60页左右报告, 最好当天可取, 价格透明不乱加项, 求校内外推荐。
- 计网复习资料** (浏览 341 点赞 77)  
作者: user28 分类: 考试信息 发布时间: 2025/12/25 10:00:42  
计算机网络求资料: TCP三次握手四次挥手、拥塞控制、DNS/HTTP、子网划分与路由协议。

A sidebar on the right is titled '热搜榜' and lists ten trending topics from 1 to 10.

## 查看个人信息

The screenshot shows the '个人信息' (Personal Information) page. At the top, it has the same navigation bar as the homepage. The main content area is titled '个人信息' and contains a table with the following data:

手机号	18800000001	昵称	user01
学号	2021001001	认证状态	approved
角色	STUDENT	是否禁言	否

## 修改个人信息

### 修改密码

\* 原密码  
请输入原密码

\* 新密码  
请输入新密码

\* 确认新密码  
请再次输入新密码

[修改密码](#)

支持昵称、密码修改。

- 若原密码不正确，则提示“修改失败”，拒绝修改密码：



## 查看帖子详情

HNU 校园交流平台 首页 发帖 我的帖子 个人中心 你好, user01 退出登录

### 求推荐打印店

作者: user07 分类: 打听求助 发布时间: 2025/12/26 01:19:42 联系方式: 邮箱: user07@example.com

需要打印装订60页左右报告，最好当天可取，价格透明不乱加项，求校内外推荐。

点赞 11 浏览 455

#### 发表评论

写下你的评论...

提交评论

#### 评论列表

user07 2025/12/30 09:16:41  
南门那家打印挺快的。  
点赞 0 回复

## 点赞帖子

### 求推荐打印店

作者: user07 分类: 打听求助 发布时间: 2025/12/26 01:19:42 联系方式: 邮箱: user07@example.com

需要打印装订60页左右报告，最好当天可取，价格透明不乱加项，求校内外推荐。

已点赞 12 浏览 457

不允许重复点赞，可以取消点赞。

## 点赞评论

user07 2025/12/30 09:16:41

南门那家打印挺快的。

已赞 1 回复

不允许重复点赞，可以取消点赞。

## 搜索帖子

选择分类

SQL



### 数据库考试范围

浏览 271 点赞 24

作者：user22 分类：考试信息 发布时间：2025/12/23 20:22:42

求确认数据库系统考试范围与题型：事务、日志恢复、调度可串行化、索引与SQL优化会占多少分？

### 找学习搭子

浏览 252 点赞 47

作者：user08 分类：恋爱交友 发布时间：2025/12/22 12:09:42

准备寒假补齐JUC和MySQL调优，想找一个愿意每周对齐目标的搭子，互相督促。

### MySQL大题方向

浏览 408 点赞 17

作者：user18 分类：考试信息 发布时间：2025/11/29 08:57:42

MySQL期末大题一般考什么：事务隔离级别、锁、日志与恢复、执行计划分析，求往年经验。

<

1

>

10 / page ▾

支持按标题或内容全文检索。

## 热搜榜

### 热搜榜

1 线代复习清单

2 元旦看演出吗

3 计网复习资料

4 社团招新趣事

5 图书馆占座吐槽

6 出二手平板

7 求助依赖冲突

8 一起刷题打卡

9 求推荐打印店

10 寝室楼电梯故障

## 发布帖子

## 发布新帖

\* 标题

数据库往年试卷

\* 分类

打听求助

\* 内容

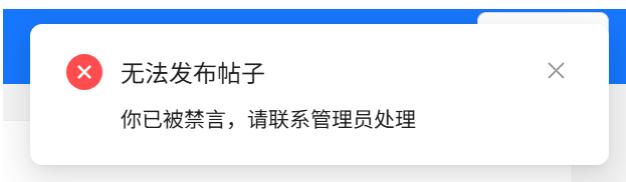
rt 求数据库往年期末卷子

联系方式

qq: 1660718567

发布

- 被禁言用户不能发布帖子：



## 查看我的帖子

## 我的帖子

数据库往年试卷

浏览 0 点赞 0 删除

分类：打听求助 发布时间：2025/12/30 12:44:56

rt 求数据库往年期末卷子

打听求助

## 删除帖子

## 我的帖子

数据库往年试卷

分类：打听求助 发布时间：2025/12/30 12:44:56

rt 求数据库往年期末卷子

打听求助

① 确认删除该帖子？

取消

删除

浏览 0 点赞 0 删除

只能删除自己发布的帖子。

## 评论帖子

user01 2025/12/30 12:32:50

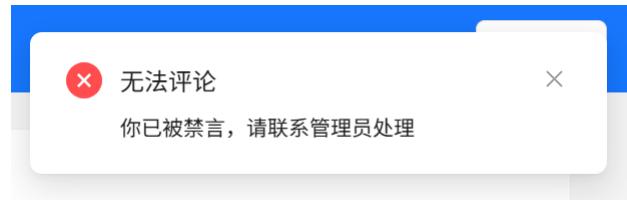
三区十那家不错

点赞 0

回复

删除

- 被禁言用户不能发布评论：



## 回复评论

user07 2025/12/30 09:16:41  
南门那家打印挺快的。

已赞 1 回复

user01 回复 user07 2025/12/30 12:35:44  
我每次去都人好多

点赞 0 回复 删除

- 被禁言用户不能回复评论：



## 删除评论

user01 2025/12/30 12:32:50 已删除  
该评论用户已自行删除  
删除

user07 2025/12/30 09:16:41  
南门那家打印挺快的。  
已赞 1 回复

user01 回复 user07 2025/12/30 12:35:44 已删除  
该评论用户已自行删除  
删除

只能删除自己发布的评论。

## 4.2 管理员端

登录

## 用户登录

\* 手机号

\* 验证码  
 获取验证码

\* 密码

**登 录**

[还没有账号？去注册](#)

## 主页面

你好, laan [退出登录](#)

选择分类  搜索帖子标题或内容

**离散数学重点** 浏览 249 点赞 63  
作者: user07 分类: 考试信息 发布时间: 2025/12/28 04:25:42  
离散数学期末求重点: 命题逻辑、集合关系、图论最短路与生成树、递推与组合计数。

**求推荐组件库** 浏览 283 点赞 38  
作者: user10 分类: 打听求助 发布时间: 2025/12/27 15:37:42  
React项目做后台管理, 纠结Ant Design和MUI, 尤其关心React 19兼容性与表格体验。

**求助NS2安装** 浏览 226 点赞 31  
作者: user18 分类: 打听求助 发布时间: 2025/12/26 10:20:42  
Ubuntu装NS2总遇到依赖与编译报错, nam也打不开, 求踩坑经验与检查清单。

**求推荐打印店** 浏览 453 点赞 11  
作者: user07 分类: 打听求助 发布时间: 2025/12/26 01:19:42  
需要打印装订60页左右报告, 最好当天可取, 价格透明不乱加项, 求校内外推荐。

**计网复习资料** 浏览 341 点赞 77  
作者: user28 分类: 考试信息 发布时间: 2025/12/25 10:00:42  
计算机网络求资料: TCP三次握手四次挥手、拥塞控制、DNS/HTTP、子网划分与路由协议。

**热搜榜**

- 1 线代复习清单
- 2 元旦看演出吗
- 3 计网复习资料
- 4 社团招新趣事
- 5 图书馆占座吐槽
- 6 出二手平板
- 7 求助依赖冲突
- 8 一起刷题打卡
- 9 求推荐打印店
- 10 寝室楼电梯故障

## 查看个人信息

### 个人信息

手机号	18666959731	昵称	laan
学号	未填写	认证状态	approved
角色	ADMIN	是否禁言	否

## 修改个人信息

### 编辑资料

昵称

学号

校园卡照片 URL

**保 存**

**修改密码**

* 原密码	请输入原密码	
* 新密码	请输入新密码	
* 确认新密码	请再次输入新密码	
<b>修改密码</b>		

支持昵称、密码修改。

## 审核用户

HNU 校园交流平台 首页 发帖 我的帖子 个人中心 管理后台 你好, laan 退出登录

**用户管理** 待审核用户

ID	手机号	昵称	学号	校园卡	注册时间	操作
1	18800000001	user01	2021001001	未提交	2025/12/30 08:44:25	<b>通过</b> 拒绝
2	18800000002	user02	2021001002	未提交	2025/12/30 08:44:25	<b>通过</b> 拒绝
3	18800000003	user03	2021001003	未提交	2025/12/30 08:44:25	<b>通过</b> 拒绝

可以“通过”或“拒绝”用户的申请。

## 禁言用户

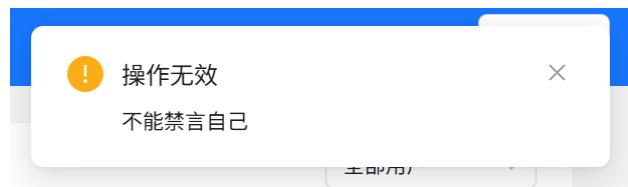
HNU 校园交流平台 首页 发帖 我的帖子 个人中心 管理后台 你好, laan 退出登录

**用户管理** 全部用户

ID	手机号	昵称	学号	校园卡	注册时间	操作
1	18800000001	user01	2021001001	未提交	2025/12/30 08:44:25	<b>禁言</b>

可以禁言除自己外的所有用户。

- 尝试禁言自己会提示“操作无效”：



## 恢复用户发言权限

HNU 校园交流平台 首页 发帖 我的帖子 个人中心 管理后台 你好, laan 退出登录

**用户管理** 全部用户

ID	手机号	昵称	学号	校园卡	注册时间	操作
1	18800000001	user01	2021001001	未提交	2025/12/30 08:44:25	<b>解禁</b>

## 删除帖子

## 离散数学重点

作者: user07 分类: 考试信息 发布时间: 2025/12/28 04:25:42 联系方式: 邮箱: user07@example.com

离散数学期末求重点: 命题逻辑、集合关系、图论最短路与生成树、递推与组合计数。

点赞 63

浏览 260

删除帖子

可以删除所有用户的帖子。

### 删除评论

user21 2025/12/30 09:16:41

校内打印一般偏贵。

点赞 0

回复

删除

可以删除所有用户的评论。

## 三、实验总结

通过本次“HNU 校园交流平台”综合实验，我们完整经历了从需求分析、数据库设计到系统实现与测试的全过程，对数据库系统课程中所学的理论知识有了更加直观和深入的理解。

在前期需求分析阶段，我们结合真实校园场景，明确了学生端与管理员端的功能边界，为后续的概念结构设计和逻辑建模奠定了基础。在数据库设计过程中，通过对用户、帖子、评论及点赞等核心实体的抽象建模，并对各关系模式进行函数依赖分析和范式验证，进一步加深了对第三范式和 BCNF 的理解，体会到良好数据库设计在减少数据冗余、保证数据一致性方面的重要作用。

在物理设计与实现阶段，我们选择基于 PostgreSQL 来实现数据库。这主要是因为其内置全文检索功能，通过 tsvector 和 GIN 索引可以实现高效的帖子搜索，优于传统的模糊查询。

在软件设计部分，通过 Spring Boot + React 的前后端分离架构，将数据库与实际 Web 应用相结合，使我们对数据库在真实业务系统中的作用有了更加全面的认识。这部分比较有意思的一个地方是通过 JWT 鉴权机制，实现了学生与管理员的双重权限体系。

总的来说，这次实验还是学到了很多。不仅巩固了数据库设计、SQL 编写、约束与索引等核心知识，还提升了我们从实际需求出发进行系统化设计的能力。通过理论与实践相结合，我们认识到数据库设计并非孤立环节，而是贯穿系统开发全过程的重要基础。