课 程 论 文

课程名称：	现代软件开发技术
学期：	2021-2022 学年第 2 学期
项目名称：	基于图片分享的社交平台
年级专业：	2019 级软件工程专业
学号：	201925220316
姓名：	刘润超
学号：	201925220319
姓名：	彭翊轩
授课教师：	肖磊
提交日期：	2022 年 05 月 23 日
论文成绩：	
项目成绩：	
总成绩：	

华南农业大学 数学与信息学院

华南农业大学数学与信息学院

《现代软件开发技术》 软件项目评分表

项目名称	基于图片分享的社交平台
团队成员	学号 201925220316，姓名 刘润超； 学号 201925220319，姓名 彭翊轩
开发技术	SpringBoot+Vue

规定功能	完成情况	选做及增加功能	完成情况
01 显示分享列表		01 限制词管理	
02 搜索分享		02 限制词检测	
03 点赞榜		03 关注用户	
04 最新发布		04 我的关注	
05 查看分享		05 我的粉丝	
06 登录/注册		06 用户设置	
07 点赞分享		07 搜索用户	
08 收藏分享		08 查看用户主页	
09 评论分享		09 热搜榜	
10 回复评论		10 密码加密	
11 我的分享			
12 发布分享			
13 修改分享			
14 分享状态			
15 我的收藏			
16 注册用户管理			

评语：

项目成绩	

目录

1. 问题描述与需求分析

1.1. 项目基本情况

SpringBoot+Vue 开发的基于图片分享的社交平台。平台有访客用户、注册用户、管理员三种角色。完成的功能有显示分享列表、搜索分享、点赞榜、最新发布、查看分享、登录/注册、点赞分享、收藏分享、评论分享、回复评论、我的分享、发布分享、修改分享、分享状态、我的收藏、注册用户管理、限制词管理、限制词检测、关注用户、我的关注、我的粉丝、用户设置、搜索用户、查看用户主页、热搜榜、密码加密。

1.2. 功能说明

1.2.1. 显示分享列表

以分页的形式显示分享列表。列表中的每个分享要展示分享的内容、图片，发布分享用户的用户名、头像，分享发布的时间，分享的点赞数、评论数和收藏数，以及当前用户是否已点赞收藏该分享。点击分享内容跳转到分享详情页面，点击分享图片展示大图，点击用户头像可跳转到用户主页。根据用户是否已点赞、收藏该分享，点赞按钮、收藏按钮以不同颜色显示。点击评论按钮可展示用户评论列表，再次点击可将用户评论列表折叠。点击评论中的查看回复按钮可查看该评论下的回复。

1.2.2. 搜索分享

在搜索框中输入内容，点击搜索按钮或按下回车键，以字符串模糊匹配的方式对分享的内容进行搜索，并将匹配的分享以分享列表的形式分页展示。

1.2.3. 点赞榜

按照点赞数降序展示分享列表。

1.2.4. 最新发布

按照发布时间最新的顺序展示分享列表。

1.2.5. 查看分享

点击分享列表中分享的内容进入分享详细页面。

1.2.6. 登录/注册

点击注册按钮进入注册页面，用户输入用户名、密码、确认密码，若用户名重复或确认密码与密码不匹配，则不允许注册，若合法，则点击注册按钮进行注册。点击登录按钮进入登录页面，输入用户名和密码进行登录，用户名存在且密码正确则登录成功。

1.2.7. 点赞分享

注册用户可点赞分享。分享的点赞按钮根据用户是否已点赞以不同的颜色展示。若未点赞，点击点赞按钮对分享点赞，若已点赞，点击点赞按钮对分享取消点赞。

1.2.8. 收藏分享

注册用户可收藏分享。分享的收藏按钮根据用户是否已收藏以不同的颜色展示。若未收藏，点击收藏按钮收藏分享，若已收藏，点击收藏按钮对分享取消收藏。

1.2.9. 评论分享

注册用户可评论分享。用户点击分享的评论按钮后，展示评论输入框，用户在输入框中输入评论内容，点击发表按钮发表评论。新发表的评论实时展示在评论列表中。每条评论需要展示出评论的内容，发布评论的用户的用户名、头像，评论发布时间以及回复数。

1.2.10. 回复评论

注册用户可回复评论。用户点击评论中的回复按钮后，展示回复输入框，用户在输入框中输入回复内容，点击提交按钮提交回复。实时更新评论的回复状态。每条回复需要展示出回复的内容，提交回复的用户的用户名、头像，回复提交时间。

注册用户可对回复进行回复。用户点击回复中的回复按钮进行回复。与对评论进行回复不同的是，对回复的回复在回复的内容前会显示“回复@xxx”，以明确所回复的用户。

1.2.11. 我的分享

以分页分享列表的形式展示当前用户发布的分享。包含分享内容、用户名、用户头像、发布时间信息。点击详情进入分享详情页面，点击编辑按钮展示分享编辑页面、点击删除按钮删除已发布的分享。

1.2.12. 发布分享

注册用户可发布分享。进入分享发布页面，可输入分享内容、设置分享是否公开、以及上传图片，已选择的图片将会展示出来，每个分享最多可有 6 个图片。点击发布按钮发布分享。

1.2.13. 修改分享

在分享修改页面中，用户可对分享的内容、分享是否公开、以及分享的图片进行编辑，点击提交按钮确认修改。

1.2.14. 分享状态

分享的状态分为“仅自己可见”和“公开”，在发布分享时可进行设置，在分享发布后，可在修改分享页面中进行修改。

1.2.15. 我的收藏

以分页分享列表的形式展示当前用户收藏的分享。包含分享内容、用户名、用户头

像、发布时间信息。点击详情进入分享详情页面，点击删除按钮可取消收藏。

1.2.16. 注册用户管理

管理员可以禁止或重新允许一个注册用户进行分享活动。处于禁止状态的注册用户的分享均不可见。

1.2.17. 限制词管理

管理员可对限制词进行管理，可添加或删除限制词。

1.2.18. 限制词检测

在用户发布分享或修改分享内容前，对分享的内容进行限制词检测，若分享内容中存在限制词，则不允许用户发布或修改分享内容。

1.2.19. 关注用户

注册用户可关注用户。进入其他用户的主页，根据用户是否已关注，关注按钮显示为“关注”或“取消关注”，点击按钮可关注用户或取消关注用户。

1.2.20. 我的关注

注册用户可查看我的关注列表，列表中展示当前用户已关注用户的用户名和头像，点击可进入该用户的个人主页，点击列表中的取消关注按钮可取消对该用户的关注。

1.2.21. 我的粉丝

注册用户可查看我的粉丝列表，列表中展示当前用户粉丝的用户名和头像，点击可进入该用户的个人主页，点击列表中的关注按钮可关注该用户。

1.2.22. 用户设置

注册用户可对个人信息和账号进行设置。用户可设置头像、修改生日、性别以及自我介绍，用户可修改用户名和密码。

1.2.23. 搜索用户

在搜索框中输入用户名，点击搜索按钮或按下回车键，以字符串模糊匹配的方式对用户进行搜索，并将匹配的用户以列表的形式分页展示。

1.2.24. 查看用户主页

进入用户主页，可查看用户头像、用户名、性别、生日和自我介绍，可以点击按钮关注或取消关注用户。

1.2.25. 热搜榜

点击分享搜索框，将展示当前的热搜榜，包含热搜关键词和被搜索次数，点击热搜榜的关键词，将按照关键词进行搜索。热搜榜每天定时刷新清零，重新累计。

1.2.26. 密码加密

用户注册时输入的密码在存入数据库前进行 MD5 加密，避免用户密码以明文的形式存入数据库。

1.3. 软件技术、开发工具和开发环境

1.3.1. 前端

软件技术：Vue3、Ant Design Vue、axios、VueRouter、VueCli
开发工具：VScode、ApiFox、git
开发环境：macOS、Edge

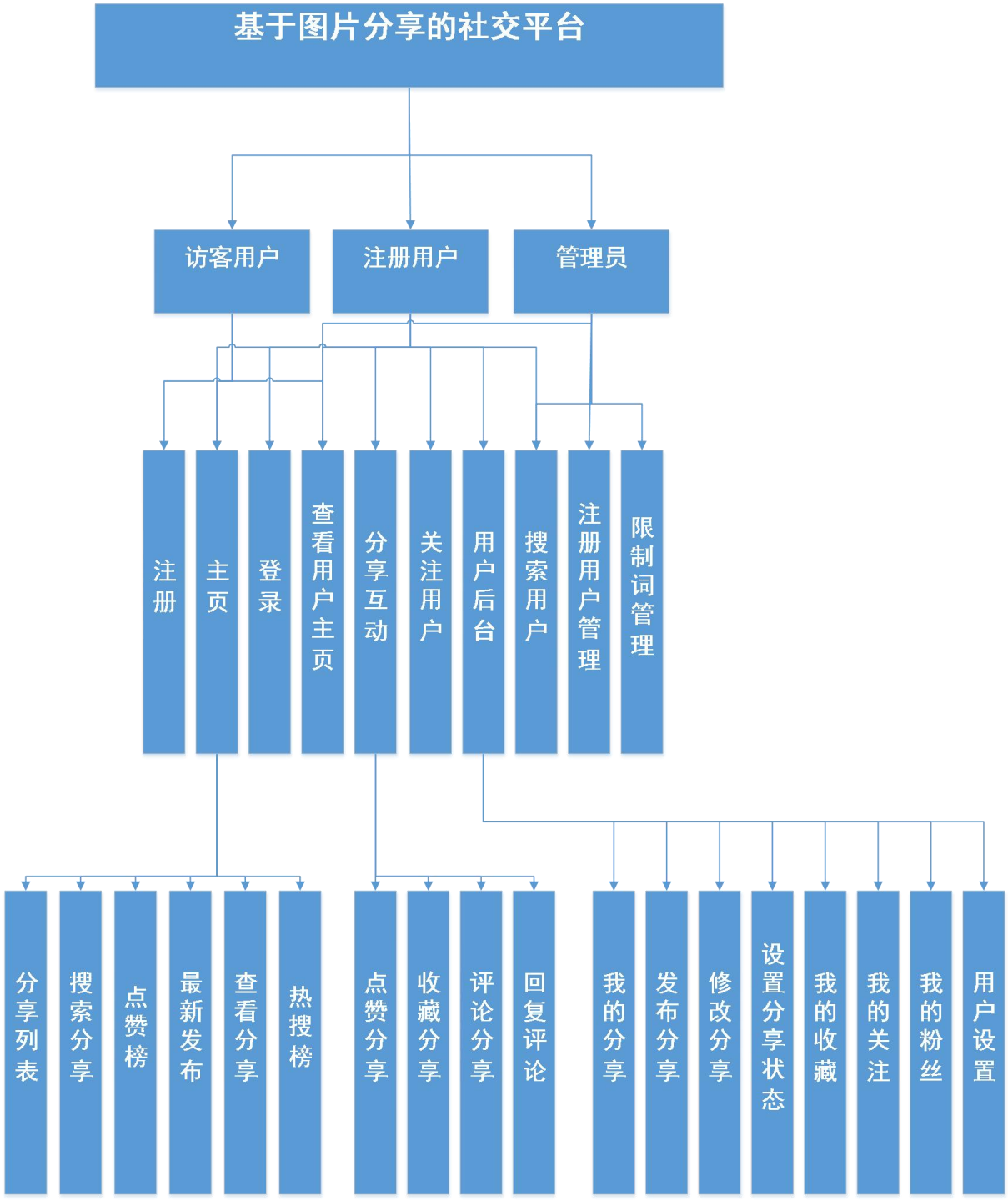
1.3.2. 后端

软件技术：SpringBoot、Spring、SpringMVC、mybatis、maven
开发工具：IntelliJ IDEA 2021.2、Navicat 15、Postman、git
开发环境：windows10、jdk1.8.0_321、MySQL 8.0.27、Redis 5.0.14

2. 系统设计

2.1. 总体设计方案

2.1.1. 总体功能结构图



2.1.2. 总体设计方案说明

系统用户分为三种角色：访客用户、注册用户和管理员。系统的功能按照这三个角色进行划分。访客用户具有注册、主页、查看其他用户主页的功能。注册用户具有主页、登录、查看其他用户主页、分享互动、关注用户、用户后台、搜索用户的功能。管理员具有查看其他用户主页、搜索用户、注册用户管理、限制词管理的功能。

功能细分：

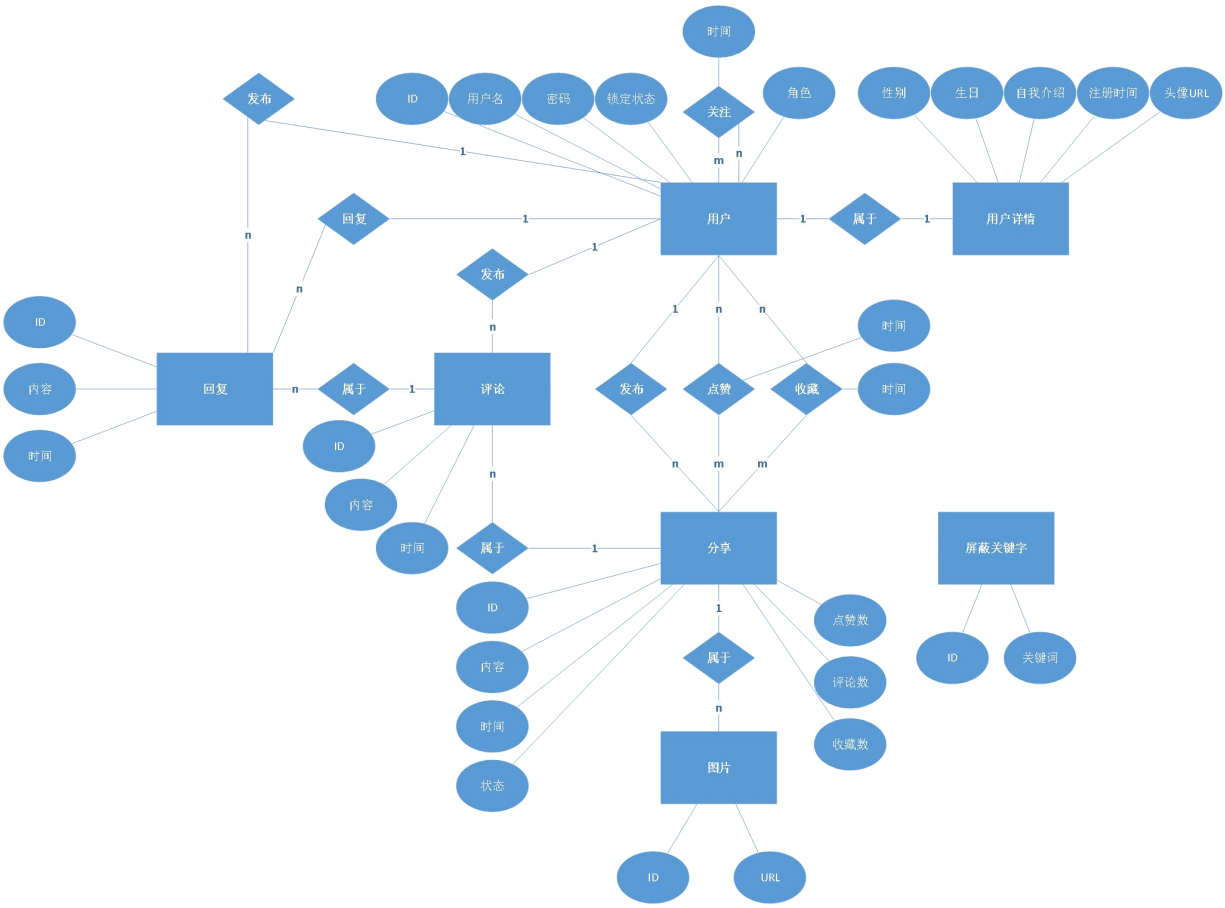
主页功能可细分为：分享列表、搜索分享、点赞榜、最新发布、查看分享、热搜榜。

分享互动功能可细分为：点赞分享、收藏分享、评论分享、回复评论。

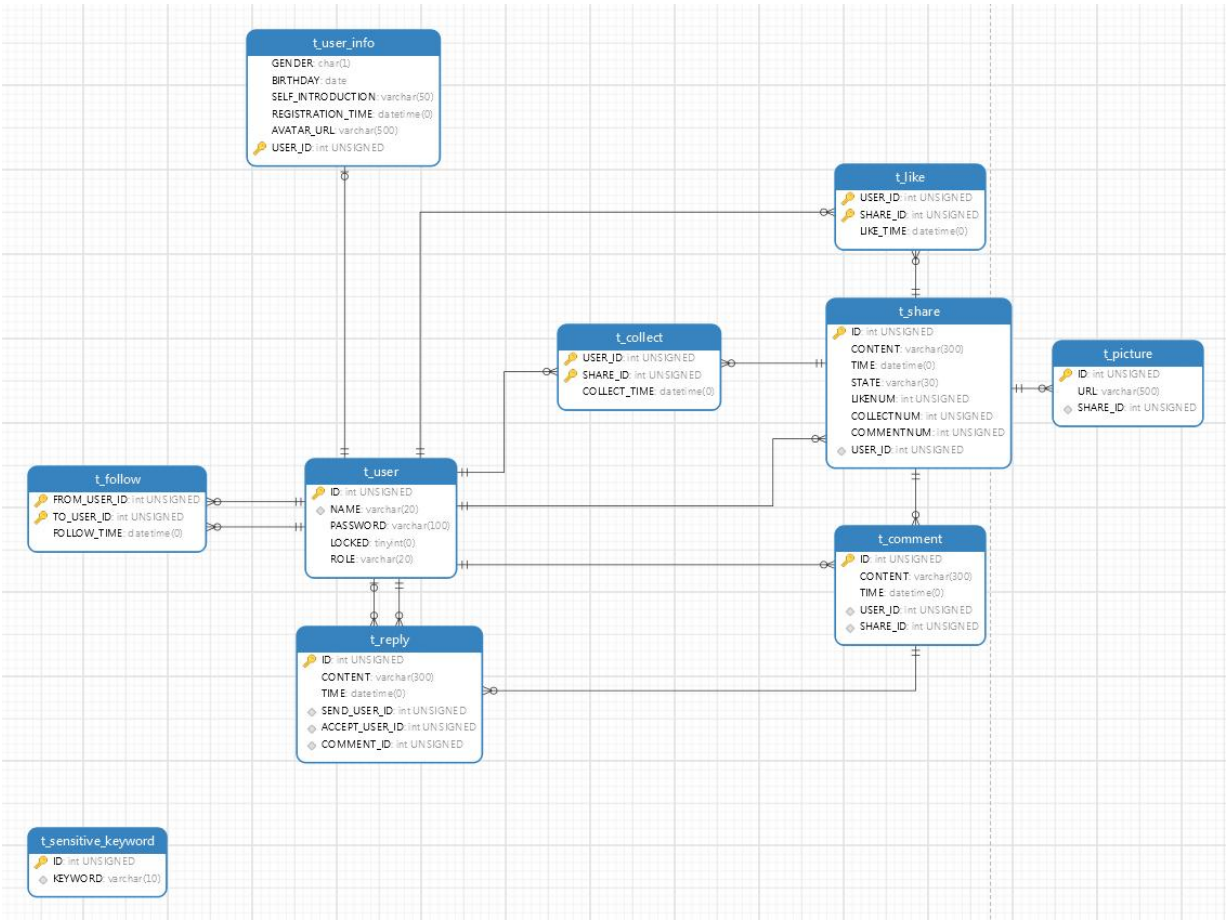
用户后台功能可细分为：我的分享、发布分享、修改分享、设置分享状态、我的收藏、我的关注、我的粉丝、用户设置。

2.2. 数据库设计

2.2.1. E-R 图



2.2.2. 关系模式图



2.2.3. 字段说明

2.2.3.1. T_USER 用户表

ID　主键，用户 ID
NAME　用户名
PASSWORD　密码
LOCKED　　用户是否处于锁定状态
ROLE　　用户角色

2.2.3.2. T_USER_INFO 用户信息表

GENDER　　性别
BIRTHDAY　生日
SELF_INTRODUCTION　　自我介绍
REGISTRATION_TIME　　注册时间
AVATAR_URL　　头像 URL
USER_ID　　主键，外键，所属的用户 ID

2.2.3.3. T_SHARE 分享表

ID 主键，分享 ID
CONTENT 分享的内容
TIME 分享发布时间
STATE 分享的状态
LIKENUM 点赞数
COLLECTNUM 收藏数
COMMENTNUM 评论数
USER_ID 外键，发布分享的用户 ID

2.2.3.4. T_COMMENT 评论表

ID 主键，评论 ID
CONTENT 评论的内容
TIME 评论发布时间
USER_ID外键，发布评论的用户 ID
SHARE_ID 外键，评论所属的分享 ID

2.2.3.5. T_REPLY 回复表

ID 主键，回复 ID
CONTENT 回复的内容
TIME 回复发布时间
SEND_USER_ID 外键，发布回复的用户 ID
ACCEPT_USER_ID 外键，接收回复的用户 ID
COMMENT_ID 外键，回复所属的评论 ID

2.2.3.6. T_SENSITIVE_KEYWORD 限制词表

ID 主键，限制词 ID
KEYWORD 限制词

2.2.3.7. T_PICTURE 图片表

ID 主键，图片 ID
URL 图片在文件系统中的路径
SHARE_ID 外键，图片所属的分享 ID

2.2.3.8. T_FOLLOW 关注表

FROM_USER_ID 主键，外键，发起关注的用户 ID
TO_USER_ID 主键，外键，被关注的用户 ID
FOLLOW_TIME 关注的时间

2.2.3.9. T_LIKE 点赞表

USER_ID　主键，外键，点赞用户 ID
SHARE_ID　主键，外键，获赞分享 ID
LIKE_TIME　点赞时间

2.2.3.10. T_COLLECT 收藏表

USER_ID　主键，外键，收藏用户 ID
SHARE_ID　主键，外键，被收藏分享 ID
COLLECT_TIME　收藏时间

2.3. 界面设计

整体界面风格为原版 ant-design 的 light 风格并没有使用其他主题，样式也只是局部性的位移调整。

前端采用单页面+路由的设计，除开点击用户主页会新增标签页外其他一切操作皆可在一页标签页中完成。

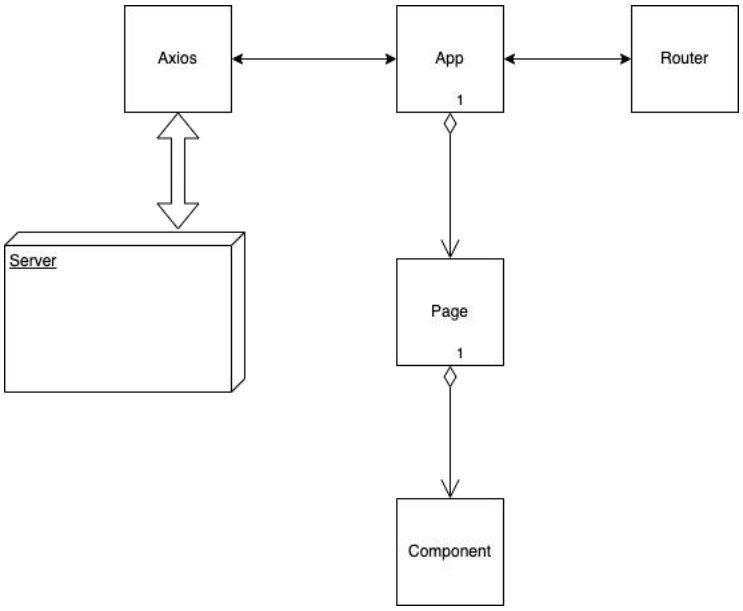
主页设计参考微博设计，整体分为上边的导航栏以及下边的主体部分，主体中又有占大块的分享窗格以及右侧的最受欢迎榜和最新榜两个悬浮卡片。可以通过分享窗格上面的标签进行切换分享排序，使用简单明了，点击其中的分享亦可跳至该分享的详情页。

至于用户自身后台以及管理员后台界面则是采用典型的页面布局，即上边导航栏、左侧菜单栏以及右大块的主体内容区域，这种布局可以很方便的进行页面和功能的扩展。当中的内容也为后台功能而做即是典型的填表以及弹出填表的对话框。

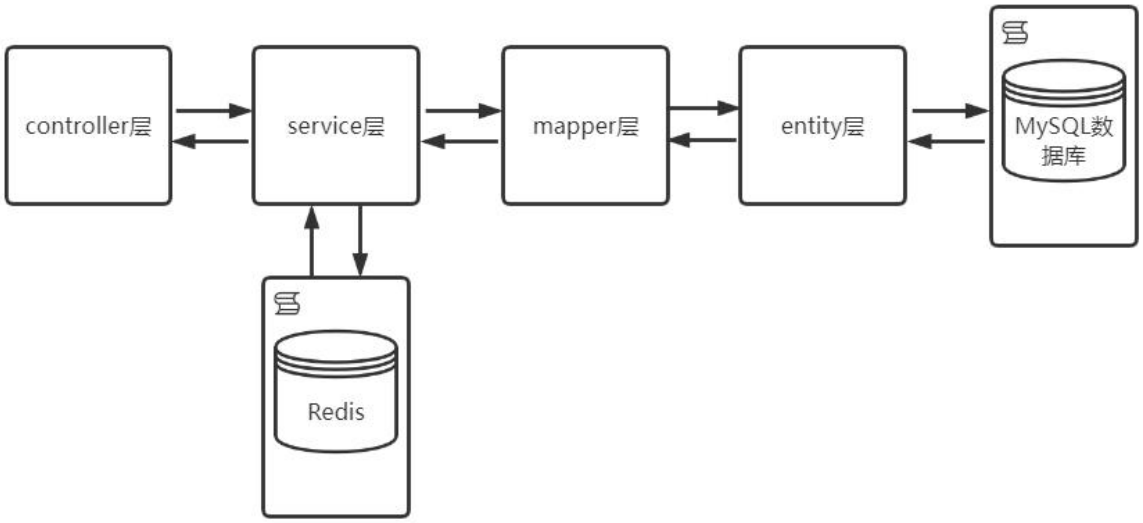
2.4. 软件系统结构设计

2.4.1. 软件系统层次结构

2.4.1.1. 前端



2.4.1.2. 后端



2.4.2. 各层次任务描述

2.4.2.1. 前端

2.4.2.1.1. App

根组件，前端应用的源头节点。

2.4.2.1.2. Page

页面组件，构成前端的一张张页面。

2.4.2.1.3. Component

组件，抽离出来的可复用的供页面和其它组件使用的子组件。

2.4.2.1.4. Router

路由器，安插在整个应用上用以分配路由与页面的插件。

2.4.2.1.5. Axios

发送 Ajax 请求的工具包。

2.4.2.2. 后端

2.4.2.2.1. controller 层

存放暴露给前端的后端接口，接收前端请求，调用 service 层的业务逻辑，并将处理后获得的数据通过 json 的形式响应给前端。

2.4.2.2.2. service 层

存放业务逻辑处理，并通过调用 mapper 层的方法与数据库交互。

2.4.2.2.3. mapper 层

对数据库进行数据持久化操作，为 service 层提供数据库增删查改的方法。

2.4.2.2.4. entity 层

存放实体类，实体类中的属性值与数据库中的字段对应。获取数据库中的数据时，将字段映射成为对象的属性。

3. 系统实现

3.1. 系统项目结构

3.1.1. 前端

项目结构说明：

public 用来存放外部资源如 favicon 和 html 文件。

assets 用来存放静态资源。

components 存放内聚性高的组件。

global 存储各种全局性的配置和模块，如全局样式和 dayjs 等实用性模块。

hooks 通过 vue3 特性用以存放解离自原本组件中的各种逻辑代码，并且可以复用于各路组件，相较于 vue2 大幅提高复用度。

pages 存放路由地址对应的各个页面。

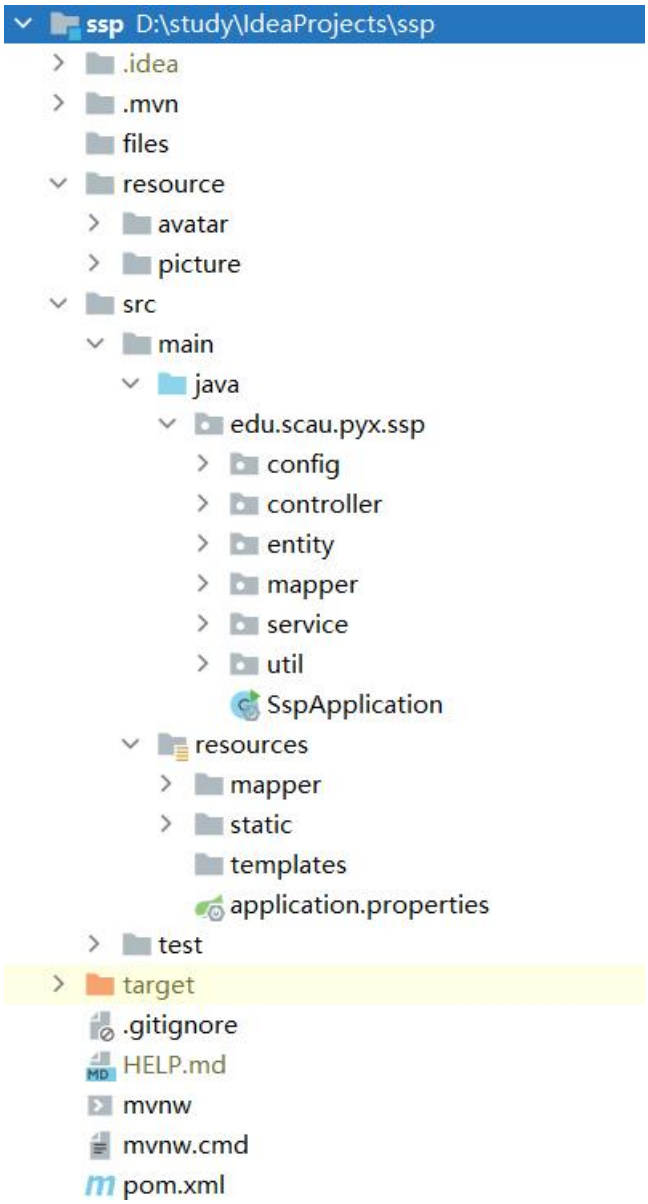
router 存放路由相关配置信息。

App.vue 是前端应用的根组件。

main.js 是为前端应用的启动入口。

其余文件为 vue 脚手架的各种配置文件。

3.1.2. 后端



项目结构说明：

files 文件夹是文件上传临时目录。由于在使用基于 springboot 的文件上传过程中，被上传的文件首先会被放入临时目录中，为避免出现临时目录不存在的情况，在 files 文件夹不存在时创建 files 文件夹，并指定 files 文件夹作为文件上传临时目录。

resource 文件夹用于存放用户头像和分享图片，其中的 avatar 文件夹存放用户头像，picture 文件夹存放分享图片。

src\main\java 文件夹下：

config 包存放配置类。

controller 包存放 controller 类。

entity 包存放 entity 类。

mapper 包存放 mapper 类。

service 包存放 service 类。

util 包存放工具类。

src\main\resources 文件夹下：

mapper 文件夹存放 mybatis 映射文件。

static 文件夹存放前端打包而成的静态文件。

3.2. 系统全局配置说明

3.2.1. 前端

router/index.js 存放所有的路由匹配组件信息，同时配有前置路由守卫，用以拦截未登录用户访问某些注册用户才被允许进入的页面。

global/axios.js 存放着配置过的 axios 模块，主要用以配置基址和 Cookie 配送。

global/config.js 存放一些常量如每一页应显示的分享数量。

global/global.less 存放全局性的样式。

vue.config.js 脚手架暴露出来的配置项，用来配置应用参数以及开发时候的跨域服务器配置。

package.json 跟 npm 打包信息相关，管控前端应用的一切依赖与依赖的部分配置。

3.2.2. 后端

MultipartConfig 配置类，对文件上传临时目录进行配置，指定文件上传临时目录为项目路径下的 files 文件夹，当目录不存在时创建该目录。

MvcConfig 配置类，配置允许跨域的路径为"/**"，允许跨域的源为"*"，允许的请求头为"*"，允许的请求方法为"*"，解决前后端不同源时的跨域问题。配置使用凭证为 true，解决跨域时浏览器携带 cookie 的问题。

RedisConfig 配置类，对 RedisTemplate 进行配置，配置后可注入 RedisTemplate 的实例。

application.properties 文件中配置项目端口、配置 MySQL 数据库信息、向 mybatis 指定实体类所在包和映射文件所在目录、指定上传文件大小的最大值、配置 Redis 数据库信息。

3.3. 层次说明

3.3.1. 前端

3.3.1.1. Page

3.3.1.1.1. 实现的页面

Admin
Home
Login
Person
Register
Share
User

3.3.1.1.2. 实现方式说明

以 Home 为例：

```
<template>
  <a-layout class="layout">
    <a-layout-header>
      <home-filled
        @click="$router.push('/allShares')"
        style="font-size: 32px"
      />
      <a-dropdown :trigger="['click']">
        <a-input-search
          v-model:value="searchText"
          placeholder="输入关键字"
          style="width: 45%"
          @search="search"
        />
        <template #overlay>
          <a-menu>
            <a-menu-item
              @click="onHotSearch(item.content)"
              v-for="(item, index) in hotSearch"
              :key="index"
            >
              <div class="hot-box">
                <span :class="{ 'hot-order': true, 'top-3': index < 3 }">{{
                  index + 1
                }}</span>
                <span class="hot-content">{{ item.content }}</span>
```

```
                        <span class="hot-times">{{ item.times }}次</span>
                      </div>
                    </a-menu-item>
                  </a-menu>
                </template>
              </a-dropdown>
              <UserBox />
            </a-layout-header>
            <a-layout-content>
              <div class="card-box">
                <Card style="margin-bottom:20px" title="最受欢迎" :isNewest="false" />
                <Card title="最新分享" :isNewest="true" />
              </div>
              <a-menu v-model:selectedKeys="selectedKeys" mode="horizontal">
                <a-menu-item key="all"> 全部 </a-menu-item>
                <a-menu-item key="new"> 最新 </a-menu-item>
                <a-menu-item key="like"> 点赞排行 </a-menu-item>
              </a-menu>
              <router-view></router-view>
            </a-layout-content>
          </a-layout>
        </template>
```

整个主页部分分为两大块，头部和主体，头部里有主页图标和搜索框以及一个登录组件，其中登录组件反映当前用户的登录状态，当是未登录时会显示登录按钮，已登录时会显示账户名和头像。

主体里面有分主次，占据主要版区的分享列表以及悬浮于右侧的排行榜与最新榜。排行榜与最新榜也跟登录组件一样因其独立性被分割出去作为单个组件。而分享列表则是通过 router-view 的方式搭配路由选择来选择其它的子路由页面。可以看出主页这一块更多的是作为一个框架用来承载其他组件和页面的。

```
<script setup>
import Card from "@/components/Card";
import { nextTick, onMounted, ref, watch } from "vue";
import { message } from "ant-design-vue";
import { HomeFilled } from "@ant-design/icons-vue";
import UserBox from "@/components/UserBox";
import { useRouter } from "vue-router";
import { getHotSearch } from "@/hooks/share";

const searchText = ref("");
const selectedKeys = ref(["all"]);
const $router = useRouter();
const hotSearch = ref([]);

const onHotSearch = (content) => {
```

```
    searchText.value = content;
    search();
};
onMounted(async () => {
    try {
        hotSearch.value = await getHotSearch();
    } catch (err) {
        console.error(err);
        message.error("获取热搜失败");
    }
});
const search = () => {
    searchText.value = searchText.value.trim();
    $router.push({
        path: "/searchShares",
        query: { content: searchText.value },
    });
    searchText.value = "";
};
watch(
    selectedKeys,
    (selectedKeys) => {
        const key = selectedKeys[0];
        switch (key) {
            case "all":
                $router.push("/allShares");
                break;
            case "new":
                $router.push("/newShares");
                break;
            case "like":
                $router.push("/likeShares");
                break;
        }
    },
    { immediate: true }
);
</script>
```

从上面也可看出整个主页的逻辑是作为基底框架服务的，具体的呈现业务都被封装进其他更具体的页面或组件中去了。

3.3.1.2. Component

3.3.1.2.1. 实现的组件

Card
Share
Update
UserBox

3.3.1.2.2. 实现方式说明

以 Share 为例：

```
<template>
  <a-card>
    <a-card-meta>
      <template #avatar>
        <a-avatar
          style="cursor: pointer"
          @click="
            $router.push({
              path: '/person',
              query: { id: share.userId },
            })
          "
          :src="getAvatarUrl(share.userAvatarUrl)"
      /></template>
      <template #title
        >{{ share.username }}
        <div class="time">
          <a-tooltip :title="dayjs(share.time).format('YYYY-MM-DD HH:mm:ss')">
            <span>{{ dayjs(share.time).fromNow() }}</span>
          </a-tooltip>
        </div>
      </template>
      <template #description
        ><p
          class="share-content"
          style="
            cursor: pointer;
            word-wrap: break-word;
            white-space: normal !important;
          "
          @click="
            $router.push({
              path: '/share',
              query: { id: share.id },
```

```
                })
              "
            >
              {{ share.content }}
            </p>
            <div class="imageBox">
              <a-image-preview-group>
                <a-image
                  v-for="pic in share.pictureList"
                  :key="pic.id"
                  :width="200"
                  :src="getImageUrl(pic.url)"
                />
              </a-image-preview-group>
            </div>
          </template>
        </a-card-meta>
        <template #actions>
          <div @click="onLike">
            <like-filled v-if="share.liked" />
            <like-outlined v-else /><span>{{ share.likeNum }}</span>
          </div>
          <div @click="onComment">
            <comment-outlined /><span>{{ share.commentNum }}</span>
          </div>
          <div @click="onCollect">
            <star-filled v-if="share.collected" />
            <star-outlined v-else /><span>{{ share.collectNum }}</span>
          </div>
        </template>
      </a-card>
      <a-card v-if="commentVisible">
        <div class="comment clearfix">
          <a-textarea
            v-model:value="commentContent"
            :maxlength="100"
            placeholder="输入内容"
          ></a-textarea>
          <a-button type="primary" @click="onPublishComment">发表</a-button>
        </div>
        <Comment
          v-for="comment in commentList"
          :key="comment.id"
          :comment="comment"
        ></Comment>
```

```
      <div class="pagination">
        <a-pagination
          v-model:current="current"
          v-model:total="total"
          v-model:pageSize="pageSize"
          hideOnSinglePage
          show-less-items
          @change="pageChange"
        />
      </div>
    </a-card>
</template>
```

本体是在 antd 里的 card 组件上做文章，在本体 card 上的 action 插槽上安放点赞、收藏、评论的按钮，再在评论被点击后将第二张 card 显示出来并遍历加载所有搜索到的评论，每一个评论也正如 Share 组件一样被独立分装，Share 组件在页面中的使用方式也正如在该组件中的评论组件中一样遍历渲染所有搜寻到的分享。同样的操作在评论组件中也上映了一遍，即在评论组件中亦有将回复组件遍历渲染的案例。层层叠套，组件与组件间界限分明、互不逾越、各安其职地将 Share 这一整个复杂的组件拆分化简。

3.3.2. 后端

3.3.2.1. controller 层

3.3.2.1.1. 实现的类

AdministratorController
CollectController
CommentController
FileController
FollowController
HotSearchController
LikeController
ReplyController
ShareController
UserController

3.3.2.1.2. 实现方式说明

以 CollectController 为例：

```
@RestController
@RequestMapping(value = "collect")
public class CollectController {
    @Autowired
    private CollectService collectService;
    @Autowired
    private ShareService shareService;
```

```
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public boolean add(@RequestBody Collect collect,HttpSession session){
        SystemUser user = (SystemUser) session.getAttribute("user");
        if(user == null || !user.getRole().equals("user")){
            return false;
        }
        collect.setUserId(user.getId());
        return collectService.add(collect);
    }

    @RequestMapping(value = "/get", method = RequestMethod.GET)
    public List<ShareListInfo> get(@RequestParam long begin, @RequestParam long length,
HttpSession session){
        SystemUser user = (SystemUser) session.getAttribute("user");
        if(user == null || !user.getRole().equals("user")){
            return null;
        }
        List<ShareListInfo>                    shareListInfoList                    =
collectService.getCollectList(begin,length,user.getId());
        shareService.setLikeAndCollectState(shareListInfoList, user.getId());
        return shareListInfoList;
    }

    @RequestMapping(value = "/getcollectnum", method = RequestMethod.GET)
    public long getCollectNum(HttpSession session){
        SystemUser user = (SystemUser) session.getAttribute("user");
        if(user == null || !user.getRole().equals("user")){
            return 0;
        }
        return collectService.getCollectNum(user.getId());
    }

    @RequestMapping(value = "/cancel/{shareId}", method = RequestMethod.DELETE)
    public boolean cancel(@PathVariable long shareId, HttpSession session){
        SystemUser user = (SystemUser) session.getAttribute("user");
        if(user == null || !user.getRole().equals("user")){
            return false;
        }
        return collectService.cancel(shareId,user.getId());
    }
}
```

CollectController 类存放收藏相关接口。由于系统采用前后端分离的方式进行开发，后端只需向前端返回 json 形式的数据，而不用控制页面的跳转，因此使用@RestController 注解标注 CollectController 类。通过使用@RequestMapping 标注类和方法配置 controller

方法的请求路径和请求方式。

add 方法中：接收前端传递的 collect 对象。使用 HttpSession 对象获取当前登录用户的信息。若当前用户为空，即未登录，或者当前登录用户的角色不是”user”，则向前端返回添加收藏失败的信息。否则，将当前用户的 ID 放入 collect 对象中，调用并将 collect 对象传递给 collectService 对象中的 add 方法。向前端返回收藏是否成功的信息。

get 方法中：接收前端传递的 begin 和 length，这两个参数是分页信息，表示当前页起始的下标和长度。使用 HttpSession 对象获取当前登录用户的信息。若当前用户为空，即未登录，或者当前登录用户的角色不是”user”，则向前端返回空值。否则调用并将分页信息和当前用户 ID 传递给 collectService 对象中的 getCollectList 方法，获取收藏列表。调用并将收藏列表和用户 ID 传递给 shareService 对象中的 setLikeAndCollectState 方法，使收藏列表中的分享附带上当前用户对该分享的点赞和收藏状态信息。将收藏列表返回给前端。

getCollectNum 方法中：使用 HttpSession 对象获取当前登录用户的信息。若当前用户为空，即未登录，或者当前登录用户的角色不是”user”，则向前端返回空的信息。否则调用并传递用户 ID 给 collectService 对象中的 getCollectNum 方法，获取并返回用户收藏的个数。

cancel 方法中：接收前端传递的 shareId，表示要取消收藏的分享的 ID。使用 HttpSession 对象获取当前登录用户的信息。若当前用户为空，即未登录，或者当前登录用户的角色不是”user”，则向前端返回失败信息。否则调用并传递分享 ID 和用户 ID 给 collectService 对象中的 cancel 方法，并返回取消收藏的结果。

3.3.2.2. service 层

3.3.2.2.1. 实现的类

CollectServiceImpl
CommentServiceImpl
FileServiceImpl
FollowServiceImpl
HotSearchServiceImpl
LikeServiceImpl
PictureServiceImpl
ReplyServiceImpl
SensitiveKeywordServiceImpl
ShareServiceImpl
UserInfoServiceImpl
UserServiceImpl

3.3.2.2.2. 实现方式说明

以 HotSearchServiceImpl 为例：

```
@Service
public class HotSearchServiceImpl implements HotSearchService {
```

```
    @Autowired
    private RedisTemplate<String,String> redisTemplate;

    @Override
    public List<HotSearch> getList() {
        List<HotSearch> hotSearchList = new ArrayList<>();
        Set<String> hotSearchSet =
redisTemplate.opsForZSet().reverseRange("hotSearchSet", 0, 9);
        assert hotSearchSet != null;
        Iterator<String> iterator = hotSearchSet.iterator();
        while (iterator.hasNext()){
            String hotSearch = iterator.next();
            long times =
redisTemplate.opsForZSet().score("hotSearchSet",hotSearch).longValue();
            hotSearchList.add(new HotSearch(hotSearch,times));
        }
        return hotSearchList;
    }
}
```

HotSearchServiceImpl 为热搜榜业务逻辑实现类。其中的 getList 方法为获取热搜榜，返回值为热搜榜链表。首先使用 RedisTemplate 类的实例 redisTemplate 获取 key 为 hotSearchSet 的 sorted set 的逆序前十的 value。若值不为 null，则遍历获取到的 value，并使用 redisTemplate 获取该 value 对应的 score。将 value 和 score 封装到 HotSearch 类的实例中，并将实例加入结果链表中，value 和 score 分别表示热搜词和搜索次数。最后返回结果链表。

3.3.2.3. mapper 层

3.3.2.3.1. 实现的类

CollectMapper
CommentMapper
FollowMapper
LikeMapper
PictureMapper
ReplyMapper
SensitiveKeywordMapper
ShareMapper
UserInfoMapper
UserMapper

3.3.2.3.2. 实现方式说明

以 CollectMapper 为例：

```
@Mapper
public interface CollectMapper {
	public boolean insert(Collect collect);

	public List<ShareListInfo> getCollectList(long begin, long length, long userId);

	public int isCollected(long shareId, long userId);

	public boolean delete(long shareId, long userId);

	public long getCollectNum(long id);
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
		PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="edu.scau.pyx.ssp.mapper.CollectMapper">
	<insert id="insert">
		INSERT INTO T_COLLECT( USER_ID,SHARE_ID )
		VALUES ( #{userId},#{shareId} );
	</insert>
	<delete id="delete">
		DELETE FROM T_COLLECT
		WHERE SHARE_ID = #{shareId} AND USER_ID = #{userId};
	</delete>
	<select id="getCollectList" resultType="edu.scau.pyx.ssp.entity.ShareListInfo">
		SELECT S.ID AS id,
			U.`NAME` AS username,
			I.AVATAR_URL AS userAvatarUrl,
			S.TIME AS time,
			S.CONTENT AS content,
			S.STATE AS state,
			S.LIKENUM AS likeNum,
			S.COLLECTNUM AS collectNum,
			S.COMMENTNUM AS commentNum
		FROM T_SHARE AS S,T_USER AS U,T_USER_INFO AS I,T_COLLECT AS C,T_USER AS SU
		WHERE S.USER_ID=U.ID AND U.ID=I.USER_ID AND C.SHARE_ID=S.ID AND
C.USER_ID=#{userId} AND S.STATE!='private' AND S.STATE!='deleted'  AND S.USER_ID=SU.ID
AND SU.LOCKED=0
		ORDER BY C.COLLECT_TIME DESC
		LIMIT #{begin},#{length};
	</select>
	<select id="isCollected" resultType="int">
```

```
        SELECT COUNT(*)
        FROM T_COLLECT
        WHERE USER_ID = #{userId} AND SHARE_ID = #{shareId};
    </select>
    <select id="getCollectNum" resultType="long">
        SELECT COUNT(*)
        FROM T_SHARE AS S,T_USER AS U,T_USER_INFO AS I,T_COLLECT AS C,T_USER AS SU
        WHERE S.USER_ID=U.ID AND U.ID=I.USER_ID AND C.SHARE_ID=S.ID AND
C.USER_ID=#{userId} AND S.STATE!='private' AND S.STATE!='deleted'  AND S.USER_ID=SU.ID
AND SU.LOCKED=0
    </select>
</mapper>
```

在 mybatis 映射文件中编写 SQL 语句实现 mapper 接口中的方法。

insert 方法：

```
INSERT INTO T_COLLECT( USER_ID,SHARE_ID )
VALUES ( #{userId},#{shareId} );
```

一条插入语句，将 collect 实例中获取的 userId 和 shareId 插入 T_COLLECT 表中。

getCollectList 方法：

```
SELECT S.ID AS id,
            U.`NAME` AS username,
            I.AVATAR_URL AS userAvatarUrl,
            S.TIME AS time,
            S.CONTENT AS content,
            S.STATE AS state,
            S.LIKENUM AS likeNum,
            S.COLLECTNUM AS collectNum,
            S.COMMENTNUM AS commentNum
FROM T_SHARE AS S,T_USER AS U,T_USER_INFO AS I,T_COLLECT AS C,T_USER AS SU
WHERE S.USER_ID=U.ID AND U.ID=I.USER_ID AND C.SHARE_ID=S.ID AND
C.USER_ID=#{userId} AND S.STATE!='private' AND S.STATE!='deleted'  AND S.USER_ID=SU.ID
AND SU.LOCKED=0
ORDER BY C.COLLECT_TIME DESC
LIMIT #{begin},#{length};
```

对 T_SHARE、T_USER、T_USER_INFO、T_COLLECT 表的联合查询。查询条件为收藏用户为当前用户，分享未设置为私有或被删除，发布分享的用户未被锁定。查询结果按照时间降序排列。查询结果行偏移量为 begin，最大 length 个。

isCollected 方法：

```
SELECT COUNT(*)
FROM T_COLLECT
WHERE USER_ID = #{userId} AND SHARE_ID = #{shareId};
```

从 T_COLLECT 表中查询用户 ID 为 userId、分享 ID 为 shareId 的记录数。

delete 方法：

```
DELETE FROM T_COLLECT
WHERE SHARE_ID = #{shareId} AND USER_ID = #{userId};
```

从 T_COLLECT 表中删除用户 ID 为 userId、分享 ID 为 shareId 的记录。

getCollectNum 方法：

```
SELECT COUNT(*)
FROM T_SHARE AS S,T_COLLECT AS C,T_USER AS SU
WHERE C.SHARE_ID=S.ID AND C.USER_ID=#{userId} AND S.STATE!='private' AND S.STATE!='deleted' AND S.USER_ID=SU.ID AND SU.LOCKED=0
```

对 T_SHARE、T_COLLECT、T_USER 表联合查询，查询条件为收藏用户 ID 为 userId、分享未被删除或设置为私有、发布分享的用户未被锁定，查询结果为符合条件的记录数。

3.3.2.4. entity 层

3.3.2.4.1. 实现的类

Administrator
Collect
Comment
CommentListInfo
Follow
HotSearch
Like
Picture
Reply
ReplyListInfo
Role
SensitiveKeyword
Share
ShareListInfo
SystemUser
UserInfo
UserListInfo

3.3.2.4.2. 实现方式说明

以 Share 为例：

```
public class Share {
    private long id;
    private String content;
    private String state;
    private long likeNum;
    private long collectNum;
    private long commentNum;
```

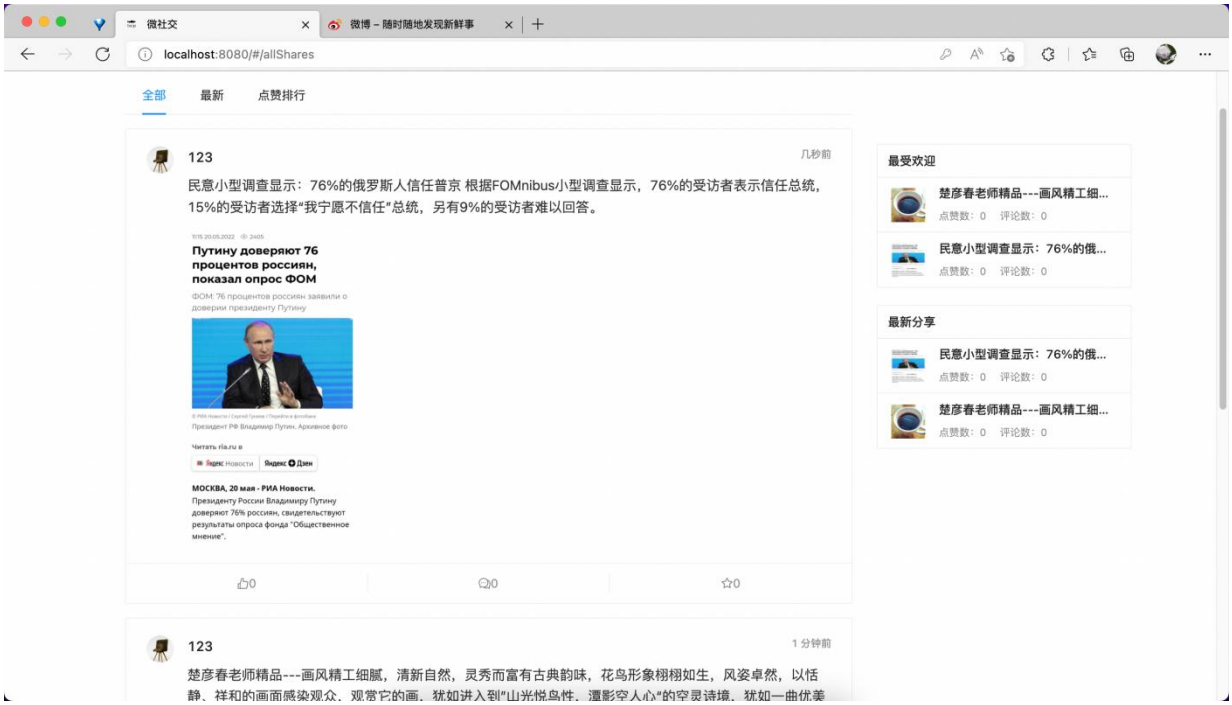
```
    private long userId;

    public Share() {
    }

    public Share(long id, String content, String state, long likeNum, long collectNum, long commentNum, long userId) {
        this.id = id;
        this.content = content;
        this.state = state;
        this.likeNum = likeNum;
        this.collectNum = collectNum;
        this.commentNum = commentNum;
        this.userId = userId;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public long getLikeNum() {
        return likeNum;
    }

    public void setLikeNum(long likeNum) {
```

```
        this.likeNum = likeNum;
    }

    public long getCollectNum() {
        return collectNum;
    }

    public void setCollectNum(long collectNum) {
        this.collectNum = collectNum;
    }

    public long getCommentNum() {
        return commentNum;
    }

    public void setCommentNum(long commentNum) {
        this.commentNum = commentNum;
    }

    public long getUserId() {
        return userId;
    }

    public void setUserId(long userId) {
        this.userId = userId;
    }
}
```

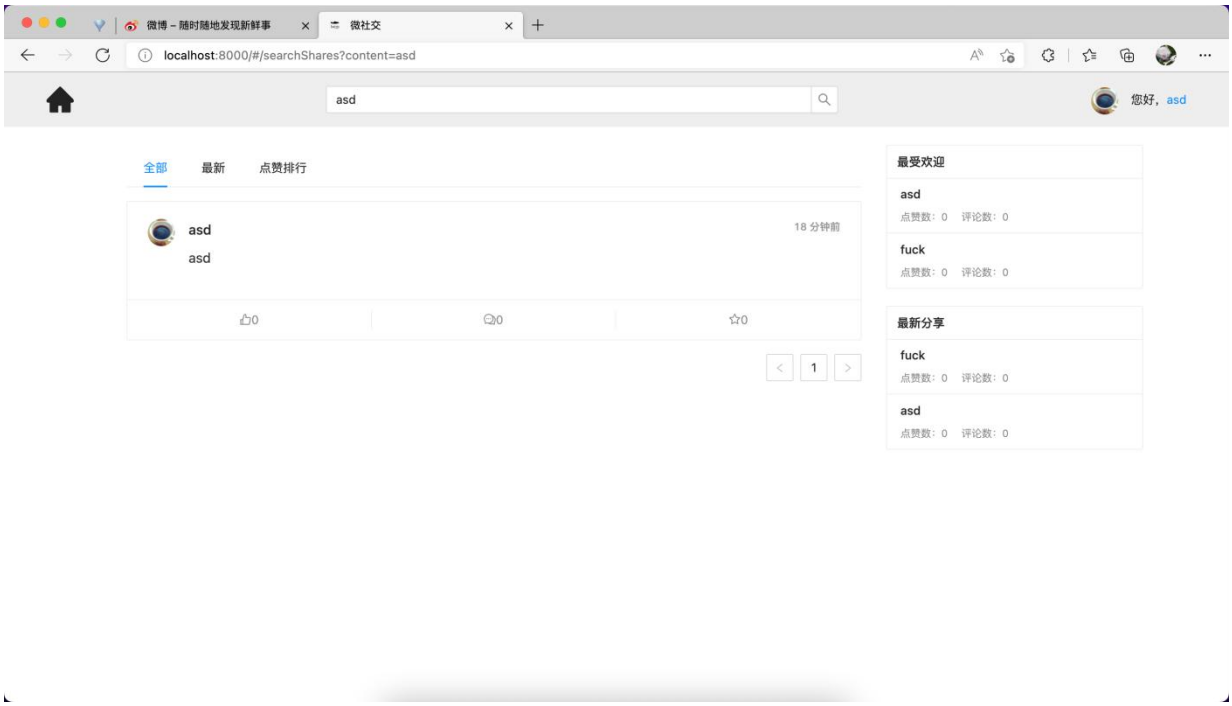
属性均使用 private 修饰，向外界提供 public 修饰的无参构造方法、有参数的构造方法和属性的 getter 和 setter 方法。

4. 软件测试

4.1. 显示分享列表

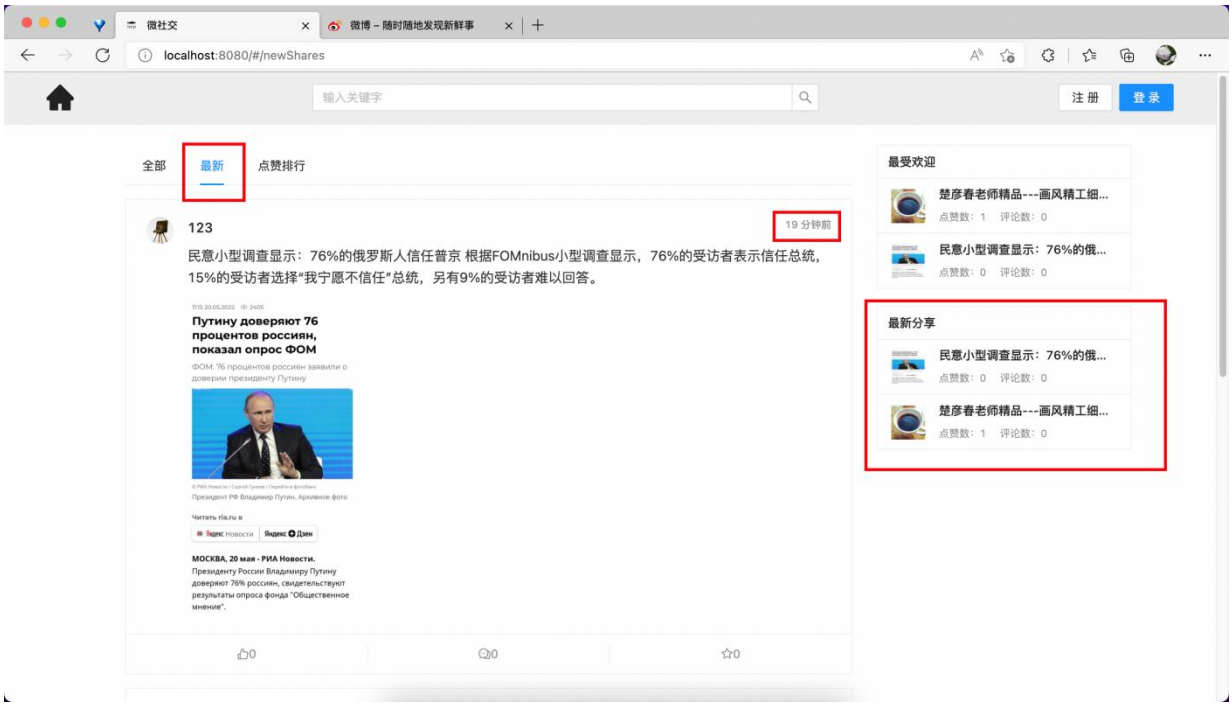


4.2. 搜索分享

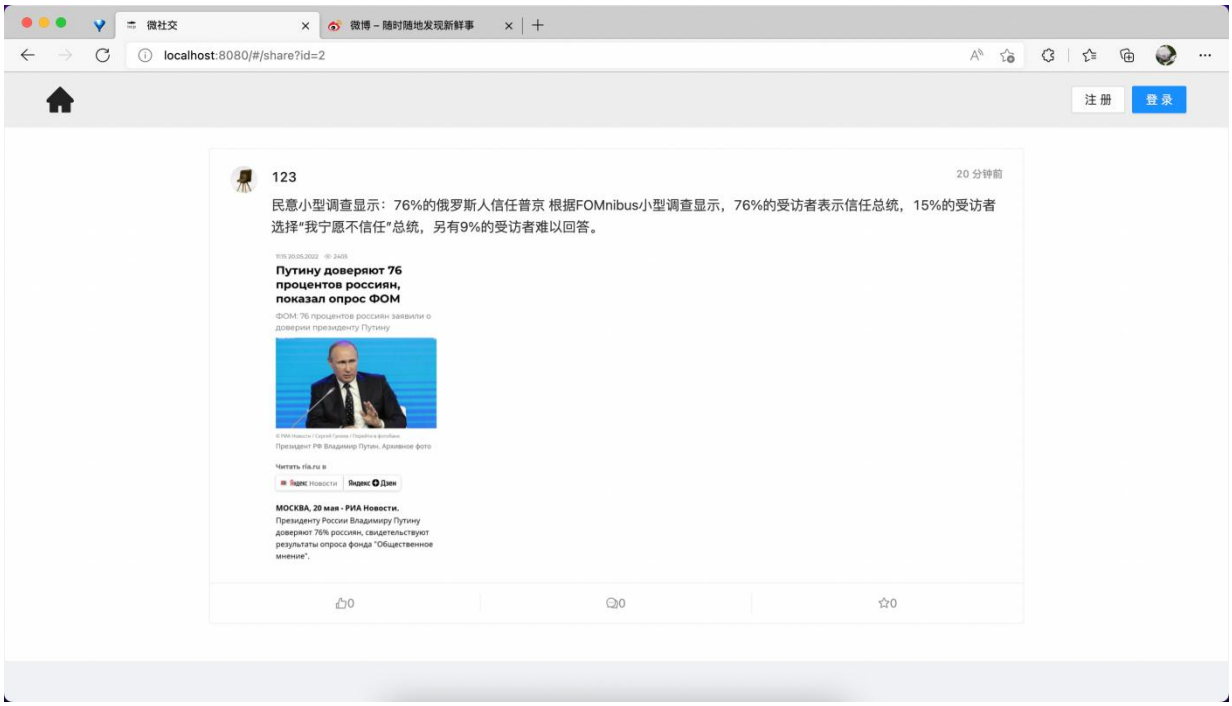


4.3. 点赞榜

4.4. 最新分享

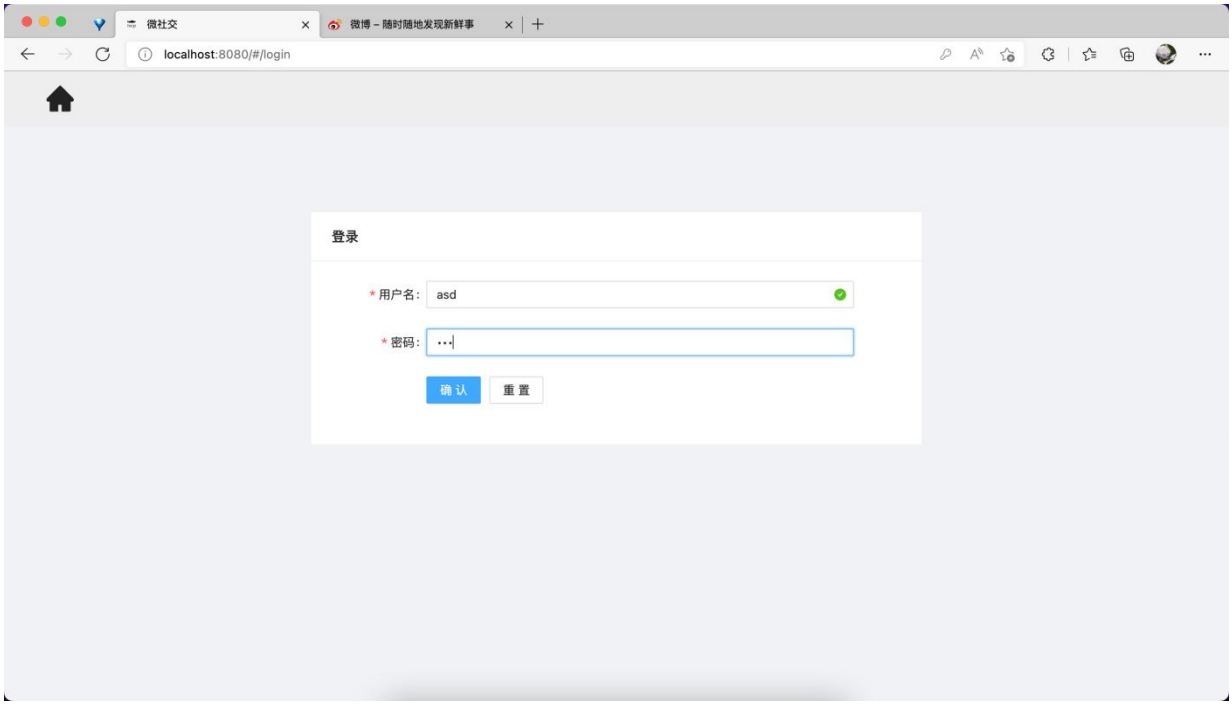


4.5. 查看分享

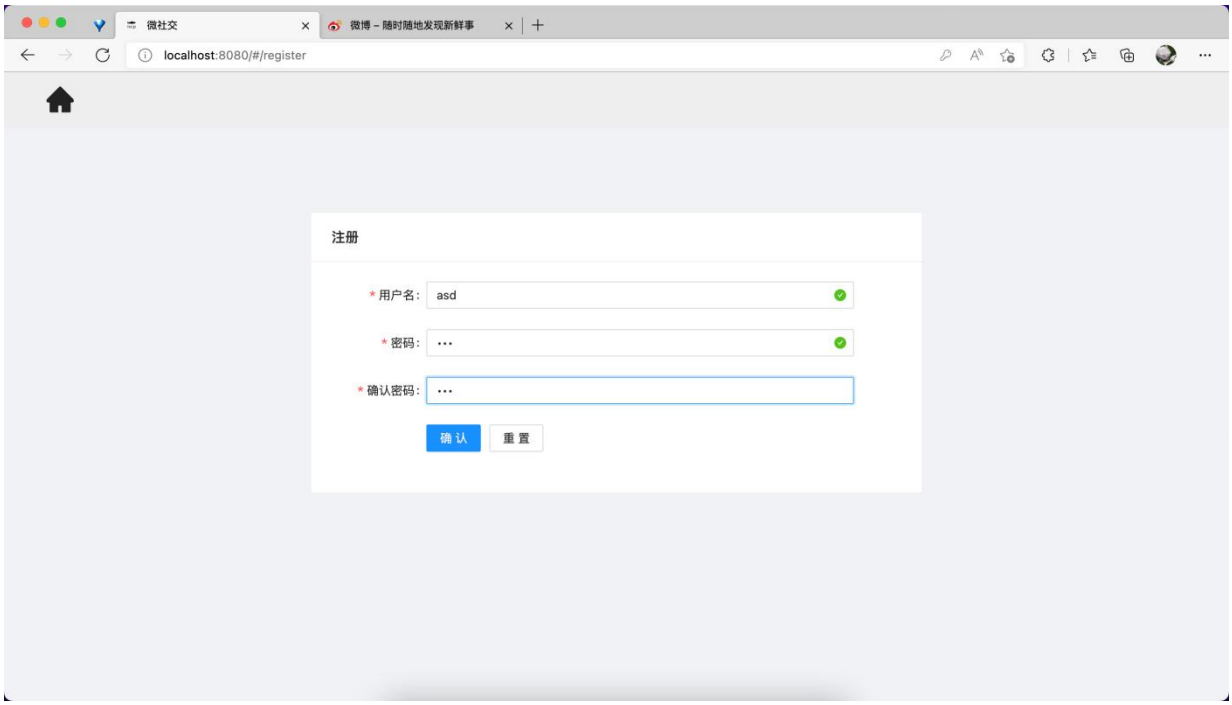


4.6. 登录/注册

4.6.1. 登录



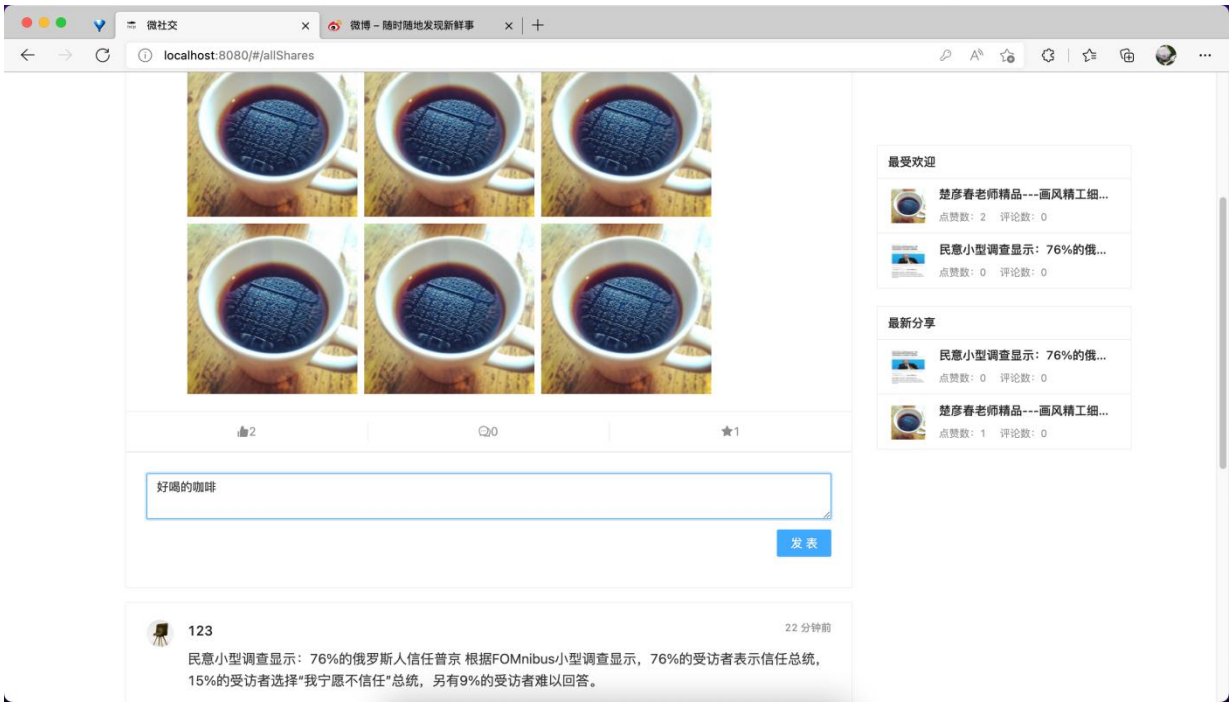
4.6.2. 注册



4.7. 点赞分享

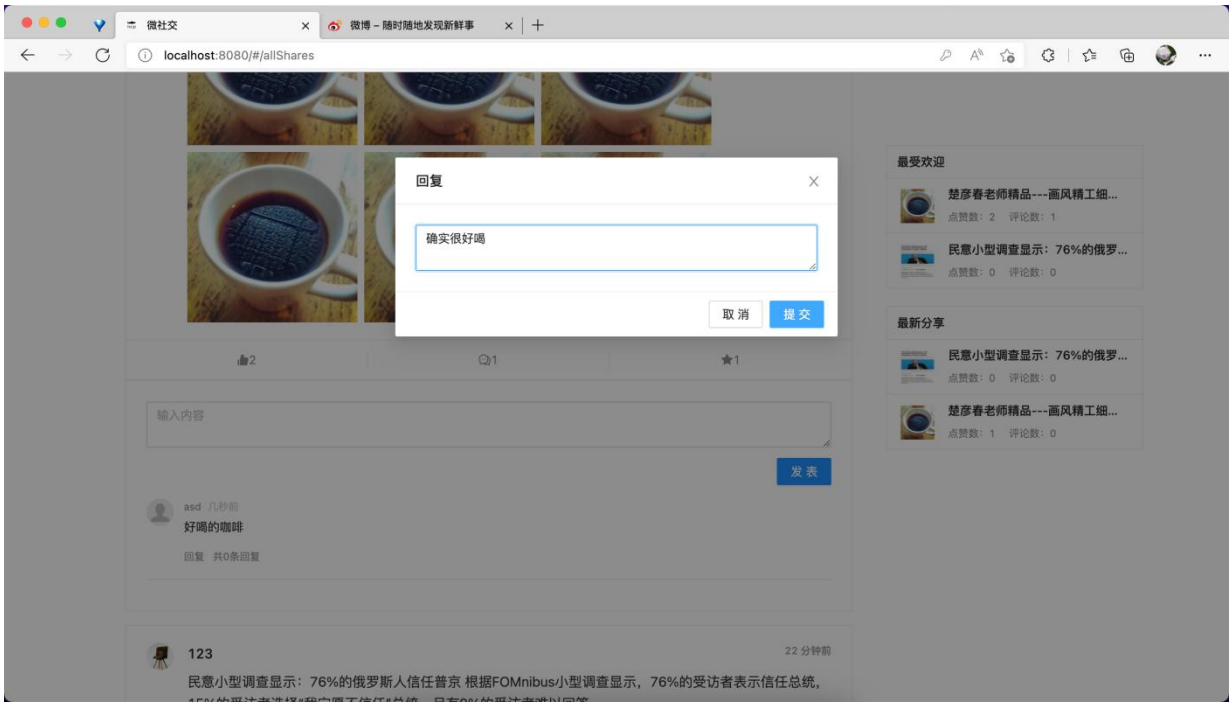
4.8. 收藏分享

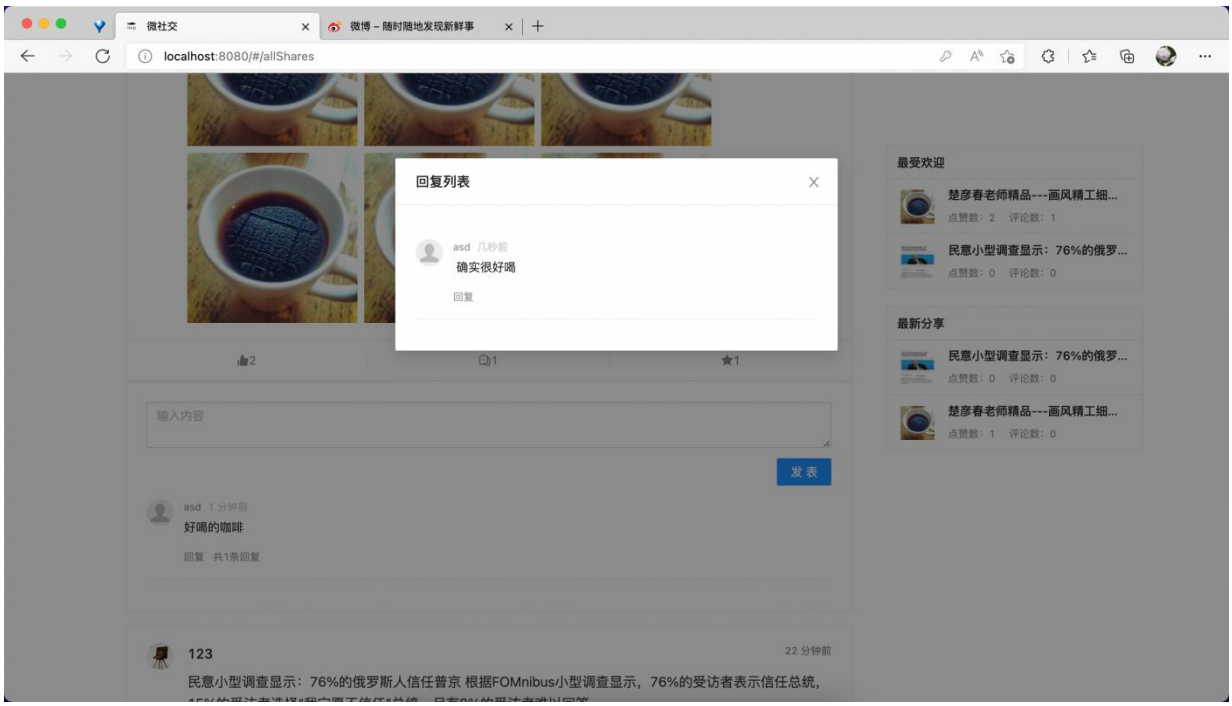
4.9. 评论分享

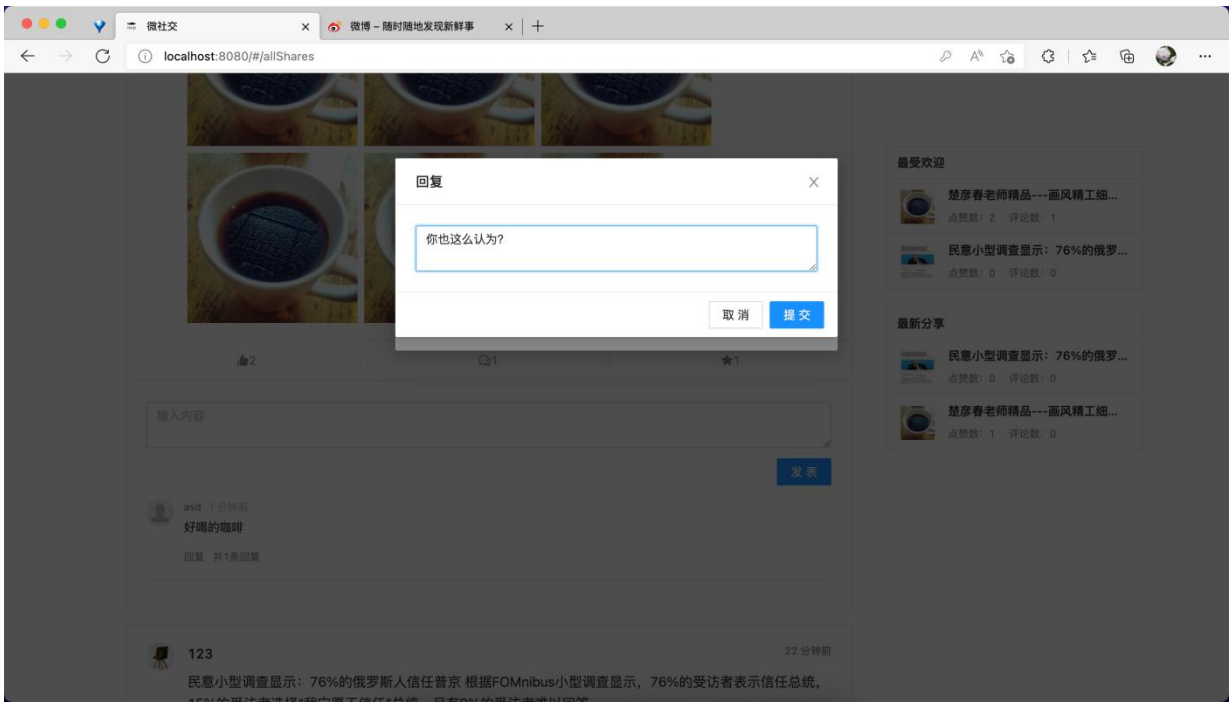

微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/allShares
最受欢迎
楚彦春老师精品---画风精工细...
点赞数：2 评论数：0
民意小型调查显示：76%的俄...
点赞数：0 评论数：0
最新分享
民意小型调查显示：76%的俄...
点赞数：0 评论数：0
楚彦春老师精品---画风精工细...
点赞数：1 评论数：0
2
0
1
好喝的咖啡
发 表
123
22 分钟前
民意小型调查显示：76%的俄罗斯人信任普京 根据FOMnibus小型调查显示，76%的受访者表示信任总统，
15%的受访者选择"我宁愿不信任"总统，另有9%的受访者难以回答。

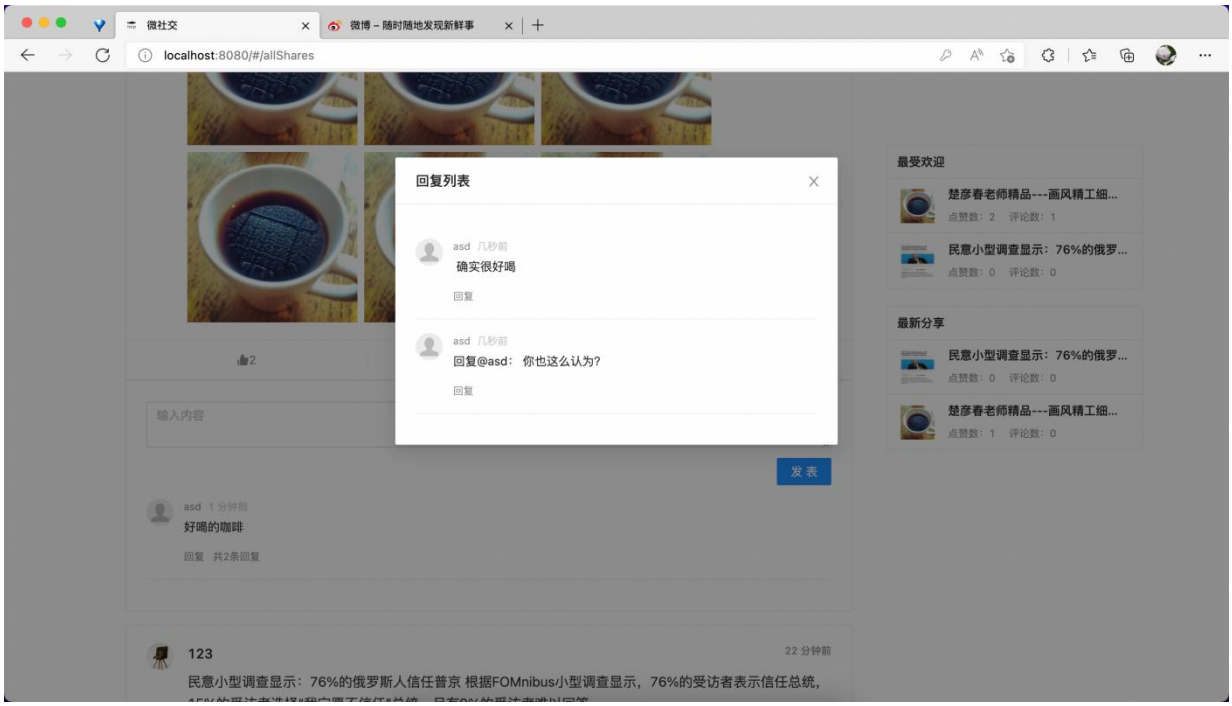

微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/allShares
最受欢迎
楚彦春老师精品---画风精工细...
点赞数：2 评论数：1
民意小型调查显示：76%的俄...
点赞数：0 评论数：0
最新分享
民意小型调查显示：76%的俄...
点赞数：0 评论数：0
楚彦春老师精品---画风精工细...
点赞数：1 评论数：0
2
1
1
输入内容
发 表
asd 几秒前
好喝的咖啡
回复 共0条回复
123
22 分钟前
民意小型调查显示：76%的俄罗斯人信任普京 根据FOMnibus小型调查显示，76%的受访者表示信任总统，

4.10. 回复评论

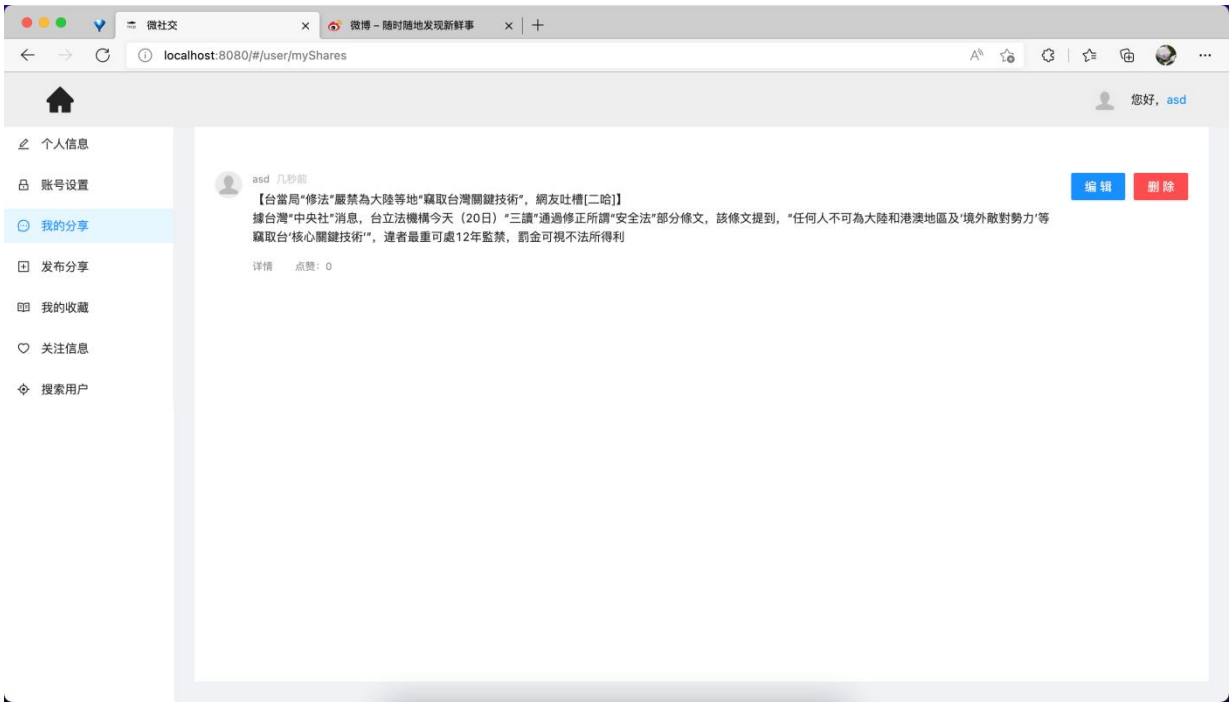

微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/allShares
回复
确实很好喝
取 消
提 交
最受欢迎
楚彦春老师精品---画风精工细...
民意小型调查显示：76%的俄罗...
最新分享
输入内容
发 表
asd 几秒前
好喝的咖啡
回复 共0条回复
123
22 分钟前
民意小型调查显示：76%的俄罗斯人信任普京 根据FOMnibus小型调查显示，76%的受访者表示信任总统，


微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/allShares
回复列表
asd 几秒前
确实很好喝
回复
最受欢迎
楚彦春老师精品---画风精工细...
民意小型调查显示：76%的俄罗...
最新分享
输入内容
发 表
asd 1 分钟前
好喝的咖啡
回复 共1条回复
123
22 分钟前
民意小型调查显示：76%的俄罗斯人信任普京 根据FOMnibus小型调查显示，76%的受访者表示信任总统，


微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/allShares
回复
你也这么认为?
取 消
提 交
最受欢迎
楚彦春老师精品---画风精工细...
民意小型调查显示：76%的俄罗...
最新分享
输入内容
发 表
asd 1 分钟前
好喝的咖啡
回复 共1条回复
123
22 分钟前
民意小型调查显示：76%的俄罗斯人信任普京 根据FOMnibus小型调查显示，76%的受访者表示信任总统，

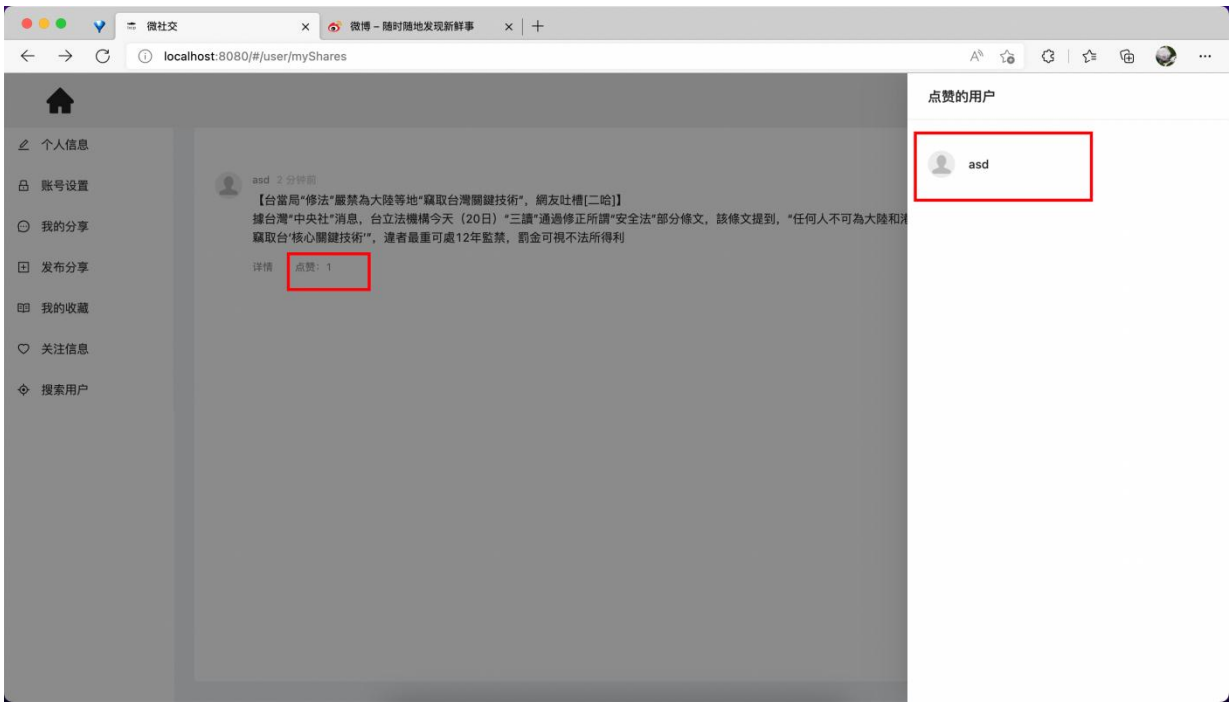

微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/allShares
回复列表
asd 几秒前
确实很好喝
回复
asd 几秒前
回复@asd： 你也这么认为?
回复
最受欢迎
楚彦春老师精品---画风精工细...
民意小型调查显示：76%的俄罗...
最新分享
输入内容
发 表
asd 1 分钟前
好喝的咖啡
回复 共2条回复
123
22 分钟前
民意小型调查显示：76%的俄罗斯人信任普京 根据FOMnibus小型调查显示，76%的受访者表示信任总统，

4.11. 我的分享

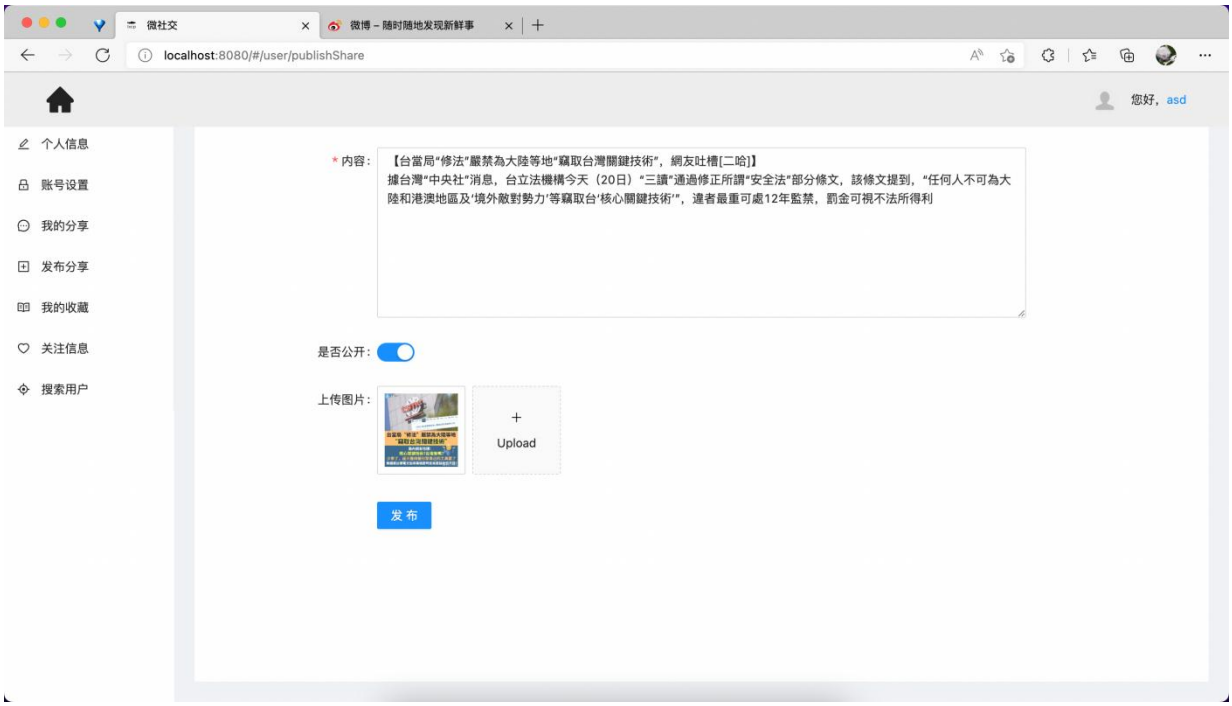


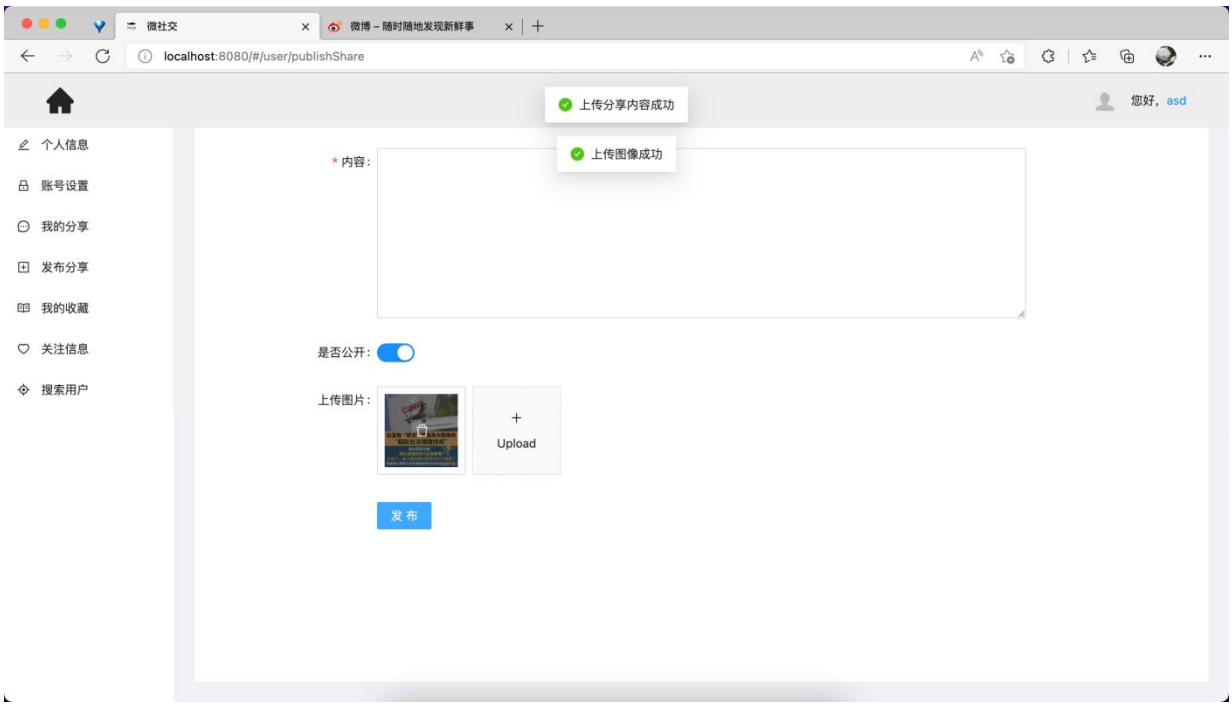
点击详情进入查看分享页面，可以进行分享的一切正常交互

点击点赞可以打开所有为这条分享点赞用户信息

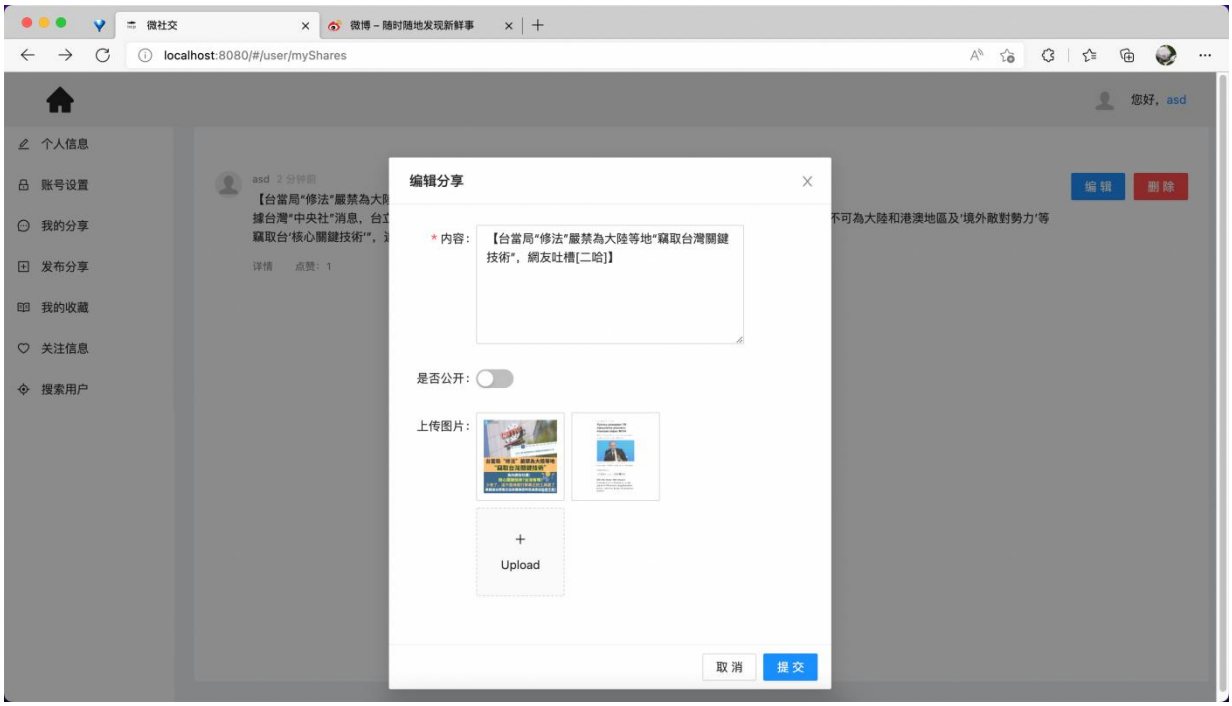

微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/user/myShares
个人信息
账号设置
我的分享
发布分享
我的收藏
关注信息
搜索用户
asd 2 分钟前
【台當局"修法"嚴禁為大陸等地"竊取台灣關鍵技術"，網友吐槽[二哈]】
據台灣"中央社"消息，台立法機構今天（20日）"三讀"通過修正所謂"安全法"部分條文，該條文提到，"任何人不可為大陸和港
竊取台'核心關鍵技術'"，違者最重可處12年監禁，罰金可視不法所得利
详情
点赞：1
点赞的用户
asd

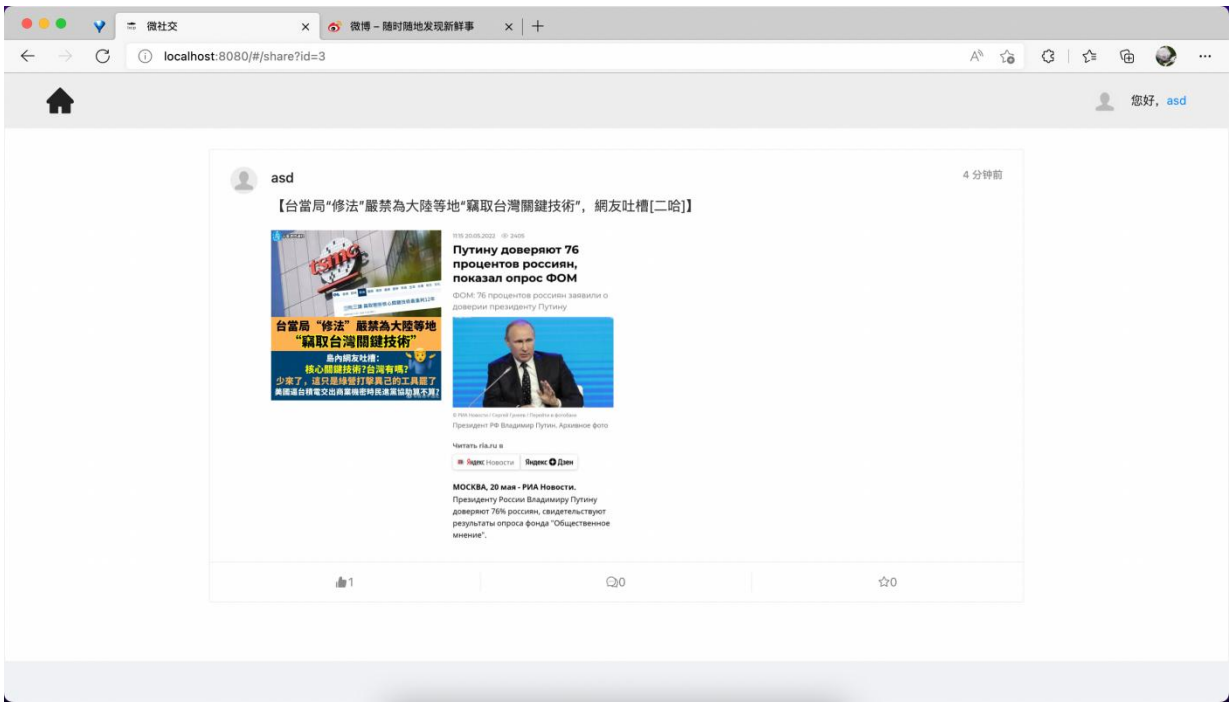
4.12. 发布分享


微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/user/publishShare
您好，asd
个人信息
账号设置
我的分享
发布分享
我的收藏
关注信息
搜索用户
* 内容：
【台當局"修法"嚴禁為大陸等地"竊取台灣關鍵技術"，網友吐槽[二哈]】
據台灣"中央社"消息，台立法機構今天（20日）"三讀"通過修正所謂"安全法"部分條文，該條文提到，"任何人不可為大陸和港澳地區及'境外敵對勢力'等竊取台'核心關鍵技術'"，違者最重可處12年監禁，罰金可視不法所得利
是否公开：
上传图片：
+
Upload
发布


微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/user/publishShare
上传分享内容成功
上传图像成功
您好，asd
个人信息
账号设置
我的分享
发布分享
我的收藏
关注信息
搜索用户
* 内容：
是否公开：
上传图片：
+
Upload
发布

4.13. 修改分享

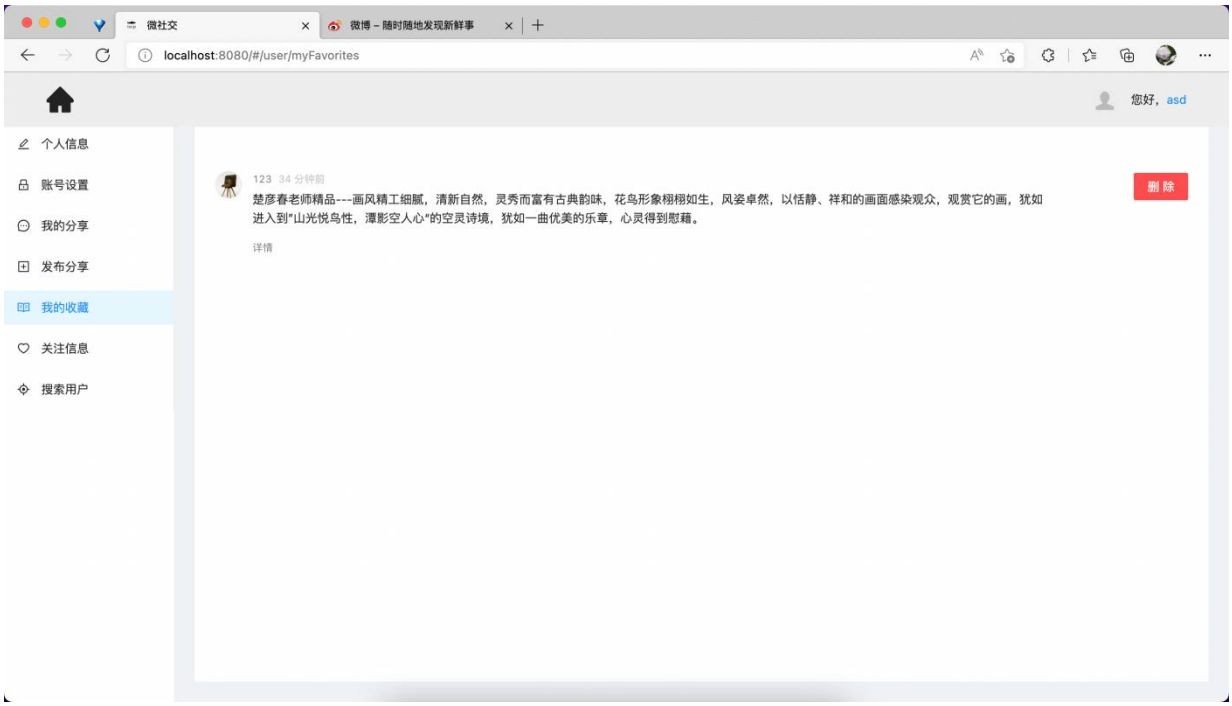

微社交
微博 – 随时随地发现新鲜事
localhost:8080/#/user/myShares
您好，asd
个人信息
账号设置
我的分享
发布分享
我的收藏
关注信息
搜索用户
编辑
删除
编辑分享
* 内容：
【台當局"修法"嚴禁為大陸等地"竊取台灣關鍵技術"，網友吐槽[二哈]】
是否公开：
上传图片：
+
Upload
取消
提交



4.14. 分享状态

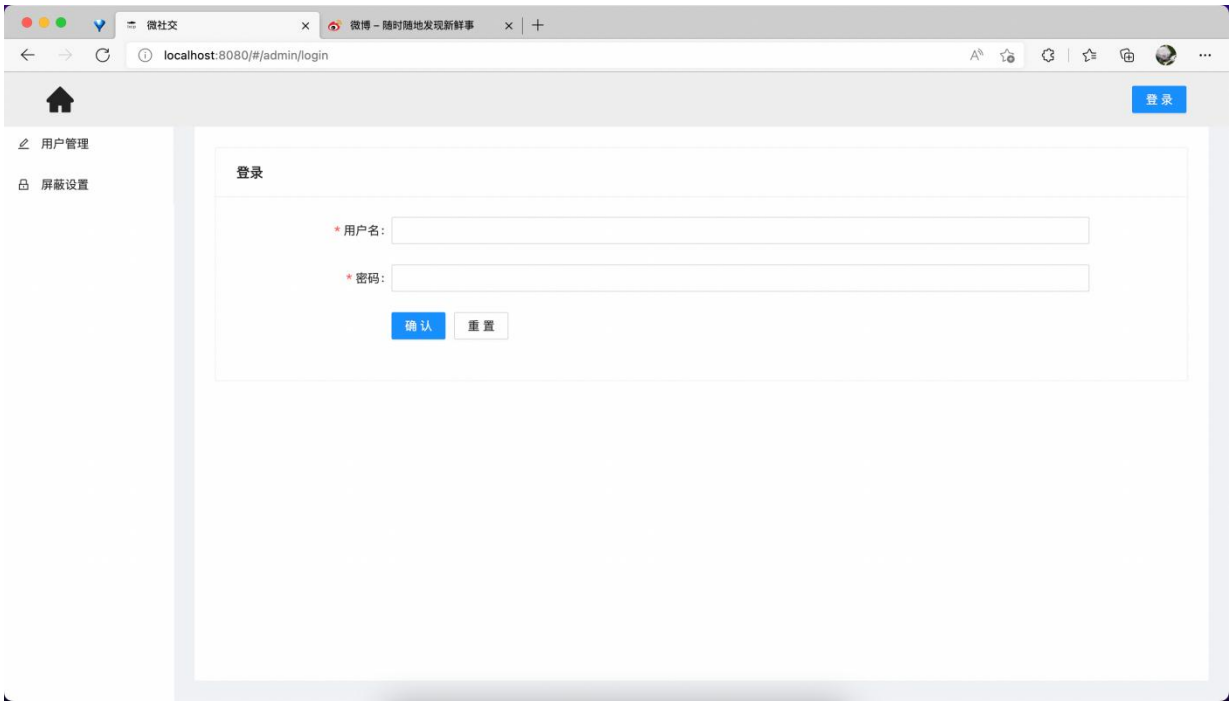
分享状态为不公开，在分享列表中不予展示，只在我的分享中可见

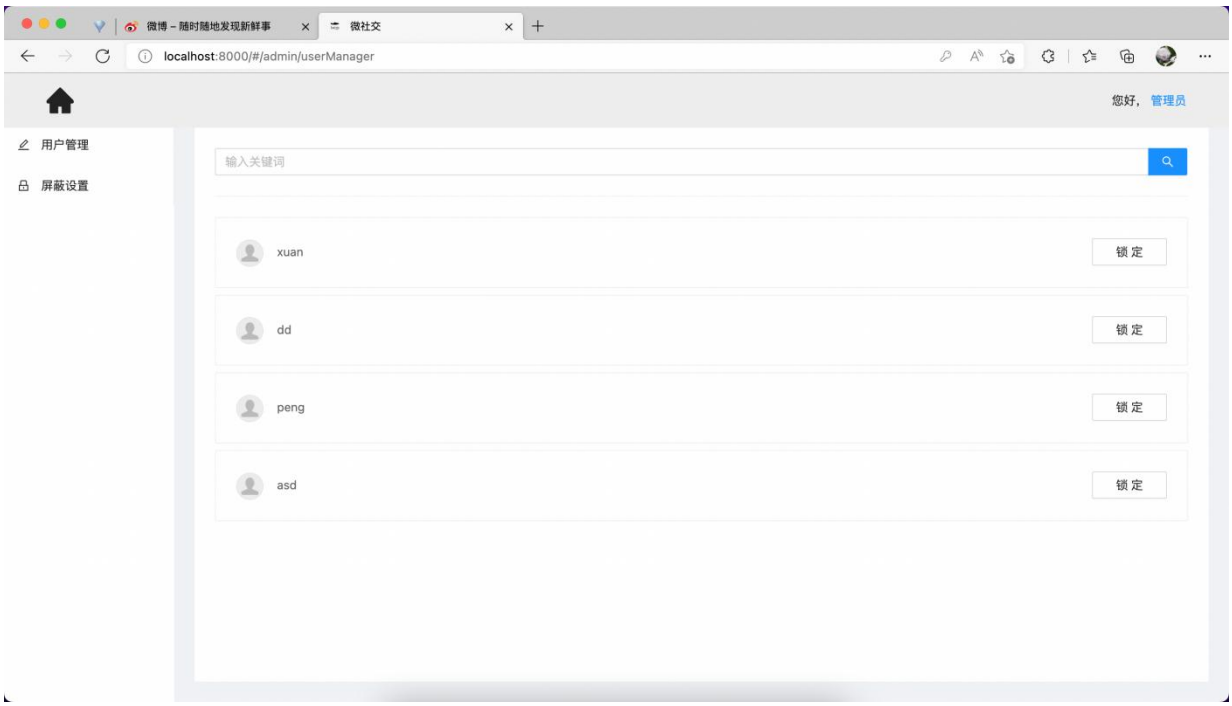
4.15. 我的收藏



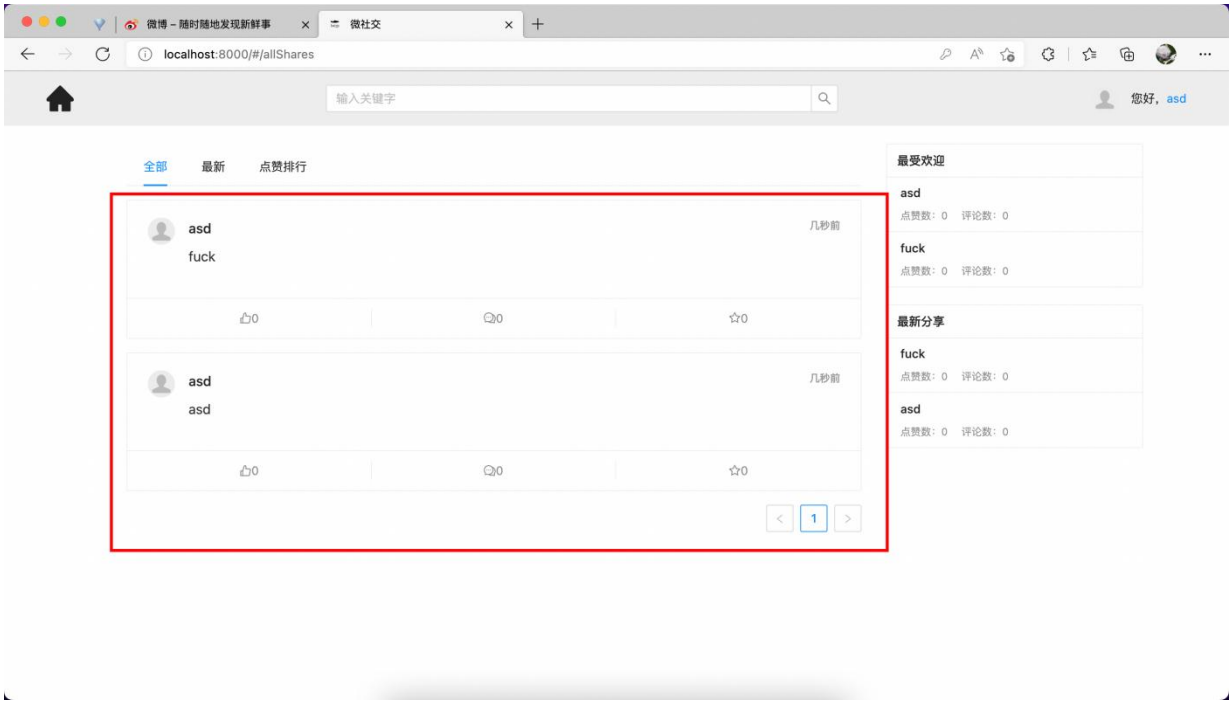
4.16. 注册用户管理

管理员登录

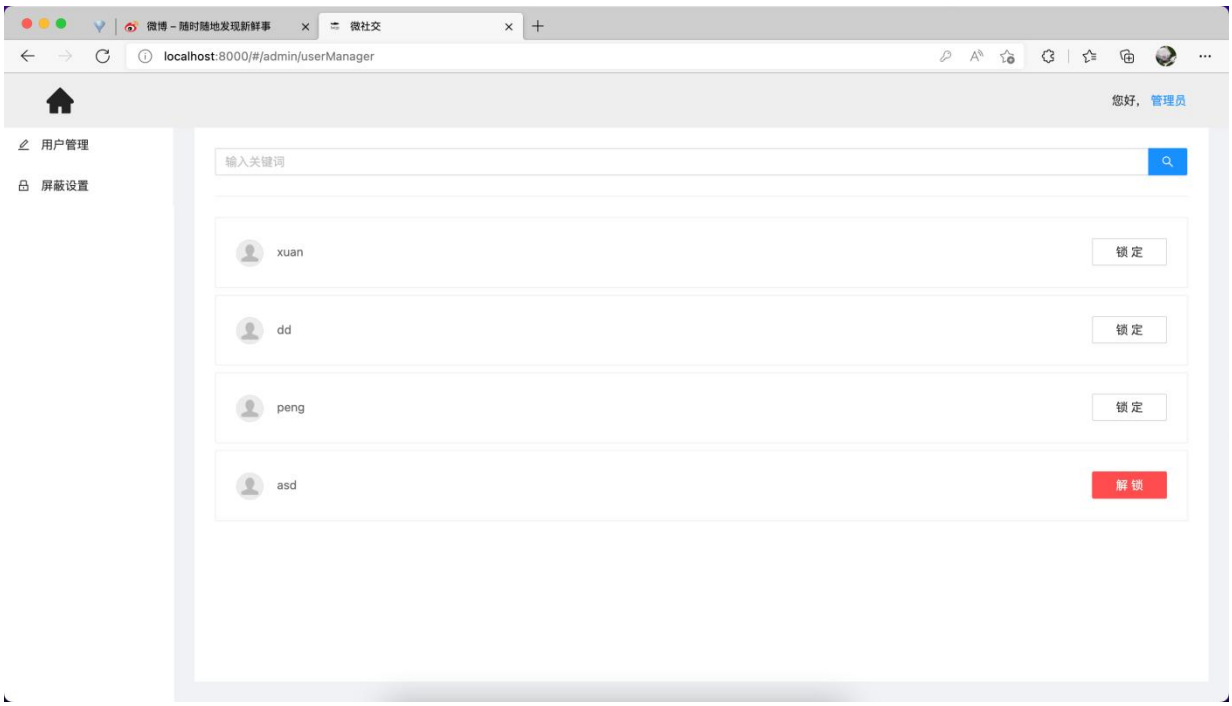




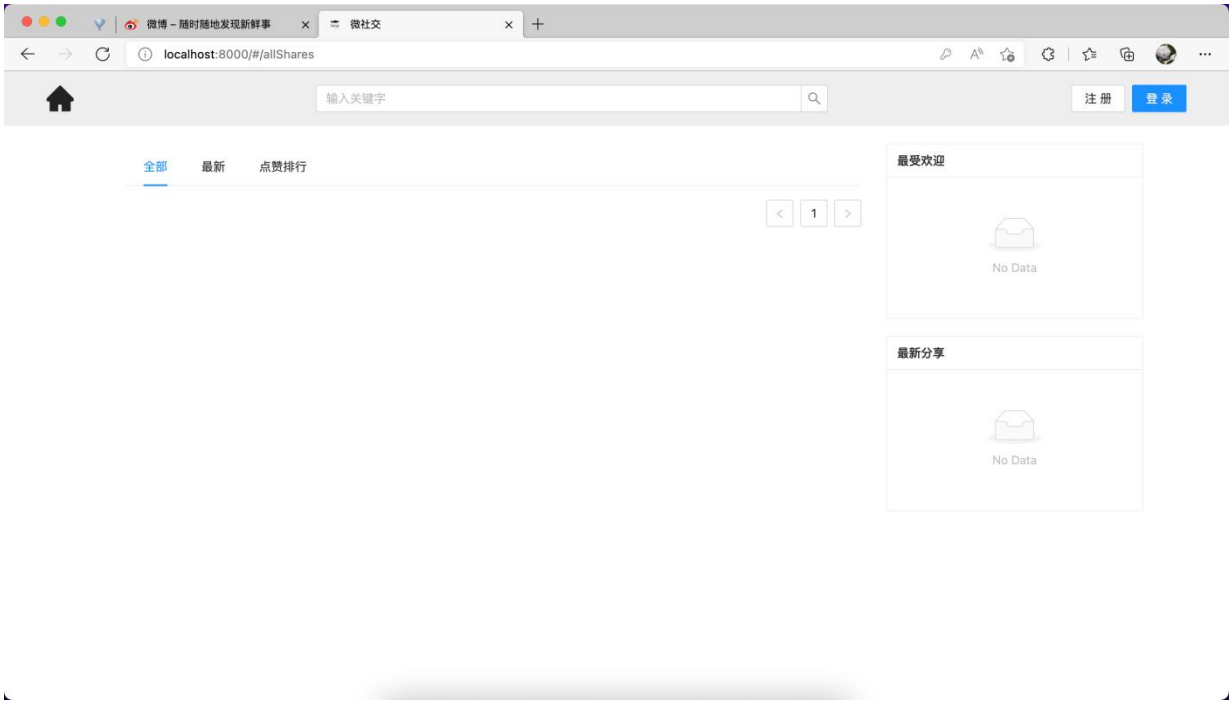
锁定某个用户后其下的所有分享不予公开



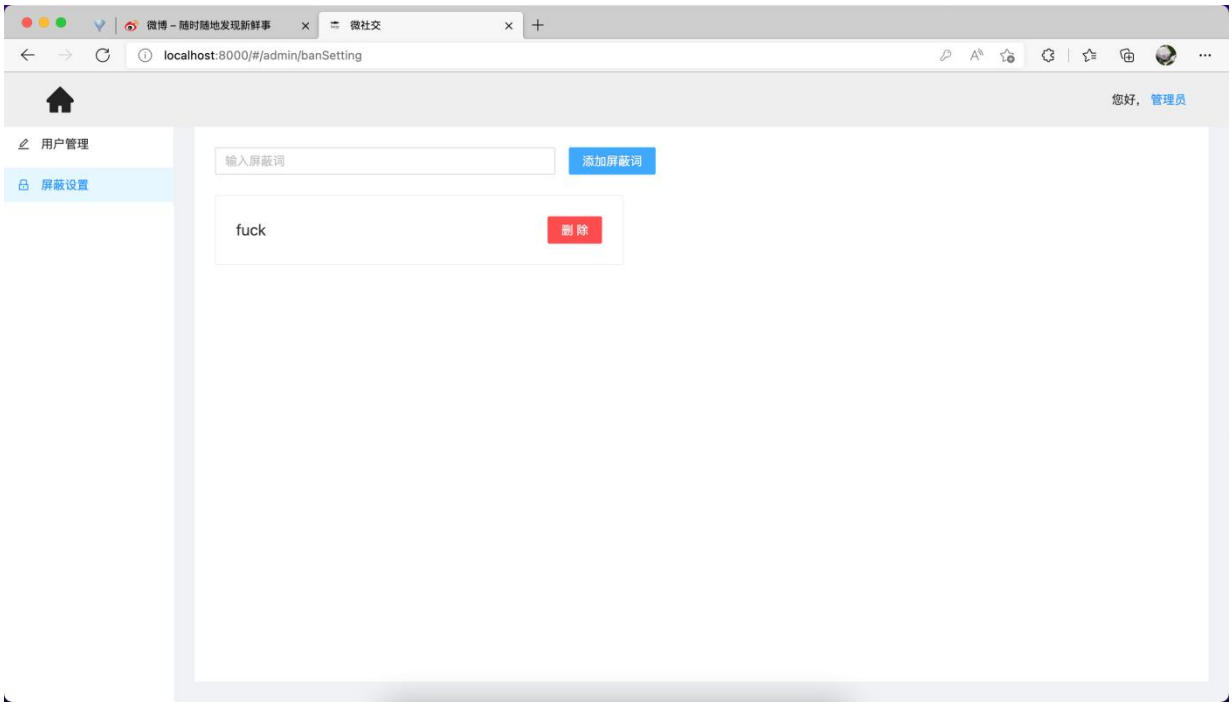
现在用户 asd 有两篇公开的分享，被管理员锁定后


微博 – 随时随地发现新鲜事
微社交
localhost:8000/#/admin/userManager
您好，管理员
用户管理
屏蔽设置
输入关键词
xuan
锁定
dd
锁定
peng
锁定
asd
解锁

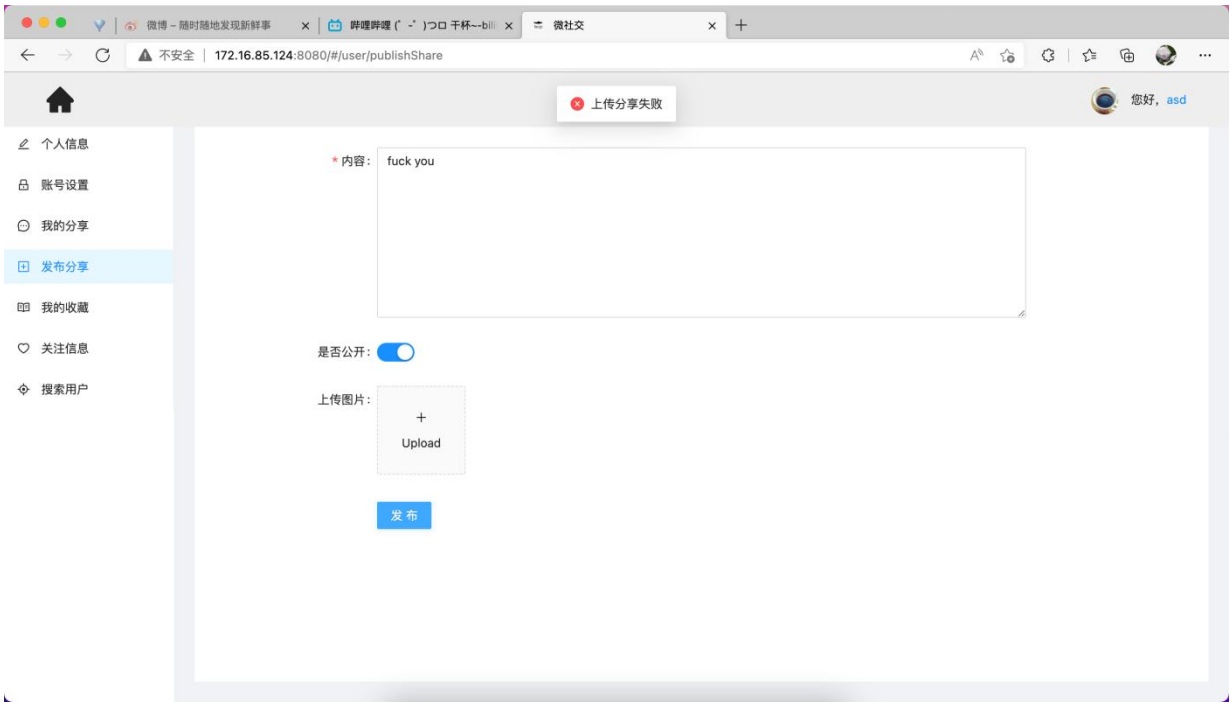
该账户的分享不可见


微博 – 随时随地发现新鲜事
微社交
localhost:8000/#/allShares
输入关键字
注册
登录
全部
最新
点赞排行
1
最受欢迎
No Data
最新分享
No Data

4.17. 限制词管理

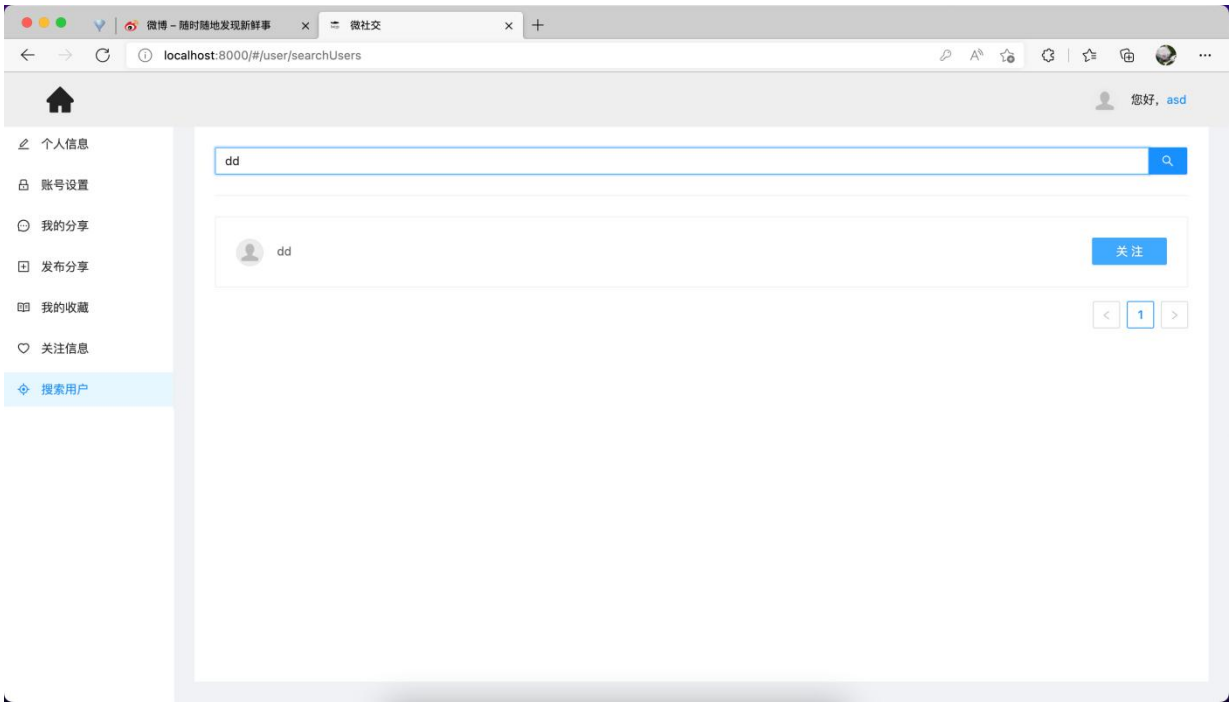


4.18. 限制词检测

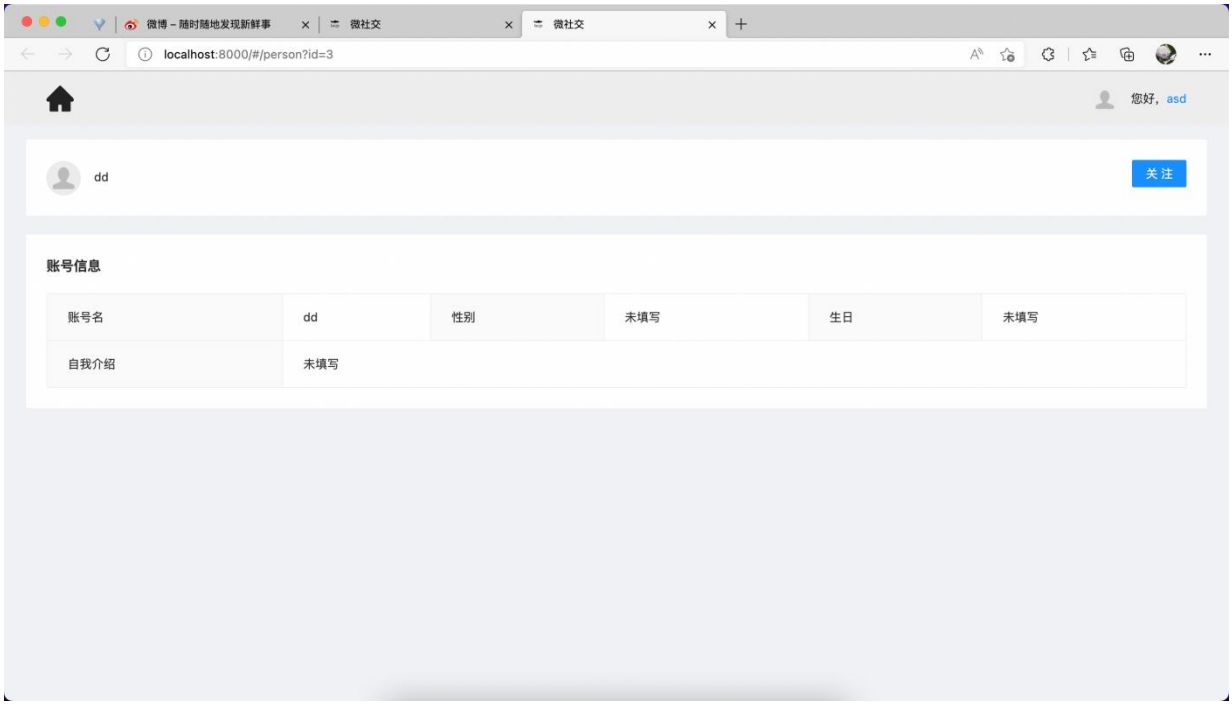


当上传有限制词的分享时是分享不出去的

4.19. 关注用户

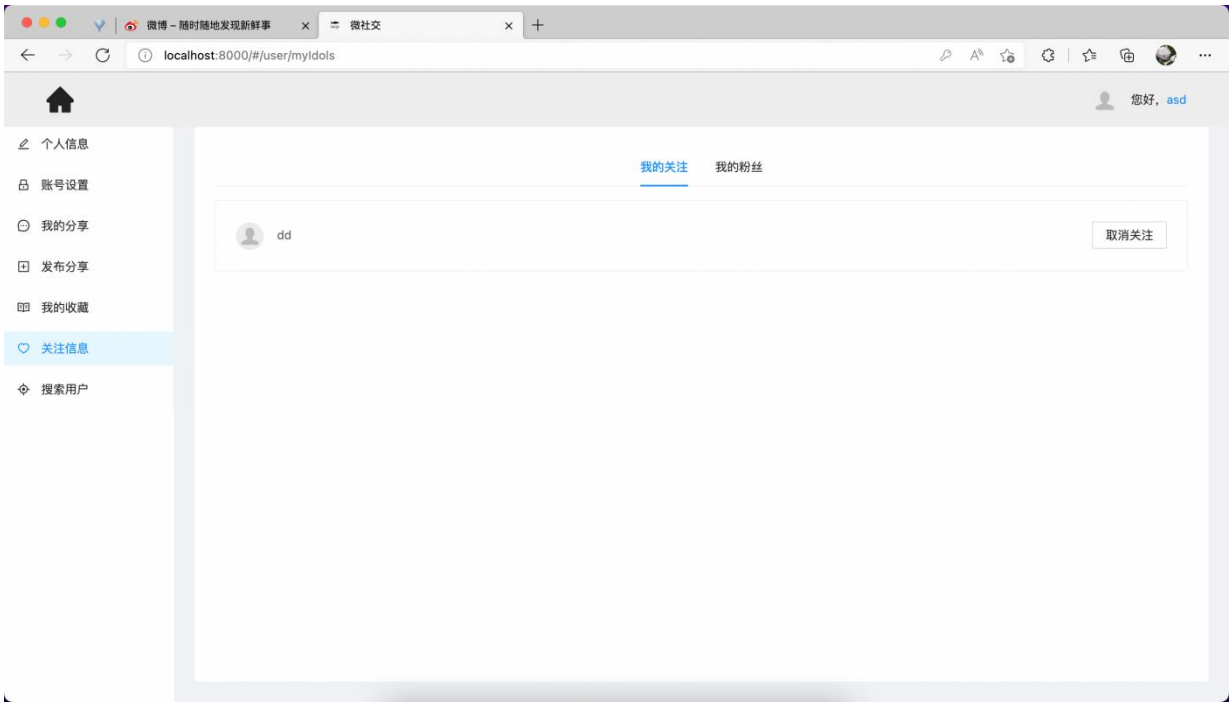


可以在搜索用户出关注用户

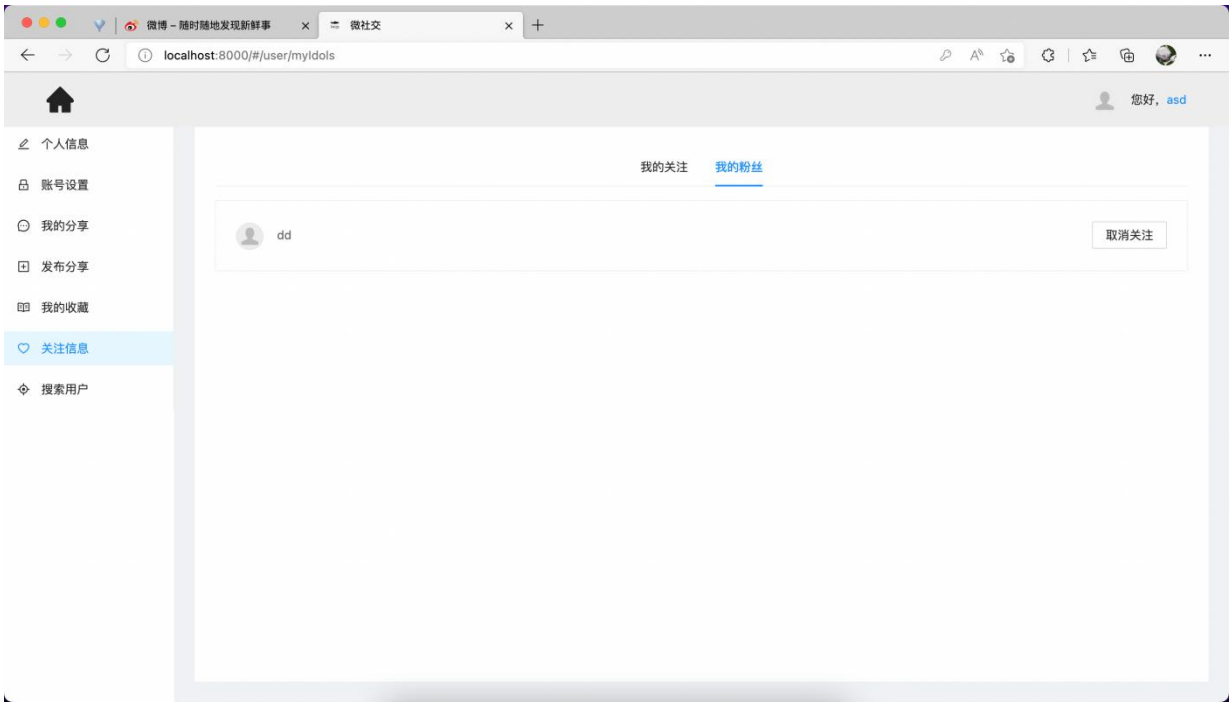


也可以在分享列表中点击作者头像进入用户界面在点击关注

4.20. 我的关注

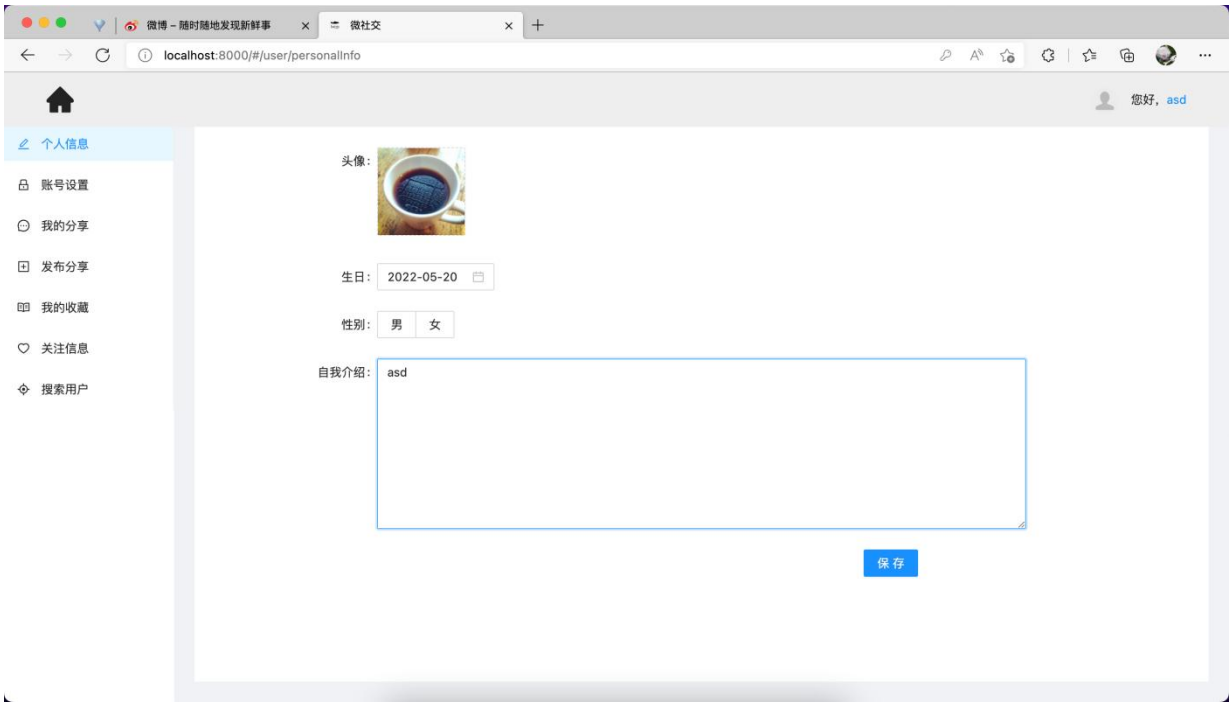


4.21. 我的粉丝

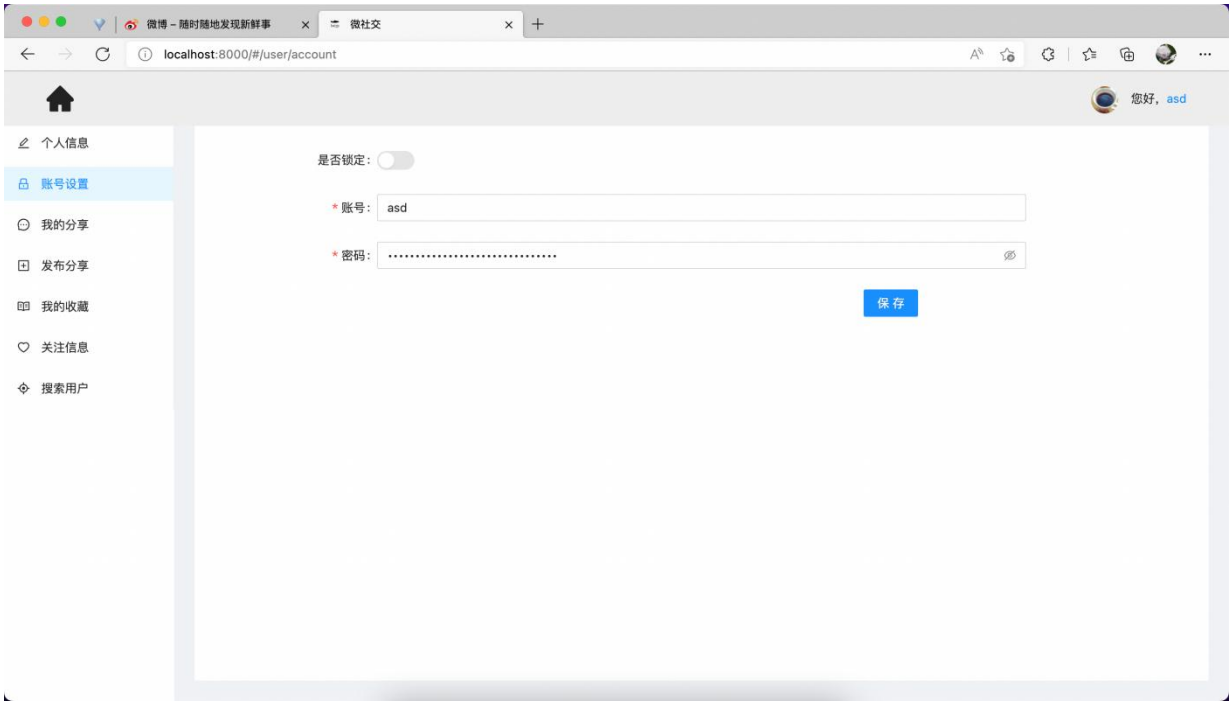


我与粉丝互相关注

4.22. 用户设置

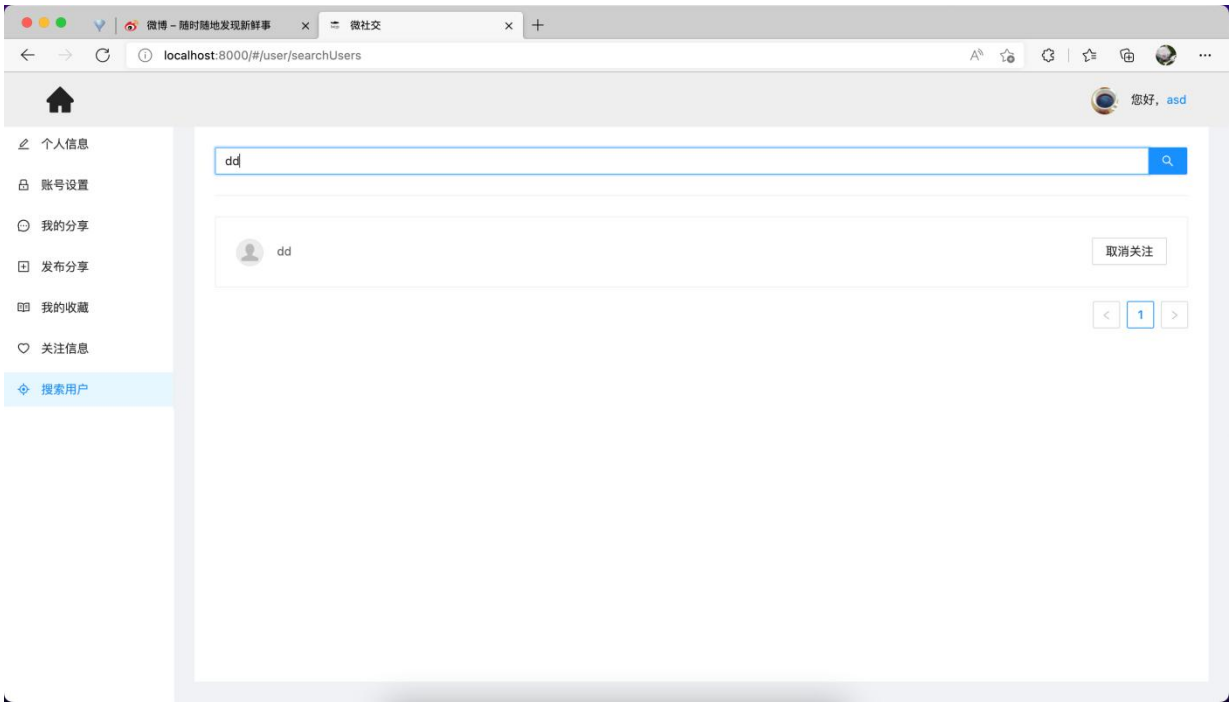


个人信息设置

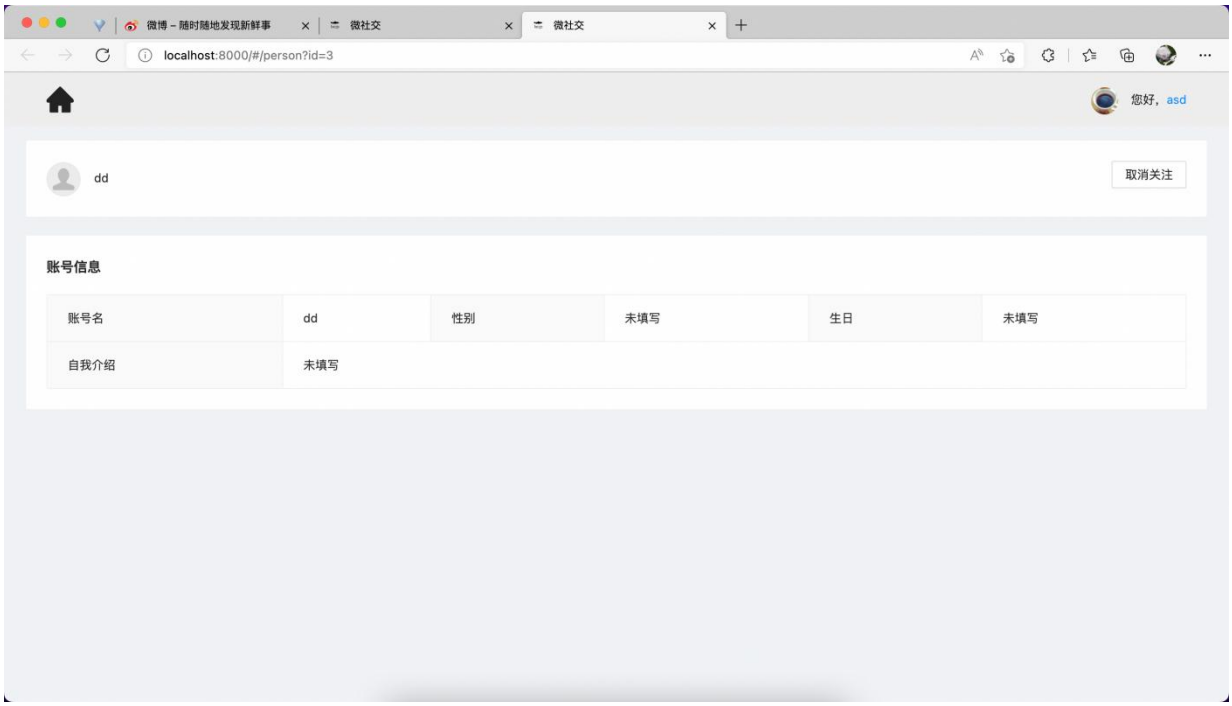


账号信息设置

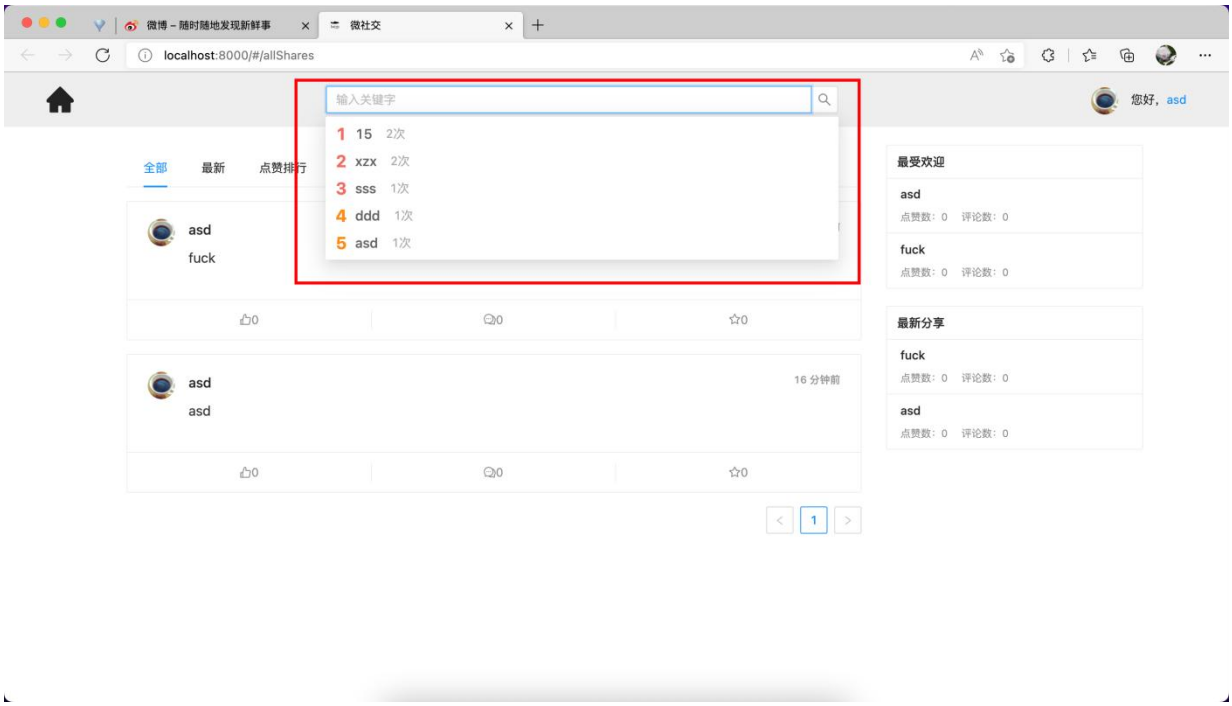
4.23. 搜索用户



4.24. 查看用户主页

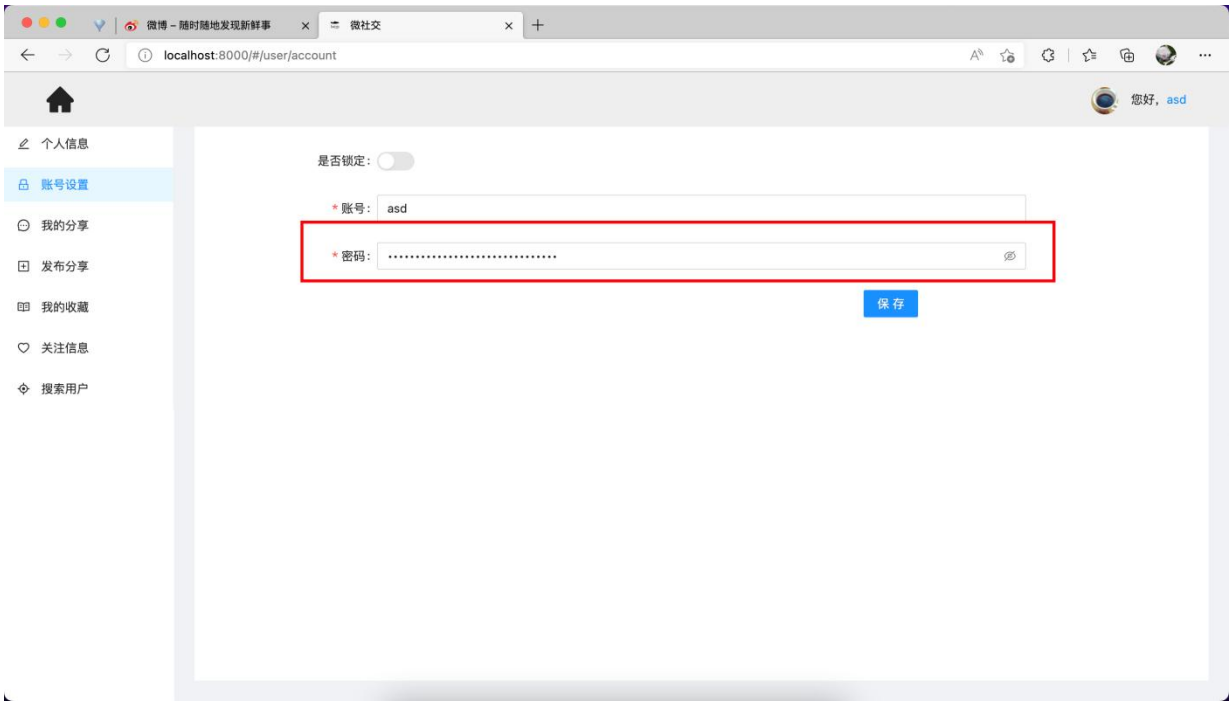


4.25. 热搜榜



显示热搜词条

4.26. 密码加密



原密码为“asd”，现已被加密

5. 项目总结

5.1. 项目实现的功能

5.1.1. 显示分享列表

以分页的形式显示分享列表。列表中的每个分享要展示分享的内容、图片，发布分享用户的用户名、头像，分享发布的时间，分享的点赞数、评论数和收藏数，以及当前用户是否已点赞收藏该分享。点击分享内容跳转到分享详情页面，点击分享图片展示大图，点击用户头像可跳转到用户主页。根据用户是否已点赞、收藏该分享，点赞按钮、收藏按钮以不同颜色显示。点击评论按钮可展示用户评论列表，再次点击可将用户评论列表折叠。点击评论中的查看回复按钮可查看该评论下的回复。

5.1.2. 搜索分享

在搜索框中输入内容，点击搜索按钮或按下回车键，以字符串模糊匹配的方式对分享的内容进行搜索，并将匹配的分享以分享列表的形式分页展示。

5.1.3. 点赞榜

按照点赞数降序展示分享列表。

5.1.4. 最新发布

按照发布时间最新的顺序展示分享列表。

5.1.5. 查看分享

点击分享列表中分享的内容进入分享详细页面。

5.1.6. 登录/注册

点击注册按钮进入注册页面，用户输入用户名、密码、确认密码，若用户名重复或确认密码与密码不匹配，则不允许注册，若合法，则点击注册按钮进行注册。点击登录按钮进入登录页面，输入用户名和密码进行登录，用户名存在且密码正确则登录成功。

5.1.7. 点赞分享

注册用户可点赞分享。分享的点赞按钮根据用户是否已点赞以不同的颜色展示。若未点赞，点击点赞按钮对分享点赞，若已点赞，点击点赞按钮对分享取消点赞。

5.1.8. 收藏分享

注册用户可收藏分享。分享的收藏按钮根据用户是否已收藏以不同的颜色展示。若未收藏，点击收藏按钮收藏分享，若已收藏，点击收藏按钮对分享取消收藏。

5.1.9. 评论分享

注册用户可评论分享。用户点击分享的评论按钮后，展示评论输入框，用户在输入

框中输入评论内容，点击发表按钮发表评论。新发表的评论实时展示在评论列表中。每条评论需要展示出评论的内容，发布评论的用户的用户名、头像，评论发布时间以及回复数。

5.1.10. 回复评论

注册用户可回复评论。用户点击评论中的回复按钮后，展示回复输入框，用户在输入框中输入回复内容，点击提交按钮提交回复。实时更新评论的回复状态。每条回复需要展示出回复的内容，提交回复的用户的用户名、头像，回复提交时间。

注册用户可对回复进行回复。用户点击回复中的回复按钮进行回复。与对评论进行回复不同的是，对回复的回复在回复的内容前会显示“回复@xxx”，以明确所回复的用户。

5.1.11. 我的分享

以分页分享列表的形式展示当前用户发布的分享。包含分享内容、用户名、用户头像、发布时间信息。点击详情进入分享详情页面，点击编辑按钮展示分享编辑页面、点击删除按钮删除已发布的分享。

5.1.12. 发布分享

注册用户可发布分享。进入分享发布页面，可输入分享内容、设置分享是否公开、以及上传图片，已选择的图片将会展示出来，每个分享最多可有 6 个图片。点击发布按钮发布分享。

5.1.13. 修改分享

在分享修改页面中，用户可对分享的内容、分享是否公开、以及分享的图片进行编辑，点击提交按钮确认修改。

5.1.14. 分享状态

分享的状态分为“仅自己可见”和“公开”，在发布分享时可进行设置，在分享发布后，可在修改分享页面中进行修改。

5.1.15. 我的收藏

以分页分享列表的形式展示当前用户收藏的分享。包含分享内容、用户名、用户头像、发布时间信息。点击详情进入分享详情页面，点击删除按钮可取消收藏。

5.1.16. 注册用户管理

管理员可以禁止或重新允许一个注册用户进行分享活动。处于禁止状态的注册用户的分享均不可见。

5.1.17. 限制词管理

管理员可对限制词进行管理，可添加或删除限制词。

5.1.18. 限制词检测

在用户发布分享或修改分享内容前，对分享的内容进行限制词检测，若分享内容中存在限制词，则不允许用户发布或修改分享内容。

5.1.19. 关注用户

注册用户可关注用户。进入其他用户的主页，根据用户是否已关注，关注按钮显示为“关注”或“取消关注”，点击按钮可关注用户或取消关注用户。

5.1.20. 我的关注

注册用户可查看我的关注列表，列表中展示当前用户已关注用户的用户名和头像，点击可进入该用户的个人主页，点击列表中的取消关注按钮可取消对该用户的关注。

5.1.21. 我的粉丝

注册用户可查看我的粉丝列表，列表中展示当前用户粉丝的用户名和头像，点击可进入该用户的个人主页，点击列表中的关注按钮可关注该用户。

5.1.22. 用户设置

注册用户可对个人信息和账号进行设置。用户可设置头像、修改生日、性别以及自我介绍，用户可修改用户名和密码。

5.1.23. 搜索用户

在搜索框中输入用户名，点击搜索按钮或按下回车键，以字符串模糊匹配的方式对用户进行搜索，并将匹配的用户以列表的形式分页展示。

5.1.24. 查看用户主页

进入用户主页，可查看用户头像、用户名、性别、生日和自我介绍，可以点击按钮关注或取消关注用户。

5.1.25. 热搜榜

点击分享搜索框，将展示当前的热搜榜，包含热搜关键词和被搜索次数，点击热搜榜的关键词，将按照关键词进行搜索。热搜榜每天定时刷新清零，重新累计。

5.1.26. 密码加密

用户注册时输入的密码在存入数据库前进行 MD5 加密，避免用户密码以明文的形式存入数据库。

5.2. 改进空间

5.2.1. 后端

项目在安全性方面存在改进空间。由于未使用安全框架，用户的验证、授权均使用

session 实现。这种方式安全性不高，由于 session 是依赖于 cookie 实现的，这就存在跨站点请求伪造的风险。并且在 controller 中需要从 session 获取用户信息，然后对用户是否已登录以及登录用户的权限信息进行判断，验证授权的耦合度较高，并且代码冗余较多。

5.2.2. 前端

因为使用的是 vue3，所以绝大部分逻辑代码都被拆分到 hooks 中，其中不乏很多的响应式数据，这些数据本应该在各个组件中各有一份独立的副本的但却因过于追求将业务逻辑分散到 hooks 中导致很多的组件共用同一份响应式数据。好在项目规模不大，各组件间的数据串联可以通过追加附加属性解决但也造成了代码冗余和臃肿的数据通信。

5.3. 个人心得体会

5.3.1. 彭翊轩

在本次现代软件开发技术期末考核项目中负责后端。本次的项目采用前后端分离的架构模式。相比于以往的 web 项目，前后端耦合度更低，后端能够更专注于对后台数据的处理。对于前端的页面和渲染，后端不需要过度关注，只需要根据接口文档制定的接口规范进行开发，通过暴露接口的方式为前端提供数据。后端采用的是分层的体系结构，通过分层达到后端各层的高内聚低耦合。其中，controller 层存放暴露给前端的后端接口，接收前端请求，调用 service 层的业务逻辑，并将处理后获得的数据通过 json 的形式响应给前端。service 层存放业务逻辑处理，并通过调用 mapper 层的方法与数据库交互。mapper 层对数据库进行数据持久化操作，为 service 层提供数据库增删查改的方法。entity 层存放实体类，实体类中的属性值与数据库中的字段对应。获取数据库中的数据时，将字段映射成为对象的属性。数据库设计是后端开发的关键，根据需求设计出良好的数据库，才能在整个后端开发过程中更高效地处理数据，并为后续的维护和拓展提供便利。

5.3.2. 刘润超

在本次的现代软件开发技术课程设计实验中我负责前端。相比于更早之前的数据库课设只是做了一个后台管理系统来说，本次系统的复杂度要远超前者，并且人数更少。而且本人也勇于尝新使用了还未曾使用的新的框架即 vue3 和 ant-design-vue，造成了此次课设工期较长，因为需要投入前期的学习时间。

不同于后台系统的增删改查各种数据只需要列出表格和编辑框就行的前端应用，本次课设需要实现更加复杂的分享页面，页面中有多条分享、每条分享又有多条评论、评论中又有多条相互间的回复这样的三级结构。为了稳定简洁高效地完成功能我真正利用到了 vue 带来的组件化能力，将分享的每一里层结构都封装到一个独立的组件中从而专心地处理本层次的业务逻辑，到里层次时又是同样的组件设计。着实享受到了 vue 的组件话、模块化带来的好处，各组件之间耦合度比起不使用框架的时候显著降低。同时因为采用 vue3 的组合式 API 又进一步将各种业务逻辑聚集在专门的模块中更加提高了内聚性和复用程度。为后续的维护和扩展带来诸多便利。