应用基础实践二

课程设计

|  |  |
| --- | --- |
| 题目名称 | 应用基础实践二 |
| 姓名 |  |
| 学号 |  |
| 专业 |  |
| 班级 |  |
| 指导教师 |  |
| 完成时间 |  |

目录

[第一章 问题描述 1](#_Toc185767442)

[（一）实验目的 1](#_Toc185767443)

[（二）问题描述 1](#_Toc185767444)

[（三）实验环境 1](#_Toc185767445)

[第二章 需求分析 2](#_Toc185767446)

[（一）功能需求 2](#_Toc185767447)

[（二）界面需求 2](#_Toc185767448)

[（二）技术需求 3](#_Toc185767449)

[第三章 概要设计 4](#_Toc185767450)

[（一）设计思想 4](#_Toc185767451)

[（二）概要设计 5](#_Toc185767452)

[1．微博客户端功能 5](#_Toc185767453)

[2．微博Web服务器功能 5](#_Toc185767454)

[3．Web服务器后台数据库功能(MySQL) 6](#_Toc185767455)

[4．用户客户端登录功能 6](#_Toc185767456)

[（三）架构图 7](#_Toc185767457)

[第四章 详细设计 8](#_Toc185767458)

[（一）前端设计 8](#_Toc185767459)

[（二）后端设计 10](#_Toc185767460)

[（三）数据库设计 11](#_Toc185767461)

[第五章 程序模块功能 12](#_Toc185767462)

[（一）前端设计 12](#_Toc185767463)

[（二）后端设计 12](#_Toc185767464)

[（三）包图 14](#_Toc185767465)

[后端： 14](#_Toc185767466)

[前端： 16](#_Toc185767467)

[（四）关键功能设计 18](#_Toc185767468)

[1．用户登录过程 18](#_Toc185767469)

[2．密码存储方案 22](#_Toc185767470)

[3．数据传输方案 22](#_Toc185767471)

[4．分页请求设置 23](#_Toc185767472)

[5．多线程同步的图片请求与下载 27](#_Toc185767473)

[6．图片上传事件的响应 29](#_Toc185767474)

[第六章 程序调试 31](#_Toc185767475)

[（一）调试方法 31](#_Toc185767476)

[（二）遇到的主要问题 31](#_Toc185767477)

[**1. SpringBoot项目编译报错** 31](#_Toc185767478)

[**2. 使用JSONObject原生java类解析JSON太麻烦** 32](#_Toc185767479)

[**3. 图片的下载与图片的加载没有同步，导致加载失败** 32](#_Toc185767480)

[**4. 图片上传到服务器后，因服务器缓存导致暂时无法请求** 32](#_Toc185767481)

[**5. 图片请求后因缓存过小下载失败** 33](#_Toc185767482)

[第七章 程序测试 34](#_Toc185767483)

[（一）⽤⼾登录测试 34](#_Toc185767484)

[（二）微博发表和评论测试 40](#_Toc185767485)

[第七章 心得体会 46](#_Toc185767486)

[（一）课程知识理解 46](#_Toc185767487)

[（二）工程思维 46](#_Toc185767488)

[（三）代码能⼒ 46](#_Toc185767489)

[（四）查阅官⽅⽂档能⼒ 47](#_Toc185767490)

[（五）软件测试 47](#_Toc185767491)

[（六）坚持不懈的意志 47](#_Toc185767492)

[（七）全栈开发经验 48](#_Toc185767493)

[附录 代码清单 49](#_Toc185767494)

[（一）前端 50](#_Toc185767495)

[（二）后端 52](#_Toc185767496)

[附录 参考文献 55](#_Toc185767497)

第一章 问题描述

（一）实验目的

通过指导学生上机实践，让学生学会综合运用所学过的《计算机程序设计基础》、《数据结构》、《算法》、《Java语言与系统设计》、《数据库》、《Web技术》、《移动应用开发》等课程的基础知识，从而能够熟练掌握开发市面上比较流行的移动应用开发的基本技能，设计和开发出具有一定规模的Android客户端应用+JSP（PHP）服务器端编程+MySQL数据库的典型移动互联网应用。

（二）问题描述

通过使用Android、JSP（或PHP）和MySQL进行编程，开发一款完整的微博应用。学生需要实践Android开发技能，构建微博客户端；使用JSP或PHP开发微博Web服务器；将MySQL作为Web服务器的后台数据库。具体要求包括：

1) 可以发表微博、查看微博及微博列表等功能。

2) 可以发表评论功能。

3) 可以实现用户客户端登录功能。

（三）实验环境

前端：Android Studio Ladybug | 2024.2.1 Patch 3

后端：IntelliJ IDEA 2024.1.1 (Ultimate Edition)

数据库：MySQL

Windows 11 10.0.22631 版本 22631

java version 22.0.1 2024-04-16

第二章 需求分析

（一）功能需求

1．微博发布功能：用户能够通过客户端发布新的微博。

2．微博查看功能 ：用户可以查看自己和其他用户发布的微博。

3．微博列表显示：显示微博列表，包括微博的标题、发布时间等信息。

4．评论功能：用户可以对微博进行评论。

5．用户登录功能：实现用户在客户端的登录功能，保障用户隐私和账户安全。

6．点赞功能：用户可以对微博进行点赞操作，以实现对内容的喜好和互动。

7．今日热榜功能：用户可以查看每天点赞数量最高的微博。

（二）界面需求

1 . 登录界面 ：提供用户登录的界面，包括用户名、密码输入框以及登录按钮。

2. 微博发表界面：用户可以在客户端输入文本、上传图片等内容，通过一个直观的界面完成微博的发表。

3．微博列表界面：展示用户及关注用户的微博列表，以时间线的形式呈现。

4．微博详情界面：当用户点击某一条微博，跳转至微博的详细内容，包括评论、点赞等操作。

6．点赞按钮和状态显示：在微博列表和详情界面中，显示点赞按钮，并实时更新点赞状态。

7．个人中心界面：用户可以查看自己的微博、个人信息等

8 今日热榜界面：用户可以查看每天点赞数量最高的微博。

（二）技术需求

1. 前端：Android应用开发，使用MVVM架构。
2. 后端：Spring Boot框架，支持REST API。
3. 数据库：MySQL存储用户、微博及评论数据。

第三章 概要设计

（一）设计思想

本实验采用分层设计思想，围绕微博应用的核心功能，结合前后端分离架构，以提高代码的模块化程度和系统的可维护性。具体实现上，前端和后端分别采用当前主流的技术栈（Android + Jetpack Compose + MVVM；Spring Boot + REST API），通过标准化的接口进行通信，确保系统的扩展性和可移植性。

为了提升系统的性能和用户体验，数据库设计充分考虑了数据量增长后的查询效率，前端则通过分页加载与懒加载策略优化界面响应速度。系统安全性方面，采用多重措施（如密码加密存储、HTTPS通信、权限控制等），保障用户数据的安全和隐私。

前端采用MVVM（Model-View-ViewModel）架构，将数据、业务逻辑和视图分层处理：Model 层：负责定义数据结构，如用户信息、微博内容、评论数据等，并调用后端接口进行数据交互；View 层：通过Jetpack Compose构建用户界面，使用响应式设计实现界面动态更新；ViewModel 层：负责管理界面的状态和业务逻辑，通过LiveData等组件实现数据驱动的UI更新，确保代码清晰、易于维护。

后端采用经典的MVC（Model-View-Controller）模式：Model 层：定义与数据库表对应的实体类，如用户、微博、评论等，并通过ORM框架（如JPA）与数据库交互；View 层：以RESTful API形式为前端提供数据服务，包括用户管理、微博管理、评论管理和点赞管理等接口；Controller 层：将用户的请求分发到具体的Service层处理逻辑，并返回处理结果。

（二）概要设计

系统分为三层结构：前端、后端和数据库，各层的职责如下：

* 前端：负责用户界面的展示与操作，通过API与后端交互。
* 后端：处理业务逻辑和数据管理，并通过ORM与数据库通信。
* 数据库：存储微博应用的核心数据，并提供高效的增删改查能力。

1．微博客户端功能

（1）用户注册与登录：实现用户在客户端的注册与登录功能，确保安全性和用户身份验证；

（2）发表微博：提供用户在客户端上发布微博的功能，包括文字、图片等多种形式的微博内容；

（3）查看微博：允许用户在客户端上查看自己和其他用户发布的微博；

（4）发表评论：允许用户在客户端上对微博进行评论；

（5）点赞功能：实现用户对微博的点赞功能，保持点赞状态同步。

（6）微博列表：实现展示微博列表，支持按时间、热度等方式排序。

（7）账号认证：通过安全的登录机制，确保用户的账号信息安全；

（8）登录状态维护：保持用户登录状态的持久性，使用户在客户端上的操作得以持续。

2．微博Web服务器功能

（1）用户管理：管理用户的注册信息，包括用户账号、密码等；

（2）微博管理：处理用户在客户端上发布、删除微博的请求，保持微博数据的一致性；

（3）评论点赞管理：处理用户对微博的评论与点赞，包括发布评论、微博点赞等功能。

（4）图片下载管理：处理用户对微博图片、头像的请求

3．Web服务器后台数据库功能(MySQL)

（1）用户信息存储：在数据库中存储用户的账号信息，确保用户注册和登录的正常进行；

（2）微博数据存储：将用户发布的微博内容存储在数据库中，支持对微博的增删改查操作；

（3）评论数据存储：存储用户对微博的评论信息，包括评论内容、评论者等；

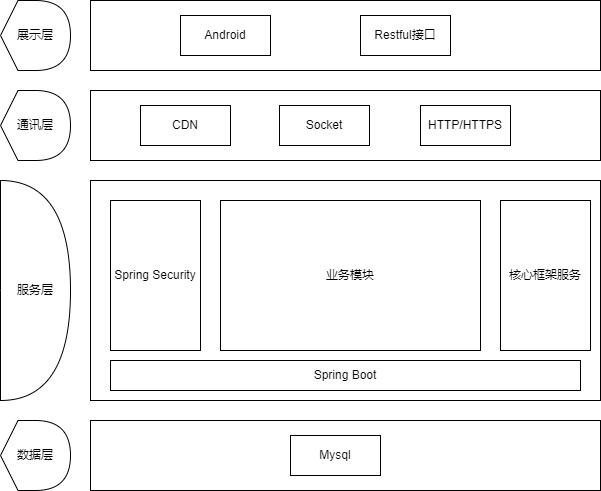
（4）点赞数据存储：存储用户对微博的点赞信息，包括点赞数量、点赞者等。

4．用户客户端登录功能

（1）账号认证：通过安全的登录机制，确保用户的账号信息安全；

（2）登录状态维护：保持用户登录状态的持久性，使用户在客户端上的操作得以持续。

（三）架构图



第四章 详细设计

（一）前端设计

1. entity 包

职责：定义业务模型，与应用中的核心数据结构对应。

包含类：

* Blog：博客实体类，存储博客相关数据。
* Comment：评论实体类，存储用户评论信息。

- User：用户实体类，存储用户相关信息。

2. ui 包

职责：处理用户界面相关的逻辑。

子包结构：

Activity：包含所有 Activity 类，负责管理应用的各个页面和用户交互。

* AddBlogActivity：添加博客页面。
* DetailsActivity：博客详情页面。
* LoginActivity：登录页面。
* ProfileActivity：用户个人资料页面。
* RegisterActivity：注册页面。
* SearchActivity：搜索页面。
* WelcomeActivity：欢迎页面。
* WhatshotActivity：热点推荐页面。

Adapter：包含适配器类，用于适配数据和视图。

* BlogAdapter：博客列表适配器。
* CommentAdapter：评论列表适配器。

Fragment：包含 Fragment 类，支持模块化 UI。

* HomeFragment：主页模块。

- WhatshotFragment：热点推荐模块。

3. utils 包

职责：提供工具类以支持项目通用功能。

包含类：

Network 包：处理网络通信相关的逻辑。

* BlogRepository：博客数据仓库，负责博客数据的增删改查。
* CommentRepository：评论数据仓库，负责评论数据的管理。
* FileDownloader：文件下载工具类。
* HttpGetRequest：GET网络请求工具类。
* HttpPostRequest：POST网络请求工具类。
* UserRepository：用户数据仓库，管理用户数据。

ImageUtils：图片处理工具类。

4. 资源目录

职责：存放应用的资源文件，包括图片、布局文件和字符串等。

子目录说明：

* drawable：存放应用的图片资源。
* layout：存放应用的 XML 布局文件。
* values 和 values-night：存放应用的字符串、主题和颜色定义。

- xml：存放 XML 配置文件。

（二）后端设计

1. Controller 层

职责：处理客户端的请求，并调用相应的 Service 层方法，返回处理结果。

包含类：

* BlogController：处理博客相关操作（如创建、获取博客）。
* CommentController：处理评论相关操作（如添加、获取评论）。
* LikeController：处理点赞功能（如点赞/取消点赞操作）
* UserController：处理用户管理功能（如用户注册、登录）。

2. DAO 层

职责：提供数据访问接口，定义操作数据库的方法。

包含子包及类：

* impl 包：包含具体 DAO 接口的实现。
* BlogDao：博客数据操作类。
* CommentDao：评论数据操作类。
* LikeDao：点赞数据操作类。

- UserDao：用户数据操作类。

3. Entity 层

职责：定义业务实体类，与数据库表结构一一对应。

包含类：

* Blog：博客实体。
* BlogResponse：用于返回博客数据的响应模型。
* Comment：评论实体。

- User：用户实体。

4. Service 层

职责：实现业务逻辑，调用 DAO 层完成数据操作。

包含子包及类：BlogService：博客相关业务逻辑。

* CommentService：评论相关业务逻辑。
* FileUploadService：文件上传处理逻辑。
* LikeService：点赞相关业务逻辑。

- UserService：用户相关业务逻辑。

（三）数据库设计

1. 用户表（Users）：存储用户基本信息，包括用户名、密码哈希、性别、自我介绍。

2. 微博表（Blogs）：存储微博内容、发布者ID、发布时间、是否有微博图片、点赞数量。

3. 评论表（Comments）：记录微博的评论信息，包括评论者ID、微博ID、评论内容、评论发布时间。

4. 点赞表（Likes）：存储微博的点赞记录，包括点赞者ID和微博ID。

第五章 程序模块功能

（一）前端设计

- 主界面模块：实现微博列表展示和跳转。

- 详情模块：显示微博详细内容及评论互动。

- 发布模块：支持文本及图片微博的创建。

（二）后端设计

程序采用 Spring Boot 框架实现，主要包含以下模块：

1. 入口模块：

类名：DemoApplication

功能：程序的启动入口。

使用 SpringApplication.run 启动 Spring Boot 应用。

1. 控制器层：

模块功能：负责处理 HTTP 请求，调用服务层提供的业务逻辑。

主要类：

* BlogController：处理博客相关操作（如创建、获取博客）。
* CommentController：处理评论相关操作（如添加、获取评论）。
* LikeController：处理点赞功能（如点赞/取消点赞操作）
* UserController：处理用户管理功能（如用户注册、登录）。
* 服务层：

模块功能：定义和实现核心业务逻辑。

主要类：

* 接口：BlogService
* 实现：BlogServiceImpl

1. 数据访问层：

模块功能：封装对数据库的操作。

主要类：

- impl 包：包含具体 DAO 接口的实现。

- BlogDao：博客数据操作类。

- CommentDao：评论数据操作类。

- LikeDao：点赞数据操作类。

- UserDao：用户数据操作类。

1. 实体层：

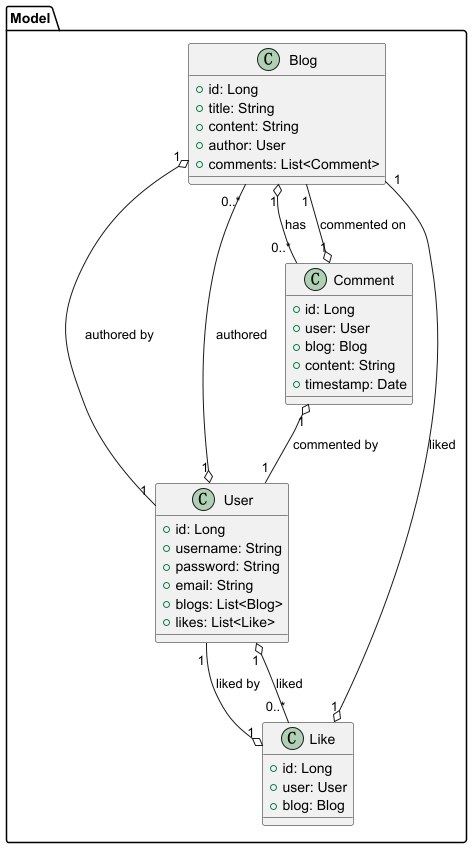
模块功能：定义数据库表结构和实体类。

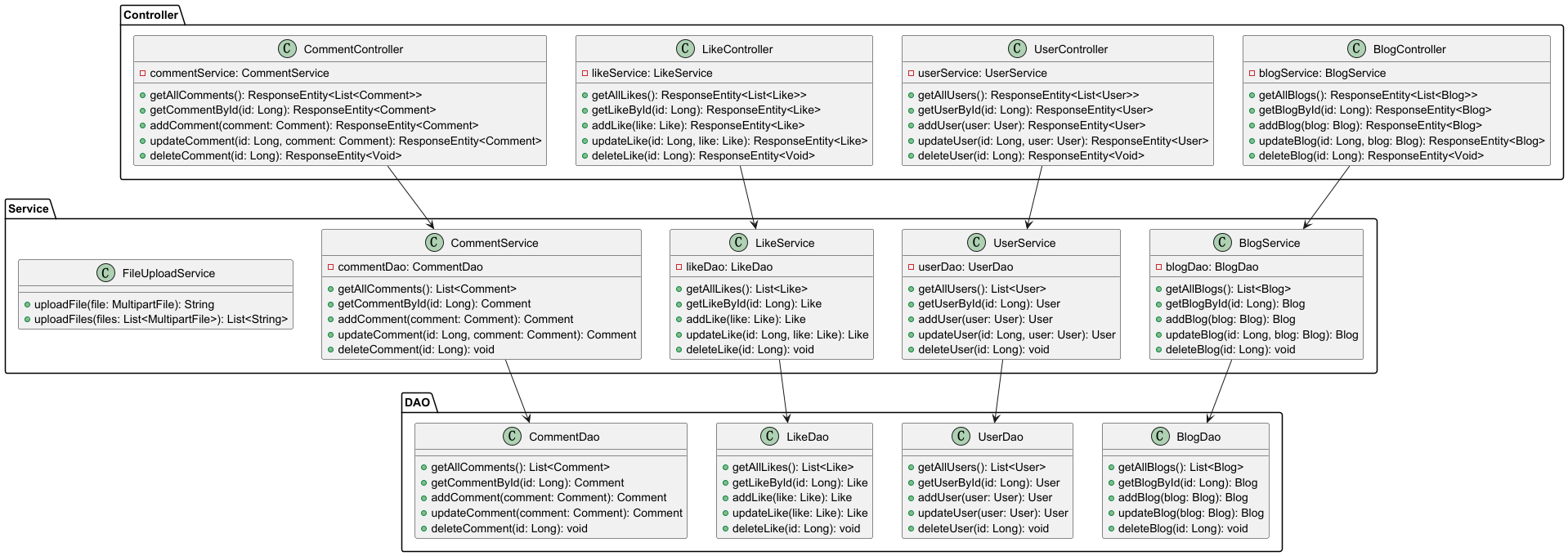
主要类：

* Blog：博客实体类，存储博客相关数据。
* Comment：评论实体类，存储用户评论信息。
* User：用户实体类，存储用户相关信息。

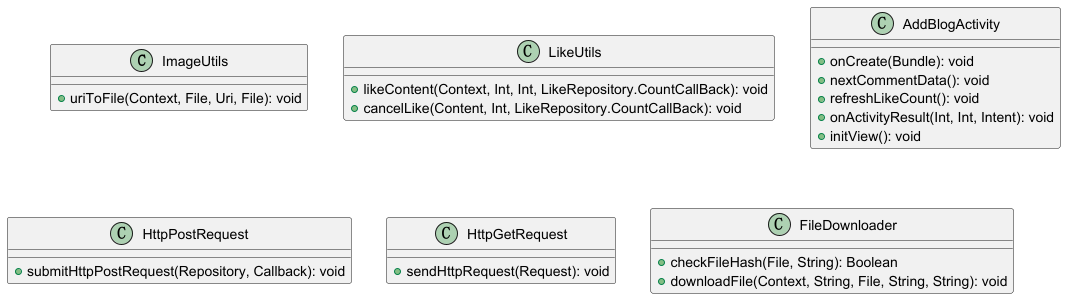
（三）包图

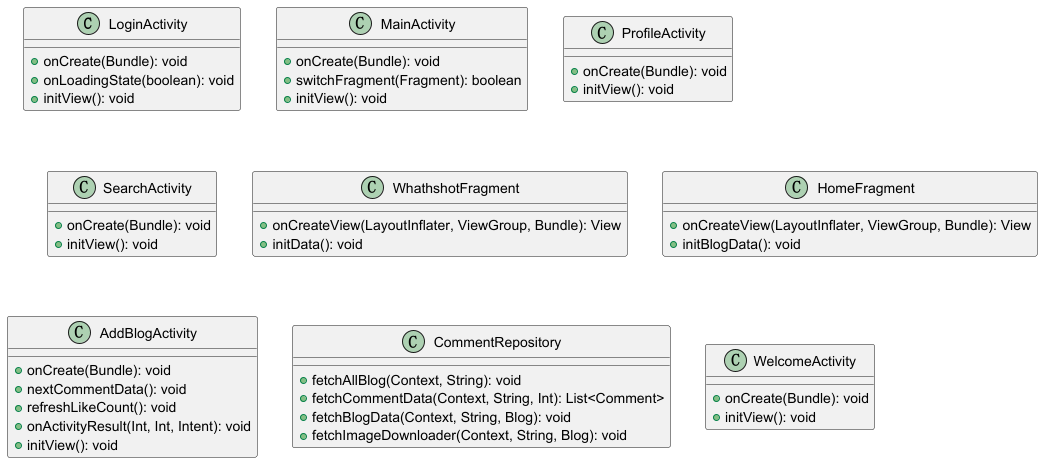
后端：

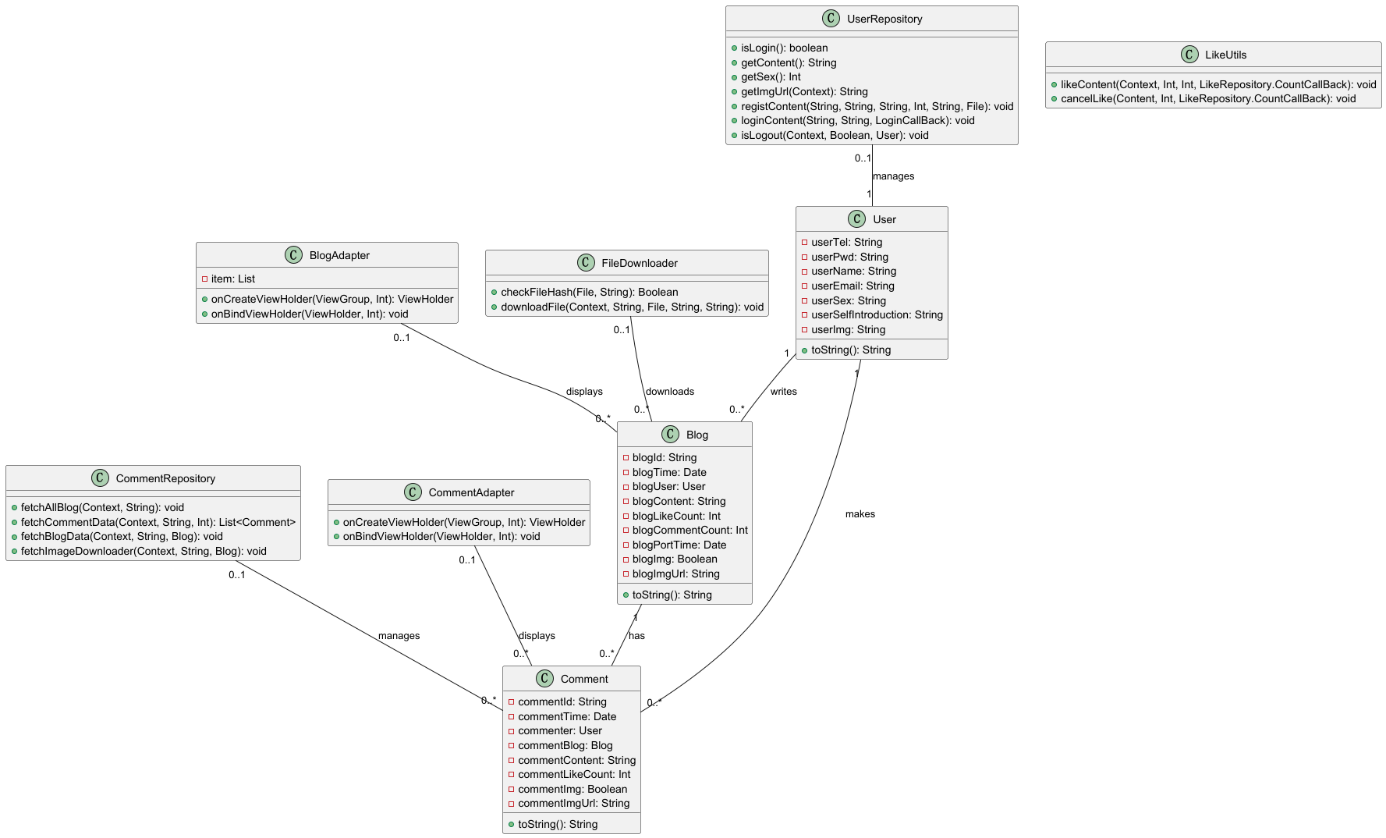




前端：







（四）关键功能设计

1．用户登录过程

欢迎界面 (WelcomeActivity)，首先显示一个倒计时提示。

* 倒计时结束后，WelcomeActivity 会检查用户是否已登录：
* 如果已登录，跳转到主界面 (MainActivity)。
* 如果未登录，跳转到登录界面 (LoginActivity)。
* 用户还可以在倒计时过程中点击跳过按钮，立即跳转到相应的页面。

public class WelcomeActivity extends AppCompatActivity {

private CountDownTimer countDownTimer;

private boolean isSkipped = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_welcome);

countDownTimer = new CountDownTimer(3000, 1000) {

@Override

public void onTick(long millisUntilFinished) {

skipButton.setText("跳过 " + (int) Math.ceil(millisUntilFinished / 1000.0));

}

@Override

public void onFinish() {

if (!isSkipped) {

navigateToMain();

}

}

}.start();

skipButton.setOnClickListener(v -> {

isSkipped = true;

countDownTimer.cancel();

navigateToMain();

});

}

private void navigateToMain() {

Intent intent;

if (UserRepository.isLoggedIn(this)) {

intent = new Intent(WelcomeActivity.this, MainActivity.class);

} else {

intent = new Intent(WelcomeActivity.this, LoginActivity.class);

}

startActivity(intent);

finish();

}

}

登录界面 (LoginActivity)：

* 在登录界面，用户需要输入用户名和密码，并点击登录按钮。
* 系统会验证输入的用户名和密码是否为空。如果为空，会显示错误提示。
* 如果用户名和密码有效，系统会发起登录请求。
* 根据登录请求的结果，禁用输入框和按钮，等待登录验证完成。
* 如果登录成功，跳转到主界面 (MainActivity)；如果登录失败，显示错误信息。

public class LoginActivity extends AppCompatActivity {

private EditText et\_username, et\_password;

private Button btn\_login;

private ProgressBar progressBar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

btn\_login.setOnClickListener(v -> {

String username = et\_username.getText().toString().trim();

String password = et\_password.getText().toString().trim();

if (username.isEmpty() || password.isEmpty()) {

Toast.makeText(LoginActivity.this, "用户名、密码不能为空", Toast.LENGTH\_SHORT).show();

return;

}

showLoadingState(true);

UserRepository.login(LoginActivity.this, "server\_url", username, password, new UserRepository.LoginCallback() {

@Override

public void onLoginSuccess() {

Toast.makeText(LoginActivity.this, "登录成功", Toast.LENGTH\_SHORT).show();

startActivity(new Intent(LoginActivity.this, MainActivity.class));

finish();

}

@Override

public void onLoginFailure(String error) {

Toast.makeText(LoginActivity.this, error, Toast.LENGTH\_SHORT).show();

}

});

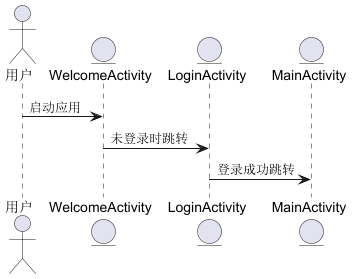
showLoadingState(false);

});

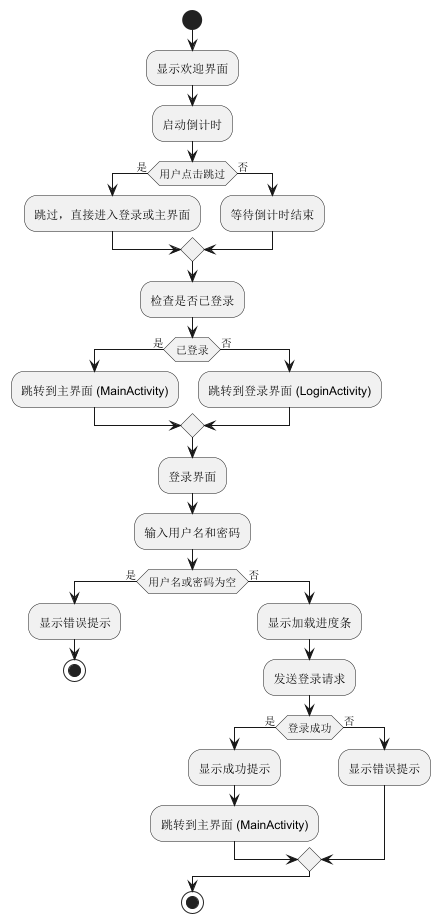
}

}

时序图：



流程图：



2．密码存储方案

为确保用户密码的安全性，不允许以明文形式存储密码。在密码验证过程中，系统仅对哈希值进行验证，以增加密码的保密性和防范明文密码泄露的风险。此安全措施有助于提高系统的整体安全性，确保用户敏感信息得到有效的保护。

以下为部分关键代码展示:

@Override

    public User login(String username, String password) {

        String passwordHashValue = DigestUtils.md5DigestAsHex(password.getBytes(StandardCharsets.UTF\_8));

        try {

            String sql = "select \* from t\_users where userName=? and userPasswordHashValue=?";

            return this.jdbcTemplate.queryForObject(sql, (resultSet, i) -> {

                User user = new User();

                user.setUserId(resultSet.getInt("userId"));

                user.setUsername(resultSet.getString("userName"));

                user.setUserSex(resultSet.getInt("userSex"));

                user.setUserSelfIntroduction(resultSet.getString("userSelfIntroduction"));

                return user;

            }, username,passwordHashValue);

        } catch (Exception e) {

            throw new InvalidPasswordException(e.getMessage());

        }

    }

3．数据传输方案

使用了 OkHttp 库中的异步请求方法来发送 HTTP GET 和 POST 请求，实现了在后台线程中发送 HTTP GET 和 POST 请求，避免了主线程阻塞。通过回调函数处理响应结果，确保了 UI 的流畅性和应用的响应性。此外，使用 synchronized 关键字确保了在特定场景下的线程安全性。

public class HttpGetRequest {

public static synchronized void sendOkHttpGetRequest(String address,okhttp3.Callback callback){

OkHttpClient client = new OkHttpClient();

Request request = new Request.Builder()

.get()

.url(address)

.build();

client.newCall(request).enqueue(callback);

}

}public class HttpPostRequest {  
 private static final MediaType JSON = MediaType.parse("application/json; charset=utf-8");

public static synchronized void okhttpPost(String url, RequestBody requestBody, okhttp3.Callback callback) {  
 OkHttpClient client = new OkHttpClient();  
 Request request = new Request.Builder()  
 .url(url)  
 .post(requestBody)  
 .build();  
 client.newCall(request).enqueue(callback);  
 }  
}

4．分页请求设置

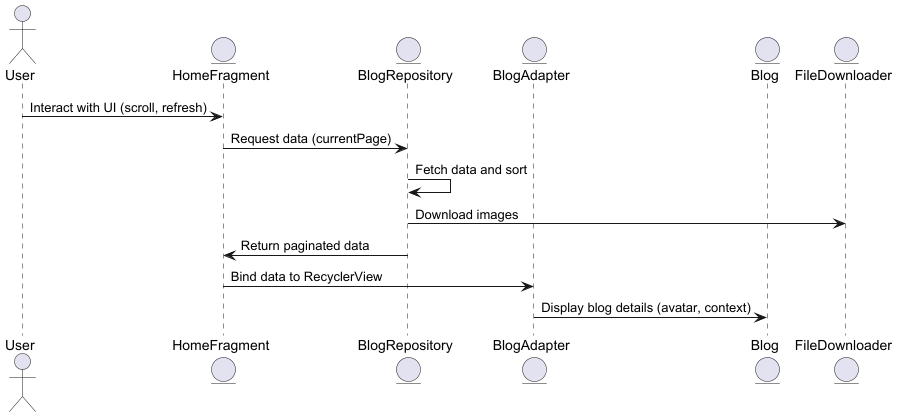
在项目中，数据分页的应用主要体现在评论加载和网页加载的场景中。数据分页的需求源于大量数据的分批加载，以提升系统性能和用户体验。在评论加载中，当微博或其他内容存在大量评论时，采用分页加载的方式可以避免一次性加载所有评论，从而减轻服务器和客户端的负担，实现快速展示。同样，在网页加载过程中，采用数据分页可以有效减少页面加载时间，降低带宽消耗，提高用户浏览网页的效率。

数据分页的实现通常基于后端系统提供的API，通过设置合适的参数，如页码和每页数据数量，来请求指定范围的数据。前端系统则根据用户的操作触发相应的请求，将数据分批加载到页面上，实现逐步展示的效果。在评论加载和网页加载过程中，数据分页的应用有效地平衡了系统性能和用户体验之间的关系，使系统能够高效处理大量数据，同时确保用户能够流畅地浏览和交互。

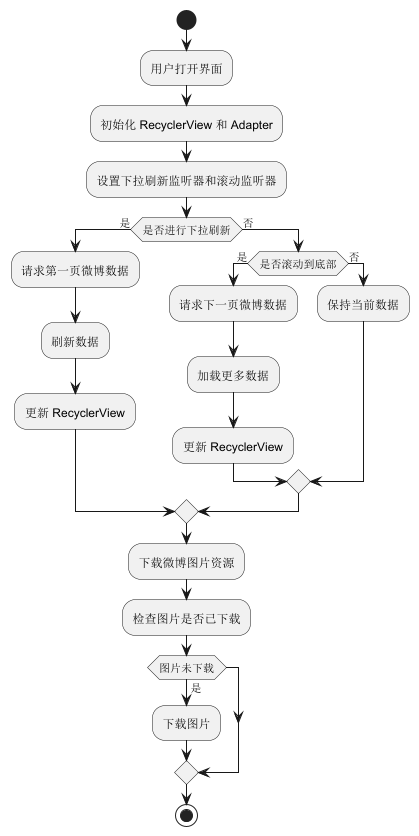
以下为部分关键代码展示：

public static List<Blog> fetchBlogData(Context context, String url, int page) {  
 if (page == 1) {  
 synchronized (lock) {  
 isDataLoaded = false;  
 fetchAllBlog(context, url);  
 while (!isDataLoaded) {  
 try {  
 lock.wait(); // 等待数据加载完成  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt();  
 Log.e("blog repository", "线程被中断", e);  
 }  
 }  
 }  
 }  
  
 List<Blog> blogList = new ArrayList<>();  
 synchronized (nowBlogList) {  
 for (int i = (page - 1) \* maxPageSize; i < Math.min(nowBlogList.size(), page \* maxPageSize); i++) {  
 blogList.add(nowBlogList.get(i));  
 }  
 }  
 Log.d("blog repository", "分页博客数据数量：" + blogList.size());  
 return blogList;  
}

时序图：



流程图：



5．多线程同步的图片请求与下载

在移动应用中，用户头像通常是从远程服务器下载的。如果直接在主线程（UI线程）进行文件下载或网络请求，会导致应用在下载过程中卡顿或无响应，这对用户体验是非常不利的。因此，必须将耗时操作放到后台线程中执行，并且确保UI的更新（如显示头像）在主线程中进行。应当避免在UI线程中进行耗时操作：通过在后台线程中处理文件下载任务，避免了UI线程阻塞，保持了应用的流畅性。确保UI线程更新：下载完成后，使用 Handler 将图片加载操作切换到主线程进行，保证图片能够正确显示。

File directory = getExternalFilesDir(Environment.DIRECTORY\_PICTURES);

Handler handler = new Handler(Looper.getMainLooper());

new Thread(() -> {

try {

// 下载文件

FileDownloader.downloadFile(

ProfileActivity.this,

getString(R.string.server\_url),

directory,

UserRepository.loggedInUser.toFileName(),

"avatars");

// 检查文件是否下载成功

File file = new File(directory, UserRepository.loggedInUser.toFileName());

if (file.exists() && file.length() > 0) {

Log.d("Glide", "文件下载成功: " + file.getAbsolutePath());

// 通知主线程加载图片

handler.post(() -> Glide.with(this)

.load(new File(directory, UserRepository.loggedInUser.toFileName())) // 用户头像 URL

.placeholder(R.drawable.null\_avatar) // 占位图

.error(R.drawable.null\_avatar) // 错误图

.into(siv\_avatar));

} else {

Log.e("Glide", "文件下载失败或文件大小为 0: " + file.getAbsolutePath());

}

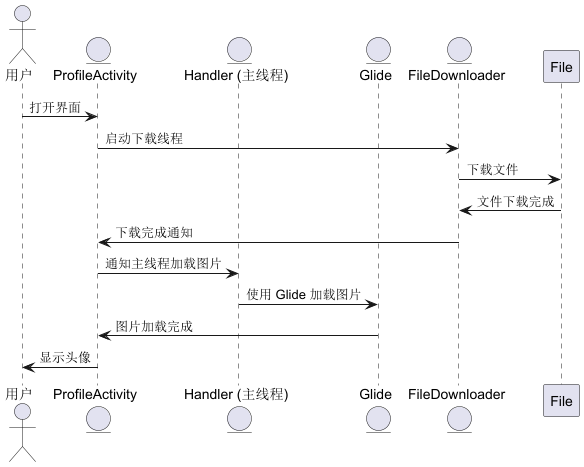
} catch (Exception e) {

Log.e("Glide", "文件下载或加载出错", e);

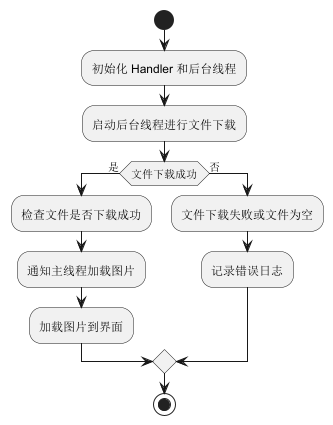
}

}).start();

时序图



流程图



6．图片上传事件的响应

当图片通过post请求上传到服务器后，Spring Boot 可能没有立刻意识到该文件已经被添加到目录中，导致请求时无法找到该文件。Spring Boot 的静态资源处理机制可能会缓存文件列表，在服务器首次启动时，它会加载这些文件。如果动态地上传文件而不重启服务器，可能就无法及时识别这些文件。可以在文件上传后，触发 Spring 的 ApplicationListener 或使用 @PostConstruct 来确保文件上传操作与静态资源的加载同步。这样可以确保文件上传后立即可供请求。

发布事件：当文件上传操作发生时，FileUploadServiceImpl 类会通过 publishFileUploadEvent 方法发布一个包含文件名的 FileUploadEvent 事件。

监听事件：FileUploadListener 类通过 @EventListener 注解监听 FileUploadEvent，当该事件发生时，监听器中的 onFileUpload 方法被调用，处理事件中的文件名数据。

@Service  
public class FileUploadServiceImpl implements FileUploadService {  
  
 private final ApplicationEventPublisher eventPublisher;  
  
 public FileUploadServiceImpl(ApplicationEventPublisher eventPublisher) {  
 this.eventPublisher = eventPublisher;  
 }  
  
 // 发布文件上传事件  
 public void publishFileUploadEvent(String fileName) {  
 FileUploadEvent event = new FileUploadEvent(this, fileName);  
 eventPublisher.publishEvent(event);  
 }  
}

@Component  
public class FileUploadListener {  
  
 @EventListener  
 public void onFileUpload(FileUploadEvent event) {  
 String fileName = event.getFileName();  
 System.out.println("File uploaded: " + fileName);  
 }  
}

public class FileUploadEvent extends ApplicationEvent {

private final String fileName;

public FileUploadEvent(Object source, String fileName) {

super(source);

this.fileName = fileName;

}

public String getFileName() {

return fileName;

}

}

第六章 程序调试

（一）调试方法

在前端程序的调试过程中，首先使用Logca输出关键变量的值，例如在文件下载的过程中。通过在文件下载操作前后输出日志，可以验证下载链接、目标文件路径、文件大小等是否符合预期。具体来说，在文件下载前，记录目标路径和下载URL；下载后，检查文件是否存在、文件大小是否正常等。

通过设置断点，逐步跟踪程序的执行流程，特别是关注多线程环境下的线程切换和 UI 更新的过程。设置断点的位置包括文件下载的代码段和 UI 更新的部分，通过调试工具的逐步执行功能，可以查看主线程和后台线程是否按预期切换，以及 UI 是否在下载完成后正确更新。例如，可以在下载完成后设置断点，确认是否执行了主线程的 UI 更新操作。

使用 try-catch 语句捕获可能发生的异常，并输出详细的堆栈信息，以帮助进一步定位问题。如果文件下载失败或者发生网络异常，通过捕获异常可以及时发现并记录问题。

（二）遇到的主要问题

**1. SpringBoot项目编译报错**

- 问题描述：在项目开发过程中，由于 JDK 版本、SpringBoot 框架版本以及所使用的第三方依赖版本不兼容，导致编译失败。常见问题包括类或方法的缺失、版本冲突等。

- 解决方案：根据 Spring 官方提供的兼容性文档，确认所用的 JDK 版本与 SpringBoot 版本是否匹配，删除本地缓存文件（如 .m2/repository）后重新下载依赖，确保没有旧版本干扰。

**2. 使用JSONObject原生java类解析JSON太麻烦**

- 问题描述：在解析复杂的 JSON 数据时，原生的 JSONObject 类需要手动逐层获取字段，代码繁琐且易出错。

- 解决方案：引入 Gson 库，使用 Gson 的封装方法处理复杂的嵌套结构，显著减少代码量并提高可读性。

**3. 图片的下载与图片的加载没有同步，导致加载失败**

- 问题描述：在开发中，图片的下载操作与加载操作未按预期顺序进行，导致图片还未下载完成就尝试加载，结果显示失败。

- 解决方案：使用多线程：将图片下载操作放在子线程中执行，避免阻塞主线程。使用线程同步工具CountDownLatch保证主线程在下载完成后加载图片。使用内置了缓存和异步下载机制的Glide 加载库。

**4. 图片上传到服务器后，因服务器缓存导致暂时无法请求**

- 问题描述：在图片上传后立即请求，但由于服务器的静态资源缓存未刷新，导致返回的仍是旧数据。

- 解决方案：配置服务器缓存策略，减少缓存时间，利用 ApplicationListener 监听文件上传事件，动态更新资源路径或刷新缓存。

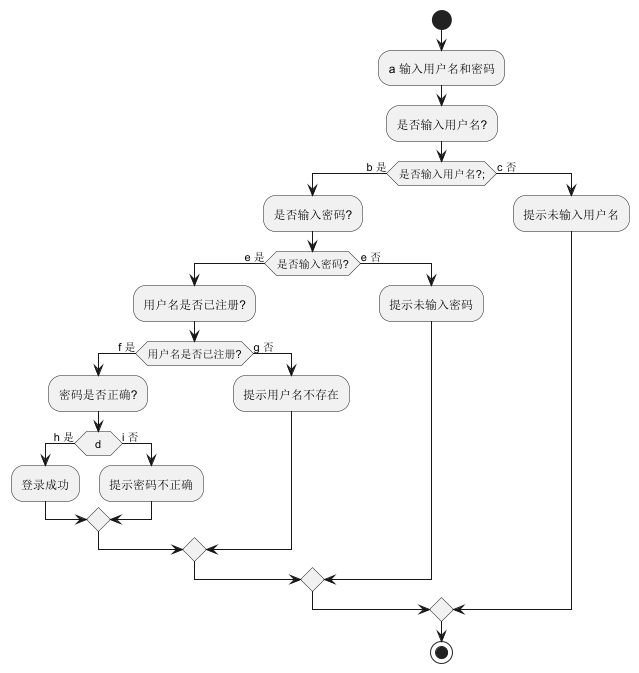
**5. 图片请求后因缓存过小下载失败**

- 问题描述：在解析复杂的 JSON 数据时，原生的 JSONObject 类需要手动逐层获取字段，代码繁琐且易出错。

- 解决方案：配置 OkHttp 的缓存，检查服务器和客户端的网络配置，确保连接超时时间和数据读取超时时间足够。

第七章 程序测试

（一）⽤⼾登录测试



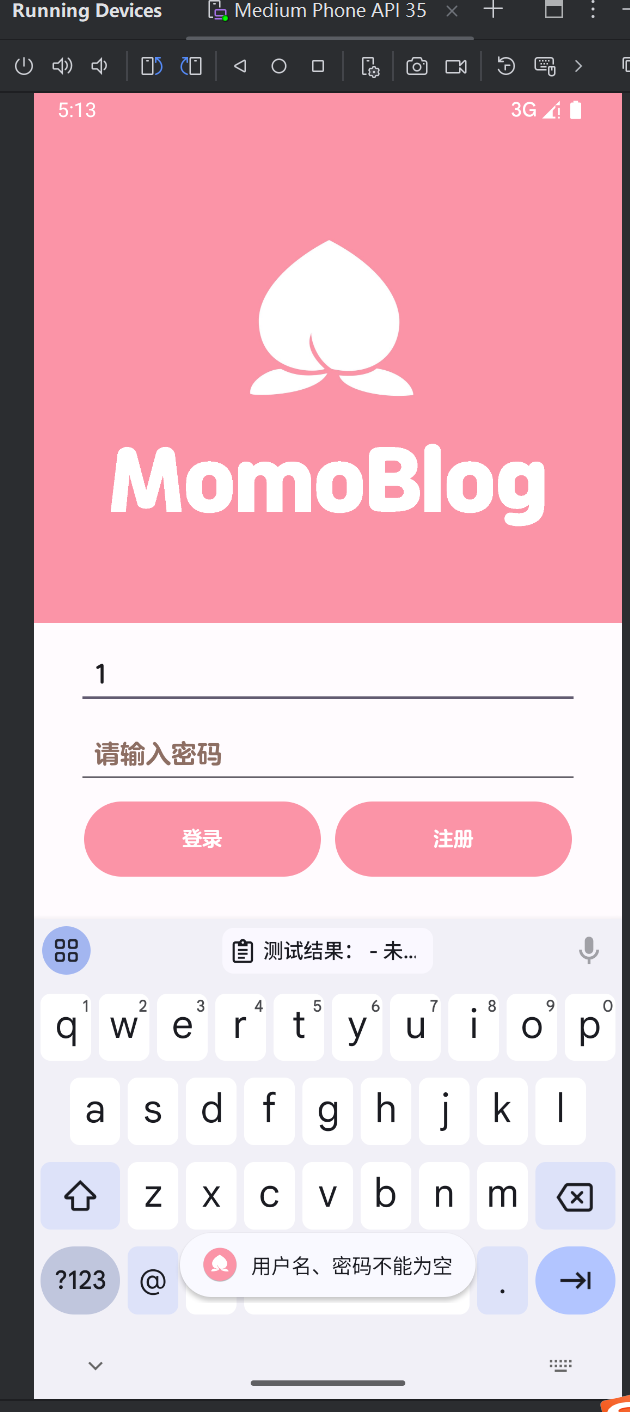


测试结果：

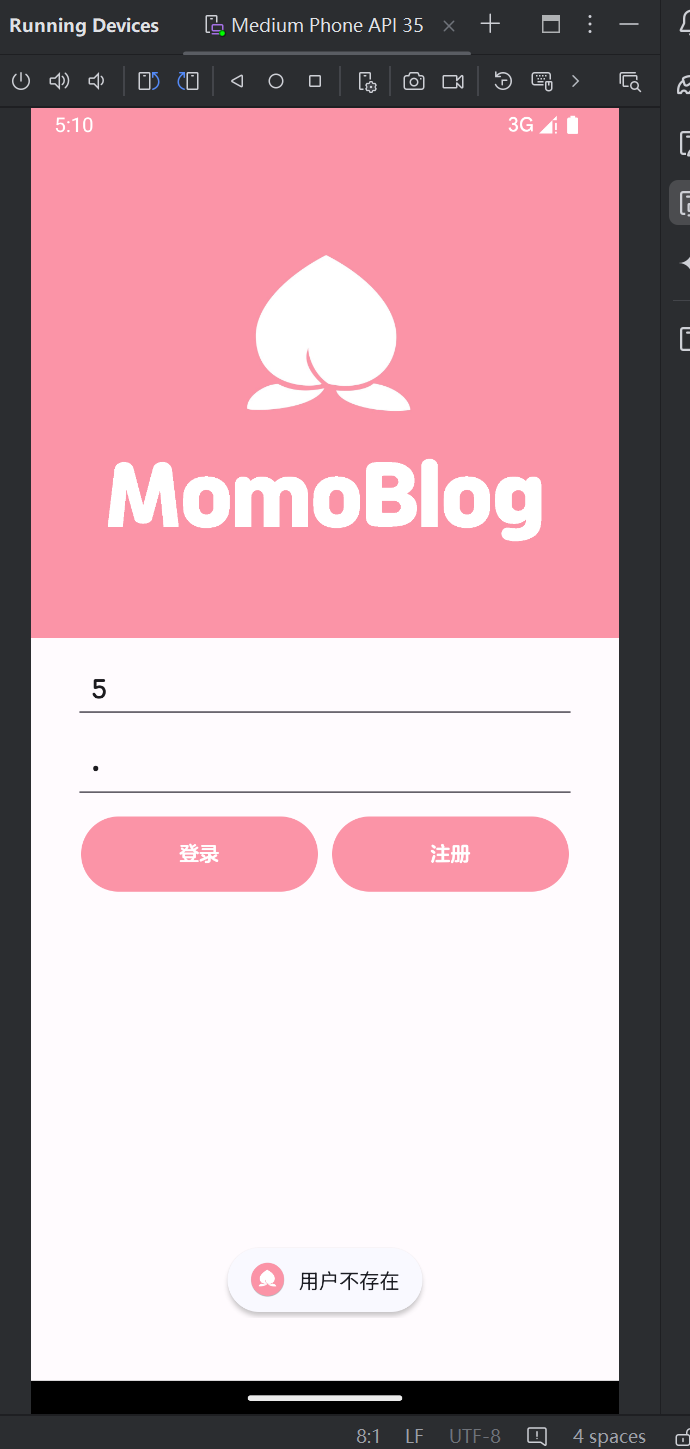
* 未输入用户名：



* 未输入密码：



* 未输入已注册的用户名：



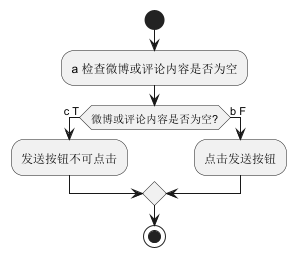
* 输入已注册的用户名和不正确的密码：

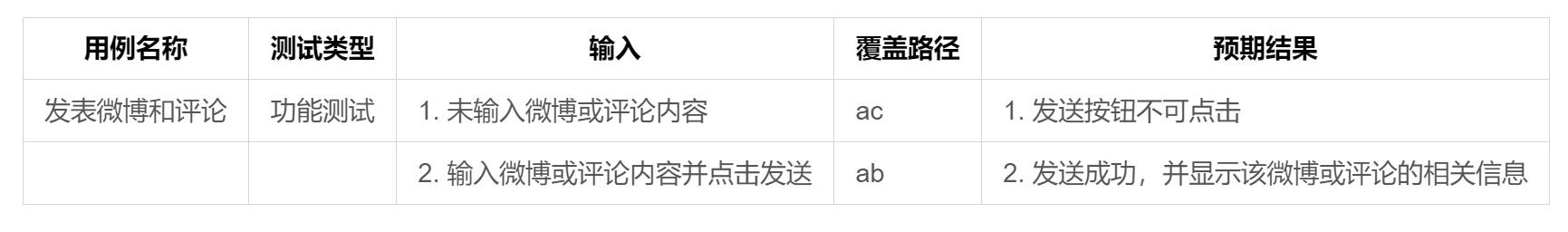


* 输入已注册的用户名和正确的密码：



（二）微博发表和评论测试





测试结果：

* 未输入微博：



* 未输入评论



* 输入微博并点击发送：



* 输入评论内容并点击发送：



第七章 心得体会

（一）课程知识理解

通过努⼒完成本次项⽬，加深了我对《计算机程序设计基础》、《数据结构》、《算法》、《Java语言与系统设计》、《数据库》、《Web技术》、《移动应用开发》等课程所学知识的理解，真正做到了理论与实践相结合。

（二）工程思维

考虑到整个项⽬是有⼀定⼯程量的，本次项⽬仅内核部分的代码编写就接近千⾏，花费了⼤概两周的 时间。考虑到⼯程量之⼤，如果事先没有按照模块化⽅法进⾏总体分析，⽽是⼀开始就深⼊到模块的 具体实现细节的话，代码的编写将变得⼗分凌乱，因此我先是通过项⽬组讨论明确了本次项⽬的⽬标，然后再根据这⼀⽬标进⾏项⽬的总体设计，最后再考虑每个模块内的具体实现，这样能够保证从总体上把握项⽬的基本架构，便于后续每⼀天的持续推进。 因为本次项⽬的代码量相较于以往所写过的⼀些课设更⼤，因此代码编写的时候尽量贴近⼯程规范， ⽐如变量、函数的命名尽可能规范、尽量降低各模块的耦合度等等。

（三）代码能⼒

为了完成本次项⽬，我写了⼤量的代码并进⾏了⼤量的调试。在以往的⼀些课设 中，代码调式往往只是在软件上进⾏，⽽本次项⽬中进⾏调试需要测试各接⼝是否有误，在实际的使⽤过程中去发现代码存在的问题，每调试⼀次⽐以往都更加繁琐。编写代码和调试的过程或许有点痛苦，但收获是绝对丰富的，每次调好⼀个bug，都有⼗分愉悦的感觉。都有离实现完整的课设更⼀步的 感觉。因此本次课设所带来的代码能⼒的提升是全⽅⾯的，不仅在于代码编写上，还在于代码调试，静态调试的能⼒逐渐提⾼，能够根据每次bug的特点，不断地输出中间变量进⾏观察，不断地锁定bug 的范围，最后找到原因修复bug。

（四）查阅官⽅⽂档能⼒

为了完成状态检测与分析这⼀模块，需要使⽤到MVC模型，⽽我在MVC模型的应⽤⽅⾯，可以说是从零开始，采取的⽅法就是边学习相关理论边完成项⽬，⽽快速⼊⻔的⽅法就是不断地根据需求查看官⽅⽂档，⼀开始还⽤的不太熟练，但因为查看的多了，也就慢慢熟练了。官⽅⽂档往往写的⼗分详细，有些还附有实例说明。查阅官⽅⽂档能⼒的提⾼对以后应该会有⼀定的帮助。

（五）软件测试

在软件测试中，我通过先绘制各模块功能流程图，再运⽤路径覆盖的思想设计测试 ⽤例，这样就⾮常⾼效地完成了测试，找出了很多⾁眼⽆法判断的bug。这也让我意识到，作为⼯程的设计者，如果考虑不周全，轻则降低⽤⼾体验感，重则可能导致系统崩溃，甚⾄造成较⼤的经济损失。

（六）坚持不懈的意志

我之所以能完成该课设，除了技术性因素外，⾮技术性因素也起到了较⼤的影响，⾯对较⼤的⼯程源码，⼼态显得⾮常重要。刚开始时我会因为任务量⼤，或者遇到从未⻅过的知识和代码写法⽽感到沮丧，甚⾄产⽣畏惧的⼼理。但是⼀路坚持下来，我逐渐发现在模块化设计的思想下，程序设计和代码编写也显得没那么困难了，遇到问题时也能⼗分冷静地阅读官⽅⽂档了，意志⼒的提升也为后续的学术研究提供了稳固的精神基础。

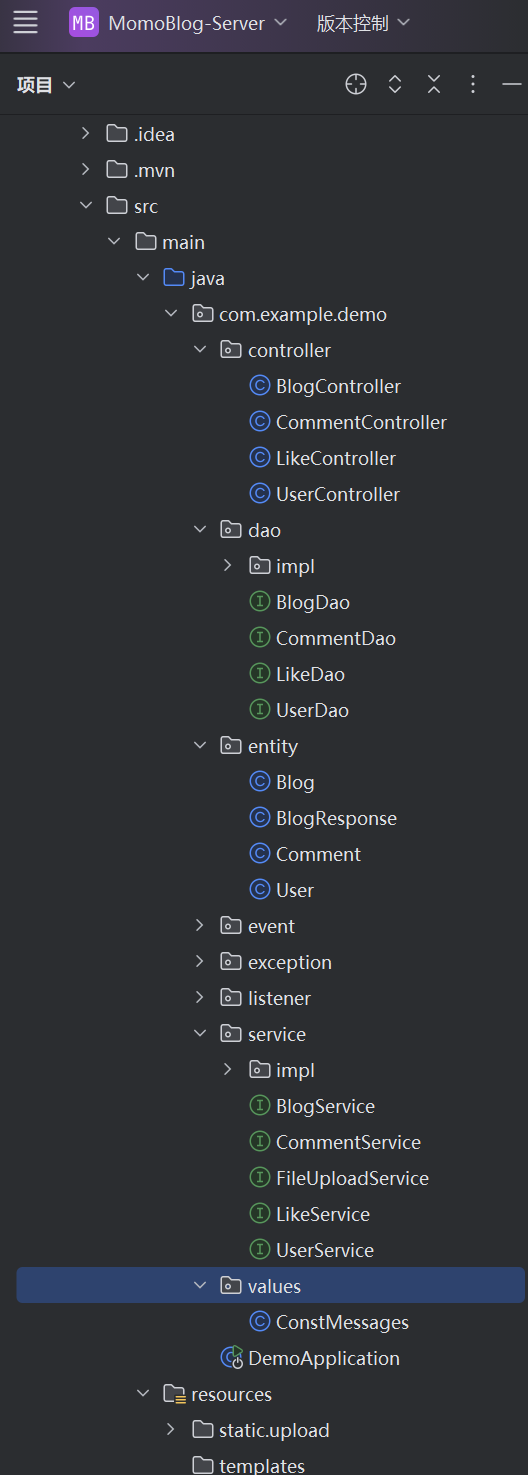
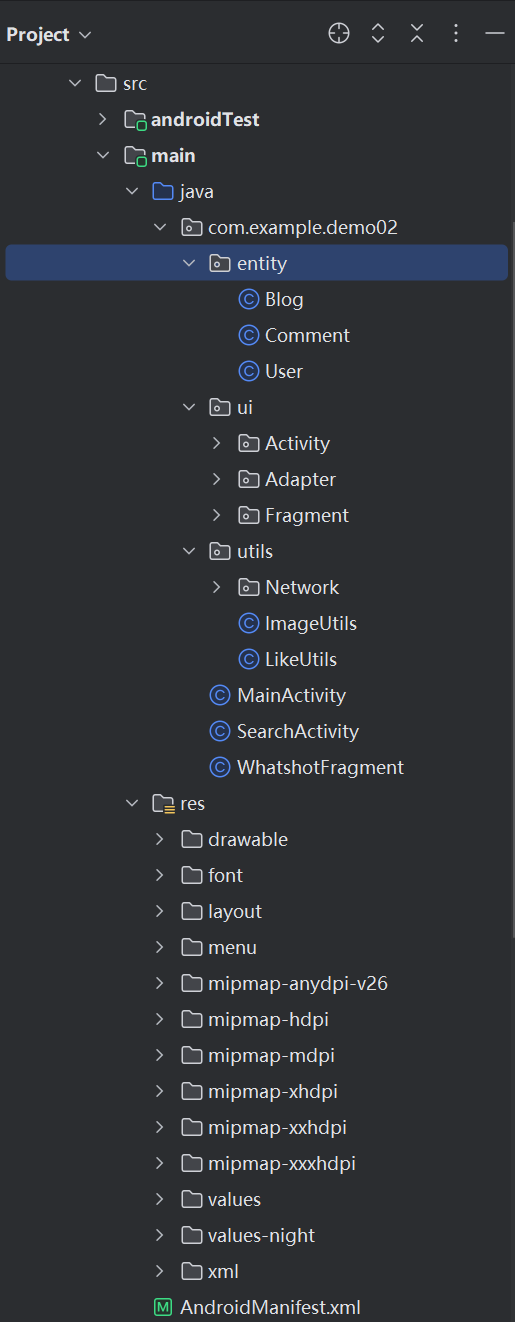
（七）全栈开发经验

通过整个实验，深刻体会了全栈开发的流程，从前端客户端到后端服务器，再到数据库的设计与应用，全面了解了移动应用的架构。通过处理实际场景中的问题，我加深了对技术栈各组件之间协作原理的认识，例如数据库性能调优、API 接口的设计规范，以及前后端分离架构的具体实现。这些实践经验不仅巩固了我的技术基础，也让我对系统整体设计有了全局的视角。

在未来的开发工作中，我希望能够以此次实验为起点，继续优化开发流程，不断探索更高效的开发模式。同时，我也将更加注重用户需求的挖掘和反馈的收集，力求在每一个细节上做到精益求精，以进一步提升应用的整体质量和用户的满意度。最终，我希望通过不断努力，开发出更多具有创新性、实用性和高品质的应用，为用户创造更大的价值。

附录 代码清单

由于完整源代码总大小高达400 kb，单个文件内无法全部包含，详见 Github 项目：



（一）前端

1. entity 包

职责：定义业务模型，与应用中的核心数据结构对应。

包含类：

* Blog：博客实体类，存储博客相关数据。
* Comment：评论实体类，存储用户评论信息。

- User：用户实体类，存储用户相关信息。

2. ui 包

职责：处理用户界面相关的逻辑。

子包结构：

Activity：包含所有 Activity 类，负责管理应用的各个页面和用户交互。

* AddBlogActivity：添加博客页面。
* DetailsActivity：博客详情页面。
* LoginActivity：登录页面。
* ProfileActivity：用户个人资料页面。
* RegisterActivity：注册页面。
* SearchActivity：搜索页面。
* WelcomeActivity：欢迎页面。
* WhatshotActivity：热点推荐页面。

Adapter：包含适配器类，用于适配数据和视图。

* BlogAdapter：博客列表适配器。
* CommentAdapter：评论列表适配器。

Fragment：包含 Fragment 类，支持模块化 UI。

* HomeFragment：主页模块。

- WhatshotFragment：热点推荐模块。

3. utils 包

职责：提供工具类以支持项目通用功能。

包含类：

Network 包：处理网络通信相关的逻辑。

* BlogRepository：博客数据仓库，负责博客数据的增删改查。
* CommentRepository：评论数据仓库，负责评论数据的管理。
* FileDownloader：文件下载工具类。
* HttpGetRequest：GET网络请求工具类。
* HttpPostRequest：POST网络请求工具类。
* UserRepository：用户数据仓库，管理用户数据。

ImageUtils：图片处理工具类。

LikeUtils：点赞操作工具类。

4. 其他核心类

MainActivity：应用的主界面，通常是应用启动后的第一个页面。

5. 资源目录

职责：存放应用的资源文件，包括图片、布局文件和字符串等。

子目录说明：

* drawable：存放应用的图片资源。
* font：存放自定义字体文件。
* layout：存放应用的 XML 布局文件。
* menu：存放菜单资源。
* mipmap-\*dpi：存放不同分辨率的启动图标。
* values 和 values-night：存放应用的字符串、主题和颜色定义。

- xml：存放 XML 配置文件。

6. 配置文件

AndroidManifest.xml：

描述应用的基本信息（如包名、Activity 声明、权限等）。

（二）后端

1. Controller 层

路径：src/main/java/com/example/demo/controller

职责：处理客户端的请求，并调用相应的 Service 层方法，返回处理结果。

包含类：

* BlogController：处理博客相关操作（如创建、获取博客）。
* CommentController：处理评论相关操作（如添加、获取评论）。
* LikeController：处理点赞功能（如点赞/取消点赞操作）
* UserController：处理用户管理功能（如用户注册、登录）。

2. DAO 层

路径：src/main/java/com/example/demo/dao

职责：提供数据访问接口，定义操作数据库的方法。

包含子包及类：

* impl 包：包含具体 DAO 接口的实现。
* BlogDao：博客数据操作类。
* CommentDao：评论数据操作类。
* LikeDao：点赞数据操作类。

- UserDao：用户数据操作类。

3. Entity 层

路径：src/main/java/com/example/demo/entity

职责：定义业务实体类，与数据库表结构一一对应。

包含类：

* Blog：博客实体。
* BlogResponse：用于返回博客数据的响应模型。
* Comment：评论实体。

- User：用户实体。

4. Service 层

路径：src/main/java/com/example/demo/service

职责：实现业务逻辑，调用 DAO 层完成数据操作。

包含子包及类：

* impl 包：Service 接口的实现。
* BlogService：博客相关业务逻辑。
* CommentService：评论相关业务逻辑。
* FileUploadService：文件上传处理逻辑。
* LikeService：点赞相关业务逻辑。

- UserService：用户相关业务逻辑。

5. 其他辅助包

event 包：事件相关处理模块（未列出具体类）。

exception 包：异常处理模块，用于自定义和统一管理项目中的异常。

listener 包：监听器模块，用于监听和处理应用中定义的事件。

values 包：常量定义。

ConstMessages：定义全局的常量信息或消息字符串。

6. 资源目录

路径：src/main/resources

职责：存放静态资源、模板文件及配置文件。

子目录：

* static.upload：存放上传的文件。

- templates：存放动态页面模板（可能用于前后端分离或渲染服务）。

7. 入口类

路径：src/main/java/com/example/demo

类名：DemoApplication

职责：项目的启动类，负责初始化 Spring Boot 应用。

附录 参考文献

1. maven公共仓库，<https://mvnrepository.com/>
2. Java(®) Platform, Standard Edition & Java Development Kit Version 14 API Specification，<https://docs.oracle.com/en/java/javase/14/docs/api/index.html>
3. Android R.java类的⼿动⽣成，<https://blog.csdn.net/ccpat/article/details/50738811>
4. Android-x86虚拟机安装配置全攻<https://www.cnblogs.com/gao241/archive/2013/03/11/2953669.html>
5. ⾯向应⽤开发者的⽂档，<https://developer.android.google.cn/docs?hl=zh-cn>
6. Uri Permission，<https://www.jianshu.com/p/35902ffb3e6c>
7. Android 11 外部存储权限适配指南及⽅案，<https://www.jianshu.com/p/e94cea26e213>
8. AsyncQueryHandler详解及使⽤\_woaishitu的博客-CSDN博客\_asyncqueryhandler，

<https://blog.csdn.net/weixin_42193691/article/details/82469627>

1. onActivityResult详细理解\_努⼒的秋波的博客-CSDN博客\_onactivityresult，

<https://blog.csdn.net/weixin_44377507/article/details/11042487>

1. 安装Android Studio遇到Unable to access Android SDK add-on list的错误 <https://blog.csdn.net/weixin_44786530/article/details/136661770>
2. Android Studio的安装，史上最详细(超多图)！！ <https://blog.csdn.net/qq_41976613/article/details/91432304>
3. 【已解决】switch语句报错Constant expression required <https://blog.csdn.net/mjh1667002013/article/details/134763804>
4. MySQL安装教程（详细版）<https://blog.csdn.net/m0_71422677/article/details/136007088>
5. 安卓移动开发基础入门 <https://blog.csdn.net/qq_61115762/article/details/136493720?spm=1001.2014.3001.5502>
6. SpringBoot配置数据源DataSource <https://blog.csdn.net/qq_20916555/article/details/80852144>
7. 程序包org.springframework.jdbc.core不存在<https://blog.csdn.net/xiaoai5324/article/details/115190961>
8. Spring学习之JdbcTemplate（详细介绍JdbcTemplate的应用）<https://blog.csdn.net/qq_39746820/article/details/124496957>
9. 【已解决】Spring常见错误：类文件具有错误的版本 61.0, 应为 52.0 <https://blog.csdn.net/CNpeaceful/article/details/134613482>
10. SpringBoot引入jdbcTemplate时报错Field jdbcTemplate in com.x required a bean of type...could not be found <https://blog.csdn.net/Hu1510592894/article/details/88854843>
11. SpringBoot 无法注入 JDBCTemplate <https://segmentfault.com/q/1010000004950486>
12. 翻车现场: 项目运行时报错找不到类: org/springframework/jdbc/core/support/JdbcDaoSupport <https://blog.csdn.net/qq_43705131/article/details/112221384>
13. SpringBoot项目编译报错 “类文件具有错误的版本 61.0, 应为 52.0” <https://blog.csdn.net/qq_45982171/article/details/128342646>
14. java.lang.NoClassDefFoundError: org/springframework/aot/AotDetector解决方法 <https://blog.csdn.net/weixin_43990604/article/details/132253318>
15. Androis Studio中使用真机调试步骤 <https://blog.csdn.net/hou09tian/article/details/119569614>
16. 【解决】异常EmptyResultDataAccessException: Incorrect result size: expected 1, actual 0 <https://blog.csdn.net/kepler_Ep/article/details/106588257>
17. OkHttp官方教程解析-彻底入门OkHttp使用 <https://blog.csdn.net/mynameishuangshuai/article/details/51303446>
18. Android OkHttp完全解析 是时候来了解OkHttp了 <https://blog.csdn.net/lmj623565791/article/details/47911083>.
19. 基于Android + Web+ MySQL设计和开发微博应用 <https://blog.csdn.net/m0_50226268/article/details/122487226>
20. 通过recycleview进行上拉加载和下拉刷新，okhttp网络请求数据加载分页 <https://blog.csdn.net/qq_36022808/article/details/99618260>
21. SwipeRefreshLayout+Recyclerview实现下拉刷新和上拉自动加载 <https://www.jianshu.com/p/96f14f6a5bb4>
22. okhttp3上传图片 <https://blog.csdn.net/itobot/article/details/85568570>
23. 使用 okhttp3库发送 get、post（json参数传递，form表单提交） java代码实现 <https://blog.csdn.net/lly576403061/article/details/131492556>
24. okhttp3用application/json请求 <https://blog.csdn.net/a77979744/article/details/114919201>
25. json的几种标准格式 <https://blog.csdn.net/weixin_48185778/article/details/109822965>
26. Glide基本使用以及加载https图片（配合OkHttp实现） <https://blog.csdn.net/android_cai_niao/article/details/108227525>
27. SQL AUTO INCREMENT(自动递增) 字段 <https://blog.csdn.net/xiaolute/article/details/90173445>
28. MD5 加密算法介绍 <https://blog.csdn.net/thlzjfefe/article/details/123449760>
29. 什么是Dao层、Entity层、Service层、Servlet层、Utils层？ <https://blog.csdn.net/Restarting2019/article/details/122296373>
30. Spring中@Autowired 注解作用是什么？具体怎么使用？ <https://blog.csdn.net/MaNong125/article/details/122628775>
31. 前后端登录时密码的加密 <https://www.cnblogs.com/jmllc/p/17809081.html>
32. spring boot 接收 post请求的多个参数 springboot接收http <https://blog.51cto.com/u_16213668/6967958>
33. Android Studio 中R标红，出现“Cannot resolve symbol ‘R’”错误！！！ <https://blog.csdn.net/i_love_program__19/article/details/80134936>
34. Java实现MD5加密的三种方式 <https://blog.csdn.net/qq_43842093/article/details/131024627>
35. 解决AndroidStudio报错：Cannot resolve symbol ‘R‘ <https://blog.csdn.net/weixin_47678542/article/details/123152973>
36. Android Studio中下载并导入Gson <https://blog.csdn.net/as_your_heart/article/details/125071455>
37. GSON + Okhttp3解析Json数据（以天气为例子解析） <https://blog.csdn.net/qq_41142037/article/details/105122145>
38. Spring Boot静态资源访问和配置全解析(看不懂你打我) https://blog.csdn.net/u010358168/article/details/81205116
39. Android如何在Android Studio的模拟器里导入图片到系统相册并可以在哪里查看的到 <https://blog.csdn.net/RuseB_/article/details/120352134>
40. Attempt to invoke virtual method 'android.content.Context.getResources()' on a null object reference <https://stackoverflow.com/questions/39293294/attempt-to-invoke-virtual-method-android-content-context-getresources-on-a-n>
41. java - NoSuchFileException when creating a file using nio - Stack Overflow.html <https://stackoverflow.com/questions/30735735/nosuchfileexception-when-creating-a-file-using-nio>
42. Android Studio使用系统自带图标\_android studio自带图标-CSDN博客. <https://blog.csdn.net/taoerchun/article/details/90216904>
43. 100 Continue - HTTP <https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Status/100>
44. sql like 通配符引发的问题 <https://blog.csdn.net/m0_37731701/article/details/119967891>
45. android icon生成ic\_launcher\_foreground android icon适配 <https://blog.51cto.com/u_16099325/6400876>
46. Servlet - 3.1下@MultipartConfig注解方式上传文件 <https://blog.csdn.net/mytt_10566/article/details/71077154>
47. 使用IDEA自动生成UML类图和时序图\_idea uml generator怎么用-CSDN博客 <https://blog.csdn.net/qq_43269093/article/details/110168656>
48. android 出键盘时页面上移-CSDN博客 <https://blog.csdn.net/weixin_35411487/article/details/141116026>
49. 如何在Android中实现键盘弹出时界面自动上移？ <https://www.kdun.com/ask/1287022.html>
50. JAVA创建文件（包括空文件） <https://blog.csdn.net/zhuxianxin0118/article/details/109485014>
51. Java List排序4种写法-CSDN博客 <https://blog.csdn.net/yexiaomodemo/article/details/130063971>
52. Android edittext 属性inputtype详解 - 简书 <https://www.jianshu.com/p/4e238eb1deb2>
53. Android：添加自定义字体并设置为默认字体 <https://blog.csdn.net/DevCyberX/article/details/132369790>
54. Android返回上一页面的方式\_android 返回上一页面-CSDN博客 <https://blog.csdn.net/CodingNotes/article/details/77697815>
55. android 如何修改主题颜色 <https://blog.51cto.com/u_16175509/10671731>
56. Android中拍照(相册中选择)并上传图片功能（包括动态获取权限）\_android 选择可以拍照上传图 或选择相册图-CSDN博客 <https://blog.csdn.net/kaolagirl/article/details/118113530>
57. Android 拍照以及相册中选择(适配高版本)————上传头像并裁剪 <https://blog.csdn.net/chen_md/article/details/130401689>
58. spring boot 实现根据用户名查找用户功能 <https://blog.csdn.net/Lushengshi/article/details/129927139>
59. 几种常用的dao层方法（用户列表，根据用户名和密码登录查询等） <https://blog.csdn.net/wy123456__/article/details/116244550>
60. 使用 NavigationUI 将界面组件连接到 NavController  \_  Android Developers <https://developer.android.google.cn/guide/navigation/integrations/ui?hl=cs>
61. Android导航组件Navigation从入门到精通 <https://blog.csdn.net/yingaizhu/article/details/105972720>
62. Android开发欢迎页点击跳过倒计时进入主页 <https://blog.csdn.net/juer2017/article/details/79069970>
63. Android 欢迎界面停留3秒的实现\_安卓app下载停留在页面-CSDN博客 <https://blog.csdn.net/huplion/article/details/52612098>
64. 【Android】实现底部选项卡切换页面效果\_选项卡切换 android开发-CSDN博客 <https://blog.csdn.net/qq_39500052/article/details/122768391>
65. Android底部导航栏的实现---------6种方法+底部大按钮跳转（最全集合）\_android 搭建底部导航-CSDN博客. <https://blog.csdn.net/chengmuzhe2690/article/details/89406085>
66. 关于BottomNavigationView的使用姿势都在这里了\_bottomnavigationview用法-CSDN博 <https://blog.csdn.net/BigBoySunshine/article/details/105774561>
67. 推荐：AutoscaleEditText - 自动缩放的Android文本输入框-CSDN博客 <https://blog.csdn.net/gitblog_00100/article/details/138558597>
68. Android TextView Autosizing 字号自动调整大小，字号自适应 <https://www.cnblogs.com/yyhimmy/p/12583593.html>
69. 【Java 进阶篇】使用 JDBC 更新数据详解\_jdbc驱动程序更新-CSDN博客 <https://blog.csdn.net/qq_21484461/article/details/133501263>
70. 在数据库中根据某个值递增更新（update）某个字段 <https://www.cnblogs.com/HYL1003597280/p/12658686.html>
71. JSON转换问题最全详解（json转List，json转对象，json转JSONObject）\_json转object-CSDN博客 <https://blog.csdn.net/JavaSupeMan/article/details/123919039>
72. @RequestBody的使用-CSDN博客 <https://blog.csdn.net/justry_deng/article/details/80972817>
73. Windows下如何查看某个端口被谁占用 | 菜鸟教程 runoob.com <https://www.runoob.com/w3cnote/windows-finds-port-usage.html>