廣東工業大學

本科毕业设计

污染地块环境管理手机终端 APP 的初步开发

学　　院　环境科学与工程学院

专　　业　环境科学

年级班别　2015 级（1）班

学　　号　3115006378

学生姓名　黄 健 楸

指导教师　王 孝 武

2019 年 5 月

摘　　要

本课题所介绍的安卓软件实现了污染场地调查任务的流程管理，帮助相关人员专注于核心业务逻辑，而不被其他冗余的工作干扰。该软件和后端结合在一起，初步解决了多种用户管理、多种任务控制的难题，基本搭建出污染场地调查的工作流管理系统，体现出工作流自动化的理念，同时也是电子政务的一部分实现。值得一提的是，定位、路线、图片等任务资料的传输，满足了过程留档这一重要需求。

本文先是细致地分析了污染场地调查中的各类需求。根据这些需求，借鉴两种流行架构，从而得出该软件总体设计，并简要介绍这两种架构共有的视图层与模型层。在总体设计的基础上，详细介绍各种页面的具体设计以及相关细节。最后还详细列举各种主要功能的测试，得出该软件基本合格的结论。

关键字：场地调查，电子政务，安卓开发，任务管理

Abstract

The Android software in this project implements the process management of the environmental site investigation task, helping the relevant personnel to focus on the core business logic instead of other redundant work. The software and the back-end are combined to solve the problems of multiple user and task management, and basically build the workflow management system for pollution site investigation, which reflects the concept of workflow automation and part of e-government. It is worth mentioning that the transmission of task data meets the important requirement of process retention.

This article analyzes the various requirements of the site survey. With these requirements, we design the software at a big picture based on two popular architectures, which share the same layer called View and Model. And the design in detail, including various pages and the business logic, followed. Finally, we list the tests on the core functions of the software, which proves that the software is basically qualified.

Keywords: Environmental site investigation, E-government, Android development, Tasks management

目录

1 绪论

随着相关法律法规的完善，我国渐渐落实了土壤污染防治行动计划，其中就包括全国各地开展的土壤污染场地调查。但其中有一些机械重复的工作，更有一些诸如定位、路线、照片的资料不便获取录入。

整个项目采取分为两大板块，后端和安卓客户端。后端部分由他人负责，依赖HTTP1.1 协议、JSON 技术、.NET 平台等，通过应用程序编程接口（API），为项目提供远程支持；安卓客户端是此文的重点，依赖安卓系统、XML 技术、远程支持等实现。

通过实现暂名为“场调记录”的安卓软件，初步完成土壤污染场地调查中的初步调查流程，并简化部分工作流程、隐藏部分业务逻辑。其中的图片上传、定位记录、路线记录等功能简化工作流程的同时，也满足了“过程留档”的需求，方便日后查询。从而提高调查工作的工作效率，也为以后拓展其他流程奠定基础。

最终，后端和安卓客户端结合在一起，初步解决了多种用户管理、多种任务控制的难题。体现出工作流自动化的理念，基本搭建出污染场地调查的工作流管理系统，同时也是电子政务的一部分实现，为公务活动中信息技术的引入添砖加瓦。

本文剩余章节的主要内容如下：

1. 需求分析：细致地分析了本项目的各类需求；
2. 总体设计分析：介绍了两种流行架构并说明采用它们的原因，同时介绍了两种架构共有的视图层与模型层；
3. 详细设计与实现：详细介绍各种页面的实现方法以及需要注意的细节；
4. 功能测试：详细列举各种主要功能的测试，得出该软件基本合格的结论。

2 需求分析

2.1 需求概述

当前，场地调查的资料上传的过程有诸多不便，故现需一个能对相关任务进行管理的工具，使管理部门、相关公司、公司员工都能按自己的职责去处理任务，从而使资料的上传、任务的反馈更加顺畅及时。

2.2 用户分析

本节分析了用户类型特征与用户整体用例。

表 2.1 用户类型特征分析表

用户类型	用户代表	用户特征
管理员	负责监控和维护服务器的人员	熟悉服务器运维
管理部门	部门负责人	熟悉调查流程、内容及其考核标准
环保公司	环保公司负责人	熟悉调查流程、内容及其上交标准
调查员工	环保公司员工	熟悉调查流程、内容及其操作标准

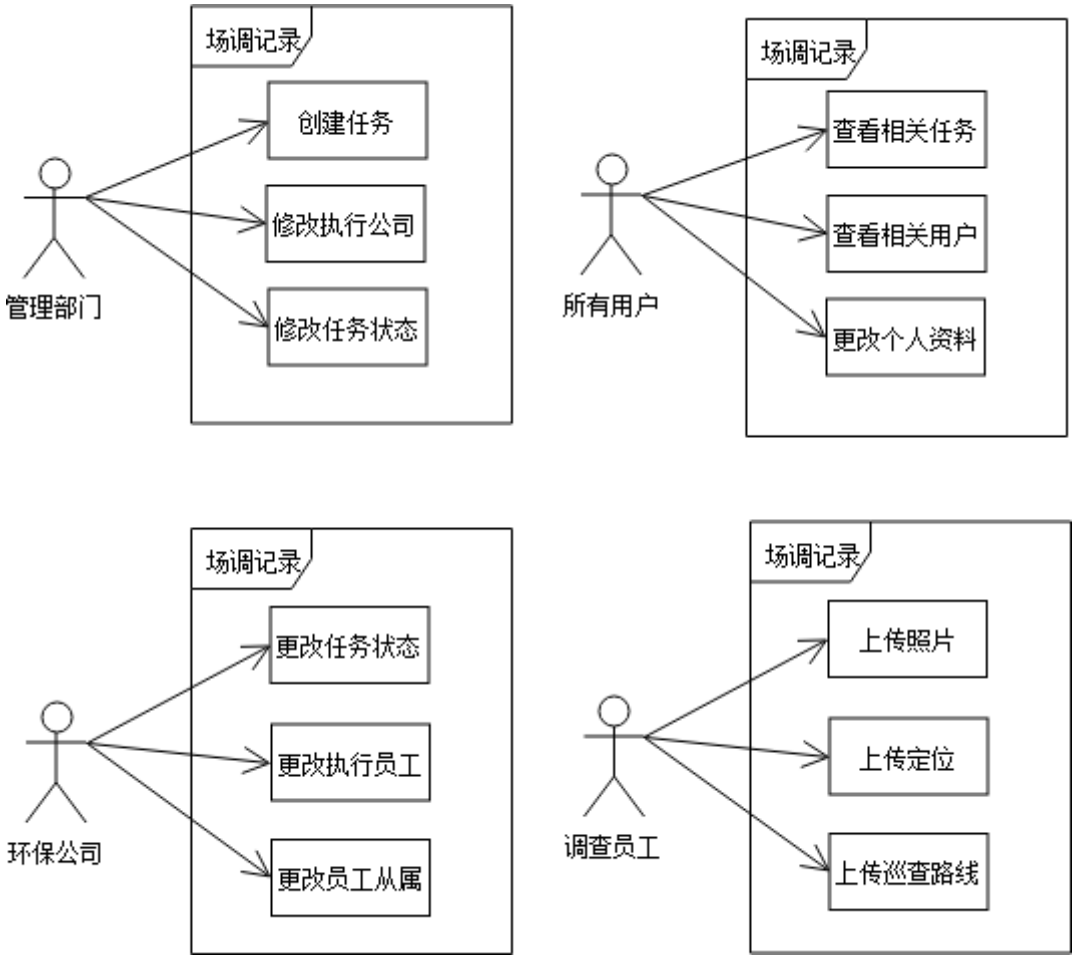

图 2.1 用户整体用例图

2.3 功能需求

2.3.1 注册登录

在使用场调记录软件的功能之前，必须先注册登录。用户名的实质为昵称，可以使用邮箱与密码或用户名与密码的组合登录，也因此，用户名与邮箱需唯一。

这里阐述的是最简单的登录逻辑，其实不方便手机用户使用，实际采用而更方便但更复杂的登录逻辑**错误!未定义书签。**在 4.3.2 登录概述一节阐述。

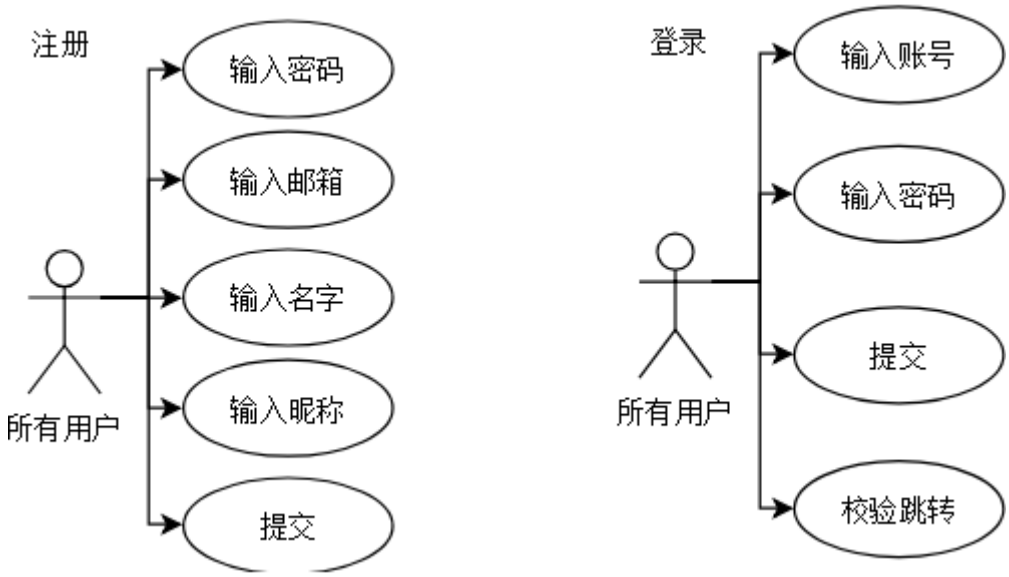

图 2.2 用户注册登录用例图

2.3.2 用户管理

本软件有四种用户使用——管理员、管理部门、公司用户、普通用户。必须有一个清晰的用户管理系统，才能使得整个任务处理流程更加简洁。

最基本的管理系统见图 2.2。所有用户都可以修改个人资料，公司用户可添加员工与删除员工，管理员可修改用户类型与用户注册资料。涉及到任务的权限管理见 2.3.3 任务管理。

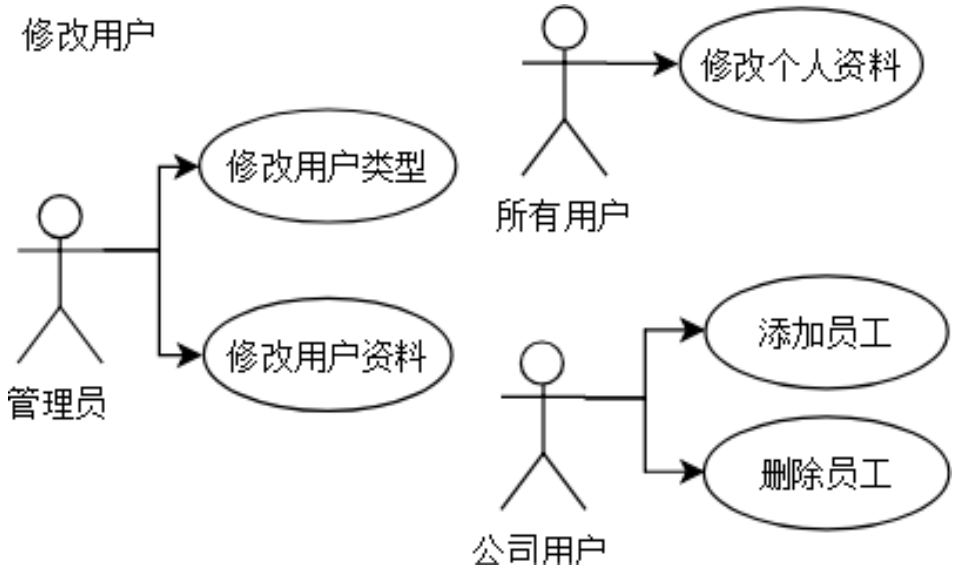

图 2-3 用户管理用例图

2.3.3 任务管理

目前，本软件只完成了初步调查任务的功能，但整个处理任务的过程是通用的，这保证了该软件的可拓展性。如图 2.4 所示，任务分为创建、修改、上传结果三大部分，对应的任务状态有创建、进行、考核和结束。各用户职责如下所述：

所有用户均可读取任务数据，但只有相关公司用户和相关普通用户可以读取执行者。

管理部门负责创建任务，提供标题、详情、类型、公司（单选）等信息；将任务的考核状态与结束状态的互换；在中途修改创建时任务的信息。

公司用户负责选择执行任务的员工（可多选），将任务状态在创建、进行、考核间逐步转化。

普通用户负责将任务资料按时上传，如照片、定位、巡查路线等。

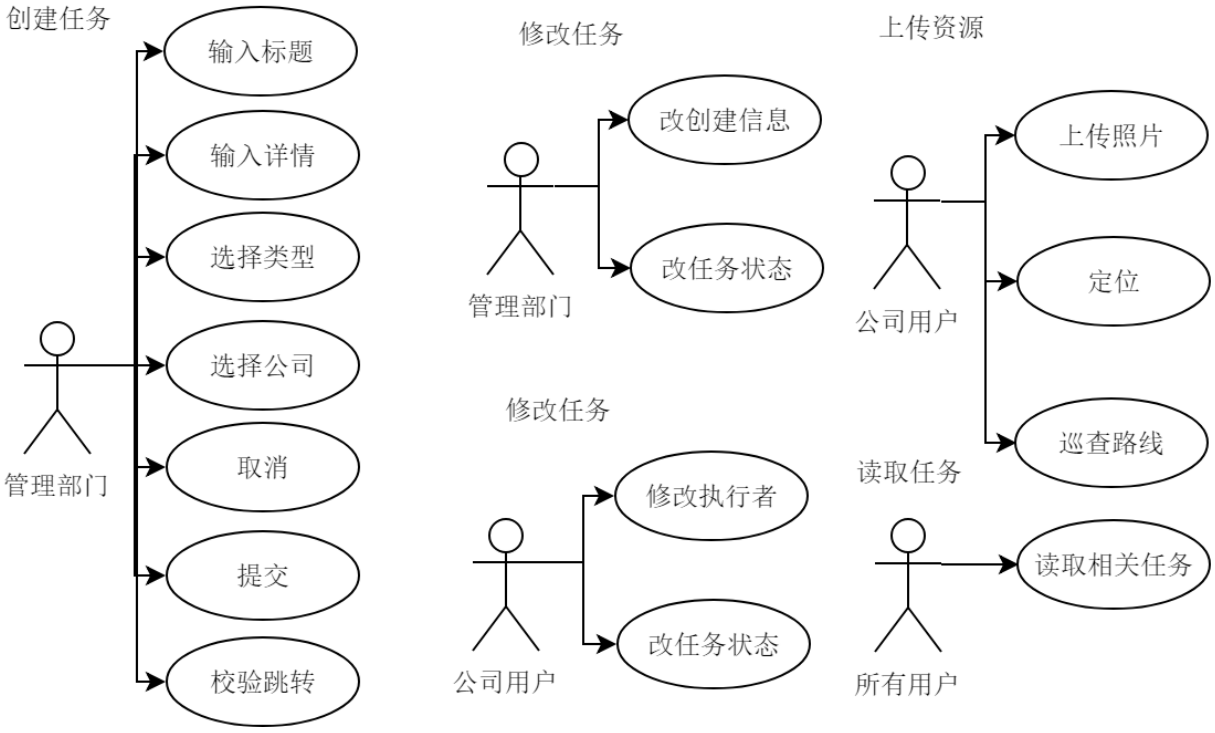

图 2.4 用户任务用例

2.3.4 上传照片

从图库批量选择或通过拍照逐一添加。添加后附上相关信息，检查无误后，点击上

传按钮即可上传。为保证传输效率，过大的图片会被压缩。

2.3.5 上传定位与巡查路线记录

功能需求为记录场地中心坐标、上传照片时的坐标，记录巡查路线。性能需求为所确定坐标偏差不超过 20 米。

2.4 性能需求

2.4.1 软件响应时间需求

对耗时较少的本地操作，如打开或切换页面和退出登录，响应时间应低于 1 秒；对耗时较高的联网操作，如下载任务详情、传任务巡查路线或图片，在网络较为顺畅的情况下不得超过 8 秒。

应合理规划线程池，以免主线程负担太重

2.4.2 软件可靠性需求

软件可靠性是指用户在软件中的一些操作要获得有效、及时、稳定的输出。如获取任务详细信息时，软件要用尽量小的网络资源，把尽量新的任务详情加载到页面中，而程序不会崩溃。

2.4.3 软件易用性需求

应用界面与交互对用户友好——美观且容易操作，还能适时配上一些提醒，从而使新接触的用户也能马上上手。

2.4.4 软件可维护性和可拓展性需求

本质上是要求程序耦合性低，符合面向对象程序设计的要求，从而使得各个功能模块基本互不干扰。其带来的效果是，添加、删除、维护一个功能模块时，不用考虑其他模块的实现细节，基本不用考虑对其他模块造成的影响。

2.5 其他需求

2.5.1 安装环境

SDK2.0 及以上、Android 5.0 及以上，运行内存大于 2G、支持蓝牙 4.0、支持 3G/4G 网络、支持 GPS 定位、支持拍照（像素大于 1300 万）。

2.5.2 开发环境

本项目使用 Android Studio 以及相关软件进行开发。开发语言为 Java，软件 UI 使用 xml 文件制作，数据库使用 SQLite，开发环境为 Window10 与 Android6.0.1。

2.5.3 安全性

保证最基本的安全性，如验证码的使用，安卓手机与后台接口之间的传输数据要按 HTTP1.1 协议实现等。

2.6 系统可行性分析

需求方面，该软件的需求明确而细致；技术方面，Android 开发现已成一套完善的体系，SQLite 简单易用、有配套且成熟的 DAO，网络交流、定位、拍照等都有较好的支持；现实方面，改软件可以减少各层级人员的工作量，并使整个调查过程进一步规范化。分析这些方面后可知，该项目的可行性较高。

3 总体设计

3.1 软件架构概述

传统的安卓软件通常采用“模型-视图-控制器”（MVC）的架构框架，这种架构模式简单易用，但由于其“控制器”的职责比较模糊，开发时容易变得臃肿，故不适合处理复杂的业务逻辑；而谷歌安卓开发团队更推荐一种新型的架构方式——“模型-视图-视图模型”（MVVM）架构——基于这种架构，系统各模块职责区分清晰，能够很好地为项目解耦，但比起“模型-视图-控制器”框架要复杂得多，且不宜用于页面简单的模块。

一个安卓应用，不同页面、功能的复杂程度不同，于是考虑让复杂的部分采用 “模型-视图-视图模型”架构，而让其他部分采用传统的“模型-视图-视图模型”架构。

MVC 结构有多种变体，但最经典的结构是这三层：模型、控制器、视图。模型层用于搜集并处理程序的数据和状态，如网络请求、数据库等；视图层通过一些图形用户界面组件构成，负责且只负责向用户展示数据并与之交互。

MVVM 的基本结构也是三层：模型、视图模型、视图。模型为封装数据相关操作和存储的逻辑，且会通过一些手段绑定视图——当模型中的数据更新时，视图的数据也会随之更新；视图是处理界面、人机交互的逻辑层，与 MVC 的视图层一致；视图模型不是视图加上模型，而是视图模型与视图状态的组合，它主要就是为视图层提供一个可靠的数据模型并同时搜集处理相关数据。

3.2 总体功能模块设计

3.2.1 概述

任务列表和用户列表比较复杂，适合 MVVM 架构；注册登录与用户资料属于简答功能，采用 MVC 架构。

3.2.2 MVVM 架构

本项目采取的 MVVM 架构，如图 3.1 所示，从底部向顶部看，采用双数据源的方式，通过存储区与 NetworkBoundResource 类（该类详见 4.2 网络与本地数据库）保证本

地数据库为单一可信来源，从而构造可靠的模型层。视图模型主要通过转发视图层（Activity/Fragment）的请求，从而减少视图层的耦合性、并使其职责更为清晰。

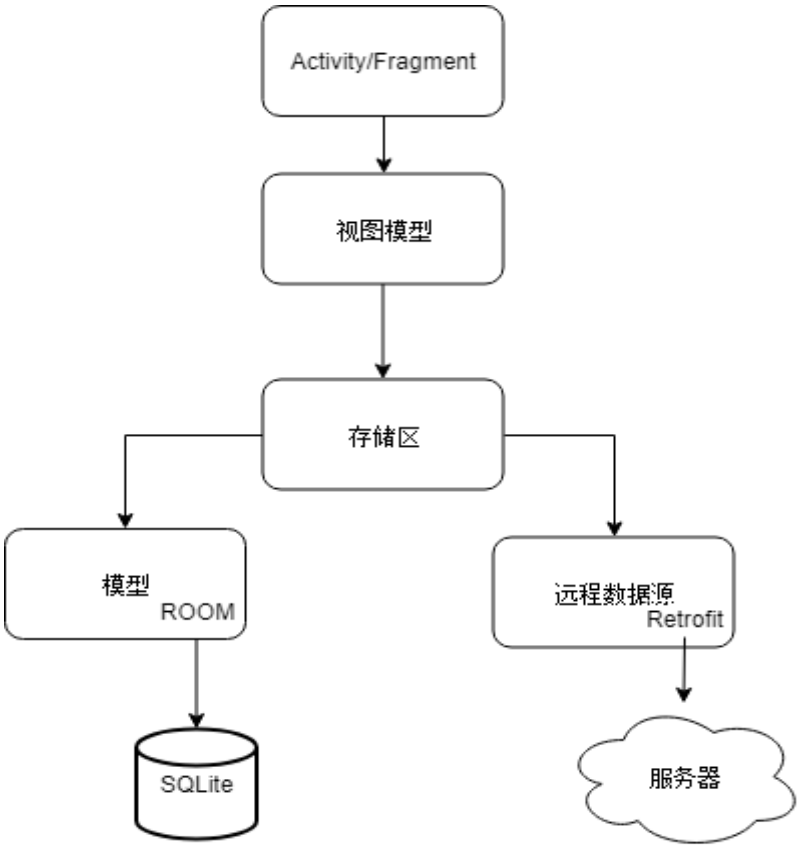

图 3.1 安卓 MVVM 架构示意图[1]

3.2.3 MVC 架构

如图 2.2 所示，本项目采取的 MVC 架构也有多个数据源，但其处理逻辑简单得多——一些操作默认需要从远程数据源（即后台 API）获取数据，存到首选项文件或数据库中，供另一些默认只从这些本地数据源读取的操作使用。

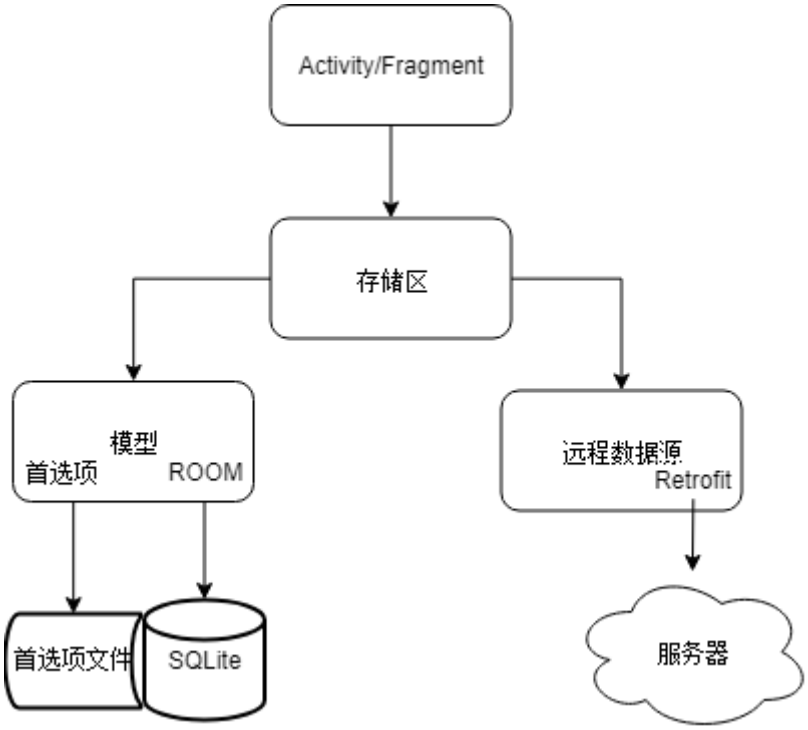

图 3.2 MVC 架构示意图

3.3 视图层设计

3.3.1 概念层面

整体来看，该软件的视图主要有列表、详情、地图、输入页面组成。

在颜色方面，本软件主要采用白色、浅橙色（#FFFBE6）、橙色（#ffa726）的单色体系——使用橙色突出关联重要功能的按钮，使用浅橙色美化背景且缓解白色带来的视觉疲劳，主体内容采用白色，使其显眼而不突出。字体颜色多采用与背景和主题对比度高的黑色和灰色。还有一些特殊的按钮会采用绿色和红色，绿色的包括上传相关、注册登录的按钮，红色的包括取消、退出登录。

3.3.2 实现层面

本软件视图层采用安卓的 XML 布局实现。使用 XML 的最大好处在于解耦——视图层在代码外部，即修改或微调视图时，无需修改源代码并重新编译。采用此方案还能方便轻松地编写 UI，避免在源码里编写重复无趣的代码，并能简化调试过程。

3.4 模型层设计

本软件的数据库主要用于缓存用户与相关任务，数据量较小，且已有后端支持保证，故本地数据库表的设计的总体要求较低，但仍需降低数据的冗余情况，并尽可能缓存以减少用户流量消耗。数据库概念设计过程构建的实体关系模型如图 3.3 所示；数据库表的设计详见表 3.1 和表 3.2。

至于远程数据源，本软件采用 Retrofit 和 OkHttp3 依赖库构建接口类与请求客户端，向后台 API 请求数据。该后台基本符合 RESTful 特性，使得 GET、POST、PUT、DELETE 等操作正常安全地进行。

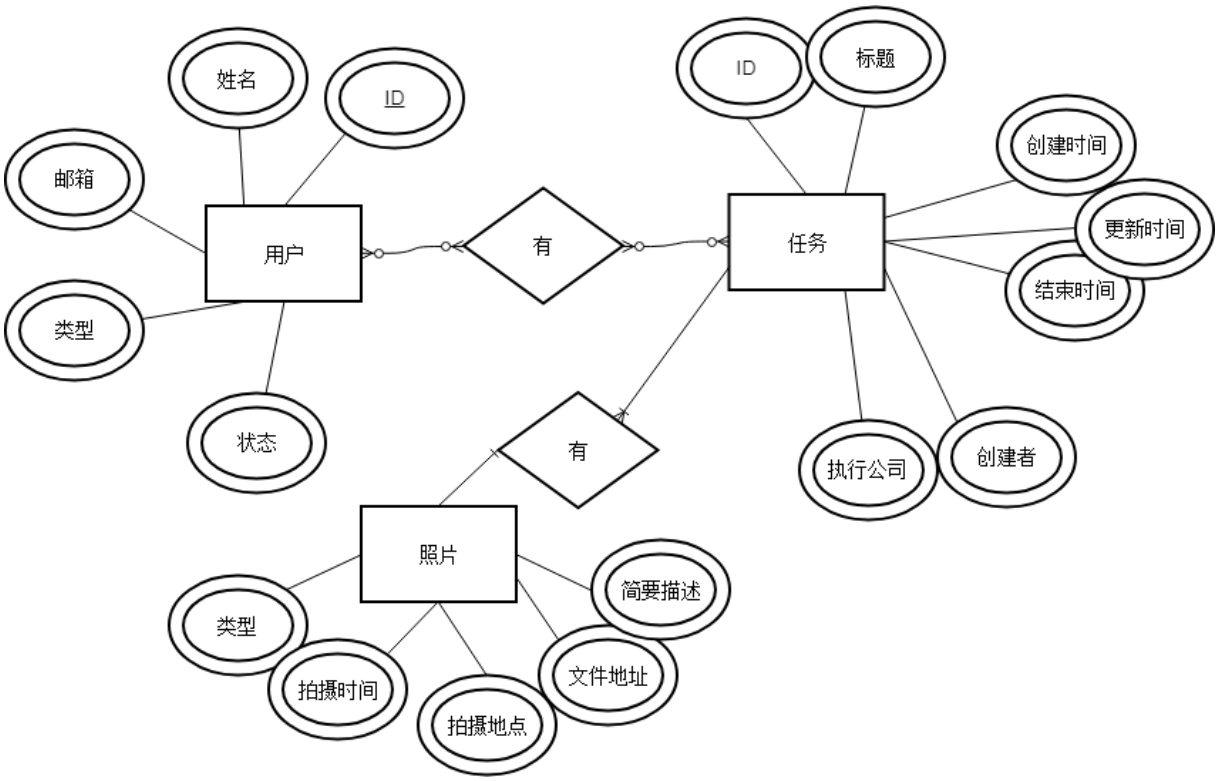

图 3.3 本软件数据库实体关系模型

表 3.1 用户表

字段	字段名	类型	约束
用户编号	id	int	主键
姓名	name	String	非空
邮箱	mail	String	非空
类型	type	int	非空
状态	status	int	非空
所属公司	company_id	int	无

表 3.2 任务表

字段	字段名	类型	约束
任务编号	taskID	int	主键
任务标题	title	String	非空
创建时间	createAt	long	非空
更新时间	updateAt	long	非空
结束时间	finishAt	long	无
类型	type	int	非空
指派人	assigner_id	int	无
被指派人	assignee_id	int	无
状态	status	int	非空

表 3.3 照片表

字段	字段名	类型	约束
照片编号	photoID	int	主键
任务编号	taskID	int	非空
子编号	subID	int	非空
上传时间	photoTime	int	非空
上传者	author_id	int	非空
图片地址	path	String	非空
图片描述	description	String	无
上传地点	location	String	非空

4 详细设计与实现

4.1 详细设计概述

本安卓软件主体采用 MVVM 架构，模型层包含远程服务器与本地数据库，视图层使用 XML 方案实现。难点在于视图模型层的搭建，这也是本章要重点讲解的。

4.2 网络与本地数据库

采用谷歌安卓开发团队推荐的做法，采用一边从网络加载数据，一边显示本地数据库中存储的副本，但要以本地数据库为单一数据源。这时，需要一个类来判断什么时候、如何使用这两个数据源。核心代码如下：

```
public NetworkBoundResource(@NotNull AppExecutors appExecutors){
    this.appExecutors = appExecutors;
    result = new MediatorLiveData<>();
    result.setValue(Resource.loading(null));
    final LiveData<ResultType> dbSource = this.loadFromDb();
    result.addSource(dbSource, data -> {
        result.removeSource(dbSource);
        if (shouldFetch(data)){
            fetchFromNetwork(dbSource);
        } else {
            result.addSource(dbSource, o ->
                    result.setValue(Resource.success(data)));
        }
    });
```

此代码的思路可由图 4.1 处理数据库与网络的流程图说明：先从本地数据库加载数据，判断所取数据是否为空并判断是否需要发送请求。若需请求数据，则判断是否请求成功，成功后则将其存入数据库并重新加载数据。

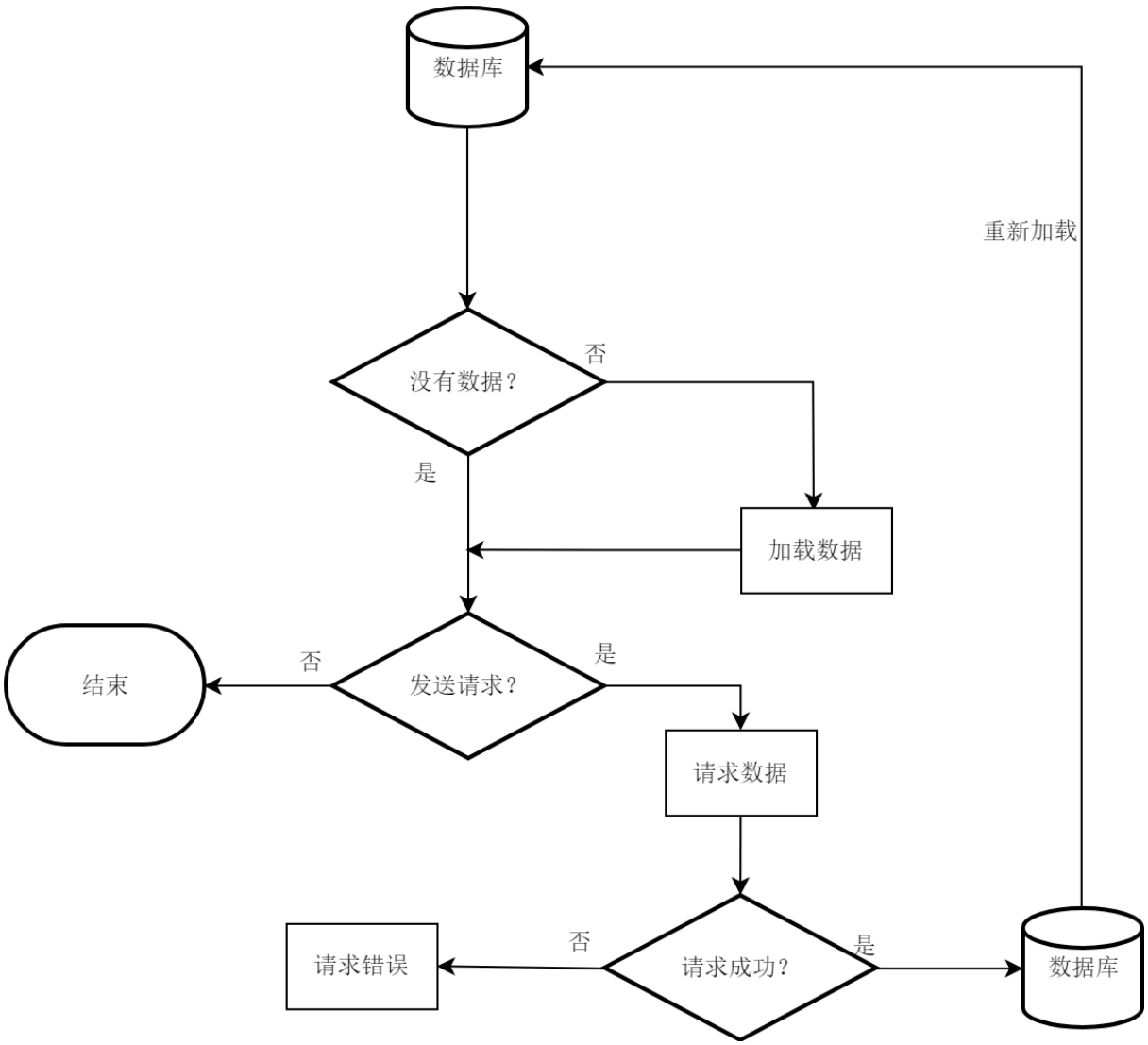

图 4.1 处理数据库与网络的流程图

4.3 注册登录的实现

注册登录采用 MVC 架构，主要逻辑都是在登录或注册按钮上设置监听器，当用户触发按钮时，控制层从视图层获取相关数据，将相关数据交给 OkHttp3 客户端，客户端再发送相关请求并返回请求结果。以登录为例的核心代码如下：

```
@OnClick(R.id.btn_login)
void login(){
    final String email = _emailText.getText().toString();
    final String password = _passwordText.getText().toString();
    login(email, password);
}

void login(String email, String password) {
    _loginButton.setEnabled(false);
    if (!validate(email, password)) {
        onLoginFailed(null);
        return;
    }
    RequestBody requestBody = new FormBody.Builder()
            .add(PASSWORD, password)
            .add(USER_STRING, email)
            .build();

    HttpUtil.POST(LOGIN, requestBody, new Callback() {
        @Override
        public void onFailure(@NonNull Call call, @NonNull IOException
e) {
            // 离线登录：利用首选项文件存储的账号密码
        }

        @Override
        public void onResponse(@NonNull Call call, @NonNull Response
response) throws IOException {
            // 根据返回的状态码判断是否登录成功，并由此决定后续操作
    });
}
```

4.3.1 注册概述

用户首次使用该应用时，需先注册账号。用户在注册页面输入用户信息和账号密码，后台确认创建用户成功后，应用将跳转至登录界面，用户可使用该账号登录。

4.3.2 登录概述

如图 4.2 所示，登录的逻辑相对注册要复杂一些，这是考虑了方便性和安全性后的结果。此登录流程默认采用 Cookies 获取会话的形式登录，使在保持联网的时候，一次登录可保持一周时间；若失败，则采用进入注册 Activity，加载用户首选项，根据记住密码和自动登录的选项来执行；若此时没有网络，则按本地保存的账号密码来判断登录情况。

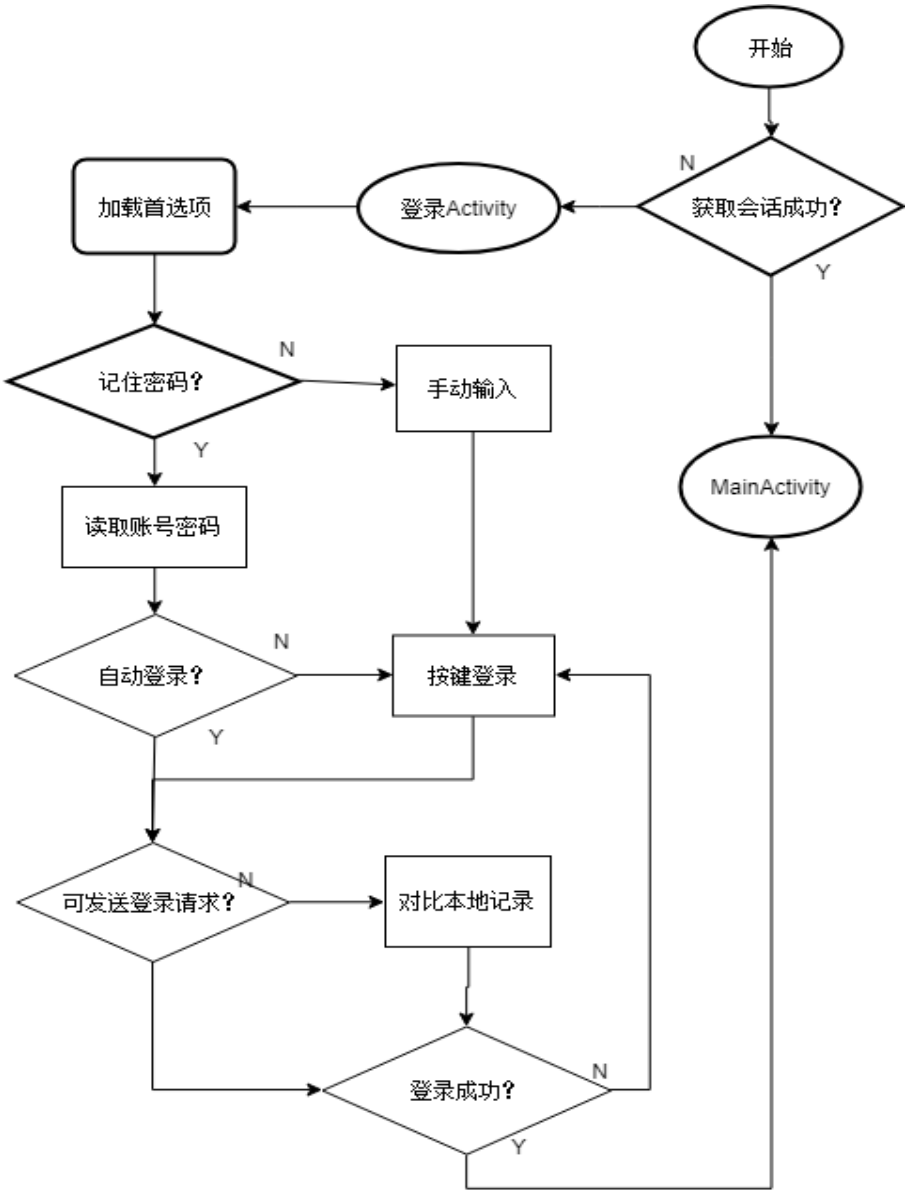

图 4.2 登录流程图

4.3.3 注册登录页面

如图 4.3 所示，登录注册页面均以淡橙色为底色，以图标加应用名为顶部，以按钮与登录注册切换链接为底部。总体样式统一美观。

值得一提的是，当选中自动登录检查框时，记住密码检查框也将会处于被选中状态，取消选中记住密码时，自动登录检查框也会处于被选中状态。

图 4.3 登录注册示例

4.4 用户管理

虽然每种用户的用户管理的相关内容不同，但是页面基本相同，故此处仅展示“小红”用户相关的示例图。相关页面如图 4.4 所示，用例说明详见 5.1 测试概述与测试用

例。

4.4.1 本用户详情页

用户详情页中，工具栏标题为“个人资料”，标题下方是包含头像、姓名、所属组织的简要介绍，再下方是详细信息列表，最后是退出登录按钮。若用户无所属组织，则不显示组织信息。这类用户包括管理员、管理部门、公司用户和无组织的普通用户。

此页面采用 MVC 架构，控制器直接向代表模型层的首选项文件请求用户的个人信息，再加载到视图层上。

4.4.2 用户列表页

用户列表页的主体内容为与本用户相关的用户。根据这个需求，管理员的用户列表页显示所有用户，管理部门的页面显示所有公司用户，公司用户显示所有员工用户，员工用户显示本公司用户以及所有隶属本公司的普通用户。在“小红”用户一例中，此页面显示的成员应包括“小明公司”、“小张”、“小红”，而图 4.4 的结果与之相符。

此页面采用 MVVM 架构，管理页面的 Fragment 通过 ViewModel 获取相关数据，ViewModel 根据不同用户类型选定请求方式，Repository 负责处理两种数据源的选择。以普通用户为例的核心代码如下（其中 NetworkBoundResource 的介绍见 4.2 网络与本地数据库，完整代码见附录）：

```
// Fragment: 通知 ViewModel 观察获取的用户列表，更新时视图层也应更新
userViewModel.getAllUsers().observe(this, usersResource ->
adapter.submitList(usersResource.data));
```

```
// viewModel: 根据用户类型转发请求
public LiveData<Resource<List<User>>> getAllUsers() {
    // 省略十数行代码
    if (userType == user && me.getCompany() != null) {
        User company = me.getCompany();
        result = repository.getUsersInCompany(me.getType(), compa-
ny.getUid());
    } else if (userType == group) {
        result = repository.getUsersInCompany(me.getType(),
me.getUid());
    } else if (userType == manage) {
        result = repository.getCompanys();
    } else if (userType == admin) {
        result = repository.getAllUsers();
    }
    return result;
}
```

```
// Repository:判定什么时候需要向网络发送请求更新数据。省略装饰器。
public LiveData<Resource<List<User>>> getUsersInCompany(int type, int
companyID){
    return (new NetworkBoundResource<List<User>, BigkeerRe-
sponse<GroupInfoResult>>(mAppExecutors) {
        protected void saveCallResult(@NonNull BigkeerResponse<Group-
InfoResult> item) {
            // 将网络请求的结果保存到数据库
        }

        protected boolean shouldFetch(@Nullable List<User> data) {
            // 判定是否需要向网络请求数据
            return data == null || data.isEmpty() || userLis-
tRateLimit.shouldFetch(USER_ME);
        }

        protected LiveData<List<User>> loadFromDb() {
            // 加载数据库的数据
            return mUserDao.loadUserByGroupID(companyID);
        }

        public LiveData<ApiResponse<BigkeerResponse<GroupInfoRe-
sult>>> createCall() {
            // 创建请求
        }

        protected void onFetchFailed() {
            // 请求失败时重设计数器
            userListRateLimit.reset(USER_ME);
        }
    }).getAsLiveData();
}
```

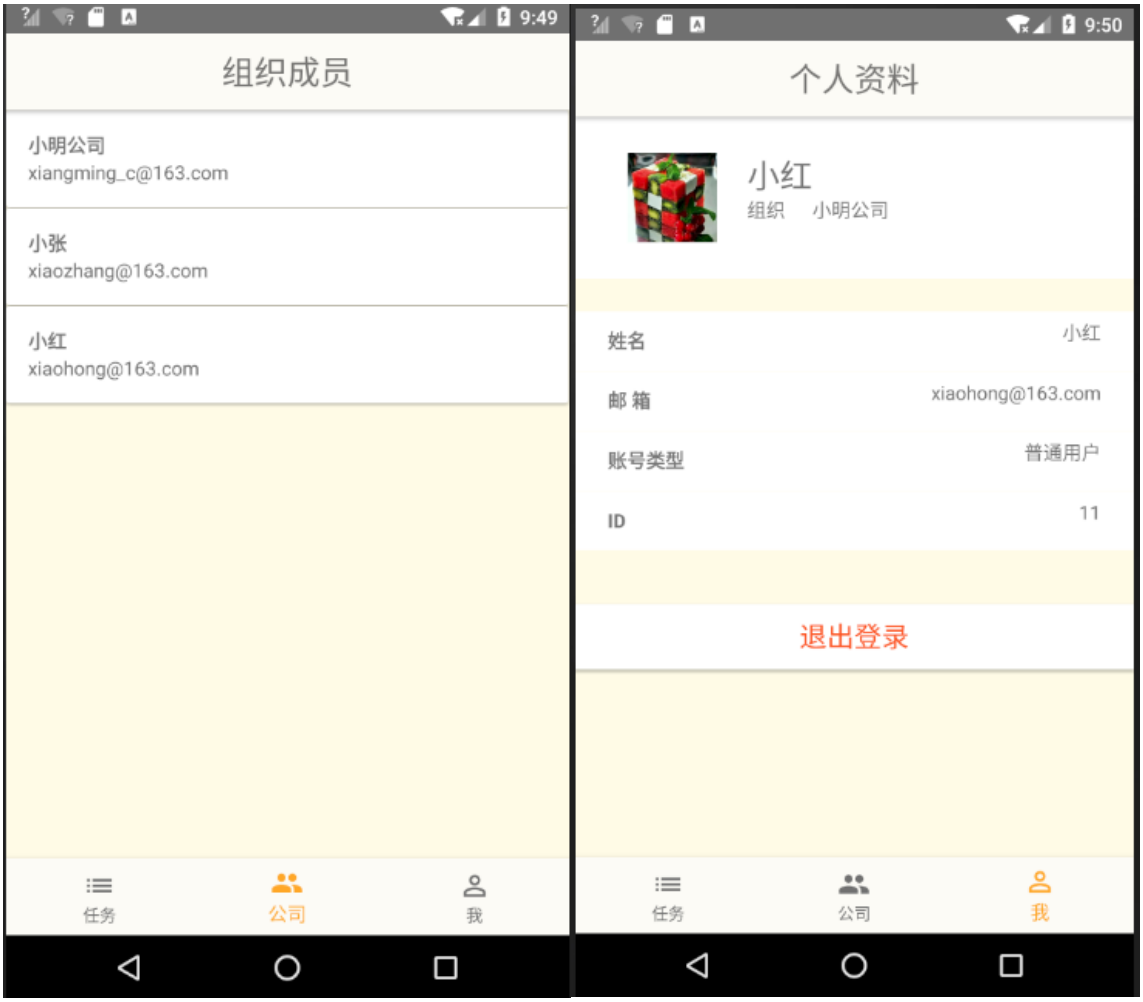

图 4.4 用户详情页与用户列表页

4.5 任务与流程

4.5.1 查看任务列表

任务列表的主体内容为与本用户相关的用户。根据这一需求，管理员的任务列表页面显示所有任务，管理部门的这一页面显示所有该部门创建的任务、并带有新建任务浮动按钮（参考图 4.5 左），公司用户显示本公司接收的所有任务，普通用户显示该用户执行的任务（参考图 4.5 右）。

本列表页也采用了 MVVM 架构，实现原理与用户列表页基本一致，可参照 4.4.2 用户列表页。具体实现见附录。

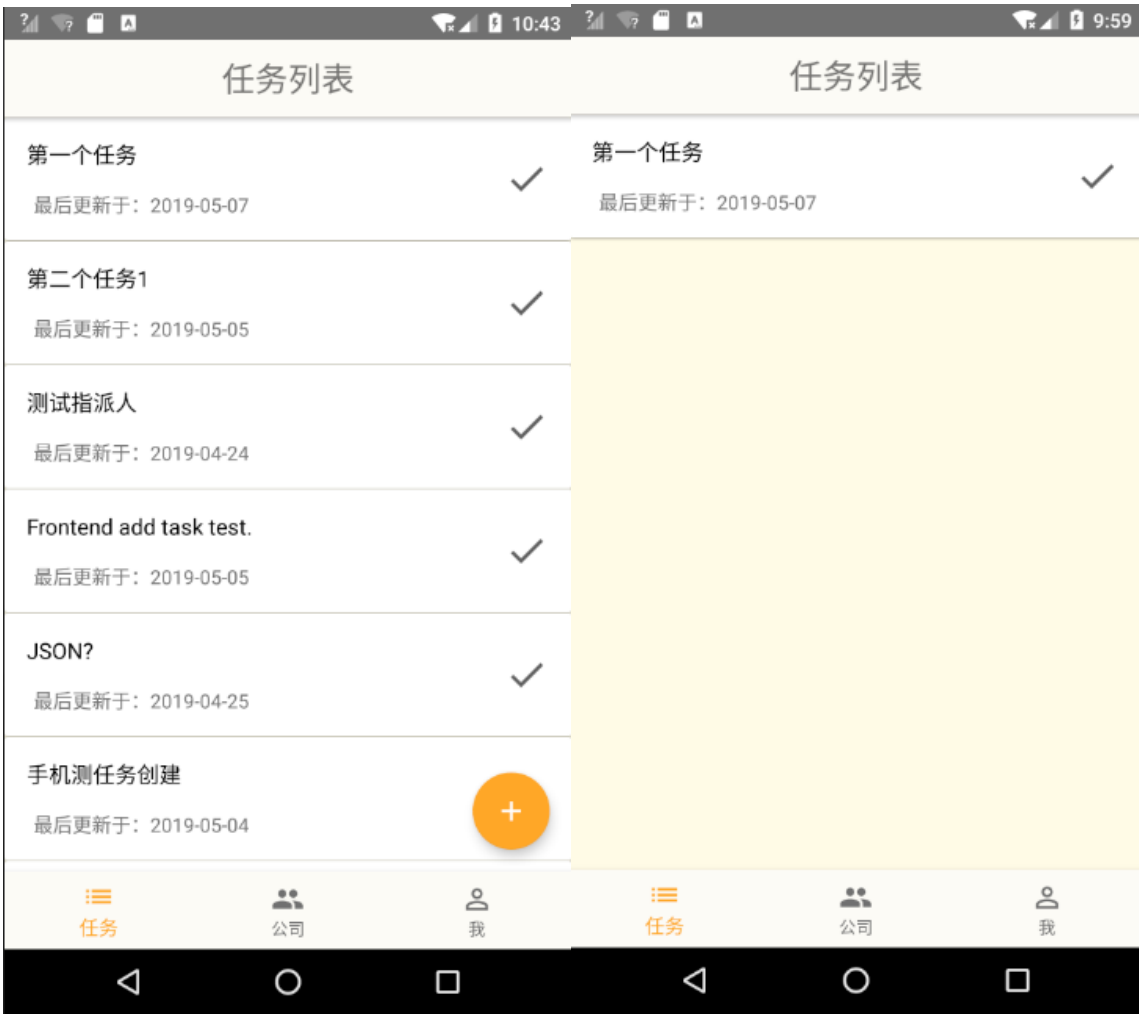

图 4.5 管理部门（左）与其他用户（右）的任务列表页

4.5.2 查看任务详情

各种用户基本都能通过点击任务列表页的某一项，见到该项任务的所有数据，但管理员和管理部门不能见到任务的执行者。

任务详情页采用 MVVM 架构。管理该页面的 Fragment 根据任务列表页打开详情页时传入的任务编号参数，通知 ViewModel 观察数据仓库中对应任务的数据，加载至视图层。由于任务中管理部门、执行公司、执行者在数据库中均由各自的用户编号代表，故需通知ViewModel观察数据仓库中对应用户的名字。其他原理与4.4.2用户列表页类似，故不再赘述。

图 4.6 任务详情页示意图

4.5.3 管理部门用户新建任务

管理部门用户可点击任务列表页右下角的浮动按钮，进入新建任务界面，输入或选择各项内容后即可点击创建。需要特殊说明的是，指定执行公司时可采用两种方案，一种是点击按钮跳转至用户列表页（如图 4.7 所示），选后跳转回新建任务界面，且执行公司输入框更新为所选用户的编号；也可直接在执行公司处输入用户编号。

图 4.7 新建任务界面示意图

4.5.4 修改任务状态

若管理部门或公司用户可修改某任务状态时，该用户可选择该任务的任务详情页面右上角菜单栏中的“修改任务状态”按钮，便会弹出更改类型选择框。

图 4.8 更改任务状态选择框示意图

4.5.5 普通用户上传照片

图片选择依赖于 Matisse 库，加载依赖于 Picasso 库，压缩方法为质量压缩，上传按照 HTTP 协议，使用“multipart/form-data”。

具体实现见附录，最终效果见图 4.9。

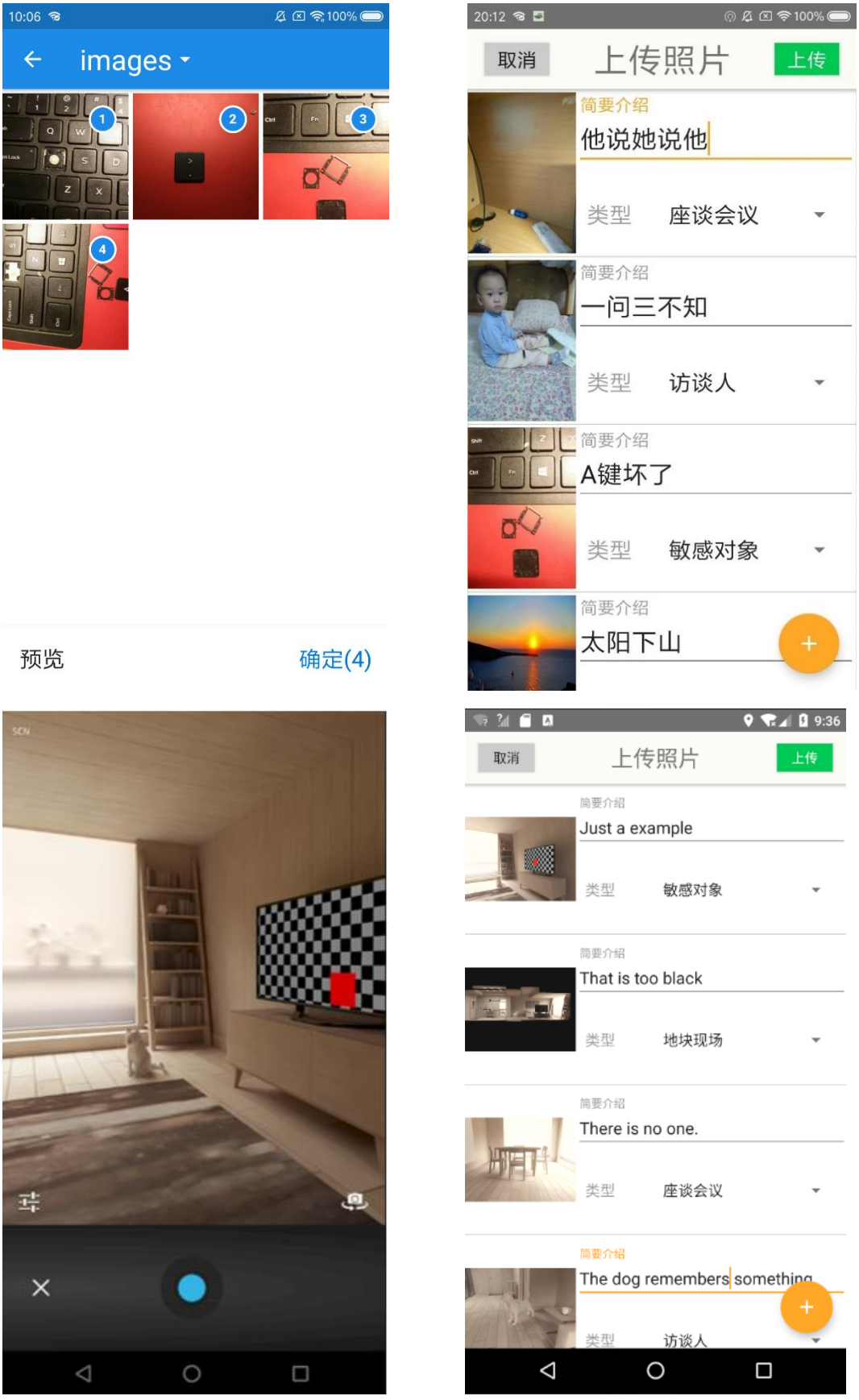

图 4.9 选择图片相关功能示意图

4.5.6 普通用户上传定位与巡查路线

如图 4.10 所示，本软件的定位功能暂依赖内置于软件中的百度地图提供的安卓SDK；巡查功能在定位功能的基础上记录了定位点，并标记出路线与端点。

图 4.10 定位与巡查路线示意图

5 功能测试

5.1 测试概述与测试用例

本项目采用的测试类型有单元测试与功能测试。这里的单元测试是指逐一测试程序的最小单元，功能测试要求直接使用各项功能以检验功能表现是否符合预期。前者从。安卓开发的单元测试可借助 JUnit4 完成，功能测试可使用 adb 与测试机完成。由于篇幅有限，本章只详细介绍相关功能测试。

在列举功能测试之前，需先介绍相关测试用例：

1. 管理员用户；
2. 管理部门用户；
3. 公司用户小明；
4. 普通用户小红；
5. 测试任务若干。

5.2 用户

5.2.1 登录注册

表 5.1 测试用户注册功能模块表

用例编号	A1	优先级	1	用例级别	重要
用例摘要	测试用户注册功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-01	开发人员	黄健楸
前置条件	软件安装完成后，进入登录页面，点击“点击注册”				
测试方法	手工测试				
输入数据	测试用例中所有用户（管理员除外）及其相关信息				
执行步骤	点击提交				
预期输出	弹出带有“注册成功”字样的 Toast，并跳转到登录界面				
实际结果	弹出的 Toast 带有“注册成功”，并跳转到登录界面				
测试日期	2019-04-01				
结论	▪ 通过		□ 未通过		

表 5.2 测试用户在线首次登录功能模块表

用例编号	A2	优先级	1	用例级别	重要
用例摘要	测试用户在线首次登录功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-01	开发人员	黄健楸
前置条件	联网，无登录状态进入应用，或退出登录，或注册成功				
测试方法	手工测试				
输入数据	输入测试用例中各用户账户以及对应密码				
执行步骤	点击提交				
预期输出	登录时出现带有“很快就好”字样的进度条。若登录成功，跳转至主页面；反之，弹出“注册失败”				
实际结果	成功登录				
测试日期	2019-04-01				
结论	▪ 通过		□ 未通过		

表 5.3 测试用户在线再次登录功能模块表

用例编号	A21	优先级	2	用例级别	一般
用例摘要	测试用户在线再次登录功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-03	开发人员	黄健楸
前置条件	用户登录成功一周内且不退出登录，在线再次启动应用				
测试方法	手工测试				
实际结果	成功进入应用主界面，在线再次登录成功				
测试日期	2019-04-03				
结论	■ 通过		□ 未通过		

表 5.4 测试用户离线登录功能模块表

用例编号	A22	优先级	2	用例级别	一般
用例摘要	测试用户离线登录功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-03	开发人员	黄健楸
前置条件	无网络，带登录状态进入应用				
测试方法	手工测试				
输入数据	输入测试用例中各用户账户以及对应密码				
执行步骤	点击提交				
预期输出	登录时出现带有“很快就好”字样的进度条。若登录成功，跳转至主页面；反之，弹出“注册失败”				
实际结果	符合预期				
测试日期	2019-04-03				
结论	■ 通过		□ 未通过		

5.2.2 用户详情

表 5.5 测试用户详情页功能模块表

用例编号	A3	优先级	1	用例级别	重要
用例摘要	测试用户详情页功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-11	开发人员	黄健楸
前置条件	各种用户成功进入应用主界面				
测试方法	手工测试				
输入数据	无				
执行步骤	点击主页面底部栏“我”按钮				
预期输出	页面正确显示姓名、邮箱、类型、ID、组织（若无组织则不显示）等信息				
实际结果	各用户的用户详情页均符合预期				
测试日期	2019-04-11				
结论	■ 通过		□ 未通过		

5.2.3 组织列表页

表 5.6 测试用户列表页功能模块表

用例编号	A4	优先级	1	用例级别	重要
用例摘要	测试用户列表页功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-14	开发人员	黄健楸
前置条件	各种用户成功进入应用主界面后				
测试方法	手工测试				
输入数据	无				
执行步骤	点击主页面底部栏“组织”按钮				
预期输出	不同用户显示不同的用户。管理员显示所有用户；管理部门显示所有公司用户；公司和普通用户显示该公司下所有公司和普通用户				
实际结果	各用户的用户列表页均符合预期				
测试日期	2019-04-14				
结论	■ 通过		□ 未通过		

5.3 任务

5.3.1 各用户任务列表页

表 5.7 测试用户列表页功能模块表

用例编号	B11	优先级	1	用例级别	重要
用例摘要	测试用户列表页功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-18	开发人员	黄健楸
前置条件	各种用户成功进入应用主界面后，即可见任务列表页				
测试方法	手工测试				
输入数据	无				
执行步骤	无				
预期输出	不同用户显示不同的任务。管理员和管理部门显示所有任务；公司和普通用户显示与自己相关的任务				
实际结果	各用户的任务列表页均符合预期				
测试日期	2019-04-18				
结论	■ 通过		□ 未通过		

5.3.2 各用户任务详情页

表 5.8 测试用户列表页功能模块表

用例编号	B12	优先级	1	用例级别	重要
用例摘要	测试用户列表页功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-04-18	开发人员	黄健楸
前置条件	各种用户成功进入应用主界面后，即可见任务列表页				
测试方法	手工测试				
输入数据	无				
执行步骤	无				
预期输出	各种用户基本都能见到相关任务的所有数据，但管理员和管理部门不能见到任务的执行者。				
实际结果	各用户的任务列表页均符合预期				
测试日期	2019-05-12				
结论	■ 通过		□ 未通过		

5.3.3 管理部门创建任务

表 5.9 测试管理部门用户创建任务功能模块表

用例编号	B21	优先级	1	用例级别	重要
用例摘要	测试管理部门用户创建任务功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-01	开发人员	黄健楸
前置条件	在线，管理部门用户成功进入应用主界面后，点击右下角浮动按钮				
测试方法	手工测试				
输入数据	任务标题、类型、详情、执行公司				
执行步骤	点击创建或取消				
预期输出	创建时，若输入数据符合要求，则弹出带成功消息的 Toast，并返回任务列表页，发现列表页也已经更新。取消则直接返回列表页。				
实际结果	各项测试均符合预期				
测试日期	2019-05-01				
结论	■ 通过		□ 未通过		

5.3.4 管理部门修改任务指派内容

表 5.10 测试管理部门任务指派功能模块表

用例编号	B23	优先级	2	用例级别	一般
用例摘要	测试管理部门任务指派功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-02	开发人员	黄健楸
前置条件	在线，管理部门用户点击某一任务的任务详情页的编辑菜单按钮				
测试方法	手工测试				
输入数据	点击相关用户				
执行步骤	点击确认或取消				
预期输出	有选择任务时，确认时返回详情页，弹出成功信号，且执行公司已更改；未选择任务时，弹出提醒信号。取消则直接返回详情页。				
实际结果	各项测试均符合预期				
测试日期	2019-05-02				
结论	■ 通过		□ 未通过		

5.3.5 公司负责人将任务状态从创建改为执行

表 5.11 测试公司用户修改任务状态功能模块表 1

用例编号	B241	优先级	1	用例级别	重要
用例摘要	测试公司用户修改任务状态功能模块 1				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-05	开发人员	黄健楸
前置条件	在线，公司用户点击某一处于创建状态的任务的任务详情页的修改状态菜单按钮				
测试方法	手工测试				
输入数据	选择执行按钮				
执行步骤	选择执行按钮				
预期输出	弹出成功信号				
实际结果	弹出成功信号				
测试日期	2019-05-05				
结论	■ 通过		□ 未通过		

5.3.6 公司负责人修改任务执行者

表 5.12 测试公司用户修改任务执行者功能模块

用例编号	B25	优先级	1	用例级别	重要
用例摘要	测试公司用户修改任务执行者功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-05	开发人员	黄健楸
前置条件	在线，公司用户点击某一任务的任务详情页的相关按钮				
测试方法	手工测试				
输入数据	点击相关用户				
执行步骤	点击确认或取消				
预期输出	若有更改时，返回成功信息，返回更新后的详情页；反之返回无更改				
实际结果	各项测试均符合预期				
测试日期	2019-05-05				
结论	■ 通过		□ 未通过		

5.3.7 执行者上传照片

表 5.13 测试员工用户上传照片功能模块表

用例编号	B261	优先级	1	用例级别	重要
用例摘要	测试员工用户上传照片功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-12	开发人员	黄健楸
前置条件	在线，员工用户在某一任务详情页，点击右下角的浮动按钮，后点击上传照片按钮。				
测试方法	手工测试				
输入数据	相片文件、类型、简要介绍，				
执行步骤	点击图库中照片或拍照，在图片列表中输入或选择信息，点击上传				
预期输出	返回成功信息与更新后的详情页，右下角浮动按钮仍处于打开状态				
实际结果	基本符合预期，但详情页要稍后才能更新				
测试日期	2019-05-14				
结论	■ 通过		□ 未通过		

5.3.8 执行者上传定位

表 5.14 测试员工用户上传定位功能模块表

用例编号	B262	优先级	1	用例级别	重要
用例摘要	测试员工用户上传定位功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-12	开发人员	黄健楸
前置条件	在线，员工用户在某一任务详情页，点击右下角的浮动按钮，后点击获取定位按钮。				
测试方法	手工测试				
输入数据	无				
执行步骤	待定位稳定后，点击上传				
预期输出	返回成功信息与更新后的详情页，右下角浮动按钮仍处于打开状态				
实际结果	符合预期				
测试日期	2019-05-12				
结论	■ 通过		□ 未通过		

5.3.9 执行者上传巡查路线

表 5.15 测试员工用户上传巡查路线功能模块表

用例编号	B263	优先级	1	用例级别	重要
用例摘要	测试员工用户上传巡查路线功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-12	开发人员	黄健楸
前置条件	在线，员工用户在某一任务详情页，点击右下角的浮动按钮，后点击巡查路线按钮。				
测试方法	手工测试				
输入数据	无				
执行步骤	巡查前点击开始，结束点击结束，检查后点击上传				
预期输出	返回成功信息与更新后的详情页，右下角浮动按钮仍处于打开状态				
实际结果	符合预期				
测试日期	2019-05-12				
结论	■ 通过		□ 未通过		

5.3.10 公司负责人将任务状态从执行改为待测

表 5.16 测试公司用户修改任务状态功能模块表 2

用例编号	B242	优先级	1	用例级别	重要
用例摘要	测试公司用户修改任务状态功能模块 2				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-05	开发人员	黄健楸
前置条件	在线，在某一上传完任务结果且处于执行状态的任务的任务详情页公司用户点击修改状态菜单按钮				
测试方法	手工测试				
输入数据	选择待测按钮				
执行步骤	选择待测按钮				
预期输出	弹出成功信号				
实际结果	弹出成功信号				
测试日期	2019-05-12				
结论	■ 通过		□ 未通过		

5.3.11 管理部门将任务状态从待测改为完成

表 5.17 测试公司用户修改任务执行者功能模块表

用例编号	B243	优先级	1	用例级别	重要
用例摘要	测试管理部门修改任务状态功能模块				
测试类型	功能测试				
用例类型	常规用例数据				
设计者	黄健楸	设计日期	2019-05-05	开发人员	黄健楸
前置条件	在线，在某一处于待测状态的任务的任务详情页界面，管理部门用户点击修改状态菜单按钮				
测试方法	手工测试				
输入数据	选择完成按钮				
执行步骤	选择完成按钮				
预期输出	弹出成功信号				
实际结果	弹出成功信号				
测试日期	2019-05-12				
结论	■ 通过		□ 未通过		

5.4 测试总结

本章通过拆解工作流程，构造出对各种基本功能的测试，而本软件已通过这些测试，可见本软件已实现了各项基本功能。

6 费用匡算

本软件的开发费用为 47 千元。该费用主要由四个部分组成：基本功能，功能测试，后续功能与测试，运行维护四个部分组成。基本功能花费主要包括开发人员的工资，每月八千元，共三月，人数为一，合计 24 千元；功能测试每次花费一千元，共四次，加上测试设备需要花费一千元，共五千元；后续功能与测试花费包括开发费用和相关测试费用，预计需两个月，每月需给一人七千元工资，合计 14 千元；运行维护每次花费一千元，每季度一次，每年共 4 千元。各部分明细可参考表 6.1 费用初步匡算表 。

表 6.1 费用初步匡算表

费用分类	费用（千元）	备注
基本功能	24	不包括远程服务器
功能测试	5	
后续功能与测试	14	
运行维护	4	一年
合计	47	

7 总结与展望

本文完成的主要工作有：

1. 简要分析项目背景，详细分析各种需求，阐明选择的架构与理由；
2. 根据各种需求设计简洁美观的交互界面；
3. 根据远端服务器接口，设计简单的本地缓存，并协调这两个数据源；
4. 详细介绍各功能模块的要点与主要注意的细节；
5. 详细列举各功能测试以及测试结果。

本软件实现了场地调查的初步调查模块，今后仍需完善以下方面的内容：

1. 添加其他类型的任务，如初布点、采样等；
2. 考虑为不同的用户开发对应的应用，进一步降低该应用的耦合性；
3. 获取反馈，进一步细化需求，如任务内容加密、用户头像等。

参考文献

[1] 谷歌安卓开发团队. 应用架构指南[EB/OL]. https://developer.android.com/jetpack/docs/guide#top_of_page

[2] 史伟玲.污染场地环境调查现场采样技术现状及存在问题的探讨[J].科技风,2019(07):115-116.

[3] 刘光逊,西伟力,尤学一.场地环境调查监测数据管理系统需求分析[J].资源节约与环保,2019(01):36+49.

[4] 吴俭,邓一荣,林彰文,肖荣波,苏嘉韵.广州市污染场地环境管理对策研究[J].环境科学与管理,2018,43(12):6-9.

[5] 岳群. 基于安卓的犯罪人员签到管理系统的设计与实现[D].吉林大学,2018.

[6] 孙述海,李鹏飞.疑似污染场地土壤环境调查方法研究[J].吉林地质,2018,37(04):67-70.

[7] 孙亚琦. 医院挂号 APP 的设计与实现[D].南京理工大学,2018.

[8] Crawley E, Cameron B, Selva D. System architecture: strategy and product development for complex systems[M]. Prentice Hall Press, 2015.

[9] Neil T. Mobile design pattern gallery: UI patterns for smartphone apps[M]. " O'Reilly Media, Inc.", 2014.

[10] Nudelman G. Android design patterns: interaction design solutions for developers[M]. John Wiley & Sons, 2013.

致谢

感谢王孝武老师给我这次机会，参与到这个软件开发的项目中来。感谢本学院的江明同学，是他提供的后台接口使这个安卓软件不是一个空壳。感谢本校计算机科学与工程学院的林毅同学与华南农业大学数学与信息学院的沈晓文同学，他们曾在相关技术层面上予本人很多帮助。

附录

限于篇幅，仅附具有代表性的几个功能的相关代码，且略过其中不重要的部分。全部可公开代码见 https://github.com/linxixizhi/taskRecord。

登录

```
public class LoginActivity extends AppCompatActivity {
    private static final String TAG = "LoginActivity";
    private static final int REQUEST SIGNUP = 0;
    private static final String PLEASELOGIN = "请登录";
    private static final String REMEMBER PASSWORD =
"remember password";
    private static final String AUTO LOGIN = "auto login";
    private static final String EMAIL = "email";
    private static final String EMAIL OR NAME = "email or name";
    private static final String PASSWORD = "password";
    private static final String USER STRING = "user";
    @BindView(R.id.input email) EditText  emailText;
    @BindView(R.id.input password) EditText  passwordText;
    @BindView(R.id.btn login) Button  loginButton;
    @BindView(R.id.link signup) TextView  signupLink;
    @BindView(R.id.remenberPasswordCheckBox) CheckBox
 remenberPasswordCheckBox;
    @BindView(R.id.autoLoginCheckbox) CheckBox
 autoLoginCheckbox;
    private ProgressDialog progressDialog;
    private SharedPreferences prefLoginSetting;
    private SharedPreferences.Editor editor;
    private UserViewModel userViewModel;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity login);
        ButterKnife.bind(this);
        prefLoginSetting = AppPreferencesHelper.getLoginPref();
        boolean isRemember =
prefLoginSetting.getBoolean(REMEMBER PASSWORD, false);
        boolean isAutoLogin =
prefLoginSetting.getBoolean(AUTO LOGIN, false);
        //isRemember
        if (isRemember){
            String emailOrName =
prefLoginSetting.getString(EMAIL OR NAME, "");
            String password =
prefLoginSetting.getString(PASSWORD, "");
             emailText.setText(emailOrName);
             passwordText.setText(password);
             remenberPasswordCheckBox.setChecked(true);
```

```
            if (isAutoLogin){
                    loginButton.setEnabled(false);
                    autoLoginCheckbox.setChecked(true);
                   login(emailOrName, password);
                }
            }
        }

        @OnClick(R.id.btn login)
        void login(){
            final String email =  emailText.getText().toString();
            final String password =
 passwordText.getText().toString();
            login(email, password);
        }

        void login(String email, String password) {
             loginButton.setEnabled(false);
            Log.d(TAG, "Login");

            if (!validate(email, password)) {
                onLoginFailed(null);
                return;
            }

            progressDialog = new ProgressDialog(LoginActivity.this,
                    R.style.AppTheme Dark Dialog);

            progressDialog.setIndeterminate(true);
            progressDialog.setMessage(SOON);
            progressDialog.show();

            RequestBody requestBody = new FormBody.Builder()
                    .add(PASSWORD, password)
                    .add(USER STRING, email)
                    .build();

            HttpUtil.POST(LOGIN, requestBody, new Callback() {
                @Override
                public void onFailure(@NonNull Call call, @NonNull
IOException e) {
                    prefLoginSetting =
AppPreferencesHelper.getLoginPref();
                    if
(email.equals(prefLoginSetting.getString(EMAIL OR NAME, null))&&
password.equals(prefLoginSetting.getString(PASSWORD, null))){
                        Log.d(TAG, "Offline Login Successed");
                        onLoginSuccess();
                    } else {
                        onLoginFailed(null);
                    }
                    progressDialog.dismiss();
                }
```

```
            @Override
            public void onResponse(@NonNull Call call, @NonNull Response
response) throws IOException {
                String responseData = response.body().string();
                progressDialog.dismiss();
                Pair<Integer, User> codeAndMe =
JsonParser.parseLoginJson(responseData);
                int statusCode = codeAndMe.a;
                User me = codeAndMe.b;
                if (me != null) {
                    AppPreferencesHelper.saveUser(me);
                    ((BasicApp)getApplication()).getUserRepository().insert(me);
                }
                switch (statusCode) {
                    case SUCCESS:
                        Log.d(TAG, "Successed");
                        onLoginSuccess();
                        break;
                    case FAIL:
                        onLoginFailed(null);
                        break;
                    case MISSINGREQ:
                        onLoginFailed("Missing request");
                        break;
                    case INVALIDREQ:
                        onLoginFailed("Invalid request");
                        break;
                    default:
                        break;
                }
            }
        });
    }

    @OnClick(R.id.link_signup)
    void startSignup(){
        Intent intent = new Intent(getApplicationContext(),
SignupActivity.class);
        startActivityForResult(intent, REQUEST_SIGNUP);
    }

    @OnCheckedChanged(R.id.remenberPasswordCheckBox)
    void RememberUnchecked(boolean isChecked){
        if (!isChecked) _autoLoginCheckbox.setChecked(false);
    }

    @OnCheckedChanged(R.id.autoLoginCheckbox)
    void AutoLoginChecked(boolean isChecked){
        if (isChecked) _remenberPasswordCheckBox.setChecked(true);
    }
```

```
    @Override
    protected void onPause() {
        super.onPause();
        if (progressDialog != null){
            progressDialog.dismiss();
            progressDialog = null;
        }
    }

    @Override
    protected void onStop() {
        prefLoginSetting = AppPreferencesHelper.getLoginPref();
        editor = prefLoginSetting.edit();
        if (_remenberPasswordCheckBox.isChecked()){
            String email = _emailText.getText().toString();
            String password = _passwordText.getText().toString();
            editor.putBoolean(REMEMBER_PASSWORD, true);
            editor.putString(EMAIL OR NAME, email);
            editor.putString(PASSWORD, password);
        } else {
            editor.putBoolean(REMEMBER_PASSWORD, false);
            editor.remove(EMAIL OR NAME);
            editor.remove(PASSWORD);
        }

        if (_autoLoginCheckbox.isChecked()){
            editor.putBoolean(AUTO_LOGIN, true);
        } else {
            editor.putBoolean(AUTO_LOGIN, false);
        }
        editor.apply();
        super.onStop();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        if (requestCode == REQUEST_SIGNUP) {
            if (resultCode == RESULT OK) {
                // By default we just finish the Activity and log them in
automatically
            Toast.makeText(getBaseContext(), PLEASELOGIN,
Toast.LENGTH_SHORT).show();
            }
        }
    }

    @Override
    public void onBackPressed() {
        moveTaskToBack(true);
    }
```

```
    public void onLoginSuccess(){
        Handler mHandler = new Handler(Looper.getMainLooper());
        mHandler.postDelayed(() -> runOnUiThread(() -> {
            _loginButton.setEnabled(true);
            Toast.makeText(getBaseContext(), YOU_LOGIN_SUCCESS,
Toast.LENGTH_LONG).show();
        }), 100);
        openMainActivity();
    }

    public void onLoginFailed(String message) {
        final String message_final;
        if (message == null){
            message final = YOU LOGIN FAILED;
        } else {
            message_final = message;
        }
        Handler mHandler = new Handler(Looper.getMainLooper());
        mHandler.postDelayed(() -> runOnUiThread(() -> {
            _loginButton.setEnabled(true);
            Toast.makeText(getBaseContext(), message_final,
Toast.LENGTH_LONG).show();
        }), 100);
    }

    public boolean validate(String email, String password) {
        boolean valid = true;
        if (email.isEmpty() || email.length() < 3) {

_emailText.setError(getResources().getString(R.string.valid_name));
            valid = false;
        } else {
            _emailText.setError(null);
        }
        if (password.isEmpty() || password.length() < 6 ||
password.length() > 15) {

_passwordText.setError(getResources().getString(R.string.valid_password));
            valid = false;
        } else {
            _passwordText.setError(null);
        }
        return valid;
    }

    public void openMainActivity() {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        finish();
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:fitsSystemWindows="true">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="56dp"
        android:paddingLeft="27dp"
        android:paddingRight="27dp">
        <include layout="@layout/app_mainline"
            android:id="@+id/app_login_mainline" />
        <com.google.android.material.textfield.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp">
            <EditText android:id="@+id/input_email"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textEmailAddress"
                android:hint="@string/email" />
        </com.google.android.material.textfield.TextInputLayout>
        <com.google.android.material.textfield.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp">
            <EditText android:id="@+id/input_password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textPassword"
                android:hint="@string/passwd"/>
        </com.google.android.material.textfield.TextInputLayout>
        <androidx.constraintlayout.widget.ConstraintLayout
              android:layout_width="match_parent"
              android:layout_height="wrap_content">
```

```
            <androidx.appcompat.widget.AppCompatCheckBox
                android:id="@+id/remenberPasswordCheckBox"
                android:layout_width="wrap_content"
                android:layout height="wrap_content"
                android:layout_marginStart="8dp"
                app:layout_constraintStart_toStartOf="parent"
                app:layout constraintTop toTopOf="parent" />
            <androidx.appcompat.widget.AppCompatCheckedTextView
                android:layout_width="wrap_content"
                android:layout height="wrap_content"
                android:layout_marginStart="3dp"
                android:layout_marginTop="8dp"
                android:text="@string/remember_password"
app:layout_constraintStart_toEndOf="@+id/remenberPasswordCheckBox"
                app:layout_constraintTop_toTopOf="parent" />
            <androidx.appcompat.widget.AppCompatCheckBox
                android:id="@+id/autoLoginCheckbox"
                android:layout_width="wrap_content"
                android:layout height="wrap_content"
                android:layout_marginEnd="3dp"
app:layout_constraintEnd_toStartOf="@+id/autoLoginCheckboxTextView"
                app:layout constraintTop toTopOf="parent" />
            <androidx.appcompat.widget.AppCompatCheckedTextView
                android:id="@+id/autoLoginCheckboxTextView"
                android:layout width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="7dp"
                android:layout marginEnd="16dp"
                android:text="@string/auto_login"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout constraintTop toTopOf="parent" />
        </androidx.constraintlayout.widget.ConstraintLayout>
        <androidx.appcompat.widget.AppCompatButton
            android:id="@+id/btn_login"
            android:layout width="fill_parent"
            android:layout_height="wrap_content"
            android:layout marginTop="24dp"
            android:layout_marginBottom="24dp"
            android:padding="5dp"
            android:textColor="@drawable/button_text_color"
            android:background="@drawable/button_color"
            android:text="@string/login"
            android:textSize="20sp"
            />
        <TextView android:id="@+id/link_signup"
            android:layout width="fill_parent"
            android:layout height="wrap_content"
            android:layout_marginBottom="24dp"
            android:text="@string/no_account"
            android:textColor="@color/secondary"
            android:gravity="center"
            android:textSize="15sp"/>
    </LinearLayout>
</ScrollView>
```

后续功能支持类

```
@Dao
public interface TaskDao {
    @Query("SELECT * FROM tasks")
    LiveData<List<Task>> getAllTasks();

    @Query("SELECT * FROM tasks where assignee id = :assigneeId")
    LiveData<List<Task>> getCompanyTasks(int assigneeId);

    @Query("SELECT * FROM tasks where assigner id = :assignerId")
    LiveData<List<Task>> getManagerTasks(int assignerId);

    @Query("SELECT * FROM tasks where executor = :userId")
    LiveData<List<Task>> getUserTasks(int userId);

    @Query("SELECT * FROM tasks WHERE taskID = :Id")
    LiveData<Task> getTask(int Id);

    @Query("SELECT tasks.* From tasks JOIN tasksFts ON (tasks.taskID
= tasksFts.rowid) " +
            "WHERE tasksFts MATCH :query")
    LiveData<List<Task>> searchAllTasks(String query);

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertAll(List<Task> tasks);

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insert(Task task);

    @Update
    void update(Task task);

    @Delete
    void delete(Task task);

    @Query("UPDATE tasks SET status = :mStatus WHERE taskID
= :taskId")
    void updateStatus(String taskId, boolean mStatus);

    @Query("DELETE FROM tasks")
    void deleteAllTasks();

    @Query("DELETE FROM tasks WHERE status
= :TaskFinishStatusNumber")
    int deleteCompletedTasks(int TaskFinishStatusNumber);
}
```

```
@Entity(tableName = "tasks")
public class Task implements Serializable {
    @PrimaryKey
    private int taskID;
    private String title;
    private String description;
    private int createAt;
    private int updateAt;
    private int finishAt;
    private int type;
    @Ignore
    private TaskResult mTaskResult;
    @Ignore
    private int resource;
    @Ignore
    private User assignee;
    @Expose(serialize = false, deserialize = false)
    private int assignee id;
    @Ignore
    private User assigner;
    @Expose(serialize = false, deserialize = false)
    private int assigner id;
    @Expose(serialize = false, deserialize = false)
    private int executor;
    private int status;

    public Task(){
    }

    @Ignore
    public Task(int taskID, String title, String description, User
assigner,
                User assignee, int type, TaskResult taskResult, int
resource, int createAt,
                int updateAt, int finishAt, int status){
        this.taskID = taskID;
        this.title = title;
        this.description = description;
        this.assignee = assignee;
        this.assignee id = (assignee == null) ? -1 :
assigner.getUid();
        this.assigner = assigner;
        this.assignee id = (assignee == null) ? -1 :
assigner.getUid();
        this.type = type;
        this.mTaskResult = taskResult;
        this.resource = resource;
        this.createAt = createAt;
        this.updateAt = updateAt;
        this.finishAt = finishAt;
        this.status = status;
    }
```

```
    @Ignore
    public Task(String title, String description, int assigner,
                int assignee, int type){
        this.title = title;
        this.description = description;
        this.assignee id = assignee;
        this.assigner id = assigner;
        this.type = type;
    }

    public Task(Task task){
        taskID = task.getTaskID();
        title = task.getTitle();
        description = task.getDescription();
        assignee = task.getAssignee();
        assigner = task.getAssigner();
        assignee id = task.getAssignee id();
        assigner id = task.getAssigner id();
        executor = task.getExecutor();
        type = task.getType();
        mTaskResult = task.getTaskResult();
        resource = task.getResource();
        createAt = task.getCreateAt();
        updateAt = task.getUpdateAt();
        finishAt = task.getFinishAt();
        status = task.getStatus();
    }
    // 忽略 setter 和 getter 方法百行。

}

@Dao
public interface PhotoDao {
    @Query("SELECT * FROM PhotoResult WHERE taskID = :taskID")
    LiveData<List<PhotoResult>> getPhotoResultByTaskID(int taskID);

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void insertAll(List<PhotoResult> photoResults);
}
```

```
public class AppPreferencesHelper {
    private static final String TAG = "AppPreferencesHelper";
    public static SharedPreferences getLoginPref(){
        return
PreferenceManager.getDefaultSharedPreferences(BasicApp.getAppContext(
));
    }
    public static SharedPreferences getLoginPref(Context context){
        return PreferenceManager.getDefaultSharedPreferences(context);
    }
    public static SharedPreferences getCookiePref(){
        return BasicApp.getAppContext().getSharedPreferences(
               "CookiePersistence", Context.MODE_PRIVATE);
    }
    public static User getCurrentUser(){
        User me = null;
        String userJson = getLoginPref().getString(LOGIN_PREF, null);
        if (userJson != null) {
            me = UserConverter.toUserEntity(userJson);
        }
        return me;
    }
    public static int getCurrentUserID(){
        User me = getCurrentUser();
        int ID = -1;
        if (me != null)
            ID = me.getUid();
        return ID;
    }
    public static int getCurrentUserLoginState(){
        int logoutState =
LoginMode.LOGGED IN MODE LOGGED OUT.getType();
        String userJson = getLoginPref().getString(LOGIN PREF, null);
        if (userJson != null) {
            User me = getCurrentUser();
            if (me != null) logoutState = me.getType();
        }
        return logoutState;
    }
    public static void saveUser(User user){
        String currentUser = UserConverter.toJson(user);
        SharedPreferences.Editor editor = getLoginPref().edit();
        editor.putString(LOGIN PREF, currentUser);
        editor.apply();
    }
    public static void clearLoginPref(){
        SharedPreferences.Editor editor = getLoginPref().edit();
        editor.clear();
        editor.apply();
    }
    public static void clearCookiePref(){
        SharedPreferences.Editor editor = getCookiePref().edit();
        editor.clear();
        editor.apply();
    }
}
```

```
public interface BigkeerService {
    @GET("Session")
    Call<ResponseBody> getSession();

    @FormUrlEncoded
    @POST(Urls.LOGIN)
    LiveData<ApiResponse<BigkeerResponse<User>>>
    login(@Field("user") String username, @Field("password") String
password);

    @GET(Urls.GET_USER_ME)
    LiveData<ApiResponse<BigkeerResponse<User>>> getUserMe();

    @GET("User")
    LiveData<ApiResponse<BigkeerResponse<ArrayResult<User>>>>
getAllUsers();

    @GET(COMPANY_GET_USERS_BY_GROUP)
    LiveData<ApiResponse<BigkeerResponse<GroupInfoResult>>>
    companyGetUsersInGroup();

    @GET(USER GET USERS BY GROUP)
    LiveData<ApiResponse<BigkeerResponse<GroupInfoResult>>>
    userGetUsersInGroup();

    @GET("User/search/{query}")
    Call<ResponseBody> searchUserByQuery(@Path("query") String
query);

    @GET("User/{uid}")
    LiveData<ApiResponse<BigkeerResponse<User>>>
getUserById(@Path("uid") int uid);

    @GET("Task/all")
    LiveData<ApiResponse<BigkeerResponse<ArrayResult<Task>>>>
    getAllTasks();

    @GET("Task/{taskID}")
    LiveData<ApiResponse<BigkeerResponse<Task>>>
    getTaskByID(@Path("taskID") int taskID);

    @GET("Task/all")
    LiveData<ApiResponse<BigkeerResponse<ArrayResult<Task>>>>
    adminGetTasks();

    @GET(GET USER TASKS)
    LiveData<ApiResponse<BigkeerResponse<ArrayResult<Task>>>>
    getUserTasks();

    @GET(GET COMPANY TASKS)
    LiveData<ApiResponse<BigkeerResponse<ArrayResult<Task>>>>
    getCompanyTasks(@Path(COMPANY ID IN TASK) int companyID);

    @GET(GET MANAGER TASKS)
    LiveData<ApiResponse<BigkeerResponse<ArrayResult<Task>>>>
    getManagerTasks(@Path(MANAGER_ID_IN_TASK) int managerID);
```

```
    @POST("Task")  // manager
    Call<ResponseBody> createTask();

    @PUT("Task")  // manager
    Call<ResponseBody> managerEditTask();

    @PATCH("Task/{taskID}/status")  // manager, company
    Call<ResponseBody> patchTaskStatus();

    @PATCH("Task/{taskID}/result")  // company
    Call<ResponseBody> patchTaskResult();

    @POST("Assign/{taskID}")  // company
    Call<ResponseBody> companyAssignUser();

    @GET("Track/{trackID}")
    Call<ResponseBody> getTrack(@Path("trackID") int trackID);

    @GET("Track/task/{taskID}")
    Call<ResponseBody> getTrackByTask(@Path("tasckID") int trackID);

    @POST("Track")  // user, return trackID
    Call<ResponseBody> userCreateTrack();

    @DELETE("Track/{trackID}")
    Call<ResponseBody> deleteTrack(@Path("trackID") int trackID);


    @GET("Photo/task/{taskID}")
    LiveData<ApiResponse<BigkeerResponse<List<PhotoResult>>>>
    getPhotoResultsByTaskID(@Path("taskID") int taskID);


    @POST(GET OR CREATE TASKS)
    Call<ResponseBody> createTask(@Body RequestBody requestBody);

    @Multipart
    @POST("Photo")
    Call<ResponseBody> postImage(@Part MultipartBody.Part file,
                                 @Part(PHOTO TASKID) RequestBody
taskID,
                                 @Part(PHOTO SUBID) RequestBody type,
                                 @Part(PHOTO TIME) RequestBody time,
                                 @Part(PHOTO LOCATION) RequestBody
location,
                                 @Part(PHOTO DESC) RequestBody
description);

    @GET("Explore/{taskID}")
    LiveData<ApiResponse<LocCenterPoint>>
    getLocCenterPointByTaskID(@Path("taskID") int taskID);


    @GET("Track/task/{taskID}")
    LiveData<ApiResponse<BigkeerResponse<List<Tracks>>>>
    getTracksByTaskID(@Path("taskID") int taskID);
}
```

```
public abstract class NetworkBoundResource<ResultType, RequestType> {
    // Called to save the result of the API response into the database.
    private final MediatorLiveData<Resource<ResultType>> result;
    private final AppExecutors appExecutors;

    @MainThread
    public NetworkBoundResource(@NonNull AppExecutors appExecutors){
        this.appExecutors = appExecutors;
        result = new MediatorLiveData<>();
        result.setValue(Resource.loading(null));
        final LiveData<ResultType> dbSource = this.loadFromDb();
        result.addSource(dbSource, data -> {
            result.removeSource(dbSource);
            if (shouldFetch(data)){
                fetchFromNetwork(dbSource);
            } else {
                result.addSource(dbSource, o ->
                        result.setValue(Resource.success(data)));
            }
        });
    }

    private void fetchFromNetwork(final LiveData<ResultType> dbSource) {
        final LiveData<ApiResponse<RequestType>> apiResponse =
this.createCall();
        result.addSource(dbSource, newData ->
                result.setValue(Resource.loading(newData)));
        result.addSource(apiResponse, response -> {
            result.removeSource(apiResponse);
            result.removeSource(dbSource);
            if (response instanceof ApiEmptyReponse) {
                appExecutors.mainThread().execute(()
                        -> result.addSource(loadFromDb(), newData
                        -> result.setValue(Resource.success(newData))));
            } else if (response instanceof ApiSussessResponse){
                appExecutors.diskIO().execute(() -> {
                    saveCallResult(((ApiSussessResponse<RequestType>)
response).getBody());
                    appExecutors.mainThread().execute(() ->
result.addSource(loadFromDb(),
                            newData ->
result.setValue(Resource.success(newData))));
                });
            } else if (response instanceof ApiErrorResponse) {
                onFetchFailed();
                result.addSource(dbSource, newData ->
                        result.setValue(Resource.error(
                                ((ApiErrorResponse<RequestType>)
response).getErrorMessage(),
                                newData)
                        ));
            }
        });
    }
```

```
    @MainThread
    protected void onFetchFailed(){}

    public final LiveData<Resource<ResultType>> getAsLiveData() {
        MediatorLiveData<Resource<ResultType>> result = this.result;
        if (result == null)
            throw new RuntimeException("null cannot be cast to non-null type
" +"androidx.lifecycle.LiveData<Resource<ResultType>>");
        return result;
    }

    @WorkerThread
    protected abstract void saveCallResult(@NonNull RequestType item);

    @MainThread
    protected abstract boolean shouldFetch(@Nullable ResultType data);

    @NonNull @MainThread
    protected abstract LiveData<ResultType> loadFromDb();


    @MainThread
    public abstract LiveData<ApiResponse<RequestType>> createCall();
}
public class ApiEmptyReponse<T> extends ApiResponse<T> {
    public ApiEmptyReponse(){
    }
}

public class ApiErrorResponse<T> extends ApiResponse<T> {
    @NonNull
    private final String errorMessage;
    @NonNull
    public final String getErrorMessage() {
        return this.errorMessage;
    }

    public <T>ApiErrorResponse(@NonNull String  errorMessage){
        this.errorMessage = errorMessage;
    }
}
```

```
public class ApiSussessResponse<T> extends ApiResponse<T> {
    private T body;
    public ApiSussessResponse (T body){
        this.body = body;
    }
    public T getBody() {
        return body;
    }
}

public abstract class ApiResponse<T> {
    @NonNull
    public ApiErrorResponse<T> create(@NonNull Throwable error){
        return new ApiErrorResponse<T>(error.getMessage());
    }

    @NonNull
    public ApiResponse<T> create(@NonNull Response<T> response){
        ApiResponse<T> apiResponse;
        if (response.isSuccessful()){
            T body = response.body();
            if (body == null || response.code() == 204){
                apiResponse = new ApiEmptyReponse<T>();
            } else {
                Headers header = response.headers();
                apiResponse = new ApiSussessResponse<T>(body);
            }
        } else {
            ResponseBody responseBody = response.errorBody();
            String msg = null;
            try{
                if (responseBody != null){
                    msg = responseBody.string();
                }
            } catch (IOException ioe){
                ioe.printStackTrace();
            }
            String errMsg = (msg == null || msg.isEmpty()) ?
response.message() : msg;
            apiResponse = new ApiErrorResponse<T>(errMsg);
        }
        return apiResponse;
    }
}
```

```
public class Resource<T> {
    @NonNull public final Status status;
    @Nullable public final T data;
    @Nullable public final String message;
    private Resource(@NonNull Status status, @Nullable T data,
                     @Nullable String message) {
        this.status = status;
        this.data = data;
        this.message = message;
    }
    public static <T> Resource<T> success(@NonNull T data) {
        return new Resource<>(Status.SUCCESS, data, null);
    }
   public static <T> Resource<T> error(String msg, @Nullable T data)
   {return new Resource<>(Status.ERROR, data, msg);}
    public static <T> Resource<T> loading(@Nullable T data) {
        return new Resource<>(Status.LOADING, data, null);
    }
    public enum Status { SUCCESS, ERROR, LOADING }
}

public final class LiveDataCallAdapter<R> implements CallAdapter<R,
LiveData<ApiResponse<R>>> {
    private final Type responseType;
    public LiveDataCallAdapter(@NonNull Type responseType) {
        this.responseType = responseType;
    }
    @NonNull
    public Type responseType() {
        return this.responseType;
    }
    @NonNull
    public LiveData<ApiResponse<R>> adapt(@NonNull final Call<R>
call) {
        return (new LiveData<ApiResponse<R>>() {
            private AtomicBoolean started = new AtomicBoolean(false);

            protected void onActive() {
                super.onActive();
                if (started.compareAndSet(false, true)) {
                    ApiResponse<R> apiResponse = new ApiResponse<R>()
{};
                    call.enqueue((new Callback<R>() {
                        public void onResponse(@NonNull Call<R> callx,
@NonNull Response<R> response) {
                            postValue(apiResponse.create(response));
                        }

                        public void onFailure(@NonNull Call<R> callx,
@NonNull Throwable throwable) {
                            postValue(apiResponse.create(throwable));
                        }
                    }));
                }

            }
        });
    }
}
```

```
public final class LiveDataCallAdapterFactory extends Factory {

    @Nullable
    public CallAdapter get(@NonNull Type returnType, @NonNull
Annotation[] annotations,
                           @NonNull Retrofit retrofit) {
        if (!Objects.equals(Factory.getRawType(returnType),
LiveData.class)) {
            return null;
        } else if (!(returnType instanceof ParameterizedType)) {
            throw new IllegalArgumentException("returnType must be
parameterized");
        } else {
            Type observableType = Factory.getParameterUpperBound(0,
                    (ParameterizedType) returnType);
            Class rawOservableType =
Factory.getRawType(observableType);
            if (!Objects.equals(rawOservableType, ApiResponse.class)) {
                throw new IllegalArgumentException("type must be a
resource");
            } else if(!(observableType instanceof ParameterizedType)) {
                throw new IllegalArgumentException("resource must be
parameterized");
            } else {
                Type bodyType = getParameterUpperBound(0,
(ParameterizedType) observableType);
                return new LiveDataCallAdapter(bodyType);
            }
        }
    }
}
```

任务列表页

```
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "MainActivity";
    @BindView(R.id.main bottom navigation) BottomNavigationView
mBottomNavigationView;
    final Fragment mTaskListFragment = new TaskListFragment();
    final Fragment mUserGroupFragment = new UserGroupFragment();
    final Fragment mUserMeFragment = new UserMeFragment();
    final FragmentManager fm = getSupportFragmentManager();
    Fragment active = mTaskListFragment;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity main);
        ButterKnife.bind(this);

mBottomNavigationView.setOnNavigationItemSelectedListener(mOnNavigati
onItemSelectedListener);
        fm.beginTransaction().add(R.id.main_fragment_container,
mUserMeFragment, "3")
                .hide(mUserMeFragment).commit();
        fm.beginTransaction().add(R.id.main fragment container,
mUserGroupFragment, "2")
                .hide(mUserGroupFragment).commit();
        fm.beginTransaction().add(R.id.main_fragment_container,
mTaskListFragment, "1").commit();
    }

    private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
        = item -> {
            switch (item.getItemId()) {
                case R.id.nav list:

fm.beginTransaction().hide(active).show(mTaskListFragment).commit();
                    active = mTaskListFragment;
                    break;
                case R.id.nav group:

fm.beginTransaction().hide(active).show(mUserGroupFragment).commit();
                    active = mUserGroupFragment;
                    break;
                case R.id.nav me:

fm.beginTransaction().hide(active).show(mUserMeFragment).commit();
                    active = mUserMeFragment;
                    break;
            }
            return true;
        };
}
```

```
public class TaskListFragment extends Fragment {
  // 片段类，连接视图和视图模型层。忽视 BindView 相关代码。
  TaskViewModel taskViewModel;
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment task list,
container, false);
        unbinder = ButterKnife.bind(this, view);
        setHasOptionsMenu(true);

((AppCompatActivity)requireActivity()).setSupportActionBar(toolba
r);
        ActionBar actionBar =
((AppCompatActivity)requireActivity()).getSupportActionBar();
        if (actionBar != null)
actionBar.setDisplayShowTitleEnabled(false);
        mToolbarTitleTextView.setText("任务列表");
        mTaskRecyclerView.setLayoutManager(new
LinearLayoutManager(getActivity()));
        final TaskAdapter adapter = new TaskAdapter();
        adapter.setClickListener(new TaskOnClickListener());
        mTaskRecyclerView.setAdapter(adapter);
        taskViewModel =
ViewModelProviders.of(this).get(TaskViewModel.class);
        taskViewModel.getCurrentUserAllTasks().observe(this,
                listResource -> {
                    if (listResource != null)
                        adapter.setAimTasks(listResource.data);
                });
        int userType =
AppPreferencesHelper.getCurrentUserLoginState();
        List<String> sUserType= FinalMap.getUserTypeList();
        if (userType == sUserType.indexOf(MANAGER_GROUP)){
            mFloatingActionButton.setOnClickListener(new
NewTaskOnClickListener());
        } else {
            CoordinatorLayout.LayoutParams p =
(CoordinatorLayout.LayoutParams)
mFloatingActionButton.getLayoutParams();
            p.setAnchorId(View.NO_ID);
            mFloatingActionButton.setLayoutParams(p);
            mFloatingActionButton.hide();
        }
        return view;
    }
```

```
private class TaskOnClickListener implements
TaskAdapter.OnItemClickListener{
       @Override
       public void onItemClick(Task task) {
           Intent intent =
TaskReadActivity.newIntent(getActivity(), task.getTaskID());
           startActivity(intent);
       }
   }
   private class NewTaskOnClickListener implements
View.OnClickListener{
       @Override
       public void onClick(View v) {
           Intent intent =
TaskNewActivity.newIntent(getActivity());
           startActivity(intent);
       }
   }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <include layout="@layout/main_toolbar"/>
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/task_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/toolbar"
        app:layout_behavior=" @string/appbar_scrolling_view_behavior"
        />
    <com.google.android.material.floatingactionbutton.FloatingAc-
tionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@drawable/ic_add_fff_24dp"/>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="@dimen/card_item_between"
    app:cardBackgroundColor="@color/white">
    <LinearLayout
        android:id="@+id/task_list_content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <LinearLayout
            android:id="@+id/task_list_item_right"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="@dimen/list_subitem_margin">
            <TextView
                android:id="@+id/task_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginBottom="10dp"
                android:textStyle="bold"
                android:textSize="14sp" />
            <TextView
                android:id="@+id/task_update_date"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="14sp" />
        </LinearLayout>
        <ImageView
            android:id="@+id/task_status_iv"
            android:layout_width="33dp"
            android:layout_height="32dp"
            android:layout_gravity="center"
            android:layout_marginEnd="@dimen/list_subitem_margin"
            app:srcCompat="@drawable/ic_help_61_24dp"
            android:contentDescription="@string/task_status_label"/>
    </LinearLayout>
</androidx.cardview.widget.CardView>
```

任务创建

```
public class TaskNewFragment extends Fragment {
    // 任务新建片段，略过十数行变量声明。
    public static TaskNewFragment newInstance() {
        TaskNewFragment fragment = new TaskNewFragment();
        return fragment;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        viewModel =
ViewModelProviders.of(this).get(TaskViewModel.class);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment new task,
container, false);
        unbinder = ButterKnife.bind(this, v);
        setHasOptionsMenu(true);
        ((AppCompatActivity)
requireActivity()).setSupportActionBar(mToolbar);
        ActionBar actionBar =
((AppCompatActivity)requireActivity()).getSupportActionBar();
        if (actionBar != null)
            actionBar.setDisplayShowTitleEnabled(false);
        mToolbarTitleTextView.setText("新建任务");

        me = viewModel.getMe();
        mAssignerTextView.setText(me.getName());
        ArrayAdapter<String> typeArrayAdapter = new
ArrayAdapter<>(requireActivity(),android.R.layout.simple list ite
m 1, sTaskType);
        mTypeSpinner.setAdapter(typeArrayAdapter);
        return v;
    }

    @OnClick(R.id.create btn)
    void createTask(){
        mCreateButtom.setEnabled(false);
        title = mTitleEditText.getText().toString();
        int assignee = assigneeID;
        if (!mAssigneeEditText.getText().toString().isEmpty())
            assignee =
Integer.parseInt(mAssigneeEditText.getText().toString());

        type = mTypeSpinner.getSelectedItemPosition();
        detail = mDetailEditText.getText().toString();
        if (!validate(title, assignee, type, detail)){
            onCreateTaskMessage(FinalMap.statusCodeLost);
            return;
        }
```

```
Call<ResponseBody> call = viewModel.createTask(title, assigneeID,
type, detail);
        call.enqueue(new Callback<ResponseBody>() {
            @Override
            public void onResponse(Call<ResponseBody> call,
Response<ResponseBody> response) {
                int statusCode = response.code();
                String message = statusCodeMap.get(statusCode);
                mAppExecutors.mainThread().execute(()
                        -> onCreateTaskMessage(message)
                );
                viewModel.resetTaskListRateLimit(me.getUid());
                requireActivity().finish();
            }

            @Override
            public void onFailure(Call<ResponseBody> call,
Throwable t) {
                    mAppExecutors.mainThread().execute(()
                            ->
onCreateTaskMessage(FinalMap.statusCodeFail));
            }
        });
    }

    @OnClick(R.id.choose assignee btn)
    void chooseAssignee(){
        Intent intent =
TaskAssignActivity.newIntent(getActivity(), null, false);
        startActivityForResult(intent, REQUEST ASSIGNEE);
    }

    private void onCreateTaskMessage(String message) {
        mCreateButtom.setEnabled(true);
        Toast.makeText(getActivity(), message,
Toast.LENGTH LONG).show();
    }

    private boolean validate(String title, int assignee, int
type, String detail) {
        boolean valid = true;
        if (title.isEmpty() || title.length() < 3){

mTitleEditText.setError(requireActivity().getResources().getStrin
g(R.string.valid string long));
            valid = false;
        }
        if (assignee < 0){

mAssigneeEditText.setError(requireActivity().getResources().getSt
ring(R.string.valid select user));
            valid = false;
        }
        if (detail.isEmpty() || detail.length() < 3){

mDetailEditText.setError(requireActivity().getResources().getStri
ng(R.string.valid string long));
            valid = false;
```

```
        }
        return valid;

    }

    @OnClick(R.id.cancel_btn)
    void cancel(){
        requireActivity().onBackPressed();
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_ASSIGNEE){
            if (resultCode == Activity.RESULT_OK){
                ArrayList<User> aeeList =
data.getParcelableArrayListExtra(ASSIGNEE);
                User assignee = aeeList.get(0);
                mAppExecutors.mainThread().execute(() -> {

mAssigneeEditText.setText(String.valueOf(assignee.getUid()));

mAssigneeTextView.setText(assignee.getName());
                });

            } else {
                mAppExecutors.mainThread().execute(() ->
                        Toast.makeText(getContext(), TAG + ":" +
"操作取消",
                                Toast.LENGTH_SHORT).show());
            }
        }
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();
        unbinder.unbind();
    }
}

public class TaskNewActivity extends SingleFragmentActivity{
    public static Intent newIntent(Context packageContext){
        Intent intent = new Intent(packageContext,
TaskNewActivity.class);
        return intent;
    }
    @Override
    protected Fragment createFragment() {
        return TaskNewFragment.newInstance();
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/an-
droid"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:fitsSystemWindows="true">
    <LinearLayout
        android:id="@+id/fragment_task"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <include
            android:id="@+id/task_toobar"
            layout="@layout/main_toolbar" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/white"
            android:orientation="vertical"
            android:paddingLeft="16dp"
            android:paddingRight="16dp">
            <com.google.android.material.textfield.TextInputLay-
out
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16sp"
                android:layout_marginBottom="16sp">
            </com.google.android.material.textfield.TextInputLay-
out>
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16sp"
                android:layout_marginBottom="16sp"
                android:orientation="horizontal"
                android:padding="10dp">
                <TextView
                    android:id="@+id/assigner_text"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="2"
                    android:gravity="center_vertical"
                    android:hint="@string/task_creator_label"
                    android:textSize="@dimen/normal_text_size" />
```

```
            <TextView
                    android:id="@+id/assigner_content"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="3"
                    android:hint="@string/task_assigner_label"
                    android:textSize="@dimen/normal_text_size"
                    android:textStyle="bold" />
            </LinearLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16sp"
                android:layout_marginBottom="16sp">
                <EditText
                    android:id="@+id/input_title"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/task_title_label" />

</com.google.android.material.textfield.TextInputLayout>
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="3dp"
                android:layout_marginBottom="3dp"
                android:orientation="horizontal"
                android:padding="10dp">
                <TextView
                    android:id="@+id/input_type"
                    android:layout_width="0dp"
                    android:layout_height="match_parent"
                    android:layout_weight="2"
                    android:hint="@string/task_type_label"
                    android:textSize="@dimen/normal_text_size" />
                <Spinner
                    android:id="@+id/task_type_spinner"
                    android:layout_width="0dp"
                    android:layout_height="match_parent"
                    android:layout_weight="3" />
            </LinearLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
                android:layout_marginTop="16sp"
                android:layout_marginBottom="16sp">
                <EditText
                    android:id="@+id/input_detail"
                    android:layout_width="match_parent"
                    android:layout_height="240dp"
                    android:gravity="top"
                    android:hint="@string/task_detail_label" />

</com.google.android.material.textfield.TextInputLayout>
            <com.google.android.material.textfield.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16sp"
                android:layout_marginBottom="16sp">
                <EditText
                    android:id="@+id/input_assignee"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/task_assignee_label"
                    android:inputType="number" />

</com.google.android.material.textfield.TextInputLayout>
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="16sp"
                android:layout_marginBottom="16sp"
                android:orientation="horizontal"
                android:padding="10dp">
                <TextView
                    android:id="@+id/assignee_text"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="2"
                    android:gravity="center_vertical"
                    android:hint="@string/task_assignee_label"
                    android:textSize="@dimen/normal_text_size" />
                <TextView
                    android:id="@+id/assignee_content"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="3"
                    android:hint="@string/unknown"
```

```
                android:textSize="@dimen/normal_text_size"
                android:textStyle="bold" />
        </LinearLayout>
        <Button
            android:id="@+id/choose_assignee_btn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/btn_margin"
            android:background="@color/greenA700"
            android:text="@string/task_choose_assignee_label"
            android:textColor="@color/white"
            android:textSize="@dimen/big_text_size" />
        <Button
            android:id="@+id/create_btn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/btn_margin"
            android:background="@color/greenA700"
            android:text="@string/task_create"
            android:textColor="@color/white"
            android:textSize="@dimen/big_text_size" />
        <Button
            android:id="@+id/cancel_btn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/btn_margin"
            android:layout_marginBottom="@dimen/btn_margin"
            android:background="@color/secondary"
            android:text="@string/task_cancel"
            android:textColor="@color/white"
            android:textSize="@dimen/big_text_size" />
    </LinearLayout>
</LinearLayout>
</ScrollView>
```

照片上传

```
public class TaskPhotoActivity extends SingleFragmentActivity {
    public static Intent newIntent(Context packageContext, int
taskID){
        Intent intent = new Intent(packageContext,
TaskPhotoActivity.class);
        intent.putExtra(EXTRA TASK ID, taskID);
        return intent;
    }
    @Override
    protected Fragment createFragment() {
        int taskID = getIntent()
                .getIntExtra(EXTRA TASK ID, -1);
        return TaskPhotoFragment.newInstance(taskID);
    }
}
```

```
public class TaskPhotoFragment extends Fragment {
    // 忽略变量声明与 binder 以及百度地图 SDK 生命周期相关代码
    public static TaskPhotoFragment newInstance(int taskId) {
        Bundle args = new Bundle();
        args.putInt(ARG Task ID, taskId);
        TaskPhotoFragment fragment = new TaskPhotoFragment();
        fragment.setArguments(args);
        return fragment;
    }
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        taskId = getArguments().getInt(ARG Task ID);
        viewModel =
ViewModelProviders.of(this).get(TaskViewModel.class);
    }
@Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
@Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment task photo,
container, false);

       ((AppCompatActivity)requireActivity()).setSupportActionBar(too
       lbar);
        ActionBar actionBar =
       ((AppCompatActivity)requireActivity()).getSupportActionBar();
        if (actionBar != null)
           actionBar.setDisplayShowTitleEnabled(false);
        mToolbarTitleTextView.setText(TAG);
        mLocationClient = BasicApp.getLocationClient(getActivity());
        mLocationClient.registerLocationListener(myLocationListener);
        mPhotoRecyclerView.setLayoutManager(new
LinearLayoutManager(getContext()));
        mAdapter = new PhotoAdapter(getContext(), new ArrayList<>(),
taskId);
        mPhotoRecyclerView.setAdapter(mAdapter);
        requestPermissions();
        return view;
    }
```

```
    @Override
    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST IMAGE && resultCode == RESULT OK) {
            List<String> paths = Matisse.obtainPathResult(data);
            mAdapter.addPaths(paths);
        }
    }

    @OnClick(R.id.add photo fab)
    public void requestPermissions(){
        List<String> permissionList = new ArrayList<>();
        if (ContextCompat.checkSelfPermission(requireActivity(),
Manifest.permission.READ EXTERNAL STORAGE) !=
PackageManager.PERMISSION GRANTED) {
            permissionList.add(Manifest.permission.READ PHONE STATE);
        }
        if (ContextCompat.checkSelfPermission(requireActivity(),
Manifest.permission.WRITE EXTERNAL STORAGE)!=
PackageManager.PERMISSION GRANTED) {

permissionList.add(Manifest.permission.WRITE EXTERNAL STORAGE);
        }
        if (ContextCompat.checkSelfPermission(requireActivity(),
Manifest.permission.ACCESS FINE LOCATION)!=
PackageManager.PERMISSION GRANTED) {

permissionList.add(Manifest.permission.ACCESS FINE LOCATION);
        }
        if (ContextCompat.checkSelfPermission(requireActivity(),
Manifest.permission.READ_PHONE_STATE) !=
PackageManager.PERMISSION_GRANTED) {
            permissionList.add(Manifest.permission.READ_PHONE_STATE);
        }
        if (ContextCompat.checkSelfPermission(requireActivity(),
Manifest.permission.WRITE_EXTERNAL_STORAGE)!=
PackageManager.PERMISSION_GRANTED) {
      permissionList.add(Manifest.permission.WRITE_EXTERNAL_STORAGE);
        }
        if (!permissionList.isEmpty()) {
            String [] permissions = permissionList.toArray(new
String[permissionList.size()]);
            ActivityCompat.requestPermissions(requireActivity(),
permissions, 1);
        }
        else {
            requestLocation();
            requestImage();
        }
    }
```

```
    @OnClick(R.id.toolbar_OK_btn)
    public void uploadPhotos(){
        List<PhotoUpload> photoUploads = mAdapter.retrieveData();
        if (photoUploads == null || photoUploads.isEmpty()){
            Toast.makeText(getContext(), "请点击右下角添加图片",
Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getContext(), "Going on",
Toast.LENGTH SHORT).show();

            for (PhotoUpload photoUpload: photoUploads) {
                Call call = viewModel.uploadPhoto(photoUpload,
mCurrentLatitude, mCurrentLongitude, taskId);
                if (call != null){
                    call.enqueue(new Callback() {
                        @Override
                        public void onResponse(Call call, Response
response) {
                            String responseData = "";
                            if (response.body() != null)
                                responseData = response.body().toString();
                        }
                        @Override
                        public void onFailure(Call call, Throwable t) {
                        }
                    });
                } else {
                    Toast.makeText(getContext(), "处理图片出错",
                            Toast.LENGTH SHORT).show();
                }
            }
        }
    }

    @OnClick(R.id.toolbar cancel btn)
    public void photoCancel(){
        requireActivity().finish();
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions,
                                           int[] grantResults) {
        switch (requestCode) {
            case 1:
                if (grantResults.length > 0) {
                    for (int result : grantResults) {
                        if (result != PackageManager.PERMISSION GRANTED){
                            Toast.makeText(requireActivity(),
                                    "同意后才能使用本功能",
                                    Toast.LENGTH SHORT).show();
                            requireActivity().finish();
                            return;
                        }
                    }
                    requestImage();
                }
```

```
    else {
            Toast.makeText(requireActivity(),
                                "已经禁用照相功能",
Toast.LENGTH_SHORT).show();
                    requireActivity().finish();
                }
                break;
             default:
        }
    }

    private void requestLocation() {
        initLocation();
        mLocationClient.start();
    }

    private void initLocation(){
        LocationClientOption locationOption = new
LocationClientOption();
        locationOption.setScanSpan(800);

locationOption.setLocationMode(LocationClientOption.LocationMode.High
t_Accuracy);
        locationOption.setIsNeedAddress(true);
        locationOption.setIsNeedLocationDescribe(true);
        locationOption.setCoorType(mCurrentCoorType);
        mLocationClient.setLocOption(locationOption);
    }

    private void requestImage() {
        Matisse.from(this)
                .choose(MimeType.of(MimeType.JPEG))
                .countable(true)
                .maxSelectable(9)
                .gridExpectedSize(getResources().getDimensionPixelSiz
e(R.dimen.grid expected size))
                .restrictOrientation(ActivityInfo.SCREEN ORIENTATION
UNSPECIFIED)
                .thumbnailScale(0.85f)
                .imageEngine(new Glide4Engine())
                .capture(true)
                .captureStrategy(new CaptureStrategy(true,
                        "cn.com.wosuo.taskrecorder.fileprovider", "场
调记录"))
                .forResult(REQUEST IMAGE);
    }

    public class MyLocationListener extends
BDAbstractLocationListener {
        @Override
        public void onReceiveLocation(BDLocation location){
            mCurrentLatitude = location.getLatitude();
            mCurrentLongitude = location.getLongitude();
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
            <include android:id="@+id/photo_toolbar"
                layout="@layout/btn_toolbar" />
            <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/photo_recycler_view"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                app:layout_behavior="
@string/appbar_scrolling_view_behavior" />
        </LinearLayout>
    </androidx.core.widget.NestedScrollView>

<com.google.android.material.floatingactionbutton.FloatingActionButto
n
        android:id="@+id/add_photo_fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@drawable/ic_add_fff_24dp"/>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```