

deepwaterooo deepwateroooMe – I am the same GitHub account person

deepwaterooo

September 5, 2022

Contents

1 要求	1
2 主要思路	2

1 要求

- Overview
 - Build an **employee directory app** that shows a list of employees from the provided endpoint.
 - The app should display a list (or any kind of **collection view!**) which shows all the employees returned from the JSON endpoint described below.
 - Each item in the view should contain a **summary of the employee**, including their **photo, name, and team** at minimum. You may add more information to the summary if you want, or **sort employees in any fashion** you'd like –sort and group by name, team, etc.
 - There should be some UX to reload the employee list from within the app at any time. The UX can be done in any way you want: **a button, pull-to-refresh**, etc.
 - If there is any additional UI/UX you would like to add, feel free to do so! We only ask that you please **do not build any more screens** than this list. Do not worry about building custom controls or UI elements –using **system-provided, standard elements** is totally fine.
 - Be sure to **appropriately handle the normal variety of errors when querying an endpoint**. The app should **display useful loading, empty, and error states** where appropriate. **If images fail to load, displaying a placeholder** is fine.
 - One extra thing we ask is that you please ensure you **do not use more network bandwidth than necessary** –load expensive resources such as photos on-demand only.
 - The **employee list should not be persisted to disk**. You can reload it from the network **on each app launch and when refresh is requested** —but no more often than that unintentionally. (Android developers in particular should take care **not to make redundant network calls** when the **phone is rotated, or when memory is low**).
 - **Images**, however, should **be cached on disk** so as to not waste device bandwidth. You may use an **open source image caching solution**, or write your own caching. Do not rely upon HTTP caching for image caching.

- Note that photos at a given URL will never change. Once one is loaded, you do not need to reload the photo. If an employee's photo changes, they will be given a new photo URL.
- Tests should be provided for the app. We do not expect 100% code coverage, so please use your best judgment for what should be tested. We're also interested only in unit tests. Feel free to skip snapshot or app tests.
- MVVM: 需要数据驱动, viewModel 里定义一个状态变量, 来标记当前的活动状态
 - If any employee is malformed, it is fine to invalidate the entire list of employees in the response - there is no need to exclude only malformed employees.
 - If there are no employees to show, the app should present an **empty state** view instead of an empty list.

2 主要思路

- 这是一个看似要求极其简单, 实则考验的知识点和深度有着相当的跨度的小项目。
- 它们一定挑都要挑我出差到 WSU 的一个星期里来考验我, 因为他们就是想要去打败一个人。呵呵, 真正想要打败一个人, 谈何容易, 就凭这???
- Retrofit + RxJava: 好像是更合适的, 可以用注解, 并且用得更为广泛
 - 搜索关键字: Retrofit + OkHttp + RxJava 网络库构建
 - **OkHttp**: 网络请求处理, 主要是在应用启动的时候, 什么时机开始发布和调用网络请求。所以这个可以不用了, 大家都喜欢新的更好用的库
- 图片本地缓存: 第三方库找一个, 还是用 AndroidX 的 Room
 - 我 现在数据库的问题是: 我 缓存保存了员工数据进数据库, 但是这里说得很清楚了, 不用保存员工数据, 只保存每个员工 id 所对应的图片就可以了
- 现在的难点: 不知道怎么定义图片数据库, 同时以 OkHTTP response 回来的连接起来
- 应用的 启动优化: 重中之重, 需要借助这个小应用看懂弄清楚, 不知道如何拆解网络请求的步骤, 什么时候加载, 初始化之类的? 以达到较好的启动优化
-
- **MVVM 设计**: 只有一个页面, 相对就简单方便多了。工作中的案例是使用 MVVM 但自己编辑逻辑处理信号下发, 与数据驱动的 UI 更新, 没有实现双向数据绑定的; 可是这里感觉 双向数据绑定更简单, 会有哪些可能的问题呢? 这里基本可以当作不需要双向, 因为一个 UI 按钮要求刷新是唯一的 UI 需求; 更多的只是需要时候的数据往 UI 加载更新; 所以 可以简单使用观察者模式, UI 观察数据的变化就可以了
- 图片的加载与处理: 用样可以使用么第三方库 **glide**
- 图片的加载与处理: 用样可以使用么第三方库 **CircularImageView**
- **AndroidX RecyclerView** 的使用: 选择相对更为高效和方便管理的库和数据结构来使用
- **Constraint Layout vs Coordinate Layout**: 暂时先用任何简单的 layout 先能运行起一个大致的框架来, 再进一步优化
- 我丢掉了的文件呀, 我写过的项目呀, 不是在进 Lucid 之前写得好好的一个项目, 现在源码全丢了。。。。该死的 GitHub.....