Technical Project Report - Android Module

MealFinder

Course:	Introdução à Computação Móvel
Date:	Aveiro, <20-04-2020>
Authors:	82773: Vinicius Benite Ribeiro 89206: José Frias 89124: Miguel Matos
Project abstract:	MealFinder is your diet friendly restaurant finder. All you need is to activate your location and choose your diets and we will find restaurants suitable for you. You can, also, review your favorite restaurants, save your favorites ones and create your own food logs.

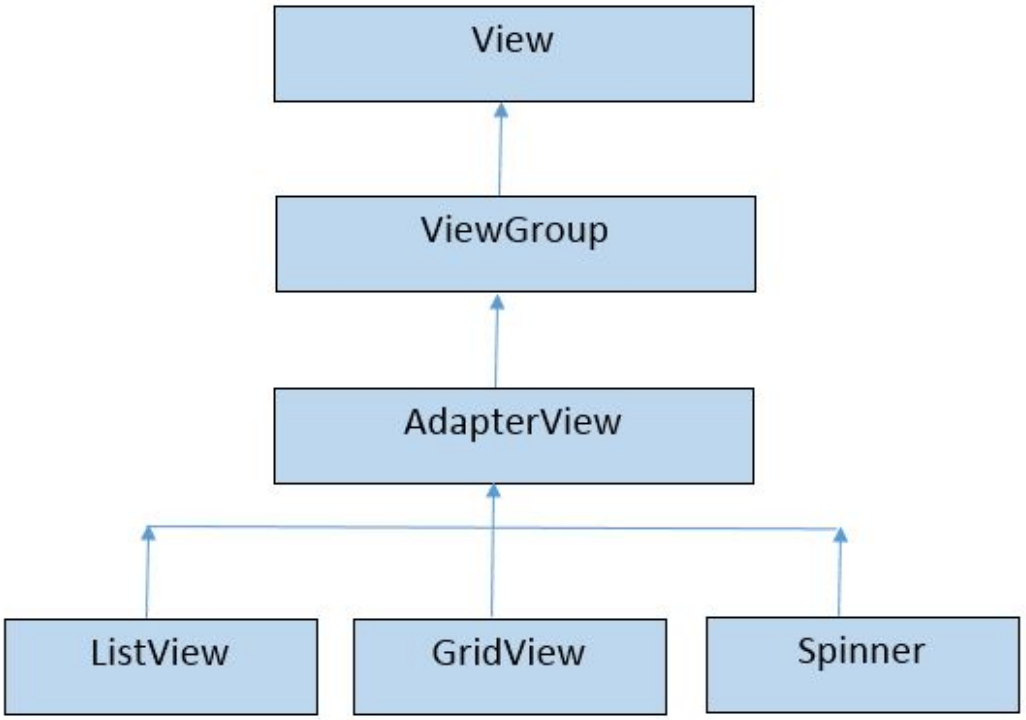
Table of contents:

1 Implemented solution

Architectural overview

Our application has four modules:

1. Adapters: the classes here acts like a bridge between the views and the actual data.



http://www.brainysolutions.org/Images/Contents/482/adapterView.gif

We extended the FirestoreRecyclerAdapter class and its related methods to implement those adapters.

2. Models: classes responsible for the data structure.
3. Network: this module is responsible for the external networks calls. We used Retrofit to make the api's calls to the Zomato[1] API to fetch restaurants details.
4. User Interface: here we find all the fragments used in our app. Also, we implemented the logic of our application inside the fragments. Reads and writes are also done in the fragments. All the user permissions to use GPS, access location and use the camera are done inside the fragments (RestaurantsDetailsFragment and

AddLogFragment). Later, we discovered that this was a bad practice to do. These UI-based classes should only contain logic that handles UI and operating system interactions. By keeping these classes as lean as possible, you can avoid many lifecycle-related problems[2].

The module “model” handles all the data models/structures. All data for restaurants, reviews, locations, and others, are handled here. Those are simple Java classes with set/get methods.

For data persistence, we used Google Firebase. Content updates are handled in the fragments, using FirebaseFirestore, CollectionReference class and FirestoreRecyclerOptions to fetch/display information. Our database has two collections: users e reviews. The first one stores user information and reviews made by that user. The second one stores all reviews and ratings made by our users to each restaurant.

Implemented interactions

Basically, the user login into the application, choose his diets and the app searches for restaurants that meet his needs based on the user's current location. The user can review and rate a restaurant, add the restaurant to his favorites and also, create food logs for his meals, to share between other users.

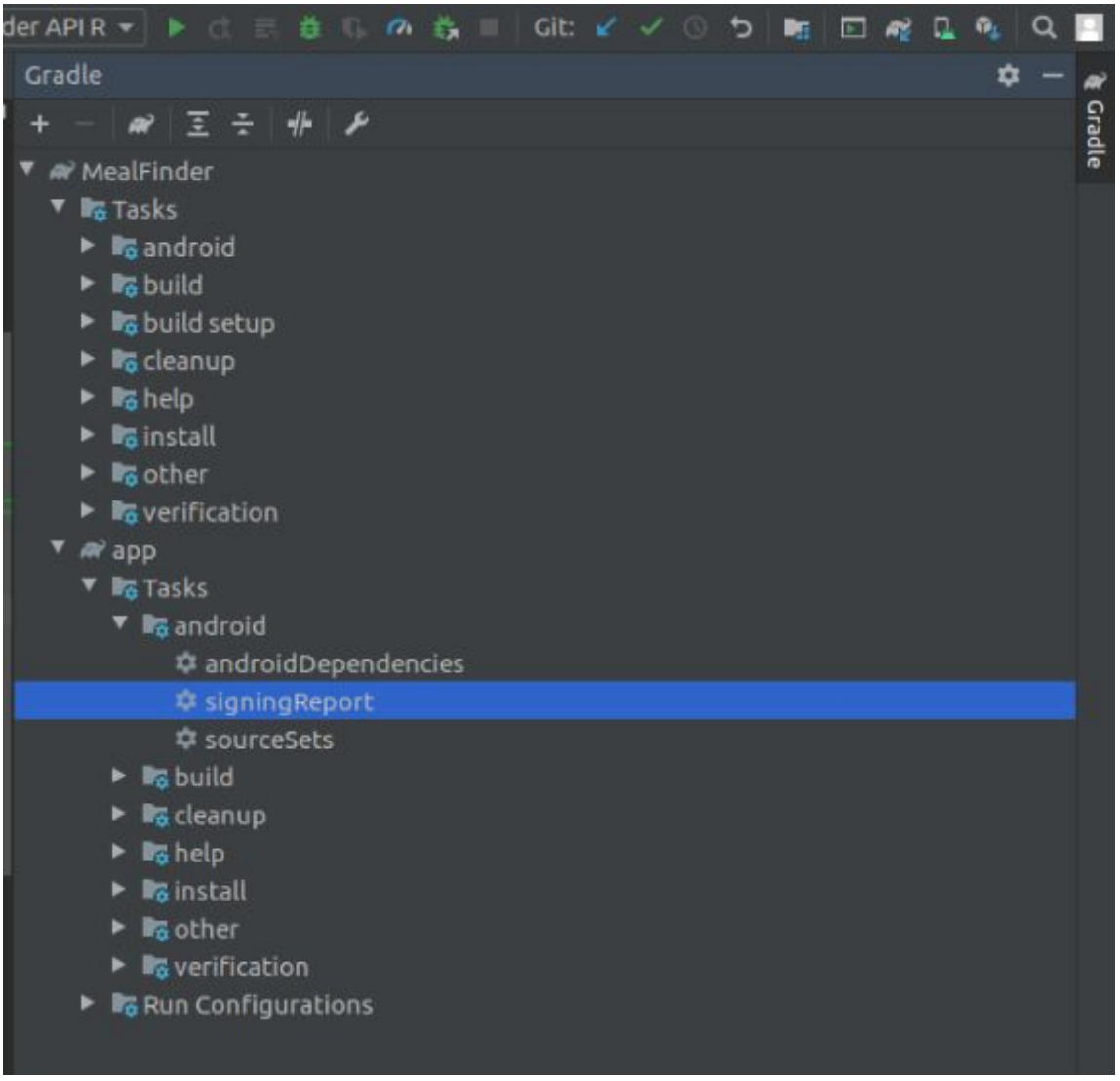
The complete diagram of interactions can be found at: **https://drive.google.com/file/d/131w29xGN-DjfL1qynSXNzmXpSGwbGPUq/view?usp=sharing**

Problems

- In some devices, if you turn off the location and again turn on, the previous recorded location information will be cleared and, if the user never turned on location before using your App, the previous location information will be null too. To solve this, we implemented the *requestNewLocationData()* that fetches the geolocation at runtime.
- If your targetSdkVersion >= 24, then we have to use FileProvider class to give access to the particular file or folder to make them accessible for other apps. We create our own class inheriting FileProvider in order to make sure our FileProvider doesn't conflict with FileProviders declared in imported

dependencies as described here.

- Fetching and uploading data from the Firebase database could take some time to complete.
- We encountered a problem with the Firebase authentication. We need to manually add a SHA1 digital impression of our devices (https://console.firebase.google.com/u/0/project/mealfinder-39b8f/settings/general/android:com.example.mealfinder) to be able to login into the application. To get the digital impression, we have to run the signinReport from Gradle.



Limitations

- The frontend is very basic, only to support the main interactions for this project.

- If the upload of images takes more time than usual, then the image will not be saved properly.

2 References and resources

Key project resources

- Code repository: https://github.com/joselfrias/meal_finder
- Ready-to-deploy APK: <put URL for apk; may be inside the code repo>

Reference materials

[1] https://developers.zomato.com/api

[2] https://developer.android.com/jetpack/docs/guide#common-principles

https://developer.android.com/docs

https://www.androidhive.info/2013/07/android-expandable-list-view-tutorial/

https://www.journaldev.com

https://firebase.google.com/docs/reference/android/packages