ING Interview Project

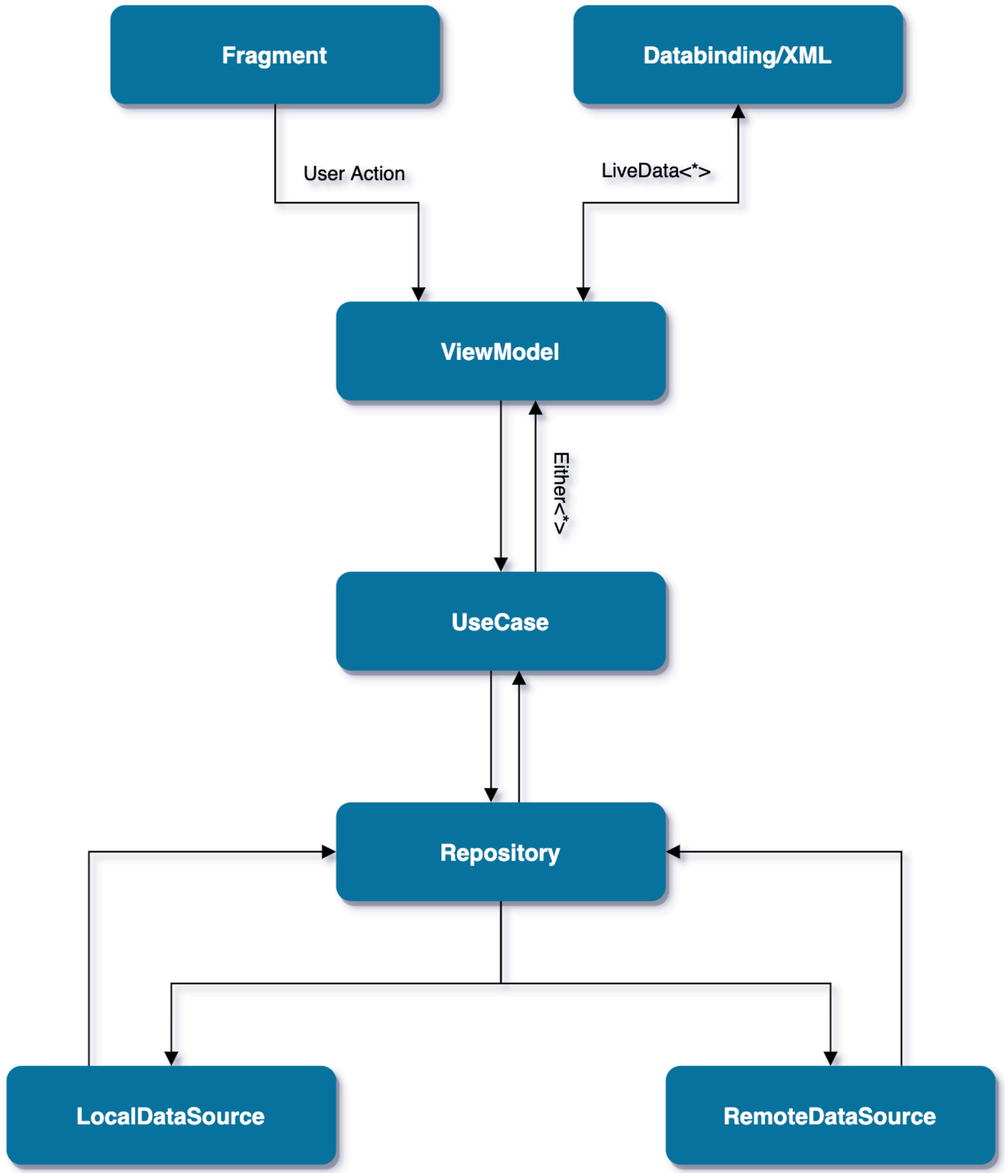
Burak Karakoca - Mobile Application Developer

Architecture Diagram

Used single activity, clean architecture and MVVM design pattern.

In the architecture diagram we have UI, ViewModel, Business, Repository layers.

Each layer has manner of single responsibility principle.



Project Structure

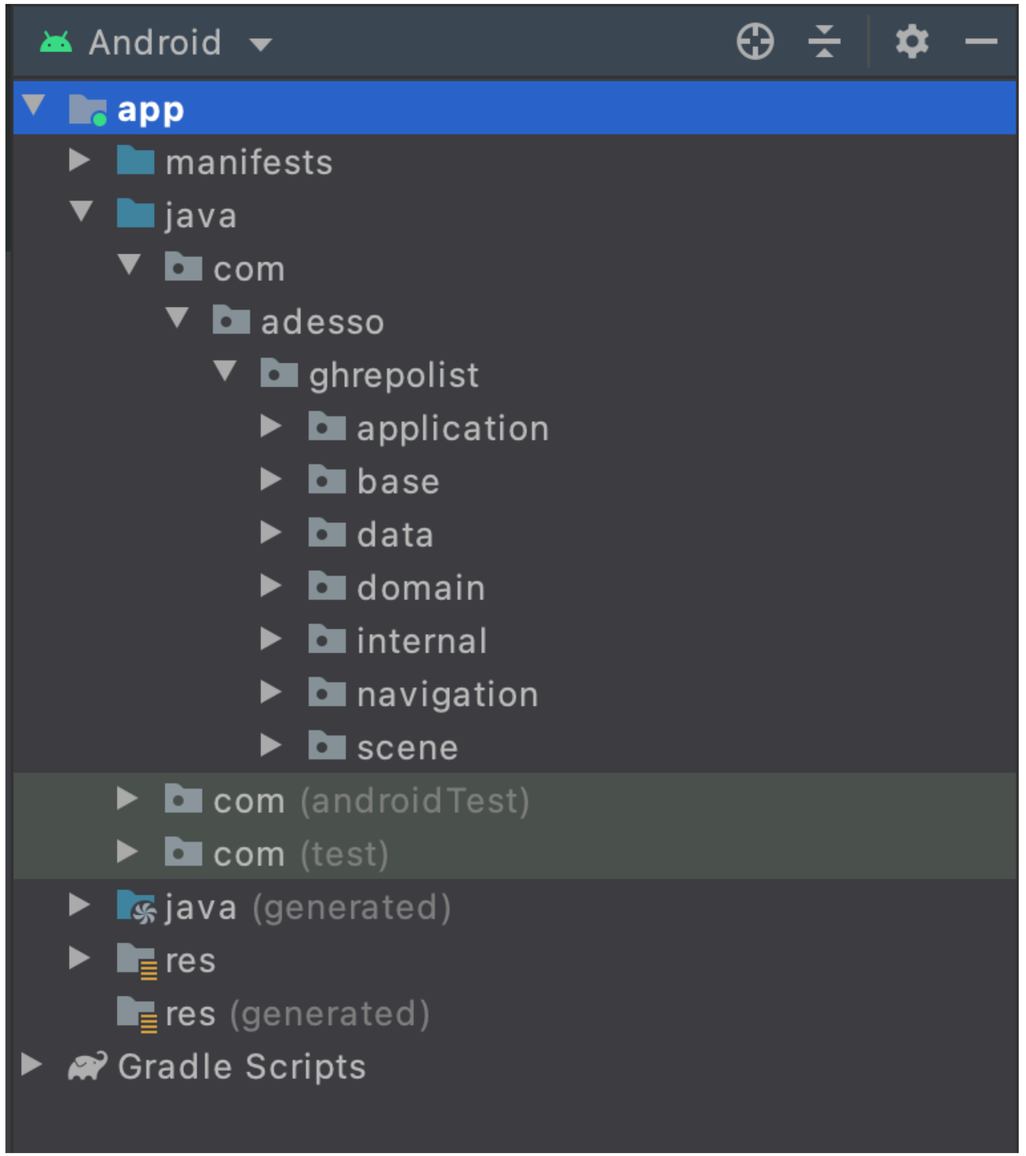
The project architecture was created according to the MVVM design pattern principles. For each layer;

- **View:** Creates and handles UI and delegates actions to ViewModel.

- **ViewModel:** Handles UI logic and gets the data from UseCase.

- **UseCase:** Business layer that delegates with Repository.

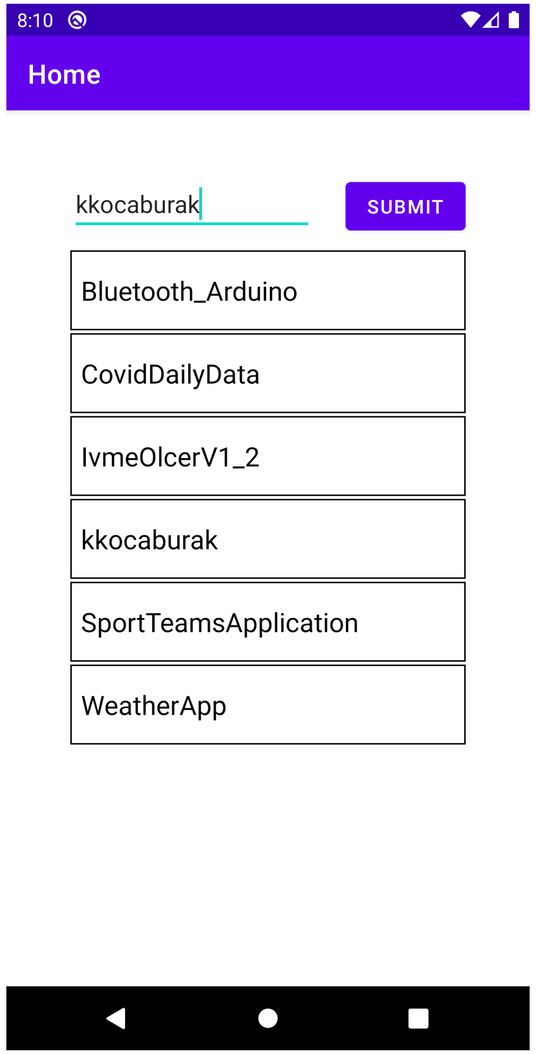
- **Repository:** Single source of data. Responsible to get data from one or more data sources.

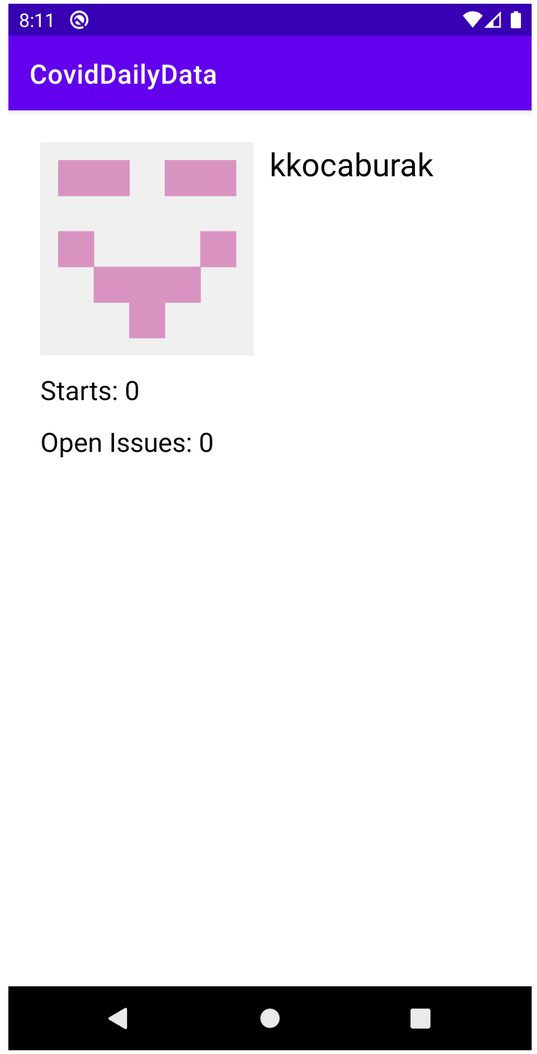


Used Technologies

- **Navigation Component:** Consistent navigation between views
- **ViewModel:** Holds UI data across configuration changes
- **LiveData:** Lifecycle aware observable and data holder
- **Retrofit:** HTTP Client
- **Gson:** JSON serializer/deserializer
- **Coroutines:** Asynchronous programming
- **Dagger:** Dependency injection
- **DataBinding:** Binds UI components in layouts to data sources
- **Glide:** Image loading and caching

Project Screenshots





Unit Testing

Used JUnit4, MockK and WebService for unit testing.

Created ViewModel, UseCase and Repository test classes.

You can see an example of unit test class in the adjacent screenshot.

RepoListViewModelTest.kt

```
42      @Test
43      fun `when submit button clicked should call fetchGitHubRepoList`() = runBlocking {  this: CoroutineScope
44          val list = listOf(
45                  GitHubRepoModelItem(
46                          repoName: "repo1",
47                          RepoOwnerModel( userName: "user1",  avatarUrl: "userUrl1"),
48                          repoStarCount: 10,
49                          openIssuesCount: 20,
50                          isFavorite: false
51                  )
52          )
53          coEvery { repoListViewModel.fetchGitHubRepoList(any()) } returns launch {  this: CoroutineScope
54              repoListViewModel.postGitHubRepoList(list)
55          }
56
57          repoListViewModel.userName = "asd"
58          repoListViewModel.onSubmitButtonClicked()
59
60          coVerify { repoListViewModel.fetchGitHubRepoList( userName: "asd") }
61      }
62
63      @Test
64      fun `when repo item clicked should navigate to related screen`() {
65          val gitHubRepoModelItem = GitHubRepoModelItem(
66              repoName: "repo1",
67              RepoOwnerModel( userName: "user1",  avatarUrl: "userUrl1"),
68              repoStarCount: 10,
69              openIssuesCount: 20,
70              isFavorite: false
71          )
72
73          repoListViewModel.onRepoItemClick(gitHubRepoModelItem)
74
75          verify { repoListViewModel.navigate(any()) }
76      }
```

General Informations

- Project Link: https://github.com/kkocaburak/GitHubRepoListApp/tree/master
- My LinkedIn Profile: https://tr.linkedin.com/in/burak-karakoca-120b36165