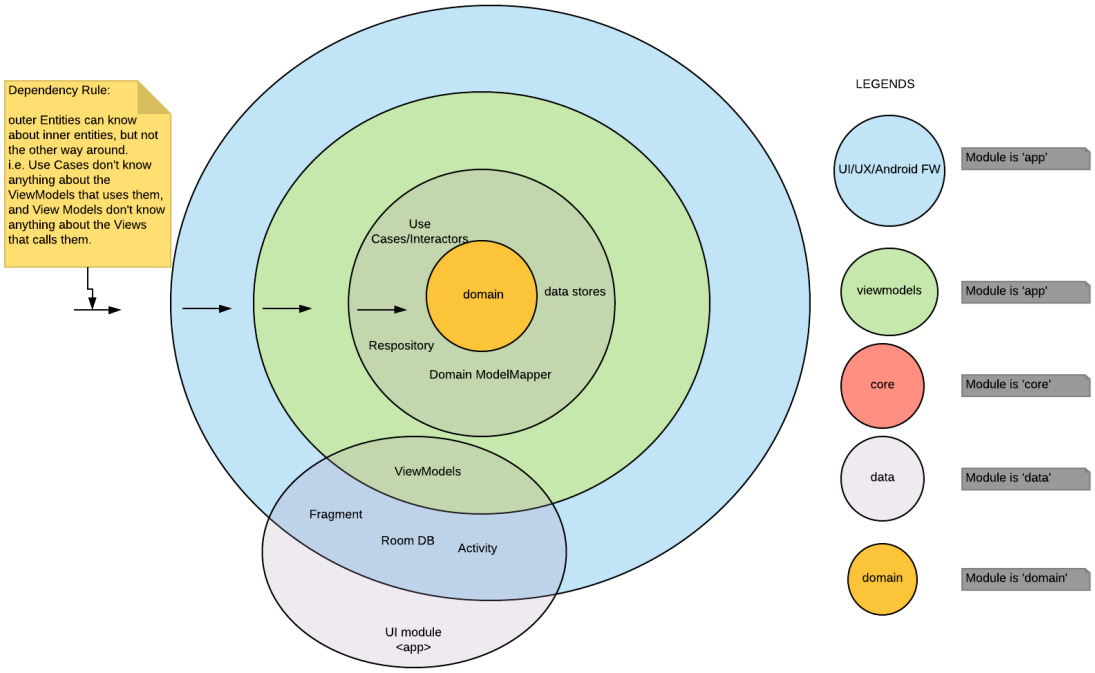
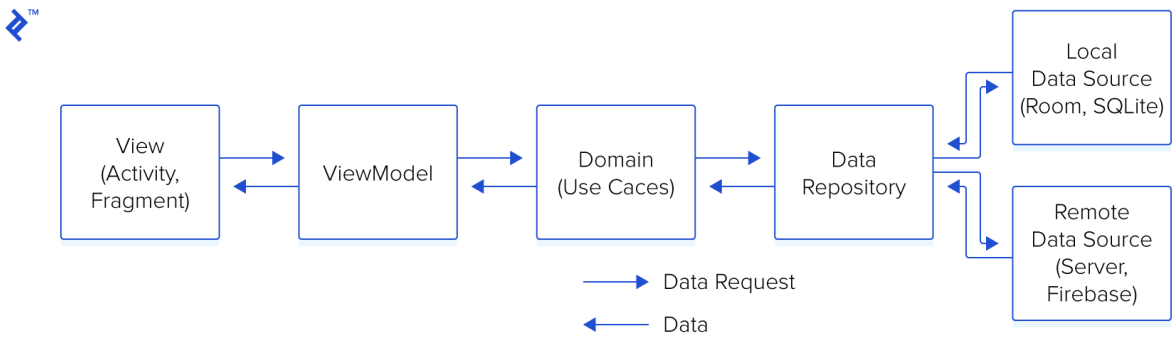
Documentation

Project Architecture Diagram

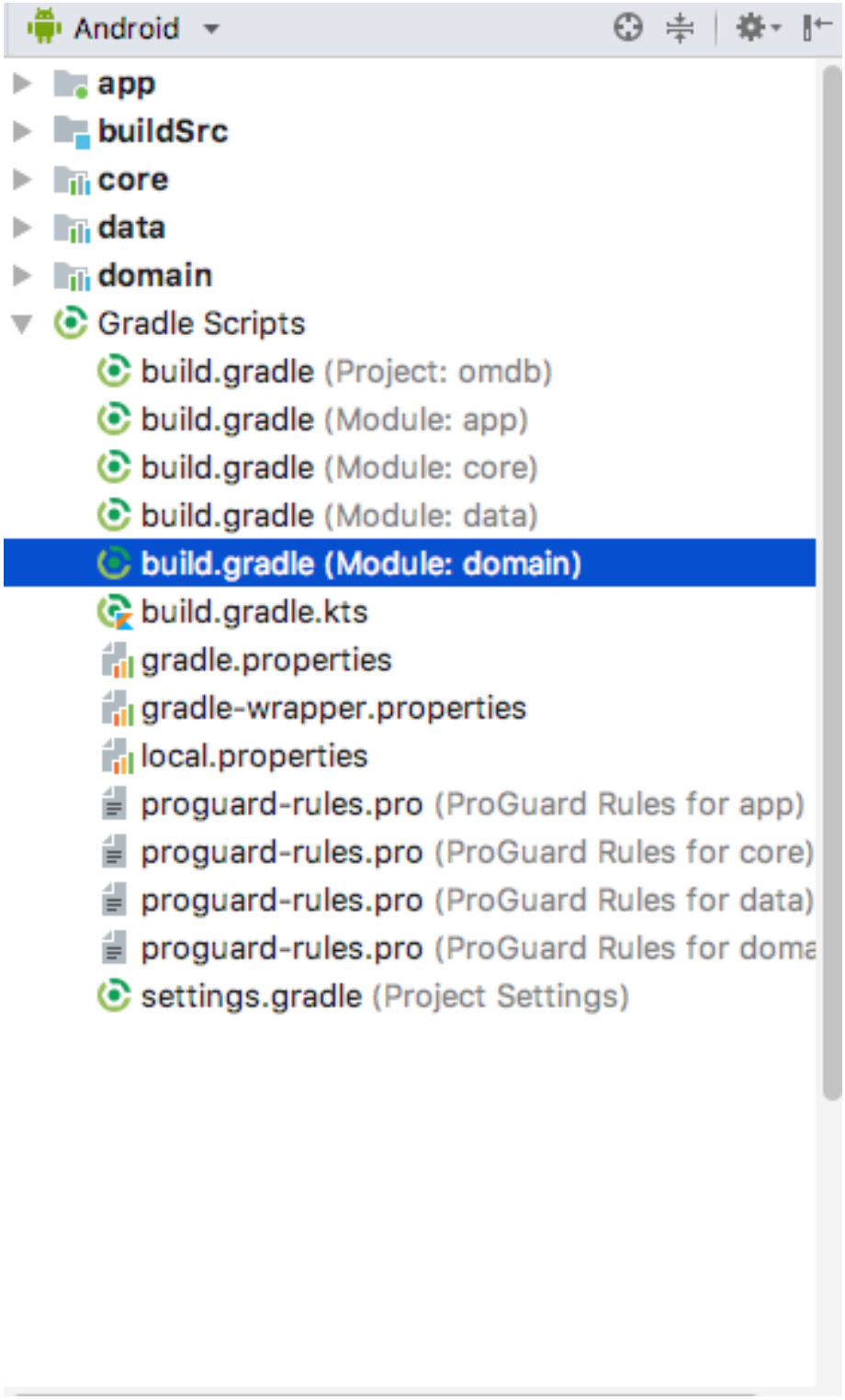
Architecture of the Entire project is a best effort attempt to be in line with the clean architecture paradigm.
Below listed diagram actually maps the clean architecture to OMDB Application modules and how the
dependency is allowed to flow.



Data Flow



Project Modules



How To Scale For API's or Data Sources

- Use the search API http://www.omdbapi.com/?s=hello
- Display it in a listview with Title, Year and Poster
- Focus on testing, writing maintainable code. (Code documentation, unit testing etc)

domain module

Domain module provides all the required abstraction , business definition and logic for the application.
Following abstractions are provided in domain module which will be used to achieve separation between data and view.

- UseCase interface is defined for the Application. The interface job is to accept a set of params and execute the task and return the result.
- Repository interface is defined for the Application
- Model object is defined for the business. This is the business data rest of the application should see or work with
- Mapper interface . Defines abstraction for translating between domain and any Model definition being used in the upper layers
- 'Either' artifact from *Fernando Cejas Open Source Project*

data module

data modules provide most of the concrete implementation for the domain/ business level abstractions and also provides an another
abstraction datastore to implement remote and cache repos.

- Generate Data Classes from provided json response using 'JsonToKotlinClass' plugin , swagger or any tool of choice
- DatStore interface is defined for the Application.
- Provide a Concrete implementation for data store interface
- Provide a Concrete implementation for repository interface
- Provide a Concrete implementation for usecase interface
- provide a concrete implementation of mapper to convert the OMDB data to domain specific data i.e 'Content'

app module

app module has the android FW , api's , ViewModel , data binding, dagger dependency FW etc

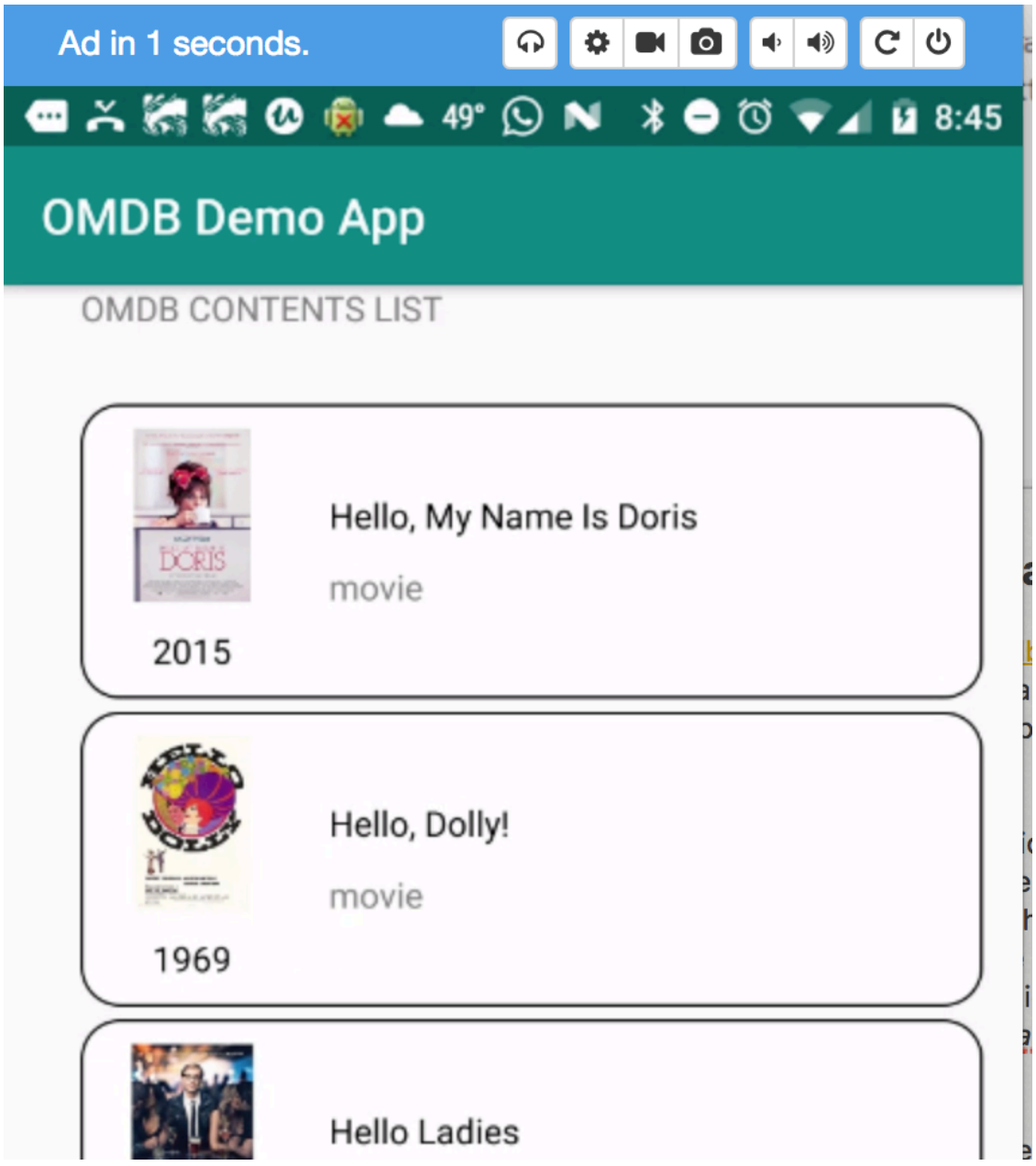
- Provide a Activity/Fragment/CustomView which has a ViewModel
- Provide ViewModel which makes use of 'UseCase' abstraction and it's concrete implementation to access the repository
- ViewModel in this application also be used for data binding
- AppComponent provides required dependency injection

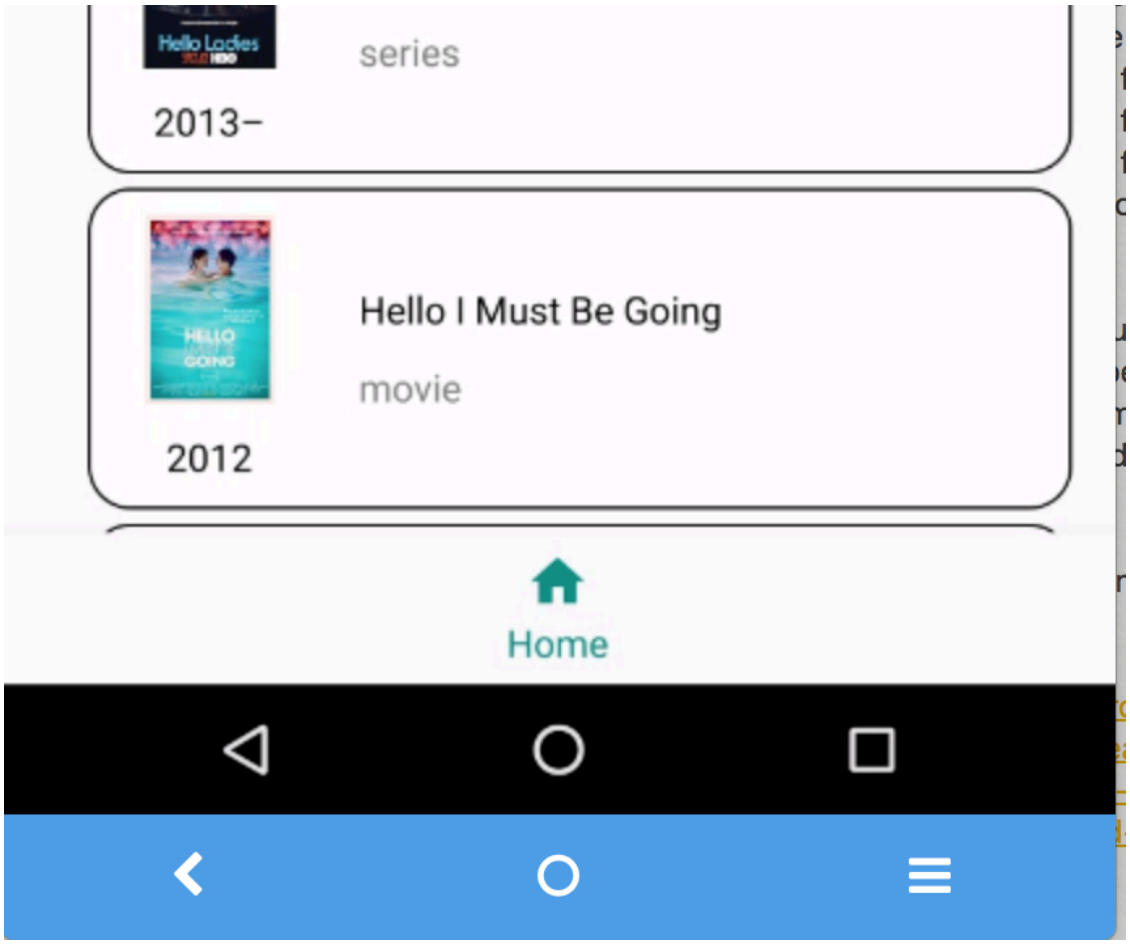
core module

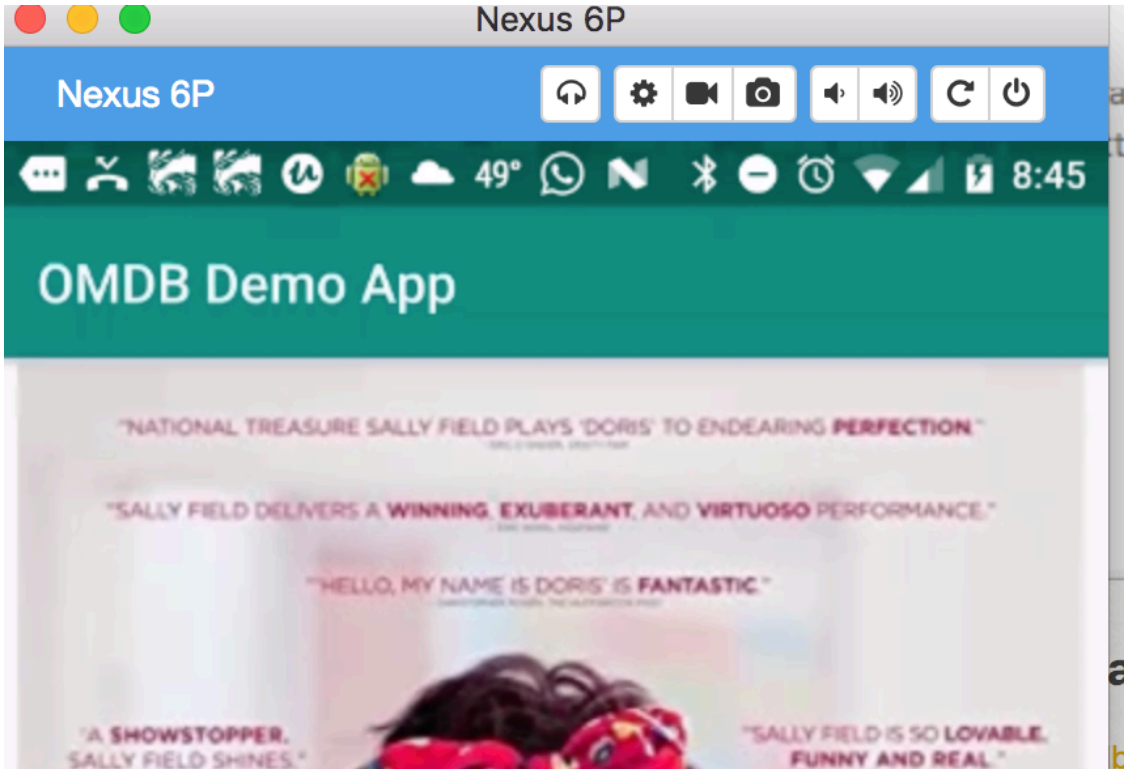
- Provide a dependency injection from services like gson, retrofit

References

https://github.com/googlesamples/android-architecture
https://medium.com/stepstone-tech/clean-architecture-with-reactive-use-cases-c943d7a8f69c
https://medium.com/@yoelglus/android-and-clean-architecture-the-use-case-interface-8716512f29a1 < This is translated to kotlin with co-routines >
https://www.toptal.com/android/android-apps-mvvm-with-clean-architecture




series
2013–
Hello I Must Be Going
movie
2012
Home


Nexus 6P
Nexus 6P
8:45
OMDB Demo App
"NATIONAL TREASURE SALLY FIELD PLAYS 'DORIS' TO ENDEARING PERFECTION."
"SALLY FIELD DELIVERS A WINNING, EXUBERANT, AND VIRTUOSO PERFORMANCE."
"HELLO, MY NAME IS DORIS' IS FANTASTIC."
"A SHOWSTOPPER. SALLY FIELD SHINES."
"SALLY FIELD IS SO LOVABLE, FUNNY AND REAL."

Poster From OMDB!