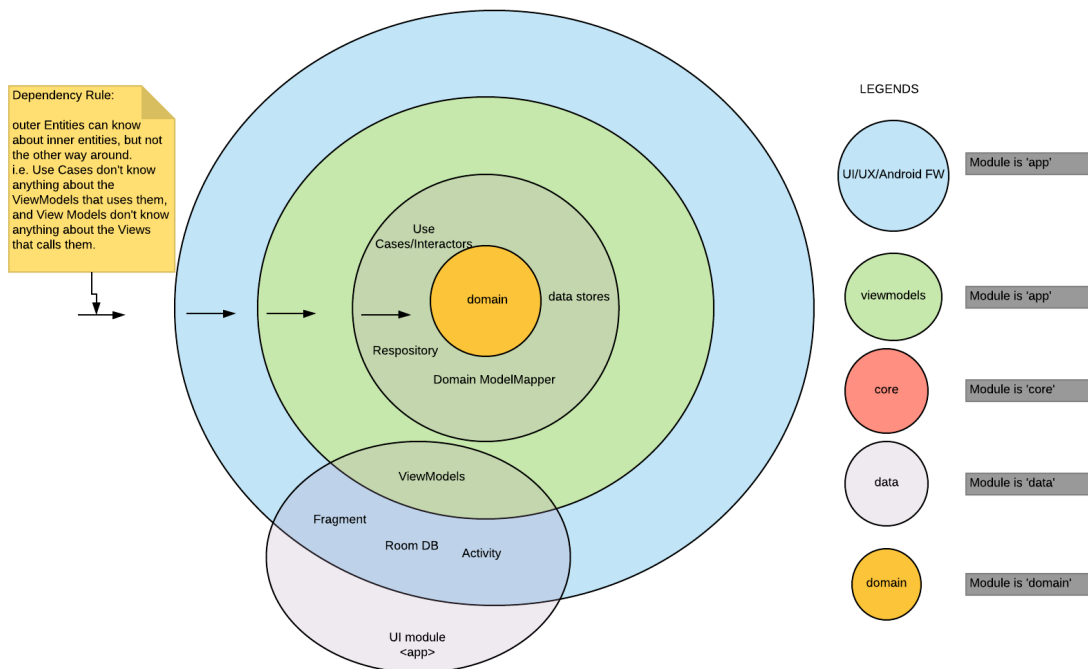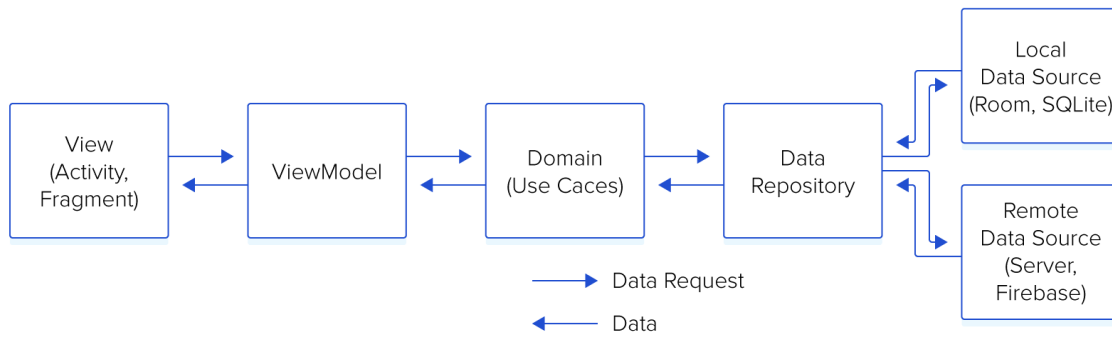# Documentation

Project Architecture Diagram

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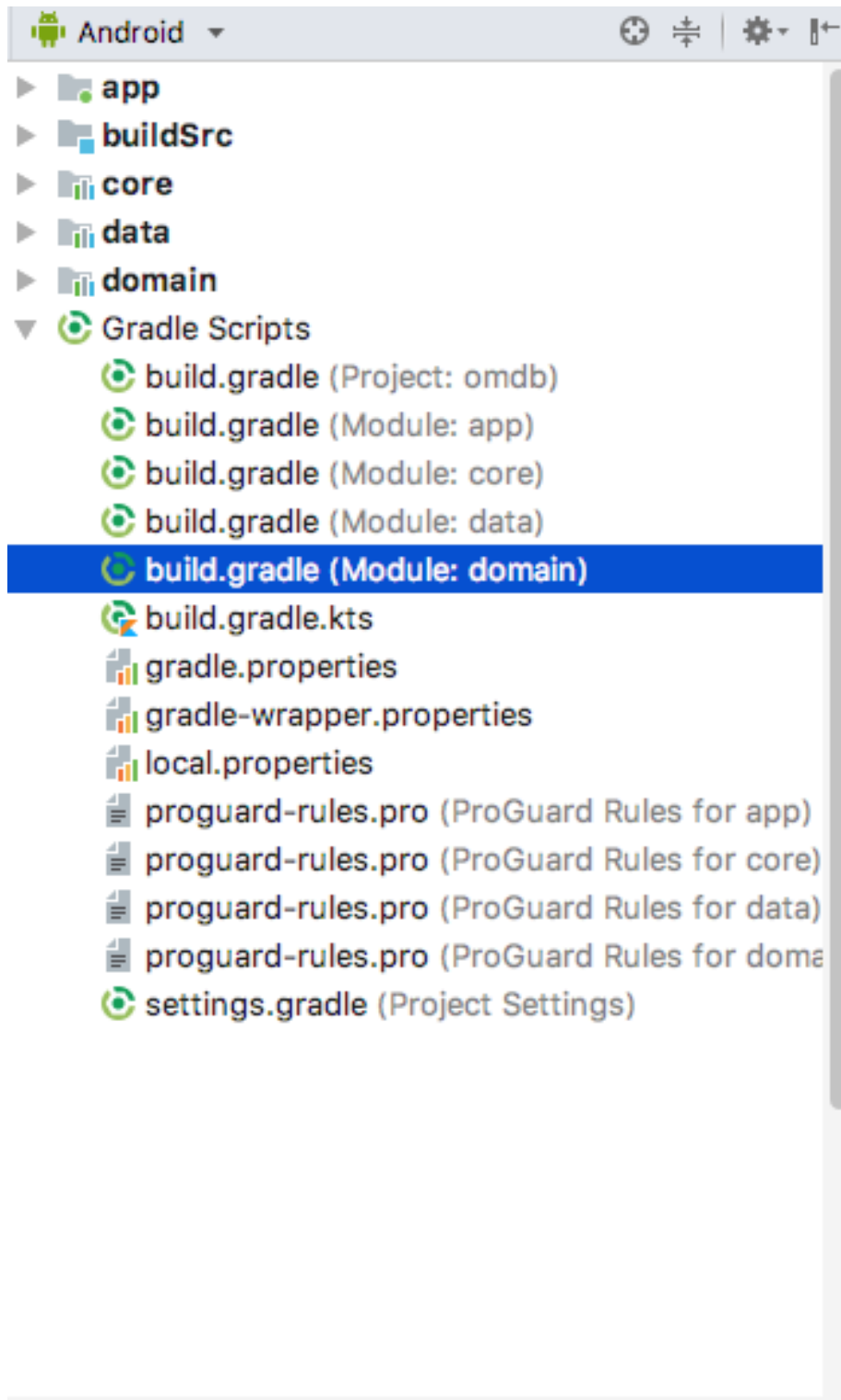
Below listed diagram actually maps the clean architecture to OMDB Application modules and how the

dependency is allowed to flow.

Dependency Rule:

outer Entities can know about inner entities, but not the other way around. i.e. Use Cases don't know anything about the ViewModels that uses them, and View Models don't know anything about the Views that calls them.

LEGENDS

UI/UX/Android FW	Module is 'app'
viewmodels	Module is 'app'
core	Module is 'core'
data	Module is 'data'
domain	Module is 'domain'

Data Flow

Diagram flow:

View (Activity, Fragment) → ViewModel → Domain (Use Caces) → Data Repository → Local Data Source (Room, SQLite)

Data Repository → Remote Data Source (Server, Firebase)

Legend:
→ Data Request
← Data

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
- Repository interface is defined for the Application
- Model object is defined . 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

- 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

- 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
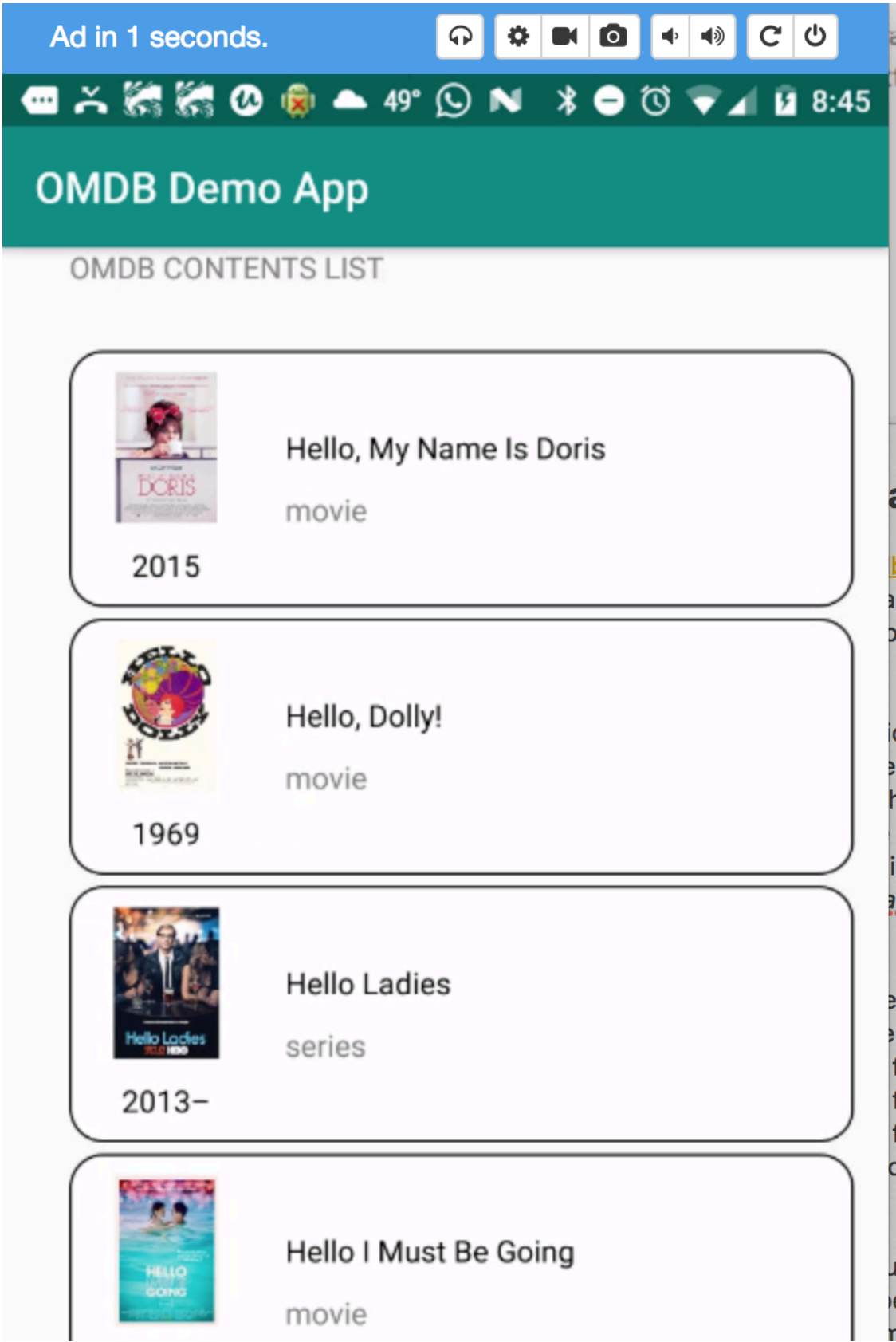- AppComponent provides required dependency injection

core module

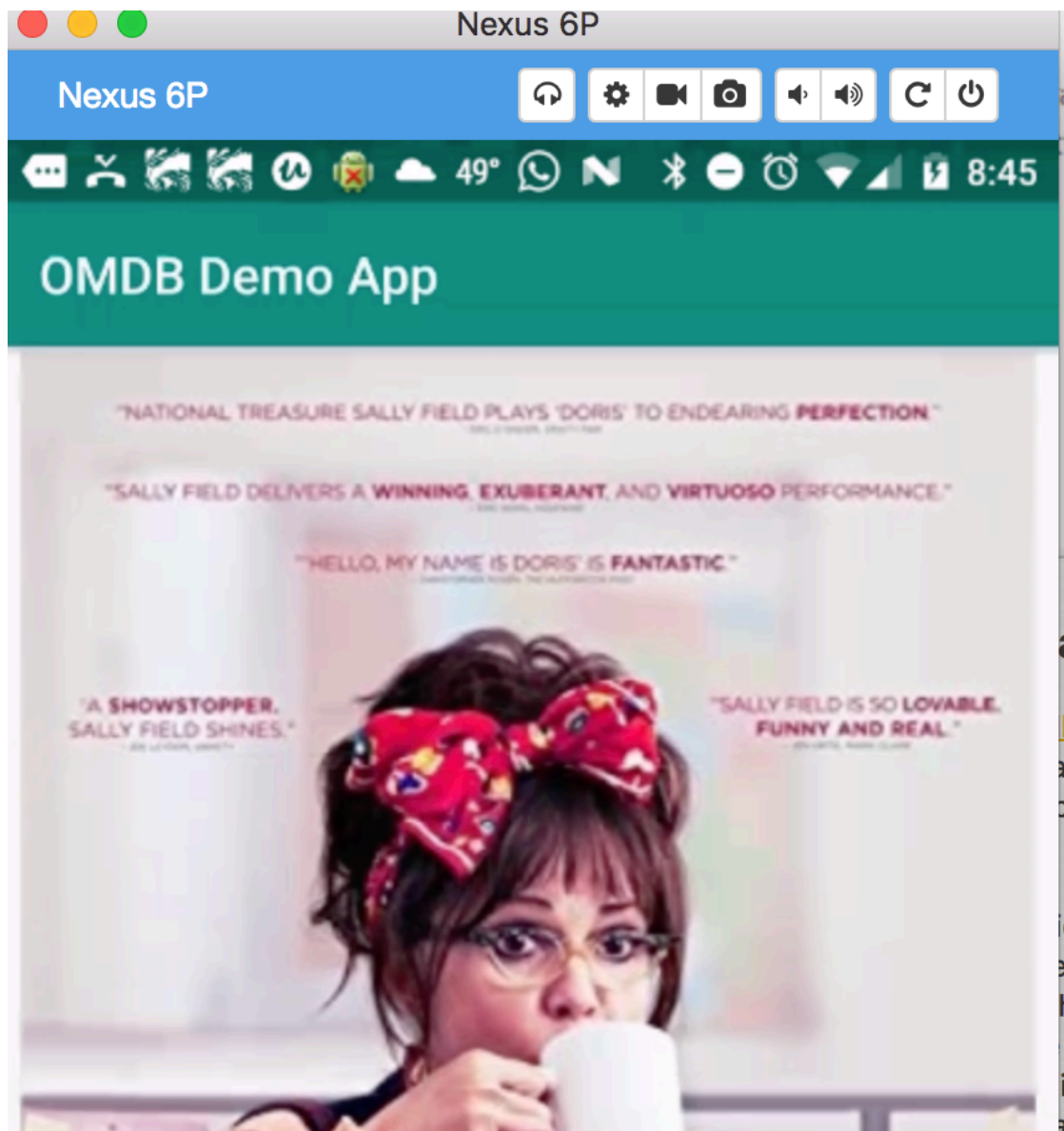- Provide a dependency injection from services like gson, retrofit

References

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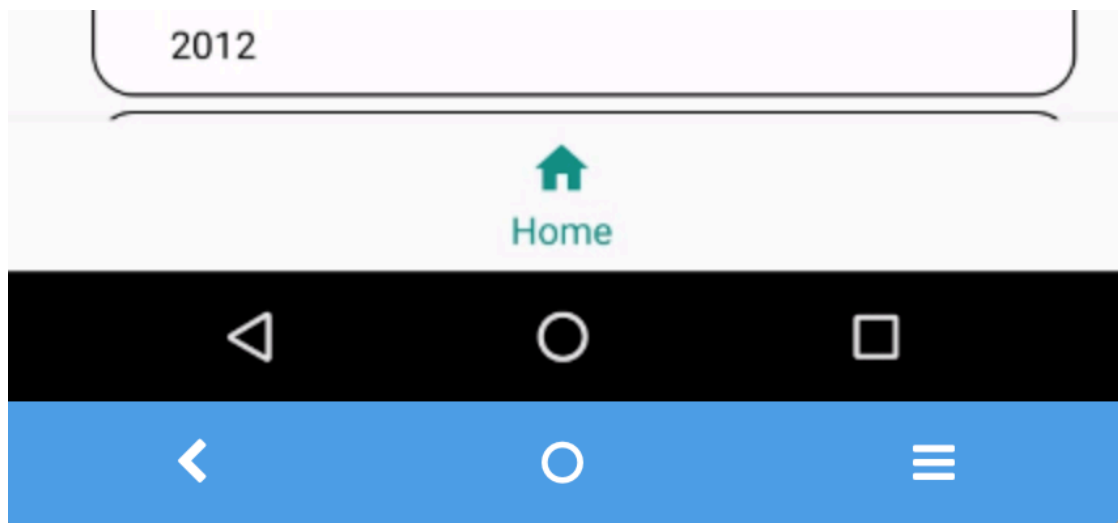https://medium.com/stepstone-tech/clean-architecture-with-reactive-use-cases-c943d7a8f69c
https://medium.com/@yoelglus/android-and-clean-architecture-the-use-case-interface-8716512f29a1 < This is translated to kotlin with co-routines >

Ad in 1 seconds.

49° 8:45

OMDB Demo App

OMDB CONTENTS LIST

Hello, My Name Is Doris

movie

2015

Hello, Dolly!

movie

1969

Hello Ladies

series

2013–

Hello I Must Be Going

movie

2012

Home

Nexus 6P

Nexus 6P

49°   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!