The app was developed using XCode 12.3 on mac version OS 11.1 “Big Sur”. “MVVM” is the design pattern implemented in the code base. There are varied definitions on the internet of what “MVVM” is so I’m going to give me interpretation of the definition.

In “MVVM” the “View” is treated as a dumb client. It only contains elements from UIKit and contains little or no logic. The “ViewModel” drives the “View”. The “ViewModel” contains all the logic that drives the “View” and has NO reference to anything in UIKit so NO “import UIKit” statement. Normally there are only one “ViewModel” for every view but it's possible to have more than one. I have not seen this in my past projects.

Public methods & delegation is the communication mechanism between the “View” and the “ViewModel”. The delegation design pattern is used for asynchronous communication and the pubic method are basically the getters that get data to the view.

The “Model” is the data models used in the app. These are structs used to create types from json dumps.

Now I’m going to go through each module and explain its purpose. All the code is in the “SourceCode” directory.

The “Models” folder contains 2 data models “DataModel” & “GenreDataModel”. The “DataModel” is used for creating types for “top movies” and the “GenreDataModel” is used creating types for genres. Both implement the “Codeable” protocol for decoding purposes.

The “Views” folder contains 2 “Views” and 2 custom cells. “TopMoviesView” is used to display top movies & the “TopMoviesDetailView” is used to display movie details. Both views mostly contains UI elements & interacts with the view models to get data to display.

The “ViewModels” folder contains 2 “View Models”, “TopMoviesViewModel” and “TopMoviesDetailViewModel”. Both view models is used to get data to the view for displaying. Both “ViewModels” contains all the logic that drives the view.

The “Services” folder contains 1 service. The “Services” module gets data from the different end points for the app. You pass in a “Request” object to the service & it will return a “Response” object. Internally the service will know what to return. The service always returns a “Response” object. This module could be moved to framework or package to be used in other apps.

The “Common” folder contains 2 modules “Utils” & “Configurations”. The “Utils” module contains utilities used through out the app. The “Configurations” module contains api key & url’s to the different end points.

My primary focus was not so much on the UI but implementation of MVVM & a well decoupled app.