Application

KCMarvel

Topics

Classes

`class CharactersRepository`

Creation of this intermediate layer between Data and Domain to carry out a CLEAN architecture, created to obtain Marvel characters. Network variable created as private so that it is inaccessible outside the same class.

`class CharactersRepositoryMock`

This mockup was created both to work on the project locally and to test that the repository layer works correctly. Network variable created as private so that it is inaccessible outside the same class.

`class CharactersUseCase`

Intermediate layer between repository and ViewModels, to manage the logic while maintaining a CLEAN architecture

`class CharactersUseCaseMock`

Mock created for testing

`class CharactersViewModel`

Using viewModel with Observable to listen for changes to its properties.

`class NetworkMarvel`

Structure for network calls, both to obtain the characters and to obtain the series of them, I use Components to create the network call in a safe and orderly manner. I have not created the body in the call since we are working with GET

`class NetworkMarvelMock`

The creation of this mock has been to test our call locally and thus not make real network calls and for testing

`class SeriesForHeroRepository`

Creation of this intermediate layer between Data and Domain to carry out a CLEAN architecture, created to obtain Marvel series. Network variable created as private so that it is inaccessible outside the same class

`class SeriesForHeroRepositoryMock`

This mockup was created both to work on the project locally and to test that the repository layer works correctly. Network variable created as private so that it is inaccessible outside the same class.

`class SeriesForheroUseCase`

Intermediate layer between repository and ViewModels, to manage the logic while maintaining a CLEAN architecture

`class SeriesForheroUseCaseMock`

Mock created for testing

`class SeriesViewModel`

Using viewModel with Observable to listen for changes to its properties.

Protocols

`protocol CharactersRepositoryProtocol`

Protocol for Characters

`protocol CharactersUseCaseProtocol`

Protocolo for the useCase

`protocol NetworkMarvelProtocol`

Protocols created for network calls in NetworkMarvel

`protocol SeriesRepositoryProtocol`

Protocol for Series

`protocol SeriesUseCaseProtocol`

Protocolo for the useCase

Structures

`struct CharacterRow`

Creating this structure for everything that each section of the hero entails, we check during image unpacking that what we receive is actually a valid image and we customize it, accompanied by its respective name.

`struct CharactersView`

We receive the hero with its respective ID and access SeriesView with its value, representing the cells as we have configured in CharacterRow.

`struct ConstantsApp`

The creation of this struct was to provide the different addresses necessary for network calls.

`struct DataMarvel`

`struct EndPoints`

Structure created for the endpoints, as we need for the series of the desired user, I have created a function so that in this way I can create the complete URL, assigning the series to the hero, address provided in the original Marvel documentation.

`struct Hero`

Structure created to receive the heroes, Codable for decoding

`struct HomeView`

HomeView is the main screen where I give the user a welcome and a waiting time so they can download while the heroes The creation of the variables animateLoading and navigationNextView have been created to manage both the animation and to inform when to move to the next window (as soon as three seconds have passed and it becomes true)

`struct KCMarvelApp`

`struct ResponseMarvel`

Structure required for data decoding, I have worked with generics to be able to recycle the code for both characters and series

`struct SecurityAccess`

I have chosen to create this structure to provide greater security to the project using the Cryptokit framework, which allows me in this case to manage the hash with md5 in the same project and thus avoid leaving it visible.

`struct Series`

Structure created to receive the series, Codable for decoding

`struct SeriesRowView`

Here we manage the row of each series, personalized to maintain independent customization.

`struct SeriesView`

Creating the View to pass the custom row with its respective id and we execute the method to obtain the series when the view is going to appear

`struct Thumbnail`

Support to receive the Marvel images since the JSON comes with the path and its separate format

Enumerations

`enum HTTPResponseCodes`

These are all the different response codes, it is necessary to mention that the 409 response is similar for Missing API Key, Missing Hash, Missing Timestamp

`enum HttpMethods`

We only have one method in the marvel API

Extended Modules

- DeveloperToolsSupport
- Swift
- UIKit