Planets

## About

This application shows a list of planets retrieved from swapi/dev. The list is displayed in alphabetical order. It also shows a few details of each plant in the list.

The application can be run on iPhone, iPad and iPod touch in both landscape and portrait mode.

## Technical highlights/components/frameworks used

Language – Swift 5.0

Xcode – 12.4

Minimum deployment target – 13.0

Networking – URLSession

Persistence storage – Codedata

## Instructions to run

Down the repo from Github. Run the app in Xcode choosing any simulator. You should be good to go.

## UI Flow

1. On launching the app, a splash screen is displayed for couple of seconds.

Graphical user interface, application

Description automatically generated

1. If there is a previous data available, no service call is made and you should see the list of planets immediately.

Graphical user interface, application

Description automatically generated Graphical user interface, application, Teams

Description automatically generated

1. If there is no previous data, a service call is made to get the data. You can confirm this by seeing an activity indicator on the screen. Once the data is received, the indicator disappears and you are provided with the list of planGraphical user interface, application

   Description automatically generated A picture containing application

   Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

1. There is a refresh button on the top right corner. Tapping on this button deletes any local data to make a fresh call to server.

Graphical user interface, application, Teams

Description automatically generated

# Future Enhancements

## Functional

1. Since all the data is w.r.t Star Wars, could rename the app is StarWars App.
2. Should have a tab bar with 6 options – Planets, Films, People, Species, Star ships and Vehicles. Each would first show a list of the appropriate tab and then the details.
3. Planets could have a the list setup such that the more you scroll, more data gets revealed till all of it is exhausted (since the api has a previous and next params).
4. The list could also have a pull to refresh to implement the refresh part when we don’t want to read from persistent store.

## Technical

1. Could have 1 base service manager which would contain all the common methods like making download task along with the existing data ask with the support for all type of http calls – post, get etc.
2. Could contain 1 base data manager which would contain common methods like json parsing.
3. Could also separate the core data setup from app delegate to a coredata manager.
4. Long shot – but if we ever want this to been in mac as well, could try designing it with SwiftUI.