20.0 hours

Project 6

The API Awakens App

Why can't I submit this project yet?

You must first complete all prior Techdegree content before you can submit this project. Additionally, you cannot submit more than one project at a time.

Submit for Review

Your current activity is Dependency Management with Carthage for iOS.

- Instructions
- How you'll be graded

A short time ago, in a Galaxy not so far away, you were taught about APIs. Now is your chance to harness the force (of newfound knowledge) and put the Star Wars API (SWAPI) to use in an iOS app.

Using what you learned about APIs, Networking, Concurrency, and JSON you will get information about three types of Star Wars entities: people, vehicles and starships. Each entity type should have its own view from an UI perspective. However, they can be based on common code and/or storyboards. See below for links to the mockups for the app. Please note that even though the API is paginated, for the basic requirement of the project you only need to retrieve and display the first page of results for a given API request.

In the mockup, you will notice that regardless of which view a user is on, there is a bar across the top showing the largest and smallest member of the group. In addition, because all measurements are given in metric units (meters), you will need to create a feature that convert the units to British units (inches), at a tap of a button. For starships and vehicles, students will need to create a button that can convert "Galactic Credits" to US Dollars, based on a exchange rate provided by the user in a text field.

Lastly, please include appropriate error handling. The app needs to demonstrate the ability to anticipate and handle errors for at least the following possible issues:

- The device went offline during an API call
- A user entering a 0 or negative exchange rate
- An error resulting from a key or element missing from the JSON returned from the API

If you would like to get an "exceeds expectations" grade, you will also need to

- Implement an additional text view on the "people" screen that displays a list of vehicles or starships that are associated with that person
- As mentioned earlier, make sure the API requests retrieve all pages of results, and any value of the entire result set can be selected by the user

Before you start

To prepare for this project you'll need to make sure you complete and understand these steps.

2 steps

- **Make sure to check the provided mockups. They'll help you to understand important aspects of the instructions.**

- **Read up on the SWAPI API (https://swapi.co), look carefully at the format and what it returns. Certain items, like birth year, return "unexpected" data types.**

Project Instructions

To complete this project, follow the instructions below. If you get stuck, ask a question on Slack or in the Treehouse Community.

10 steps

- **You can download and use the empty Swift starter files template to start your project.**

- **Create the appropriate types for people, vehicles, and starships. Consider your options in terms of structs, classes, composition, inheritance, etc.**

- **Create asynchronous networking code to retrieve JSON results from the SWAPI API. Make sure your code is reusable for the different entities (people, vehicles, starships) that you will be handling and displaying.**

- **Create logic to parse the JSON result and display the names of all members of the entities in the Picker Wheel. You may need to check documentation for guidance on how to use the Picker control.**

- **Create three screen layouts which users can switch between. One each for people, starships, and vehicles. Make use of reusable elements, storyboards, etc. as you see fit. One possibility is to use a single layout and change/show/hide UI elements programmatically.**

- **Create logic such that when a user selects from the pick wheel, all the related fields on the screen are being populated.**

- **Create logic to populate the Quick Facts Bar. You may want to use generics for this.**

- **Create a feature such that a user can convert metric units (meters) to British units (inches) with a tap of a button.**

- **Create a feature such that a user can input an exchange rate in a text field and then convert between Galactic Credits and US Dollars.**

- **Add error handling which include, but not limited to:**

    - The device going offline when an API call is in progress
    - A user entering a 0 or negative exchange rate
    - An error resulting from a key or element missing from the JSON returned from the API

Extra Credit

To get an "exceeds" rating, you can expand on the project in the following ways:

2 steps

- **When a person is selected, all (if any) associated vehicles and starships are listed as well**
- **All the data returned from the API is used to populate the picker, not just the first page of returned data**
- **NOTE: Getting an "Exceed Expectations" grade.**
    - See the rubric in the "**How You'll Be Graded**" tab above for details on what you need to receive an "Exceed Expectations" grade.
    - Passing grades are final. If you try for the "Exceeds Expectations" grade, but miss an item and receive a “Meets Expectations” grade, you won’t get a second chance. Exceptions can be made for items that have been misgraded in review.
    - Always mention in the comments of your submission or any resubmission, what grade you are going for. Some students want their project to be rejected if they do not meet all Exceeds Expectations Requirements, others will try for all the "exceeds" requirement but do not mind if they pass with a Meets Expectations grade. Leaving a comment in your submission will help the reviewer understand which grade you are specifically going for

Download files

Zip file

Project Resources

File Download

Screen Mockups

Need Help?

Have questions about this project? Start a discussion with the community and Treehouse staff.

Get Help