Flutter task

Task definition

Your task is to create a catchy *Flutter* app that presents data from a REST Space-X API. The architecture and design of this application are up to you, you just need to meet some requirements listed below.

You can get the data from a REST Space-X API here.

Present at least 2 separate Space-X API sections in your application. Please present *Past and Upcoming Launches* and another section of your choice (e.g. *Company Data, Rocket Data, Capsules, Crew, Launchpads,* etc*.)*

One section should contain a list and detail screen. One screen with a list view of available data and a different screen with the detail. The detail is displayed when the user taps on the list item.

Add a *SearchBar* to the *List view* so that users can search for specific data. Users can decide by which parameters the data are ordered or filtered by various parameters (*e.g. allow filtering by launch_year and rocket_id in Past Launches*).

Add functionality that allows users to save the current data filter. Save this parameter and apply it on every app launch.

For navigation between sections use navigation of your choice. Either *Drawer* menu, *BottomNavigationBar*, *TabBar*, *SegmentedControl,* or whatever you like. But don't forget that your app has to have at least two sections.

The design and architecture of the app are completely up to you. We will be reviewing your code, not your design, so don’t worry about using default Material or Cupertino widgets.

Make sure to support *Landscape* and *Portrait* modes and make the app runnable for mobile devices. Web, desktop, and tablet versions are optional but they will give you some plus points.

applifting.cz
info@applifting.cz

Filip Kirschner | COO
+420 739 627 797

Requirements

- Use **the latest stable Flutter** version
- Use **http** package for all networking
- You can use a JSON parser of your choice (*we recommend **json_serializable** package*)
- The code should be production-grade. Use null-safety.
- Use **Git** for version control. Create a git repository and share it with us once your project is finished. Create two branches, **Main** and **Develop**. Make regular commits to *Develop* with descriptive commit messages in the imperative. After you finish and test everything, create version 1.0 of your app by merging *Develop* to *Main*.

Tips and advice

- Don't try to reinvent the wheel, make good use of **Google** and **Stack Overflow!**
- You can use any other packages from pub.dev. But be prepared to justify why you did so.
- You might use git *Conventional Commit Messages*, it's completely up to you. Please be consistent in your repo. Make sure you know what the *Gitflow workflow* is.
- Be consistent in your code. Conform to Flutter style guidelines and Effective Dart guide.
- Unit test coverage is optional, but we would love to see some. Widget tests would be also nice
- Error scenarios should be taken into consideration. Users should always know what's wrong and have the option to reload the data if possible.
- You can employ persistent memory (*save and load data, e.g. cache start screen data into persistent memory to be able to show it immediately after start without waiting for the API request to finish*)
- Although UI and UX are important, we are more concerned with your thought processes and how you architect your application. Consider and take into account features that might be added in the future (*and have your code reflect that*).
- Be opinionated regarding any architecture you use and take your time to make it a reflection of your thought process.

Alright, that's it. Good luck and, most importantly, **have fun!**

applifting.cz
info@applifting.cz
Filip Kirschner | COO
+420 739 627 797