Lists

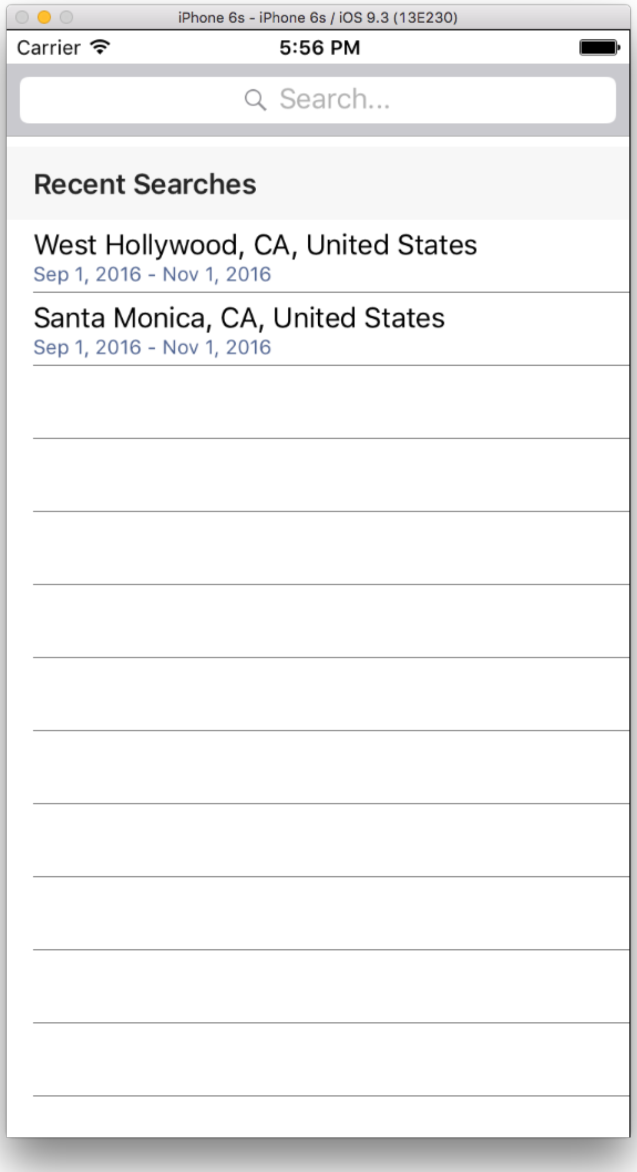
By: Mosh Hamedani

Exercise

This exercise is a bit more challenging than the previous ones. So, do your best to build this app with the best practices you know.

Once you're done, look at my solution and see if there are any areas that you can improve your code.

Remember: it's the challenges that help you grow.



Recent Searches

West Hollywood, CA, United States
Sep 1, 2016 - Nov 1, 2016

Santa Monica, CA, United States
Sep 1, 2016 - Nov 1, 2016

Scenario

Airbnb is a service that enables people to find and rent vacation homes.

The user can search for rentals and apply filters (eg price, number of bedrooms, etc). They can save their searches so they can get back to them later.

These searches are stored in Airbnb's database and they're available both on their website as well as their app.

So, we're assuming that our simple app is going to fetch the saved searches via a remote service. For the purpose of this exercise, hardcode the list of searches in the app, similar to what I've done before.

Lists

By: Mosh Hamedani

Requirements

1. We have one group called “Recent Searches”. All searches are under this group.

2. User should be able to filter the results by location. Filter should be case-insensitive.

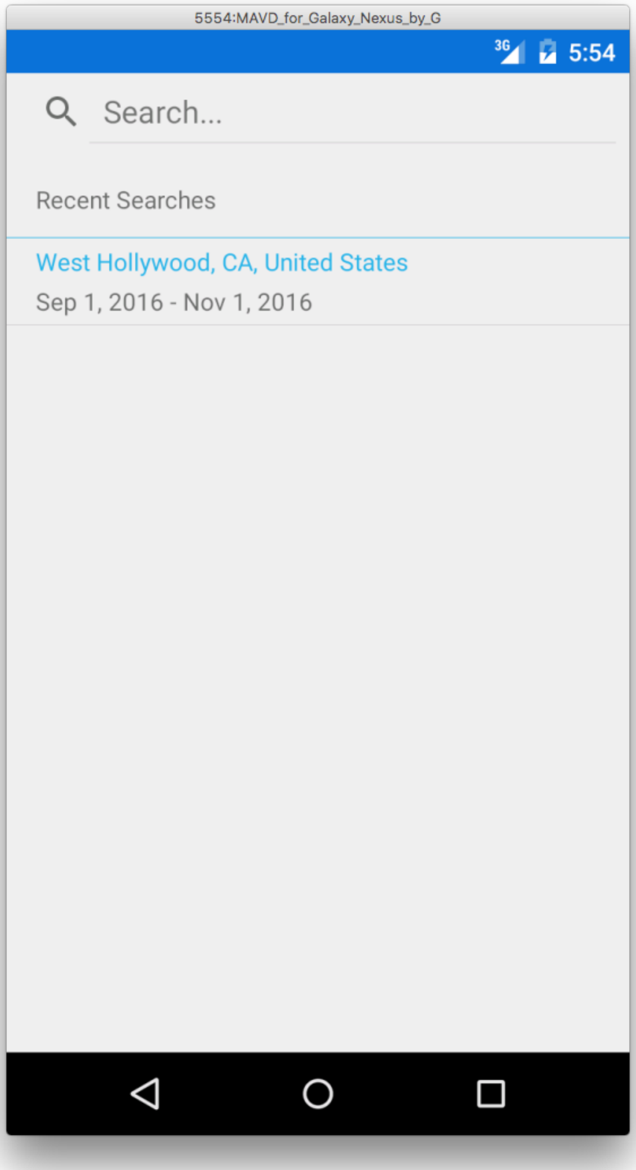
3. User should be able to delete a search.

4. User should be able to refresh the list by pulling it down. If a search is deleted, it should not appear once the list is refreshed.

5. If there is a filter applied to the list, it should be taken into account while refreshing the list.

6. When a search is selected, its location should be displayed in an alert.

7. There should be 20 units padding on top of the page for iOS.


5554:MAVD_for_Galaxy_Nexus_by_G
3G
5:54
Search...
Recent Searches
West Hollywood, CA, United States
Sep 1, 2016 - Nov 1, 2016

Read the architectural requirements on the next page before you get started.

Architectural Requirements

Instead of writing all the code in the code-behind, you should separate out various concerns into different classes. Here are some guidelines:

Service

All the communication with the remote service should be encapsulated within a class called **SearchService**. This class should expose two methods:

```
IEnumerable<Search> GetSearches(string filter = null)
```

```
void DeleteSearch(int searchId)
```

Internally, it keeps a hardcoded list of searches. If in the future we decide to replace the hardcoded list with a call to a remote service, we only need to modify this class. The rest of the application (eg our pages) are unaware of this.

Models

You should have two models: **Search** and **SearchGroup**, which is a collection of searches.

Each Search should have the following properties:

- Id (int)
- Location (string)
- CheckIn (DateTime)
- CheckOut (DateTime)