Upcoming Events

Date: February 2, 2020

By: Tito Ciuro

Main Features

- loads the events from mock.json asynchronously.
- displays the events showing title, start time and end time.
- the events are displayed in chronological order.
- the events are grouped by day.
- when a conflicting event is detected, a warning icon appears in the event.
- tapping a conflicting event will show a list with the related event(s).
- detect AM-PM/24 hour format changes and reloads the Upcoming Events controller or the Event Info controller. This setting can be set in Settings > General > Date & Time > 24-Hour Time.
- works in Dark Mode.

Notes

APPLICATION

- app launch time varies between 50 to 150 ms.
- an activity indicator is shown while the events are being loaded. For the purpose of the demo, an artificial delay has been introduced to show that, if the mock file was larger and it took longer to load the events, that the activity indicator would let the user know it's busy.
- no storyboards. All implemented programmatically.
- no 3rd party libraries.
- built with Xcode 11.3.1, as found in the AppStore on February 2, 2020.

MODEL

- an *Event* encapsulates a title, start time and end time.
- a *Day* represents a distinct date which contains one or more events sorted chronologically while keeping track of conflicting events.

SERVICE

- the service adopts the *EventDataServicing* protocol.
- the idea being that different services could be written (e.g. *FileEventDataService*, *NetworkEventDataService*) and managed by *EventDataService*, the umbrella service that would route the request accordingly.
- the service vends objects of type *Event* and *Day*.

CONFLICT DETECTION ALGORITHM

- the algorithm itself is linear, but only because the events are already sorted chronologically (see Model > Day). Since the algorithm requires a pre-sorted timeline, it's O(nlogn) to sort + O(n) to traverse the timeline = O(nlogn).

- the conflict detection is optimized because it's performed on a day-per-day basis, lazily, on demand.

- however, because the Upcoming Events UI displays a warning sign in the cell of the conflicting event, this triggers the conflict detection mechanism. In this case, the optimization is a moot point. However, a different UI not requiring conflict detection beforehand could benefit from a nice performance boost.

ARCHITECTURE

- since both UpcomingEventsVC and EventConflictVC display a table view with events, I have architected them around a single controller which manages the events table view logic. Both UpcomingEventsVC and EventConflictVC adopt the DailyEventsVC as a child view controller. This allows the code to be modular

and highly reusable.
- *UpcomingEventsVC* uses MVP.
- *EventConflictVC* and *DailyEvents* uses MVC since the event passed from *UpcomingEventsVC* is the data to be displayed. That is, there is no need to access the service, and for this simple case a *Presenter* would be a bit redundant.

SOURCE CODE

- the code has been annotated, especially the API.

UNIT TESTS

- a few unit tests have been added to test and verify the core functionality.
