MagicShop and Watch Mobile App

Magayanes, Janine Aira

janine.magayanes@code.berlin

Technical Documentation
Module: Collaboration SE_O7
Semester: Winter 2020

The first mobile app I developed is an ecommerce store using React Native with Firestore for the database. By going to the source folder, you can see the following sub-folders and App.js:

1. App.js handles the app startup, routing and other functions of the application. By creating a react native app, this file is automatically populated. It uses the following libraries:

- React is for built in hooks like “usestate”

- Redux maintains the state of an entire application in a single immutable state tree

- React Redux lets React components read data from a Redux store, and dispatch actions to the store to update data.

- Redux Thunk middleware allows writing action creators that return a function instead of an action.

App.js is also where the store is initiated and fonts are fetched. The root reducer serves as the summary of all the available reducers for the app. The AppLoading component is used to render an initial component while waiting for the login page.

2. Below are the list of folders and contents available on this app:

• Expo: dependencies needed in running it in Expo.

• Assets: fonts, images and other media

• Components- Shop: all components related to the shop e.g cart, order and product items

• Components - UI: all components that are used for design: card, header and input items

• Constants: for reused constants e.g colors and fonts

• Data: dummy items before the Firestore database was integrated

• Models: files that define the specifications of the Items in the database.

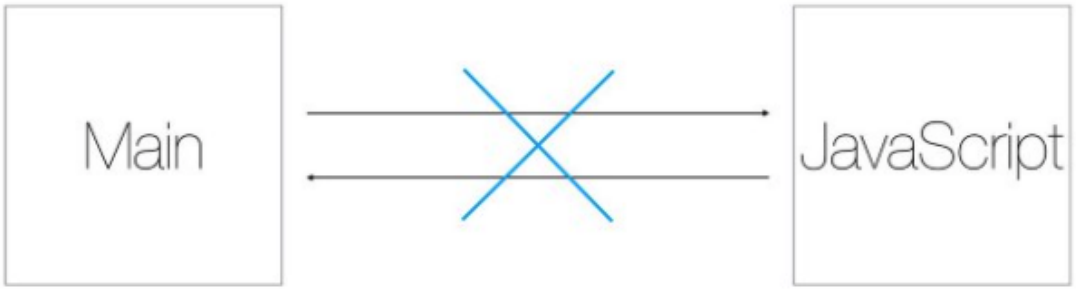
• Navigation: components that are related to the navigation

• Screens: components that are used on specific views/ screens

• Store - Action: transforms data or making AJAX requests

• Store - Reducer: contains the logic for specific action

There are two threads running in the application as shown in the diagram below:



First is the main thread that runs in each standard native app. It displays the ui elements and processes user gestures. The second thread is specific to React Native. This executes the JavaScript code in a separate JavaScript engine. The JavaScript handles the business logic of the application. It also defines the structure and the functionalities of the user interface.

Between these two threads is the React Native bridge, which is the core of React Native. The bridge has three important characteristics:

Asynchronous - It allows the asynchronous communication between threads so they never block each other.

Batched - It delivers messages from one thread to the other for optimisation.

Serializable - The two threads never share or operate with the same data. Instead, they exchange serialized messages.

The authentication is handled from the Firebase end through an identity toolkit. Tokens are used to identify current session and user.

Sources:

1. React/ React Native Documentation https://reactnative.dev/docs/getting-started
2. Firebase Documentation https://firebase.google.com/docs/?gclid=CjwKCAiA2O39BRBjEiwApB2Ikna2BrQ8uOwvqX6-vbGmjvKA_-nnJ95XjH9jpmSrv93WgcSeLuM7WhoCEC8QAvD_BwE
3. Academind https://www.youtube.com/watch?v=6ZnfsJ6mM5c

The second app is a simple TV app for an Just Watch. This uses mobx state tree to handle the store and some actions. In the source folder, the App.js contains all the components and screens. By calling the observer app, the App is given access to the store in the MovieStore.js file. It also uses the carousel component for better scrolling experience.

Source:

1. Mobx State Tree Documentation https://mobx-state-tree.js.org/intro/getting-started
2. React Native Conference https://www.youtube.com/watch?v=l32binaPlSY&t=1357s