Using React Native in an Android App

Yifan Gu, Chaohui Xu, Mao Zheng
Department of Computer Science
University of Wisconsin-La Crosse
La Crosse WI, 54601
mzheng@uwlax.edu

Abstract

Nowadays there is a mobile app for many different needs: ride sharing, finance, gaming, insurance claims, email, music, etc. If you can imagine it, it is probably available for download. More importantly, as users interact with companies, they expect to do business with them via their mobile devices. In our project, we chose to develop a native app instead of a hybrid app for the reason of a better user experience. A native app is a smartphone application developed specifically for a mobile operating system. The apps that came with the phone for sending text messages, taking pictures, or setting reminders are native apps. Hybrid apps are, at core, websites packaged into a native wrapper. Basecamp, Instagram, Yelp, or your mobile banking apps are usually hybrid apps.

In this paper, we are developing a lite version of a ride sharing mobile app to mainly introduce Reactive Native in the client side mobile app development. React, sometimes called React.js or ReactJS, is an open source JavaScript library for developing web user interfaces. React Native allows developers to build mobile apps using only JavaScript. It uses the same design as React, letting developers compose a rich mobile user interface from declarative components, by putting fundamental user interface building blocks together using JavaScript and React.

In our project, a mobile app has two main types of users: riders and drivers. A rider can request a car by providing his/her preferred start location and the destination of the trip. The rider can also view the cost of the trip, pay and rate the driver after the trip. A driver can view all the requests from different riders, select a request, and pick up the rider. It requires real-time communications between the rider and the driver. The information about the trip and all the requests will be recorded into a database. The project is hosted in Microsoft Azure cloud platform. React Native is used in developing the client mobile app. Messages between the server and the client app are sent via Google Cloud Messaging. MongoDB is a backbone database, and Redis is used to as the cache to allow a fast system response.

1 Introduction

Nowadays there is a mobile app for many different needs: ride sharing, finance, gaming, insurance claims, email, music, etc. If you can imagine it, it is probably available for download. More importantly, as users interact with companies, they expect to do business with them via their mobile devices. When we enter the world of mobile app, there are currently two choices, a native app vs. a hybrid app. Native apps are native to the user's OS and are hence built using those guidelines. The apps that came with the phone for sending text messages, taking pictures, or setting reminders are native apps. Hybrid apps are, at core, websites packaged into a native wrapper. Basecamp, Instagram, Yelp or your mobile banking apps are usually hybrid apps.

In our project, we chose to develop a native app instead of a hybrid app for the reason of a better user experience. It also has significant advantages since a native app is able to easily access and utilize the built-in capabilities of the user's devices, like GPS or camera.

We are developing a lite version of a ride sharing application to mainly introduce Reactive Native in the client side mobile app development. React, sometimes called React.js or ReactJS, is an open source JavaScript library for developing web user interfaces. React Native is a React-like framework for building native applications using only JavaScript. It uses the same design as React, letting developers compose a rich mobile user interface from declarative components by putting fundamental user interface building blocks together using JavaScript and React.

In our project, a mobile app has two main types of users: riders and drivers. A rider can request a car by providing his/her preferred start location and the destination of the trip. The rider can also view the cost of the trip, pay and rate the driver after the trip. A driver can view all the requests from different riders, select a request, and pick up the rider. It requires real-time communications between the rider and the driver. The information about the trip and all the requests will be recorded into a database. The project is hosted in the Microsoft Azure cloud platform. React Native is used in developing the client mobile app. Messages between the server and the client app are sent via Google Cloud Messaging. MongoDB is a backbone database, and Redis is used to as the cache to allow a fast system response.

2 Using React Native in Building Mobile App

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI from declarative components. It also reduces the development time. Instead of recompiling, the developer can reload the app instantly. With hot reloading, the new code can be run while retaining the application state. We mainly discuss the three technical perspectives used in our user interface design below.

2.1 JavaScript Syntax Extension JSX

JSX is a preprocessor step that adds XML syntax to JavaScript. It allows us to mix XML with JavaScript, so that we can declare UI components easily and neatly.

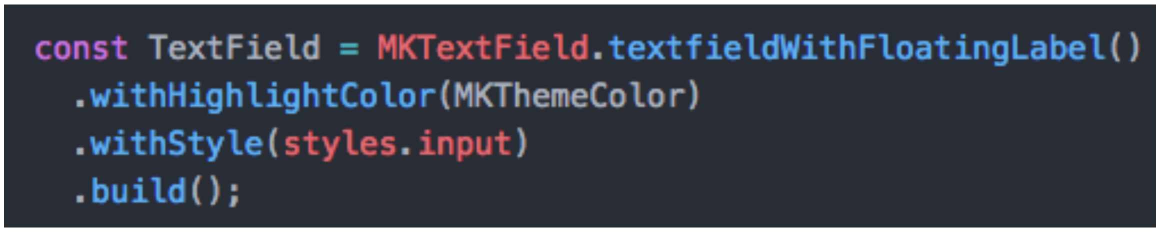
The following code segment shows the example of JSX used in our development for login screen.

```
render(){
  return(
        <View style={styles.container} primary={themeColor}>
         ...
     <TextField placeholder="Email" onChangeText={(email) =>
this.setState({email})}/>
     <TextField placeholder="Password" onChangeText={(password) =>
this.setState({password})}/>
     <View style={styles.loginBtn}>
      <Button text="SIGN IN" primary={themeColor} onPress={this.login.bind(this)}
raised/>
     </View>
     ...
    </View>
  )
 }
```

2.2 Componentization

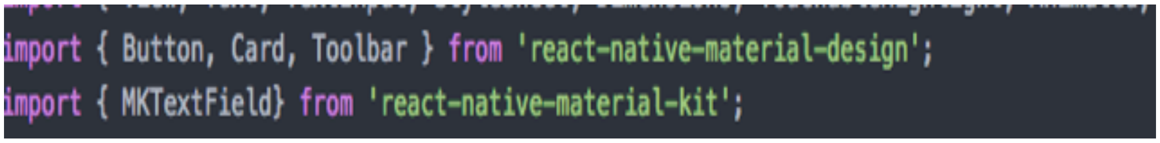
React Native provides some basic user interface components, like "TouchableHighlight" (similar to Button) and "TextInput". Developers can build their own components on top of them. Writing React Native code is like using building blocks. We first use the components provided by React Native to build some higher level components and then use those higher level components to build some other components on top of them. The whole app is built up in this way. All the components in the development process can be re-used.

In our project, we also used some well-developed, open-source components. For example, two React Native components packages are included in the applications: react-native-material-design and react-native-material-kit. These two packages provide a set of React Native components which implement Material Design. Material Design is a UI design language developed by Google. For example, there is a component -- "MKTextField", which is built on the top of React Native's component "TextInput", but instead of a plain text field, MKTExtField is a floating placeholder with ripple effect. So we can just use "MKTextField", otherwise we need to spend extra time on styling the 'TextInput'. Below Figure 1 is the code segment we use MKTextField, and Figure 2 shows we need to import those component packages.



```
const TextField = MKTextField.textfieldWithFloatingLabel()
  .withHighlightColor(MKThemeColor)
  .withStyle(styles.input)
  .build();
```

Figure 1 Code Segment of Using MKTextField



```
import { Button, Card, Toolbar } from 'react-native-material-design';
import { MKTextField} from 'react-native-material-kit';
```

Figure 2 Import Material Design Components

2.3 Layout

The flexbox, is a new layout mode in CSS3. It provides an improvement over the block model. Flexbox is designed to provide a consistent layout on different screen sizes. React Native adopts this new layout mode. The code segment below will ensure that it always stays in the center of the screen and the width is 0.9 of the screen width.

```
const styles = StyleSheet.create({
 container:{
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center'
 },
 input: {
  width: window.width * .9,
  marginTop: 10
 },
 loginBtn: {
  width: window.width * .9,
  marginTop: 10
 }
})
```

3 A Ride Sharing Android Application

We are developing a lite version of a ride sharing application to mainly introduce and practice Reactive Native in client side mobile app development. There are two client side applications, one is for the rider, and the other is for the driver. The rider can request a car by providing his/her preferred start location and the destination of the trip. The rider can also select vehicle type, view the cost of the trip, pay and rate the driver after the trip. The driver can view all the riding requests from different riders within predefined distance, select a request, and pick up the rider. It requires real-time communications between the rider and the driver.

Figure 3 is a screen shot of the rider's app for making a trip request. Figure 4 is the screen shot for the driver's app. The driver received the rider's request.

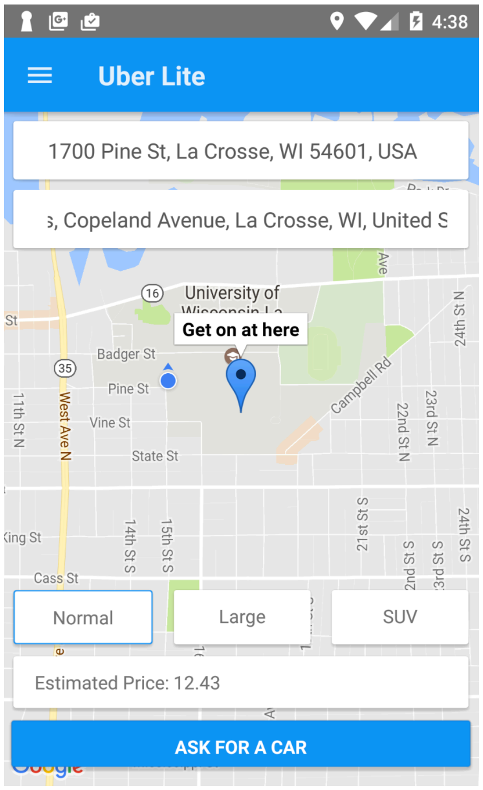


Figure 3 Rider's App

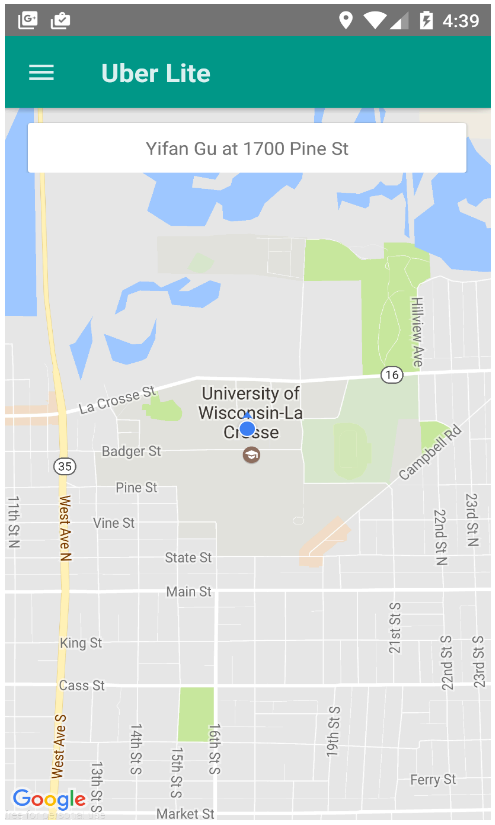


Figure 4 Driver's app

The information about the trip and all the requests will be recorded into a database. The project is hosted in Microsoft Azure cloud platform. React Native is used in developing the client mobile app. Messages between server and client app are sent via Google Cloud Messaging. MongoDB is a backbone database, and Redis is used as the cache to allow a fast system response.

4 Testing

The developers have conducted white-box testing during the development of the app. Black-box testing has been used to integration testing and system testing. The scenario below has been repeatly used to focus on the real-time communications between riders and drivers.

Scenario Testing:

- Rider1 makes trip1 request
- Rider2 makes trip2 request
- Driver1 and Driver 2 are within 10 miles from two trips' start locations
- Driver3 is more than 10 miles far away from the two trips' start locations
- Driver1 and Driver2 both receive the trip1 and trip2 requests, but Driver3 does not
- Driver1 picks up the trip1's order
- Then Driver2 can only picks up the trip2's order

The application has been demonstrated the correct results and all the trip information was recorded in the database correctly.

5 Conclusions

This ride sharing application has been used to practice a number of the latest technologies. In this paper, we mainly introduced React Native for the client apps development, especially in the user interface design perspectives. We have been conducted a number of testing to ensure a small set of functionalities were implemented correctly and efficiently. For our next version, we are interested in adding the reservation functionality so that the rider can request a trip for the next day. In the currently version, all the trip requests are for the current time. A response from a driver must be within 15 minutes or the rider needs to make another trip request.

References

[1] Weiser, M. "*The computer for the 21st century*", Scientific American, 1991 pp. 94-104.

[2] Android Developer's Guide. http://developer.android.com/guide/index.html

[3] https://ymedialabs.com/hybrid-vs-native-mobile-apps-the-answer-is-clear/

[4] https://facebook.github.io/react-native/