### Fake Whatsapp I - START

IMPORTANTE

|  |
| --- |
| 1.- <https://facebook.github.io/react-native/docs/getting-started>  2.- <https://facebook.github.io/react-native/docs/running-on-device.html> |

1.- Creamos el proyecto y nos situamos en el directorio

|  |
| --- |
| react-native init fakewhatsapp  cd fakewhatsapp |

2.- Arrancamos el proyecto

|  |
| --- |
| Ventana1: react-native start o react-native start --reset-cache (muy efectiva)  Ventana2: react-native run-android |

3.- Desglose de componentes

|  |  |
| --- | --- |
| whatsapp-android-640x580.jpg | 1.- Rojo - Contenedor <View> padre  2.- Verde - ScrollableTabView  3.- Naranja - Componente header  4.- Azul - Componente nombre app  5.- Amarillo - Iconos |

4.- Abrir el fichero App.js y eliminar todos los componentes excepto el componente <View> padre. También eliminar todos los estilos excepto **container**. Flex: 1 nos indica que este componente padre abarca la pantalla completa.

5.- Asignamos los nombres de estilos a cada elemento

|  |
| --- |
| 1.- Rojo - mainContainer  2.- Verde - contentContainer  3.- Naranja - headerContainer  4.- Azul - leftHeaderContainer  5.- Amarillo - rightHeaderContainer |

6.- Añadimos estilos

|  |
| --- |
| const styles = StyleSheet.create({  mainContainer: {  flex: 1,  backgroundColor: '#F5FCFF'  },  headerContainer: {  flex: 1,  flexDirection: "row",  justifyContent: "space-between",  backgroundColor: "#075e54",  alignItems: "center"  },  contentContainer: {  flex: 8  },  rightHeaderContainer: {  flexDirection: "row",  alignItems: "flex-end"  },  leftHeaderContainer: {  flexDirection: "row",  alignItems: "flex-start"  },  logo: {  color: "#ffffff",  fontSize: 18,  marginLeft: 10,  fontWeight: "bold"  }  }); |

7.- Instalamos el paquete de iconos

|  |
| --- |
| npm install react-native-vector-icons --save  react-native link |

8.- Importamos iconos Material Design

|  |
| --- |
| import Icon from 'react-native-vector-icons/MaterialIcons'; |

9.- Añadimos los elementos

|  |
| --- |
| <View style={styles.mainContainer}>  <View style={styles.headerContainer}>  <View style={styles.leftHeaderContainer}>  <Text style={styles.logo}>WhatsApp</Text>  </View>  <View style={styles.rightHeaderContainer}>  <Icon name="search" color="#fff" size={23} style={{padding:5}} />  <Icon name="more-vert" color="#fff" size={23} style={{padding:5}} />  </View>  </View>  <View style={styles.contentContainer}>    </View>  </View> |

10.- Estado actual



### Fake Whatsapp II - TAB MENU

1.- Creamos una nueva carpeta en la raiz llamada ***app*** y dentro de ella una llamada ***components***

2.- Dentro de components creamos los ficheros **Chats.js, Calls.js y States.js**

3.- Fichero Chats.js (los demás igual)

|  |
| --- |
| import React, { Component } from 'react';  import {  View,  Text  } from 'react-native';  export default class Chats extends Component {  render() {  return(  <View><Text>Chats</Text></View>  )  }  } |

4.- Instalamos paquete de tabs

|  |
| --- |
| npm install react-native-scrollable-tab-view --save |

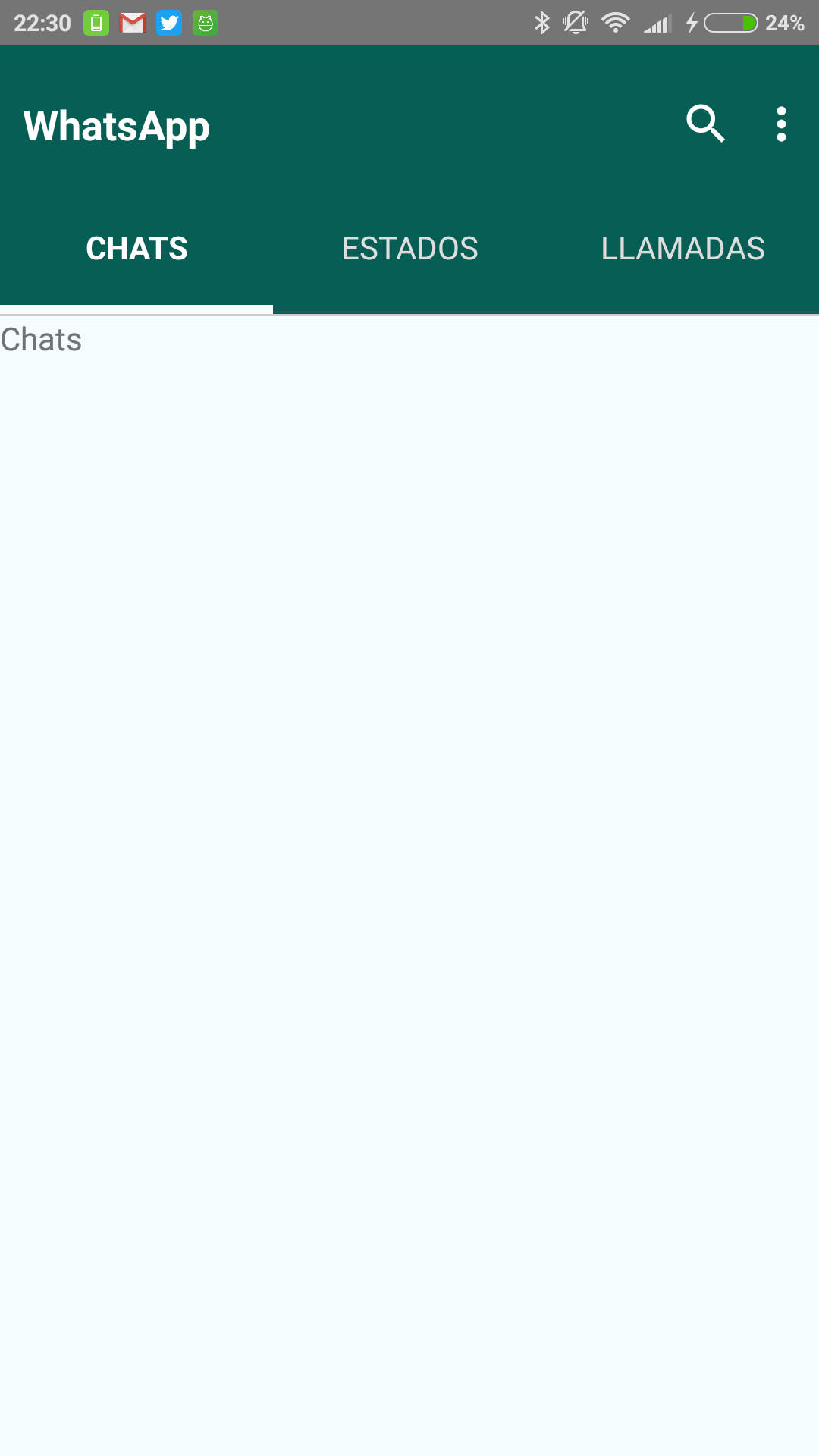
5.- En **App.js** importamos el paquete y los componentes que hemos creado

|  |
| --- |
| import ScrollableTabView from 'react-native-scrollable-tab-view'  import Chats from './app/components/Chats'  import States from './app/components/States'  import Calls from './app/components/Calls' |

6.- En **contentContainer** añadimos el componente **ScrollableTabView** y dentro los otros 3 componentes que serán diferentes vistas

|  |
| --- |
| <ScrollableTabView  tabBarUnderlineIcon="#fff"  tabBarUnderlineStyle={{backgroundColor: "#fff"}}  tabBarActiveTextColor="#fff"  tabBarInactiveTextColor="#ddd"  tabBarBackgroundColor="#075e54"  >  <Chats tabLabel="CHATS" />  <States tabLabel="ESTADOS" />  <Calls tabLabel="LLAMADAS" />  </ScrollableTabView> |

7.- Estado actual



En la siguiente fase recuperaremos datos para cada vista y renderizaremos el componente FlatList.

Los fake-data se encuentran en las siguientes ubicaciones:

Calls:

[fake\_calls.json](https://gist.githubusercontent.com/vocoder12/29f330a490720dc1404739a3cfe38ee2/raw/07cef43c84c309b08a91ef7cf2cda6618b062556/fake_calls.json)

Chats:

[fake\_chat.json](https://gist.githubusercontent.com/vocoder12/85508236a70a3261e441f2fc1607e25c/raw/6841728372cafe0408bbe318267112644a5662aa/fake_chats.json)

States:

[fake\_states.json](https://gist.githubusercontent.com/vocoder12/10530f68f2bc29ac1a29e76307a9c2aa/raw/ea68f8222be751bb3ea2ac8c9b171b2ece6586a6/fake_states.json)

### 

### 

### Fake Whatsapp III - REMOTE DATA

1.- Creamos una nueva carpeta llamada ***data*** dentro de ***app*** y dentro un nuevo fichero llamado **data.js**. En este fichero guardaremos todas las URL`s que atacaremos.

|  |
| --- |
| export const FAKE\_CALLS = 'https://gist.githubusercontent.com/vocoder12/29f330a490720dc1404739a3cfe38ee2/raw/07cef43c84c309b08a91ef7cf2cda6618b062556/fake\_calls.json';  export const FAKE\_CHATS = 'https://gist.githubusercontent.com/vocoder12/85508236a70a3261e441f2fc1607e25c/raw/6841728372cafe0408bbe318267112644a5662aa/fake\_chats.json';  export const FAKE\_STATES = 'https://gist.githubusercontent.com/vocoder12/10530f68f2bc29ac1a29e76307a9c2aa/raw/ea68f8222be751bb3ea2ac8c9b171b2ece6586a6/fake\_states.json'; |

2.- Instalamos paquete axios

|  |
| --- |
| npm install axios --save |

3.- En el fichero Chats.js importamos librería axios y la ruta

|  |
| --- |
| import axios from 'axios';  import { FAKE\_CHATS } from '../data/data'; |

4.- Chat.js está basado en clase, por lo tanto creamos el constructor e inicializamos el estado.

|  |
| --- |
| constructor(props) {  super(props)  this.state = {  chatList : [],  loaded: false  }  } |

5.- Método *componentDidMount*, esta es la fase del ciclo de vida de un componente para realizar las *peticiones http* y al realizarla con éxito actualizamos el estado.

|  |
| --- |
| componentDidMount() {  axios.get(FAKE\_CHATS)  .then((response) => {  this.setState({  chatList: response.data,  loaded: true  })  })  .catch(function (error) {  *console*.log(error);  });  } |

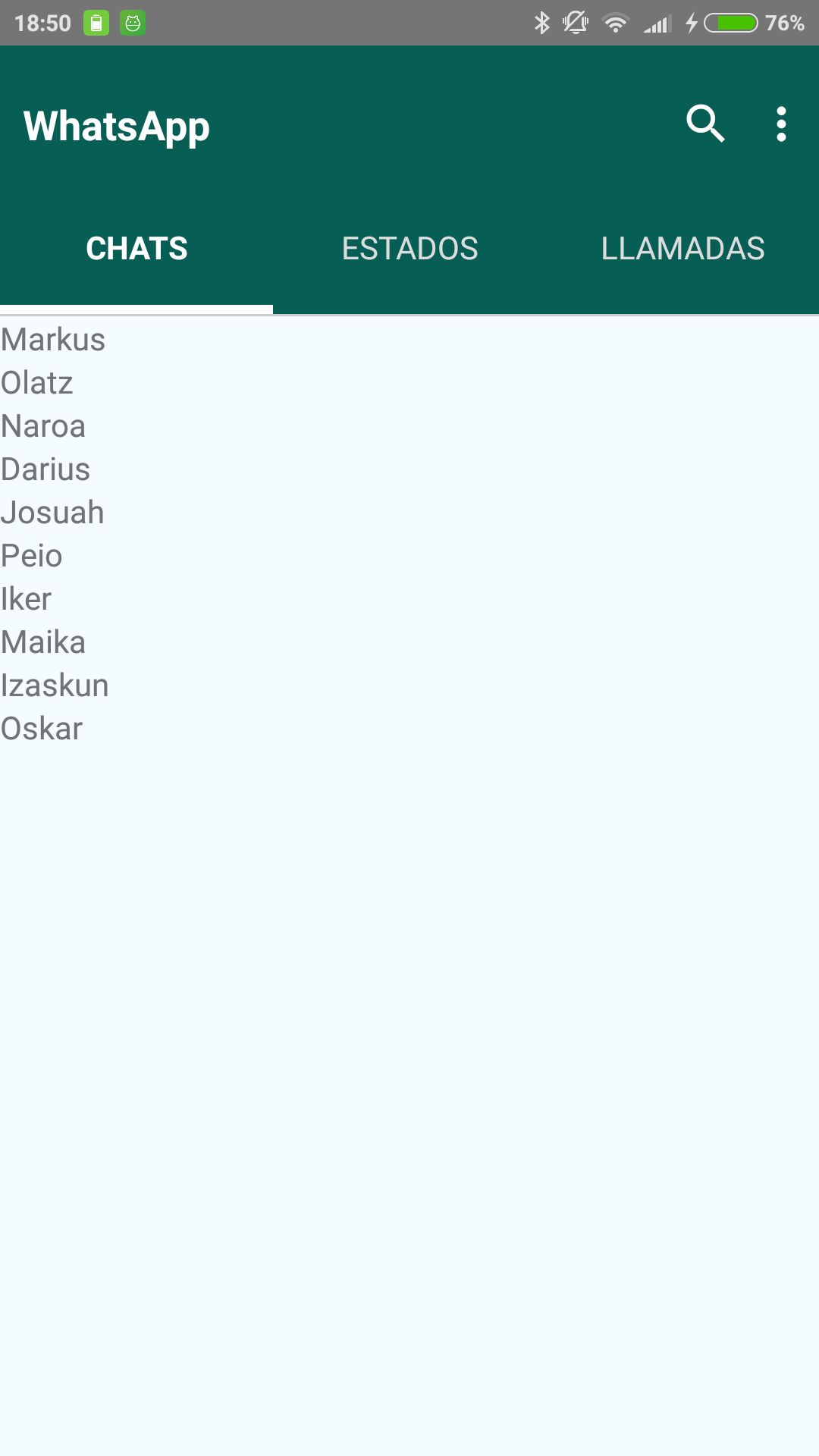
6.- Importamos el componente ActivityIndicator para representar un spinner de carga y el compnente FlatList para renderizado de listas

|  |
| --- |
| import { Text,FlatList,ActivityIndicator } from 'react-native'; |

7.- Renderizamos la lista. Si la propiedad *loaded* de nuestro estado es **true** quiere decir que hay datos, sino mostramos el **spinner**.

|  |
| --- |
| render() {  if(this.state.loaded) {  return(  <FlatList  data={this.state.chatList}  renderItem={({item}) => <Text>{item.first\_name}</Text>}  />  )  } else {  return(  <ActivityIndicator size="large" />  )  }  } |

8.- Estado actual de la lista de datos



### Fake Whatsapp IV - CHATS

1.- Creamos un nuevo componente llamado ListChat.js

2.- Desglose de componentes

|  |  |
| --- | --- |
| Screenshot_2017-09-10-19-19-23-919_com.jpg | 1.- Rojo - listItemContainer  2.- Azul - avatarContainer  3.- Violeta - chatDetailsContainer  4.- Amarillo1 - chatDetailsContainerWrap  5.- Amarillo2 - chatDetailsContainerWrap  6.- Verde1 - nameContainer  7.- Verde2 - dateContainer  8.- Naranja - msgContainer |

3.-Añadimos reglas de estilos

|  |
| --- |
| const styles = StyleSheet.create({  listItemContainer: {  flex: 1,  flexDirection: "row",  padding: 10  },  avatarContainer: {  flex: 1,  alignItems: "flex-start"  },  chatDetailsContainer: {  flex: 4,  borderBottomColor: "rgba(92,94,94,0.5)",  borderBottomWidth: 0.25  },  chatDetailsContainerWrap: {  flex: 1,  flexDirection: "row",  padding: 5  },  nameContainer: {  alignItems: "flex-start",  flex: 1  },  dateContainer: {  flexDirection: "row",  justifyContent : "space-between",  alignItems: "flex-end"  },  msgContainer: {  flex: 1  },  nameText: {  fontWeight: "bold",  color: "#000"  },  dateText: {  fontSize: 12  },  avatar: {  borderRadius: 30,  width: 60,  height: 60  }  }); |

4.- Creamos la estructura de elementos

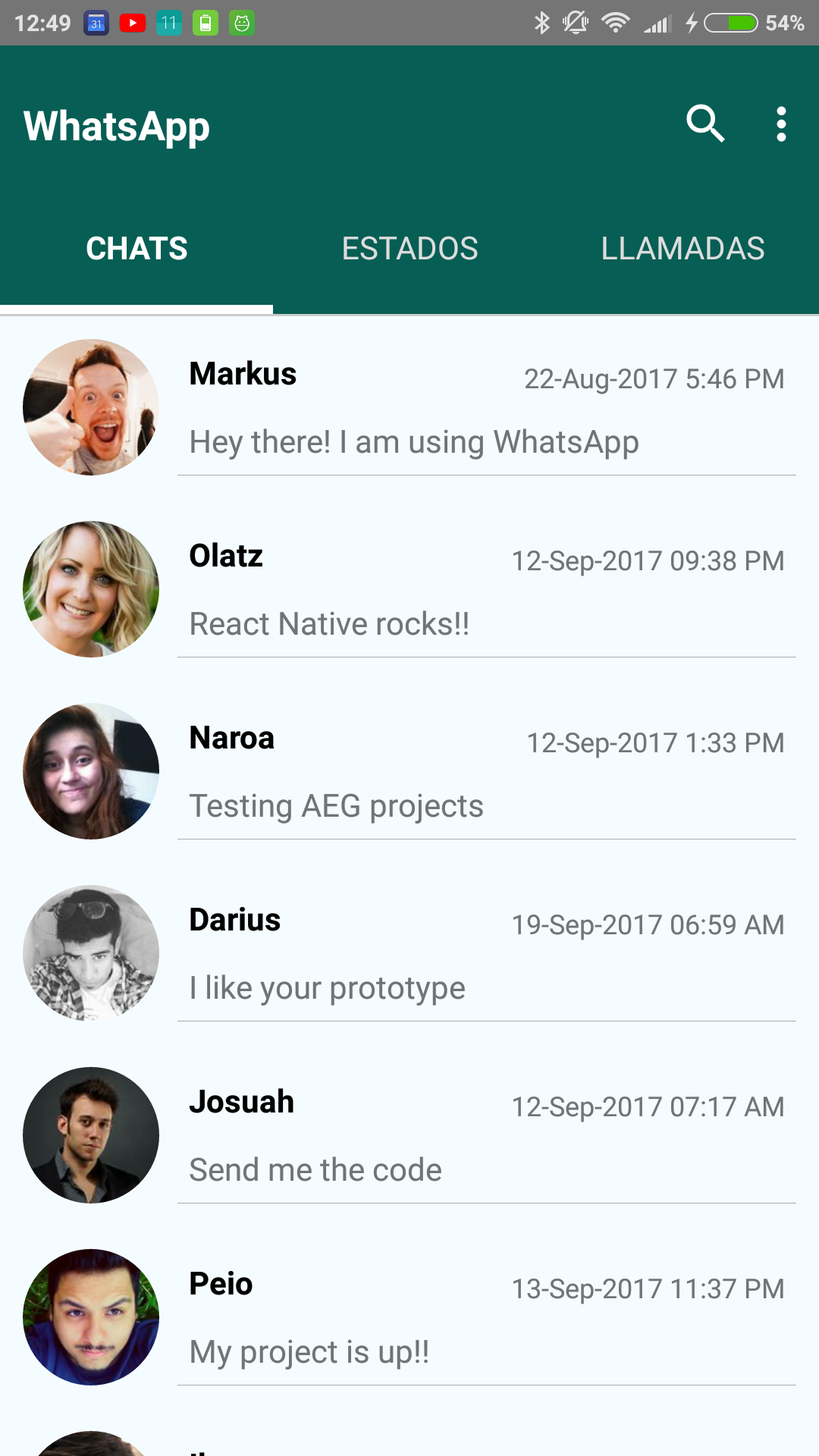
|  |
| --- |
| <View style = {styles.listItemContainer}>  <View style= {styles.avatarContainer}>  <Image  style={styles.avatar}  source={{uri: this.props.image}}  />  </View>  <View style = {styles.chatDetailsContainer}>  <View style={styles.chatDetailsContainerWrap}>  <View style={styles.nameContainer}>  <Text style={styles.nameText}>{this.props.first\_name}</Text>  </View>  <View style={styles.dateContainer}>  <Text style={styles.dateText}>{this.props.date} {this.props.time}</Text>  </View>  </View>  <View style={styles.chatDetailsContainerWrap}>  <View style={styles.msgContainer}>  <Text style={styles.msgText}>{this.props.message}</Text>  </View>  </View>  </View>  </View> |

5.- Importamos nuestro componente *ListChat* a *Chat.js* y lo implementamos

|  |
| --- |
| import ListChat from './ListChat'; |

|  |
| --- |
| render() {  if(this.state.loaded) {  return(  <FlatList  data={this.state.chatList}  renderItem={({item}) => (  <ListChat  first\_name={item.first\_name}  message={item.message}  image={item.image}  date={item.date}  time={item.time}  />  )}  />  )  } else {  return(  <ActivityIndicator size="large" />  )  }  } |

6.- Estado actual



### 

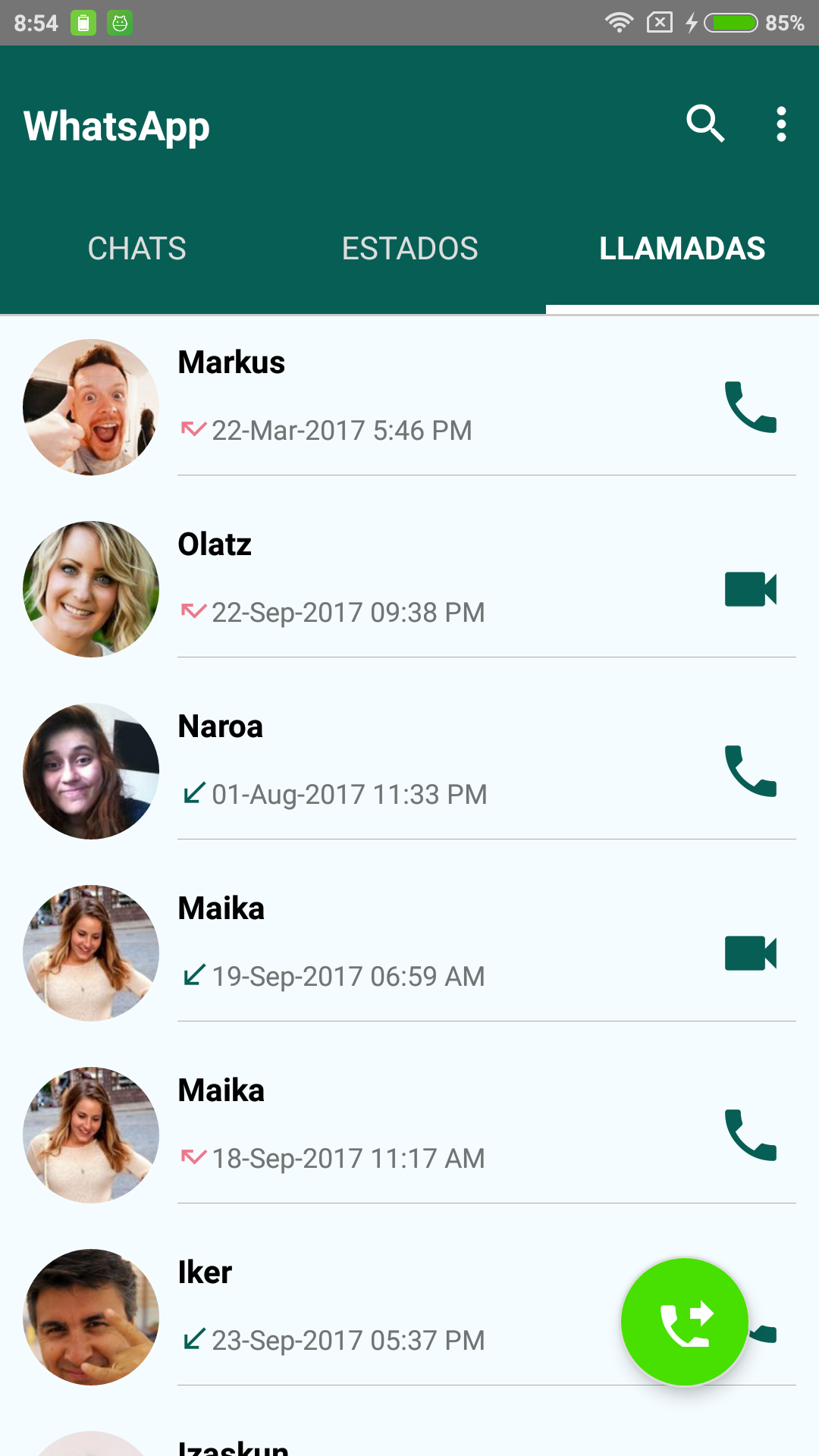
### 

### 

### 

### Fake Whatsapp VI - CALLS

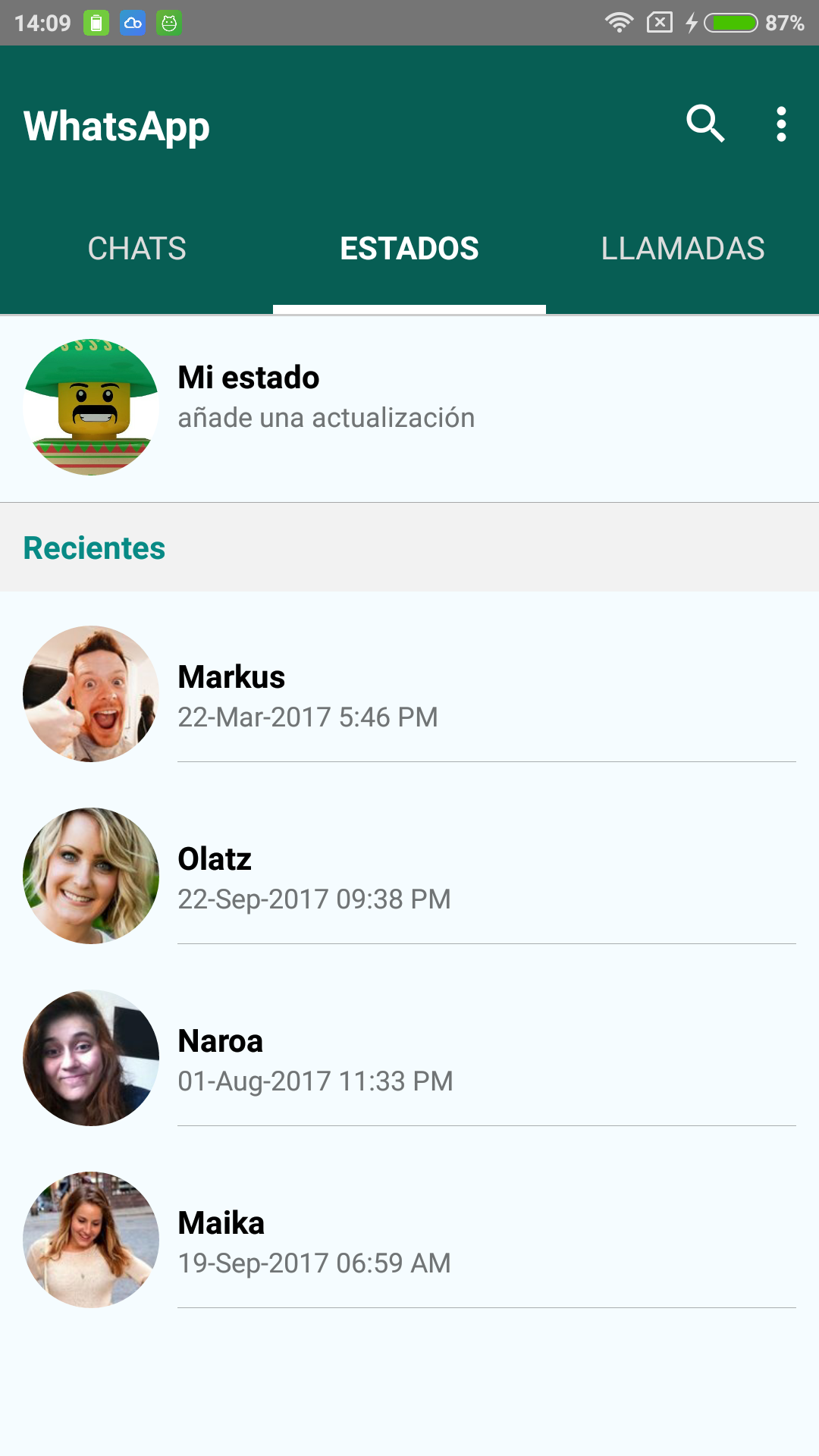
Tarea



### 

### Fake Whatsapp VII - STATUS

Tarea



### Fake Whatsapp VII - BOTONES FLOTANTES(FAB)

Tarea