

**Despliegue de Servicios Multimedia**

Máster Universitario en Ingeniería de Telecomunicación

**AppGaztaroa**

***Commit 02: “Componentes React Native”***

**Marko Galarza Galarza**

**Mikel Sagues García**

Componentes React Native

*Commit 02: “Componentes React Native”*

Una vez que ya hemos creado un primer esqueleto básico de una aplicación React Native, es hora de ir avanzando poco a poco en la construcción de la Aplicación Gaztaroa.

En este caso, haremos uso de la clase *Component* de React para generar nuestros primeros componentes en React Native. Por lo tanto, los conceptos que ya hemos asimilado en la primera parte de la asignatura siguen vigentes a la hora de trabajar con aplicaciones React Native.

Los diferentes elementos de nuestra aplicación se albergarán dentro de estos componentes, pero, a diferencia de lo que ocurre en React, dónde los elementos son elementos HTML, en el caso de React Native emplearemos elementos React Native dentro de dichos componentes.

A posteriori, cuando se compile el proyecto, esos elementos React Native serán convertidos a elementos específicos de cada plataforma (iOS o Android), permitiendo de ese modo su ejecución en los dispositivos móviles.

Además de eso, crearemos nuestros primeros elementos de una interfaz de usuario empleando las clases nativas de React Native y haremos uso de otros elementos de interfaz de usuario que importaremos de la librería *React Native Elements*.

De este modo, cuando finalice este ejercicio habremos comprendido el funcionamiento básico de la clase *Component* en aplicaciones React Native y habremos creado los primeros elementos de la interfaz de usuario de la aplicación.

1. Componentes React Native

En primer lugar, crearemos una carpeta de nombre “*componentes*” en el directorio principal de la aplicación. Es en esta carpeta donde guardaremos todos los ficheros JavaScript que albergarán los diferentes componentes de nuestra aplicación.

A continuación, descargamos los dos ficheros que se adjuntan a este documento y que emplearemos en este ejercicio:

* *imagenes.zip*
* *excursiones.js*

Copiamos “*imagenes.zip*” en la carpeta “c*omponentes*” que acabamos de crear y descomprimimos el fichero .zip en esa misma carpeta. De este modo, deberíamos tener una carpeta de nombre “*imagenes*” con varias imágenes dentro, dentro de la carpeta “componentes”. Una vez hecho esto, podemos borrar el fichero comprimido.

Creamos una carpeta de nombre “*comun*” en el directorio principal y copiamos dentro de dicha carpeta el fichero “*excursiones.js*”. En esta carpeta almacenaremos objetos JavaScript que serán utilizados por varios componentes. El fichero *excursiones.js* contiene información en forma de objetos JavaScript que será mostrada por nuestra aplicación, tal y como veremos en los siguientes pasos.

Para poder hacer uso de los componentes gráficos de las librerías *React Native Elements* y *react-native-safe-area-context,* primero debemos instalarlas en la carpeta principal de nuestra aplicación:

yarn add @rneui/themed @rneui/base

yarn add react-native-safe-area-context

A continuación, en la carpeta *componentes* crearemos un fichero de nombre *CampobaseComponent.js*, cuyo código fuente será el siguiente:

import React, { Component } from 'react';

import Calendario from './CalendarioComponent';

import { EXCURSIONES } from '../comun/excursiones';

class Campobase extends Component {

  constructor(props) {

    super(props);

    this.state = {

      excursiones: EXCURSIONES

    };

  }

  render() {

    return (

        <Calendario excursiones={this.state.excursiones} />

    );

  }

}

export default Campobase;

Observando detenidamente el código, se puede comprobar lo que ya hemos apuntado más arriba: un componente React Native es un caso particular de un componente React y muchos de los mecanismos que ya conocemos son de aplicación en este entorno (*constructor*, *state, props, etc.*).

Algunos aspectos que conviene destacar:

* Nótese la forma en que la función *return* llama al componente *Calendario* que crearemos a continuación y la forma en que le pasa un parámetro.
* Importante también comprender la forma en que dicho parámetro es adquirido desde el fichero JavaScript que hemos alojado en la carpeta *comun.*
* Fundamental recordar siempre que el componente que acabamos de crear debe ser exportado al final del código para hacerlo accesible a otros componentes de nuestra aplicación.

A continuación, en la carpeta *componentes* creamos un nuevo componente de nombre *CalendarioComponent.js* con el siguiente código fuente:

import React from 'react';

import { ListItem, Avatar } from '@rneui/themed';

import { SafeAreaView, FlatList } from 'react-native';

function Calendario(props) {

    const renderCalendarioItem = ({item, index}) => {

        return (

            <ListItem key={index} bottomDivider>

                <Avatar source={require('./imagenes/40Años.png')} />

                <ListItem.Content>

                    <ListItem.Title>{item.nombre}</ListItem.Title>

                    <ListItem.Subtitle>{item.descripcion}</ListItem.Subtitle>

                </ListItem.Content>

            </ListItem>

        );

    };

    return (

        <SafeAreaView>

            <FlatList

                data={props.excursiones}

                renderItem={renderCalendarioItem}

                keyExtractor={item => item.id.toString()}

            />

        </SafeAreaView>

    );

}

export default Calendario;

Aspectos a destacar del código anterior y que conviene analizar en detalle a partir de la bibliografía que se señala al final de este ejercicio:

* El componente que acabamos de crear es un componente funcional (*functional component*), por lo que no tiene estado.
* El componente recibe parámetros (*props*) tal y como hemos visto más arriba.
* Hacemos uso de dos elementos para mostrar la lista de excursiones. Consultar la bibliografía asociada a cada uno de ellos, para comprender su funcionamiento y explorar sus capacidades:
  + FlatList (de React Native)
  + ListItem (de React Native Elements)
* Comprender la forma en que el *ListItem* adquiere la imagen a mostrar.
* Reparar en la forma en que el componente funcional hace uso de la función JavaScript *renderCalendarioItem*.

Por último, sustituimos el código fuente del fichero *app.js* de este modo:

import { StatusBar } from 'expo-status-bar';

import React from 'react';

import { View } from 'react-native';

import Campobase from './componentes/CampobaseComponent';

import { SafeAreaProvider } from 'react-native-safe-area-context';

export default function App() {

  return(

  <SafeAreaProvider>

    <View>

      <Campobase/>

      <StatusBar style="auto" />

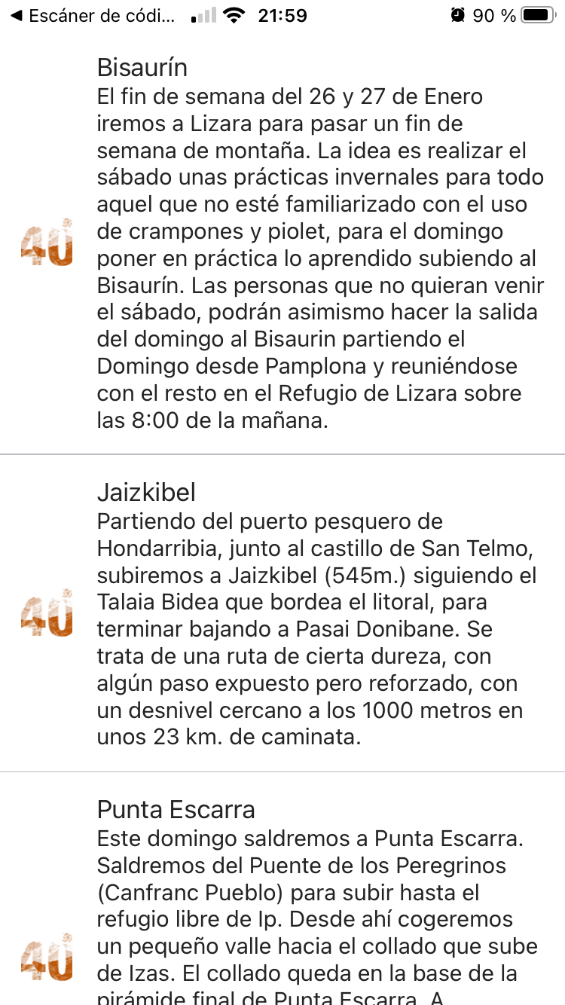
    </View>

  </SafeAreaProvider>

  );

}

Una vez completado lo anterior, la página principal de nuestra aplicación debería mostrar lo siguiente:



Si todo es correcto, ya podemos hacer el *commit* correspondiente a este ejercicio.

1. Bibliografía

* **React Native Elements**  
  https://reactnativeelements.com/docs
* **React Native Safe Area Context**

https://github.com/th3rdwave/react-native-safe-area-context

* **Flatlist** (React Native)  
  https://reactnative.dev/docs/flatlist
* **ListItem** (React Native Elements)  
  https://reactnativeelements.com/docs/components/listitem