upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

DESPLIEGUE DE SERVICIOS MULTIMEDIA
Máster Universitario en Ingeniería de Telecomunicación

AppGaztaroa

Commit 07: "Botones o Iconos"

Marko Galarza Galarza
Mikel Sagues García

Botones o Iconos

Commit 07: "Botones o Iconos"

En este capítulo vamos a introducir en nuestra aplicación uno de los mecanismos más habituales de interactuar con el usuario: los botones. Sin embargo, vamos a hacerlo sin hacer uso del componente *Button* de React Native, ni de ninguna otra librería, ya que nos valdremos de Iconos para implementar dicha funcionalidad.

Además, completaremos la vista de detalle de las excursiones, renderizando una nueva *Card* en la que se mostrarán los comentarios de los participantes de la excursión.

1. Añadir comentarios a la vista del componente *DetalleExcursion*

En primer lugar, crearemos un nuevo *Card* para renderizar los comentarios asociados al detalle de cada una de las excursiones. Para ello:

- Crearemos una nueva clase funcional de nombre *renderComentario(props)* dentro del fichero *DetalleExcursionComponent.js*, de manera análoga a lo que hicimos con la clase funcional *renderExcursion(props)*.
- La información a mostrar será obtenida del fichero JavaScript *comentarios.js* que se adjunta a este documento y que copiaremos en la carpeta *comun* dentro de nuestra aplicación.
- En definitiva, debemos modificar el código fuente del fichero *DetalleExcursionComponent.js* tal y como se muestra a continuación:

```
[...]
import { Text, View, ScrollView, FlatList} from 'react-native';
import { COMENTARIOS } from '../comun/comentarios';

[...]

function RenderComentario(props) {

    const comentarios = props.comentarios;

    return (
      <Card>
        <Card.Title>Comentarios</Card.Title>
        <Card.Divider/>
        [... renderizar aquí el contenido de los comentarios ...]
      </Card>
    );
}
```

```
[...]

        this.state = {
        [... actualizar estado para tener acceso a los comentarios ...]
        };

[...]

  render(){
    const {excursionId} = this.props.route.params;
    return(
        <ScrollView>
            <RenderExcursion
                excursion={this.state.excursiones[+excursionId]}
            />
            <RenderComentario
                comentarios={this.state.comentarios.filter((comentario) =
> comentario.excursionId === excursionId)}
            />
        </ScrollView>
    );
  }

[...]
```

2. Botones o Iconos

A continuación, añadiremos nuestros primeros botones a la aplicación, empleando el componente *Icon* de la librería *React-Native-Elements*.

Nuestra “excusa” para crear este primer botón será el permitir al usuario que marque como “favorita” una de las excursiones del Club Deportivo Gaztaroa. Este hecho, además de modificar la apariencia del botón para mostrar que ya ha sido clicado (y que, en consecuencia, la excursión en la que estamos la consideramos como “favorita”), también debe implementar una sencilla lógica que permita a la aplicación saber que el botón ya ha sido clicado.

Para conseguir esta funcionalidad, el código del fichero *DetalleExcursionComponent.js* debe actualizarse siguiendo la siguiente guía, a partir de lo que ya hemos construido en el primer apartado de este capítulo:

```
[...]
import { Card, Icon } from '@rneui/themed';
[...]
                <Icon
                    raised
                    reverse
```

```
                    name={ props.favorita ? 'heart' : 'heart-o'}
                    type='font-awesome'
                    color='#f50'
                    onPress={() => props.favorita ? console.log('La excur
sión ya se encuentra entre las favoritas') : props.onPress()}
                />

[...]
        this.state = {
            [...]
            favoritos: []
        };

[...]

    marcarFavorito(excursionId) {
        this.setState({favoritos: this.state.favoritos.concat(excursionId
)});
    }

[...]

                <RenderExcursion
                    excursion={this.state.excursiones[+excursionId]}
                    favorita={this.state.favoritos.some(el => el === excu
rsionId)}
                    onPress={() => this.marcarFavorito(excursionId)}
                />
```

Cuestiones a comentar:

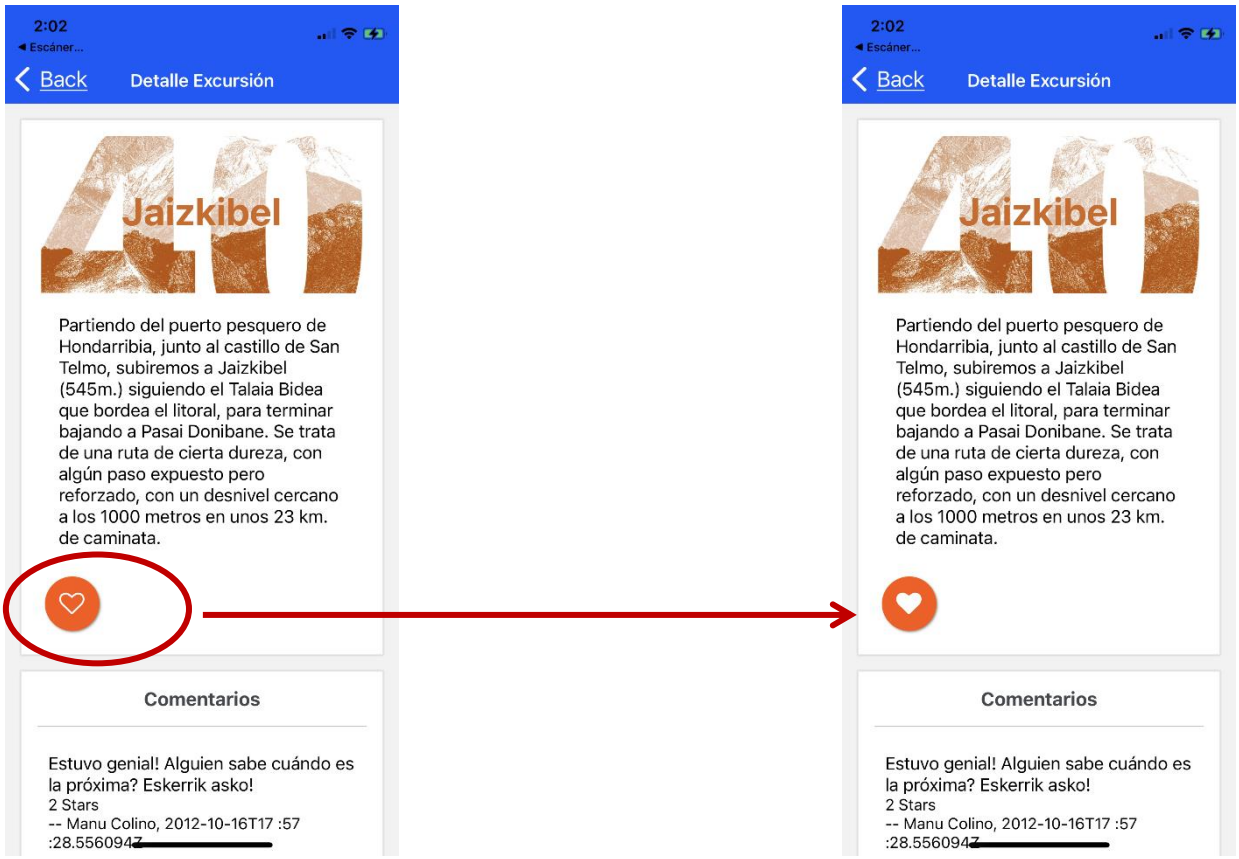
- Icon
  - Consultar la documentación para ver el efecto de cada uno de los parámetros que se han empleado, además de explorar otras funcionalidades.
- Icon / Font-Awesome
  - Hacemos uso de la librería Font-Awesome, que se encuentra integrada en los iconos de React Native Elements. En la documentación de *Icon* en React Native Elements podemos encontrar información detallada en relación a este tema, incluyendo otras librerías alternativas a la que hemos utilizado.
- Del mismo modo que hacemos en cualquier aplicación JavaScript que se ejecute en el Navegador, en React Native también podemos mostrar información en la consola (no será visible en la aplicación), empleando el comando *console.log()*:

  `console.log('mensaje a mostrar')`
- Nótese que la aplicación sólo es capaz de recordar si una excursión ha sido marcada como favorita dentro del componente *DetalleExcursion*. El array *favoritos[]* se reseteará cada vez que accedamos al componente *DetalleExcursion*, por lo que esta información se perderá en caso de volver atrás

o salir de esta pantalla. En futuros ejercicios veremos la forma en que esta información puede almacenarse de forma permanente.

- Como siempre, es importante tener claro el código JavaScript que utilizamos en nuestra aplicación. En particular, en este ejercicio, el uso de *filter* y de *some.*

Una vez completados los dos primeros apartados de este capítulo, la vista de detalle de las excursiones quedaría como se muestra a continuación:



3. Customizar el Drawer Menu

El objetivo de esta tercera sección es aprovechar lo que hemos aprendido hasta ahora, para customizar también el *Drawer Menu,* haciéndolo un poquito más atractivo.

Para ello, crearemos:

- Iconos a la izquierda de cada una de las opciones del menú.
- Una zona en la parte superior del menú, para mostrar el logotipo de Gaztaroa, además del nombre del club de montaña.

El resultado que estamos buscando es el siguiente:

A continuación, se muestran algunas de las modificaciones en el código fuente del fichero *CampobaseComponent.js* que nos permiten alcanzar dicho objetivo:

```
[...]
import { View, Platform, StyleSheet, Image, Text } from 'react-native';
import { createDrawerNavigator,   DrawerContentScrollView, DrawerItemList
 } from '@react-navigation/drawer';
import { Icon } from '@rneui/themed';
import { SafeAreaView } from 'react-native-safe-area-context';
[...]

function CustomDrawerContent(props) {
  return (
    <DrawerContentScrollView {...props}>
      <SafeAreaView style={styles.container} forceInset={{ top: 'always',
 horizontal: 'never' }}>
        <View style={styles.drawerHeader}>
          <View style={{flex:1}}>
```

```
          <Image source={require('./imagenes/logo.png')} style={styles.dr
awerImage} />
          </View>
          <View style={{flex: 2}}>
            <Text style={styles.drawerHeaderText}> Gaztaroa</Text>
          </View>
        </View>
        <DrawerItemList {...props} />
      </SafeAreaView>
    </DrawerContentScrollView>
  );
}

[...]

function DrawerNavegador() {
  return (
      <Drawer.Navigator
      initialRouteName="Home"
      drawerContent={props => <CustomDrawerContent {...props} />}
      screenOptions={{
        headerShown: false,
        drawerStyle: {
          backgroundColor: '#c2d3da',
        },
      }}
      >
        <Drawer.Screen name="Campo base" component={HomeNavegador}
          options={{
            drawerIcon: ({ tintColor}) => (
              <Icon
              name='home'
              type='font-awesome'
              size={24}
              color={tintColor}
              />
            )
            }}
        />

[...]

const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
  drawerHeader: {
    backgroundColor: '#015afc',
```

```
    height: 100,
    alignItems: 'center',
    justifyContent: 'center',
    flex: 1,
    flexDirection: 'row'
  },
  drawerHeaderText: {
    color: 'white',
    fontSize: 24,
    fontWeight: 'bold'
  },
  drawerImage: {
    margin: 10,
    width: 80,
    height: 60
  }
});

export default Campobase;
```

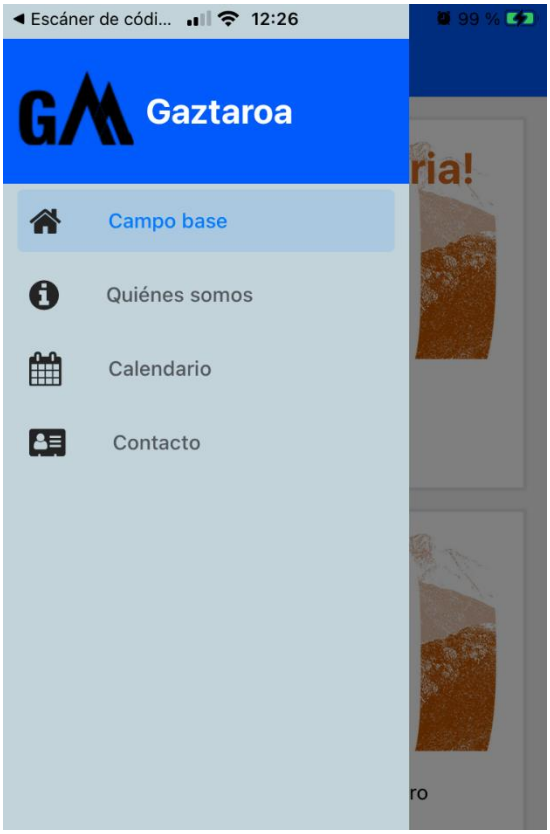
Importante, como siempre, analizar en detalle dicho código, con especial atención a las siguientes claves:

- SafeAreaView (contenedor)
- La opción *drawerContent* de *Drawer.Navigator*.
- Icon / Drawer Icon
    - El código mostrado es el asociado a la opción "Contacto" del *Drawer Menu*. Para el resto de opciones se han elegido los siguientes iconos de Font-Awesome:
        - *'info-circle'* (para la opción "Quiénes somos")
        - *'calendar'* (para la opción "Calendario")
        - *'address-card'* (para la opción "Campo Base")
- En este punto hemos hecho un uso relativamente avanzado de los estilos en-línea que son la base de la customización de los componentes en React Native. Este es, por lo tanto, un buen momento para profundizar en el conocimiento de los mismos. Además, se ha customizado el *layout* de los diferentes elementos dentro de dicho menú, empleando la propiedad *Flex*. Se recomienda consultar las referencias que se muestran al final del documento para profundizar en ambos puntos.

Adicionalmente, deseamos añadir un botón en la parte superior izquierda de cada una de las pantallas, cuya función será desplegar el *Drawer Menu*. Para ello, haremos uso de un icono, en combinación con la propiedad *headerLeft* y el método *DrawerActions.toggleDrawer()*, tal y como se muestra a continuación, para el caso del navegador de la página "Quienes somos". La excepción será la pantalla *DetalleExcursion* en la que mantendremos el botón para volver a la pantalla *Calendario*:

```
import { NavigationContainer, DrawerActions } from '@react-
navigation/native';

[...]
function QuienesSomosNavegador({navigation}) {
  return (
    <Stack.Navigator
[...]
        headerLeft: () => (<Icon name="menu" size={28} color= 'white' onP
ress={ () => navigation.dispatch(DrawerActions.toggleDrawer()) }/>),
      }}
    >
```



Llegados a este punto, solo tenemos que comprobar que todo es correcto, hacer el *commit* correspondiente a este capítulo y subir a nuestro tablero de Trello un breve documento con las claves del trabajo realizado.

4. Bibliografía

- **Icon** (React Native Elements)
  https://reactnativeelements.com/docs/icon/
- **Font-Awesome**
  https://fontawesome.com/icons
- **Button** (React Native Elements); no se emplean en este ejercicio.
  https://reactnativeelements.com/docs/button/
- **Console Log**
  https://docs.expo.io/workflow/logging/
  https://docs.expo.io/workflow/debugging/
  https://medium.com/nmc-techblog/advanced-console-log-tips-tricks-fa3762930bca
- **Safe Areas in React Native Navigation**
  https://reactnavigation.org/docs/handling-safe-area/
- **React Native Safe Area Context**
  https://github.com/th3rdwave/react-native-safe-area-context
- **React Native Navigation Custom Drawer Content**
  https://reactnavigation.org/docs/drawer-navigator/#drawercontent
- **Toogle Drawer**
  https://reactnavigation.org/docs/drawer-actions/#toggledrawer
- **Estilos en React Native** (inline styles)
  https://reactnative.dev/docs/style
- **Anchura y altura de componentes en React Native**
  https://reactnative.dev/docs/height-and-width
- **React Native Layout con Flexbox**
  https://reactnative.dev/docs/flexbox
  https://medium.com/wix-engineering/the-full-react-native-layout-cheat-sheet-a4147802405c
  https://yogalayout.com/playground/
  https://reactnative.dev/docs/layout-props
- **Some (JavaScript)**
  https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/some
  https://www.w3schools.com/jsref/jsref_some.asp
- **Filter (JavaScript)**
  https://www.w3schools.com/jsref/jsref_filter.asp
  https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array/filter
- **Configuración de la barra de estado (*status bar*)**
  https://docs.expo.io/guides/configuring-statusbar