PRO C89 V2 **LEER HISTORIAS USANDO FIREBASE (PARTE 2)**

Ahora para la parte de de la pantalla de índice, queremos asegurarnos que obtenemos las historias de la base de datos de firebase.

Para eso, regresemos a nuestra pantalla de índice (**Feed.js)** y creemos una nueva función **fetchStories()** y llamémosla en **componentDidMount()**.

Primero agregaremos un **nuevo estado** para nuestras historias, que será una **matriz vacía**. Nuestro constructor sería:

export default class Feed extends Component {

  °

      fontsLoaded: false,

      light\_theme: true,

      stories: []

    };

  }

°

  componentDidMount() {

    this.\_loadFontsAsync();

    this.fetchUser();

    this.fetchStories();

  }

  fetchStories = () => {

    firebase

      .ref("/posts/")

      .on(

        "value",

        snapshot => {

          let stories = [];

          if (snapshot.val()) {

            Object.keys(snapshot.val()).forEach(function (key) {

              stories.push({

                key: key,

                value: snapshot.val()[key]

              });

            });

          }

          this.setState({ stories: stories });

          this.props.setUpdateToFalse();

        },

        function (errorObject) {

          console.log("La lectura falló: " + errorObject.code);

        }

      );

  }

En esta función, estamos revisando todos los valores en la referencia de los objetos de **publicaciones** y lo que sea que obtengamos, estamos iterando sobre todos los valores clave usando la función **map()** y almacenarlos dentro de otra matriz llamada **stories**.

Entonces estamos actualizando la propiedad **this.state** para cargar la información.

Ahora, necesitamos usar esta nueva historia desde el estado, pero puede haber un caso que no haya historias en la base de datos.

Ahora lidiemos con ambos en nuestra función **render()**:

**Continuamos en el archivo de Feed.js**

    render() {

      if (!this.state.fontsLoaded) {

      return <AppLoading />;

    } else {

        return (

            <View style={this.state.light\_theme ? styles.containerLight : styles.container}>

          <SafeAreaView style={styles.droidSafeArea} />

          <View style={styles.appTitle}>

            <View style={styles.appIcon}>

              <Image

                source={require("../assets/logo.png")}

                style={styles.iconImage}>

              </Image>

            </View>

            <View style={styles.appTitleTextContainer}>

              <Text style={this.state.light\_theme ? styles.appTitleTextLight : styles.appTitleText}> App narrar Historias </Text>

            </View>

          </View>

          {!this.state.stories[0] ? (

            <View style={styles.noStories}>

              <Text

                style={

                  this.state.light\_theme

                    ? styles.noStoriesTextLight

                    : styles.noStoriesText

                }

              >

                No hay historias disponibles

              </Text>

            </View>

          ) : (

            <View style={styles.cardContainer}>

              <FlatList

                keyExtractor={this.keyExtractor}

                data={this.state.stories}

                renderItem={this.renderItem}

              />

            </View>

          )}

          <View style={{ flex: 0.08 }} />

        </View>

        )

      }

    }

}

Aquí, cerca de la función **FlatList()**, hemos agregado una condición dentro de un par de corchetes {}.

***En React Native, podemos agregar el código JavaScript dentro del enunciado de vuelta de una función render() usando corchetes.***

Estamos revisando si el primer objeto/información/historia desde la base de datos está presente en las historias o no.

Este código no hubiera funcionado ya que desde un inicio tuvimos una matriz vacía en nuestro estado de forma predeterminada, pero ya que estamos tratando de obtener historias, si no hay historias disponibles, mostrará un valor **nulo** en nuestra matriz.

Por lo tanto, estamos revisando si el primer valor es válido o no. Si es un objeto, es un valor válido, sino, será **nulo**, el cual no es un valor válido.

Si no es un valor válido, estamos mostrando un texto que dice que **“no hay historias disponibles”**. De lo contrario, estamos usando el componente **<FlatList>**.

Dentro de Flatlist, hemos cambiado el valor de los atributos de nuestra **información** al estado.

Necesitamos también agregar estilo subsecuente para el nuevo texto creado que muestra que no hay historias:

**Nota**: vamos a copiar la sección azul, ya que hay estilos que se quitaron o se modificaron, adicional vamos a poner los nuevos estilos del código (color morado)

**Código de estilos de Feed.js**

const styles = StyleSheet.create({

   °

  droidSafeArea: {

    marginTop: Platform.OS === "android" ? StatusBar.currentHeight : RFValue(35)

  },

    appTitle: {

    flex: 0.07,

    flexDirection: "row"

  },

  appIcon: {

    flex: 0.3,

    justifyContent: "center",

    alignItems: "center"

  },

  iconImage: {

    width: "100%",

    height: "100%",

    resizeMode: "contain"

  },

  appTitleTextContainer: {

    flex: 0.7,

    justifyContent: "center"

  },

  appTitleText: {

    color: "white",

    fontSize: RFValue(28),

    fontFamily: "Bubblegum-Sans"

  },

  appTitleTextLight: {

    color: "black",

    fontSize: RFValue(28),

    fontFamily: "Bubblegum-Sans"

  },

  cardContainer: {

    flex: 0.85

  },

  noStories: {

    flex: 0.85,

    justifyContent: "center",

    alignItems: "center"

  },

  noStoriesTextLight: {

    fontSize: RFValue(40),

    fontFamily: "Bubblegum-Sans"

  },

  noStoriesText: {

    color: "white",

    fontSize: RFValue(40),

    fontFamily: "Bubblegum-Sans"

  }

});

¡Genial! Hemos terminado, sin embargo, aún necesitamos hacer cambios a nuestra **StoryCard**. Eso es porque nuestra **historia** que estamos pasando no está en la misma estructura que antes.

Esta vez, nuestra historia es un objeto que tiene una **clave** como la ID única de la historia y **valor** como la información de la historia. Hagamos los cambios a nuestra **StoryCard**, para incluir los cambios de la **clave** y el **valor**.

Primero cambiemos el constructor en nuestra **StoryCard.js** para almacenar las claves y valores de forma separada:

import \* as Font from "expo-font";

import db from "../config";

import firebase from "firebase";

let customFonts = {

  "Bubblegum-Sans": require("../assets/fonts/BubblegumSans-Regular.ttf")

};

export default class StoryCard extends Component {

  constructor(props) {

    super(props);

    this.state = {

      fontsLoaded: false,

      light\_theme: true,

      story\_id: this.props.story.key,

      story\_data: this.props.story.value,

      is\_liked: false,

      likes: this.props.story.value.likes

    };

  }

°

  likeAction = () => {

    if (this.state.is\_liked) {

      db

        .ref("posts")

        .child(this.state.story\_id)

        .child("likes")

        .set(firebase.database.ServerValue.increment(-1));

      this.setState({ likes: (this.state.likes - 1), is\_liked: false });

    } else {

      db

        .ref("posts")

        .child(this.state.story\_id)

        .child("likes")

        .set(firebase.database.ServerValue.increment(1));

      this.setState({ likes: (this.state.likes + 1), is\_liked: true });

    }

  };

fetchUser = () => {

    let theme;

    db

      .ref("/users/HIxeXreDeIWEDwzd20lFWh64Mtx1/")

°

      });

  };

Después, dentro de la función **render()**, creemos una variable llamada **story** que es igual a nuestro estado **story\_data**.

Recuerda que cuando guardemos nuestra historia, la salvemos de tal forma que su valor sea **image\_1, image\_2 … image\_5**.

Por lo tanto, necesitamos crear un objeto aquí que haga un mapeo del valor de estas claves con la ruta de su imagen respectivamente como se muestra.

**Continuamos en StoryCard.js**

  render() {

    let story = this.state.story\_data;

    if (!this.state.fontsLoaded) {

      return <AppLoading />;

    } else {

            let images = {

        image\_1: require("../assets/story\_image\_1.png"),

        image\_2: require("../assets/story\_image\_2.png"),

        image\_3: require("../assets/story\_image\_3.png"),

        image\_4: require("../assets/story\_image\_4.png"),

        image\_5: require("../assets/story\_image\_5.png")

      };

      return (

        <TouchableOpacity

Ahora cambia la fuente del componente **<Image>** que muestra la imagen de la historia también.

Y los estilos para él seguirán siendo los mismos.

Ahora, estábamos usando **this.props.story** para obtener la historia hasta ahora en esta función, pero esta vez, solo usaremos la variable **story** ya que tenemos la información de nuestra historia almacenada en ella.

  render() {

    let story = this.state.story\_data;

°

      };

      return (

        <TouchableOpacity

          style={styles.container}

          onPress={() =>

            this.props.navigation.navigate("StoryScreen", {

              story: story,

              story\_id:this.state.story\_id })

          } >

          <SafeAreaView style={styles.droidSafeArea} />

          <View

            style={this.state.light\_theme ? styles.cardContainerLight : styles.cardContainer } >

            <Image

              source={images[story.preview\_image]}

              style={styles.storyImage}

            ></Image>

            <View style={styles.titleContainer}>

              <View style={styles.titleTextContainer}>

                <Text

                  style={ this.state.light\_theme ? styles.storyTitleTextLight : styles.storyTitleText } >

                  {story.title}

                </Text>

                <Text

                  style={ this.state.light\_theme ? styles.storyAuthorTextLight : styles.storyAuthorText } >

                  {story.author}

                </Text>

                <Text

                  style={this.state.light\_theme ? styles.descriptionTextLight : styles.descriptionText } >

                  {story.description}

                </Text>

              </View>

            </View>

            <View style={styles.actionContainer}>

              <TouchableOpacity  style={ this.state.is\_liked  ? styles.likeButtonLiked : styles.likeButtonDisliked }

                onPress={() => this.likeAction()}  >

                <Ionicons name={"heart"} size={RFValue(30)}

                  color={this.state.light\_theme ? "black" : "white"}  />

                <Text style={ this.state.light\_theme ? styles.likeTextLight : styles.likeText } >

                  {this.state.likes}

                </Text>

              </TouchableOpacity>

            </View>

          </View>

        </TouchableOpacity>

      );

    }

  }

}

const styles = StyleSheet.create({

  °

  actionContainer: {

    °

  },

  likeButtonLiked: {

    width: RFValue(160),

    height: RFValue(40),

    justifyContent: "center",

    alignItems: "center",

    flexDirection: "row",

    backgroundColor: "#eb3948",

    borderRadius: RFValue(30)

  },

  likeButtonDisliked: {

    width: RFValue(160),

    height: RFValue(40),

    justifyContent: "center",

    alignItems: "center",

    flexDirection: "row",

    borderColor: "#eb3948",

    borderWidth: 2,

    borderRadius: RFValue(30)

  },

  likeText: {

    color: "white",

    fontFamily: "Bubblegum-Sans",

    fontSize: 25,

    marginLeft: 25,

    marginTop: 6

  },

  likeTextLight: {

    fontFamily: "Bubblegum-Sans",

    fontSize: 25,

    marginLeft: 25,

    marginTop: 6

  }

});

Ahora vamos a poner en el archivo de **StoryScreen.js** el cambio de **color del tema**. También habilitar el botón de los **likes** y también la imagen de la historia seleccionada, ya que tenemos por default la **imagen 1.**

import \* as Speech from "expo-speech";

import AppLoading from "expo-app-loading";

import \* as Font from "expo-font";

import db from '../config';

import firebase from 'firebase';

let customFonts = {

  "Bubblegum-Sans": require("../assets/fonts/BubblegumSans-Regular.ttf")

};

export default class StoryScreen extends Component {

  constructor(props) {

    super(props);

    this.state = {

      fontsLoaded: false,

      speakerColor: "gray",

      speakerIcon: "volume-high-outline",

      light\_theme: true,

      likes: this.props.route.params.story.likes,

      is\_liked: false,

    };

  }

Vamos a montar en el componente la función **fetchUser** para cargar los cambios del tema, y crearemos la función para las acciones de los **likes** de nuestras historias, en la cual lee la info de este almacenada en la BD.

  componentDidMount() {

    this.\_loadFontsAsync();

    this.fetchUser();

  }

  fetchUser = () => {

    let theme;

    db

      .ref('/users/HIxeXreDeIWEDwzd20lFWh64Mtx1') *//Checar usuario de nuestra BD*

      .on('value', (snapshot) => {

        theme = snapshot.val().current\_theme;

        this.setState({ light\_theme: theme === 'light' });

      });

  };

  async initiateTTS(title, author, story, moral) {

    °

      Speech.stop();

    }

  }

likeAction = () => {

    if (this.state.is\_liked) {

      db

        .ref('posts')

        .child(this.props.route.params.story\_id)

        .child('likes')

        .set(firebase.database.ServerValue.increment(-1));

      this.setState({ likes: this.state.likes - 1, is\_liked: false });

    } else {

      db

        .ref('posts')

        .child(this.props.route.params.story\_id)

        .child('likes')

        .set(firebase.database.ServerValue.increment(1));

      this.setState({ likes: this.state.likes + 1, is\_liked: true });

    }

  };

  render() {

    if (!this.props.route.params) {

Después vamos a cargar los datos en sus etiquetas correspondientes, además a habilitar los estilos para cada una de estas.

  render() {

    if (!this.props.route.params) {

      this.props.navigation.navigate("Home");

    } else if (!this.state.fontsLoaded) {

      return <AppLoading />;

    } else {

            let images = {

        image\_1: require("../assets/story\_image\_1.png"),

        image\_2: require("../assets/story\_image\_2.png"),

        image\_3: require("../assets/story\_image\_3.png"),

        image\_4: require("../assets/story\_image\_4.png"),

        image\_5: require("../assets/story\_image\_5.png")

      }

      return (

       <View style={ this.state.light\_theme ? styles.containerLight : styles.container }>

          <SafeAreaView style={styles.droidSafeArea} />

          <View style={styles.appTitle}>

            <View style={styles.appIcon}>

              <Image

                source={require('../assets/logo.png')}

                style={styles.iconImage}></Image>

            </View>

            <View style={styles.appTitleTextContainer}>

              <Text style={ this.state.light\_theme ? styles.appTitleTextLight : styles.appTitleText }>

                App narrar historias

              </Text>

            </View>

          </View>

          <View style={styles.storyContainer}>

            <ScrollView style={ this.state.light\_theme ? styles.storyCardLight : styles.storyCard }>

              <Image

                source={images[this.props.route.params.story.preview\_image]}

                style={styles.image}></Image>

              <View style={styles.dataContainer}>

                <View style={styles.titleTextContainer}>

                  <Text style={ this.state.light\_theme ? styles.storyTitleTextLight : styles.storyTitleText }>

                    {this.props.route.params.story.title}

                  </Text>

                  <Text style={ this.state.light\_theme ? styles.storyAuthorTextLight : styles.storyAuthorText }>

                    {this.props.route.params.story.author}

                  </Text>

                  <Text style={ this.state.light\_theme ? styles.storyAuthorTextLight : styles.storyAuthorText }>

                    {this.props.route.params.story.created\_on}

                  </Text>

                </View>

                <View style={styles.iconContainer}>

                  <TouchableOpacity onPress={() => this.initiateTTS(

                        this.props.route.params.story.title,

                        this.props.route.params.story.author,

                        this.props.route.params.story.story,

                        this.props.route.params.story.moral

                      ) }>

                    <Ionicons name={this.state.speakerIcon} size={RFValue(30)} color={this.state.speakerColor}

                      style={{ margin: RFValue(15) }}  />

                  </TouchableOpacity>

                </View>

              </View>

              <View style={styles.storyTextContainer}>

                <Text style={ this.state.light\_theme ? styles.storyTextLight : styles.storyText }>

                  {this.props.route.params.story.story}

                </Text>

                <Text style={ this.state.light\_theme ? styles.moralTextLight : styles.moralText }>

                  Moral - {this.props.route.params.story.moral}

                </Text>

              </View>

              <View style={styles.actionContainer}>

                <TouchableOpacity style={ this.state.is\_liked ? styles.likeButtonLiked : styles.likeButtonDisliked }

                  onPress={() => this.likeAction()}>

                  <Ionicons name={'heart'} size={RFValue(30)} color={this.state.light\_theme ? 'black' : 'white'} />

                  <Text style={this.state.light\_theme ? styles.likeTextLight : styles.likeText }>

                    {this.state.likes}

                  </Text>

                </TouchableOpacity>

              </View>

            </ScrollView>

          </View>

        </View>

      );

    }

  }

}

No olvidemos cargar los estilos mencionados anteriormente

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#15193c',

  },

  containerLight: {

    flex: 1,

    backgroundColor: 'white',

  },

  droidSafeArea: {

    marginTop:

      Platform.OS === 'android' ? StatusBar.currentHeight : RFValue(35),

  },

  appTitle: {

    flex: 0.07,

    flexDirection: 'row',

  },

  appIcon: {

    flex: 0.3,

    justifyContent: 'center',

    alignItems: 'center',

  },

  iconImage: {

    width: '100%',

    height: '100%',

    resizeMode: 'contain',

  },

  appTitleTextContainer: {

    flex: 0.7,

    justifyContent: 'center',

  },

  appTitleText: {

    color: 'white',

    fontSize: RFValue(28),

    fontFamily: 'Bubblegum-Sans',

  },

  appTitleTextLight: {

    color: 'black',

    fontSize: RFValue(28),

    fontFamily: 'Bubblegum-Sans',

  },

  storyContainer: {

    flex: 1,

  },

  storyCard: {

    margin: RFValue(20),

    backgroundColor: '#2f345d',

    borderRadius: RFValue(20),

  },

  storyCardLight: {

    margin: RFValue(20),

    backgroundColor: 'white',

    borderRadius: RFValue(20),

    shadowColor: 'rgb(0, 0, 0)',

    shadowOffset: {

      width: 3,

      height: 3,

    },

    shadowOpacity: 0.5,

    shadowRadius: 5,

    elevation: 2,

  },

  image: {

    width: '100%',

    alignSelf: 'center',

    height: RFValue(200),

    borderTopLeftRadius: RFValue(20),

    borderTopRightRadius: RFValue(20),

    resizeMode: 'contain',

  },

  dataContainer: {

    flexDirection: 'row',

    padding: RFValue(20),

  },

  titleTextContainer: {

    flex: 0.8,

  },

  storyTitleText: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(25),

    color: 'white',

  },

  storyTitleTextLight: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(25),

    color: 'black',

  },

  storyAuthorText: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(18),

    color: 'white',

  },

  storyAuthorTextLight: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(18),

    color: 'black',

  },

  iconContainer: {

    flex: 0.2,

  },

  storyTextContainer: {

    padding: RFValue(20),

  },

  storyText: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(15),

    color: 'white',

  },

  storyTextLight: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(15),

    color: 'black',

  },

  moralText: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(20),

    color: 'white',

  },

  moralTextLight: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(20),

    color: 'black',

  },

  actionContainer: {

    justifyContent: 'center',

    alignItems: 'center',

    margin: RFValue(10),

  },

  likeButtonLiked: {

    flexDirection: 'row',

    width: RFValue(160),

    height: RFValue(40),

    justifyContent: 'center',

    alignItems: 'center',

    backgroundColor: '#eb3948',

    borderRadius: RFValue(30),

  },

  likeButtonDisliked: {

    flexDirection: 'row',

    width: RFValue(160),

    height: RFValue(40),

    justifyContent: 'center',

    alignItems: 'center',

    borderColor: '#eb3948',

    borderRadius: RFValue(30),

    borderWidth: 2,

  },

  likeText: {

    color: 'white',

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(25),

    marginLeft: RFValue(5),

  },

  likeTextLight: {

    fontFamily: 'Bubblegum-Sans',

    fontSize: RFValue(25),

    marginLeft: RFValue(5),

  },

});