



# Resolución Defensa Hito 3

ESTUDIANTE: GLENN JULIÁN CASTRO DUARTE

ASIGNATURA: PROGRAMACIÓN DE DISPOSITIVOS MÓVILES

DOCENTE: WILLIAM BARRA

# Contenido:

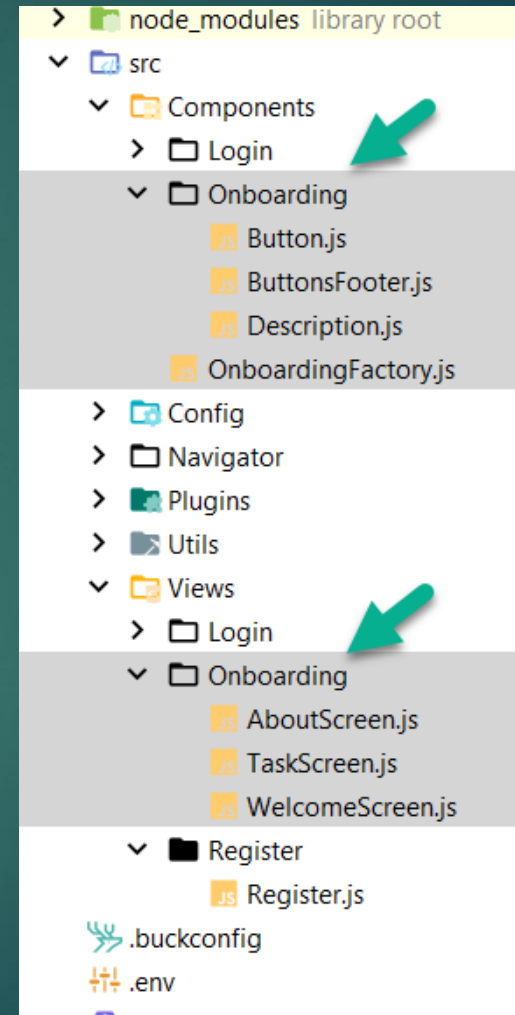
- ▶ Resumen
- ▶ Análisis de los problemas
- ▶ Desarrollo

# Resumen:

- ▶ Esta presentación mostrara la resolución de la evaluación del hito 3 para la materia de Programación de Dispositivos móviles paso a paso

# Análisis del problema:

- ▶ El objetivo es realizar la implementación de las tecnologías React Native y Firebase para poder gestionar autenticación de usuarios y gestión de bases de datos en tiempo real. Operado desde una aplicación móvil multiplataforma compuesta de un Onboarding y de una página de LOGIN.



# Análisis del problema:

- El desafío consiste en la generación de una aplicación que haga uso de la propiedades de navegación entre Screens para posteriormente validar el email mediante FireBase para obtener el siguiente resultado:



# Defensa Hito 3 - Caso de uso: Onboarding

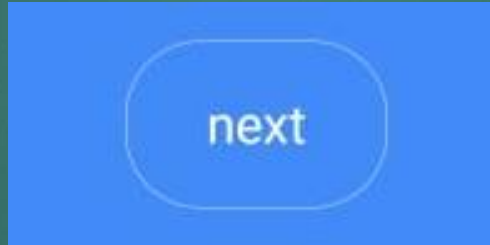
- ▶ Pregunta 1: Crear el componente Button.
- ▶ Pregunta 2: Crear el componente Description.
- ▶ Pregunta 3: Crear el componente ButtonsFooter.
- ▶ Pregunta 4: Crear el componente OnboardingFactory.
- ▶ Pregunta 5: Crear los SCREENS para el navigator y verificar el login a FIREBASE.
  - ▶ Crear la vista AboutScreen.
  - ▶ Crear la vista TaskScreen.
  - ▶ Crear la vista WelcomeScreen.
  - ▶ Verificar y crear el login Firebase.



# Desarrollo

# Pregunta 1: generar el componente Button

- ▶ Se deberá generar un botón genérico, característica transparente y capaz efectuar todas las características de un botón. Deberá ser capaz de utilizarse libremente como un componente independiente.





# Pregunta 1:

- ▶ Se usará componente `TouchableOpacity` que convertirá un determinado espacio para definir que será interactuable, la propiedad `onPress` que define la función que realizara el botón que será, esta propiedad junto con la de texto se importará mediante el `props`.

```
<View>
  <TouchableOpacity
    style={stylesButton.container }
    onPress={this.props.onPress}>
    <Text style={stylesButton.text}>
      {this.props.titleButton}
    </Text>
  </TouchableOpacity>
</View>
```

# Pregunta 1:

- ▶ Mediante la importación de las siguientes librerías:

```
import React, {Component} from 'react';
import {StyleSheet, View, Text, Button,TouchableOpacity} from 'react-native';
import Colors from '../Config/Colors';

export default class Buton extends Component{
  constructor(props){
    super(props);
  }
}
```

# Pregunta 1:

- Para darle el estilo deseado, se modificara el BackGroundColor en transparent, el border color en blanco, y el radius en 5

```
const stylesButton = StyleSheet.create({
  container: {
    width: '100%',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: Colors.red,
    marginBottom: 12,
    paddingVertical: 12,
    borderRadius: 5,
    borderWidth: StyleSheet.hairlineWidth,
    borderColor: 'rgba(255,255,255,0.7)',
  },
  text: {
    color: Colors.white,
    textAlign: 'center',
    height: 20,
  },
});
```

## Pregunta 2: generar el componente FooterButton



# Pregunta 2:

- ▶ Serán necesarias las librerías siguientes:

```
✓ import React, {Component} from 'react';  
  import {StyleSheet, View, Text, TouchableOpacity} from 'react-native';  
  import Colors from '../../Config/Colors';  
  import Buton from './Button';  
  import { styles } from 'expo-ui-kit';  
✓ export default class ButtonFooter extends Component{  
✓ constructor(props){  
  super(props);
```

# Pregunta 2:

- Se llamara 2 button, asignado la propiedad onPress y el title a los props del componente para usarse de manera dinámica despues

```
render(){
  return(
    <View style={stylesButton.row}>
      <Buton
        titleButton={this.props.Prev}
        onPress={this.props.onPressPrev}
      />
      <Buton
        titleButton={this.props.Next}
        onPress={this.props.onPressNext}
      />
    </View>
  );
}
```

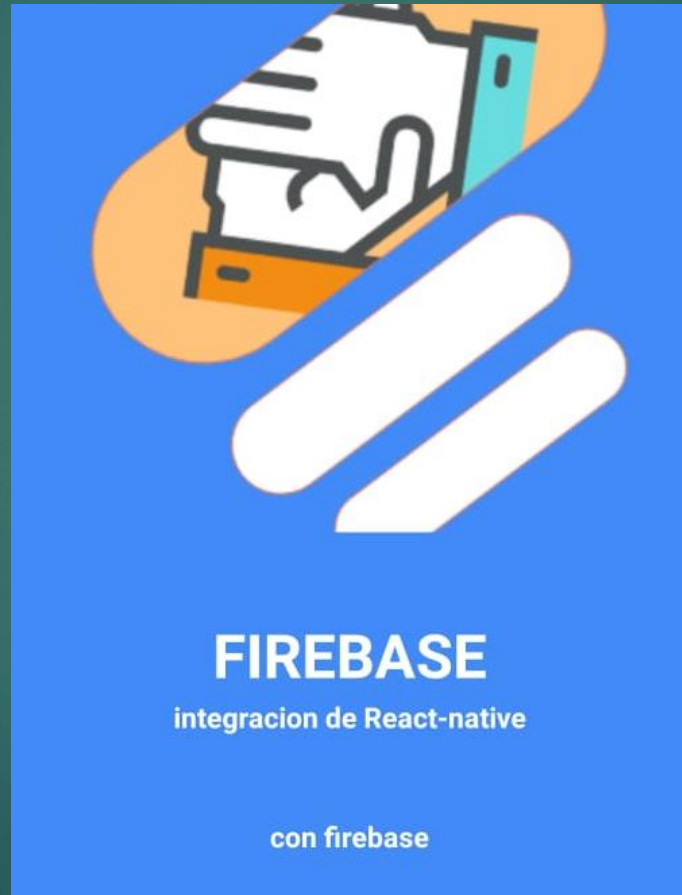
# Pregunta 2:

Para alinear los botones de forma horizontal, se asignara el style row al contenedor que albergue a los botones, uno tras otro.

Se usara este fragmento de código en la sección de styles

```
row:{  
  flex:1,  
  flexDirection:'row',  
  justifyContent:'space-between',  
  marginBottom:10,  
},
```

# Pregunta 3: generar el componente Descripción





# Pregunta 3:

- ▶ Se usaran las siguientes librerías:

```
import React, {Component} from 'react'
import {
  StyleSheet,
  View,
  Text,
  Image
} from 'react-native';
;
import Images from '../../Config/images';
```

# Pregunta 3:

- Se generará a base de una imagen que se usará como logo, 2 contenedores(View), una para el logo y el otro para los textos que se importarán dinámicamente mediante los props de js.

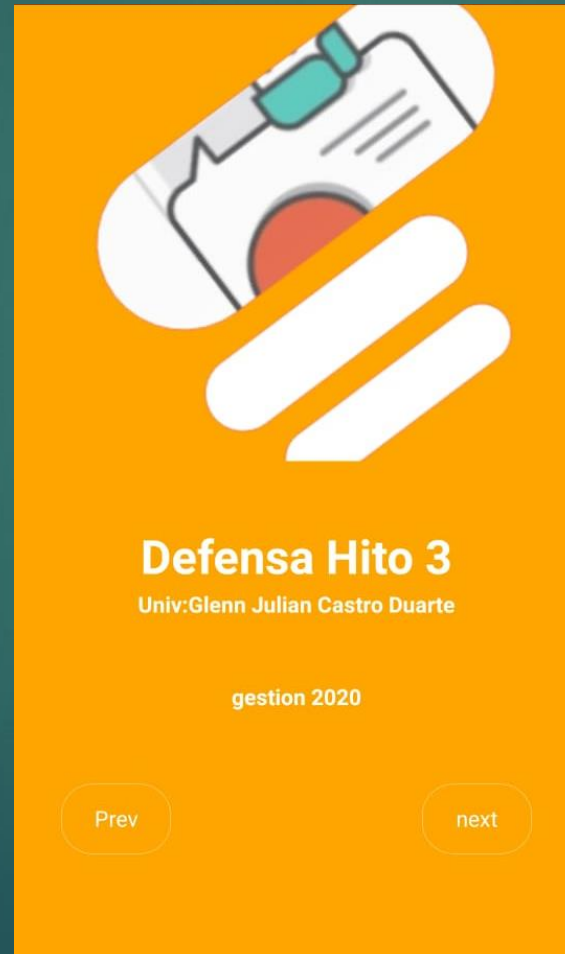
```
<>
  <View style={styles.container}>
    <Image source={this.props.source} style={styles.image}/>
    <Text style={styles.title}>
      {this.props.text1}
    </Text>
  </View>
  <View style={styles.container}>
    <Text style={styles.text}>
      {this.props.text2}
    </Text>
    <Text style={styles.text}>
      {this.props.text3}
    </Text>
  </View>
</>
);
```

# Pregunta 3:

- Los estilos ayudaran a acomodar los textos, con los styles title y text, también contener la imagen en el tamaño deseado con el style image

```
8 }
9 const styles = StyleSheet.create({
10   container:{
11     flex:1,
12     alignItems:'center',
13     justifyContent:'center',
14   },
15   image:{
16     width:300
17     ,
18     height:300,
19   },
20   text:{
21     color:'white',
22     fontWeight:'bold',
23     backgroundColor:'transparent',
24     marginTop:40,
25   },
26   title:{
27     color:'white',
28     fontSize:30 ,
29     fontWeight:'bold',
30     backgroundColor:'transparent',
31     marginTop:40,
32   },
33 })
```

# Pregunta 4: generar el componente onBoardingFactory



# Pregunta 4:

- ▶ Para generar este componente además de importar las librerías también deberemos importar nuestros otros componentes para hacer uso de ellos

```
import React, {useState,Component} from 'react';
import {
  StyleSheet,
  View,
  SafeAreaView,
  KeyboardAvoidingView,
  Alert,
} from 'react-native';

import Colors from '../../Config/Colors';
import ButtonFooter from '../../Components/Onboarding/ButtonFooter';
import Description from '../../Components/Onboarding/Description';
import {NavigationContainer} from '@react-navigation/native';
import {createStackNavigator} from '@react-navigation/stack';
```

# Pregunta 4:

- ▶ Se llamarán a los componentes por sus respectivos nombres y asignar los valores que cada componente pide mediante la propiedad props, en este caso se asigna otra vez el valor props

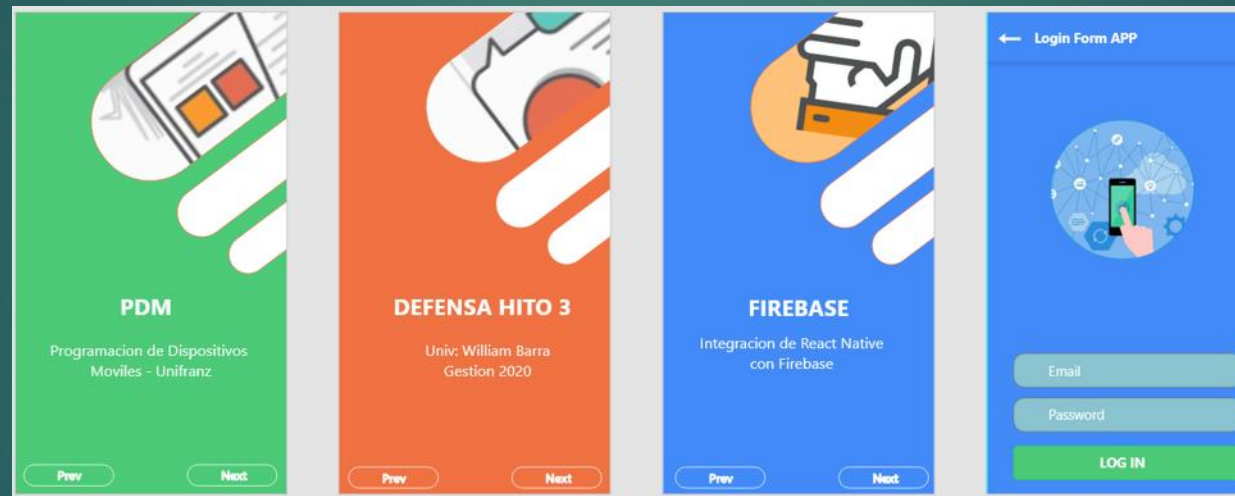
```
export default class OnboardingFactory extends Component{
  constructor(props){
    super(props);
  }
  render(){
    return(
      <View style={this.props.stilo} >
        <View style={{marginTop:100}}>
          <Description
            source={this.props.source}
            text1={this.props.text1}
            text2={this.props.text2}
            text3={this.props.text3}
          />
          <ButtonFooter
            style={{marginTop:100}}
            Next={this.props.Next}
            onPressNext={this.props.onPressNext}
            Prev={this.props.Prev}
            onPressPrev={this.props.onPressPrev}
          />
        </View>
      </View>
    );
  }
}
```

# Pregunta 4:

- Los styles utilizados darán forma a los contenedores de los componentes que vayamos a importar, dándole forma y dimensiones concretas a nuestros componentes.

```
1  const stylesLoginScreen = StyleSheet.create({
2    container: {
3      flex: 1,
4      backgroundColor: Colors.blue,
5      justifyContent: 'center',
6      alignItems: 'center',
7    },
8    logo: {
9      width: '100%',
10     resizeMode: 'contain',
11     alignSelf: 'center',
12   },
13   form: {
14     justifyContent: 'center',
15     width: '80%',
16     marginBottom: 20,
17   },
18 });
```

# Pregunta 5: generación de Screens welcomeScreen, AboutScreen, TaskScreen y Login





# Pregunta 5:

- Todas las screens, exceptuando Login, usaran las siguientes librerías e importaciones :

```
import React, {useState} from 'react';
import {
  StyleSheet,
  View,
  SafeAreaView,
  KeyboardAvoidingView,
  Alert,
} from 'react-native';

import Images from '../Config/images';
import Colors from '../Config/Colors';
import OnboardingFactory from '../Components/Onboarding/OnboardingFactory';
```

# Pregunta 5: WelcomeScreen

- ▶ Al declara como const, podemos hacer uso de la propiedad navigate, que será asignado a la propiedad onPress de nuestros botones, junto con los datos característicos de cada screen

```
const WelcomeScreen = ({navigation}) => {  
  return(  
    <OnboardingFactory  
      stilo={stylesLoginScreen.container}  
      source={Images.SCREEN1}  
      Next={['next']}  
      Prev=['Prev']  
      text1={'PDM'}  
      text2={'Progamacion de dispositivos moviles'}  
      text3={'Unifranz'}  
      onPressNext={()=> navigation.navigate('AboutScreen')}  
    ></OnboardingFactory>  
  )  
}
```

# Pregunta 5: About Screen

- La propiedad navigate de los botones se les asigna junto con una ruta que es recogida desde un mainNavigator, donde se asignaran las rutas y las Screens que se abrirán con las interacción.

```
const TaskScreen = ({navigation}) => {  
  return(  
    <OnboardingFactory  
      stilo={stylesLoginScreen.container}  
      source={Images.SCREEN3}  
      Next={'next'}  
      Prev={'Prev'}  
      text1={'FIREBASE'}  
      text2={'integracion de React-native'}  
      text3={'con firebase'}  
      onPressPrev={() => navigation.navigate('AboutScreen')}  
      onPressNext={() => navigation.navigate('login')}  
    ></OnboardingFactory>  
  )  
}
```

# Proyecto 5: Task Screen

- No es necesario importar las librerías Stack y navigation en las screens, mas si es necesario usarlas en el MainNavigator

```
const AboutScreen = ({navigation}) => {  
  return(  
    <OnboardingFactory  
      stilo={stylesLoginScreen.container}  
      source={Images.SCREEN2}  
      Next={'next'}  
      Prev={['Prev']}  
      text1={'Defensa Hito 3'}  
      text2={'Univ:Glenn Julian Castro Duarte'}  
      text3={'gestion 2020'}  
      onPressNext={()=> navigation.navigate('TaskScreen')}  
      onPressPrev={()=> navigation.navigate('WelcomeScreen')}  
    ></OnboardingFactory>  
  )  
}
```

# Pregunta 5: Login

- ▶ Se importaran distintos elemento y componente previamente diseñados de un Login Prediseñado junto las librerías:

```
import React, {useState} from 'react';
import {
  StyleSheet,
  View,
  SafeAreaView,
  KeyboardAvoidingView,
  Alert,
} from 'react-native';

import ButtonLogin from '../../Components/login/Button';
import TextInputLogin from '../../Components/login/TextInput';
import LogoLogin from '../../Components/login/logo';
import EmailTextField from '../../Components/login/EmailTextField';
import DismissKeyboard from '../../Components/login/DismissKeyboard';
import FirebasePlugin from '../../plugins/firebase/Firebase';

import Utils from '../../utils/util';
import Images from '../../Config/images';
import Constants from '../../Config/constants';
import Colors from '../../Config/Colors';
```

# Pregunta 5: Login

- ▶ Se declarar como una constante, junto con la declaración de las siguientes variables que verificarán los campos y validarán si lo que se ha ingresado en ellos es la sintaxis correcta

```
const LoginScreen = ({navigation}) => {  
  const [email, setEmail] = useState('');  
  const [errorEmail, setErrorEmail] = useState('');  
  const [password, setPassword] = useState('');  
  const [errorPassword, setErrorPassword] = useState('');  
  const [isLoading, setIsLoading] = useState(false);
```

# Pregunta 5 : Login

- ▶ Mediante estas constantes se dará información a las variables en caso que al terminar de ingresar información esta cumpla con los parámetros establecidos en estas constantes

```
const _validateEmailAddress = () => {  
  let isValidEmail = Utils.isValidEmail(email);  
  isValidEmail  
    ? setErrorEmail('')  
    : setErrorEmail(Constants.STRING.EMAIL_ERROR);  
  return isValidEmail;  
};  
  
/**  
 * @name _validatePassword  
 * @returns {boolean}  
 * @private  
 */  
const _validatePassword = () => {  
  let isValidPassword = Utils.isValidField(password);  
  isValidPassword  
    ? setErrorPassword('')  
    : setErrorPassword(Constants.STRING.PASSWORD_ERROR);  
  return isValidPassword;  
};
```

# Pregunta 5: Login

- ▶ En esta constante se ejecutarán la validación de campos junto a toma de datos y enviarlos a otra constante que se encargara de utilizar las funciones de firebase, únicamente si los campos están validados

```
const _onPress = () => {  
  let emailData = _validateEmailAddress();  
  let passwordData = _validatePassword();  
  
  if (emailData && passwordData) {  
    loginApp(email, password);  
  } else {  
    Alert.alert(Constants.STRING.EMPTY_TITLE, Constants.STRING.EMPTY_VALUES);  
  }  
};  
  
/**  
 * @name loginApp  
 * @param {string} email  
 * @param {string} password  
 */
```



# Pregunta 5: Login

- Esta constante usa la función Auth de la librería de firebase para almacenar el email y la contraseña o consultar si ya existe

```
const loginApp = (email, password) => {  
  try {  
    setIsLoading(true);  
    FirebasePlugin.auth()  
      .signInWithEmailAndPassword(email, password)  
      .then(user => {  
        setIsLoading(false);  
        navigation.navigate('register');  
      })  
      .catch(error => {  
        FirebasePlugin.auth()  
          .createUserWithEmailAndPassword(email, password)  
          .then(user => {  
            setIsLoading(false);  
            navigation.navigate('register');  
          })  
          .catch(error => {  
            setIsLoading(false);  
            Alert.alert('Invalid Values', error.message);  
          });  
      });  
  });  
} catch (error) {  
  setIsLoading(true);  
  Alert.alert('Firebase Error', error.message);  
}
```

# Pregunta 5: Login

- Se asignaran las constantes a las variables de recepción de datos para que puedan ejecutarse al momento de presionar el boton

```
<DismissKeyboard>
  <KeyboardAvoidingView
    style={stylesLoginScreen.container}
    behavior="height"
    enabled>
    <View style={stylesLoginScreen.container}>
      <SafeAreaView>
        <LogoLogin style={stylesLoginScreen.logo} />
        <View style={stylesLoginScreen.form}>
          <EmailTextField
            onChangeText={email => {
              setEmail(email);
            }}
            onEndEditing={_validateEmailAddress}
            error={errorEmail}
            source={Images.EMAIL}
            placeholder={Constants.STRING.EMAIL}
            secureTextEntry={false}
            autoComplete={false}
          />
          <TextInputLogin
            onChangeText={password => {
              setPassword(password);
            }}
            onEndEditing={_validatePassword}
            error={errorPassword}
            source={Images.USERNAME}
            placeholder={Constants.STRING.PASSWORD}
            secureTextEntry={true}
            autoComplete={false}
          />
        </View>
      </SafeAreaView>
    </View>
  </KeyboardAvoidingView>
</DismissKeyboard>
```

# Pregunta 5: Login

- ▶ Con ello al realizar una validación correcta, se redireccionara a una screen provisional llamada RegisterScreen.js

```
        </>
        <ButtonLogin
          isLoading={isLoading}
          onPress={_onPress}
          titleButton={Constants.STRING.TITLE_BUTTON}
        />
      </View>
    </SafeAreaView>
  </View>
</KeyboardAvoidingView>
</DismissKeyboard>
);
};
```

# Pregunta 5: Register Screen

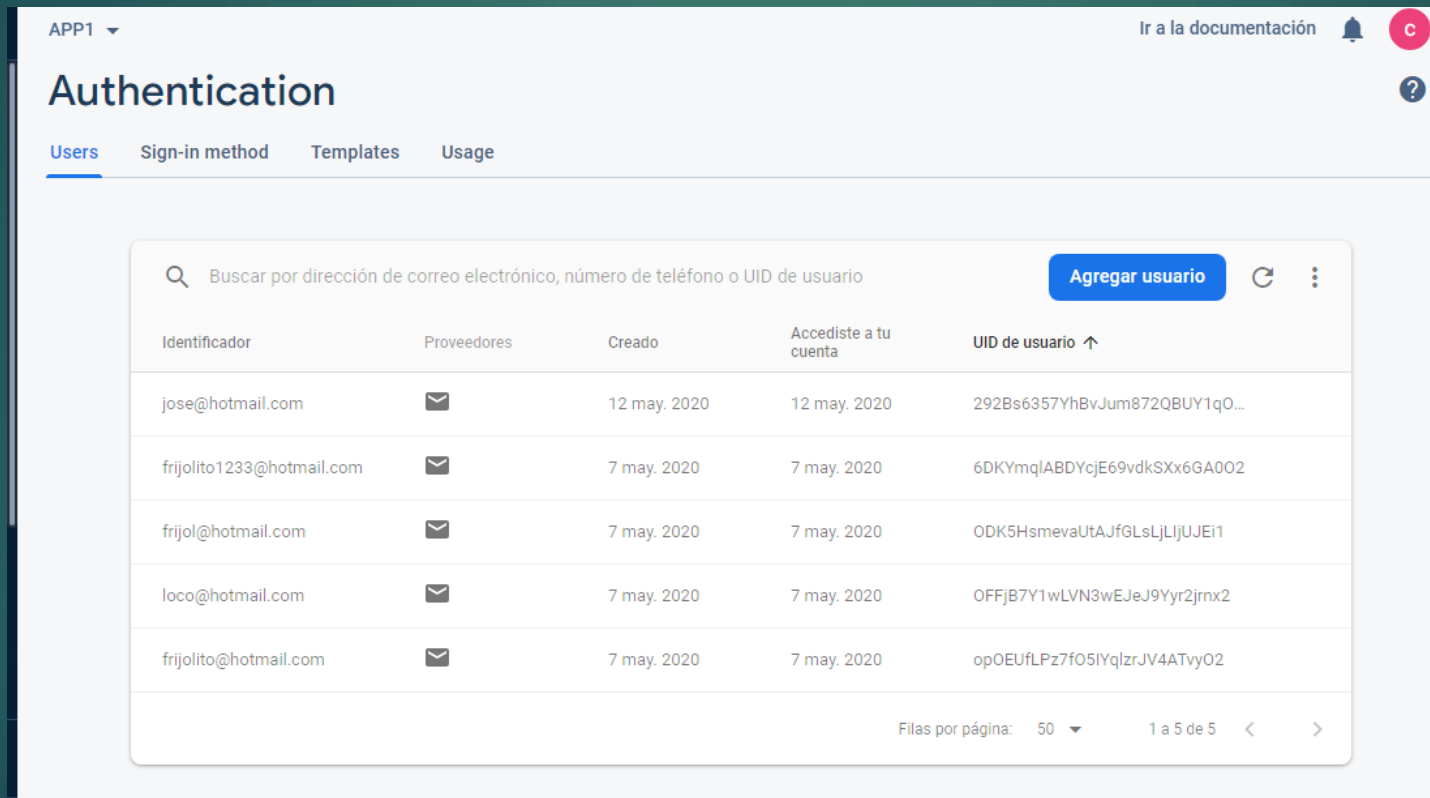
- Una vez validado, la propiedad navigate ejecutara la ruta "register"

```
src > View > register > JS Register.js > [E] styles > text
1  import React from 'react';
2  import {StyleSheet, View, Text} from 'react-native';
3
4  import Constants from '../../Config/constants';
5  import Colors from '../../Config/Colors';
6  import ButtonLogin from '../../Components/login/Button';
7
8  const RegisterScreen = ({navigation}) => {
9    const onPress = () => {
10      console.log('register');
11    };
12
13    return (
14      <View style={styles.container}>
15        <Text>Register Screen</Text>
16        <ButtonLogin onPress={onPress} titleButton={Constants.STRING.REGISTER} />
17      </View>
18    );
19  };
20
```



# Pregunta 5: Registro en firebase

- El resultado final deberá poder apreciarse en nuestra cuenta



The screenshot shows the Firebase Authentication console. At the top, there's a header with 'APP1' and a dropdown, 'Ir a la documentación', and a user profile icon. Below the header, the 'Authentication' title is followed by tabs: 'Users' (selected), 'Sign-in method', 'Templates', and 'Usage'. A search bar is present with the placeholder text 'Buscar por dirección de correo electrónico, número de teléfono o UID de usuario'. To the right of the search bar is a blue button labeled 'Agregar usuario' and a refresh icon. The main content is a table with five columns: 'Identificador', 'Proveedores', 'Creado', 'Accediste a tu cuenta', and 'UID de usuario'. The table contains five rows of user data. At the bottom right, there's a pagination control showing 'Filas por página: 50' and '1 a 5 de 5'.

Identificador	Proveedores	Creado	Accediste a tu cuenta	UID de usuario ↑
jose@hotmail.com	✉	12 may. 2020	12 may. 2020	292Bs6357YhBvJum872QBUI1qO...
frijolito1233@hotmail.com	✉	7 may. 2020	7 may. 2020	6DKYmqIABDYcjE69vdkSXx6GA002
frijol@hotmail.com	✉	7 may. 2020	7 may. 2020	ODK5HsmevaUtAJfGLsLjLIJUEi1
loco@hotmail.com	✉	7 may. 2020	7 may. 2020	OFFjB7Y1wLVN3wEJeJ9Yyr2jrnX2
frijolito@hotmail.com	✉	7 may. 2020	7 may. 2020	op0EUfLPz7f05IYqlzrJV4ATvy02