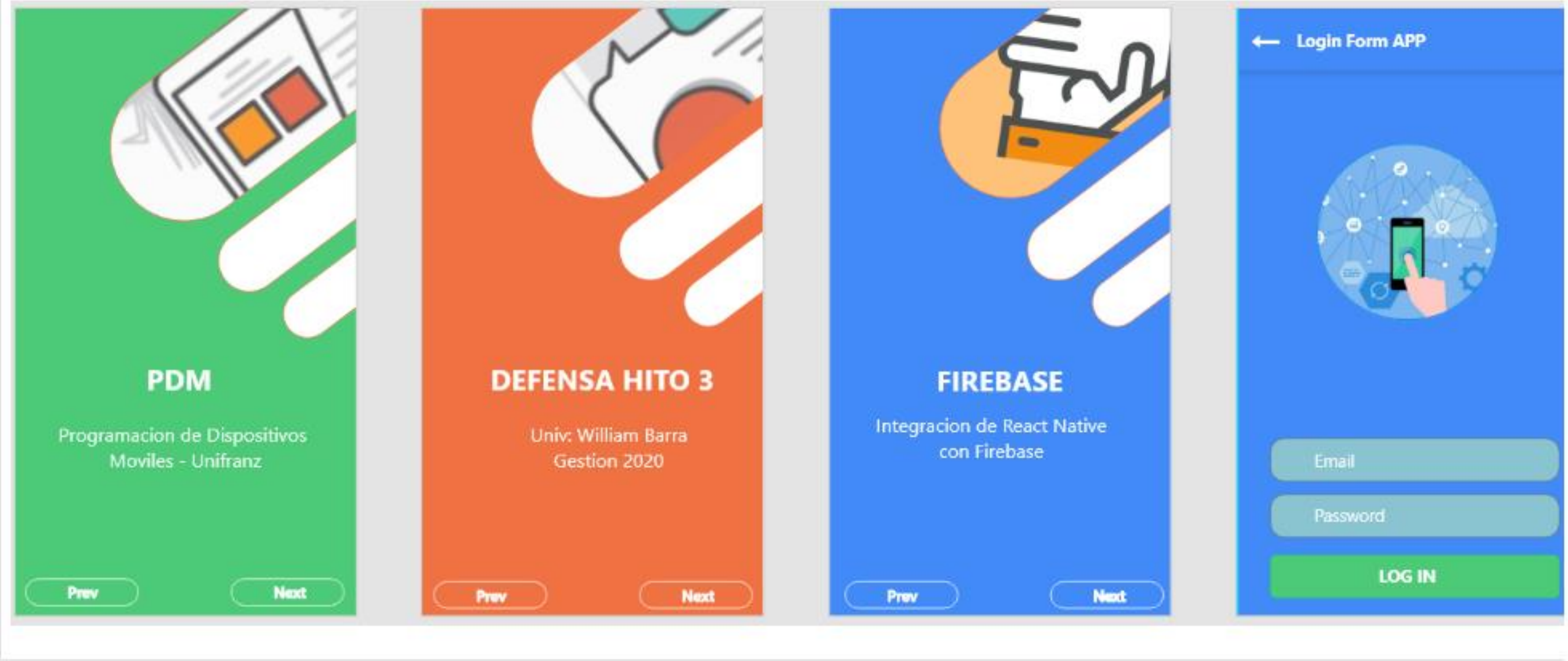
DEFENSA HITO3 PROGRAMACION DE DISPOSITOVOS MOVILES

ESTEBAN DANIEL PEREIRA NUÑEZ

7MO SEMESTRE

Examen Hito3 – Onboarding & Firebase



PDM

Programacion de Dispositivos Moviles - Unifranz

DEFENSA HITO 3

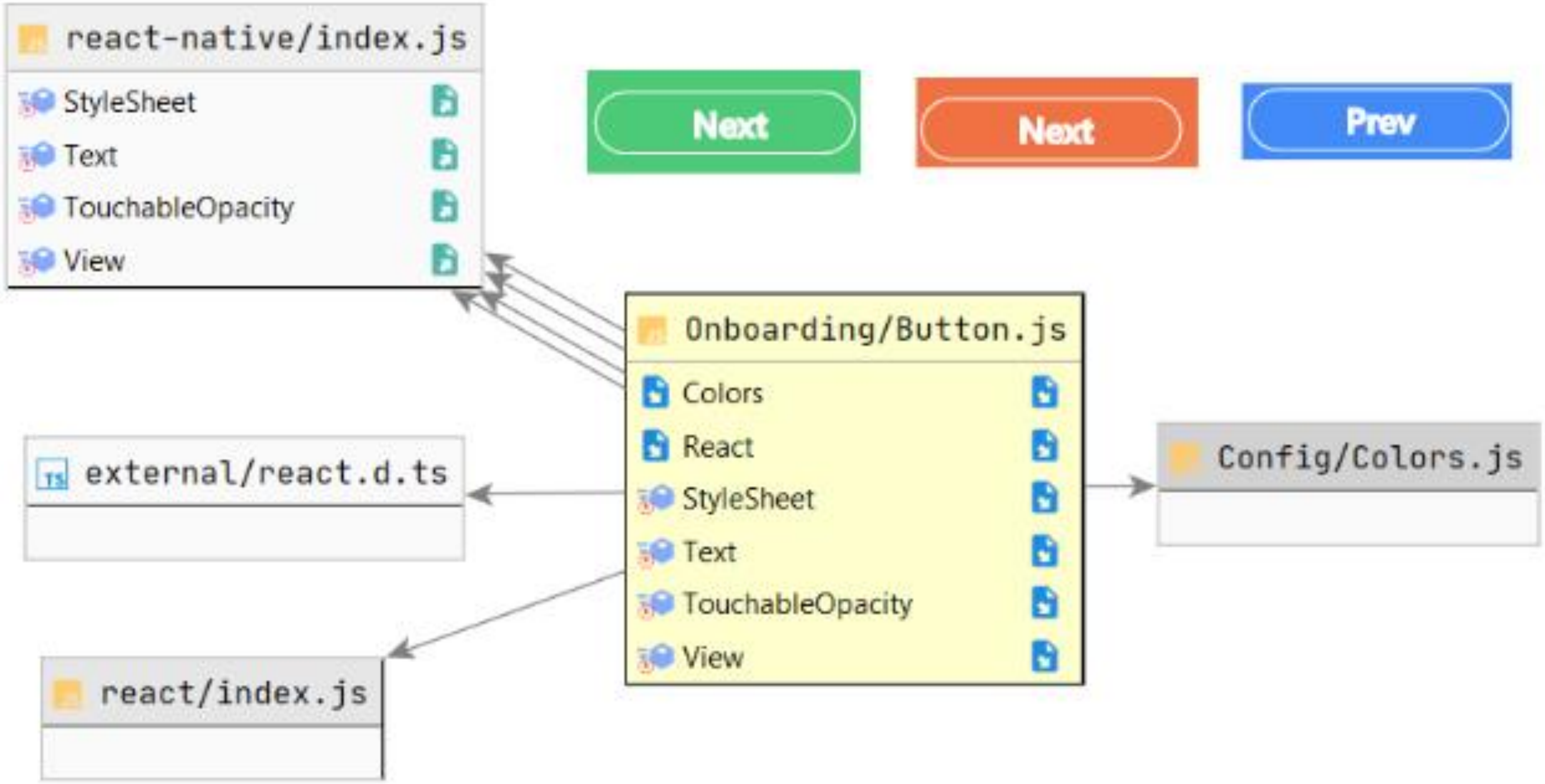
Univ: William Barra
Gestion 2020

FIREBASE

Integracion de React Native con Firebase

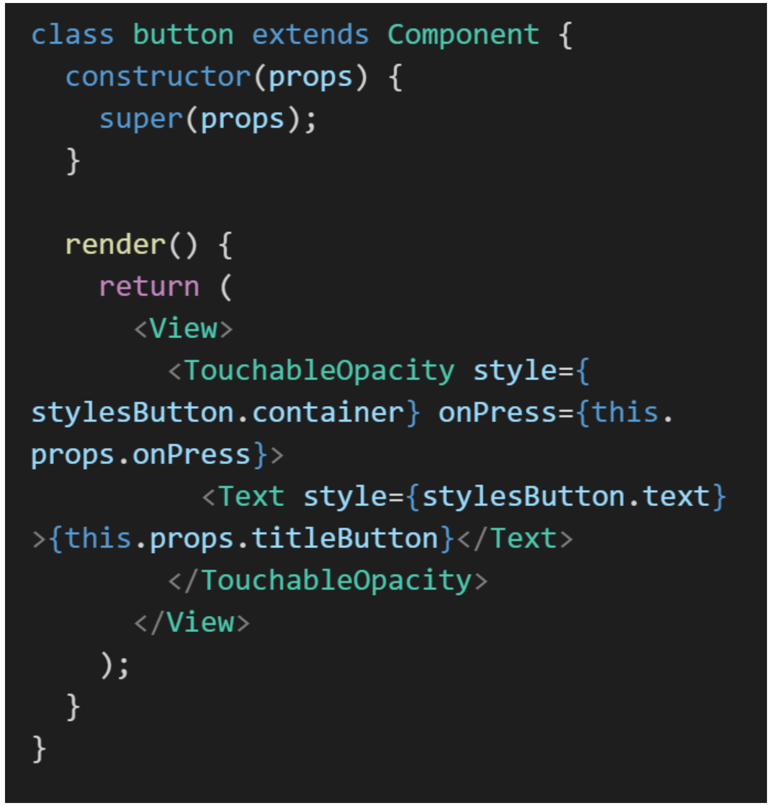
Login Form APP

Paso-1:Crear el componente Button para las acciones PREV y NEXT.



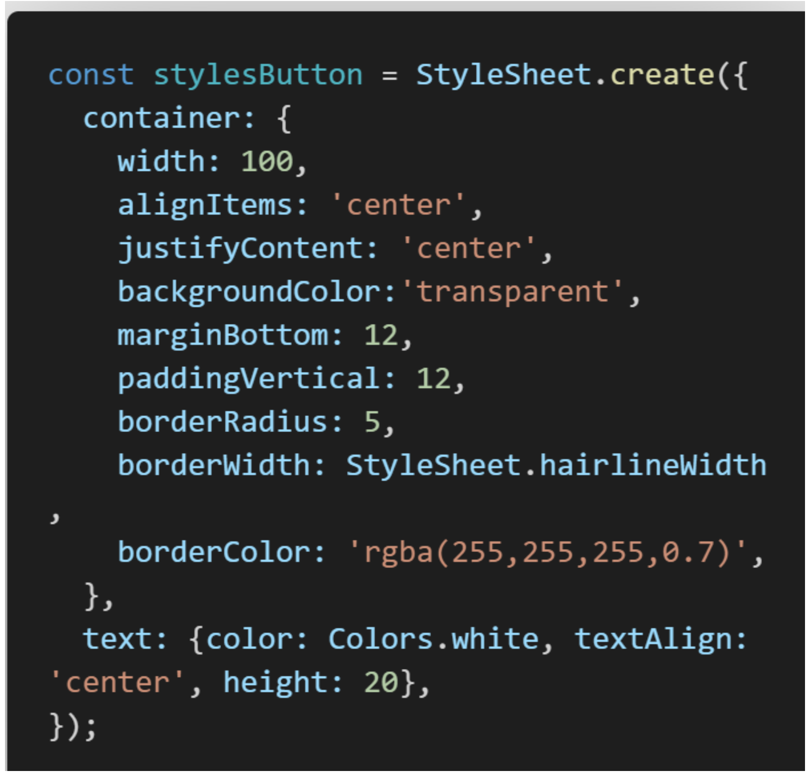
Codigo y funcionamiento

Código



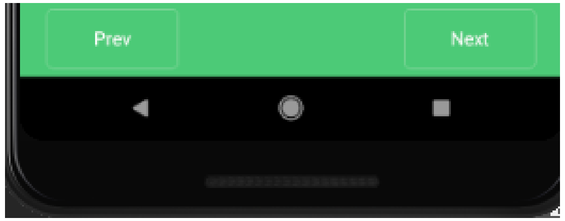
```
class button extends Component {
  constructor(props) {
    super(props);
  }

  render() {
    return (
      <View>
        <TouchableOpacity style={
stylesButton.container} onPress={this.
props.onPress}>
          <Text style={stylesButton.text}
>{this.props.titleButton}</Text>
        </TouchableOpacity>
      </View>
    );
  }
}
```

Estilos

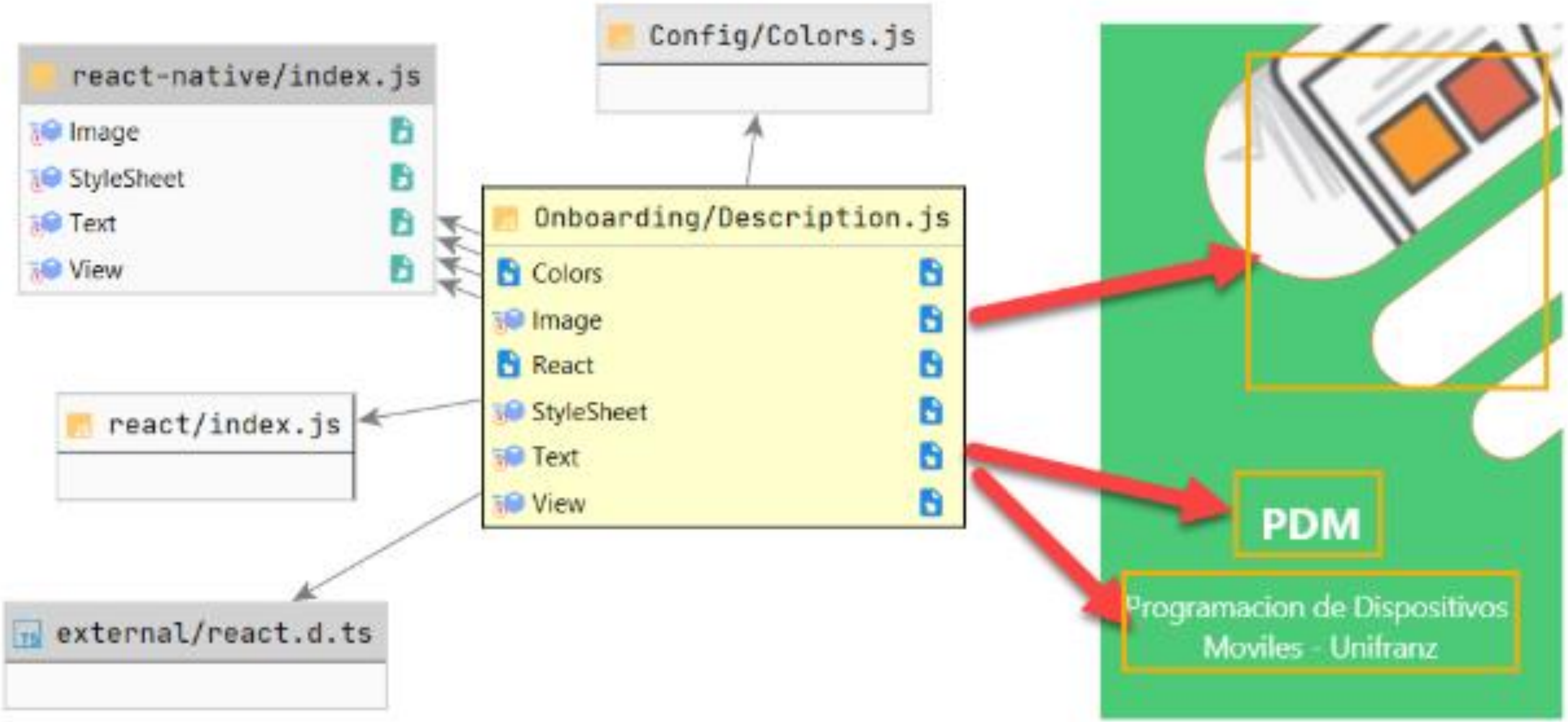


```
const stylesButton = StyleSheet.create({
  container: {
    width: 100,
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor:'transparent',
    marginBottom: 12,
    paddingVertical: 12,
    borderRadius: 5,
    borderWidth: StyleSheet.hairlineWidth
,
    borderColor: 'rgba(255,255,255,0.7)',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

Vista

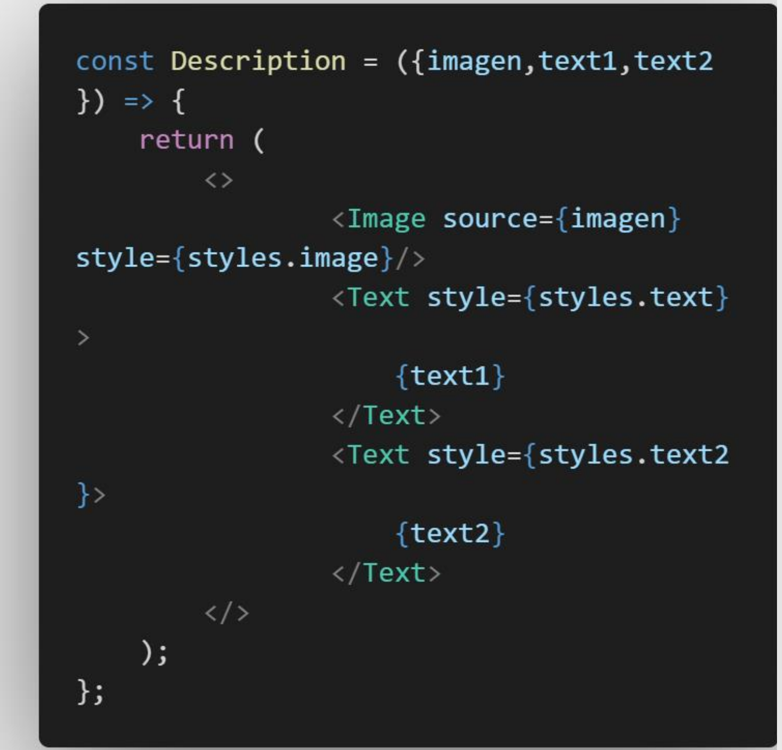


Paso-2: Crear el componente Description.js



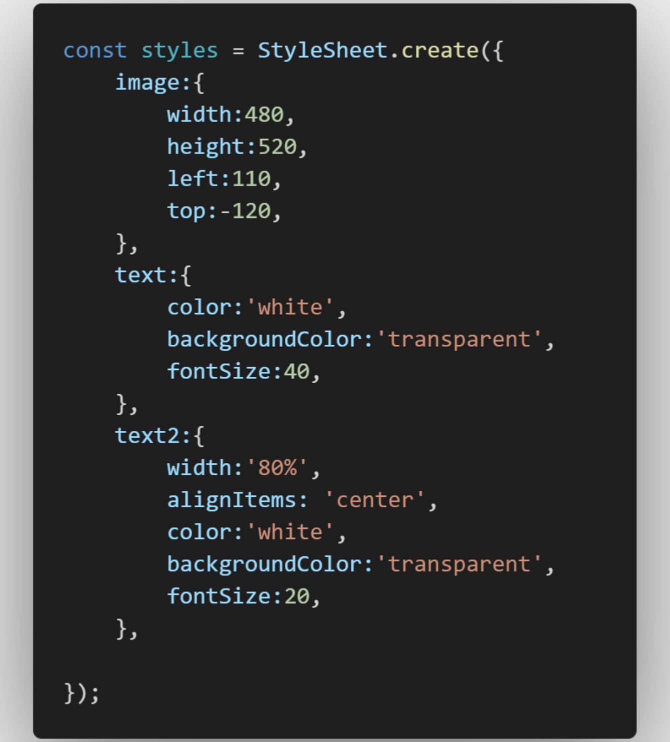
Codigo y funcionamiento

Código



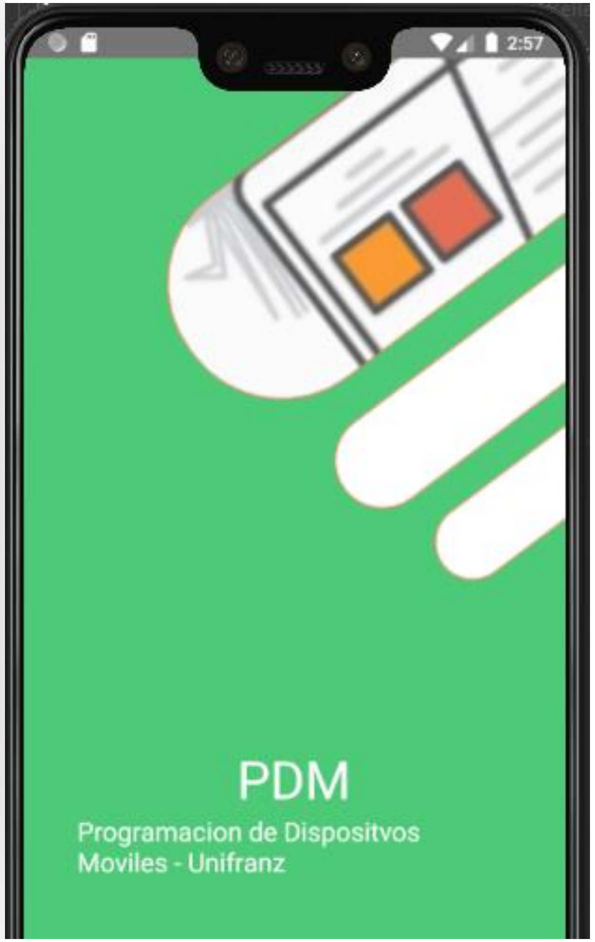
```
const Description = ({imagen,text1,text2
}) => {
    return (
        <>
                <Image source={imagen}
style={styles.image}/>
                <Text style={styles.text}
>
                    {text1}
                </Text>
                <Text style={styles.text2
}>
                    {text2}
                </Text>
        </>
    );
};
```

Estilos



```
const styles = StyleSheet.create({
    image:{
        width:480,
        height:520,
        left:110,
        top:-120,
    },
    text:{
        color:'white',
        backgroundColor:'transparent',
        fontSize:40,
    },
    text2:{
        width:'80%',
        alignItems: 'center',
        color:'white',
        backgroundColor:'transparent',
        fontSize:20,
    },

});
```

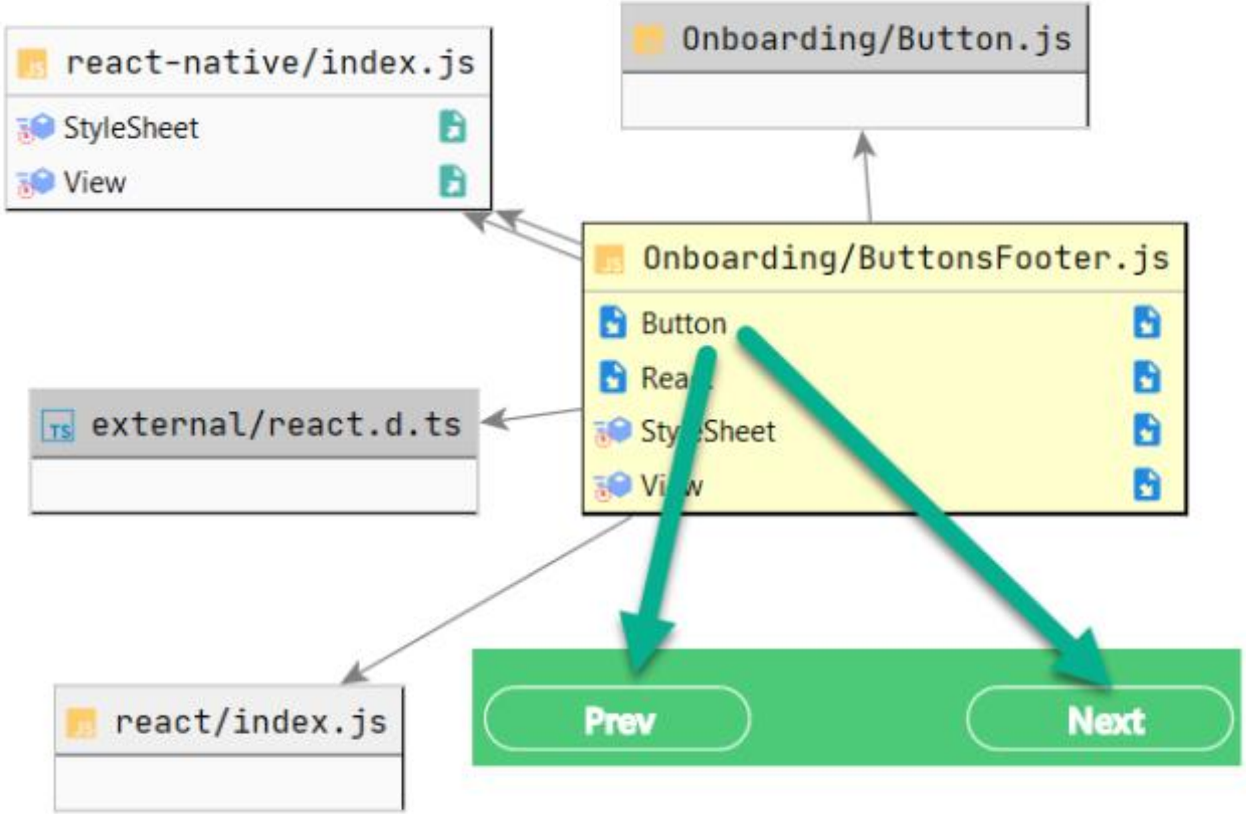
Vista



PDM

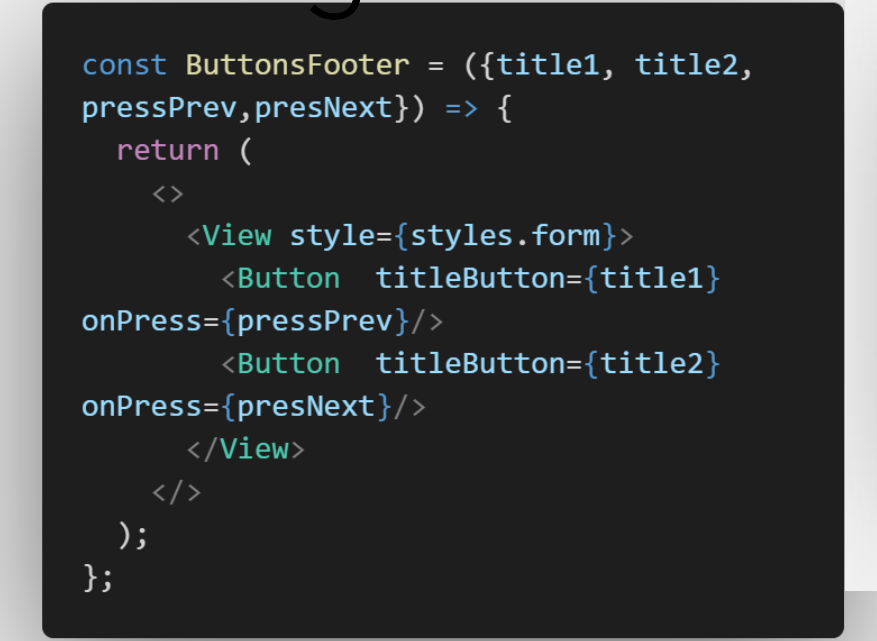
Programacion de Dispositvos
Moviles - Unifranz

Paso-3: Crear el componente ButtonsFoteer.js



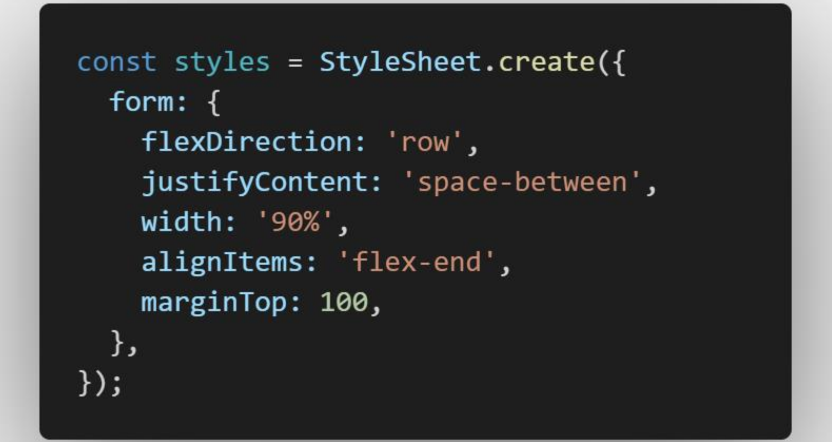
Codigo y funcionamiento

Código



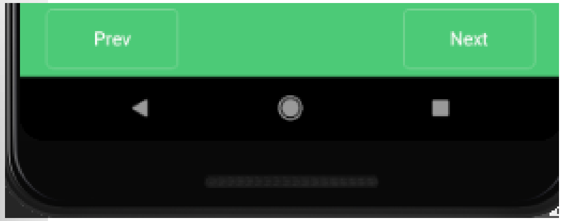
```
const ButtonsFooter = ({title1, title2,
pressPrev,presNext}) => {
  return (
    <>
      <View style={styles.form}>
        <Button  titleButton={title1}
onPress={pressPrev}/>
        <Button  titleButton={title2}
onPress={presNext}/>
      </View>
    </>
  );
};
```

Estilos

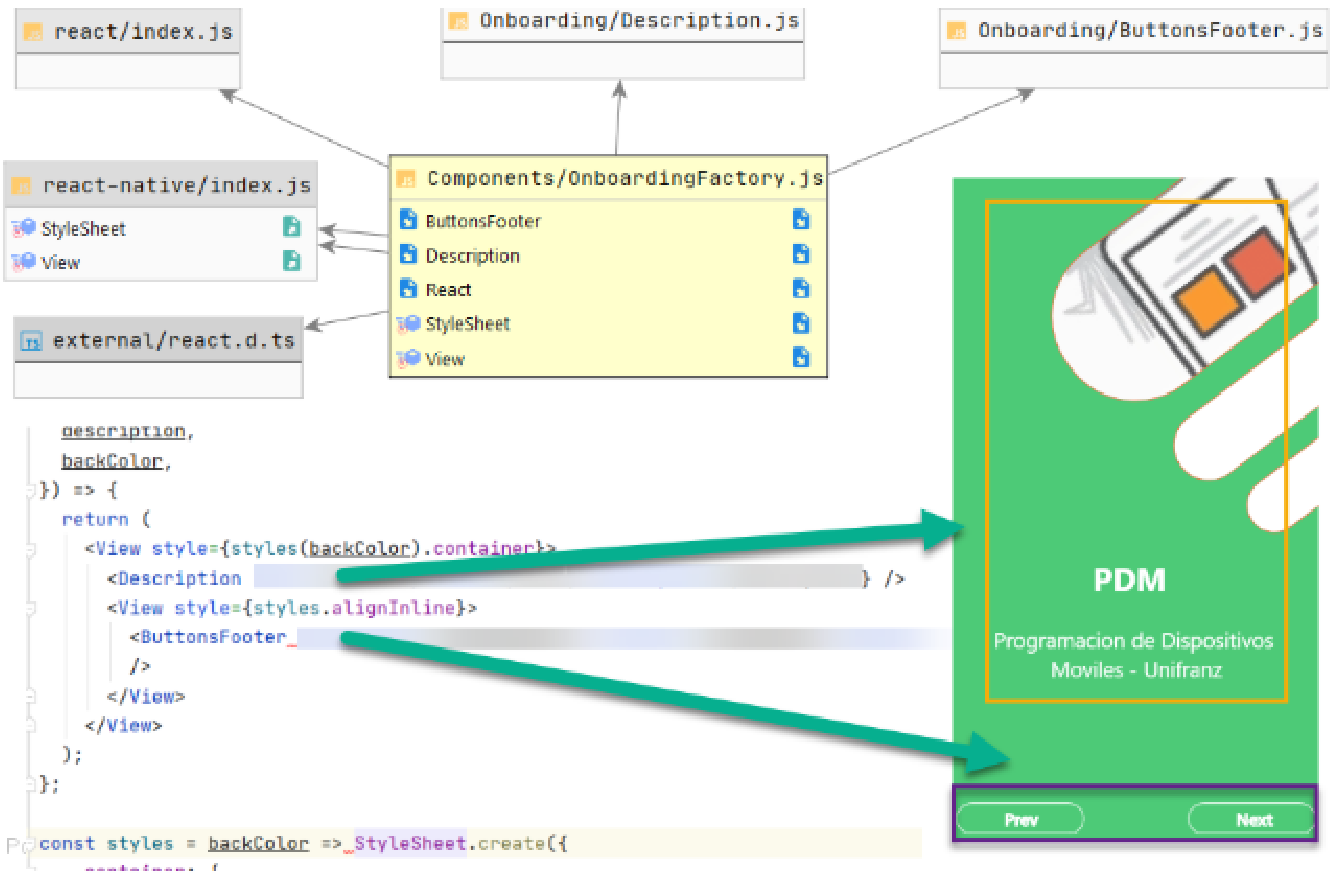


```
const styles = StyleSheet.create({
  form: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    width: '90%',
    alignItems: 'flex-end',
    marginTop: 100,
  },
});
```

Vista



Paso-4: Crear el componente OnboardingFactory.js

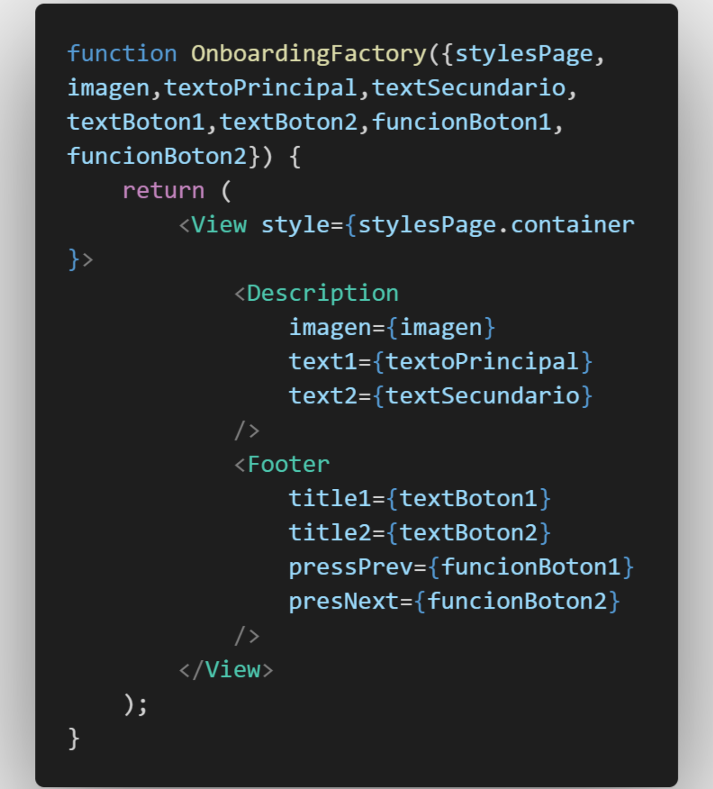


react/index.js
Onboarding/Description.js
Onboarding/ButtonsFooter.js
react-native/index.js
StyleSheet
View
Components/OnboardingFactory.js
ButtonsFooter
Description
React
StyleSheet
View
external/react.d.ts
PDM
Programacion de Dispositivos Moviles - Unifranz
Prev
Next

```
    description,
    backColor,
}) => {
    return (
        <View style={styles(backColor).container}>
            <Description [redacted] } />
            <View style={styles.alignInline}>
                <ButtonsFooter [redacted]
                />
            </View>
        </View>
    );
};

const styles = backColor => StyleSheet.create({
    container: {
```

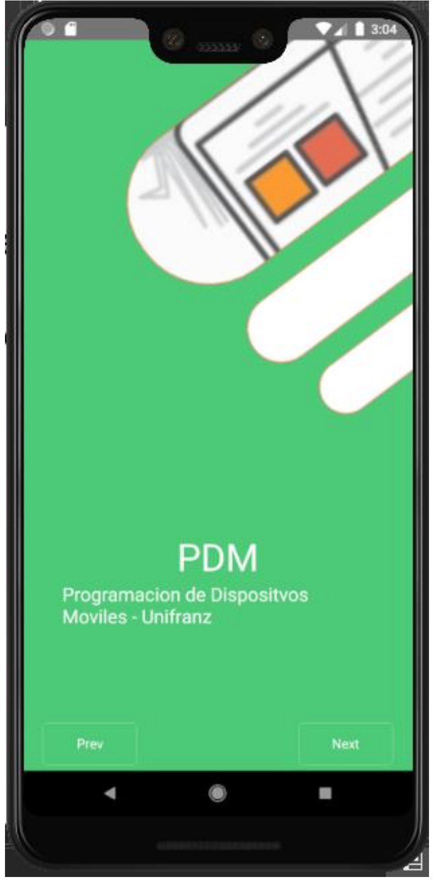
Codigo y funcionamiento

Código



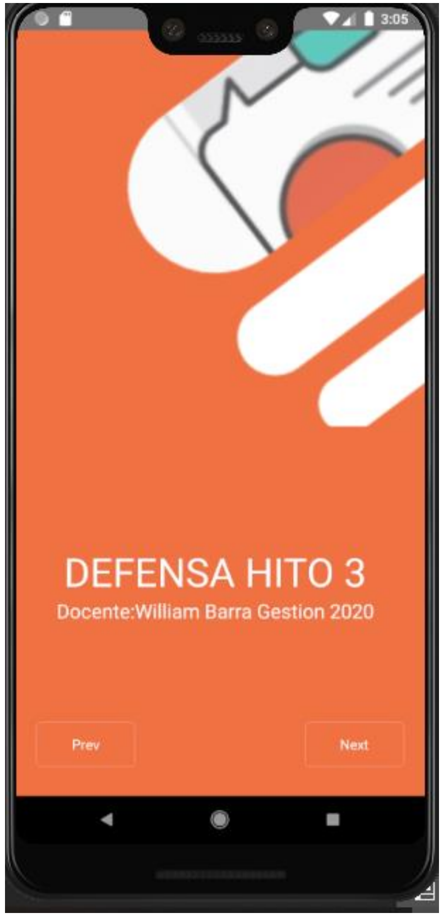
```
function OnboardingFactory({stylesPage,
imagen,textoPrincipal,textSecundario,
textBoton1,textBoton2,funcionBoton1,
funcionBoton2}) {
    return (
        <View style={stylesPage.container
}>
            <Description
                imagen={imagen}
                text1={textoPrincipal}
                text2={textSecundario}
            />
            <Footer
                title1={textBoton1}
                title2={textBoton2}
                pressPrev={funcionBoton1}
                presNext={funcionBoton2}
            />
        </View>
    );
}
```

Vista



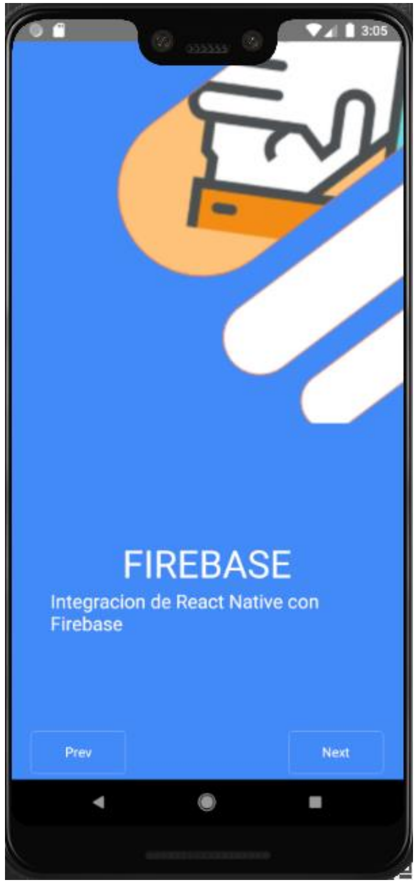
PDM

Programacion de Dispositvos Moviles - Unifranz



DEFENSA HITO 3

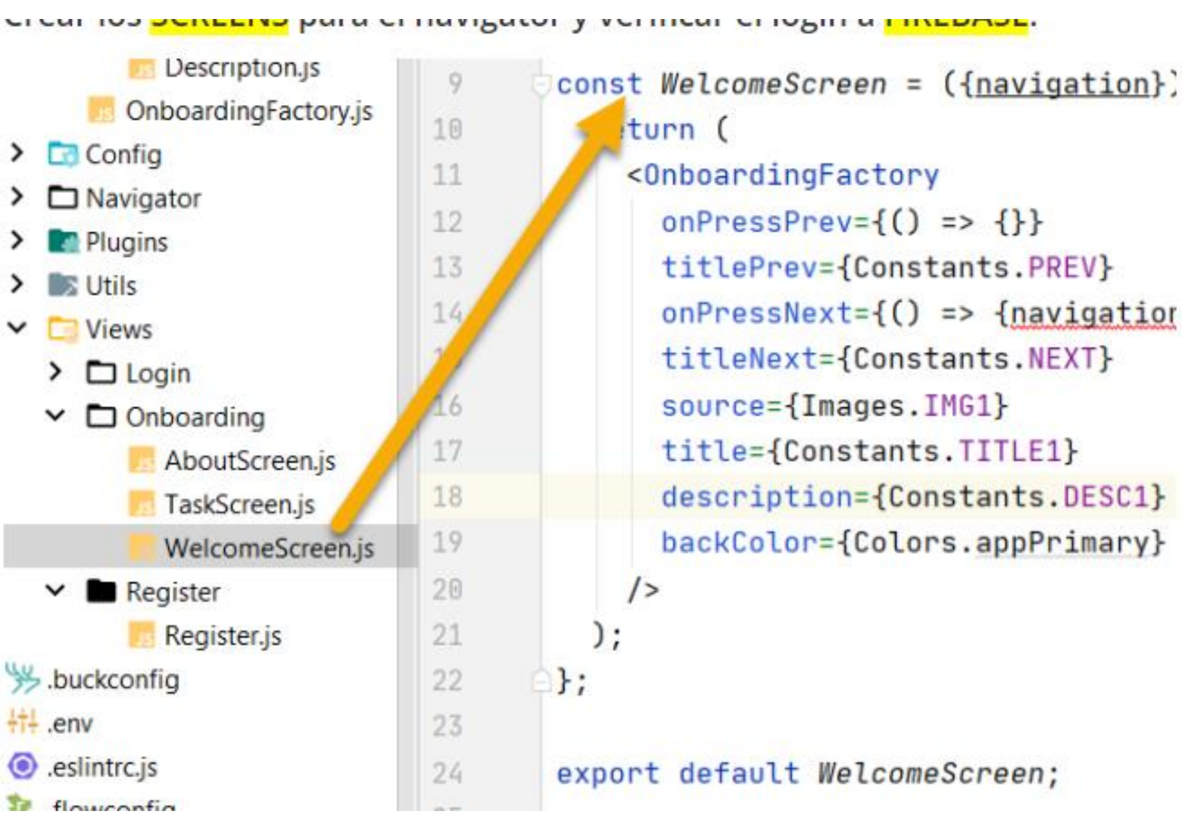
Docente:William Barra Gestion 2020



FIREBASE

Integracion de React Native con Firebase

Paso-5:Crear los SCREENS para el navigator y verificar el login a FIREBASE.



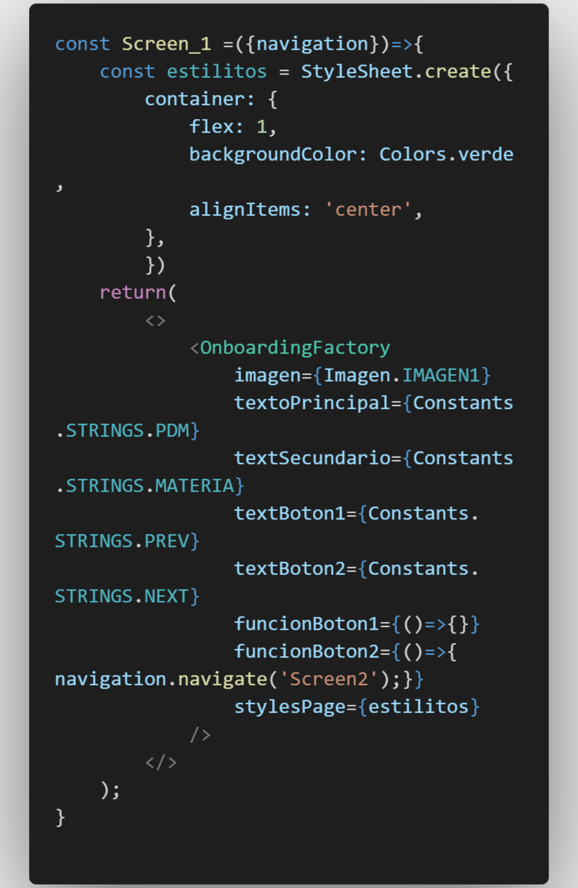
Crear los SCREENS para el navigator y verificar el login a FIREBASE.

```
Description.js
OnboardingFactory.js
> Config
> Navigator
> Plugins
> Utils
v Views
  > Login
  v Onboarding
      AboutScreen.js
      TaskScreen.js
      WelcomeScreen.js
  v Register
      Register.js
.buckconfig
.env
.eslintrc.js
.flowconfig
```

```
const WelcomeScreen = ({navigation})
  return (
    <OnboardingFactory
      onPressPrev={() => {}}
      titlePrev={Constants.PREV}
      onPressNext={() => {navigatior
      titleNext={Constants.NEXT}
      source={Images.IMG1}
      title={Constants.TITLE1}
      description={Constants.DESC1}
      backColor={Colors.appPrimary}
    />
  );
};

export default WelcomeScreen;
```

Codigo y funcionamiento

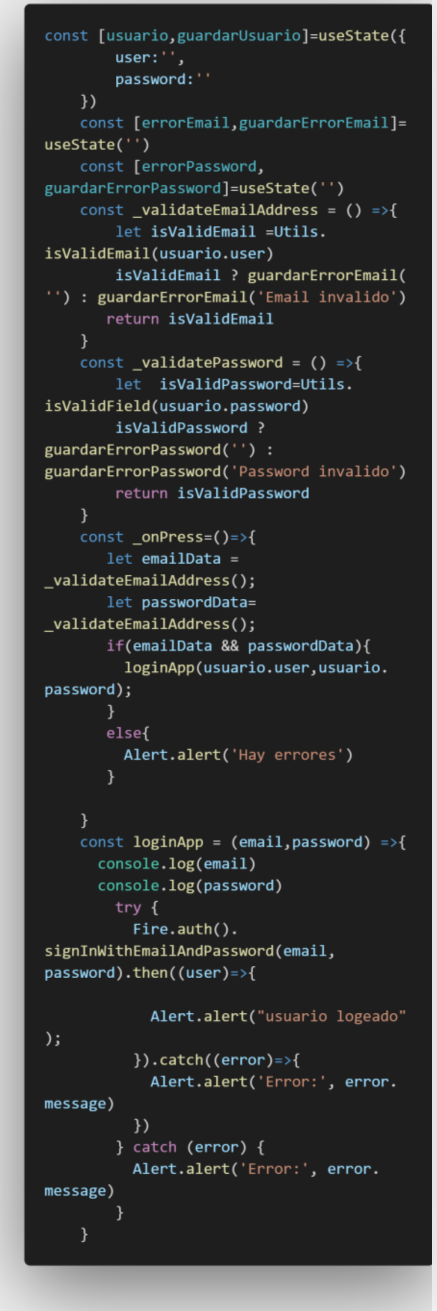
1



```
const Screen_1 =({navigation})=>{
    const estilitos = StyleSheet.create({
        container: {
            flex: 1,
            backgroundColor: Colors.verde
,
            alignItems: 'center',
        },
        })
    return(
        <>
            <OnboardingFactory
                imagen={Imagen.IMAGEN1}
                textoPrincipal={Constants
.STRINGS.PDM}
                textSecundario={Constants
.STRINGS.MATERIA}
                textBoton1={Constants.
STRINGS.PREV}
                textBoton2={Constants.
STRINGS.NEXT}
                funcionBoton1={()=>{}}
                funcionBoton2={()=>{
navigation.navigate('Screen2');}}
                stylesPage={estilitos}
            />
        </>
    );
}
```

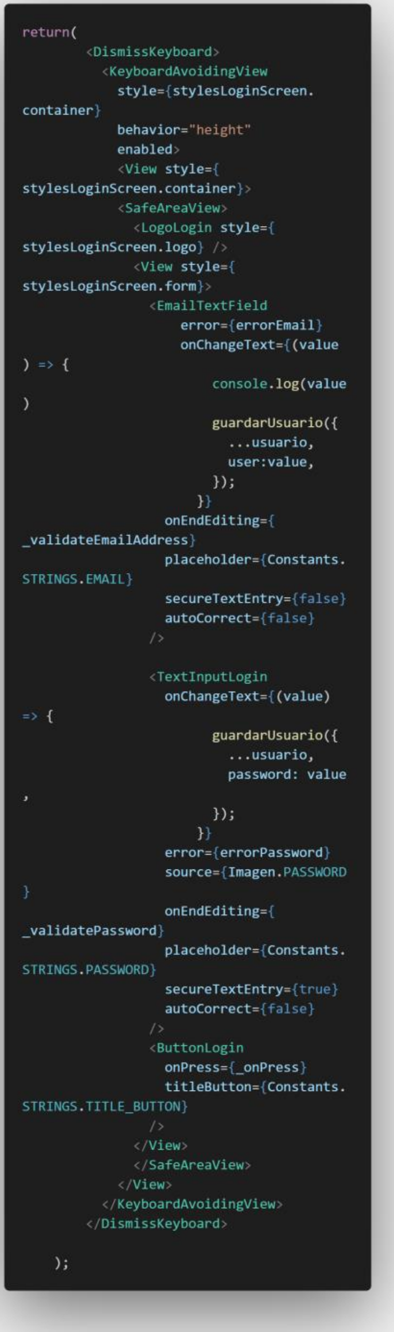
1.Screen
2.BackEnd Login
3.FrontEnd Login

2



```
const [usuario,guardarUsuario]=useState({
        user:'',
        password:''
    })
    const [errorEmail,guardarErrorEmail]=
useState('')
    const [errorPassword,
guardarErrorPassword]=useState('')
    const _validateEmailAddress = () =>{
        let isValidEmail =Utils.
isValidEmail(usuario.user)
        isValidEmail ? guardarErrorEmail(
'') : guardarErrorEmail('Email invalido')
       return isValidEmail
    }
    const _validatePassword = () =>{
        let  isValidPassword=Utils.
isValidField(usuario.password)
        isValidPassword ?
guardarErrorPassword('') :
guardarErrorPassword('Password invalido')
        return isValidPassword
    }
    const _onPress=()=>{
       let emailData =
_validateEmailAddress();
       let passwordData=
_validateEmailAddress();
       if(emailData && passwordData){
         loginApp(usuario.user,usuario.
password);
       }
       else{
         Alert.alert('Hay errores')
       }

    }
    const loginApp = (email,password) =>{
      console.log(email)
      console.log(password)
        try {
          Fire.auth().
signInWithEmailAndPassword(email,
password).then((user)=>{

            Alert.alert("usuario logeado"
);
          }).catch((error)=>{
            Alert.alert('Error:', error.
message)
          })
        } catch (error) {
          Alert.alert('Error:', error.
message)
        }
    }
```

3



```
return(
        <DismissKeyboard>
          <KeyboardAvoidingView
            style={stylesLoginScreen.
container}
            behavior="height"
            enabled>
            <View style={
stylesLoginScreen.container}>
            <SafeAreaView>
              <LogoLogin style={
stylesLoginScreen.logo} />
              <View style={
stylesLoginScreen.form}>
                <EmailTextField
                    error={errorEmail}
                    onChangeText={(value
) => {
                        console.log(value
)
                        guardarUsuario({
                          ...usuario,
                          user:value,
                        });
                      }}
                  onEndEditing={
_validateEmailAddress}
                  placeholder={Constants.
STRINGS.EMAIL}
                  secureTextEntry={false}
                  autoCorrect={false}
                />

                <TextInputLogin
                  onChangeText={(value)
=> {
                        guardarUsuario({
                          ...usuario,
                          password: value
,
                        });
                      }}
                  error={errorPassword}
                  source={Imagen.PASSWORD
}
                  onEndEditing={
_validatePassword}
                  placeholder={Constants.
STRINGS.PASSWORD}
                  secureTextEntry={true}
                  autoCorrect={false}
                />
                <ButtonLogin
                  onPress={_onPress}
                  titleButton={Constants.
STRINGS.TITLE_BUTTON}
                />
              </View>
              </SafeAreaView>
            </View>
          </KeyboardAvoidingView>
        </DismissKeyboard>

    );
```

FIN