



**Diseño y programación de software  
multiplataforma  
Ciclo 02 2024**

**Docente: Ing. Alexander Siguenza**

**“Foro #2 Inicio de sesión de React Native”**

**Presentado por:**

**David Eduardo Rodriguez Vigil – RV202840**

**El Salvador, 24 de noviembre de 2024**

## **Autenticación en React Native con conexión a firebase.**

### **Autenticación con Correo Electrónico y Contraseña:**

Este tipo de autenticación permite registrar y autenticar a los diferentes usuarios usando el correo electrónico y la contraseña que el usuario halla ingresado en el registro. Es muy simple y útil para aplicaciones que no requieren métodos de autenticación más complejos.

Características:

- Los usuarios se registran e inician sesión usando un correo electrónico y contraseña.
- Firebase almacena de forma segura las credenciales.
- Compatible con la verificación de correo electrónico.

Ventajas:

- Fácil de implementar.
- Familiar para la mayoría de los usuarios.
- Puede combinarse con la verificación de correo para mejorar la seguridad.

Desventajas:

- Requiere que los usuarios recuerden una contraseña.
- Menor conversión en comparación con métodos sociales.

Consideraciones:

- Habilitar las validaciones y verificación de correo electrónico para evitar cuentas falsas.
- Implementar validaciones para contraseñas seguras.

### **Autenticación con Proveedores Externos:**

Este tipo de autenticación con firebase puede soportar la autenticación con varios proveedores externos como pueden ser: Google, Facebook, Apple, Microsoft, etc. Esto es útil para los diferentes usuarios que prefieren iniciar sesión con cuentas que

ya poseen en otras plataformas. Además de que cada proveedor tiene su propia configuración y permisos que deben habilitarse en la consola de firebase y requiere una integración de SDK específicos en el proyecto.

Características:

- Soporte para Google, Facebook, Microsoft, etc.
- Firebase maneja el flujo de inicio de sesión con los SDK de los proveedores.

Ventajas:

- Aumenta la tasa de conversión.
- Los usuarios no necesitan crear una nueva cuenta o contraseña.
- Ideal para integrarse con otros servicios.

Desventajas:

- Configuración adicional necesarias como API, SDK específicos.
- Dependencia de servicios de terceros.
- Requiere permisos específicos en dispositivos móviles.

Consideraciones:

- Asegurar de cumplir con las políticas de privacidad de los proveedores para poder ofrecer este inicio de sesión.
- Habilita múltiples proveedores para cubrir preferencia de los usuarios y poder bríndales un mejor servicio.

### **Autenticación con Número de Teléfono:**

Este tipo de autenticación permiten que a través que los usuarios ingresen su número de teléfono reciban un SMS, donde el usuario recibirá un código de verificación para inicio de sesión. Esto requiere una configuración en firebase para permitir la autenticación por teléfono y configurar correctamente las políticas de verificación.

Características:

- Los usuarios reciben a sus teléfonos un código de verificación por SMS para la autenticación.
- Firebase gestiona el envío de los mensajes y la validación del código.

#### Ventajas:

- No requiere contraseñas
- Ideal para mercados donde el número de teléfono es el método más utilizado para los usuarios.
- Reduce el tiempo de registro.

#### Desventajas:

- Dependencia a los servicios de mensajería.
- Puede ser menos confiable en áreas con mala recepción móvil.
- La seguridad depende del acceso al dispositivo.

#### Consideraciones:

- Configurar políticas de uso justo para evitar abusos del servicio.
- Manejo de excepciones para errores en la entrega de SMS.

### **Autenticación Anónima:**

Este tipo de autenticación permite que los diferentes usuarios interactúen con la aplicación sin una cuenta registrada y es útil cuando se les quiere dar acceso a contenido temporal o limitado antes de solicitar un registro formal.

#### Características:

- Permite a los diferentes usuarios con la aplicación sin crear una cuenta.
- Se puede actualizar a una cuenta permanente más tarde.

#### Ventajas:

- Baja la fricción para nuevos usuarios.
- Útil para aplicaciones que ofrecen contenido sin compromiso inicial.

#### Desventajas:

- Los datos del usuario no se transfieren si pierden el dispositivo antes de registrarse.
- Limitado a funcionalidades temporales.

#### Consideraciones:

- Ideal para pruebas o periodos de evaluación.
- Implementar las opciones para registrar e inicio de sesión para hacer las cuentas permanentes.

## **Autenticación personalizada:**

Este tipo de autenticación permite que si la aplicación tiene su propio sistema de autenticación o se conecta con otros servicios que emiten tokens de identidad, firebase permite autenticar a los usuarios con tokens personalizados.

### Características:

- Los tokens personalizados permiten autenticar usuarios a través de sistemas de autenticación propios o externos.

### Ventajas:

- Total, control sobre la autenticación.
- Compatible con sistemas preexistentes o más avanzados.

### Desventajas:

- Complejo de implementar y mantener.
- Requiere infraestructura backend para generar tokens.

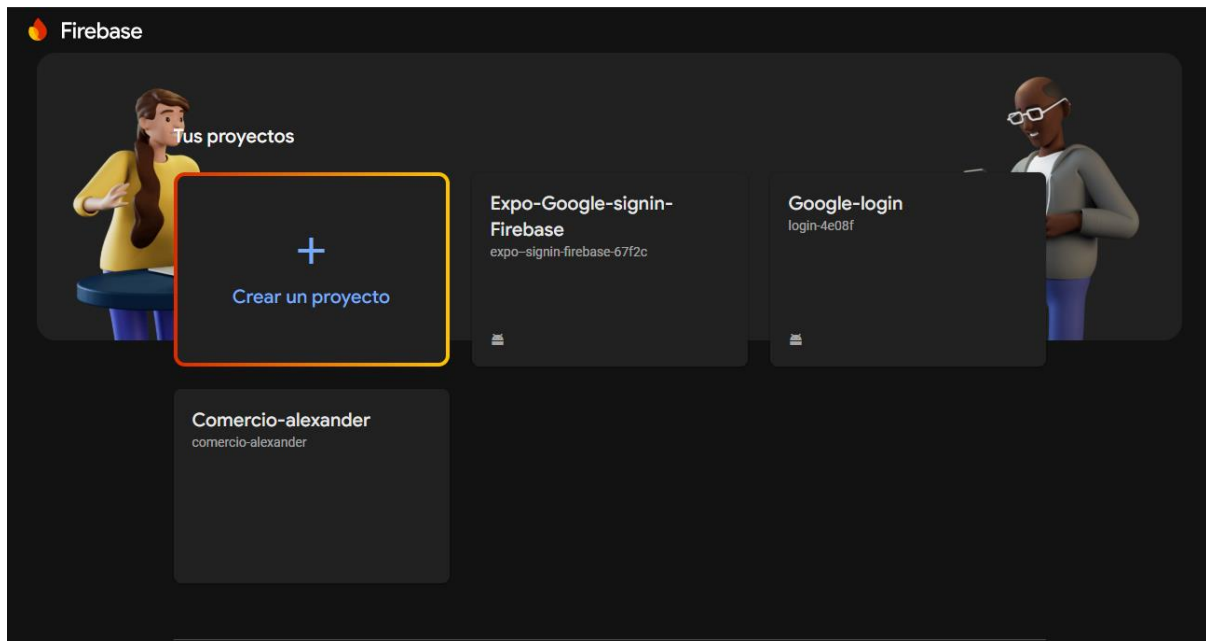
### Consideraciones:

- Útil para integraciones corporativas o escenarios complejos.
- Asegurar de generar tokens de forma segura.

# Desarrollo de creación de autenticación en react native.

## Autenticación Social Google

Como primer paso tenemos que acceder a Firebase y crear un proyecto para almacenar los usuarios y los datos de estos.



Como segundo paso tenemos que crear el proyecto de react native con el comando de `npx create-expo-app` con el nombre del proyecto para la creación de este y podamos realizar el inicio de sesión y la autenticación social en este caso de google.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Eduardo> cd desktop
PS C:\Users\Eduardo\desktop> npx create-expo-app@latest expo-google-signin-firebase
```

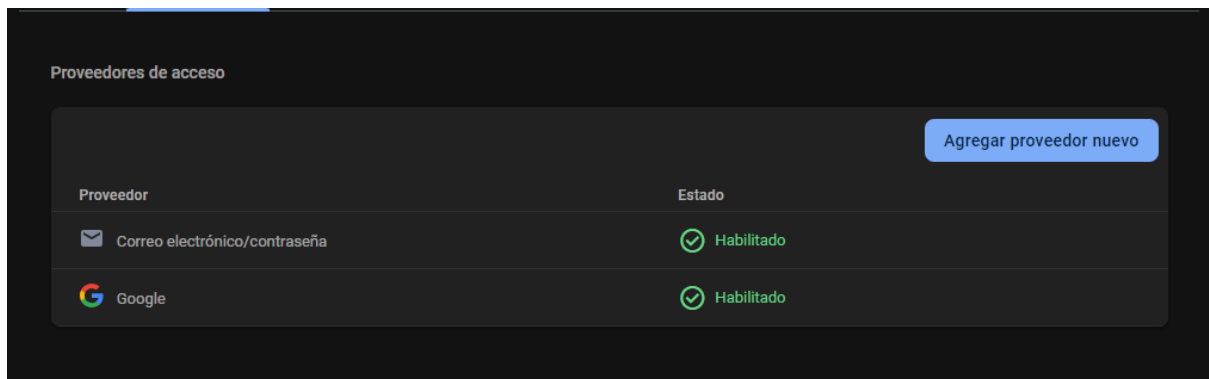
Como tercer paso tenemos que instalar las diferentes dependencias que usaremos para la realización de la autenticación social los cuales en esta ocasión serán las

dependencias de react native google singin y expo dev client para la realización de este proyecto.

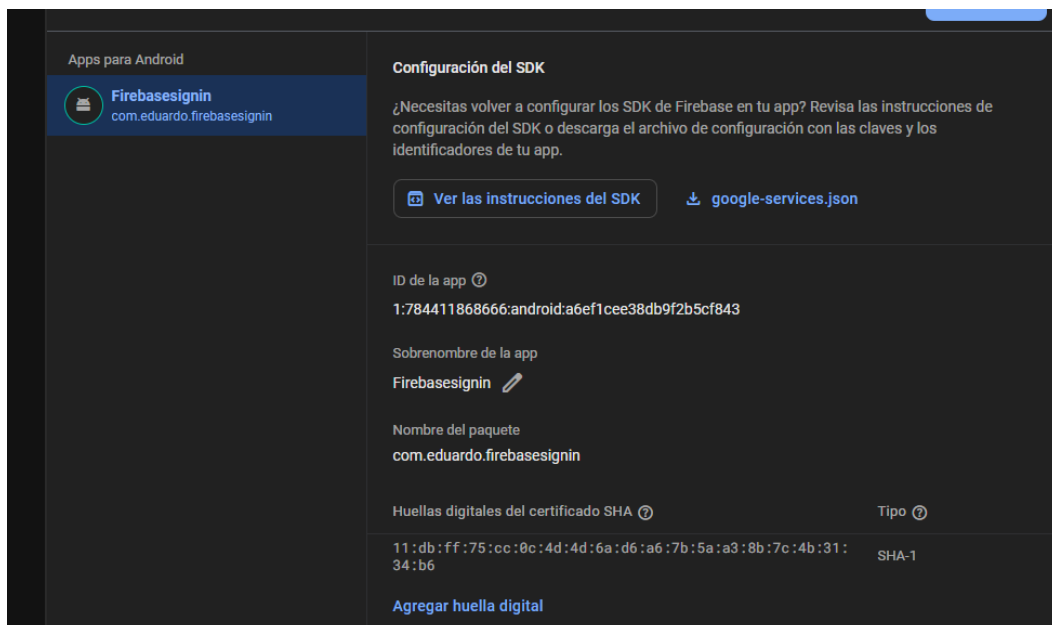
```
PS C:\Users\Eduardo\desktop> npx expo install expo-dev-client
```

```
PS C:\Users\Eduardo\desktop> npx expo install @react-native-google-signin/google-signin
```

Ademas configuraremos nuestro proyecto de Firebase para poder autentificar atreves de Google, para almacenar los usuarios y los datos de estos mismos.



Tambien instalaremos y usaremos eas para la creación de una keystore para la conectividad con firebase en el proyecto y configuraremos esta llave de modo SHA 1 el cual es necesario en Firebase.



Después de instalar las diferentes dependencias comenzaremos a realizar las configuraciones en el proyecto de react native en los cuales realizaremos un pre:build para las configuraciones predeterminadas y añadiremos a nuestro app.json las configuraciones y los paquetes necesarios que ocuparemos.



```

1  export default{
2    "expo": {
3      "name": "expo-google-signin-firebase",
4      "slug": "expo-google-signin-firebase",
5      "version": "1.0.0",
6      "orientation": "portrait",
7      "icon": "./assets/images/icon.png",
8      "scheme": "myapp",
9      "userInterfaceStyle": "automatic",
10     "newArchEnabled": true,
11     "ios": {
12       "supportsTablet": true
13     },
14     "android": {
15       "adaptiveIcon": {
16         "foregroundImage": "./assets/images/adaptive-icon.png",
17         "backgroundColor": "#ffffff"
18       },
19       "package": "com.eduardo.firebasesignin",
20       "googleServicesFile": process.env.GOOGLE_SERVICES_JSON
21     },
22     "web": {
23       "bundler": "metro",
24       "output": "static",
25       "favicon": "./assets/images/favicon.png"
26     },
27     "plugins": [
28       "expo-router",
29       [
30         "expo-splash-screen",
31         {
32           "image": "./assets/images/splash-icon.png",
33           "imageWidth": 200,

```

Al terminar de instalar las diferentes dependencias y hacer las configuraciones necesarias para el proyecto comenzaremos a realizar el código para la autenticación social.

Comenzaremos con el código en nuestro index.txs el cual será nuestro inicio de sesión y donde se realizará la autenticación.

```

1  import { StatusBar } from "expo-status-bar";
2  import { Button, StyleSheet, Text, View } from "react-native";
3  import {
4    GoogleSignin,
5    GoogleSigninButton,
6  } from "@react-native-google-signin/google-signin";
7  import { useEffect, useState } from "react";
8
9  export default function index() {
10   const [error, setError] = useState();
11   const [userInfo, setUserInfo] = useState();
12
13   useEffect(() => {
14     GoogleSignin.configure({
15       webClientId:
16         "784411868666-mr1pc1kmk3p541c6i1j8f0h2krolq1bd.apps.googleusercontent.com",
17     });
18   }, []);
19
20   const signin = async () => {
21     try {
22       await GoogleSignin.hasPlayServices();
23       const user = await GoogleSignin.signIn();
24       setUserInfo(user);
25       setError();
26     } catch (e) {
27       setError(e);
28     }
29   };
30
31   const logout = () => {
32     setUserInfo();
33     GoogleSignin.revokeAccess();
34     GoogleSignin.signOut();

```

```

    GoogleSignin.signOut();
  };

  return (
    <View style={styles.container}>
      <Text>{JSON.stringify(error)}</Text>
      {userInfo && <Text> Bienvenidos a la aplicacion {JSON.stringify(userInfo.user)}</Text>}
      {userInfo ? (
        <Button title="Logout" onPress={logout} />
      ) : (
        <GoogleSigninButton
          size={GoogleSigninButton.Size.Standard}
          color={GoogleSigninButton.Color.Dark}
          onPress={signin}
        />
      )}
      <StatusBar style="auto" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
});

```

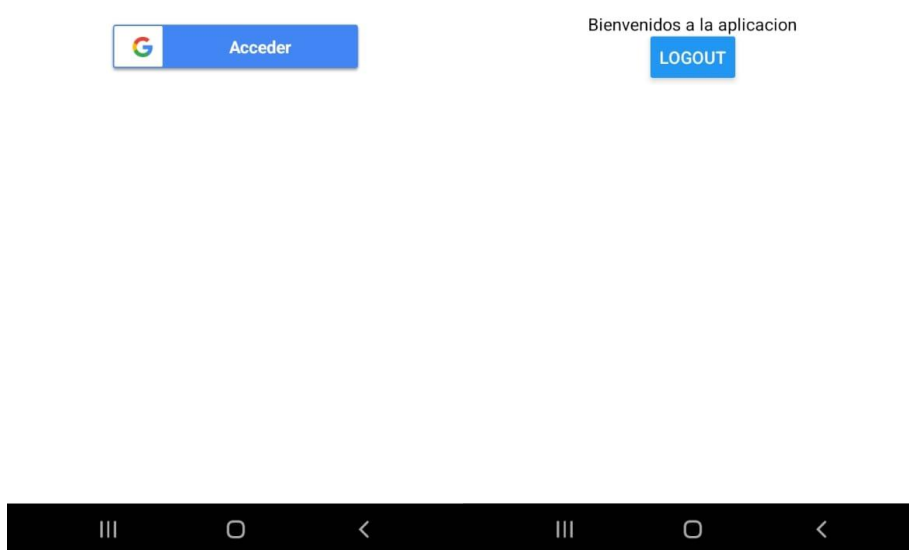
Para probar si el código funciona bien se tiene que poner el comando de npm expo run en el cual nos ayudara a levantar el proyecto para probarlo.

```
> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
> shift+m | more tools
> Press o | open project code in your editor

> Press ? | show all commands

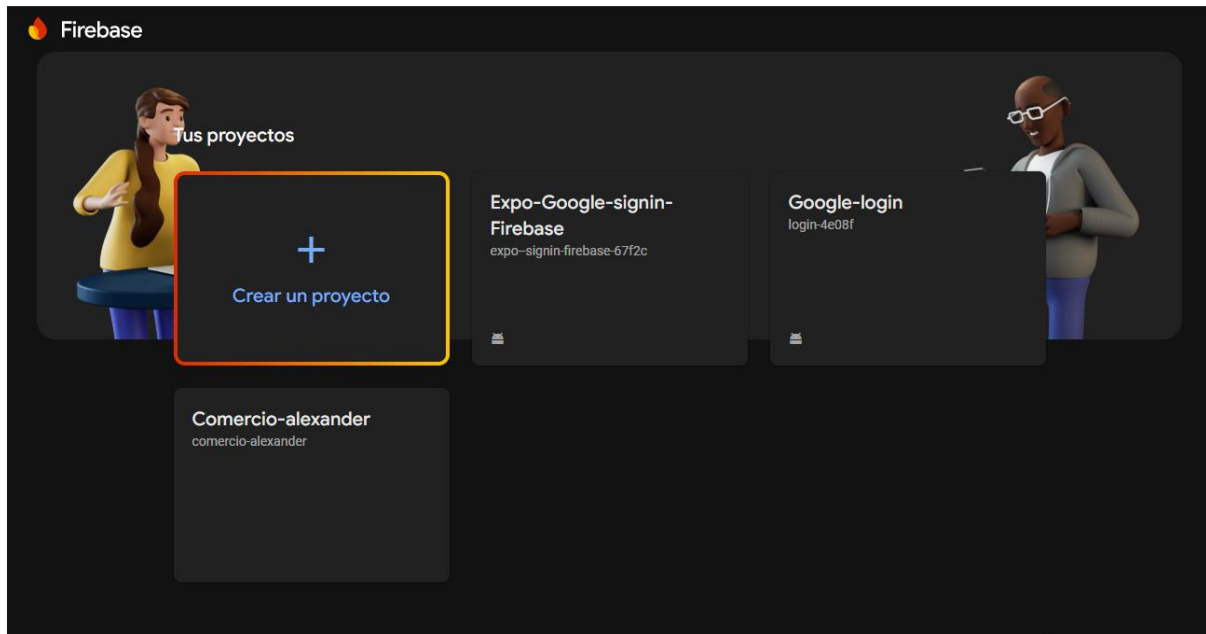
Logs for your project will appear below. Press Ctrl+C to exit.
> Open in the web browser...
> Press ? | show all commands
λ Bundled 5561ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\node\render.js (909 modules)
Android Bundled 7598ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (1249 modules)
Web Bundled 7637ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (905 modules)
Web Bundled 772ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (895 modules)
LOG [web] Logs will appear in the browser console
(NOBRIDGE) LOG Bridgeless mode is enabled
INFO 
⚠ JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (requires Google Chrome or Microsoft Edge).
Android Bundled 643ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (1 module)
(NOBRIDGE) LOG Bridgeless mode is enabled
INFO 
⚠ JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (requires Google Chrome or Microsoft Edge).
```

El proyecto se veria de la siguiente manera:



## Autenticación de correo electrónico y contraseña.

Como primer paso tenemos que acceder a Firebase y crear un proyecto para almacenar los usuarios y los datos de estos.



Como segundo paso tenemos que crear el proyecto de react native con el comando de npx create-expo-app con el nombre del proyecto para la creación de este y podamos realizar el inicio de sesión y la autenticación por correo electrónico y contraseña.

```
PS C:\Users\Eduardo\Desktop> npx create-expo-app@latest expo-signin-firebase
Creating an Expo project using the default template.

To choose from all available templates pass in the --template arg:
  $ npx create-expo-app --template

To choose from all available examples pass in the --example arg:
  $ npx create-expo-app --example

✔ Downloaded and extracted project files.
> npm install
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested
way to coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated @babel/plugin-proposal-class-properties@7.18.6: This proposal has been merged to the ECMAScript standard and thus this plugin is
no longer maintained. Please use @babel/plugin-transform-class-properties instead.
npm warn deprecated @babel/plugin-proposal-nullish-coalescing-operator@7.18.6: This proposal has been merged to the ECMAScript standard and thus thi
s plugin is no longer maintained. Please use @babel/plugin-transform-nullish-coalescing-operator instead.
npm warn deprecated rimraf@2.6.3: Rimraf versions prior to v4 are no longer supported
npm warn deprecated @babel/plugin-proposal-optional-chaining@7.21.0: This proposal has been merged to the ECMAScript standard and thus this plugin i
s no longer maintained. Please use @babel/plugin-transform-optional-chaining instead.
npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated abab@2.0.6: Use your platform's native atob() and btoa() methods instead
npm warn deprecated domexception@4.0.0: Use your platform's native DOMException instead
npm warn deprecated @xmldom/xmldom@0.7.13: this version is no longer supported, please update to at least 0.8.*
added 1179 packages, and audited 1180 packages in 2m
104 packages are looking for funding
```

Como tercer paso tenemos que instalar las diferentes dependencias que usaremos para la realización de la autenticación social los cuales en esta ocasión serán las

dependencias de react native firebase tanto app como auth de estas dependencias para la realización de este proyecto.

```
PS C:\Users\Eduardo\Desktop\expo-signin-firebase> code .
PS C:\Users\Eduardo\Desktop\expo-signin-firebase> npm install firebase @react-native-firebase/app @react-native-firebase/auth
added 116 packages, and audited 1296 packages in 57s

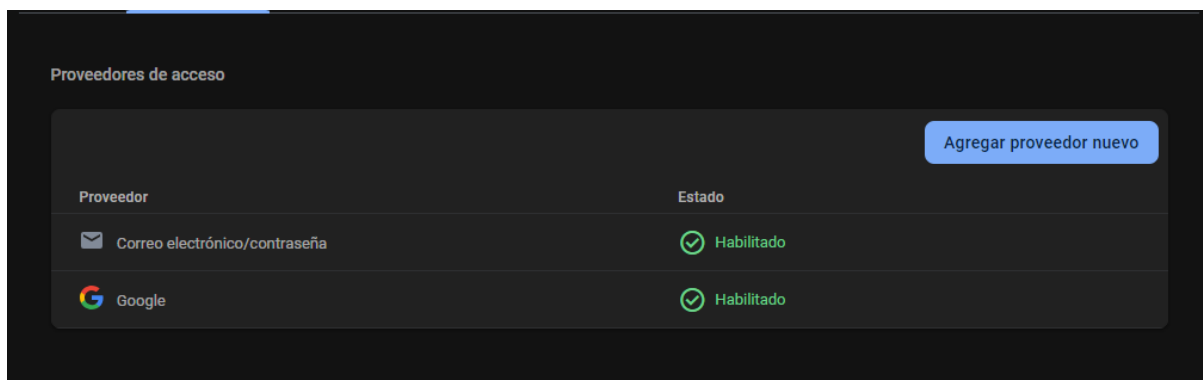
104 packages are looking for funding
  run 'npm fund' for details

5 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
```

Además configuraremos nuestro proyecto de Firebase para poder autenticar a través de correo electrónico y contraseña, para almacenar los usuarios y los datos de estos mismos.



Al terminar de instalar las diferentes dependencias y hacer las configuraciones necesarias para el proyecto comenzaremos a realizar el código para la autenticación social.

Comenzaremos con el código en nuestro index.tsx el cual será nuestro inicio de sesión y donde se realizará la autenticación.

```

1 import React, { useState, useEffect } from 'react';
2 import { View, Text, TextInput, Button, StyleSheet, ScrollView } from 'react-native';
3 import { initializeApp } from '@firebase/app';
4 import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword, onAuthStateChanged, signInWithCredential } from '@firebase/auth';
5
6 const firebaseConfig = {
7   apiKey: "AIzaSyB99W79TDh6aqA4ABzXiugM63oGVALz8Vs",
8   authDomain: "expo-signin-firebase-44fcd.firebaseio.com",
9   projectId: "expo-signin-firebase-44fcd",
10  storageBucket: "expo-signin-firebase-44fcd.firebaseio.com",
11  messagingSenderId: "308885494427",
12  appId: "1:308885494427:web:4bc76e0a2805677c78c43e"
13 };
14
15 const app = initializeApp(firebaseConfig);
16
17 const AuthScreen = ({ email, setEmail, password, setPassword, isLogin, setIsLogin, handleAuthentication }) => {
18   return (
19     <View style={styles.authContainer}>
20       <Text style={styles.title}>{isLogin ? 'Sign In' : 'Sign Up'}</Text>
21
22       <TextInput
23         style={styles.input}
24         value={email}
25         onChangeText={setEmail}
26         placeholder="Email"
27         autoCapitalize="none"
28       />
29
30       <TextInput
31         style={styles.input}
32         value={password}
33         onChangeText={setPassword}
34         placeholder="Password"

```

```

35       />
36     <View style={styles.buttonContainer}>
37       <Button title={isLogin ? 'Sign In' : 'Sign Up'} onPress={handleAuthentication} color="#3498db" />
38     </View>
39
40     <View style={styles.bottomContainer}>
41       <Text style={styles.toggleText} onPress={() => setIsLogin(!isLogin)}>
42         {isLogin ? 'Need an account? Sign Up' : 'Already have an account? Sign In'}
43       </Text>
44     </View>
45   </View>
46 );
47 }
48
49
50 const AuthenticatedScreen = ({ user, handleAuthentication }) => {
51   return (
52     <View style={styles.authContainer}>
53       <Text style={styles.title}>Welcome</Text>
54       <Text style={styles.emailText}>{user.email}</Text>
55       <Button title="Logout" onPress={handleAuthentication} color="#e74c3c" />
56     </View>
57   );
58 };
59
60 export default App = () => {
61   const [email, setEmail] = useState('');
62   const [password, setPassword] = useState('');
63   const [user, setUser] = useState(null); // Track user authentication state
64   const [isLogin, setIsLogin] = useState(true);
65
66   const auth = getAuth(app);

```

```

67     const unsubscribe = onAuthStateChanged(auth, (user) => {
68         setUser(user);
69     });
70
71     return () => unsubscribe();
72 }, [auth]);
73
74
75 const handleAuthentication = async () => {
76     try {
77         if (user) {
78             // If user is already authenticated, log out
79             console.log('User logged out successfully!');
80             await signOut(auth);
81         } else {
82             // Sign in or sign up
83             if (isLogin) {
84                 // Sign in
85                 await signInWithEmailAndPassword(auth, email, password);
86                 console.log('User signed in successfully!');
87             } else {
88                 // Sign up
89                 await createUserWithEmailAndPassword(auth, email, password);
90                 console.log('User created successfully!');
91             }
92         }
93     } catch (error) {
94         console.error('Authentication error:', error.message);
95     }
96 };
97
98

```

```

98     return (
99         <ScrollView contentContainerStyle={styles.container}>
100             {user ? (
101                 // Show user's email if user is authenticated
102                 <AuthenticatedScreen user={user} handleAuthentication={handleAuthentication} />
103             ) : (
104                 // Show sign-in or sign-up form if user is not authenticated
105                 <AuthScreen
106                     email={email}
107                     setEmail={setEmail}
108                     password={password}
109                     setPassword={setPassword}
110                     isLogin={isLogin}
111                     setIsLogin={setIsLogin}
112                     handleAuthentication={handleAuthentication}
113                 />
114             )}
115         </ScrollView>
116     );
117 }
118
119 const styles = StyleSheet.create({
120     container: {
121         flexGrow: 1,
122         justifyContent: 'center',
123         alignItems: 'center',
124         padding: 16,
125         backgroundColor: '#f0f0f0',
126     },
127     authContainer: {
128         width: '80%',
129         maxWidth: 400,
130         backgroundColor: 'fff',
131     },
132 });

```

```
130 padding: 16,  
131 borderRadius: 8,  
132 elevation: 3,  
133 },  
134 title: {  
135   fontSize: 24,  
136   marginBottom: 16,  
137   textAlign: 'center',  
138 },  
139 input: {  
140   height: 40,  
141   borderColor: '#ddd',  
142   borderWidth: 1,  
143   marginBottom: 16,  
144   padding: 8,  
145   borderRadius: 4,  
146 },  
147 buttonContainer: {  
148   marginBottom: 16,  
149 },  
150 toggleText: {  
151   color: '#3498db',  
152   textAlign: 'center',  
153 },  
154 bottomContainer: {  
155   marginTop: 20,  
156 },  
157 emailText: {  
158   fontSize: 18,  
159   textAlign: 'center',  
160   marginBottom: 20,  
161 }
```

Para probar si el código funciona bien se tiene que poner el comando de npm expo run en el cual nos ayudara a levantar el proyecto para probarlo.



```

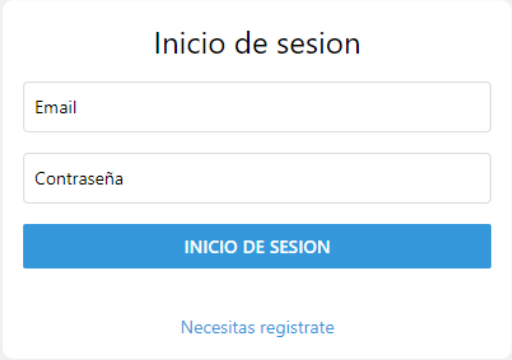
> Press j | open debugger
> Press r | reload app
> Press m | toggle menu
> shift+m | more tools
> Press o | open project code in your editor

> Press ? | show all commands

Logs for your project will appear below. Press Ctrl+C to exit.
> Open in the web browser...
> Press ? | show all commands
λ Bundled 5561ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\node\render.js (909 modules)
Android Bundled 7598ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (1249 modules)
Web Bundled 7637ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (905 modules)
Web Bundled 772ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (895 modules)
LOG [web] Logs will appear in the browser console
(NOBRIDGE) LOG Bridgeless mode is enabled
INFO
⚠ JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (requires Google Chrome or Microsoft Edge).
Android Bundled 643ms C:\Users\Eduardo\Desktop\expo-google-signin-firebase\node_modules\expo-router\entry.js (1 module)
(NOBRIDGE) LOG Bridgeless mode is enabled
INFO
⚠ JavaScript logs will be removed from Metro in React Native 0.77! Please use React Native DevTools as your default tool. Tip: Type j in the terminal to open (requires Google Chrome or Microsoft Edge).

```

El proyecto se veria de la siguiente manera:



Inicio de sesion

Email

Contraseña

INICIO DE SESION

[Necesitas registrate](#)

Registro

Email

Contraseña

REGISTRO

Ya tiene cuenta!!

Welcome

derodriguezvigil@gmail.com

LOGOUT

Link del repositorio de autenticación de social Google:

<https://github.com/eduardovigil/React-native-signin>

Link del repositorio de autenticación de correo electrónico y contraseña:

<https://github.com/eduardovigil/React-signin-firebase>