Mi Comida Favorita

Aplicación React Native con Firebase Authentication y Cloud Firestore

Requerimientos Previos

- Node.js (versión 14 o superior)
- npm o yarn
- Expo CLI
- IDE (Visual Studio Code recomendado)
- Cuenta de Firebase
- Emulador Android/iOS o dispositivo físico

Librerías Necesarias

```
# Dependencias principales
expo-cli
@react-navigation/native
@react-navigation/native-stack
firebase
react-native-elements
expo-constants
```

1. Configuración Inicial

1.1 Crear proyecto Expo

```
# Crear nuevo proyecto
npx create-expo-app MiComidaFavorita --template blank
cd MiComidaFavorita

# Instalar dependencias
npm install @react-navigation/native @react-navigation/native-stack
npm install firebase
npm install react-native-elements
npm install expo-constants
npm install react-native-safe-area-context
```

1.2 Configurar Firebase

1. Ir a Firebase Console (https://console.firebase.google.com/)
2. Crear nuevo proyecto "MiComidaFavorita"
3. Habilitar Authentication (Email/Password)

4. Crear Cloud Firestore
5. Registrar la aplicación web
6. Copiar configuración de Firebase

1.3 Estructura del Proyecto

```
MiComidaFavorita/
├── src/
│   ├── config/
│   │   └── firebase.js
│   ├── screens/
│   │   ├── LoginScreen.js
│   │   ├── RegisterScreen.js
│   │   └── HomeScreen.js
│   ├── components/
│   │   └── ProfileForm.js
│   └── navigation/
│       └── AppNavigator.js
├── App.js
└── package.json
```

2. Configuración de Firebase

2.1 src/config/firebase.js

```
import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';
import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {
  // Pegar configuración de Firebase Console
  apiKey: "tu-api-key",
  authDomain: "tu-auth-domain",
  projectId: "tu-project-id",
  storageBucket: "tu-storage-bucket",
  messagingSenderId: "tu-sender-id",
  appId: "tu-app-id"
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);
```

3. Navegación

3.1 src/navigation/AppNavigator.js

```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
```

```
import LoginScreen from '../screens/LoginScreen';
import RegisterScreen from '../screens/RegisterScreen';
import HomeScreen from '../screens/HomeScreen';

const Stack = createNativeStackNavigator();

export default function AppNavigator() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Login">
        <Stack.Screen name="Login" component={LoginScreen} options={{ headerShown: false
}} />
        <Stack.Screen name="Register" component={RegisterScreen} options={{ headerShown:
false }} />
        <Stack.Screen name="Home" component={HomeScreen} options={{ headerShown: true,
title: 'Mi Perfil' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

4. Pantallas

4.1 src/screens/LoginScreen.js

```
import React, { useState } from 'react';
import { View, StyleSheet } from 'react-native';
import { Input, Button, Text } from 'react-native-elements';
import { signInWithEmailAndPassword } from 'firebase/auth';
import { auth } from '../config/firebase';

export default function LoginScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleLogin = async () => {
    try {
      const userCredential = await signInWithEmailAndPassword(auth, email, password);
      navigation.replace('Home');
    } catch (error) {
      setError('Error al iniciar sesión: ' + error.message);
    }
  };

  return (
    <View style={styles.container}>
      <Text h3 style={styles.title}>Mi Comida Favorita</Text>
      <Input
        placeholder="Email"
        value={email}
```

```
        onChangeText={setEmail}
        autoCapitalize="none"
      />
      <Input
        placeholder="Contraseña"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
      />
      {error ? <Text style={styles.error}>{error}</Text> : null}
      <Button
        title="Iniciar Sesión"
        onPress={handleLogin}
        containerStyle={styles.button}
      />
      <Button
        title="Registrarse"
        type="outline"
        onPress={() => navigation.navigate('Register')}
        containerStyle={styles.button}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    justifyContent: 'center',
  },
  title: {
    textAlign: 'center',
    marginBottom: 30,
  },
  button: {
    marginVertical: 10,
  },
  error: {
    color: 'red',
    textAlign: 'center',
    marginBottom: 10,
  },
});
```

4.2 src/screens/RegisterScreen.js

```
import React, { useState } from 'react';
import { View, StyleSheet } from 'react-native';
import { Input, Button, Text } from 'react-native-elements';
import { createUserWithEmailAndPassword } from 'firebase/auth';
```

```
import { auth } from '../config/firebase';

export default function RegisterScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleRegister = async () => {
    try {
      const userCredential = await createUserWithEmailAndPassword(auth, email, password);
      navigation.replace('Home');
    } catch (error) {
      setError('Error al registrarse: ' + error.message);
    }
  };

  return (
    <View style={styles.container}>
      <Text h3 style={styles.title}>Registro</Text>
      <Input
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        autoCapitalize="none"
      />
      <Input
        placeholder="Contraseña"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
      />
      {error ? <Text style={styles.error}>{error}</Text> : null}
      <Button
        title="Registrarse"
        onPress={handleRegister}
        containerStyle={styles.button}
      />
      <Button
        title="Volver al Login"
        type="outline"
        onPress={() => navigation.navigate('Login')}
        containerStyle={styles.button}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    justifyContent: 'center',
```

```
  },
  title: {
    textAlign: 'center',
    marginBottom: 30,
  },
  button: {
    marginVertical: 10,
  },
  error: {
    color: 'red',
    textAlign: 'center',
    marginBottom: 10,
  },
});
```

4.3 src/screens/HomeScreen.js

```
import React, { useState, useEffect } from 'react';
import { View, StyleSheet } from 'react-native';
import { Input, Button, Text } from 'react-native-elements';
import { auth } from '../config/firebase';
import { signOut } from 'firebase/auth';
import { db } from '../config/firebase';
import { doc, getDoc, setDoc } from 'firebase/firestore';

export default function HomeScreen({ navigation }) {
  const [profile, setProfile] = useState({
    nombre: '',
    apellido: '',
    comidaFavorita: ''
  });

  useEffect(() => {
    loadProfile();
  }, []);

  const loadProfile = async () => {
    try {
      const docRef = doc(db, 'usuarios', auth.currentUser.uid);
      const docSnap = await getDoc(docRef);

      if (docSnap.exists()) {
        setProfile(docSnap.data());
      }
    } catch (error) {
      console.error('Error al cargar perfil:', error);
    }
  };

  const handleUpdate = async () => {
    try {
```

```
      await setDoc(doc(db, 'usuarios', auth.currentUser.uid), profile);
      alert('Perfil actualizado exitosamente');
    } catch (error) {
      console.error('Error al actualizar perfil:', error);
      alert('Error al actualizar perfil');
    }
  };

  const handleSignOut = async () => {
    try {
      await signOut(auth);
      navigation.replace('Login');
    } catch (error) {
      console.error('Error al cerrar sesión:', error);
    }
  };

  return (
    <View style={styles.container}>
      <Text h4 style={styles.title}>Mi Perfil</Text>
      <Input
        placeholder="Nombre"
        value={profile.nombre}
        onChangeText={(text) => setProfile({...profile, nombre: text})}
      />
      <Input
        placeholder="Apellido"
        value={profile.apellido}
        onChangeText={(text) => setProfile({...profile, apellido: text})}
      />
      <Input
        placeholder="Comida Favorita"
        value={profile.comidaFavorita}
        onChangeText={(text) => setProfile({...profile, comidaFavorita: text})}
      />
      <Button
        title="Actualizar Perfil"
        onPress={handleUpdate}
        containerStyle={styles.button}
      />
      <Button
        title="Cerrar Sesión"
        type="outline"
        onPress={handleSignOut}
        containerStyle={styles.button}
      />
    </View>
  );
}

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

```
    flex: 1,
    padding: 20,
  },
  title: {
    textAlign: 'center',
    marginBottom: 30,
  },
  button: {
    marginVertical: 10,
  },
});
```

5. Componente Principal

5.1 App.js

```
import React from 'react';
import { SafeAreaProvider } from 'react-native-safe-area-context';
import AppNavigator from './src/navigation/AppNavigator';

export default function App() {
  return (
    <SafeAreaProvider>
      <AppNavigator />
    </SafeAreaProvider>
  );
}
```

6. Pruebas y Ejecución

1. Asegurarse de que todas las dependencias estén instaladas:

```
npm install
```

2. Iniciar la aplicación:

```
npx expo start
```

3. Probar en emulador o dispositivo físico:
    - Presionar 'a' para Android
    - Presionar 'i' para iOS
    - Escanear código QR con Expo Go en dispositivo físico

7. Funcionalidades Implementadas

- Registro de usuarios con email y contraseña
- Inicio de sesión

- Persistencia de sesión
- Formulario de perfil con datos personales
- Almacenamiento en Firestore
- Cerrar sesión

8. Mejoras Posibles

1. Validación de formularios
2. Manejo de errores más detallado
3. Indicadores de carga
4. Diseño más elaborado
5. Foto de perfil
6. Implementar más proveedores de autenticación

9. Debugging Común

1. **Error: "Firebase App not initialized"**
    - Verificar firebaseConfig en firebase.js
    - Asegurar que las credenciales sean correctas
2. **Error: "Authentication failed"**
    - Verificar que Authentication esté habilitado en Firebase Console
    - Comprobar que el método email/password esté activado
3. **Error: "Permission denied" en Firestore**
    - Revisar reglas de seguridad en Firestore
    - Verificar que el usuario esté autenticado

10. Reglas de Seguridad Firestore

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /usuarios/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
  }
}
```