Tarea: Mejoras al Proyecto "Mi Comida Favorita"

Módulo 5 - Backend as a Service (BaaS)

Descripción General

En esta tarea mejoraremos la aplicación "Mi Comida Favorita" desarrollada en clase, implementando validaciones, indicadores de carga y funcionalidades adicionales.

Objetivos de Aprendizaje

- Implementar validaciones de formularios en React Native
- Manejar estados de carga en la interfaz de usuario
- Mejorar la experiencia de usuario con feedback visual
- Trabajar con almacenamiento de archivos en Firebase (opcional)

Requerimientos Funcionales

1. Mejoras en el Formulario de Registro

Requisitos Obligatorios:

- Implementar validaciones para todos los campos:
  - Email válido (formato correcto)
  - Contraseña (mínimo 8 caracteres)
  - Debe contener al menos:
    - Una letra mayúscula
    - Una letra minúscula
    - Un número
    - Un carácter especial (!@#$%^&*)
- Agregar campo de confirmación de contraseña
- Validar que ambas contraseñas coincidan
- Mostrar mensajes de error específicos para cada validación
- No permitir el envío del formulario si hay errores

Ejemplo de Implementación:

```
const validatePassword = (password) => {
  const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[!@#$%^&*])[A-Za-z\d!@#$%^&*]
{8,}$/;
  return passwordRegex.test(password);
```

```
};

const validateForm = () => {
  let errors = {};

  if (!email) errors.email = 'El email es requerido';
  else if (!/\S+@\S+\.\S+/.test(email)) errors.email = 'Email inválido';

  if (!password) errors.password = 'La contraseña es requerida';
  else if (!validatePassword(password)) {
    errors.password = 'La contraseña debe tener mínimo 8 caracteres, una mayúscula, una
minúscula, un número y un carácter especial';
  }

  if (password !== confirmPassword) {
    errors.confirmPassword = 'Las contraseñas no coinciden';
  }

  return errors;
};
```

2. Mejoras en el Formulario de Login

Requisitos Obligatorios:

- Validar formato de email
- Validar que la contraseña no esté vacía
- Mostrar mensajes de error específicos
- Deshabilitar el botón de login mientras los campos sean inválidos

Ejemplo de Implementación:

```
const validateLoginForm = () => {
  const isEmailValid = /\S+@\S+\.\S+/.test(email);
  const isPasswordValid = password.length > 0;
  return isEmailValid && isPasswordValid;
};

// En el botón de login:
<Button
  title="Iniciar Sesión"
  disabled={!validateLoginForm()}
  onPress={handleLogin}
/>
```

3. Implementación de Loading States

Requisitos Obligatorios:

- Mostrar indicador de carga durante:

    - Proceso de registro
    - Proceso de login
    - Carga de datos del perfil
    - Actualización de datos del perfil
- Deshabilitar botones durante la carga
- Implementar manejo de errores con feedback visual

Ejemplo de Implementación:

```
const [isLoading, setIsLoading] = useState(false);

const handleLogin = async () => {
  setIsLoading(true);
  try {
    await signInWithEmailAndPassword(auth, email, password);
    navigation.replace('Home');
  } catch (error) {
    setError(error.message);
  } finally {
    setIsLoading(false);
  }
};

// En el JSX:
{isLoading ? (
  <ActivityIndicator size="large" color="#0000ff" />
) : (
  <Button
    title="Iniciar Sesión"
    onPress={handleLogin}
    disabled={isLoading}
  />
)}
```

4. Foto de Perfil (Extra - Opcional)

Requisitos para Puntos Extra:

- Implementar selección de imagen desde la galería
- Implementar toma de foto desde la cámara
- Almacenar la imagen en Firebase Storage
- Mostrar la imagen en el perfil
- Permitir actualizar la imagen

Ejemplo de Implementación:

```
import * as ImagePicker from 'expo-image-picker';
```

```
import { ref, uploadBytes, getDownloadURL } from 'firebase/storage';
import { storage } from '../config/firebase';

const pickImage = async () => {
  const result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.Images,
    allowsEditing: true,
    aspect: [1, 1],
    quality: 0.5,
  });

  if (!result.canceled) {
    await uploadImage(result.assets[0].uri);
  }
};

const uploadImage = async (uri) => {
  const response = await fetch(uri);
  const blob = await response.blob();
  const storageRef = ref(storage, `profiles/${auth.currentUser.uid}`);

  await uploadBytes(storageRef, blob);
  const url = await getDownloadURL(storageRef);

  await updateDoc(doc(db, 'usuarios', auth.currentUser.uid), {
    photoURL: url
  });
};
```

Entregables

1. Código fuente del proyecto en un repositorio Git
2. README con:
    - Instrucciones de instalación
    - Descripción de mejoras implementadas
    - Screenshots de la aplicación
3. Video corto demostrando la funcionalidad

Fecha de Entrega

- Fecha límite: Lunes 20 de Enero, 6pm
- Presentación en clase: Lunes 20 de Enero, 7pm

Recursos Útiles

- Documentación de React Native
- Firebase Storage Documentation

- Expo ImagePicker Documentation
- React Native Elements

Notas Adicionales

- Se valorará la limpieza y organización del código
- Las validaciones deben ser en tiempo real
- La interfaz debe ser responsiva y amigable
- Se deben manejar todos los posibles errores
- El código debe estar comentado apropiadamente