UNIVERSIDAD DON BOSCO



***“Taller 2: Segundo Desafío Practico”***

**Materia:** DPS104 G01L

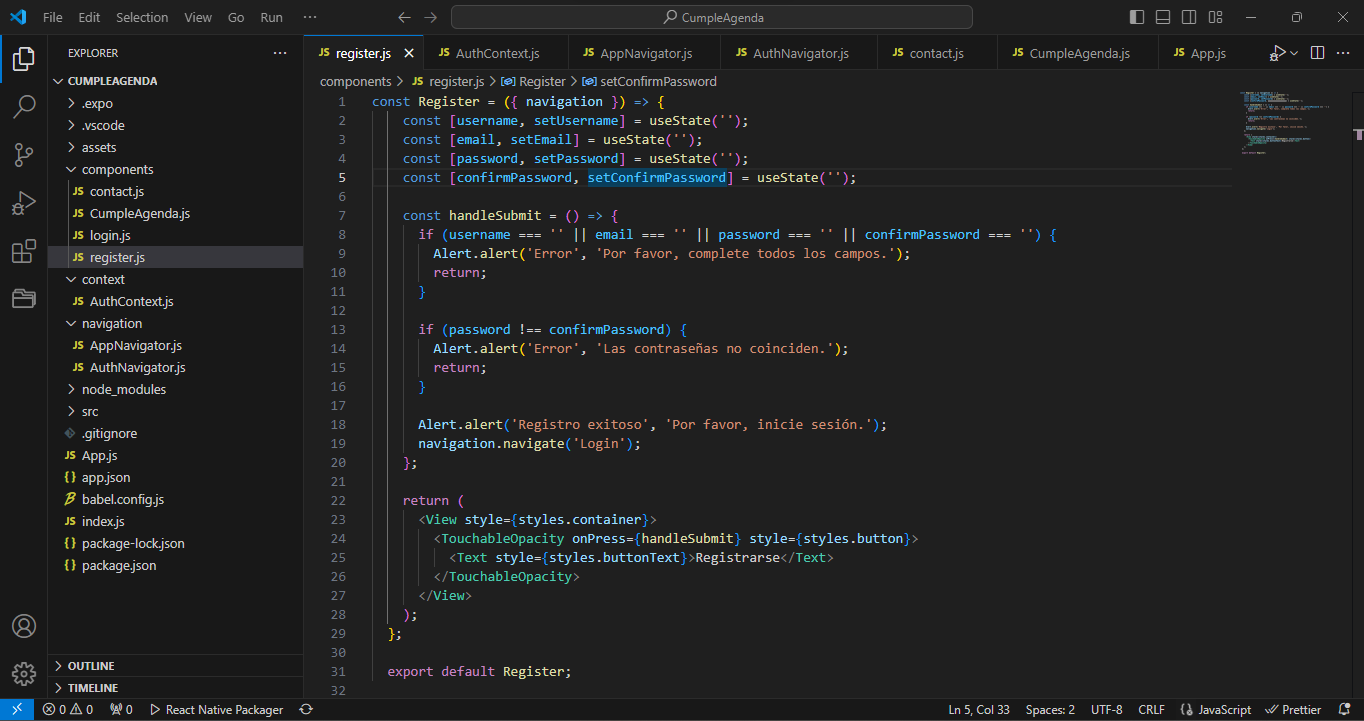
**Integrantes:** Cristian Enrique Pineda Muñoz

**Número de Carnet:** PM190654

**Docente:** Karens Medrano

**Ejercicio 01: CumpleAgenda (35% completado)**

**https://github.com/Cris0024/CumpleAgenda.git**

****

A continuación, se mostrará el script de todos los componentes, archivos de navegación, etc.

**Inicio Sesión**

Los compontes que forman parte de la ventana tanto para iniciar sesión como para registrarse son login.js y register.js

**const Login = ({ setUser, navigation }) => {**

**const [email, setEmail] = useState('');**

**const [password, setPassword] = useState('');**

**const handleSubmit = () => {**

**if (email === '' || password === '') {**

**Alert.alert('Error', 'Por favor, ingrese tanto el correo electrónico como la contraseña.');**

**return;**

**}**

**setUser({ email });**

**Alert.alert('Éxito', 'Inicio de sesión exitoso');**

**};**

**return (**

**<View style={styles.container}>**

**<Text style={styles.label}>Correo Electrónico:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese su correo electrónico"**

**value={email}**

**onChangeText={setEmail}**

**keyboardType="email-address"**

**/>**

**<Text style={styles.label}>Contraseña:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese su contraseña"**

**value={password}**

**onChangeText={setPassword}**

**secureTextEntry**

**/>**

**<TouchableOpacity onPress={handleSubmit} style={styles.button}>**

**<Text style={styles.buttonText}>Iniciar Sesión</Text>**

**</TouchableOpacity>**

**<TouchableOpacity onPress={() => navigation.navigate('Register')} style={styles.registerButton}>**

**<Text style={styles.registerText}>¿No tienes una cuenta? Regístrate</Text>**

**</TouchableOpacity>**

**</View>**

**);**

**};**

**export default Login;**

register.js

**const Register = ({ navigation }) => {**

**const [username, setUsername] = useState('');**

**const [email, setEmail] = useState('');**

**const [password, setPassword] = useState('');**

**const [confirmPassword, setConfirmPassword] = useState('');**

**const handleSubmit = () => {**

**if (username === '' || email === '' || password === '' || confirmPassword === '') {**

**Alert.alert('Error', 'Por favor, complete todos los campos.');**

**return;**

**}**

**if (password !== confirmPassword) {**

**Alert.alert('Error', 'Las contraseñas no coinciden.');**

**return;**

**}**

**Alert.alert('Registro exitoso', 'Por favor, inicie sesión.');**

**navigation.navigate('Login');**

**};**

**return (**

**<View style={styles.container}>**

**<TouchableOpacity onPress={handleSubmit} style={styles.button}>**

**<Text style={styles.buttonText}>Registrarse</Text>**

**</TouchableOpacity>**

**</View>**

**);**

**};**

**export default Register;**

Otros archivos que están relacionados con el inicio de sesión son AuthContext.js ubicado en la carpeta “context”.

**import React, { createContext, useState } from 'react';**

**// Crear el contexto de autenticación**

**export const AuthContext = createContext();**

**const AuthProvider = ({ children }) => {**

**const [user, setUser] = useState(null);**

**// Función para iniciar sesión y establecer el usuario**

**const login = (userInfo) => {**

**setUser(userInfo);**

**};**

**// Función para cerrar sesión y eliminar el usuario**

**const logout = () => {**

**setUser(null);**

**};**

**return (**

**<AuthContext.Provider value={{ user, login, logout }}>**

**{children}**

**</AuthContext.Provider>**

**);**

**};**

**export default AuthProvider;**

y también archivos ubicados en la carpeta navigation denominados AuthNavigator.js

**import React from 'react';**

**import { NavigationContainer } from '@react-navigation/native';**

**import { createStackNavigator } from '@react-navigation/stack';**

**import login from '../components/login'; // Asegúrate de que la ruta sea correcta**

**import register from '../components/register'; // Asegúrate de que la ruta sea correcta**

**const Stack = createStackNavigator();**

**function AuthNavigator() {**

**return (**

**<NavigationContainer>**

**<Stack.Navigator initialRouteName="Login">**

**<Stack.Screen name="login" component={login} />**

**<Stack.Screen name="register" component={register} />**

**</Stack.Navigator>**

**</NavigationContainer>**

**);**

**}**

**export default AuthNavigator;**

y también está el archivo AppNavigator.js en donde se encuentra el Drawer y el contenido de la ventana principal.

**import React, { useContext } from 'react';**

**import { createDrawerNavigator } from '@react-navigation/drawer';**

**import { Button } from 'react-native';**

**import HomeScreen from '../screens/HomeScreen'; // Asegúrate de que la ruta sea correcta**

**import ProfileScreen from '../screens/ProfileScreen'; // Asegúrate de que la ruta sea correcta**

**import { AuthContext } from '../context/AuthContext'; // Manejo del contexto de autenticación**

**const Drawer = createDrawerNavigator();**

**function AppNavigator() {**

**const { logout } = useContext(AuthContext); // No es necesario pasar `user` aquí si no lo usas**

**return (**

**<Drawer.Navigator initialRouteName="Home">**

**<Drawer.Screen**

**name="Home"**

**component={HomeScreen}**

**options={{**

**headerRight: () => (**

**<Button onPress={logout} title="Cerrar Sesión" />**

**),**

**}}**

**/>**

**<Drawer.Screen name="Profile" component={ProfileScreen} />**

**</Drawer.Navigator>**

**);**

**}**

**export default AppNavigator;**

**Registro de Contactos**

El componente que se utiliza para guardar la información de una persona que se quiere recordar su cumpleaños es contact.js

**import React, { useState } from 'react';**

**import { Text, StyleSheet, View, TextInput, TouchableOpacity, Alert, DatePickerIOS, Platform, DatePickerAndroid, Button } from 'react-native';**

**const contact = ({ onSubmit }) => {**

**const [nombre, setNombre] = useState('');**

**const [apellido, setApellido] = useState('');**

**const [email, setEmail] = useState('');**

**const [telefono, setTelefono] = useState('');**

**const [fechaCumpleaños, setFechaCumpleaños] = useState(new Date());**

**const handleDateChange = (event, selectedDate) => {**

**const currentDate = selectedDate || fechaCumpleaños;**

**setFechaCumpleaños(currentDate);**

**};**

**const handleSubmit = () => {**

**if (nombre === '' || apellido === '' || email === '' || telefono === '') {**

**Alert.alert('Error', 'Por favor, complete todos los campos.');**

**return;**

**}**

**onSubmit({ nombre, apellido, email, telefono, fechaCumpleaños });**

**};**

**return (**

**<View style={styles.container}>**

**<Text style={styles.label}>Nombre:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el nombre"**

**value={nombre}**

**onChangeText={setNombre}**

**/>**

**<Text style={styles.label}>Apellido:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el apellido"**

**value={apellido}**

**onChangeText={setApellido}**

**/>**

**<Text style={styles.label}>Correo Electrónico:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el correo electrónico"**

**value={email}**

**onChangeText={setEmail}**

**keyboardType="email-address"**

**/>**

**<Text style={styles.label}>Número de Teléfono:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el número de teléfono"**

**value={telefono}**

**onChangeText={setTelefono}**

**keyboardType="phone-pad"**

**/>**

**<Text style={styles.label}>Fecha de Cumpleaños:</Text>**

**{Platform.OS === 'ios' ? (**

**<DatePickerIOS**

**date={fechaCumpleaños}**

**onDateChange={handleDateChange}**

**/>**

**) : (**

**<Button**

**title={fechaCumpleaños.toDateString()}**

**onPress={async () => {**

**try {**

**const { action, year, month, day } = await DatePickerAndroid.open({**

**date: fechaCumpleaños,**

**mode: 'date'**

**});**

**if (action !== DatePickerAndroid.dismissedAction) {**

**setFechaCumpleaños(new Date(year, month, day));**

**}**

**} catch (error) {**

**console.warn('Error al abrir el selector de fecha:', error);**

**}**

**}}**

**/>**

**)}**

**<TouchableOpacity onPress={handleSubmit} style={styles.button}>**

**<Text style={styles.buttonText}>Guardar Datos</Text>**

**</TouchableOpacity>**

**</View>**

**);**

**};**

**const styles = StyleSheet.create({**

**container: {**

**flex: 1,**

**justifyContent: 'center',**

**padding: 16,**

**backgroundColor: '#F5F5F5'**

**},**

**label: {**

**fontSize: 18,**

**marginVertical: 8,**

**fontWeight: 'bold'**

**},**

**input: {**

**height: 40,**

**borderColor: '#ddd',**

**borderWidth: 1,**

**borderRadius: 5,**

**paddingHorizontal: 8,**

**marginBottom: 16**

**},**

**button: {**

**backgroundColor: '#007BFF',**

**paddingVertical: 10,**

**borderRadius: 5**

**},**

**buttonText: {**

**color: '#FFF',**

**textAlign: 'center',**

**fontSize: 18**

**}**

**});**

**export default contact;**

**Recordatorio de Cumpleaños**

El componente contact.js se encargará de mostrar los recordatorios relacionados con los cumpleaños de las personas que fueron registradas.

**import React, { useState } from 'react';**

**import { Text, StyleSheet, View, TextInput, TouchableOpacity, Alert, DatePickerIOS, Platform, DatePickerAndroid, Button } from 'react-native';**

**const contact = ({ onSubmit }) => {**

**const [nombre, setNombre] = useState('');**

**const [apellido, setApellido] = useState('');**

**const [email, setEmail] = useState('');**

**const [telefono, setTelefono] = useState('');**

**const [fechaCumpleaños, setFechaCumpleaños] = useState(new Date());**

**const handleDateChange = (event, selectedDate) => {**

**const currentDate = selectedDate || fechaCumpleaños;**

**setFechaCumpleaños(currentDate);**

**};**

**const handleSubmit = () => {**

**if (nombre === '' || apellido === '' || email === '' || telefono === '') {**

**Alert.alert('Error', 'Por favor, complete todos los campos.');**

**return;**

**}**

**onSubmit({ nombre, apellido, email, telefono, fechaCumpleaños });**

**};**

**return (**

**<View style={styles.container}>**

**<Text style={styles.label}>Nombre:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el nombre"**

**value={nombre}**

**onChangeText={setNombre}**

**/>**

**<Text style={styles.label}>Apellido:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el apellido"**

**value={apellido}**

**onChangeText={setApellido}**

**/>**

**<Text style={styles.label}>Correo Electrónico:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el correo electrónico"**

**value={email}**

**onChangeText={setEmail}**

**keyboardType="email-address"**

**/>**

**<Text style={styles.label}>Número de Teléfono:</Text>**

**<TextInput**

**style={styles.input}**

**placeholder="Ingrese el número de teléfono"**

**value={telefono}**

**onChangeText={setTelefono}**

**keyboardType="phone-pad"**

**/>**

**<Text style={styles.label}>Fecha de Cumpleaños:</Text>**

**{Platform.OS === 'ios' ? (**

**<DatePickerIOS**

**date={fechaCumpleaños}**

**onDateChange={handleDateChange}**

**/>**

**) : (**

**<Button**

**title={fechaCumpleaños.toDateString()}**

**onPress={async () => {**

**try {**

**const { action, year, month, day } = await DatePickerAndroid.open({**

**date: fechaCumpleaños,**

**mode: 'date'**

**});**

**if (action !== DatePickerAndroid.dismissedAction) {**

**setFechaCumpleaños(new Date(year, month, day));**

**}**

**} catch (error) {**

**console.warn('Error al abrir el selector de fecha:', error);**

**}**

**}}**

**/>**

**)}**

**<TouchableOpacity onPress={handleSubmit} style={styles.button}>**

**<Text style={styles.buttonText}>Guardar Datos</Text>**

**</TouchableOpacity>**

**</View>**

**);**

**};**

**const styles = StyleSheet.create({**

**container: {**

**flex: 1,**

**justifyContent: 'center',**

**padding: 16,**

**backgroundColor: '#F5F5F5'**

**},**

**label: {**

**fontSize: 18,**

**marginVertical: 8,**

**fontWeight: 'bold'**

**},**

**input: {**

**height: 40,**

**borderColor: '#ddd',**

**borderWidth: 1,**

**borderRadius: 5,**

**paddingHorizontal: 8,**

**marginBottom: 16**

**},**

**button: {**

**backgroundColor: '#007BFF',**

**paddingVertical: 10,**

**borderRadius: 5**

**},**

**buttonText: {**

**color: '#FFF',**

**textAlign: 'center',**

**fontSize: 18**

**}**

**});**

**export default contact;**

y por último configuramos el archivo App.js de la siguiente forma

**import React from 'react';**

**import AuthNavigator from '../navigation/AuthNavigator';**

**import AuthProvider from '../context/AuthContext';**

**const App = () => {**

**return (**

**<AuthProvider>**

**<AuthNavigator />**

**</AuthProvider>**

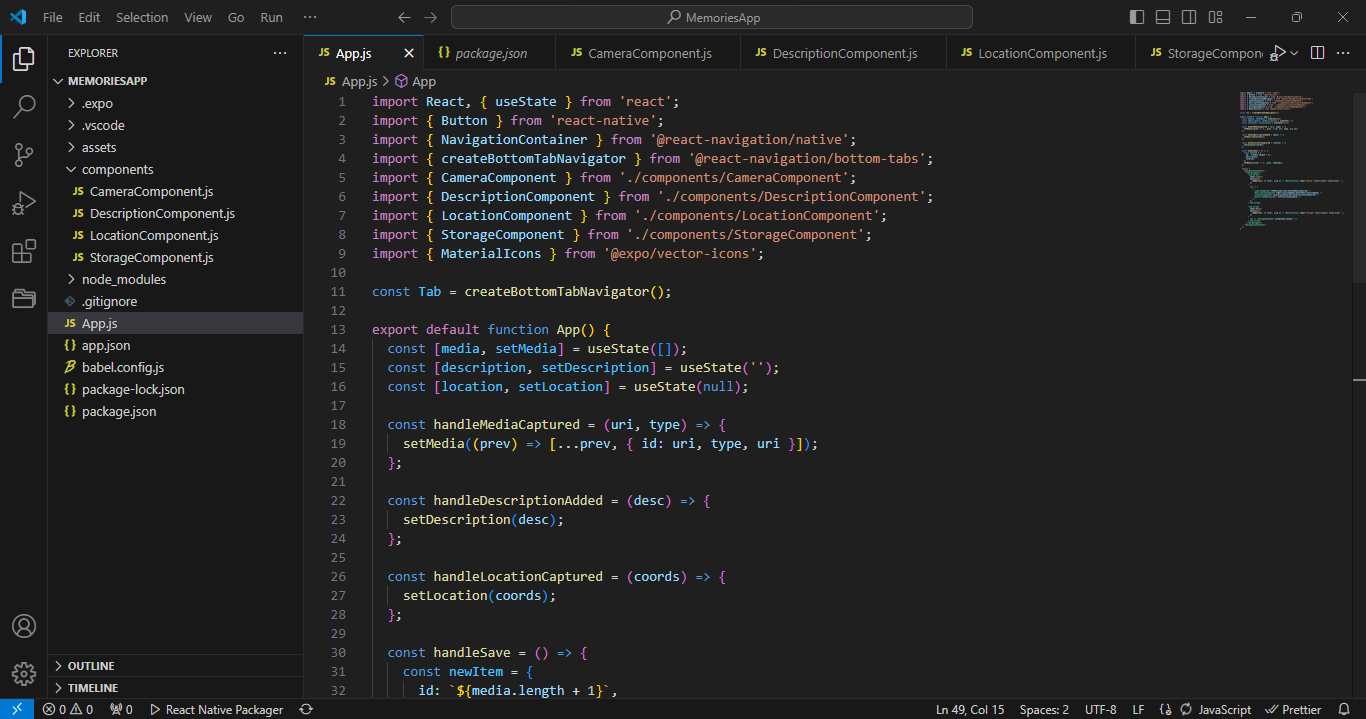
**);**

**};**

**export default App;**

**Ejercicio 02: MemoriesApp (35% completado)**

https://github.com/Cris0024/MemoriesApp.git



A continuación, se mostrará el script de todos los componentes, archivos adicionales, etc.

Los componentes que estructuran esta App están relacionados con cada una de las funcionalidades.

CameraComponent.js (Accede a la cámara)

**import React, { useState, useRef, useEffect } from 'react';**

**import { View, StyleSheet, Text, TouchableOpacity, Image } from 'react-native';**

**import { Camera } from 'expo-camera';**

**export const CameraComponent = ({ onMediaCaptured }) => {**

**const [hasPermission, setHasPermission] = useState(null);**

**const [cameraType, setCameraType] = useState(Camera.Constants.Type.back);**

**const [preview, setPreview] = useState(null);**

**const cameraRef = useRef(null);**

**useEffect(() => {**

**const getPermissions = async () => {**

**const { status } = await Camera.requestCameraPermissionsAsync();**

**setHasPermission(status === 'granted');**

**};**

**getPermissions();**

**}, []);**

**const toggleCameraType = () => {**

**if (Camera.Constants) {**

**setCameraType(cameraType === Camera.Constants.Type.back ? Camera.Constants.Type.front : Camera.Constants.Type.back);**

**}**

**};**

**const takePicture = async () => {**

**if (cameraRef.current) {**

**const photo = await cameraRef.current.takePictureAsync();**

**setPreview(photo.uri);**

**onMediaCaptured(photo.uri, 'image');**

**}**

**};**

**if (hasPermission === null) {**

**return <Text>Solicitando permisos...</Text>;**

**}**

**if (hasPermission === false) {**

**return <Text>Acceso a la cámara denegado</Text>;**

**}**

**return (**

**<View style={styles.container}>**

**{preview ? (**

**<Image source={{ uri: preview }} style={styles.preview} />**

**) : (**

**<Camera style={styles.camera} type={cameraType} ref={cameraRef}>**

**<View style={styles.buttonContainer}>**

**<TouchableOpacity style={styles.button} onPress={toggleCameraType}>**

**<Text style={styles.text}>Cambiar cámara</Text>**

**</TouchableOpacity>**

**<TouchableOpacity style={styles.button} onPress={takePicture}>**

**<Text style={styles.text}>Tomar foto</Text>**

**</TouchableOpacity>**

**</View>**

**</Camera>**

**)}**

**</View>**

**);**

**};**

**const styles = StyleSheet.create({**

**container: {**

**flex: 1,**

**},**

**camera: {**

**flex: 1,**

**},**

**buttonContainer: {**

**flex: 1,**

**flexDirection: 'row',**

**justifyContent: 'space-between',**

**margin: 20,**

**},**

**button: {**

**flex: 0.1,**

**alignItems: 'center',**

**backgroundColor: '#fff',**

**padding: 10,**

**},**

**text: {**

**fontSize: 18,**

**color: '#000',**

**},**

**preview: {**

**flex: 1,**

**justifyContent: 'center',**

**alignItems: 'center',**

**},**

**});**

DescriptionComponent.js (Agrega una descripción editable a las imágenes o videos capturados)

**import React, { useState } from 'react';**

**import { View, TextInput, Button, StyleSheet } from 'react-native';**

**export const DescriptionComponent = ({ onDescriptionAdded }) => {**

**const [description, setDescription] = useState('');**

**const handleSaveDescription = () => {**

**onDescriptionAdded(description);**

**setDescription('');**

**};**

**return (**

**<View style={styles.container}>**

**<TextInput**

**style={styles.input}**

**placeholder="Agrega una descripción"**

**value={description}**

**onChangeText={setDescription}**

**/>**

**<Button title="Guardar descripción" onPress={handleSaveDescription} />**

**</View>**

**);**

**};**

**const styles = StyleSheet.create({**

**container: {**

**padding: 10,**

**},**

**input: {**

**borderWidth: 1,**

**borderColor: '#ccc',**

**padding: 8,**

**marginBottom: 10,**

**borderRadius: 5,**

**},**

**});**

LocationComponent.js (Este componente obtendrá la ubicación GPS)

**import React, { useEffect, useState } from 'react';**

**import \* as Location from 'expo-location';**

**import { View, Text, Button, StyleSheet } from 'react-native';**

**export const LocationComponent = ({ onLocationCaptured }) => {**

**const [location, setLocation] = useState(null);**

**useEffect(() => {**

**(async () => {**

**const { status } = await Location.requestForegroundPermissionsAsync();**

**if (status !== 'granted') {**

**alert('Permiso de ubicación denegado');**

**return;**

**}**

**const loc = await Location.getCurrentPositionAsync({});**

**setLocation(loc);**

**onLocationCaptured(loc.coords);**

**})();**

**}, []);**

**return (**

**<View style={styles.container}>**

**{location ? (**

**<Text>Ubicación: {location.coords.latitude}, {location.coords.longitude}</Text>**

**) : (**

**<Text>Obteniendo ubicación...</Text>**

**)}**

**</View>**

**);**

**};**

**const styles = StyleSheet.create({**

**container: {**

**padding: 10,**

**},**

**});**

StorageComponent.js (Se encargará de almacenar los datos localmente)

import React from 'react';

import { View, Text, Button, FlatList, StyleSheet } from 'react-native';

export const StorageComponent = ({ savedItems }) => {

  return (

    <View style={styles.container}>

      <FlatList

        data={savedItems}

        keyExtractor={(item) => item.id}

        renderItem={({ item }) => (

          <View style={styles.item}>

            <Text>{item.description}</Text>

            <Text>{item.location.latitude}, {item.location.longitude}</Text>

          </View>

        )}

      />

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    padding: 10,

  },

  item: {

    padding: 10,

    borderBottomWidth: 1,

    borderBottomColor: '#ccc',

  },

});

App.js (donde se integrarán todos los componentes)

**import React, { useState } from 'react';**

**import { Button } from 'react-native';**

**import { NavigationContainer } from '@react-navigation/native';**

**import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';**

**import { CameraComponent } from './components/CameraComponent';**

**import { DescriptionComponent } from './components/DescriptionComponent';**

**import { LocationComponent } from './components/LocationComponent';**

**import { StorageComponent } from './components/StorageComponent';**

**import { MaterialIcons } from '@expo/vector-icons';**

**const Tab = createBottomTabNavigator();**

**export default function App() {**

**const [media, setMedia] = useState([]);**

**const [description, setDescription] = useState('');**

**const [location, setLocation] = useState(null);**

**const handleMediaCaptured = (uri, type) => {**

**setMedia((prev) => [...prev, { id: uri, type, uri }]);**

**};**

**const handleDescriptionAdded = (desc) => {**

**setDescription(desc);**

**};**

**const handleLocationCaptured = (coords) => {**

**setLocation(coords);**

**};**

**const handleSave = () => {**

**const newItem = {**

**id: `${media.length + 1}`,**

**description,**

**location,**

**};**

**setMedia((prev) => [...prev, newItem]);**

**};**

**return (**

**<NavigationContainer>**

**<Tab.Navigator>**

**<Tab.Screen**

**name="Camera"**

**options={{**

**tabBarIcon: ({ color, size }) => <MaterialIcons name="camera" color={color} size={size} />,**

**}}**

**>**

**{() => (**

**<>**

**<CameraComponent onMediaCaptured={handleMediaCaptured} />**

**<DescriptionComponent onDescriptionAdded={handleDescriptionAdded} />**

**<LocationComponent onLocationCaptured={handleLocationCaptured} />**

**<Button title="Guardar" onPress={handleSave} />**

**</>**

**)}**

**</Tab.Screen>**

**<Tab.Screen**

**name="Media"**

**options={{**

**tabBarIcon: ({ color, size }) => <MaterialIcons name="storage" color={color} size={size} />,**

**}}**

**>**

**{() => <StorageComponent savedItems={media} />}**

**</Tab.Screen>**

**</Tab.Navigator>**

**</NavigationContainer>**

**);**

**}**