UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ESCUELA DE COMPUTACIÓN

Diseño y Programación de Software Multiplataforma

PRUEBA PRÁCTICA

02

CICLO

02-2024

DOCENTE:

Ing. Karens Medrano

GRUPO:

01L

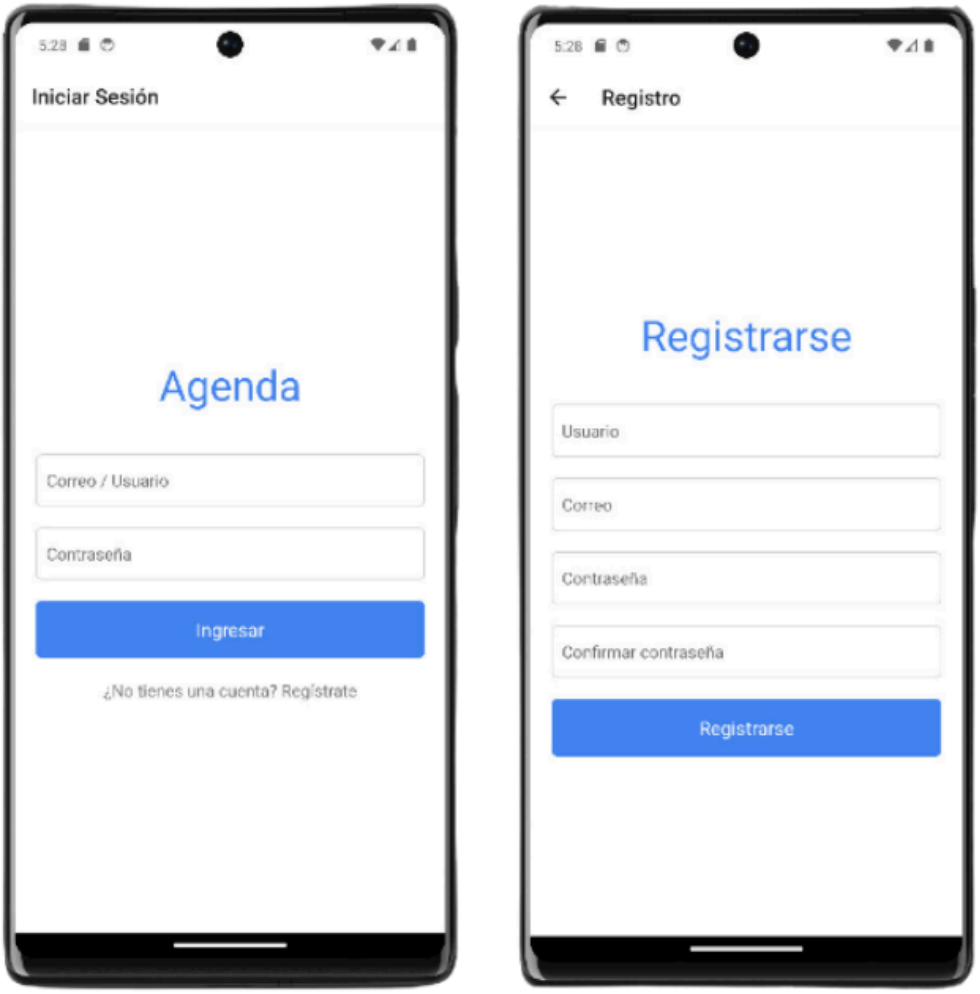
Integrantes:	**Carnet:**
Ponce López, Alejandra del Carmen	PL210665
Rodríguez Parada, Gabriela Lourdes	RP210191

Ejercicio 1:

Diseñar una aplicación móvil para ayudarte a gestionar contactos de manera eficiente y recordar los cumpleaños importantes de tus amigos, familiares y colegas. Con una interfaz intuitiva y características prácticas, CumpleAgenda te permite mantener tus relaciones sociales sólidas y nunca olvidar un cumpleaños importante.

Requerimientos:

- **Inicio de Sesión:** La aplicación integra un sistema de inicio de sesión seguro que protege los datos del usuario, requiriendo que los usuarios creen una cuenta e inicien sesión para acceder a todas sus funciones.



- **Registro de Contactos:** La aplicación permite a los usuarios agregar fácilmente nuevos contactos, incluyendo nombres, números de teléfono, direcciones de correo electrónico, y también registrar la fecha de nacimiento de cada contacto en su agenda.

- **Recordatorios de Cumpleaños:**

  CumpleAgenda tiene recordatorios para los cumpleaños de tus contactos. En donde se muestra la lista de cumpleañeros y los clasifica por colores según la fecha de su nacimiento:

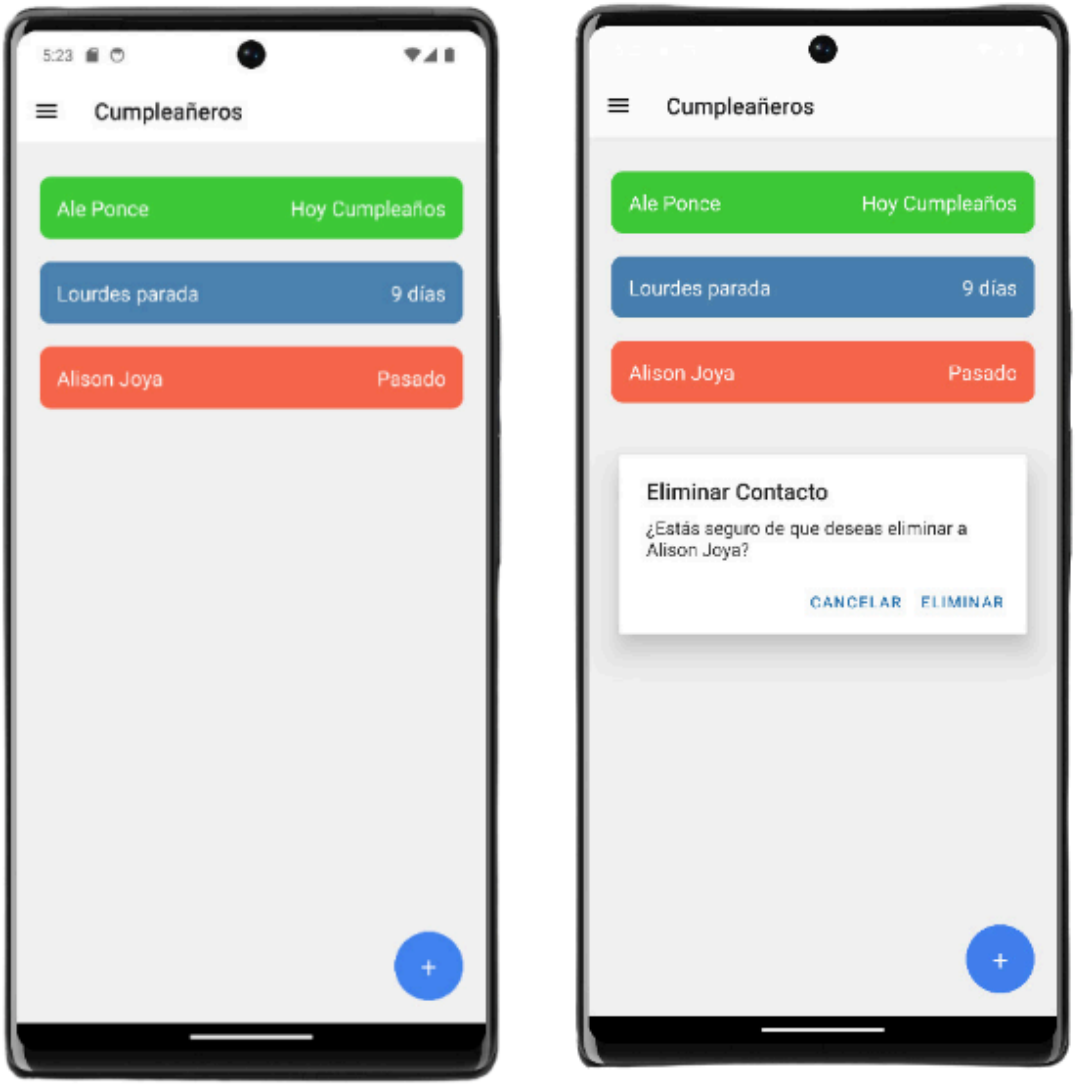
  ▪ Verde: Cumpleañeros del día de hoy

  ▪ Rojo: Cumpleaños que ya paso la fecha

  ▪ Azul: Cumpleaños que aún faltan días para su cumpleaños

- **Eliminar Usuario:**

  CumpleAgenda permite a los usuarios eliminar a los contactos fácilmente al mantener presionado al contacto deberá mostrar una ventana emergente que preguntará si deseamos eliminar el contacto.

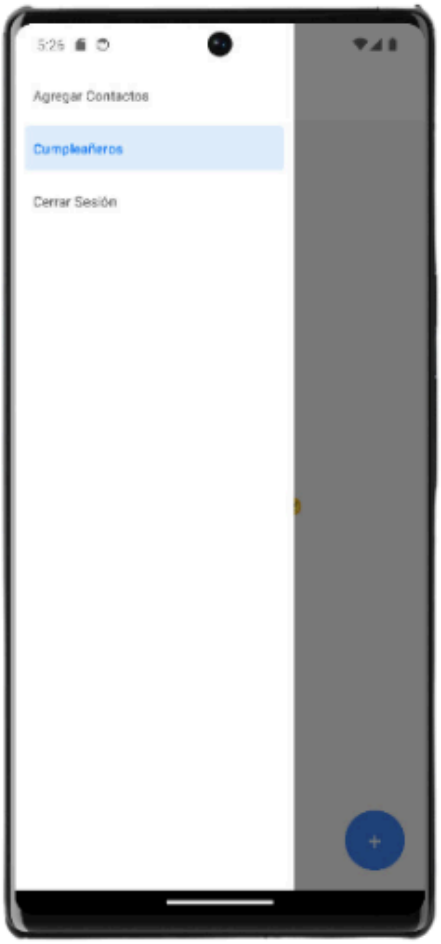


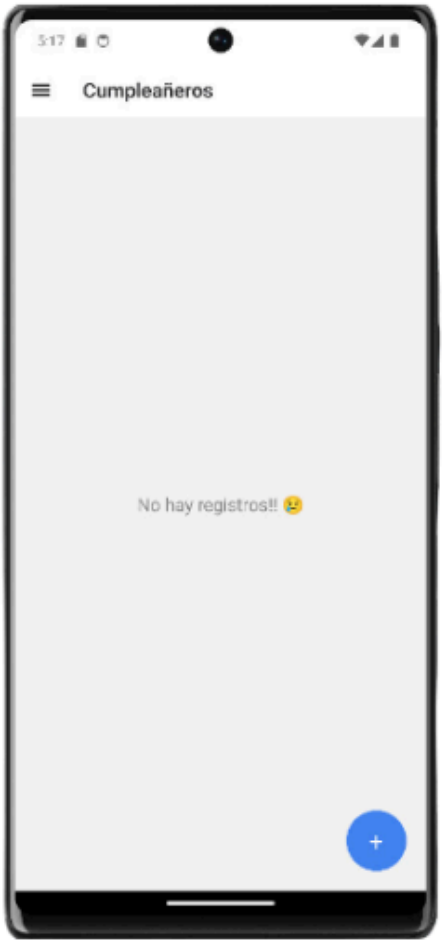
Interfaz Intuitiva:

CumpleAgenda presenta una interfaz fácil de usar que permite a los usuarios navegar rápidamente a través de un menú desplegable para registrar contactos y cumpleaños.

Esta pantalla muestra automáticamente si la lista esta vacía, proporciona una explicación clara de que no hay registros de contactos.

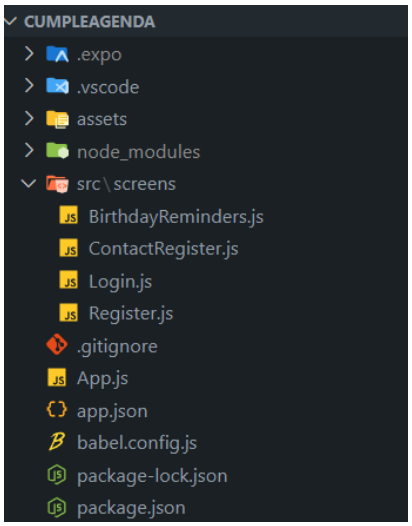
Además, para añadir nuevos cumpleañeros, los usuarios pueden acceder al formulario tanto desde el menú como a través del botón flotante circular de "Agregar"





Estructura del Proyecto

La estructura del proyecto es la siguiente:



App.js

```
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createDrawerNavigator } from '@react-navigation/drawer';
import { createStackNavigator } from '@react-navigation/stack';
import ContactRegister from './src/screens/ContactRegister';
import BirthdayReminders from './src/screens/BirthdayReminders';
import Login from './src/screens/Login';
import Register from './src/screens/Register';
import { Button, View } from 'react-native';

const Drawer = createDrawerNavigator();
const Stack = createStackNavigator();

const Logout = ({ navigation }) => {
  React.useEffect(() => {
    navigation.reset({
      index: 0,
      routes: [{ name: 'Login' }],
    });
  }, [navigation]);

  return (
    <View>
      <Button title="Cerrar Sesión" onPress={() => navigation.reset({
        index: 0,
        routes: [{ name: 'Login' }],
      })} />
    </View>
  );
};

const DrawerNavigator = () => {
  return (
    <Drawer.Navigator>
      <Drawer.Screen name="Agregar Contactos"
component={ContactRegister} />
      <Drawer.Screen name="Cumpleañeros" component={BirthdayReminders}
/>
      <Drawer.Screen name="Cerrar Sesión" component={Logout} />
```

```
    </Drawer.Navigator>
  );
};

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Login" screenOptions={{
headerShown: true }}>
        <Stack.Screen name="Login" component={Login} options={{ title:
'Iniciar Sesión' }} />
        <Stack.Screen name="Register" component={Register} options={{
title: 'Registro' }} />
        <Stack.Screen name="Home" component={DrawerNavigator}
options={{ headerShown: false }} />
        <Stack.Screen name="Agregar Contacto"
component={ContactRegister} options={{ title: 'Agregar Contacto' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Login.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, Alert, StyleSheet }
from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

const Login = ({ navigation }) => {
  const [usernameOrEmail, setUsernameOrEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    try {
      const storedUserData = await AsyncStorage.getItem('users');
      const users = storedUserData ? JSON.parse(storedUserData) : [];

      const user = users.find(
```

```
        u => (u.username === usernameOrEmail || u.email ===
usernameOrEmail) && u.password === password
      );

      if (user) {
        Alert.alert('Login exitoso');
        navigation.navigate('Home');
      } else {
        Alert.alert('Error', 'Credenciales incorrectas');
      }
    } catch (error) {
      console.log(error);
      Alert.alert('Error', 'Hubo un problema al procesar el inicio de
sesión');
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Agenda</Text>
      <TextInput
        style={styles.input}
        placeholder="Correo / Usuario"
        value={usernameOrEmail}
        onChangeText={setUsernameOrEmail}
      />
      <TextInput
        style={styles.input}
        placeholder="Contraseña"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
      />
      <TouchableOpacity style={styles.button} onPress={handleLogin}>
        <Text style={styles.buttonText}>Ingresar</Text>
      </TouchableOpacity>
      <TouchableOpacity onPress={() =>
navigation.navigate('Register')}>
        <Text style={styles.registerText}>¿No tienes una cuenta?
Regístrate</Text>
```

```
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', padding: 20,
backgroundColor: '#fff' },
  title: { fontSize: 40, color: '#3B82F6', textAlign: 'center',
marginBottom: 40 },
  input: { borderWidth: 1, borderColor: '#ccc', padding: 10,
marginBottom: 20, borderRadius: 5, fontSize: 16 },
  button: { backgroundColor: '#3B82F6', padding: 15, borderRadius: 5,
alignItems: 'center' },
  buttonText: { color: '#fff', fontSize: 18 },
  registerText: { textAlign: 'center', marginTop: 20, fontSize: 16,
color: '#777' },
  registerLink: { color: '#3B82F6', fontWeight: 'bold' },
});

export default Login;
```

Register.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, Alert, StyleSheet }
from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
const Register = ({ navigation }) => {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [confirmPassword, setConfirmPassword] = useState('');

  const validateEmail = (email) => {
    const emailPattern = /^[a-zA-Z0-9._-]+@gmail\.com$/;
    return emailPattern.test(email);
  };

  const handleRegister = async () => {
```

```
    if (!username.match(/^[A-Za-z]+$/)) {
      Alert.alert('Error', 'El nombre de usuario debe contener solo
letras');
      return;
    }
    if (!validateEmail(email)) {
      Alert.alert('Error', 'El correo debe tener el formato
alumno@gmail.com');
      return;
    }
    if (password !== confirmPassword) {
      Alert.alert('Error', 'Las contraseñas no coinciden');
      return;
    }

    const newUser = { username, email, password };
    try {
      const storedUserData = await AsyncStorage.getItem('users');
      const users = storedUserData ? JSON.parse(storedUserData) : [];
      users.push(newUser);
      await AsyncStorage.setItem('users', JSON.stringify(users));

      Alert.alert('Registro exitoso');
      navigation.navigate('Login');
    } catch (error) {
      console.log(error);
      Alert.alert('Error', 'Hubo un problema al registrarse');
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Registrarse</Text>
      <TextInput
        style={styles.input}
        placeholder="Usuario"
        value={username}
        onChangeText={setUsername}
      />
      <TextInput
```

```
        style={styles.input}
        placeholder="Correo"
        value={email}
        onChangeText={setEmail}
        keyboardType="email-address"
      />
      <TextInput
        style={styles.input}
        placeholder="Contraseña"
        value={password}
        onChangeText={setPassword}
        secureTextEntry
      />
      <TextInput
        style={styles.input}
        placeholder="Confirmar contraseña"
        value={confirmPassword}
        onChangeText={setConfirmPassword}
        secureTextEntry
      />
      <TouchableOpacity style={styles.button}
onPress={handleRegister}>
        <Text style={styles.buttonText}>Registrarse</Text>
      </TouchableOpacity>
    </View>
  );
};
const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', padding: 20,
backgroundColor: '#fff' },
  title: { fontSize: 40, color: '#3B82F6', textAlign: 'center',
marginBottom: 40 },
  input: { borderWidth: 1, borderColor: '#ccc', padding: 10,
marginBottom: 20, borderRadius: 5, fontSize: 16 },
  button: { backgroundColor: '#3B82F6', padding: 15, borderRadius: 5,
alignItems: 'center' },
  buttonText: { color: '#fff', fontSize: 18 },
})

export default Register;
```

ContactRegister.js

```
import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert }
from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import DateTimePicker from '@react-native-community/datetimepicker';
const ContactRegister = ({ navigation }) => {
  const [name, setName] = useState('');
  const [surname, setSurname] = useState('');
  const [email, setEmail] = useState('');
  const [phone, setPhone] = useState('');
  const [birthday, setBirthday] = useState(new Date());
  const [showDatePicker, setShowDatePicker] = useState(false);
  const validateAndSaveContact = async () => {
    if (!name || !surname || !email || !phone || !birthday) {
      Alert.alert('Error', 'Por favor llena todos los campos');
      return;
    }
    if (!/^[A-Za-z]+$/.test(name) || !/^[A-Za-z]+$/.test(surname)) {
      Alert.alert('Error', 'Nombre y apellido deben contener solo
letras');
      return;
    }
    if (!/^\d{8}$/.test(phone)) {
      Alert.alert('Error', 'El número de teléfono debe tener 8
dígitos');
      return;
    }
    if
(!/^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/.test(email)) {
      Alert.alert('Error', 'El correo debe tener una estructura:
alumno@gmail.com ');
      return;
    }

    const contact = { name, surname, email, phone, birthday };
    try {
      const existingContacts = await AsyncStorage.getItem('contacts');
      const contacts = existingContacts ? JSON.parse(existingContacts)
: [];
```

```
      contacts.push(contact);
      await AsyncStorage.setItem('contacts',
JSON.stringify(contacts));
      Alert.alert('Contacto agregado exitosamente');
      navigation.navigate('Cumpleañeros');
    } catch (error) {
      Alert.alert('Error', 'Hubo un problema al guardar el contacto');
    }
  };
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Agregar Persona</Text>
      <TextInput style={styles.input} placeholder="Nombre"
value={name} onChangeText={setName} />
      <TextInput style={styles.input} placeholder="Apellido"
value={surname} onChangeText={setSurname} />
      <TextInput style={styles.input} placeholder="Correo"
value={email} onChangeText={setEmail} keyboardType="email-address" />
      <TextInput style={styles.input} placeholder="Número de teléfono"
value={phone} onChangeText={setPhone} keyboardType="numeric" />
      <TouchableOpacity style={styles.input} onPress={() =>
setShowDatePicker(true)}>
        <Text>{birthday ? birthday.toDateString() : 'Fecha de
cumpleaños'}</Text>
      </TouchableOpacity>
      {showDatePicker && (
        <DateTimePicker
          value={birthday}
          mode="date"
          display="default"
          onChange={(event, selectedDate) => {
            setShowDatePicker(false);
            if (selectedDate) setBirthday(selectedDate);
          }}
        />
      )}
      <TouchableOpacity style={styles.button}
onPress={validateAndSaveContact}>
        <Text style={styles.buttonText}>Agregar Persona</Text>
      </TouchableOpacity>
```

```
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fff',
    padding: 20,
  },
  title: {
    fontSize: 24,
    fontWeight: 'bold',
    marginBottom: 20,
  },
  input: {
    width: '100%',
    height: 50,
    borderColor: '#ccc',
    borderWidth: 1,
    borderRadius: 5,
    paddingHorizontal: 10,
    marginVertical: 10,
  },
  button: {
    width: '100%',
    backgroundColor: '#3B82F6',
    padding: 15,
    borderRadius: 5,
    alignItems: 'center',
    marginTop: 20,
  },
  buttonText: {
    color: '#fff',
    fontSize: 16,
  },
});
export default ContactRegister;
```

BirthdayReminders.js

```
import React, { useEffect, useState } from 'react';
import { View, Text, ScrollView, StyleSheet, TouchableOpacity, Alert }
from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { useFocusEffect } from '@react-navigation/native';
const BirthdayReminders = ({ navigation }) => {
 const [contacts, setContacts] = useState([]);
  useFocusEffect(
    React.useCallback(() => {
      const fetchContacts = async () => {
        try {
          const storedContacts = await
AsyncStorage.getItem('contacts');
          if (storedContacts) {
            setContacts(JSON.parse(storedContacts));
          } else {
            setContacts([]);
          }
        } catch (error) {
          console.log('Error al recuperar los contactos:', error);
        }
      };
      fetchContacts();
    }, [])
  );
  const getBirthdayStatus = (birthday) => {
    const today = new Date();
    const birthDate = new Date(birthday);
    today.setHours(0, 0, 0, 0);
    birthDate.setHours(0, 0, 0, 0);
    birthDate.setFullYear(today.getFullYear());
    if (birthDate.getDate() === today.getDate() &&
birthDate.getMonth() === today.getMonth()) {
      return 'Hoy Cumpleaños';
    } else if (birthDate < today) {
      return 'Pasado';
    } else {
      const daysUntil = Math.ceil((birthDate.getTime() -
today.getTime()) / (1000 * 60 * 60 * 24));
```

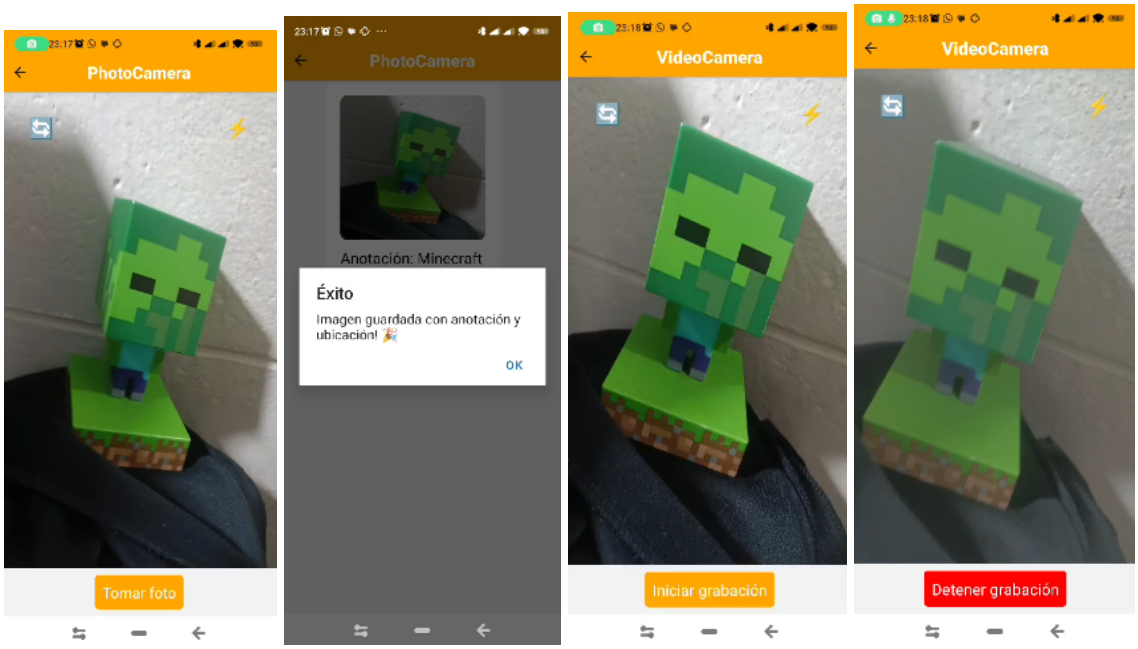
```
      return `${daysUntil} días`;
    }
  };
  const getColorByStatus = (status) => {
    if (status === 'Hoy Cumpleaños') return '#32CD32';
    if (status === 'Pasado') return '#FF6347';
    return '#4682B4';
  };
  const handleLongPress = (contact) => {
    Alert.alert(
      'Eliminar Contacto',
      `¿Estás seguro de que deseas eliminar a ${contact.name}
${contact.surname}?`,
      [
        { text: 'Cancelar', style: 'cancel' },
        { text: 'Eliminar', onPress: () => deleteContact(contact) },
      ],
      { cancelable: true }
    );
  };

  const deleteContact = async (contactToDelete) => {
    const updatedContacts = contacts.filter(contact => contact !==
contactToDelete);
    setContacts(updatedContacts);
    await AsyncStorage.setItem('contacts',
JSON.stringify(updatedContacts));
  };

  return (
    <View style={styles.container}>
      {contacts.length === 0 ? (
        <View style={styles.noContactsContainer}>
          <Text style={styles.noContactsText}>No hay registros!!
😢</Text>
        </View>
      ) : (
        <ScrollView>
          {contacts.map((contact, index) => {
            const status = getBirthdayStatus(contact.birthday);
```

```
            const color = getColorByStatus(status);
            return (
              <TouchableOpacity
                key={index}
                style={[styles.contactItem, { backgroundColor: color
}]}
                onLongPress={() => handleLongPress(contact)}
              >
                <Text style={styles.contactText}>{`${contact.name}
${contact.surname}`}</Text>
                <Text style={styles.contactText}>{status}</Text>
              </TouchableOpacity>
            );
          })}
        </ScrollView>
      )}
      <TouchableOpacity style={styles.addButton} onPress={() =>
navigation.navigate('Agregar Contacto')}>
        <Text style={styles.addButtonText}>+</Text>
      </TouchableOpacity>
    </View>
  );
};

const styles = StyleSheet.create({
  container: { flex: 1, padding: 20 },
  contactItem: { padding: 15, borderRadius: 10, marginVertical: 10,
flexDirection: 'row', justifyContent: 'space-between' },
  contactText: { fontSize: 18, color: '#fff' },
  addButton: { backgroundColor: '#3B82F6', width: 60, height: 60,
borderRadius: 30, justifyContent: 'center', alignItems: 'center',
position: 'absolute', bottom: 20, right: 20 },
  addButtonText: { color: '#fff', fontSize: 24 },
  noContactsContainer: { flex: 1, justifyContent: 'center',
alignItems: 'center' },
  noContactsText: { fontSize: 18, color: '#888', textAlign: 'center'
},
});

export default BirthdayReminders;
```

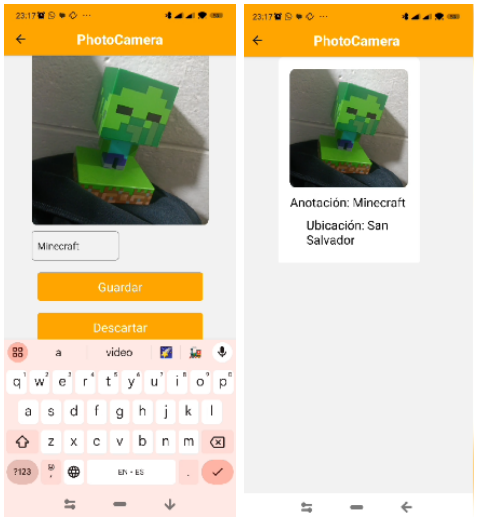
Ejercicio 2:

Desarrollar una aplicación móvil MemoriesApp que permita a los usuarios capturar imágenes y videos, agregar anotaciones(texto) sobre estas imágenes y videos, y almacenar la ubicación geográfica (coordenadas GPS) donde se capturaron. La aplicación debe incluir las siguientes funcionalidades:

1. **Captura de Imágenes y Videos:** La aplicación permite al usuario alternar entre la cámara frontal y trasera, ofreciendo opciones tanto para capturar imágenes como para grabar videos. Una vez capturado la imagen o el video, se muestra una vista previa para que el usuario decida si desea guardarlo o descartarlo.



2. **Mensaje o descripción:** Al capturar una imagen o video, el usuario puede agregar una anotación de texto, la cual es posicionada en la parte inferior de la imagen. Estas anotaciones deben ser editables antes de guardar la imagen o el video.

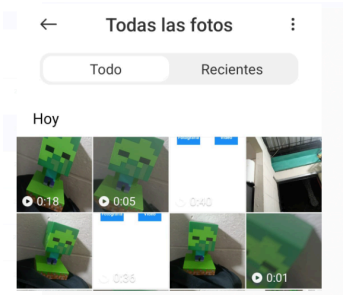


3. **Geolocalización:** La aplicación obtiene automáticamente la ubicación geográfica (latitud y longitud) al capturar una imagen o video. Esta ubicación(geocodificación) se almacena junto con la imagen o video y con cualquier anotación añadida.

```
  if (locationStatus.status === 'granted') {
    const location = await Location.getCurrentPositionAsync({});
    const reverseGeocode = await Location.reverseGeocodeAsync(location.coords);
    if (reverseGeocode.length > 0) {
      const city = reverseGeocode[0].city || reverseGeocode[0].region || 'Ubicación desconocida';
      setLocation(city);
    } else {
      setLocation('Ubicación desconocida');
    }
  }
};
```

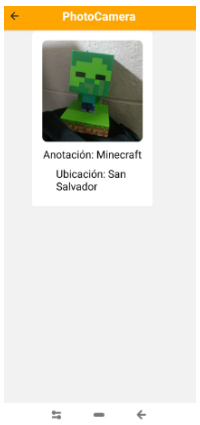
Se utiliza expo location para obtener la localización de la imagen.

4. **Almacenamiento:** Los archivos de imágenes y videos, junto con sus anotaciones y datos de ubicación, deben ser almacenados localmente en el dispositivo. La aplicación debe mostrar una lista de todas las imágenes y videos guardados, junto con sus anotaciones y la ubicación en un mapa.



Las imagenes y videos se almacenan en el dispositivo.

El único problema de la aplicación es que en files no muestra la lista con las diferentes imagenes, en lugar de ello, se muestra cada cuadro por separado una vez que la imagen es tomada.



Código:

App.js

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

import Icon from "react-native-vector-icons/Ionicons";




export function MediaButton({ label = 'none', icon= 'film', onPress})
{


  return (
    <TouchableOpacity onPress={onPress} style={styles.touchable}>
      <View style={styles.button_default}>
        <Icon name={icon} size={40} color="white" />
        <Text style={styles.text}>{label}</Text>
      </View>
    </TouchableOpacity>
  );
}


const styles = StyleSheet.create({
  button_default: {
    flexDirection: 'column',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: 'dodgerblue',
    height:120,
```

```
    width:120,
    margin:20
  },
  text: {
    fontSize: 20,
    fontWeight: 'bold',
    color: 'white'
  },
  touchable:{
    flex:1,
    alignItems: 'center'
  }
});
```

VideoCamera.js

```
import React, { useState, useRef, useEffect } from 'react';
import { Text, View, StyleSheet, TouchableOpacity, FlatList,
TextInput, Platform, Alert } from 'react-native';
import { Camera, CameraType } from 'expo-camera';
import * as MediaLibrary from 'expo-media-library';
import * as Location from 'expo-location';


export function VideoCamera() {
  const [hasCameraPermission, setHasCameraPermission] =
useState(null);
  const [hasMicrophonePermission, setHasMicrophonePermission] =
useState(null);
```

```
  const [videos, setVideos] = useState([]);
  const [type, setType] = useState(CameraType.back);
  const [flash, setFlash] = useState(Camera.Constants.FlashMode.off);
  const [isRecording, setIsRecording] = useState(false);
  const [annotation, setAnnotation] = useState('');
  const [location, setLocation] = useState(null);
  const [currentVideo, setCurrentVideo] = useState(null);
  const cameraRef = useRef(null);

  useEffect(() => {
    const setupPermissions = async () => {
      const { status: cameraStatus } = await
Camera.requestCameraPermissionsAsync();
      setHasCameraPermission(cameraStatus === 'granted');

      if (Platform.OS !== 'web') {
        const { status: microphoneStatus } = await
Camera.requestMicrophonePermissionsAsync();
        setHasMicrophonePermission(microphoneStatus === 'granted');
      } else {
        setHasMicrophonePermission(true);
      }

      await MediaLibrary.requestPermissionsAsync();
      const locationStatus = await
Location.requestForegroundPermissionsAsync();

      if (locationStatus.status === 'granted') {
```

```
        const location = await Location.getCurrentPositionAsync({});
        const reverseGeocode = await
Location.reverseGeocodeAsync(location.coords);
        if (reverseGeocode.length > 0) {
          const city = reverseGeocode[0].city ||
reverseGeocode[0].region || 'Ubicación desconocida';
          setLocation(city);
        } else {
          setLocation('Ubicación desconocida');
        }
      }
    };

    setupPermissions();
  }, []);

  const startRecording = async () => {
    if (!hasCameraPermission || !hasMicrophonePermission) {
      Alert.alert('Permisos no concedidos', 'No tienes permisos para
la cámara o el micrófono');
      return;
    }

    if (cameraRef.current) {
      try {
        setIsRecording(true);
        const videoData = await cameraRef.current.recordAsync();
        setCurrentVideo(videoData.uri);
```

```
    } catch (error) {
      console.log(error);
      setIsRecording(false);
    }
  }
};

const stopRecording = () => {
  if (cameraRef.current && isRecording) {
    cameraRef.current.stopRecording();
    setIsRecording(false);
  }
};

const saveVideo = async (uri) => {
  if (uri && location) {
    try {
      const asset = await MediaLibrary.createAssetAsync(uri);

      const metadata = {
        uri: asset.uri,
        annotation,
        location,
      };

      setVideos(prevVideos => [...prevVideos, metadata]);
```

```
        Alert.alert('Éxito', 'Video guardado con anotación y
ubicación! 🎉');
        setCurrentVideo(null);
        setAnnotation('');
        setLocation(null);
      } catch (error) {
        console.log(error);
      }
    }
  };

  const discardVideo = () => {
    setCurrentVideo(null);
    setAnnotation('');
    setLocation(null);
  };

  const renderVideoCard = ({ item }) => (
    <View style={styles.card}>
      <Text style={styles.message}>Anotación: {item.annotation}</Text>
      <Text style={styles.message}>Ubicación: {item.location}</Text>
      <TouchableOpacity onPress={() => saveVideo(item.uri)}
style={styles.button}>
        <Text style={styles.buttonText}>Guardar</Text>
      </TouchableOpacity>
    </View>
  );
```

```
  if (hasCameraPermission === false || (Platform.OS !== 'web' &&
hasMicrophonePermission === false)) {
    return <Text>No tienes acceso a la cámara o al micrófono</Text>;
  }

  return (
    <View style={styles.container}>
      {currentVideo ? (
        <View>
          <TextInput
            placeholder="Añadir anotación"
            value={annotation}
            onChangeText={setAnnotation}
            style={styles.annotationInput}
          />
          <TouchableOpacity onPress={() => saveVideo(currentVideo)}
style={styles.button}>
            <Text style={styles.buttonText}>Guardar</Text>
          </TouchableOpacity>
          <TouchableOpacity onPress={discardVideo}
style={styles.button}>
            <Text style={styles.buttonText}>Descartar</Text>
          </TouchableOpacity>
        </View>
      ) : (
        <>
          {videos.length === 0 ? (
            <Camera
```

```
              style={styles.camera}
              type={type}
              ref={cameraRef}
              flashMode={flash}
            >
              <View style={styles.controls}>
                <TouchableOpacity
                  onPress={() => setType(type === CameraType.back ?
CameraType.front : CameraType.back)}
                  style={styles.iconButton}
                >
                  <Text style={styles.iconText}>🔄</Text>
                </TouchableOpacity>
                <TouchableOpacity
                  onPress={() => setFlash(flash ===
Camera.Constants.FlashMode.off ? Camera.Constants.FlashMode.on :
Camera.Constants.FlashMode.off)}
                  style={styles.iconButton}
                >
                  <Text style={styles.iconText}>{flash ===
Camera.Constants.FlashMode.off ? '⚡' : '💡'}</Text>
                </TouchableOpacity>
              </View>
            </Camera>
          ) : (
            <FlatList
              data={videos}
              renderItem={renderVideoCard}
```

```
              keyExtractor={(item, index) => index.toString()}
            />
          )}

          {videos.length === 0 && (
            <>
              {isRecording ? (
                <TouchableOpacity onPress={stopRecording}
style={[styles.button, { backgroundColor: 'red' }]}>
                  <Text style={styles.buttonText}>Detener
grabación</Text>
                </TouchableOpacity>
              ) : (
                <TouchableOpacity onPress={startRecording}
style={styles.button}>
                  <Text style={styles.buttonText}>Iniciar
grabación</Text>
                </TouchableOpacity>
              )}
            </>
          )}
        </>
      )}
    </View>
  );
}

const styles = StyleSheet.create({
```

```
container: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
},
camera: {
  flex: 1,
  width: '100%',
},
button: {
  backgroundColor: 'orange',
  padding: 10,
  borderRadius: 5,
  alignItems: 'center',
  justifyContent: 'center',
  margin: 10,
},
buttonText: {
  color: '#fff',
  fontSize: 18,
},
iconButton: {
  padding: 10,
},
iconText: {
  fontSize: 24,
},
```

```
  controls: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    paddingHorizontal: 30,
    marginTop: 20,
  },
  annotationInput: {
    borderColor: 'gray',
    borderWidth: 1,
    borderRadius: 5,
    padding: 10,
    width: '90%',
    marginBottom: 10,
  },
  card: {
    width: '90%',
    padding: 20,
    backgroundColor: '#fff',
    borderRadius: 10,
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 10,
    marginBottom: 20,
    alignItems: 'center',
  },
  message: {
```

```
    fontSize: 18,
    color: 'black',
    marginBottom: 10,
  },
});
```

PhotoCamera.js

```
import React, { useState, useRef, useEffect } from 'react';
import { Text, View, StyleSheet, Image, TouchableOpacity, FlatList,
TextInput, Alert } from 'react-native';
import { Camera, CameraType } from 'expo-camera';
import * as MediaLibrary from 'expo-media-library';
import * as Location from 'expo-location';


export function PhotoCamera() {
  const [hasCameraPermission, setHasCameraPermission] =
useState(null);
  const [images, setImages] = useState([]);
  const [type, setType] = useState(CameraType.back);
  const [flash, setFlash] = useState(Camera.Constants.FlashMode.off);
  const [annotation, setAnnotation] = useState('');
  const [currentImage, setCurrentImage] = useState(null);
  const [location, setLocation] = useState(null);
  const cameraRef = useRef(null);


  useEffect(() => {
    const setupPermissions = async () => {
```

```
      const cameraStatus = await
Camera.requestCameraPermissionsAsync();
      const mediaStatus = await
MediaLibrary.requestPermissionsAsync();
      const locationStatus = await
Location.requestForegroundPermissionsAsync();

      setHasCameraPermission(cameraStatus.status === 'granted' &&
mediaStatus.status === 'granted');

      if (locationStatus.status === 'granted') {
        const location = await Location.getCurrentPositionAsync({});
        const reverseGeocode = await
Location.reverseGeocodeAsync(location.coords);
        if (reverseGeocode.length > 0) {
          const city = reverseGeocode[0].city ||
reverseGeocode[0].region || 'Ubicación desconocida';
          setLocation(city);
        } else {
          setLocation('Ubicación desconocida');
        }
      }
    };

    setupPermissions();
  }, []);

  const takePicture = async () => {
    if (cameraRef.current) {
```

```
      try {
        const data = await cameraRef.current.takePictureAsync();
        setCurrentImage(data.uri);
      } catch (error) {
        console.log(error);
      }
    }
  };

  const savePicture = async (uri) => {
    if (uri && location) {
      try {
        const asset = await MediaLibrary.createAssetAsync(uri);

        const metadata = {
          uri: asset.uri,
          annotation,
          location,
        };

        setImages(prevImages => [...prevImages, metadata]);

        Alert.alert('Éxito', 'Imagen guardada con anotación y
ubicación! 🎉');
        setCurrentImage(null);
        setAnnotation('');
        setLocation(null);
```

```
    } catch (error) {
      console.log(error);
    }
  }
};

const discardPicture = () => {
  setCurrentImage(null);
  setAnnotation('');
  setLocation(null);
};

const renderImageCard = ({ item }) => (
  <View style={styles.card}>
    <Image source={{ uri: item.uri }} style={styles.image} />
    <Text style={styles.message}>Anotación: {item.annotation}</Text>
    <Text style={styles.message}>Ubicación: {item.location}</Text>
  </View>
);

if (hasCameraPermission === false) {
  return <Text>No tienes acceso a la cámara</Text>;
}

return (
  <View style={styles.container}>
    {currentImage ? (
```

```
        <View>
          <Image source={{ uri: currentImage }}
style={styles.previewImage} />
          <TextInput
            placeholder="Añadir anotación"
            value={annotation}
            onChangeText={setAnnotation}
            style={styles.annotationInput}
          />
          <TouchableOpacity onPress={() => savePicture(currentImage)}
style={styles.button}>
            <Text style={styles.buttonText}>Guardar</Text>
          </TouchableOpacity>
          <TouchableOpacity onPress={discardPicture}
style={styles.button}>
            <Text style={styles.buttonText}>Descartar</Text>
          </TouchableOpacity>
        </View>
      ) : (
        <>
          {images.length === 0 ? (
            <Camera
              style={styles.camera}
              type={type}
              ref={cameraRef}
              flashMode={flash}
            >
              <View style={styles.controls}>
```

```
                <TouchableOpacity
                  onPress={() => setType(
                    type === CameraType.back ? CameraType.front :
CameraType.back
                  )}
                  style={styles.iconButton}
                >
                  <Text style={styles.iconText}>🔄</Text>
                </TouchableOpacity>
                <TouchableOpacity
                  onPress={() => setFlash(
                    flash === Camera.Constants.FlashMode.off
                      ? Camera.Constants.FlashMode.on
                      : Camera.Constants.FlashMode.off
                  )}
                  style={styles.iconButton}
                >
                  <Text style={styles.iconText}>
                    {flash === Camera.Constants.FlashMode.off ? '⚡' :
'💡'}
                  </Text>
                </TouchableOpacity>
              </View>
            </Camera>
          ) : (
            <FlatList
              data={images}
              renderItem={renderImageCard}
```

```
              keyExtractor={(item, index) => index.toString()}
            />
          )}
          {images.length === 0 && (
            <TouchableOpacity onPress={takePicture}
style={styles.button}>
              <Text style={styles.buttonText}>Tomar foto</Text>
            </TouchableOpacity>
          )}
        </>
      )}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  camera: {
    flex: 1,
    width: '100%',
  },
  controls: {
    flexDirection: 'row',
```

```
    justifyContent: 'space-between',
    paddingHorizontal: 30,
    marginTop: 20,
  },
  button: {
    backgroundColor: 'orange',
    padding: 10,
    borderRadius: 5,
    alignItems: 'center',
    justifyContent: 'center',
    margin: 10,
  },
  buttonText: {
    color: '#fff',
    fontSize: 18,
  },
  iconButton: {
    padding: 10,
  },
  iconText: {
    fontSize: 24,
  },
  previewImage: {
    width: 300,
    height: 300,
    borderRadius: 10,
    marginBottom: 10,
```

```
},
annotationInput: {
  borderColor: 'gray',
  borderWidth: 1,
  borderRadius: 5,
  padding: 10,
  width: '90%',
  marginBottom: 10,
},
card: {
  width: '90%',
  padding: 20,
  backgroundColor: '#fff',
  borderRadius: 10,
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 2 },
  shadowOpacity: 0.1,
  shadowRadius: 10,
  marginBottom: 20,
  alignItems: 'center',
},
image: {
  width: '100%',
  height: 200,
  borderRadius: 10,
  marginBottom: 10,
},
```

```
  message: {
    fontSize: 18,
    color: 'black',
    marginBottom: 10,
  },
});
```

MediaButton.js

```
import { Text, View, StyleSheet, TouchableOpacity } from
'react-native';
import Icon from "react-native-vector-icons/Ionicons";



export function MediaButton({ label = 'none', icon= 'film', onPress})
{

  return (
    <TouchableOpacity onPress={onPress} style={styles.touchable}>
      <View style={styles.button_default}>
        <Icon name={icon} size={40} color="white" />
        <Text style={styles.text}>{label}</Text>
      </View>
    </TouchableOpacity>
  );
}
```

```
const styles = StyleSheet.create({
  button_default: {
    flexDirection: 'column',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: 'dodgerblue',
    height:120,
    width:120,
    margin:20
  },
  text: {
    fontSize: 20,
    fontWeight: 'bold',
    color: 'white'
  },
  touchable:{
    flex:1,
    alignItems: 'center'
  }
});
```

HomeScreen.js

```
import React, { useState, useEffect, useRef } from 'react';
import { Text, SafeAreaView, StyleSheet, View } from 'react-native';
import {  BarButton } from './barButton';
import { MediaButton } from './MediaButton';
import { Camera, CameraType } from 'expo-camera';
```

```
import * as MediaLibrary from 'expo-media-library';



export function HomeScreen({ navigation }){


  return(
    <>
      <View style={styles.content}>
        <MediaButton label="Fotografía" icon='camera' onPress={()=>
navigation.navigate('PhotoCamera')}/>
        <MediaButton label="Video" icon='videocam' onPress={()=>
navigation.navigate('VideoCamera')}/>
      </View>
      <View style={styles.bar}>
        <BarButton label="Multimedia" icon="film" state="selected"/>
        <BarButton label="Lista de archivos" icon="file-tray-full"
onPress={()=> navigation.navigate('Lista de Archivos')}/>
      </View>
    </>
  );
}


const styles = StyleSheet.create({
  bar: {
    backgroundColor: 'orange',
    flexDirection: 'row',
    alignItems: 'stretch',
    justifyContent: 'center',
```

```
    padding:0,
    height: 80
  },
  content:{
    flex:1,
    backgroundColor:"white",
    flexDirection: 'row',
    paddingVertical: 50
  }
});
```

barButton.js

```
import { Text, View, StyleSheet, TouchableOpacity } from
'react-native';
import Icon from "react-native-vector-icons/Ionicons";

var color = 'orange';

export function BarButton({ label = 'none', icon= 'film',
state='default', onPress}) {

  if (state == 'default'){
    return (
      <TouchableOpacity onPress={onPress} style={styles.touchable}>
        <View style={styles.button_default}>
          <Icon name={icon} size={30} color="white" />
          <Text style={styles.text}>{label}</Text>
        </View>
```

```
    </TouchableOpacity>
  );
  }
  else {
    return (
    <View style={styles.button_selected}>
      <Icon name={icon} size={30} color="white" />
      <Text style={styles.text}>{label}</Text>
    </View>
    );
  }
}

const styles = StyleSheet.create({
  button_default: {
    flex:1,
    flexDirection: 'column',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: 'orange'
  },
  button_selected: {
    flex:1,
    flexDirection: 'column',
    alignItems: 'center',
    justifyContent: 'center',
    backgroundColor: 'darkorange'
```

```
  },
  text: {
    fontSize: 16,
    fontWeight: 'bold',
    color: 'white'
  },
  touchable:{flex:1}
});
```