Control de documento

Nombre del proyecto	Healthec
Cierre de iteración	I5 24/03/2023
Generador por	Carlos Antonio Madrigal Trejo
Aprobado por	Carlos Antonio Madrigal Trejo
Alcance de la distribución del documento	Control interno para todo el proyecto.

Índice

Sobre este documento

La calidad se logra por medio de la revisión constante de las actividades que conducen desde la idea al producto. Al momento del cierre de una iteración es buen momento para hacer un alto, y evaluar lo logrado, los problemas encontrados y los retos a enfrentar.

El presente documento marca el final de la iteración I5, y contiene una evaluación de los artefactos y actividades realizadas durante la misma.

Se recogen también las impresiones y observaciones hechas durante el desarrollo de la iteración, así como el esfuerzo invertido en cada una de las disciplinas involucradas.

Resumen de la Iteración

Identificación

Código de la iteración	Fase a la que pertenece	Fecha de inicio	Fecha de cierre	Comentarios
I5	Implementación	20/03/23	24/03/23	Se completó correctamente.

Hitos especiales

En este Sprint se logró hacer un avance en el desarrollo del código relacionado con el menú principal y la meditación para dormir. Además se logró implementar un mejor diseño para el login de la aplicación.

Artefactos y evaluación

Artefacto	Meta (%)	Comentarios
PROG-02	100%	Se retomó el trabajo que se había hecho en el anterior sprint y se mejoraron aspectos visuales de la interfaz. Además se añadió un módulo individual para el registro.
PROG-03	100%	Se logró desarrollar en su totalidad la implementación del menú principal.
PROG-04	70%	Se avanzó con el desarrollo del activity de meditación.

Artefacto	Aspecto a evaluar	Evaluación	Comentarios
PROG-02	Programación	100%	Se cumplió en su totalidad y en tiempo la tarea.
PROG-03	Programación	100%	Se cumplió en su totalidad y en tiempo la tarea.
PROG-04	Programación	70%	Se logró avanzar considerablemente, se retomará en el siguiente sprint las tareas mínimas faltantes.

Riesgos y problemas (Riesgo con su identificador si sucedió o no sucedió)

Notas y observaciones

	Riesgo	Ocurrió
RIE-02	Fallas de hardware	No
RIE-04	Enfermedades	No
RIE-10	Falta de ética y moral del personal	No
RIE-12	Requisitos confusos o ambiguos	No
RIE-14	Ambiente laboral deficiente	No
RIE-16	Documentación deficiente	No
RIE-19	Estimación del tiempo inadecuada	No
RIE-21	Falta de comunicación con el cliente	No
RIE-22	Falta de claridad en los roles de actividades	No
RIE-23	Falta de experiencia del líder de proyecto	No

Asignación de recursos

Rol	Horas-Hombre	Desempeñado por(área)	Observaciones
Programador	2 horas.	Carlos Antonio Madrigal Trejo	Implementó una buena opción para el login.
Programador	3 horas.	Fernando Pérez Romero	Desarrolló buena lógica en la implementación del menú.
Programador	3 horas.	Omar Adrián Tapia Guzmán	Diseñó una buena interfaz amigable al usuario en el login.
Programador	2 horas.	Rubén Dario Vidaña	Realizó un buen desempeño en el diseño en el apartado de meditación para dormir.
Programador	2 horas.	Carlos Daniel Lopéz Romo	Realizó un buen avance en la programación de la meditación para dormir.

Anexos

Anexo A.

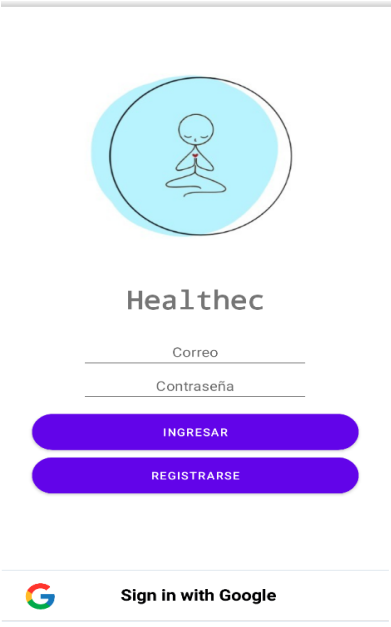
PROG-02 - Activity del Log In

LoginActivity.java

```
package mx.GPS.healthec;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.Task;

public class LoginActivity extends AppCompatActivity {

   //Referencia a los botones y otros controles en el layout
   Button btn_registrar, btn_ingresar;
   EditText edt_email, edt_password;
   ImageView google_login;

   GoogleSignInOptions gso;
   GoogleSignInClient gsc;


   //METODO ONCREATE, SE EJECUTA CUNADO LOGINACTIVITY SE INICIA
   @Override
   //Llama al método onCreate() de la clase base Activity utilizando el parámetro
savedInstanceState.
   //este método establece el diseño de la actividad y busca los elementos de la interfaz de
   // usuario para poder interactuar con ellos posteriormente en la aplicación.

   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_login);
       //Se le asigna a las variables el elemento creado por la computadora en el
layout---------//
      btn_ingresar = findViewById(R.id.btn_ingresar);
       btn_registrar = findViewById(R.id.btn_registrar);

       edt_email = findViewById(R.id.edt_email);
       edt_password = findViewById(R.id.edt_password);

       google_login = findViewById(R.id.google);

       gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
               .requestEmail()
               .build();
```

```
        gsc = GoogleSignIn.getClient(this, gso);

        //Listeners de los
botones-----------------------------------------------------------------//
        btn_ingresar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //SE DECLARA UNA INSTANCIA DE UserModel llamada usuario
                UserModel usuario;
                //Se intenta crear una nueva instancia de UserModel utilizando los valores
ingresados
                // en los campos de correo electrónico y contraseña de la interfaz de usuario.
                // Si se produce una excepción, se muestra un mensaje de error y se crea una
instancia de
                // UserModel con valores predeterminados ("error" y "-1").
                try{
                    usuario = new UserModel(-1, edt_email.getText().toString(),
                            edt_password.getText().toString());
                    Toast.makeText(LoginActivity.this, usuario.toString(),
Toast.LENGTH_SHORT).show();
                } catch( Exception e){
                    Toast.makeText(LoginActivity.this, "Es necesario rellenar todos los
campos",
                            Toast.LENGTH_SHORT).show();
                    usuario = new UserModel(-1, "error", "error");
                }

                //Se declara una instancia de DataBaseHealthec, que es una clase que maneja la
base de datos
                // SQL utilizada por la aplicación.
                DataBaseHealthec dataBaseHealthec = new
DataBaseHealthec(LoginActivity.this);


                //Se llama al método "addOne" de la instancia de DataBaseHealthec, pasando
como argumento
                // la instancia de UserModel creada anteriormente. Este método intenta agregar
el usuario a

                // la base de datos y devuelve un valor booleano que indica si la operación fue
exitosa o no.
                try{
                 boolean exist = dataBaseHealthec.exists(usuario);
                    if( exist ){
                        //codigo para entrar al menu principal donde se encuentran todas las
opciones
                       finish();
                        startActivity( new Intent(LoginActivity.this, MenuActivity.class));
                    }
                } catch (Exception e){
                    //codigo para representar que el usuario no existe
                   Toast.makeText(LoginActivity.this, "No existe ninguna cuenta con esos
datos",
                            Toast.LENGTH_LONG).show();
                }
            }
        });

//-------------------------------------------------------------------------------------------//
        btn_registrar.setOnClickListener(new View.OnClickListener() {

            //METODO ONCLICK este método se encarga de crear un nuevo usuario con los valores
            // ingresados en la interfaz de usuario y agregarlo a la base de datos de la
aplicación.
            @Override
```

```
            public void onClick(View view) {
                finish();
                Intent intent = new Intent(LoginActivity.this, RegistroActivity.class);
                startActivity(intent);
            }
        });

//-----------------------------------------------------------------------------------------//
        google_login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                SignIn();
            }
        });

    }
    private void SignIn() {
        Intent intent = gsc.getSignInIntent();
        startActivityForResult(intent, 100);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
{
        super.onActivityResult(requestCode, resultCode, data);

        if(requestCode == 100){
            Task<GoogleSignInAccount> task =
GoogleSignIn.getSignedInAccountFromIntent(data);
            try {
                task.getResult(ApiException.class);
                HomeActivity();
            } catch (ApiException e) {
                Toast.makeText(this, "Error 1", Toast.LENGTH_SHORT).show();
            }
        }
    }
    private void HomeActivity() {
        finish();
        Intent intent = new Intent(LoginActivity.this, RegistroActivity.class);
        startActivity(intent);
    }
}
```



RegistroActivity.java

```
package mx.GPS.healthec;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;

public class RegistroActivity extends AppCompatActivity {
   Button btn_registroAceptar;
   EditText edt_emailRegistro, edt_passwordRegistro, edt_nombreRegistro;

   GoogleSignInOptions gso;
   GoogleSignInClient gsc;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_registro);
       btn_registroAceptar = findViewById(R.id.btn_registroAceptar);
       edt_emailRegistro = findViewById(R.id.edt_correoRegistro);
       edt_passwordRegistro = findViewById(R.id.edt_correoRegistro);
       edt_nombreRegistro = findViewById(R.id.edt_correoRegistro);

       gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
               .requestEmail()
               .build();

       gsc = GoogleSignIn.getClient(this, gso);

       DataBaseHealthec dataBaseHealthec = new DataBaseHealthec(RegistroActivity.this);

       GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);

       if( account != null ){
           UserModel usuario;
           try{
               usuario = new UserModel(-1, account.getEmail(), null,
account.getDisplayName());
               Toast.makeText(RegistroActivity.this, usuario.toString(),
Toast.LENGTH_SHORT).show();
           }
           catch (Exception e){
               Toast.makeText(RegistroActivity.this, "Error con la cuenta de google",
                       Toast.LENGTH_SHORT).show();
               usuario = new UserModel(-1, "error", "error", "error");
           }

           boolean success = dataBaseHealthec.addOne(usuario);

           try{
```

```
                if(success){
                    finish();
                    startActivity( new Intent(RegistroActivity.this, MenuActivity.class));
                }
            }catch (Exception e){
                Toast.makeText(RegistroActivity.this, "No se pudo registrar correctamente",
                        Toast.LENGTH_LONG).show();
            }

        }

        btn_registroAceptar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                UserModel usuario;
                try{
                    usuario = new UserModel(-1, edt_nombreRegistro.getText().toString(),
                            edt_emailRegistro.getText().toString(),
                            edt_nombreRegistro.getText().toString());
                    Toast.makeText(RegistroActivity.this, usuario.toString(),
Toast.LENGTH_SHORT).show();
                } catch( Exception e){
                    Toast.makeText(RegistroActivity.this, "Es necesario rellenar todos los
campos",
                            Toast.LENGTH_SHORT).show();
                    usuario = new UserModel(-1, "error", "error", "error");
                }

                try{
                boolean success = dataBaseHealthec.addOne(usuario);
                    if(success){
                        finish();
                        startActivity( new Intent(RegistroActivity.this,
MenuActivity.class));
                    }
                }catch (Exception e){
                    Toast.makeText(RegistroActivity.this, "No se pudo registrar
correctamente",
                            Toast.LENGTH_LONG).show();
                }
            }
        });
    }

}
```

AndroidManifest.xml

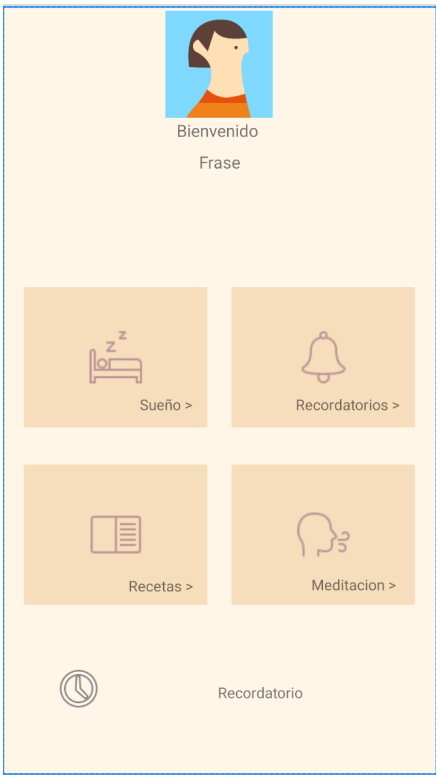
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools">

   <application
       android:allowBackup="true"
       android:dataExtractionRules="@xml/data_extraction_rules"
       android:fullBackupContent="@xml/backup_rules"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:supportsRtl="true"
       android:theme="@style/Theme.Healthec"
       tools:targetApi="31">
       <activity
           android:name=".MainActivity"
           android:exported="false" />
       <activity
           android:name=".LoginActivity"
           android:exported="true">
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
   </application>

</manifest>
```

Anexo B.

PROG-03 Programar Activity del Menú

MenuActivity.java

```
package mx.GPS.healthec;

import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

public class MenuActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu);
    }

    public void btnModuloSueñoClick ( View v ){
        startActivity(new Intent( MenuActivity.this, SleepActivity.class));
    }

    public void btnModuloRecetasClick ( View v ){
        startActivity(new Intent( MenuActivity.this, RecetasActivity.class));
    }

    public void btnModuloRecordatorioClick( View v ){
        startActivity(new Intent( MenuActivity.this, RecordatoriosActivity.class));
    }

    public void btnModuloMeditacionClick( View v ){
        startActivity(new Intent( MenuActivity.this, MeditacionActivity.class));
    }
}
```

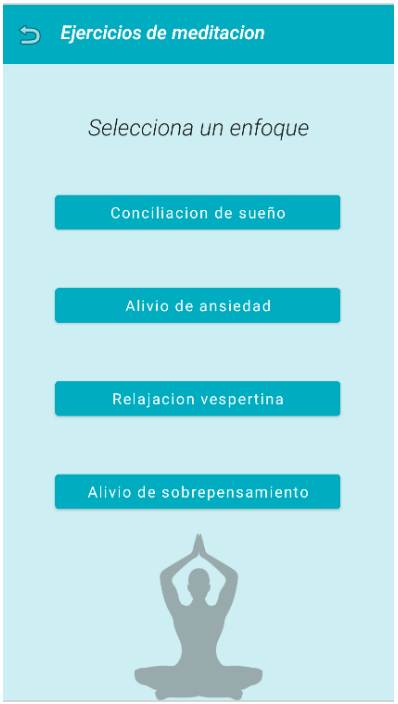


Anexo C.

PROG-04 Programar Activity de Meditación para dormir

MeditacionActivity.java

```
package mx.GPS.healthec;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
public class MeditacionActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_meditacion);
        ImageView imageView = findViewById(R.id.imageView4);
        float alphaValue = 0.5f; // establece el valor de transparencia entre 0 (totalmente
transparente) y 1 (totalmente opaco)
        imageView.setAlpha(alphaValue);
    }

    //boton para entrar a la seccion conciliacion del sueño
    public void btnSueño( View v ) { setContentView(R.layout.activity_sueño);

    }
    //boton para entrar a la seccion alvio de ansiendad
    public void btnAnsiedad( View v ) { setContentView(R.layout.activity_ansiedad);

    }
    //boton para entrar a la seccion relajacion vespertina
    public void btnRelajacion( View v ) { setContentView(R.layout.activity_relajacion);

    }
    //boton para entrar a la seccion de alivio de sobrepensamiento
    public void btnAlivio( View v ) {setContentView(R.layout.activity_alivio);

    }
    //boton para regresar al menu principal
    public void btnRegresar( View v ) {
        setContentView(R.layout.activity_menu);
    }

}
```



Anexo D.

Tabla de McCall

CAPACIDAD	FACTOR	Métrica	Calificación
Operación	Corrección: Grado de cumplimiento de las especificaciones y objetivos del usuario	Compleción: Grado en que se logró implementar en la app los requerimientos especificados por las necesidades del usuario final	5
		Consistencia: Los requerimientos del usuario final cumplen con las técnicas de documentación por norma	3
		Trazabilidad: Los elementos del diseño de la app se identifican a partir de los requerimientos del usuario final	4
	Confiabilidad: Grado en el sistema está disponible para usarse.	Complejidad: La complejidad de las funciones de la app interfieren con la disponibilidad para usarse	4
		Consistencia: El diseño de la app permite el usuario utilizarla en cualquier momento	5
		Tolerancia a errores: Las fallas en la disponibilidad de la app interfieren en las necesidades de los usuarios	3
	Usabilidad: Grado de esfuerzo necesario que se requiere para aprender a utilizarlo.	Facilidad de formación: Facilidad en que el usuario puede aprender a utilizar la app	5
		Operatividad: La app cuenta con una guía para su facilidad de operación	3
	Integridad o Seguridad: Grado en el que se controla el acceso al programa o los datos por usuarios no autorizados.	Facilidad de auditoría: La app cuenta con las autenticaciones necesarias para impedir a usuarios no autorizados acceder a información sensible	4
		Instrumentación: La app cuenta con herramientas para identificar automáticamente brechas de seguridad en la información del usuario	5
		Seguridad: La app cuenta con elementos de protección para asegurar la protección de información importante.	5
	Eficiencia o Performance:	Concisión: Nivel de optimización de código en las funciones de la app.	3

	Cantidad de recursos y código requeridos por un programa para realizar una función.	Eficiencia de ejecución: Nivel de eficiencia de ejecución en las funciones de la app.	4
		Operatividad: Nivel de facilidad de operación en las funciones de la app.	4
Transición	Portabilidad: Grado que mide el esfuerzo para migrar un programa de un entorno de operación a otro.	Auto documentación: El código de la app cuenta con la claridad necesaria para portar en otro entorno sin documentación.	5
		Generalidad:La app en general es capaz de ser migrada a otro entorno.	3
		Modularidad: El código de la app es capaz de ser migrado por módulos.	3
	Reusabilidad: Grado de esfuerzo requerido para que el programa o una de sus partes pueda ser utilizado en otro proyecto.	Autodocumentación: El código de la app cuenta con la claridad necesaria para que otro desarrollador pueda utilizarlo en otro proyecto.	5
		Independencia hardware: La app o cualquier parte de su código puede ser utilizado en cualquier modelo celular.	4
		Independencia del sistema: La app o cualquier parte de su código puede ser utilizado en cualquier sistema operativo o versión vigente.	3
	Interoperabilidad: Grado de esfuerzo dedicado para que un sistema o programa pueda operar conjuntamente con otro.	Estd. Comunicaciones: Grado de uso de estándares para que la app pueda operar con otro software conjuntamente.	5
		Estandarización de datos: Nivel de manejo de interoperabilidad con otros softwares.	5
Revisión	Facilidad Mantenimiento: Esfuerzo requerido para localizar y	Consistencia: Nivel de documentación empleada para reducir el esfuerzo requerido en corrección de errores.	4
		Modularidad: Nivel de modularidad de las funcionalidades para su fácil mantenimiento.	5

	corregir un error en un programa en funcionamiento.	Simplicidad: Nivel de simplicidad del código para su fácil mantenimiento por cualquier desarrollador incluso ajeno al proyecto.	4
	Flexibilidad: Esfuerzo requerido para modificar un software en funcionamiento.	Capacidad de expansión: Grado permitido para ampliar la app en funcionamiento.	3
		Complejidad: Nivel de complejidad para ampliar la app en funcionamiento.	3
		Consistencia: Nivel de documentación empleada para poder ampliar la app en funcionamiento.	5
	Facilidad de Prueba: Grado de esfuerzo requerido para probar un programa verificando que realice adecuadamente sus funciones.	Auto documentación: La app puede ser probada debido a la claridad del código proporcionada por la documentación realizada.	3
		Facilidad de auditoría: Nivel de facilidad de auditoría de las funciones de la app.	4
		Instrumentación: Nivel de aplicación de herramientas que apliquen pruebas automatizadas a la app.	5

Glosario de términos

Hardware: Conjunto de elementos físicos o materiales que constituyen una computadora o un sistema informático.

Software: Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Interfaz: En el contexto de la programación, una interfaz es un conjunto de métodos y propiedades que se pueden implementar en una clase para permitir la comunicación con otras clases y componentes. Una interfaz define un contrato que debe cumplirse para que las clases puedan comunicarse entre sí de manera efectiva.

Hash: En el contexto de la seguridad informática, un hash es un valor numérico que se genera a partir de un conjunto de datos utilizando un algoritmo de hash. Los algoritmos de hash son funciones criptográficas que toman una entrada de datos y producen una salida de longitud fija que es única para esa entrada. Los hashes se utilizan para verificar la integridad de los datos, proteger las contraseñas y garantizar la autenticidad de los mensajes.

Salting: El salting (en español, "salteado") es una técnica de seguridad utilizada en la criptografía de contraseñas. Consiste en agregar una cadena aleatoria de datos a la contraseña antes de realizar el hash. El salting hace que las contraseñas sean más difíciles de romper mediante ataques de fuerza bruta o de diccionario, ya que el hash resultante será diferente para cada usuario. El proceso de salting y hashing juntos se conoce como "salting and hashing" o "hashing with salt".

Métodos: En programación, los métodos son bloques de código que realizan una tarea específica. Los métodos pueden recibir parámetros y devolver valores, lo que les permite

interactuar con otros componentes del programa. Los métodos se utilizan para organizar y modularizar el código, lo que facilita el mantenimiento y la reutilización del mismo.

Base de datos: Una base de datos es un sistema de almacenamiento y recuperación de información. Las bases de datos se utilizan comúnmente en aplicaciones informáticas para almacenar información estructurada y permitir la recuperación y manipulación de los datos de manera eficiente.

Sensor: Un sensor es un componente de hardware que detecta y responde a cambios en su entorno. En el contexto de la programación móvil, los sensores se utilizan comúnmente en los dispositivos móviles para detectar la ubicación, el movimiento, la luz, la proximidad y otros aspectos del entorno del dispositivo. Los datos del sensor se pueden utilizar para crear experiencias interactivas en tiempo real y para mejorar la precisión de las aplicaciones.

Tareas en segundo plano: Las tareas en segundo plano son operaciones tipo "desencadenar y olvidar" y su progreso de ejecución no tiene ningún impacto en la interfaz de usuario o el proceso de llamada. Esto significa que el proceso de llamada no espera a la finalización de las tareas.