Control de documento

Nombre del proyecto	Healthec
Cierre de iteración	I6 31/03/2023
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 I6, 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
I6	Implementación	27/03/23	31/03/23	Se completó correctamente.

Hitos especiales

En este Sprint se logró hacer un avance en el desarrollo del código relacionado con las recomendaciones de recetas saludables. Además se realizó un avance en la sección de meditación, la implementación de mensajes motivacionales en el menú e inserción de las frases en la base de datos.

Artefactos y evaluación

Artefacto	Meta (%)	Comentarios
PROG-03	100%	Se añadió una función extra.
PROG-04	75%	Se logró desarrollar los botones para dirigirse al respectivo activity de cada meditación.
PROG-05	40%	Se avanzó con el desarrollo del activity de recetas saludables.

Artefacto	Aspecto a evaluar	Evaluación	Comentarios
PROG-03	Programación	100%	Se añadió una función extra.
PROG-04	Programación	75%	Se logró avanzar poco en este sprint, se retomará en el siguiente.
PROG-05	Programación	40%	Se logró avanzar considerablemente, se retomará en el siguiente sprint las tareas 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	Añadió las recetas a la base de datos.
Programador	3 horas.	Fernando Pérez Romero	Desarrolló la lógica para el despliegue de las recetas.
Programador	3 horas.	Omar Adrián Tapia Guzmán	Implementó una función extra para el login.
Programador	2 horas.	Rubén Dario Vidaña	Realizó un buen diseño de las recetas saludables.
Programador	2 horas.	Carlos Daniel Lopéz Romo	Realizó un avance en la programación de la meditación.

Anexos

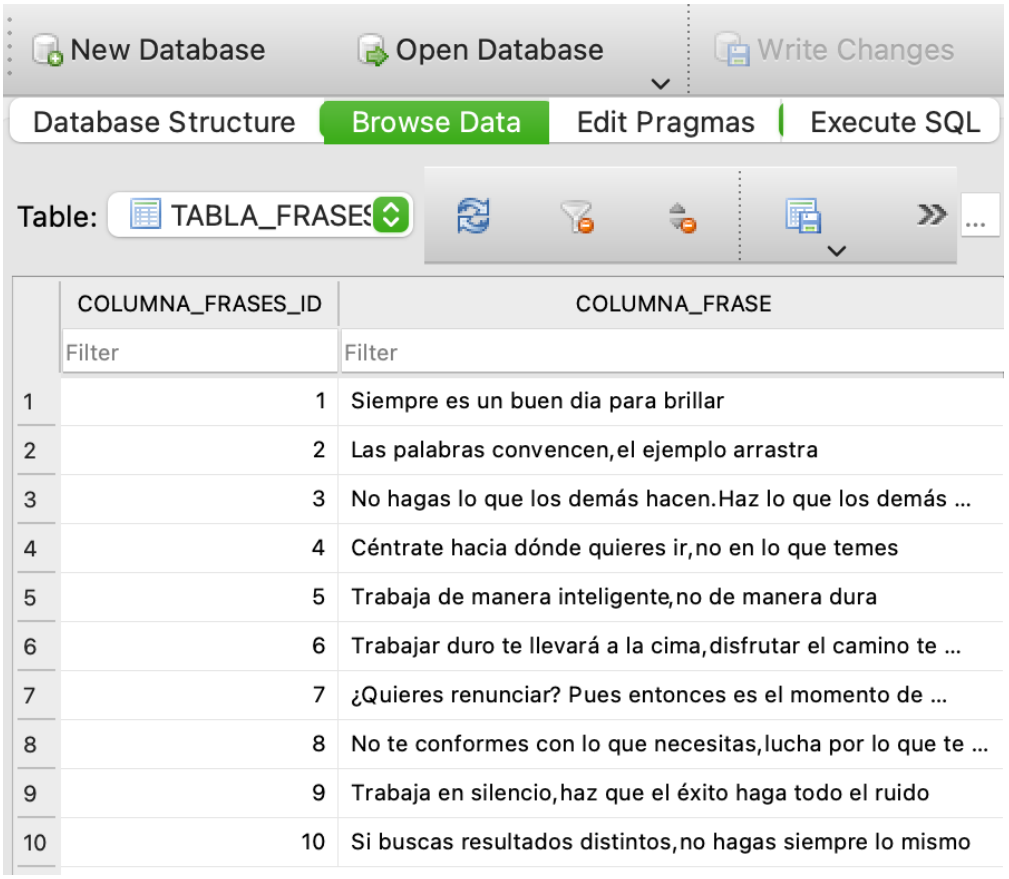
Anexo A.

PROG-03 Programar Activity del Menú

MenuActivity.java

```
package mx.GPS.healthec;

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

Creación de tabla e inserción de datos para las frases del MenuActivity.java



	COLUMNA_FRASES_ID	COLUMNA_FRASE
	Filter	Filter
1	1	Siempre es un buen dia para brillar
2	2	Las palabras convencen,el ejemplo arrastra
3	3	No hagas lo que los demás hacen.Haz lo que los demás ...
4	4	Céntrate hacia dónde quieres ir,no en lo que temes
5	5	Trabaja de manera inteligente,no de manera dura
6	6	Trabajar duro te llevará a la cima,disfrutar el camino te ...
7	7	¿Quieres renunciar? Pues entonces es el momento de ...
8	8	No te conformes con lo que necesitas,lucha por lo que te ...
9	9	Trabaja en silencio,haz que el éxito haga todo el ruido
10	10	Si buscas resultados distintos,no hagas siempre lo mismo

Anexo B.

PROG-04 Programar Activity de Meditación para dormir

activity_meditacion_para_dormir.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <TextView
       android:id="@+id/tituloTextView"
       android:text="¡Bienvenido/a a tu meditación para dormir!"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="20sp"
       android:layout_gravity="center_horizontal"
       android:layout_marginTop="24dp"
       android:layout_marginBottom="24dp"/>

   <TextView
       android:id="@+id/instruccionesTextView"
       android:text="Comencemos con la meditación. Siéntate en una posición cómoda y cierra
los ojos..."
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:textSize="16sp"
       android:layout_gravity="center_horizontal"/>

   <Button
       android:id="@+id/empezarButton"
       android:text="Empezar"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_gravity="center_horizontal"
       android:layout_marginTop="24dp"/>

</LinearLayout>
```

MeditacionParaDormirActivity.java

```
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MeditacionParaDormirActivity extends AppCompatActivity {

   private TextView tituloTextView;
   private TextView instruccionesTextView;
   private Button empezarButton;

   private Handler handler;

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

       // Obtener referencias a los elementos de la vista
       tituloTextView = findViewById(R.id.tituloTextView);
       instruccionesTextView = findViewById(R.id.instruccionesTextView);
       empezarButton = findViewById(R.id.empezarButton);

       // Crear el handler para los tiempos de espera
```

```
        handler = new Handler();

        // Configurar el click listener para el botón
        empezarButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                empezarMeditacion();
            }
        });
    }

    private void empezarMeditacion() {
        // Cambiar el texto del título y las instrucciones
        tituloTextView.setText("¡Meditación en progreso!");
        instruccionesTextView.setText("Comienza por prestar atención a tu respiración.
Siente cómo entra y sale el aire por tus fosas nasales. No trates de controlar la
respiración, simplemente obsérvala con atención. Si tu mente divaga, simplemente vuelve a
centrarte en la respiración. Haz esto durante unos minutos...");

        // Esperar un momento para que el usuario se acomode
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                //
```

Anexo C.

PROG-05 Programar Activity de Recetas Saludables

DatabaseHealthec.java

```
public List<RecetasModel> getRecipes(){
        List<RecetasModel> returnList = new ArrayList<>();
        //obtiene la información de la base de datos
        String queryString = "SELECT " + COLUMNA_RECETARIOS_INGREDIETNES + " FROM " +
TABLA_RECETARIOS;
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(queryString, null);

        if(cursor.moveToFirst()){
            //Se hace un bucle a traves del cursor y crea un nuevo objeto de UserModel los
cuales se pondran en la lista que se retorna.
            do{
                int recetaID = cursor.getInt(0);
                String recetaNombre = cursor.getString(1);
                String recetaPasos = cursor.getString(2);
                String recetaIngredientes = cursor.getString(3);

                RecetasModel receta = new RecetasModel(recetaID, recetaNombre, recetaPasos,
recetaIngredientes);
                returnList.add(receta);

            } while (cursor.moveToNext());
        } else {
            //Fallo, no anade nada a la lista
        }

        cursor.close();
        db.close();
        return returnList;
    }

INSERT INTO TABLA_RECETARIOS (COLUMNA_RECETARIOS_ID,
COLUMNA_RECETARIOS_RECETA, COLUMNA_RECETARIOS_PASOS, COLUMNA_RECETARIOS_INGREDIENTES)
VALUES
(1,'Miso Soup','1. Bring the water and dashi granules to a boil.
2. Reduce the heat and whisk in miso paste.
3. Stir in the tofu and green onions.
4. Simmer and serve.','4 cups water
2 teaspoons dashi granules
3 tablespoons miso paste
1 (8 ounce) package silken tofu, diced
2 green onions, sliced diagonally into 1/2 inch pieces'),
(2,'Baked kale chips','1.Preheat an oven to 300 degrees F (150 degrees C). Line a rimmed
baking sheet with parchment paper.
2.With a knife or kitchen shears carefully remove kale leaves from the thick stems and tear
into bite size pieces.
3.Wash and thoroughly dry kale with a salad spinner.
4.Wash and thoroughly dry kale with a salad spinner.
5.Bake until the edges start to brown but are not burnt, 20 to 30 minutes.
6.Enjoy!','1 bunch kale.
1 tablespoon olive oil.
1 teaspoon flaked sea salt.'),
(3,'Sarahs Homemade Applesauce','1.Combine apples, water, sugar, and cinnamon in a
saucepan; cover and cook over medium heat until apples are soft, about 15 to 20 minutes.
2.Allow apple mixture to cool, then mash with a fork or potato masher until it is the
consistency you like.','4 apples - peeled, cored and chopped.
¾ cup water.
¼ cup white sugar.
½ teaspoon ground cinnamon.'),
(4,'Homemade BBQ Sauce','1.Gather all ingredients.
2.Combine brown sugar, ketchup, vinegar, water, and Worcestershire sauce in a blender.
Season with mustard, paprika, salt, pepper, and hot pepper sauce. Blend until smooth.
3.Enjoy!','1 ½ cups brown sugar.
1 ½ cups ketchup.
½ cup red wine vinegar.
½ cup water.
1 tablespoon Worcestershire sauce.
2 ½ tablespoons dry mustard.
```

```
2 teaspoons paprika.
2 teaspoons salt.
1 ½ teaspoons black pepper.
2 dashes hot pepper sauce.'),
(5,'Homemade Crispy Hash Browns','2 medium russet potatoes, shredded.
½ medium onion, finely chopped.
¼ cup all-purpose flour.
1 egg.
1 cup oil for frying, or as needed.
salt and pepper to taste.','1.Rinse shredded potatoes until water is clear, then drain and
squeeze dry.
2.Place shreds in a bowl and mix in the onion, flour, and egg until evenly distributed.
3.Heat about 1/4 inch of oil in a large heavy skillet over medium-high heat. When oil is
sizzling hot, place potatoes into the pan in a 1/2 inch thick layer. Cover the whole bottom
of the pan, or make separate piles like pancakes.
4.Cook until nicely browned on the bottom, then flip over and brown on the other side. It
should take at least 5 minutes per side. If you are cooking them in one big piece, it can
be cut into quarters for easier flipping.
5.Remove from pan, and drain on paper towels. Season with salt and pepper and serve
immediately.
6.Serve hot and enjoy!');
```

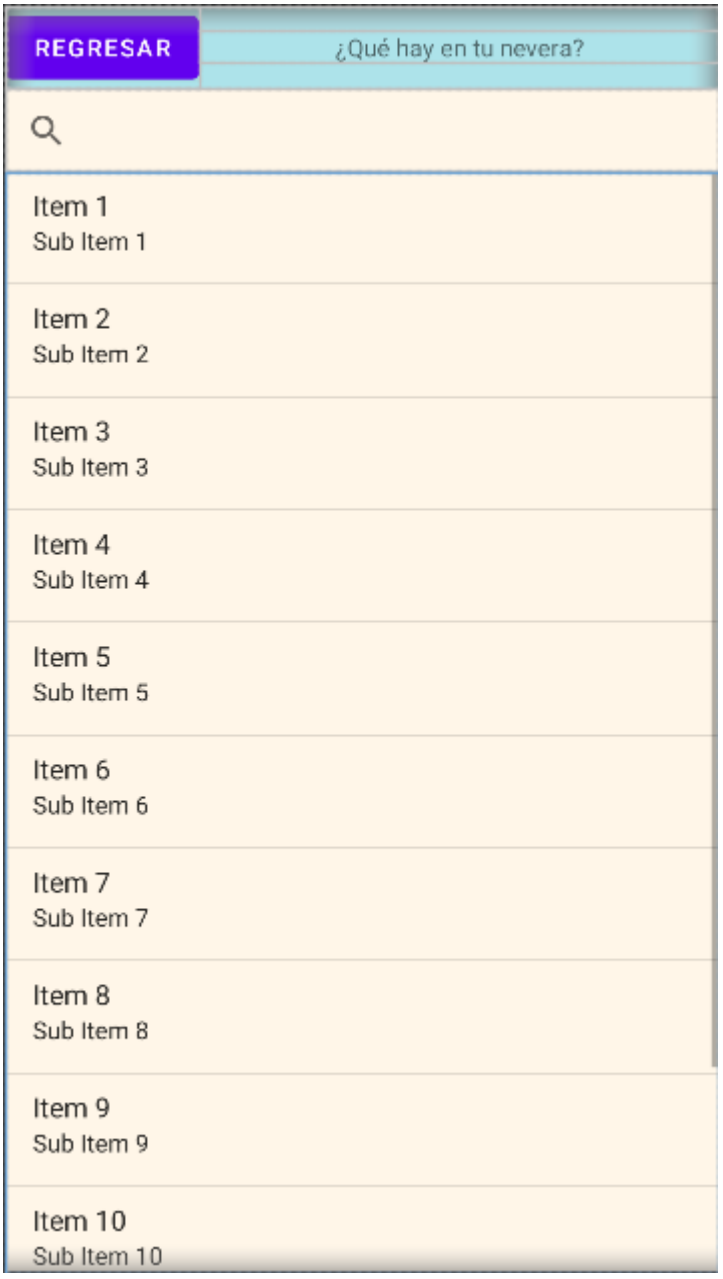
RecetasModel.java

```
package mx.GPS.healthec;

public class RecetasModel {
    private int idReceta;

    private String nombreReceta;

    private String pasosReceta;

    private String ingredientesReceta;

      public RecetasModel(int idReceta, String nombreReceta, String pasosReceta, String
ingredientesReceta) {
        this.idReceta = idReceta;
        this.nombreReceta = nombreReceta;
        this.pasosReceta = pasosReceta;
        this.ingredientesReceta = ingredientesReceta;
    }

    public RecetasModel() {
    }

    @Override
    public String toString() {
        return "MenuModel{" +
                "idReceta=" + idReceta +
                ", nombreReceta='" + nombreReceta + '\'' +
                ", pasosReceta='" + pasosReceta + '\'' +
                ", ingredientesReceta='" + ingredientesReceta + '\'' +
                '}';
    }

    public int getIdReceta() {
        return idReceta;
    }

    public void setIdReceta(int idReceta) {
        this.idReceta = idReceta;
    }

    public String getNombreReceta() {
        return nombreReceta;
    }

    public void setNombreReceta(String nombreReceta) {
        this.nombreReceta = nombreReceta;
    }

    public String getPasosReceta() {
        return pasosReceta;
```

```
    }

    public void setPasosReceta(String pasosReceta) {
        this.pasosReceta = pasosReceta;
    }

    public String getIngredientesReceta() {
        return ingredientesReceta;
    }

    public void setIngredientesReceta(String ingredientesReceta) {
        this.ingredientesReceta = ingredientesReceta;
    }
}
```

RecetasActivity.java

```
package mx.GPS.healthec;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import org.checkerframework.checker.units.qual.A;

import java.util.List;

public class RecetasActivity extends AppCompatActivity {
    //Constantes
    ListView lv_recetasList;
    ArrayAdapter recetasArrayAdapter;
    DataBaseHealthec db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recetas);

        //ListView
        lv_recetasList = findViewById(R.id.lv_recetas);

        //Llenado del ViewList
        db = new DataBaseHealthec(RecetasActivity.this);
               recetasArrayAdapter  =  new  ArrayAdapter<RecetasModel>(RecetasActivity.this,
android.R.layout.simple_list_item_1, db.getRecipes());
        lv_recetasList.setAdapter(recetasArrayAdapter);

    }
}
```


REGRESAR
¿Qué hay en tu nevera?
Item 1
Sub Item 1
Item 2
Sub Item 2
Item 3
Sub Item 3
Item 4
Sub Item 4
Item 5
Sub Item 5
Item 6
Sub Item 6
Item 7
Sub Item 7
Item 8
Sub Item 8
Item 9
Sub Item 9
Item 10
Sub Item 10

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	3
		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.	3
	Eficiencia o Performance: Cantidad de recursos y código requeridos por un programa para realizar una función.	Concisión: Nivel de optimización de código en las funciones de la app.	3
		Eficiencia de ejecución: Nivel de eficiencia de ejecución en las funciones de la app.	4
		Operatividad: Nivel de facilidad de operación en las	3

		funciones de la app.	
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.	2
		Modularidad: El código de la app es capaz de ser migrado por módulos.	2
	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.	4
		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 corregir un error en un programa en funcionamiento.	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.	2
		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.	2
		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.	5
		Instrumentación: Nivel de aplicación de herramientas que apliquen pruebas automatizadas a la app.	3

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.