



ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DEPARTAMENTO DE INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

MAESTRÍA EN COMPUTACIÓN

DESARROLLO DE APLICACIÓN MÓVIL EN ANDROID STUDIO COMO AGENTE DE RESPUESTA DEL PROYECTO

PIS2002

ARROYO AUZ CHRISTIAN XAVIER

[christian.arroyo@epn.edu.ec](mailto:christian.arroyo@epn.edu.ec)

Quito, febrero 2024



# OBJETIVOS DE LA APP

- Presentar la interfaz gráfica de usuario (GUI) de la aplicación móvil que conforma el agente de respuesta.
- Determinar mejoras a funcionalidades existentes y establecer funcionalidades nuevas que podrán ser incluidas en la aplicación.
- Desarrollar e implementar la aplicación móvil considerando los riesgos de seguridad definidos en la fase de diseño.
- Establecer los parámetros claves de cada una de las fuentes de datos (conductor, vehículo, condiciones meteorológicas y flujo de tráfico)

# INTRODUCCIÓN

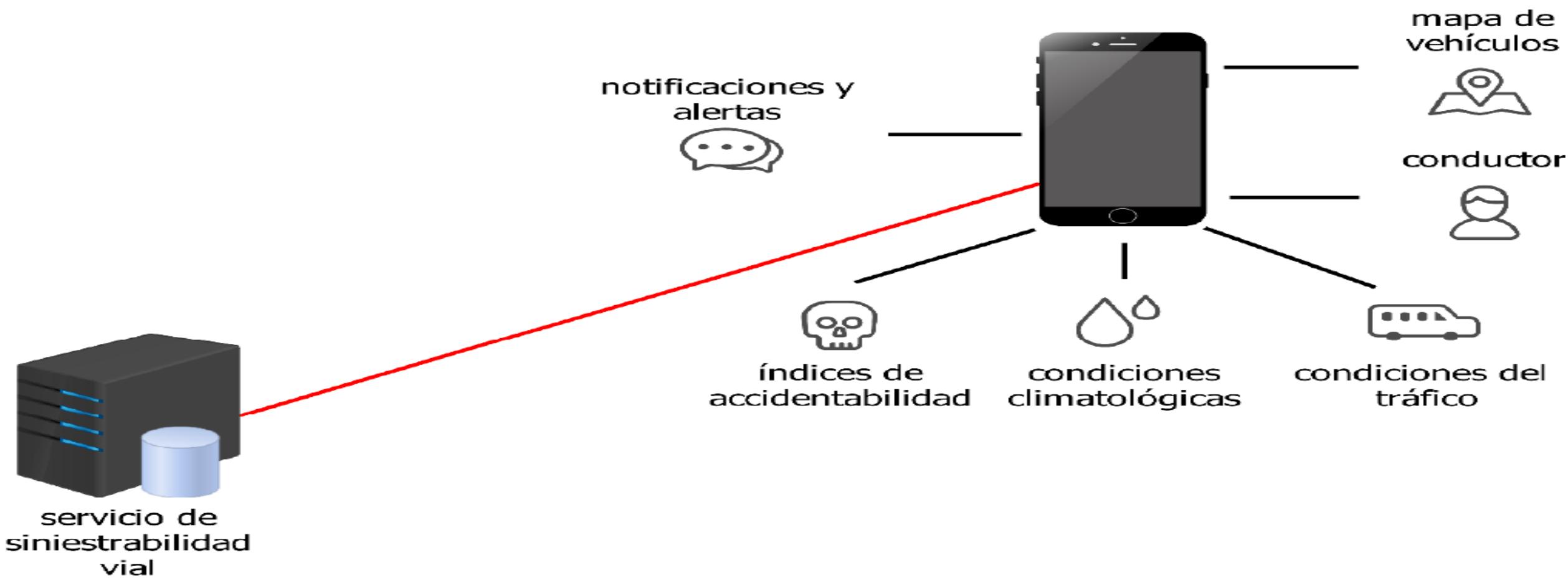
- El agente de respuesta forma parte de un conjunto de sistemas autónomos para la gestión de accidentabilidad vehicular.
- El agente de respuesta se encarga de solicitar información procesada al servicio de siniestralidad vehicular (SIGOAVE).
- El agente de respuesta es una aplicación móvil que presenta el índice de accidentabilidad del vehículo y ciertos parámetros claves del conductor, vehículo, condiciones climatológicas y del flujo de tráfico que sustentan ese índice. Además, la aplicación genera notificaciones y alertas en base a la información provista por SIGOAVE.
- Una de las principales características de esta aplicación móvil es que funcione de manera autónoma, pero pueda trabajar en sinergia con la aplicación móvil del agente de adquisición.



ESCUELA  
POLÍTÉCNICA  
NACIONAL

# INTRODUCCIÓN

- La siguiente figura presenta las principales funcionalidades de la aplicación móvil definidas en la fase de diseño.



# ALCANCE

La aplicación móvil del agente de respuesta incluye lo siguiente:

- Una pantalla o “activity” para gestionar el inicio de sesión por medio de credenciales (nombre de usuario y contraseña).
- Pantallas para presentar los siguientes parámetros claves:
  - Vehículo: velocidad, revoluciones por minuto, nivel de gasolina, temperatura del motor y ángulo de inclinación del volante.
  - Condiciones meteorológicas: lugar o zona, temperatura, precipitación, humedad, y velocidad del viento.
  - Flujo de tráfico: ubicación, velocidad promedio, número de vehículos, ocupación promedio y dirección del flujo.
  - Conductor: edad, genero, uso de lentes, estado de licencia y puntos disponibles, y condiciones médicas.
- Una pantalla que incluye un mapa para ubicar los vehículos que se encuentran en la zona.
- Cada una de las pantallas presenta como encabezado información relacionada al índice de accidentabilidad.



# RESULTADOS – PARTE 1

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Creación de un nuevo proyecto en Android Studio (Actividad en Blanco).
  - Selección de nombre, ubicación, lenguaje y SDK mínimo del proyecto.
  - Diseño de la Interfaz gráfica principal de la aplicación.



# RESULTADOS – PARTE 1

New Project X

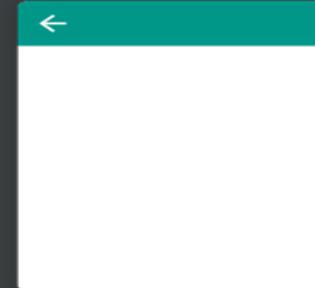
Templates

Phone and Tablet

Wear OS

Android TV

Automotive



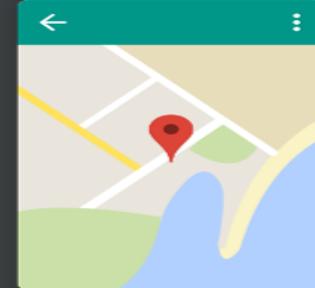
Empty Activity



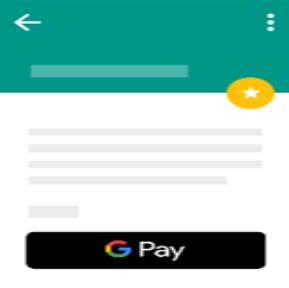
Fullscreen Activity



Google AdMob Ads Activity



Google Maps Activity



Google Pay Activity



Login Activity

Previous Next Cancel Finish



# RESULTADOS – PARTE 1

New Project

X

## Empty Activity

Creates a new empty activity

Name

Package name

Save location

Language

Minimum SDK

Your app will run on approximately **94.4%** of devices.  
[Help me choose](#)

Use legacy android.support libraries

Using legacy android.support libraries will prevent you from using  
the latest Play Services and Jetpack libraries

Previous

Next

Cancel

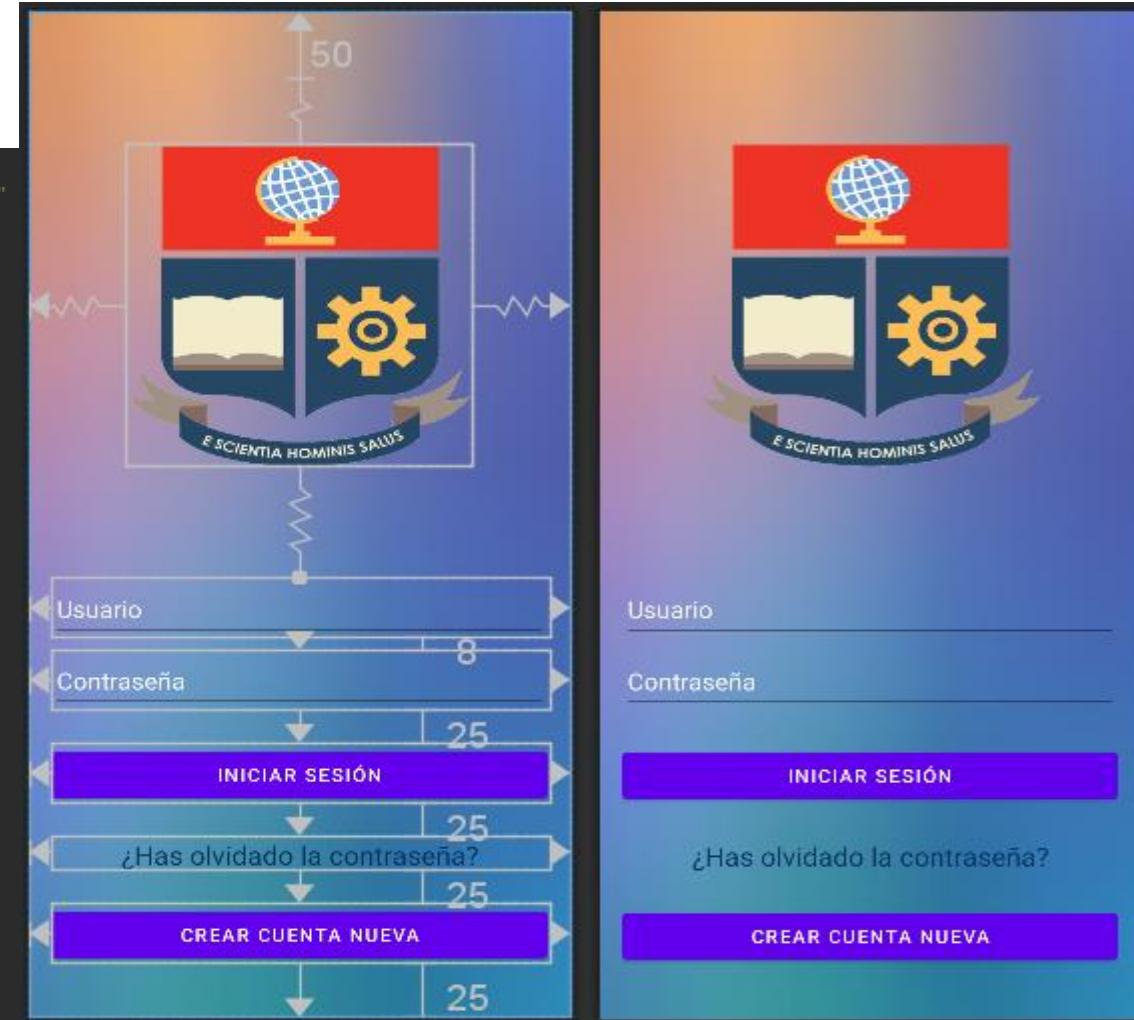
Finish

# RESULTADOS – PARTE 1

- A continuación se muestra segmentos de código referente a la interfaz gráfica de login.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:background="@drawable/imagen_fondo"
    tools:context=".clases.login.LoginActivity"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:id="@+id/contenedorLogin"
    android:paddingBottom="16dp"
    android:paddingRight="16dp"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    tools:ignore="ExtraText">

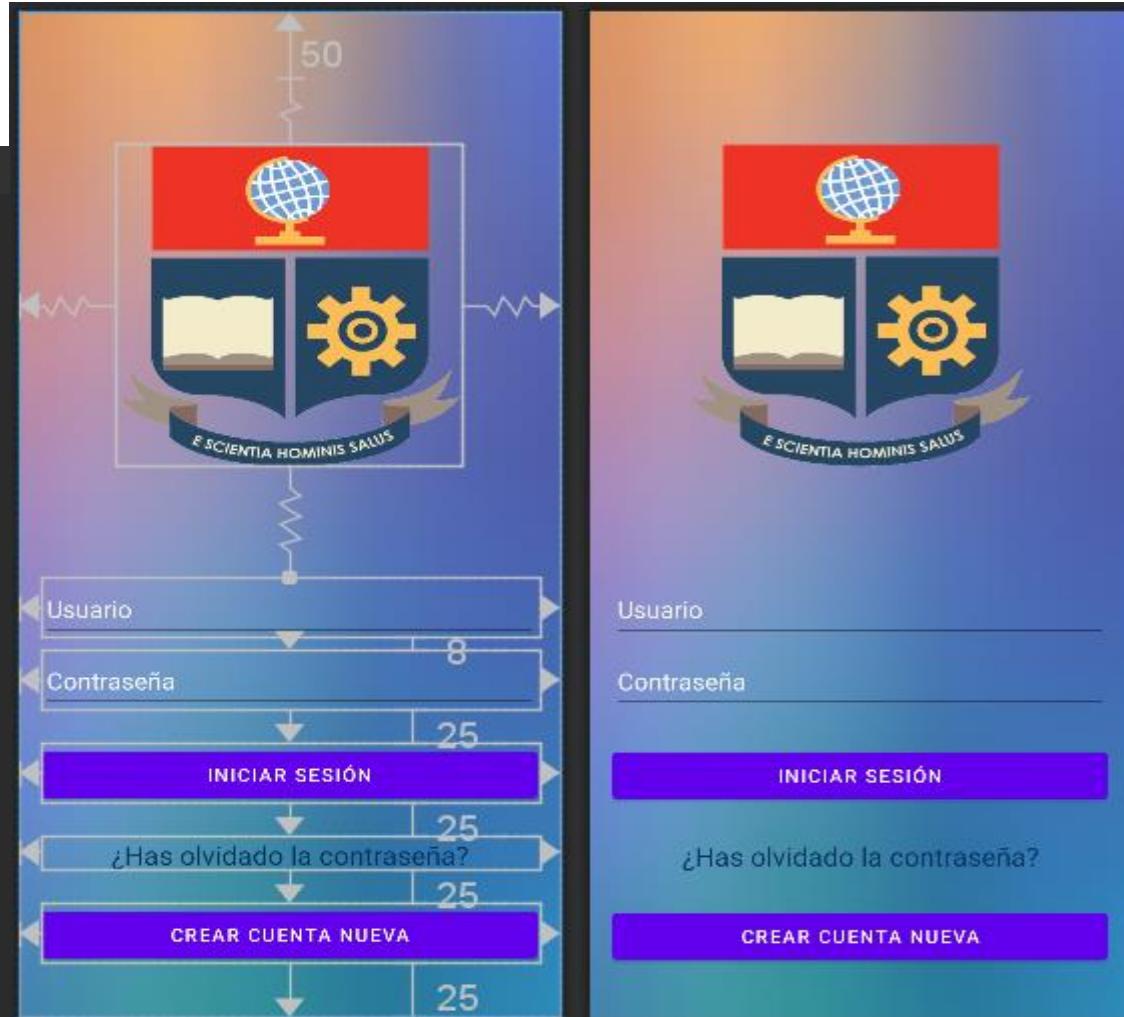
    <ImageView
        android:id="@+id/imagenLogotipo"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_marginTop="50dp"
        android:layout_marginBottom="50dp"
        android:contentDescription="TODO"
        android:foreground="@drawable/logo_epn"
        app:layout_constraintBottom_toTopOf="@+id/txtUsuario"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="ContentDescription,HardcodedText" />
```



# RESULTADOS – PARTE 1

- Para el diseño general de la aplicación se procedió a diseñar los colores, tamaños de letra, fuente e iconos.

```
1  <!--suppress XmlUnusedNamespaceDeclaration --&gt;
2
3 &lt;resources xmlns:tools="http://schemas.android.com/tools"&gt;
4
5   &lt;style name="Theme.PIS_20_02" parent="Theme.MaterialComponents.DayNight.DarkActionBar"&gt;
6
7     &lt;item name="android:statusBarColor"&gt;?attr/colorPrimaryVariant&lt;/item&gt;
8
9     &lt;item name="colorSecondaryVariant"&gt;@color/color_teal_700&lt;/item&gt;
10    &lt;item name="colorPrimaryVariant"&gt;@color/color_purple_700&lt;/item&gt;
11    &lt;item name="colorPrimary"&gt;@color/color_purple_500&lt;/item&gt;
12    &lt;item name="colorSecondary"&gt;@color/color_teal_200&lt;/item&gt;
13    &lt;item name="colorOnSecondary"&gt;@color/color_negro&lt;/item&gt;
14    &lt;item name="colorOnPrimary"&gt;@color/color_blanco&lt;/item&gt;
15
16  &lt;/style&gt;
17
18  &lt;style name="Theme.PIS_20_02.NoActionBar"&gt;
19
20    &lt;item name="windowActionBar"&gt;false&lt;/item&gt;
21    &lt;item name="windowNoTitle"&gt;true&lt;/item&gt;
22
23  &lt;/style&gt;
24
25  &lt;style name="Theme.PIS_20_02.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" /&gt;
26
27  &lt;style name="Theme.PIS_20_02.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" /&gt;
28
29 &lt;/resources&gt;</pre>
```



# RESULTADOS – PARTE 1

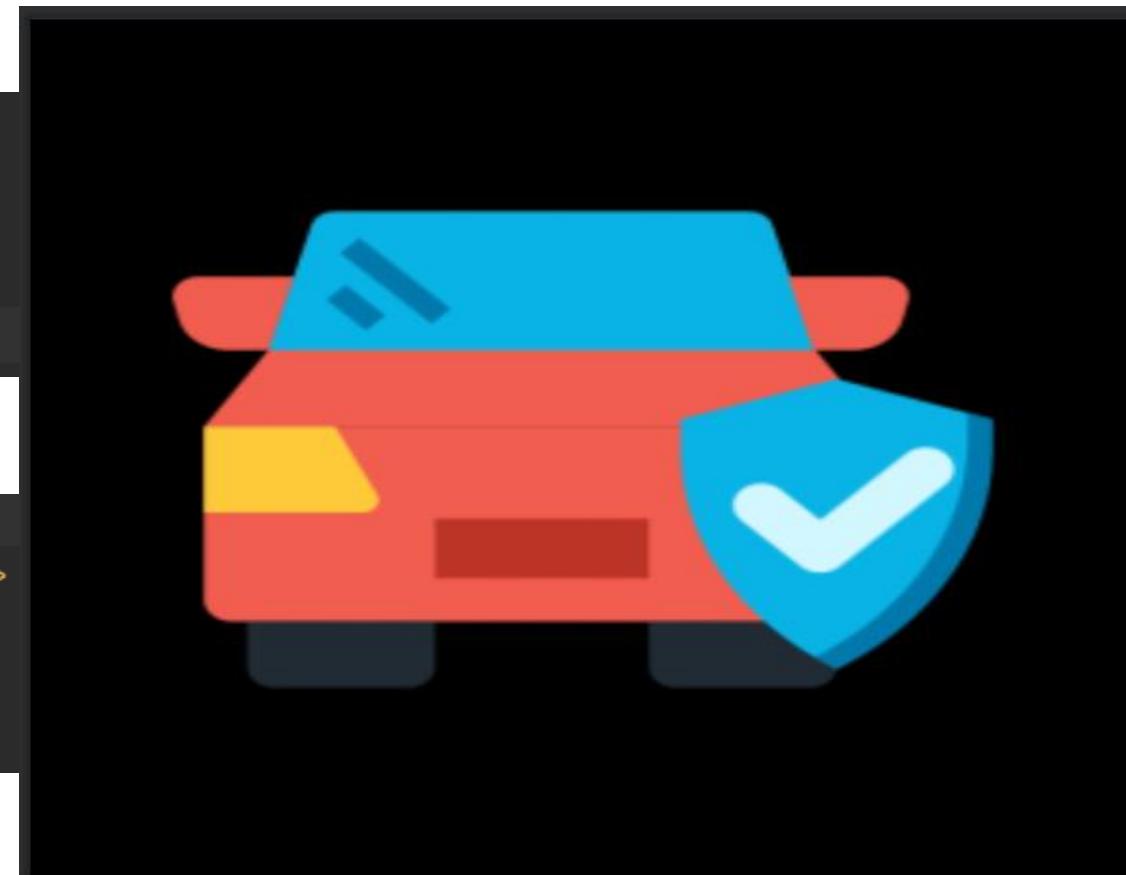
- Para el diseño general de la aplicación se procedió a diseñar los colores, tamaños de letra, fuente e iconos.

- Icono cuadrado:

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@drawable/ic_launcher_background"/>
    <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```

- Icono Redondo:

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@drawable/ic_launcher_background"/>
    <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```



# RESULTADOS – PARTE 1

- Configuración de los colores de fondo. Se muestra parte del código.

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:viewportHeight="108"
    android:viewportWidth="108"
    android:height="108dp"
    android:width="108dp">

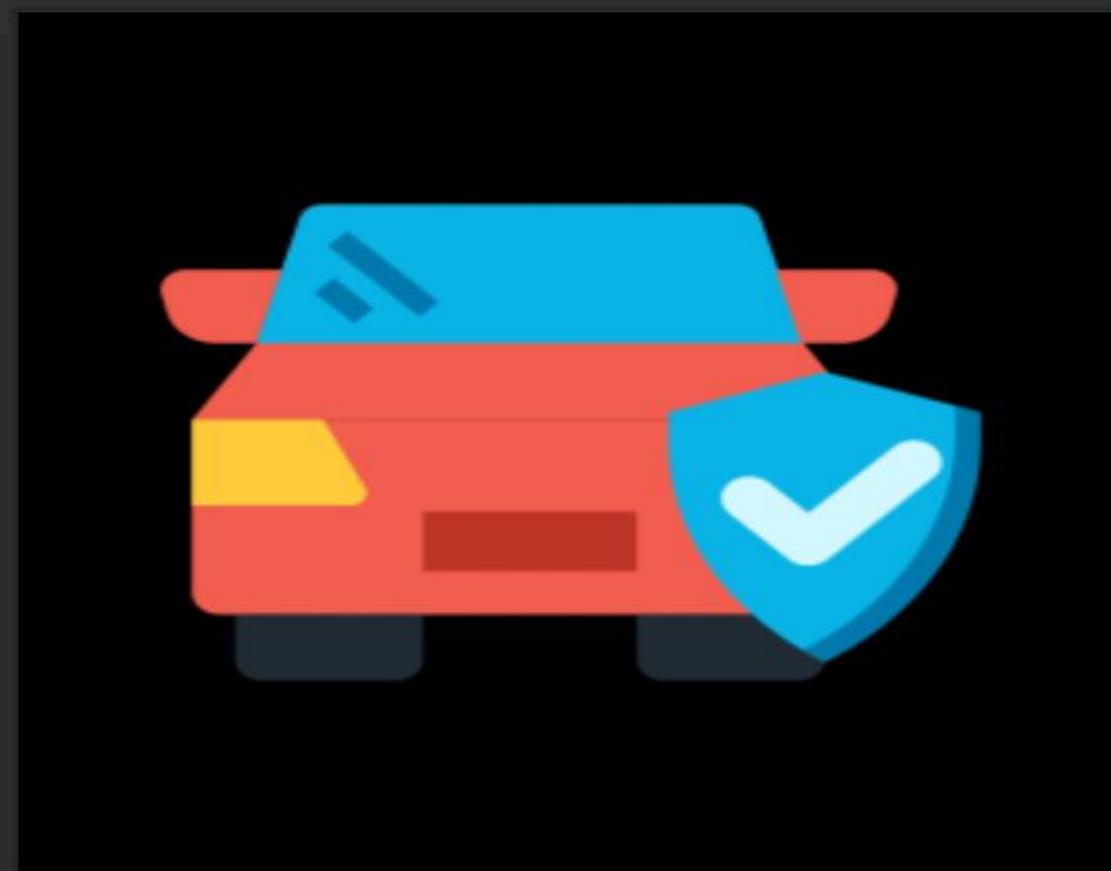
    <group android:scaleX="0"
        android:translateX="54"
        android:translateY="54"
        android:scaleY="0">
        <path android:fillColor="#3DDC84"
            android:pathData="M0,0h108v108h-108z"/>

        <path android:fillColor="#00000000" android:pathData="M9,0L9,108"
            android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

        <path android:fillColor="#00000000" android:pathData="M19,0L19,108"
            android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

        <path android:fillColor="#00000000" android:pathData="M29,0L29,108"
            android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>

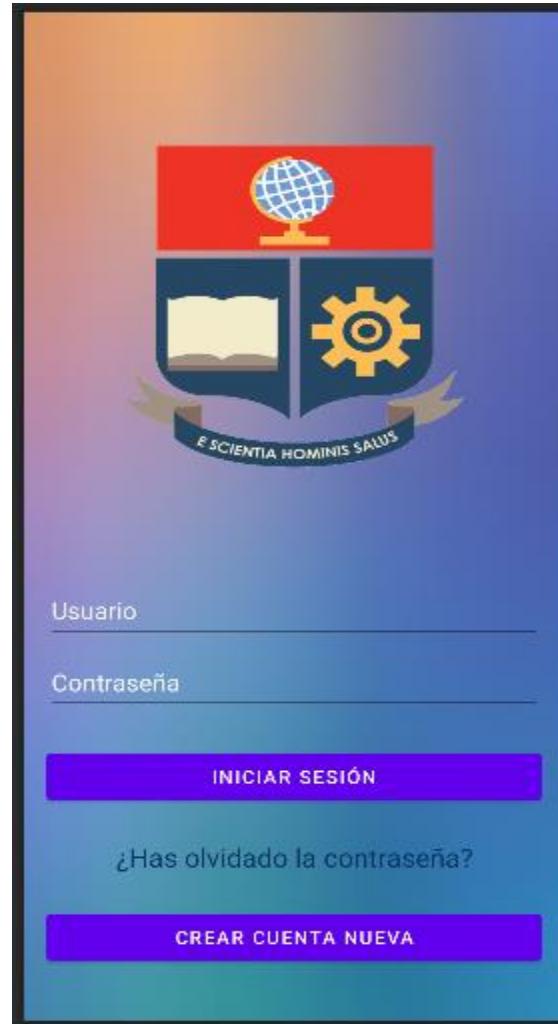
        <path android:fillColor="#00000000" android:pathData="M39,0L39,108"
            android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    
```





# RESULTADOS – PARTE 2

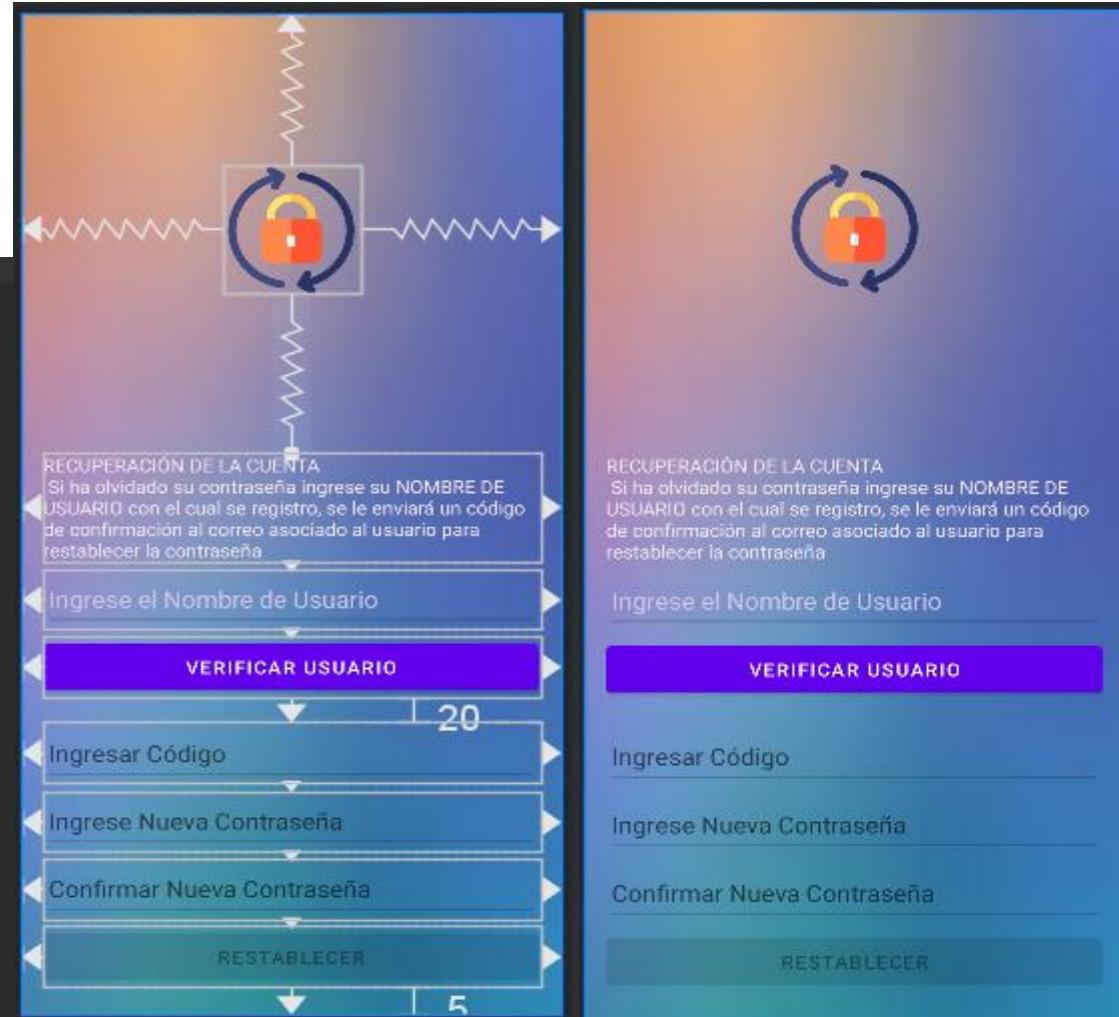
- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Diseño de la Interfaz grafica para creación de una cuenta nueva.
  - Diseño interfaz grafica para recuperación de cuenta.
  - Diseño prototipo de interfaz gráfica “Principal” tras iniciar sesión.
  - Funcionalidades de eventos a los elementos de la interfaz gráfica de Login.



# RESULTADOS – PARTE 2

- Diseño de la interfaz gráfica que permita la recuperación de las credenciales de usuario.
- A continuación se muestra segmentos de código referente a la interfaz gráfica de recuperación de cuenta.

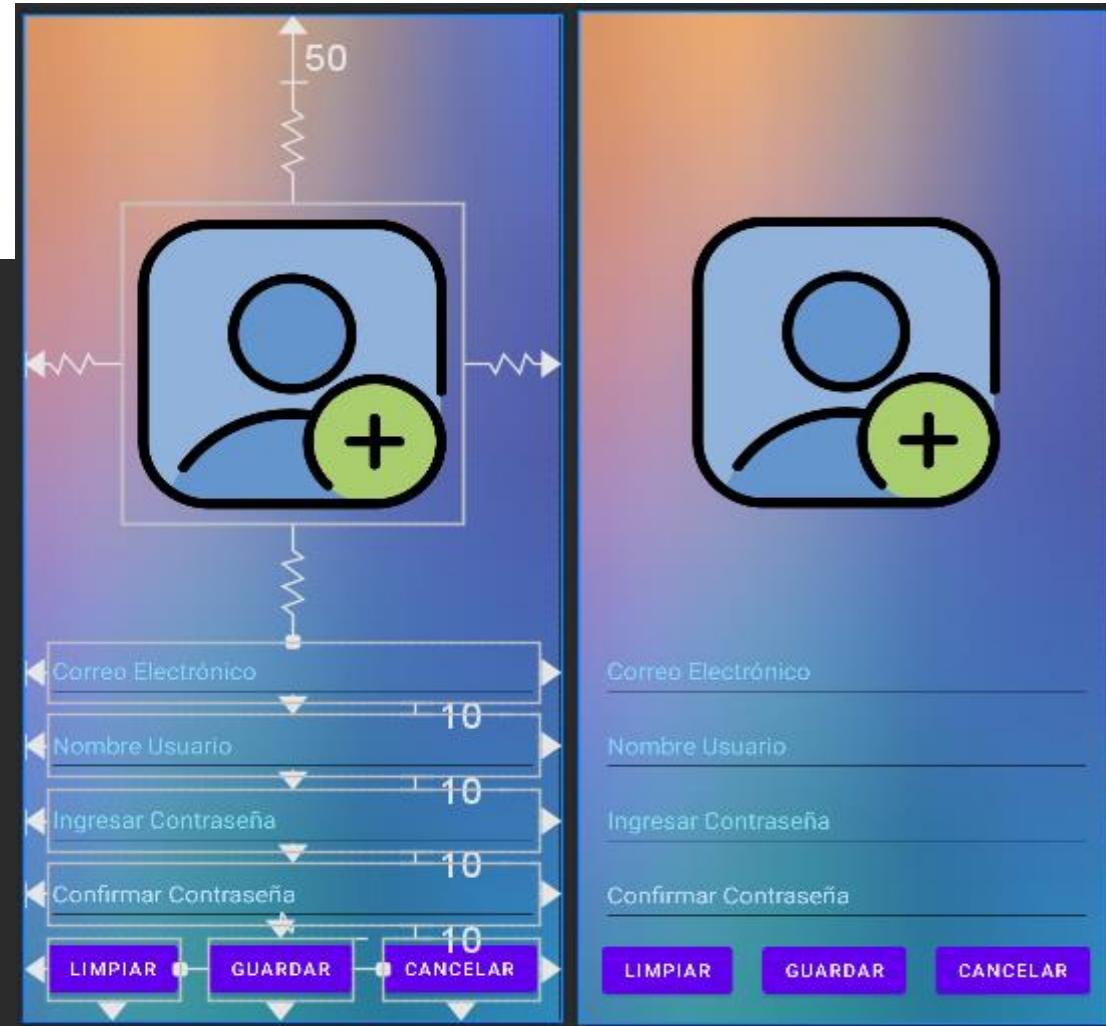
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      tools:context=".clases.recuperar.RecuperarActivity"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:background="@drawable/imagen_fondo"
7      android:layout_height="match_parent"
8      android:layout_width="match_parent"
9      android:id="@+id/RecuperarActivity"
10     android:paddingBottom="16dp"
11     android:paddingRight="16dp"
12     android:paddingLeft="16dp"
13     android:paddingTop="16dp"
14     tools:ignore="ExtraText">
15
16     <ImageView
17         android:id="@+id/imagenLogotipo"
18         android:layout_width="100dp"
19         android:layout_height="100dp"
20         android:layout_marginBottom="20dp"
21         android:contentDescription="TODO"
22         android:foreground="@drawable/restablecer"
23         app:layout_constraintBottom_toTopOf="@+id/txtInformacionRecuperar"
24         app:layout_constraintEnd_toEndOf="parent"
25         app:layout_constraintStart_toStartOf="parent"
26         app:layout_constraintTop_toTopOf="parent"
27         tools:ignore="ContentDescription,HardcodedText" />
```



# RESULTADOS – PARTE 2

- Diseño de la interfaz gráfica que permita la creación de nuevos usuarios en la APP.
- A continuación se muestra segmentos de código referente a la interfaz gráfica de creación de usuarios.

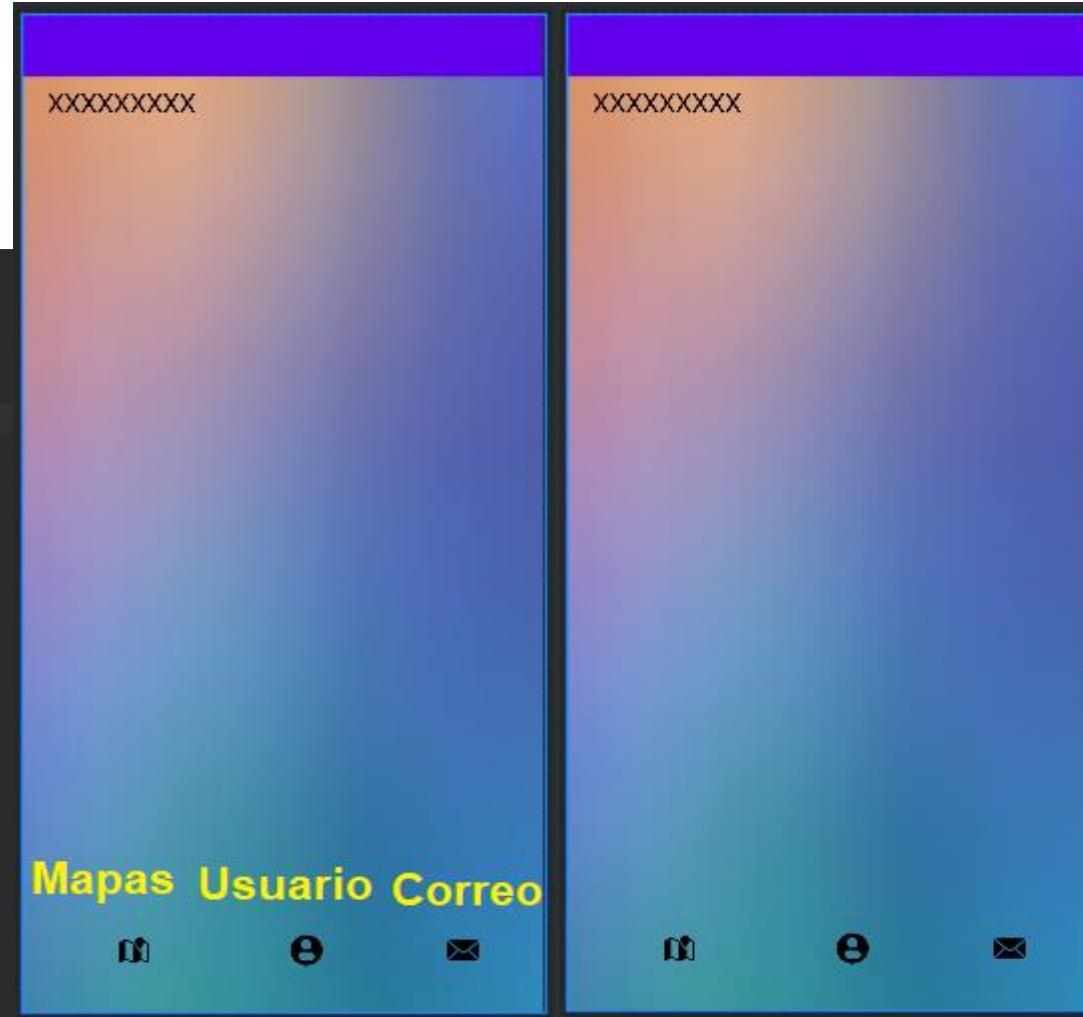
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      tools:context=".clases.nuevo.NuevoActivity"
6      android:background="@drawable/imagen_fondo"
7      android:layout_height="match_parent"
8      android:layout_width="match_parent"
9      android:id="@+id/NuevoActivity"
10     android:paddingBottom="16dp"
11     android:paddingRight="16dp"
12     android:paddingLeft="16dp"
13     android:paddingTop="16dp"
14     tools:ignore="ExtraText">
15
16     <ImageView
17         android:id="@+id/imagenLogotipo"
18         android:layout_width="250dp"
19         android:layout_height="250dp"
20         android:layout_marginTop="50dp"
21         android:layout_marginBottom="10dp"
22         android:contentDescription="TODO"
23         android:foreground="@drawable/nuevo"
24         app:layout_constraintBottom_toTopOf="@+id/txtNuevoCorreo"
25         app:layout_constraintEnd_toEndOf="parent"
26         app:layout_constraintStart_toStartOf="parent"
27         app:layout_constraintTop_toTopOf="parent"
28         tools:ignore="ContentDescription,HardcodedText" />
```



# RESULTADOS – PARTE 2

- Diseño de la interfaz gráfica Principal, la cual se muestra después de que el usuario inicia sesión con éxito.
- A continuación se muestra segmentos de código referente a la interfaz gráfica Principal.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      tools:context=".clases.principal.PrincipalActivity"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:background="@drawable/imagen_fondo"
7      android:id="@+id/contenedorPrincipal"
8      android:layout_height="match_parent"
9      android:layout_width="match_parent" >
10
11     <com.google.android.material.appbar.AppBarLayout
12         android:id="@+id/barraPestanas"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:theme="@style/Theme.PIS_20_02.AppBarOverlay">
16
17         <com.google.android.material.tabs.TabLayout
18             android:id="@+id/botonesPestanas"
19             android:layout_width="match_parent"
20             android:layout_height="wrap_content"
21             android:background="#00FFFFFF"
22             tools:ignore="SpeakableTextPresentCheck"
23             app:tabSelectedTextColor="@color/color_negro"
24             app:tabTextColor="@color/color_purple_200"/>
25
26     </com.google.android.material.appbar.AppBarLayout>
```



# RESULTADOS – PARTE 2

- Diseño de la interfaz gráfica Principal, la cual se muestra después de que el usuario inicia sesión con éxito.
- La interfaz gráfica principal a sus vez esta compuesta por tres actividades de tipo fragmento. Estos fragmentos son Fragmento de Datos, Fragmento de condiciones y Fragmento de Flujos.
- La interfaz gráfica principal es de tipo TAB, por lo cual puede contener varios fragmentos en su interior.

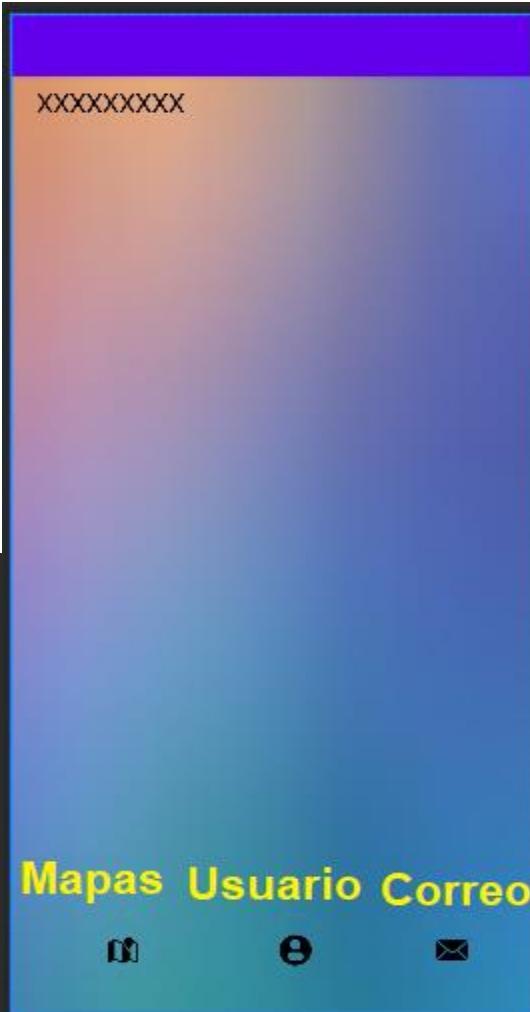
```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    tools:context=".clases.principal.PosicionFragmentoPestana"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:background="@drawable/imagen_fondo"  
    android:layout_height="match_parent"  
    android:layout_width="wrap_content"  
    android:id="@+id/contenedorDatos"  
    android:paddingBottom="16dp"  
    android:paddingRight="16dp"  
    android:paddingLeft="16dp"  
    android:paddingTop="16dp"  
    tools:ignore="ExtraText">
```

 BAJA 100%	 BAJA 100%	 BAJA 100%
Velocidad XXXXXXXX	Lugar XXXXXXXXX	Ubicación XXXXXXXXX
RPM XXXXXXXX	Temperatura XXXXXXXXX	Velocidad Promedio XXXXXXXXX
Nivel Baterías XXXXXXXX	Precipitación XXXXXXXXX	Número de Vehículos XXXXXXXXX
Temperatura Motor XXXXXXXX	Humedad XXXXXXXXX	Ocupación Promedio XXXXXXXXX
Ángulo Inclinación XXXXXXXX	Velocidad Viento XXXXXXXXX	Dirección Flujo XXXXXXXXX

# RESULTADOS – PARTE 2

- Diseño de la interfaz gráfica Principal, la cual se muestra después de que el usuario inicia sesión con éxito, posee tres botones flotantes.
- Los botones flotantes corresponde a las funcionalidades de Mapas, Usuario Correo.
- Mapas: permite ingresar a una nueva actividad en la cual se presenta un mapa con la posición actual del usuario.
- Usuario: permite ingresar a las configuraciones del usuario e información adicional. Adicionalmente, posee un apartado que permite cerrar la sesión del usuario.
- Correo: permite enviar un correo o notificación a los desarrolladores de la aplicación con comentarios sobre la aplicación o reportando algún tipo de error.

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/botonInformacionUsuario"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_marginEnd="150dp"
    android:layout_marginBottom="20dp"
    android:src="@drawable/usuario"
    app:backgroundTint="@android:color/transparent"
    app:elevation="0dp"
    app:rippleColor="@android:color/transparent"
    tools:ignore="ContentDescription,SpeakableTextPresentCheck,ImageContrastCheck" />
```



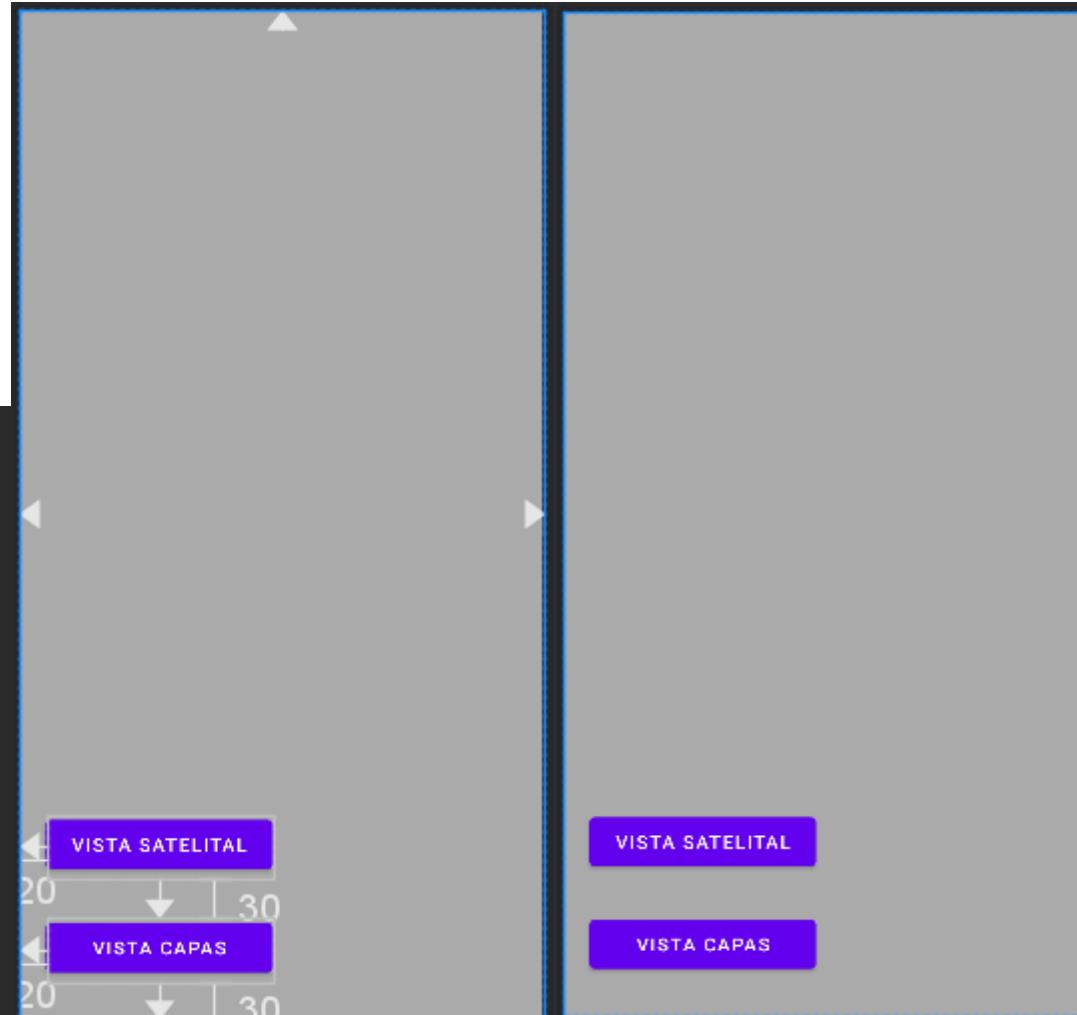
# RESULTADOS – PARTE 3

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Diseño de la Interfaz gráfica de Mapas.
  - Diseño de la interfaz gráfica de Usuario.
  - Diseño de la interfaz gráfica de Correo.

# RESULTADOS – PARTE 3

- Diseño de la interfaz gráfica correspondiente a los mapas de la aplicación.
- Esta actividad de la aplicación permite localizar y mostrar en tiempo real la ubicación del usuario en tiempo real. Adicionalmente permite cambiar el modelo de vista de los mapas, siendo la primera y por default el modelo cuadriculado y el segundo siendo un modelo con vista satelital.

```
<com.google.android.gms.maps.MapView
    android:id="@+id/presentarMapa"
    android:name="com.google.android.gms.maps"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    map:layout_constraintEnd_toEndOf="parent"
    map:layout_constraintStart_toStartOf="parent"
    map:layout_constraintTop_toTopOf="parent"
    tools:ignore="SpeakableTextPresentCheck" />
```



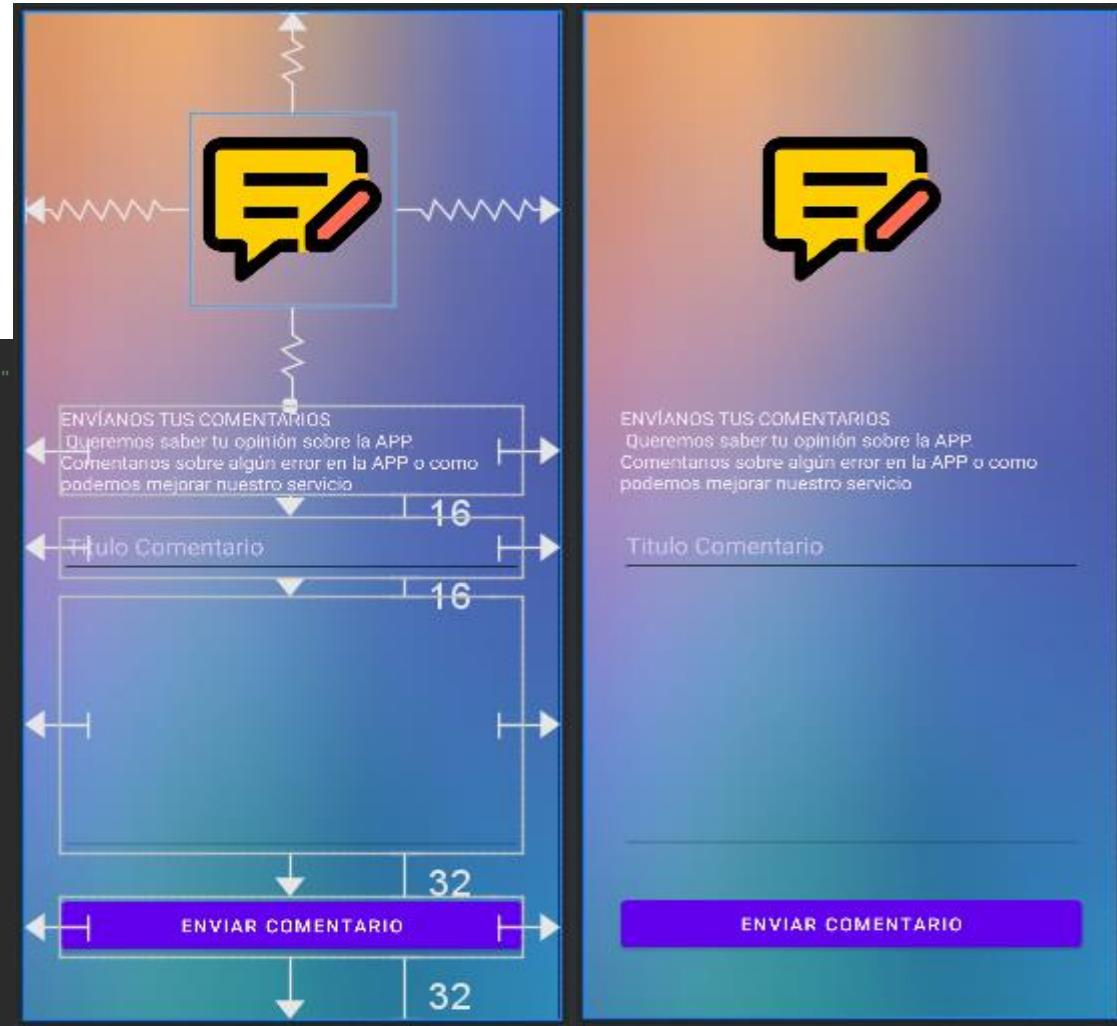
# RESULTADOS – PARTE 3

- Diseño de la interfaz gráfica correspondiente a los correos. Esta sesión permite al usuario comunicarse con los desarrolladores de la aplicación para poder enviar comentarios o reportar errores en la aplicación.
- El botón incluido en la interfaz gráfica permite capturar los datos ingresados en los campos de la interfaz y pasarlo al gestor de correos propio del celular y así poder enviar un correo.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      tools:context=".clases.comentarios.ComentariosActivity"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:background="@drawable/imagen_fondo"
7      android:layout_height="match_parent"
8      android:layout_width="match_parent"
9      android:id="@+id/RecuperarActivity"
10     android:paddingBottom="16dp"
11     android:paddingRight="16dp"
12     android:paddingLeft="16dp"
13     android:paddingTop="16dp"
14     tools:ignore="ExtraText">
15
16
17     <ImageView
18         android:id="@+id/imagenComentarios"
19         android:layout_width="150dp"
20         android:layout_height="150dp"
21         android:layout_marginBottom="16dp"
22         android:contentDescription="TODO"
23         android:foreground="@drawable/comentarios"
24         app:layout_constraintBottom_toTopOf="@+id/txtEnviarComentarios"
25         app:layout_constraintEnd_toEndOf="parent"
26         app:layout_constraintStart_toStartOf="parent"
27         app:layout_constraintTop_toTopOf="parent"
28         tools:ignore="ContentDescription,HardcodedText" />

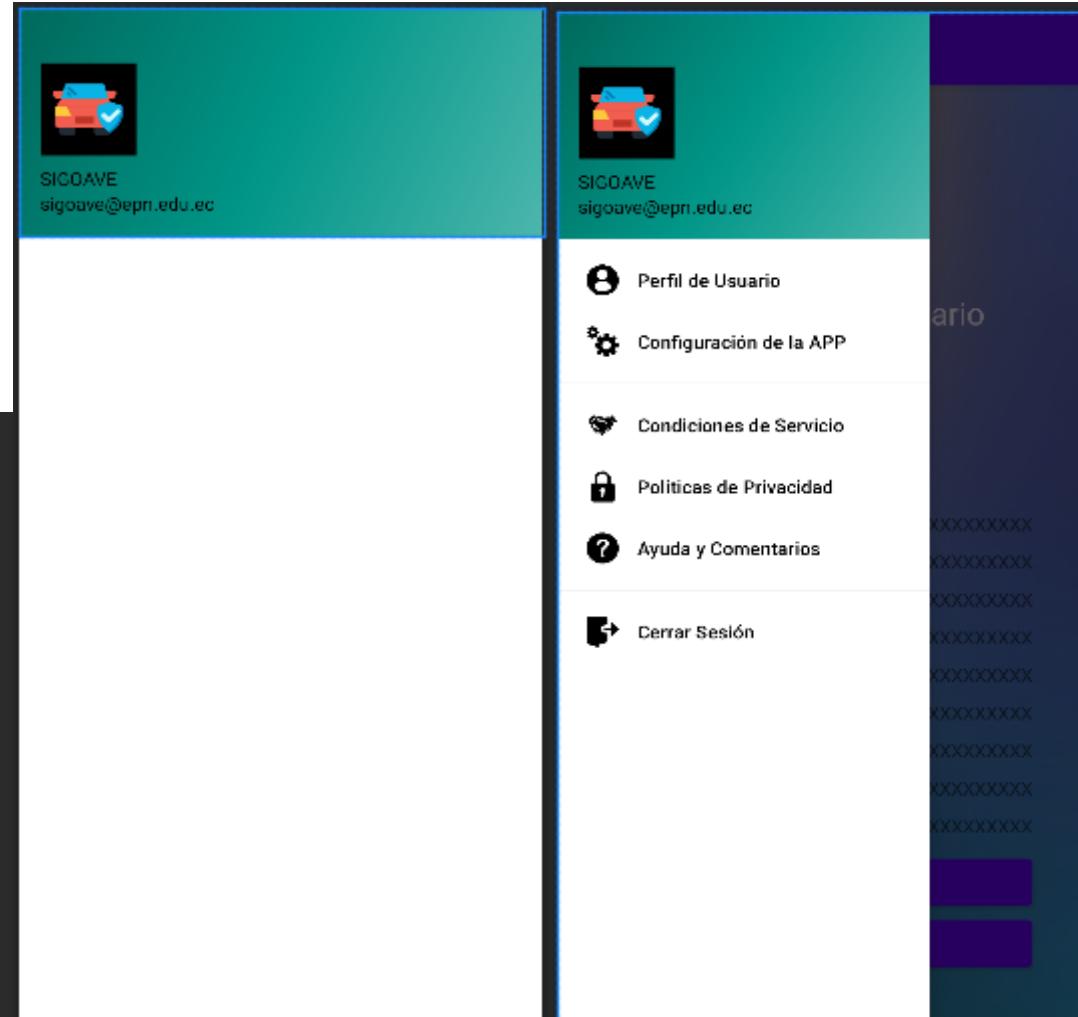
```



# RESULTADOS – PARTE 3

- Diseño de la interfaz gráfica correspondiente a la configuración y administración de usuarios. Este apartado muestra información correspondiente al usuario, permite cerrar la sesión de usuario y finalmente, permite realizar ciertas configuraciones de funcionamiento de la aplicación.
- Esta actividad es de tipo menú lateral y esta compuesto por varios fragmentos de actividad.

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <menu xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:tools="http://schemas.android.com/tools"
4       tools:showIn="navigation_view">
5
6       <group
7           android:id="@+id/navegador_grupo_1"
8           android:checkableBehavior="single">
9               <item
10                  android:id="@+id/perfil_de_usuario"
11                  android:icon="@drawable/usuario"
12                  android:title="Perfil de Usuario" />
13               <item
14                  android:id="@+id/configuracion_usuario"
15                  android:icon="@drawable/configuracion"
16                  android:title="Configuración de la APP" />
17           </group>
```



# RESULTADOS – PARTE 3

- Esta actividad es de tipo menú lateral y esta compuesto por varios fragmentos de actividad. Estos fragmentos son: Perfil de Usuario, Configuraciones de la APP, Condiciones de Servicio, Políticas de Privacidad, Ayuda y Comentarios, finalmente, Cerrar Sesión.

SICOAVE  
sicoave@epn.edu.ec

- Perfil de Usuario
- Configuración de la APP
- Condiciones de Servicio
- Políticas de Privacidad
- Ayuda y Comentarios
- Cerrar Sesión

**Información de Usuario**

Nombre y Apellido

Nombre de Usuario	XXXXXXXXXX
Correo Electrónico	XXXXXXXXXX
Teléfono Usuario	XXXXXXXXXX
Edad	XXXXXXXXXX
Género	XXXXXXXXXX
Uso de Lentes	XXXXXXXXXX
Estado Licencia	XXXXXXXXXX
Puntos Licencia	XXXXXXXXXX
Condiciones Médicas	XXXXXXXXXX

**MODIFICAR USUARIO**

**ELIMINAR USUARIO**

**Configuraciones de Usuario**

La configuración del usuario le permite configurar las preferencias de idioma y de correo electrónico, mientras que los datos de la cuenta le permiten controlar su configuración del uso compartido de datos.

**IR A CONFIGURACIONES**

**Condiciones de Servicio**

Queda prohibido alterar o modificar ninguna parte de la APP a los contenidos de la misma, eludir, desactivar o manipular de cualquier forma (o tratar de eludir, desactivar o manipular) las funciones de seguridad u otras funciones del programa y utilizar la APP o sus contenidos para un fin comercial o publicitario

**Políticas de Privacidad**

Se recopilarán datos de un modo que, por sí mismos, no pueden ser asociados directamente a una persona determinada. Estos datos de carácter no personal se pueden recopilar, tratar, transferir y publicar con cualquier intención

**ENVIANDO TUS COMENTARIOS**  
Queremos saber tu opinión sobre la APP. Comentanos sobre algún error en la APP o como podemos mejorar nuestro servicio

**Título Comentario**

**ENVIAR COMENTARIO**

**OBTENER AYUDA ONLINE**

**¿Cerrar Sesión?**

**CERRAR SESIÓN**

# RESULTADOS – PARTE 4

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Implementación de la lógica de negocios de a la aplicación (Amplify y Cognito).
  - Implementación de la base de datos en AWS.
  - Implementación de los servicios de Google Maps.
  - Implementación de las alertas visuales y sonoras de la aplicación.
  - Implementación de la lectura del Dataset de trabajo.
  - Implementación de la comunicación con el servicio REST mediante POST.
  - Corrección de errores.

# RESULTADOS – PARTE 4 – LOGIN

```
Amplify.Auth.fetchAuthSession(result -> LoginActivity.this.runOnUiThread(new Runnable() {
    //Segmento de código que se ejecuta en un hilo secundario para verificar la conexión con los servicios de AWS
    public void run() {
        //Mensaje de consola que informa al usuario que se estableció conexión con los servicios de AWS
        Log.i( tag: "Información", result.toString());
        try {
            //Segmento de código que verifica si el usuario ya había iniciado sesión previamente
            Amplify.Auth.getCurrentUser(result -> LoginActivity.this.runOnUiThread(new Runnable() {
                public void run() {
                    //Mensaje de consola que informa al usuario que se auténtico al usuario con los servicios de AWS
                    Log.i( tag: "Información", result.toString());
                    //Segmento de código para inhabilitar los botones de la actividad
                    cambiarRecuperar.setEnabled(false);
                    cambiarInicio.setEnabled(false);
                    cambiarNuevo.setEnabled(false);
                    password.setEnabled(false);
                    usuario.setEnabled(false);
                    //Mensaje emergente de saludo al usuario con el nombre de usuario
                    String informacion = "Bienvenido" + " " + result.getUsername();
                    Toast.makeText(getApplicationContext(), informacion, Toast.LENGTH_SHORT).show();
                    //Redirigiendo al usuario a la actividad Principal de la aplicación enviando el parámetro de nombre
                    Intent intent = new Intent( packageContext: LoginActivity.this, PrincipalActivity.class);
                    intent.putExtra( name: "username", result.getUsername());
                    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                    LoginActivity.this.startActivity(intent);
                }
            })
        }
    }
})
```

# RESULTADOS – PARTE 4 – MANIFEST

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />

<!-- Información base de la aplicación como tema e icono de inicio--&gt;
&lt;application
    android:name=".base_datos.PrincipalAmplify"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="SIGOAVE"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.PIS_20_02"
    tools:targetApi="31"&gt;

    <!-- Información necesaria para el funcionamiento de Google Maps --&gt;
    &lt;meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyDKEfKSNe3HZBKo3p_r0flcWTyE0CpxsMU" /&gt;

    <!-- Información base de la clase Comentarios y conexión con su clase padre --&gt;
    &lt;activity
        android:name=".clases.comentarios.ComentariosActivity"
        android:exported="false"
        android:label="Enviar Comentario"
        android:parentActivityName=".clases.principal.PrincipalActivity"&gt;</pre>
```



# RESULTADOS – PARTE 4 – BASE DE DATOS

```
public class PrincipalAmplify extends Application {
    //En el método onCreate() ejecuta la lógica de
    //arranque básica de la aplicación
    public void onCreate() {
        super.onCreate();
        //Inicia la conexión con Cognito de AWS, presentando un mensaje en consola si se establece la conexión o no.
        try {
            Amplify.addPlugin(new AWSAuthPlugin());
            Log.i( tag: "MyCognitoApp", msg: "Initialized Cognito");
        } catch (AmplifyException e) {
            Log.e( tag: "MyCognitoApp", msg: "Could not initialize Cognito", e);
        }
        //Inicia la conexión con Amplify de AWS, presentando un mensaje en consola si se establece la conexión o no.
        try {
            Amplify.configure(getApplicationContext());
            Log.i( tag: "MyAmplifyApp", msg: "Initialized Amplify");
        } catch (AmplifyException error) {
            Log.e( tag: "MyAmplifyApp", msg: "Could not initialize Amplify", error);
        }
    }
}
```



# RESULTADOS – PARTE 4 – VERIFICACIÓN

```
Amplify.Auth.confirmSignUp(nombre, codigo, result -> CódigoActivity.this.runOnUiThread(new Runnable() {
    //Segmento de código que se ejecuta cuando la información enviada es correcta
    public void run() {
        //Presentación de información al usuario por consola y mediante un mensaje emergente
        Log.i( tag: "Información", msg: "Usuario creado con éxito");
        Toast.makeText(getApplicationContext(), text: "Usuario creado con éxito", Toast.LENGTH_SHORT).show();
        //Redirigiendo al usuario a la actividad de Login
        Intent intent = new Intent( packageContext: CódigoActivity.this, LoginActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        CódigoActivity.this.startActivity(intent);
    }
}), error -> CódigoActivity.this.runOnUiThread(new Runnable() {
    //Segmento de código que se ejecuta cuando la información enviada es incorrecta
    public void run() {
        //Presentación de información al usuario por consola y mediante un mensaje emergente
        Log.e( tag: "Error", error.toString());
        Toast.makeText(getApplicationContext(), text: "El Código Ingresado es Incorrecto", Toast.LENGTH_SHORT).show();
    }
}));
```

# RESULTADOS – PARTE 4 – COMENTARIOS

```
cambiarPrincipal.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        try {
            //Verificando que no existan campos que se encuentre vacios
            if(!textoTitulo.getText().toString().isEmpty() && !textoComentario.getText().toString().isEmpty())
            {
                //Cadena de texto para verificar el estado del mensaje
                enviado = "No Enviado";
                //Obteniendo el cuerpo del mensaje a ser enviado
                String comentario = textoComentario.getText().toString();
                String titulo = textoTitulo.getText().toString();
                String correo = "sigoave@epn.edu.ec";
                //Iniciando la aplicación de correos definida en el celular
                Intent intent = new Intent(Intent.ACTION_SEND);
                //Enviando la información a la aplicación de correos del celular
                intent.putExtra(Intent.EXTRA_EMAIL, new String[]{correo});
                intent.putExtra(Intent.EXTRA_SUBJECT, titulo);
                intent.putExtra(Intent.EXTRA_TEXT, comentario);
                //Indica que el cuerpo contiene un mensaje encapsulado con la sintaxis de un mensaje RFC 822
                intent.setType("message/rfc822");
                startActivity(intent);
                //Cadena de texto para verificar el estado del mensaje
                enviado = "Enviado";
            }
        }
    }
})
```

# RESULTADOS – PARTE 4 – MAPAS

```
public void onMapReady(@NonNull GoogleMap googleMap) {
    try {
        //Creación de variables para crear y cargar los mapas en pantalla con la ubicación del usuario
        misMapas = googleMap;
        misMapas.getUiSettings().setMyLocationButtonEnabled(true);
        misMapas.getUiSettings().setZoomControlsEnabled(true);

        //Verificación de permisos de acceso a la ubicación
        if (ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            return;
        }
        //Segmento de código que permite mostrar la ubicación actual del usuario en la pantalla
        misMapas.setMyLocationEnabled(true);
        //Segmento de código que permite llamar a los servicios GPS
        LocationManager locationManager = (LocationManager) MapasActivity.this.getSystemService(Context.LOCATION_SERVICE)
        LocationListener locationListener = new LocationListener() {
            public void onLocationChanged(Location location) {
                //Obteniendo las coordenadas del usuario (latitud y longitud)
                LatLng miUbicacion = new LatLng(location.getLatitude(), location.getLongitude());
                //Seteando la ubicación del usuario en pantalla
                Objects.requireNonNull(misMapas.addMarker(new MarkerOptions().position(miUbicacion).title("Mi Ubicación")));
                misMapas.moveCamera(CameraUpdateFactory.newLatLng(miUbicacion));
                //Parametros que permiten mejorar la interacción con el usuario
                //Permite hacer zoom en la pantalla y cambiar la posición de la cámara
                CameraPosition posicionCamara = new CameraPosition.Builder().target(miUbicacion).zoom(18).bearing(0).tilt(45);
                misMapas.animateCamera(CameraUpdateFactory.newCameraPosition(posicionCamara));
            }
        };
    }
}
```



# RESULTADOS – PARTE 4 – MENÚ LATERAL

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    vinculador = ActivityMenuBinding.inflate(getLayoutInflater());
    setContentView(vinculador.getRoot());

    setSupportActionBar(vinculador.appBarMenuLateral.toolbar);

    DrawerLayout fragmento = vinculador.drawerLayout;
    NavigationView vista = vinculador.navView;
    configuracionLateral = new AppBarConfiguration.Builder(R.id.perfil_de_usuario, R.id.configuracion_usuario, R.id.condiciones_servicio)
        .setDrawerLayout(fragmento)
        .setToolbar(vista.toolbar)
        .setNavHostNavController(navController)
        .build();
    NavController navController = Navigation.findNavController(activity: this, R.id.contenedor_menu_lateral);
    NavigationUI.setupActionBarWithNavController(activity: this, navController, configuracionLateral);
    NavigationUI.setupWithNavController(vista, navController);
}

public boolean onSupportNavigateUp() {
    NavController navController = Navigation.findNavController(activity: this, R.id.contenedor_menu_lateral);
    return NavigationUI.navigateUp(navController, configuracionLateral) || super.onSupportNavigateUp();
}
```



# RESULTADOS – PARTE 4 – NOTIFICACIONES DE VOZ

```
private final TextToSpeech.OnInitListener conversor = new TextToSpeech.OnInitListener() {  
    public void onInit(int estado) {  
        //Selección del idioma y región  
        Locale espanol = new Locale(language: "es", country: "EC");  
        //Verificación si el TTS esta iniciado  
        if (estado == TextToSpeech.SUCCESS) {  
            //Seteando el lenguaje deseado al convertidor  
            int resultado = textoVoz.setLanguage(espanol);  
            cargado = true;  
            if (resultado == TextToSpeech.LANG_MISSING_DATA || resultado == TextToSpeech.LANG_NOT_SUPPORTED) {  
                //Mensaje de consola que informa al usuario que el idioma o región no son soportados  
                Log.e(tag: "Error.", msg: "Lenguaje no permitido.");  
            }  
        } else {  
            //Mensaje de error que indica que el TTS no se pudo iniciar  
            Log.e(tag: "Error.", msg: "Falló al iniciar.");  
        }  
    }  
};
```



ESCUELA  
POLÍTÉCNICA  
NACIONAL

# RESULTADOS – PARTE 4 – NUEVO USUARIO

```
protected void onCreate(Bundle savedInstanceState) {
    //Creación y presentación visual de la actividad
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_nuevo);

    //Inicializando las variables con los elementos de la actividad
    confirmar = findViewById(R.id.txtNuevoConfirmarContrasena);
    password = findViewById(R.id.txtNuevoContrasena);
    cancelar = findViewById(R.id.botonCancelarNuevo);
    limpiar = findViewById(R.id.botonLimpiarNuevo);
    guardar = findViewById(R.id.botonGuardarNuevo);
    usuario = findViewById(R.id.txtNuevoNombre);
    correo = findViewById(R.id.txtNuevoCorreo);

    //Segmento de código que permite la verificación de la longitud de la contraseña cuando este campo pierde el foco
    //Usado en el campo de confirmación de contraseña
    confirmar.setOnFocusChangeListener(new View.OnFocusChangeListener() {
        public void onFocusChange(View v, boolean hasFocus) {
            if (hasFocus) {
                //Mensaje lateral secundario que informa al usuario que la contraseña debe tener una longitud mínima de
                if (confirmar.getText().toString().trim().length() < 8) {
                    confirmar.setError("El Password debe tener una longitud mínima de 8");
                } else {
                    //Mensaje lateral secundario al usuario que informa que la contraseña no posee la longitud mínima
                    confirmar.setError(null);
                }
            }
        }
    });
}
```

# RESULTADOS – PARTE 4 – DATASET

```
public void setCurrent_weather_category(String current_weather_category) {  
    this.current_weather_category = current_weather_category;  
    //TODO  
}  
  
public void setDistance_travelled_total(double distance_travelled_total) {  
    this.distance_travelled_total = distance_travelled_total;  
    //TODO  
}  
  
public void setSpeed_vs_accidents_onsite(int speed_vs_accidents_onsite) {  
    this.speed_vs_accidents_onsite = speed_vs_accidents_onsite;  
    //TODO  
}  
  
public void setSpeed_vs_steering_angle_(int speed_vs_steering_angle_) {  
    this.speed_vs_steering_angle_ = speed_vs_steering_angle_;  
    //TODO  
}  
  
public void setReal_feel_temperature(double real_feel_temperature) {  
    this.real_feel_temperature = real_feel_temperature;  
    //TODO  
}
```



# RESULTADOS – PARTE 4 – AGENTE REST POST

```
url = new URL(wsURL);
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
JSONObject parametrosPost = new JSONObject();
parametrosPost.put("name", "Input", "value": "[" + steering_angle_ + ", " + speed + ", " + rpm_ + ", " + acceleration + "]");

urlConnection.setReadTimeout(5000);
urlConnection.setConnectTimeout(5000);
urlConnection.setRequestMethod("POST");
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);

OutputStream os = new BufferedOutputStream(urlConnection.getOutputStream());
BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, StandardCharsets.UTF_8));
writer.write(getpostDataString(parametrosPost));
writer.flush();
writer.close();
os.close();
```



# RESULTADOS – PARTE 4 – JSON

```
"UserAgent": "aws-amplify-cli/2.0",
"Version": "1.0",
"auth": {
  "plugins": {
    "awsCognitoAuthPlugin": {
      "UserAgent": "aws-amplify-cli/0.1.0",
      "Version": "0.1.0",
      "IdentityManager": {
        "Default": {}
      },
      "CredentialsProvider": {
        "CognitoIdentity": {
          "Default": {
            "PoolId": "us-west-1:f8eb96d9-3c57-49ed-b27c-5158eaa72a2f",
            "Region": "us-west-1"
          }
        }
      },
      "CognitoUserPool": {
        "Default": {
          "PoolId": "us-west-1_XVBERR46S",
          "AppClientId": "4irc1gn06ntp06b6vi6ngofjo2",
          "Region": "us-west-1"
        }
      }
    }
  }
},
```

# RESULTADOS – PARTE 5

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Finalización de la Implementación de la comunicación con el servicio REST mediante POST.
  - Presentación de los dato obtenidos del servicio REST.
  - Presentación de Alertas por Voz.
  - Presentación de Alertas Visuales.
  - Presentación Información de Usuario.
  - Corrección de errores.



# RESULTADOS – PARTE 5 – AGENTE REST POST

```
url = new URL(wsURL);
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
JSONObject parametrosPost = new JSONObject();
parametrosPost.put("name", "Input", "value": "[" + steering_angle_ + ", " + speed + ", " + rpm_ + ", " + acceleration + "]");

urlConnection.setReadTimeout(5000);
urlConnection.setConnectTimeout(5000);
urlConnection.setRequestMethod("POST");
urlConnection.setDoInput(true);
urlConnection.setDoOutput(true);

OutputStream os = new BufferedOutputStream(urlConnection.getOutputStream());
BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(os, StandardCharsets.UTF_8));
writer.write(getpostDataString(parametrosPost));
writer.flush();
writer.close();
os.close();
```

# RESULTADOS – PARTE 5 – AGENTE REST POST

- Creado el método POST se debe presentar la información al usuario. Para ello se debe crear variables que permita obtener y enviar la información a los usuarios.
- Creadas las variables se crea el constructor y el método para presentar la información.

```
//Creación del constructor de la clase para presentar la información
public String toString() {
    //return "steering_angle = " + steering_angle_ + " speed = " + speed + " rpm = " + rpm_ + " acceleration = " + acceleration + " throttle_position = " + throttle
    return "[" + steering_angle_ + ", " + speed + ", " + rpm_ + ", " + acceleration + ", " + throttle_position + ", " + engine_temperature + ", " + system_voltage +
    //return "'Input': [[ " + steering_angle_ + ", " + speed + ", " + rpm_ + ", " + acceleration + ", " + throttle_position + ", " + engine_temperature + ", " + system
}
```

- Desde el Agente se debe enviar la información a las actividades pertinente para presentar la información.

```
protected void onPostExecute(String s) {
    //Segmento de código que permite obtener la respuesta del servicio REST para que esta sea presentada en las actividades
    super.onPostExecute(s);
    FragmentoModeloVista cambio = new FragmentoModeloVista();
    cambio.datosRespuesta(s);
    //Mensaje de consola que permite mostrar el resultado del servicio REST
    Log.i( tag: "RESULTADO API: ", s);
    resultadoAPI = s;
}
```

```
//Variables con los campos del Dataset
private String has_precipitation_category;
private String current_weather_category;
private double distance_travelled_total;
private int speed_vs_accidents_onsite;
private int speed_vs_steering_angle_;
private double real_feel_temperature;
private String is_day_time_category;
private int speed_vs_precipitation;
private double barometric_pressure;
private double distance_travelled;
private double throttle_position;
private double steering_angle_;
private int engine_temperature;
private int relative_humidity;
private int has_precipitation;
private double system_voltage;
private double steering_angle;
private int accidents_onsite;
private double precipitation;
private String uv_index_text;
private int current_weather;
private double acceleration;
private int wind_direction;
```



# RESULTADOS – PARTE 5 – AGENTE REST POST

- Dentro del formulario principal se presentan toda la información obtenida desde el agente y el servicio REST de AWS.

```
if (s.equals("{\"Output\": 4}") && get_set_MuyAlta) {  
    //Presentando las alertas gráficas de nivel 4 al usuario  
    resultadoTexto.setText(getString(R.string.medida_datos_muy_alta));  
    resultadoTexto.setTextSize(24);  
    resultadoBandera.setImageResource(R.drawable.rojo);  
    //Iniciando alerta sonora de nivel 4  
    String mensaje = "Nivel de Riesgo Muy Alta";  
    conversor.palabra(mensaje);  
    mostrarRojo(mensaje);  
}  
  
if (s.equals("{\"Output\": 2}") && get_set_Alta) {  
    //Presentando las alertas gráficas de nivel 2 al usuario  
    resultadoTexto.setText(getString(R.string.medida_datos_media));  
    resultadoBandera.setImageResource(R.drawable.amarillo);  
    //Iniciando alerta sonora de nivel 2  
    String mensaje = "Nivel de Riesgo Media";  
    conversor.palabra(mensaje);  
    mostrarAmarillo(mensaje);  
}  
  
if (s.equals("{\"Output\": 3}") && get_set_Alta) {  
    //Presentando las alertas gráficas de nivel 3 al usuario  
    resultadoTexto.setText(getString(R.string.medida_datos_alta));  
    resultadoBandera.setImageResource(R.drawable.naranja);  
    //Iniciando alerta sonora de nivel 3  
    String mensaje = "Nivel de Riesgo Alta";  
    conversor.palabra(mensaje);  
    mostrarNaranja(mensaje);  
}  
  
if (s.equals("{\"Output\": 1}") && get_set_Baja) {  
    //Presentando las alertas gráficas de nivel 1 al usuario  
    resultadoTexto.setText(getString(R.string.medida_datos_baja));  
    resultadoBandera.setImageResource(R.drawable.verde);  
    //Iniciando alerta sonora de nivel 1  
    String mensaje = "Nivel de Riesgo Baja";  
    conversor.palabra(mensaje);  
    mostrarVerde(mensaje);  
}
```



# RESULTADOS – PARTE 5 – NOTIFICACIONES DE VOZ

- Primer paso es crear la clase que permita convertir a voz un texto dado. Esta clase hace uso de las librerías TextToSpeech, las cuales están configuradas en varios idiomas y regiones. Entre los idiomas y regiones se permite la configuración del idioma español y la región de Ecuador.

```
private final TextToSpeech.OnInitListener conversor = new TextToSpeech.OnInitListener() {  
    public void onInit(int estado) {  
        //Selección del idioma y región  
        Locale espanol = new Locale(language: "es", country: "EC");  
        //Verificación si el TTS esta iniciado  
        if (estado == TextToSpeech.SUCCESS) {  
            //Seteando el lenguaje deseado al convertidor  
            int resultado = textoVoz.setLanguage(espanol);  
            cargado = true;  
            if (resultado == TextToSpeech.LANG_MISSING_DATA || resultado == TextToSpeech.LANG_NOT_SUPPORTED) {  
                //Mensaje de consola que informa al usuario que el idioma o región no son soportados  
                Log.e(tag: "Error.", msg: "Lenguaje no permitido.");  
            }  
        } else {  
            //Mensaje de error que indica que el TTS no se pudo iniciar  
            Log.e(tag: "Error.", msg: "Falló al iniciar.");  
        }  
    }  
};
```

# RESULTADOS – PARTE 5 – NOTIFICACIONES DE VOZ

- Creada la clase que permite la conversión de texto a voz se procede a llamar a la misma desde la actividad principal.

```
//Creación de un instancia a la clase de notificaciones de voz TTS  
private NotificacionesVoz conversor = null;
```

```
//Iniciando alerta sonora de nivel 4  
String mensaje = "Nivel de Riesgo Muy Alta";  
conversor.palabra(mensaje);
```

- Para evitar problemas la conversión de texto a voz debe ser finalizada al terminar la actividad.

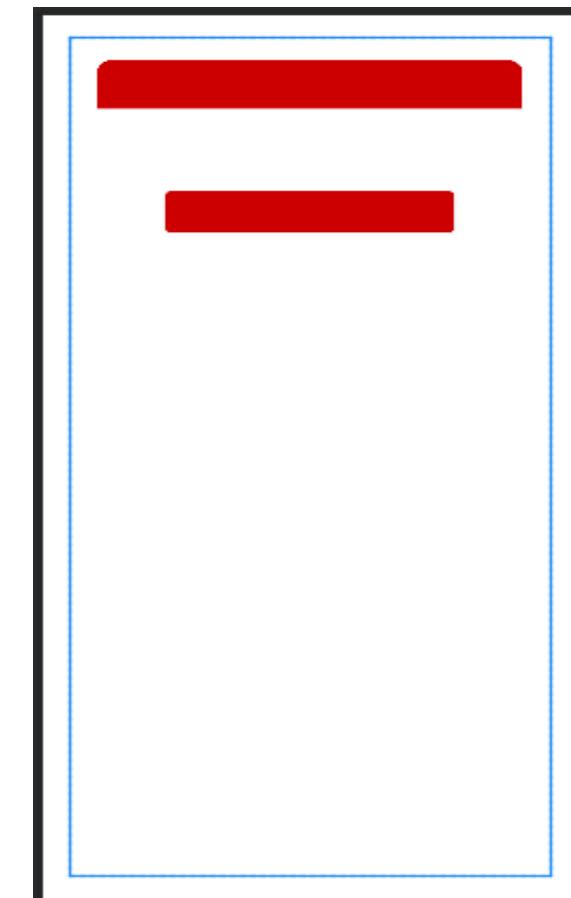
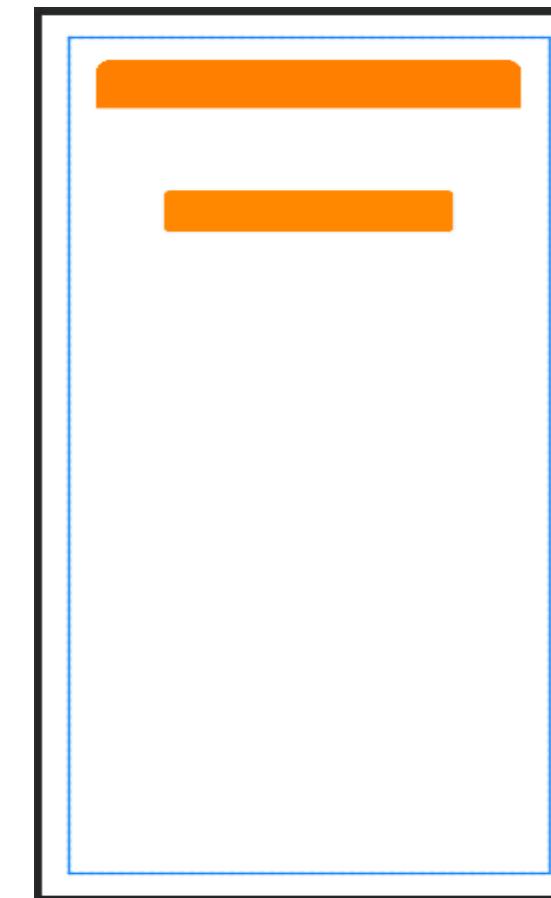
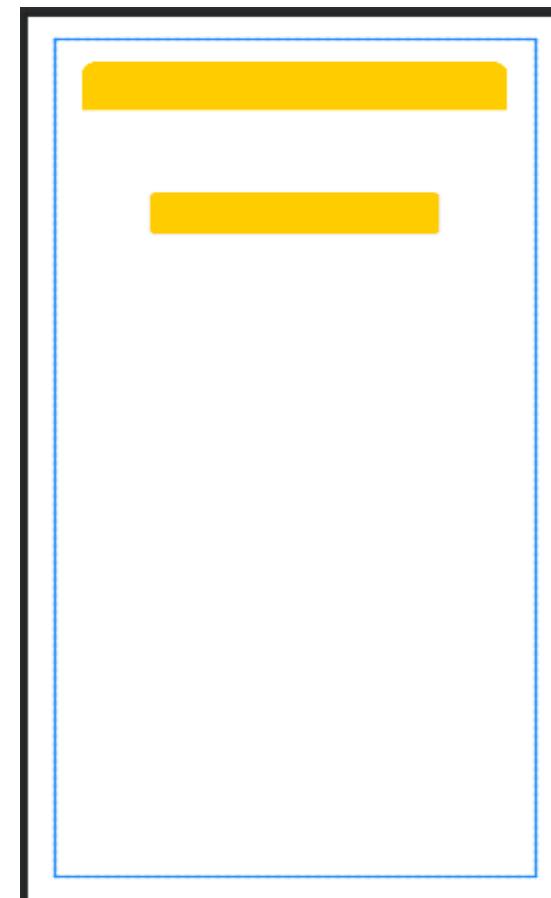
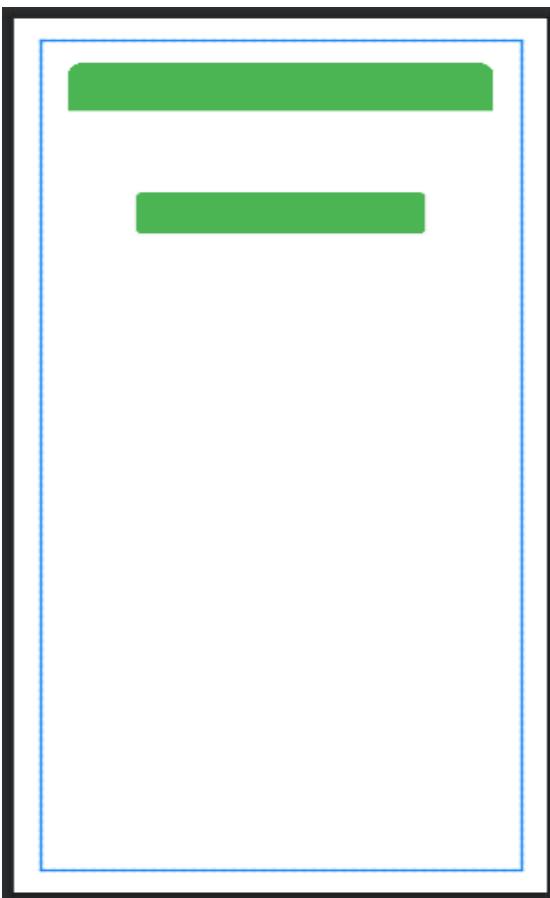
```
public void onDestroy() {  
    //Finaliza la actividad  
    super.onDestroy();  
    conversor.apagar();  
    //TODO  
}
```



ESCUELA  
POLÍTÉCNICA  
NACIONAL

# RESULTADOS – PARTE 5 – ALERTAS VISUALES

- Primer paso es crear la interfaz gráfica de las alertas gráficas, estas alertas se diferencian por su color y estructura. El nivel de alerta viene a ser: bajo, medio, alto y muy alto.





# RESULTADOS – PARTE 5 – ALERTAS VISUALES

- A continuación se presenta un segmento de código referente a los parámetros establecidos en la configuración de las alertas visuales.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Creación de los elementos visuales y sus parametros correspondientes -->
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:id="@+id/contenedorRojo"
    android:layout_margin="20dp"
    android:padding="20dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/layoutDialogRojo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/fondo_color_blanco"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/tituloRojo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/fondo_color_rojo"
            android:padding="10dp">
```



# RESULTADOS – PARTE 5 – ALERTAS VISUALES

- Segmento de código que permite presentar las alertas al usuario según su nivel. El nivel de alerta es enviado desde el agente de respuesta de AWS en base a la información que se le envió del Dataset.

```
final TextView velocidad = vinculador.txtDatosVelocidad;
//Llamando al método establecido en la clase Modelo Vista para obtener la información del Dataset
FragmentoModeloVista.getVelocidad().observe(getViewLifecycleOwner(), new Observer<String>() {
    public void onChanged(@Nullable String s) {
        //Estableciendo el texto de forma visual que se envió del Dataset
        velocidad.setText(s);
    }
});

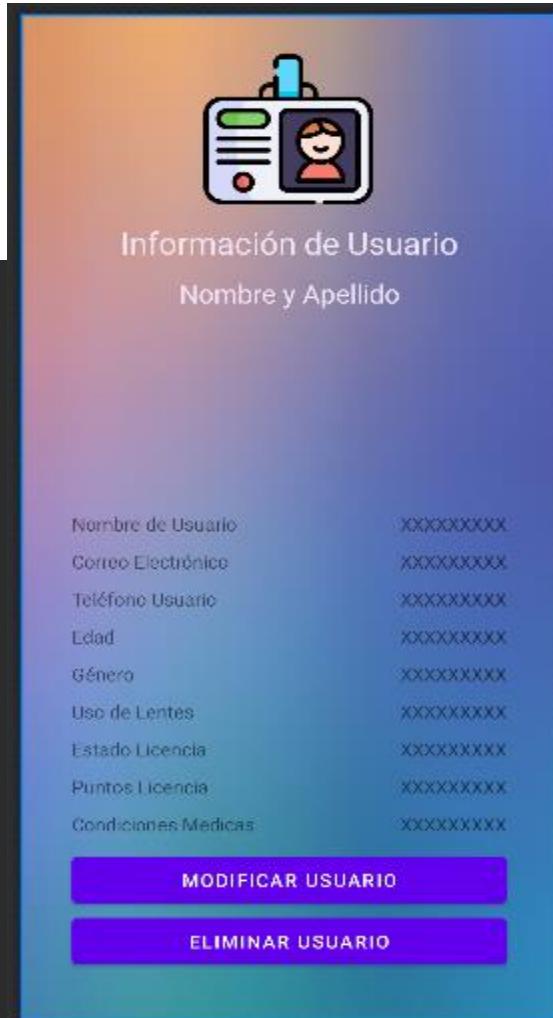
if (s.equals("{\"Output\": 3}") && get_set_Alta) {
    //Presentando las alertas gráficas de nivel 3 al usuario
    resultadoTexto.setText(getString(R.string.medida_datos_alta));
    resultadoBandera.setImageResource(R.drawable.naranja);
    //Iniciando alerta sonora de nivel 3
    String mensaje = "Nivel de Riesgo Alta";
    conversor.palabra(mensaje);
    mostrarNaranja(mensaje);
}
```

# RESULTADOS – PARTE 5 – INFORMACIÓN DE USUARIO

- Dentro del menú lateral principal, una de las opciones que se presenta es la Información de usuario. Esta información permite saber datos relevantes sobre el usuario.
- Desde AWS mediante Amplify y Cognito se puede obtener el nombre de usuario y su correo electrónico.

```
//Segmento de código que permite obtener de los servicios de AWS el correo del usuario actualmente logueado
Amplify.Auth.fetchUserAttributes(attributes -> getActivity().runOnUiThread(new Runnable() {
    public void run() {
        //Mostrando en pantalla el correo del usuario
        correoUsuario.setText(attributes.get(2).getValue());
    }
}), error -> getActivity().runOnUiThread(new Runnable() {
    //Mensaje de consola que informa al usuario que no se pudo obtener el correo del usuario
    public void run() {
        Log.e(tag: "Error", error.toString());
    }
}));

//Segmento de código que permite obtener de los servicios de AWS el nombre de usuario actualmente logueado
Amplify.Auth.getCurrentUser(result -> getActivity().runOnUiThread(new Runnable() {
    public void run() {
        //Mostrando en pantalla el nombre de usuario
        nombreUsuario.setText(result.getUsername());
    }
}), error -> getActivity().runOnUiThread(new Runnable() {
    //Mensaje de consola que informa al usuario que no se pudo obtener el nombre de usuario
    public void run() {
        Log.e(tag: "Error", error.toString());
    }
}));
```





# RESULTADOS – PARTE 6

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Administración del nivel de alertas de usuario
  - Creación de una nueva interfaz de usuario para la administración de alertas.
  - Modificación interfaz de Usuario para la presentación de alertas.
  - Corrección de errores.



# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Primero, se modifica la interfaz de usuario referente a la configuración de usuario, habilitando un botón que permita redirigir al usuario a la nueva interfaz de administración de alertas de usuario.

```
//Creando el evento click en un botón inicializado
configuracion.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        //Mensaje emergente de información del usuario
        String mensaje = "Configuraciones de Usuario";
        Toast.makeText(getApplicationContext(), mensaje, Toast.LENGTH_SHORT).show();
        //Redirigiendo al usuario a la actividad de configuraciones de usuario
        Intent intent = new Intent(getApplicationContext(), ConfiguracionActivity.class);
        startActivity(intent);
    }
});
```



# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Creación de la nueva interfaz de usuario.
- Se muestra segmento de código referente a las propiedades de la actividad.
- Dentro de los parámetros de la actividad se encuentra cuatro botones de tipo “Switch” que posee dos estados. Estado activado y estado desactivado (True and False).

```
<Switch
    android:id="@+id/switchAlto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="20dp"
    android:layout_marginBottom="30dp"
    android:text="Desactivada"
    app:layout_constraintBottom_toTopOf="@+id/switchMuyAlto"
    app:layout_constraintEnd_toEndOf="parent"
    tools:ignore="TextContrastCheck,TouchTargetSizeCheck,UseSwitchCompatOrMaterialXml" />

<Switch
    android:id="@+id/switchMedio"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="20dp"
    android:layout_marginBottom="30dp"
    android:text="Desactivada"
    app:layout_constraintBottom_toTopOf="@+id/switchAlto"
    app:layout_constraintEnd_toEndOf="parent"
    tools:ignore="TextContrastCheck,TouchTargetSizeCheck,UseSwitchCompatOrMaterialXml" />
```



# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Establecida la parte gráfica, se procede a la creación del código que permita almacenar de manera local las configuración del usuario.
- Se hace uso de la librería “SharedPreferences”, que permite la creación, lectura, actualización y eliminación de un archivo local que almacena las configuraciones de la aplicación.

```
import com.proyecto.pis_20_02.clases.menu.MenuLateralActivity;
import androidx.appcompat.app.AppCompatActivity;
import android.content.SharedPreferences;
import android.annotation.SuppressLint;
import com.proyecto.pis_20_02.R;
import android.content.Context;
import android.content.Intent;
import android.widget.Button;
import android.widget.Switch;
import android.widget.Toast;
import android.view.View;
import android.os.Bundle;
```



# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Se muestra segmento de código que permite verificar si el archivo de configuración esta creado. Si el archivo de configuración no esta creado se procede con su creación.

```
//Verificación o Creación del archivo local de configuraciones de Usuario  
SharedPreferences configuracionUsuario = getSharedPreferences( name: "Preferencias", Context.MODE_PRIVATE);
```

- Se procede a crear las variables que permitirán guardar la información requerida por el usuario para su futuro CRUD.

```
//Creación de variables para enviar, presentar y recibir información  
Button guardarConfiguracionesSet;  
Boolean get_set_MuyAlta = false;  
Boolean get_set_Media = false;  
Boolean get_set_Alta = false;  
Boolean get_set_Baja = false;  
Switch muyAlta;  
Switch media;  
Switch alta;  
Switch baja;
```

# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Creadas las variables se procede a almacenar la información en las variables creadas.

```
private void guardarConfiguracion() {  
    //Verificación o Creación del archivo local de configuraciones de Usuario  
    SharedPreferences configuracionUsuario = getSharedPreferences( name: "Preferencias", Context.MODE_PRIVATE);  
    //Obteniendo los datos a almacenar  
    Boolean alertaMuyAlta = get_set_MuyAlta;  
    Boolean alertaMedia = get_set_Media;  
    Boolean alertaAlta = get_set_Alta;  
    Boolean alertaBaja = get_set_Baja;  
    //Creando un objeto tipo editor que permite modificar el archivo Preferencias creado  
    SharedPreferences.Editor editar = configuracionUsuario.edit();  
    editar.putBoolean( s: "Alerta Muy Alta", alertaMuyAlta);  
    editar.putBoolean( s: "Alerta Media", alertaMedia);  
    editar.putBoolean( s: "Alerta Alta", alertaAlta);  
    editar.putBoolean( s: "Alerta Baja", alertaBaja);  
    editar.apply();  
    //Mensaje emergente de información al usuario  
    String informacion = "Configuraciones Guardadas";  
    Toast.makeText(getApplicationContext(), informacion, Toast.LENGTH_SHORT).show();  
}
```



# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Guardada la información se procede a verificar que el archivo se haya creado con la información del usuario.

The screenshot shows the Android Studio interface. On the left, the Project and Structure toolbars are visible. The main editor window displays the Java code for `ConfiguracionActivity.java`. The code handles configuration storage and retrieval using SharedPreferences. A toast message is shown confirming successful configuration saving. The right side of the screen shows the Device File Explorer for an emulator running API 29. It lists the contents of the application's internal storage, including a newly created `Preferencias.xml` file under the `com.proyecto.pis_20_02/shared_prefs` directory, which corresponds to the saved configuration data.

```
47     Switch alta;
48     Switch baja;
49
50     protected void onCreate(Bundle savedInstanceState) {...}
51
52     private void guardarConfiguracion() {
53         //Verificación o Creación del archivo local de configuraciones de Usuario
54         SharedPreferences configuracionUsuario = getSharedPreferences("Preferencias", Context.MODE_PRIVATE);
55         //Obteniendo los datos a almacenar
56         Boolean alertaMuyAlta = get_set_MuyAlta;
57         Boolean alertaMedia = get_set_Media;
58         Boolean alertaAlta = get_set_Alta;
59         Boolean alertaBaja = get_set_Baja;
60         //Creando un objeto tipo editor que permite modificar el archivo Preferencias creado
61         SharedPreferences.Editor editar = configuracionUsuario.edit();
62         editar.putBoolean("Alerta Muy Alta", alertaMuyAlta);
63         editar.putBoolean("Alerta Media", alertaMedia);
64         editar.putBoolean("Alerta Alta", alertaAlta);
65         editar.putBoolean("Alerta Baja", alertaBaja);
66         editar.apply();
67         //Mensaje emergente de información al usuario
68         String informacion = "Configuraciones Guardadas";
69         Toast.makeText(getApplicationContext(), informacion, Toast.LENGTH_SHORT).show();
70     }
71
72     private void cargarConfiguracion() {...}
73
74     protected void onDestroy() {...}
75 }
```

Name	Permissions	Date	Size
com.google.android.packageinstall	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.partnersetup	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.permissioncor	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.printservice.re	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.projection.ges	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.sdksetup	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.setupwizard	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.soundpicker	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.syncadapters.o	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.tts	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.videos	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.webview	drwxrwx--x	2023-04-10 11:37	4 KB
com.google.android.youtube	drwxrwx--x	2023-04-10 11:37	4 KB
com.proyecto.pis_20_02	drwxrwx--x	2023-04-10 11:37	4 KB
cache	drwxrws--x	2023-04-14 12:33	4 KB
code_cache	drwxrws--x	2023-04-10 11:42	4 KB
files	drwxrwx--x	2023-04-17 10:49	4 KB
no_backup	drwxrwx--x	2023-04-10 11:43	4 KB
shared_prefs	drwxrwx--x	2023-04-17 09:59	4 KB
com.amplify.credentialStore	-rw-rw----	2023-04-17 09:59	9 KB
Preferencias.xml	-rw-rw----	2023-04-17 10:53	269 B
com.ustwo.lwp	drwxrwx--x	2023-04-10 11:37	4 KB
local	drwxrwx--x	2023-04-10 11:37	4 KB

Bottom navigation bar: Version Control, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, App Inspection, Layout Inspector.

Bottom status bar: Launch succeeded (today 12:34), 235:6 LF, UTF-8, 4 spaces, lock icon.



# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Verificando el archivo creado con la información del usuario.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name="Alerta Muy Alta" value="false" />
    <boolean name="Alerta Media" value="false" />
    <boolean name="Alerta Alta" value="false" />
    <boolean name="Alerta Baja" value="false" />
</map>
```

# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Carga de la información en la misma interfaz, si se desea modificar nuevamente la configuración de usuario.

```
private void cargarConfiguracion() {  
    //Verificación o Creación del archivo local de configuraciones de Usuario  
    SharedPreferences configuracionUsuario = getSharedPreferences( name: "Preferencias", Context.MODE_PRIVATE);  
    //Obteniendo los datos a presentar  
    boolean alertaMuyAlta = configuracionUsuario.getBoolean( s: "Alerta Muy Alta", b: false);  
    boolean alertaMedia = configuracionUsuario.getBoolean( s: "Alerta Media", b: false);  
    boolean alertaAlta = configuracionUsuario.getBoolean( s: "Alerta Alta", b: false);  
    boolean alertaBaja = configuracionUsuario.getBoolean( s: "Alerta Baja", b: false);  
    //Cambiando la información visual de los elementos  
    muyAlta.setChecked(alertaMuyAlta);  
    get_set_MuyAlta = alertaMuyAlta;  
    media.setChecked(alertaMedia);  
    get_set_Media = alertaMedia;  
    alta.setChecked(alertaAlta);  
    baja.setChecked(alertaBaja);  
    get_set_Alta = alertaAlta;  
    get_set_Baja = alertaBaja;  
    //Verificación texto a presentar Alerta Activada o Desactivada  
    if (alertaMuyAlta) {...} else {...}  
    if (alertaMedia) {...} else {...}  
    if (alertaAlta) {...} else {...}  
    if (alertaBaja) {...} else {...}  
}
```

# RESULTADOS – PARTE 6 – ADMINISTRACIÓN DE ALERTAS DE USUARIO

- Llamada a la configuración en la actividad principal para mostrar las alertas según las configuraciones del usuario.

```
//Verificación o Creación del archivo local de configuraciones de Usuario
SharedPreferences configuracionUsuario = getActivity().getSharedPreferences( name: "Preferencias", Context.MODE_PRIVATE);
//Obteniendo los datos a presentar
boolean alertaMuyAlta = configuracionUsuario.getBoolean( s: "Alerta Muy Alta", b: false);
boolean alertaMedia = configuracionUsuario.getBoolean( s: "Alerta Media", b: false);
boolean alertaAlta = configuracionUsuario.getBoolean( s: "Alerta Alta", b: false);
boolean alertaBaja = configuracionUsuario.getBoolean( s: "Alerta Baja", b: false);
//Cambiando la información visual de los elementos
get_set_MuyAlta = alertaMuyAlta;
get_set_Media = alertaMedia;
get_set_Alta = alertaAlta;
get_set_Baja = alertaBaja;
//Comparando el valor recibido con el nivel 4
if (s.equals("{\"Output\": 4}") && get_set_MuyAlta) {
    //Presentando las alertas gráficas de nivel 4 al usuario
    resultadoTexto.setText(getString(R.string.medida_datos_muy_alta));
    resultadoTexto.setTextSize(24);
    resultadoBandera.setImageResource(R.drawable.rojo);
    //Iniciando alerta sonora de nivel 4
    String mensaje = "Nivel de Riesgo Muy Alta";
    conversor.palabra(mensaje);
    mostrarRojo(mensaje);
}
```



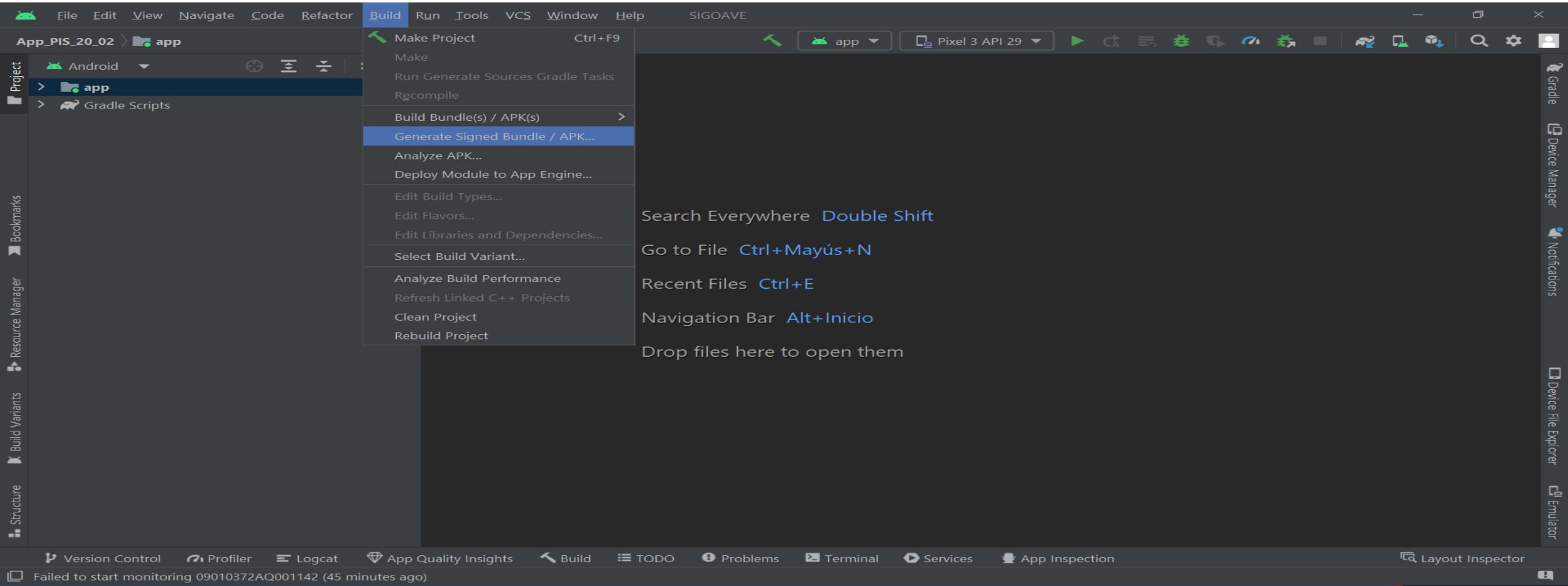
# RESULTADOS – PARTE 7

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Corrección de errores.
  - Creación de un APK Firmada para su uso.



# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

- Buscar la opción “Generate Signed APK” en el apartado: Build → Generated Signed Bundle / APK y dar clic.





# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

- En la nueva venta se debe escoger la opción de “APN” y presionar en siguiente.
- Tras esto se presentara una nueva ventana donde se debe proporcionar una llave, alias y contraseña.

Generate Signed Bundle or APK

**Android App Bundle**

Generate a signed app bundle for upload to app stores for the following benefits:

- Smaller download size
- On-demand app features
- Asset-only modules

[Learn more](#)

**APK**

Build a signed APK that you can deploy to a device

[?](#) [Next](#) [Cancel](#)

Generate Signed Bundle or APK

Module

Key store path

[Create new...](#) [Choose existing...](#)

Key store password

Key alias

Key password

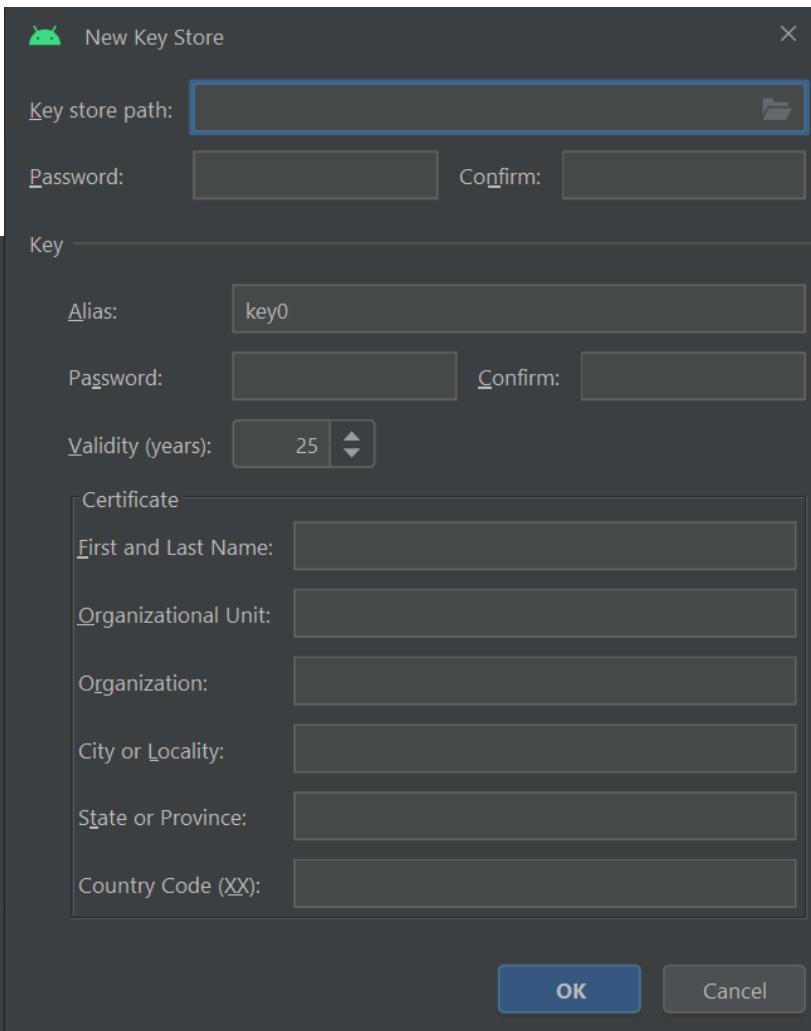
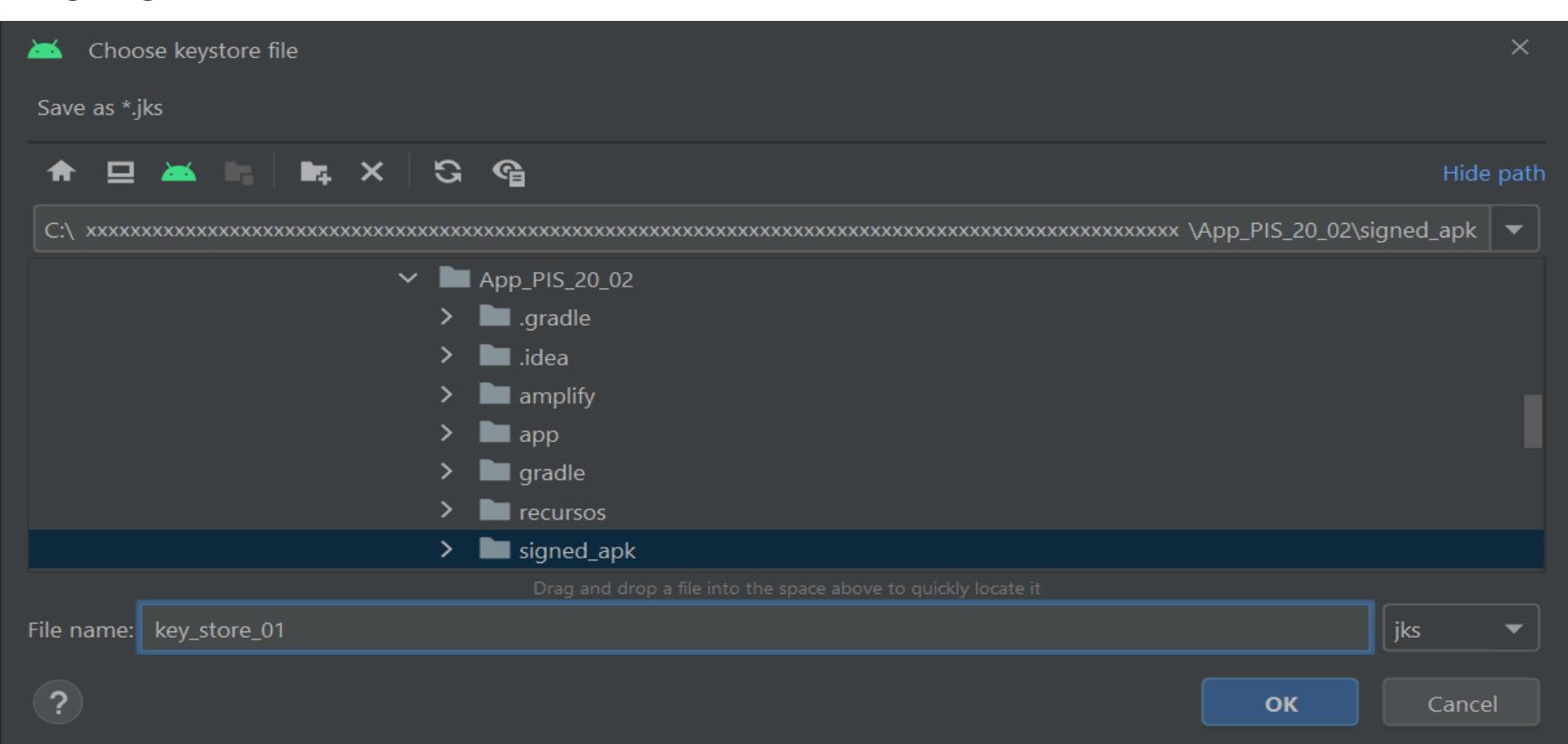
Remember passwords

[?](#) [Previous](#) [Next](#) [Cancel](#)



# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

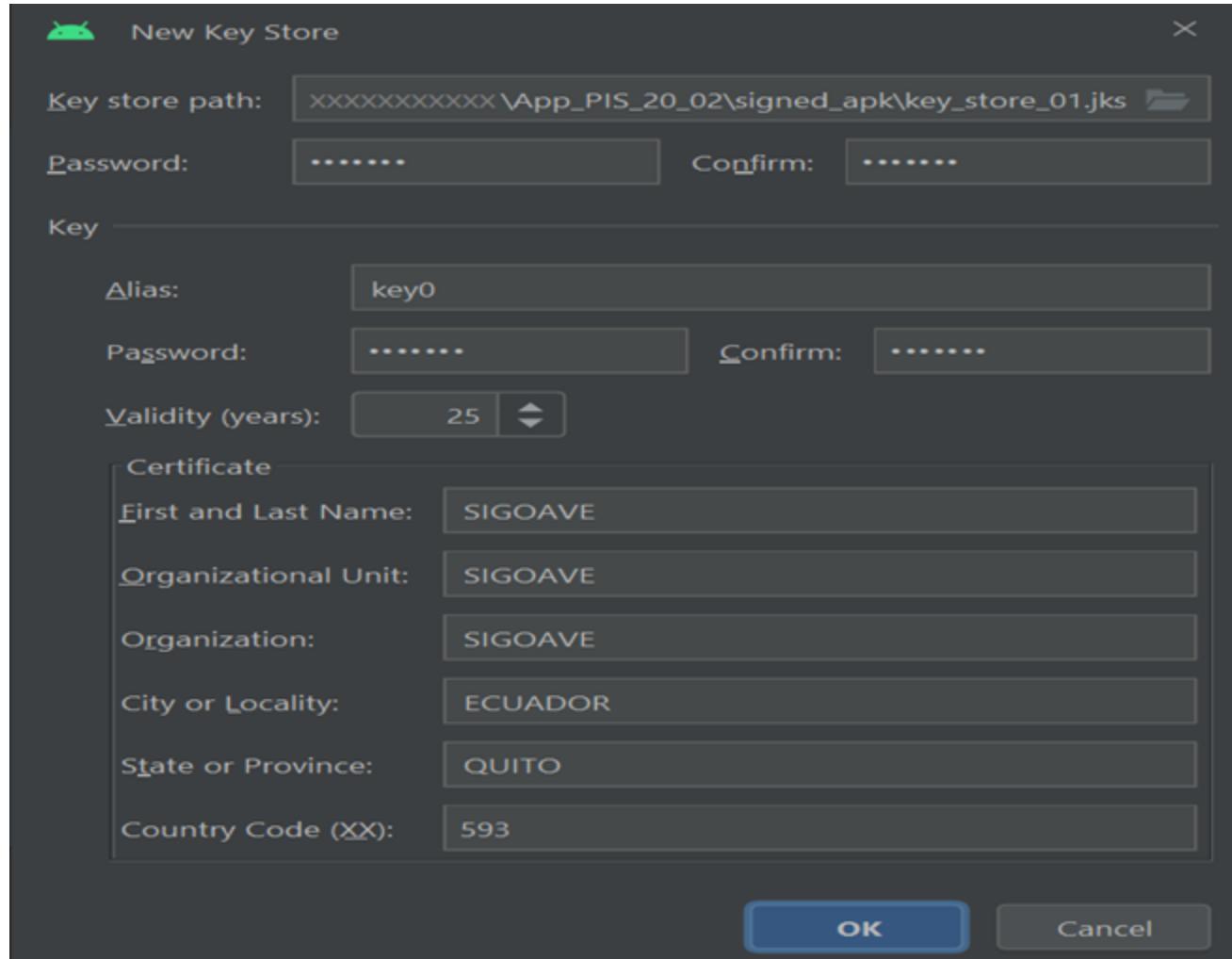
- En este apartado se debe presionar en “Create New”. En la nueva ventana en el parámetro “Key store path” se debe escoger un ruta o carpeta donde se almacenara y crear la APK del proyecto, dar un nombre al archivo a crear y finalmente presionar en “OK”.





# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

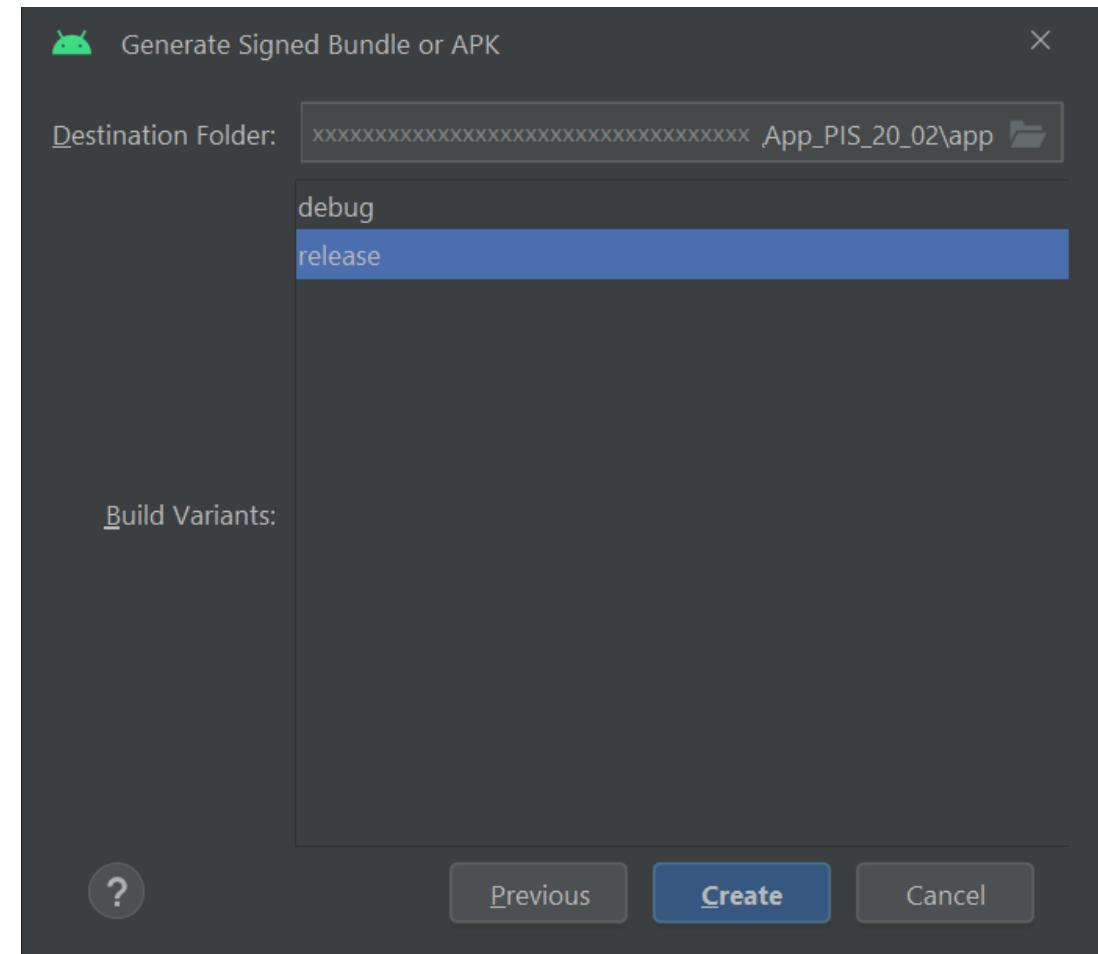
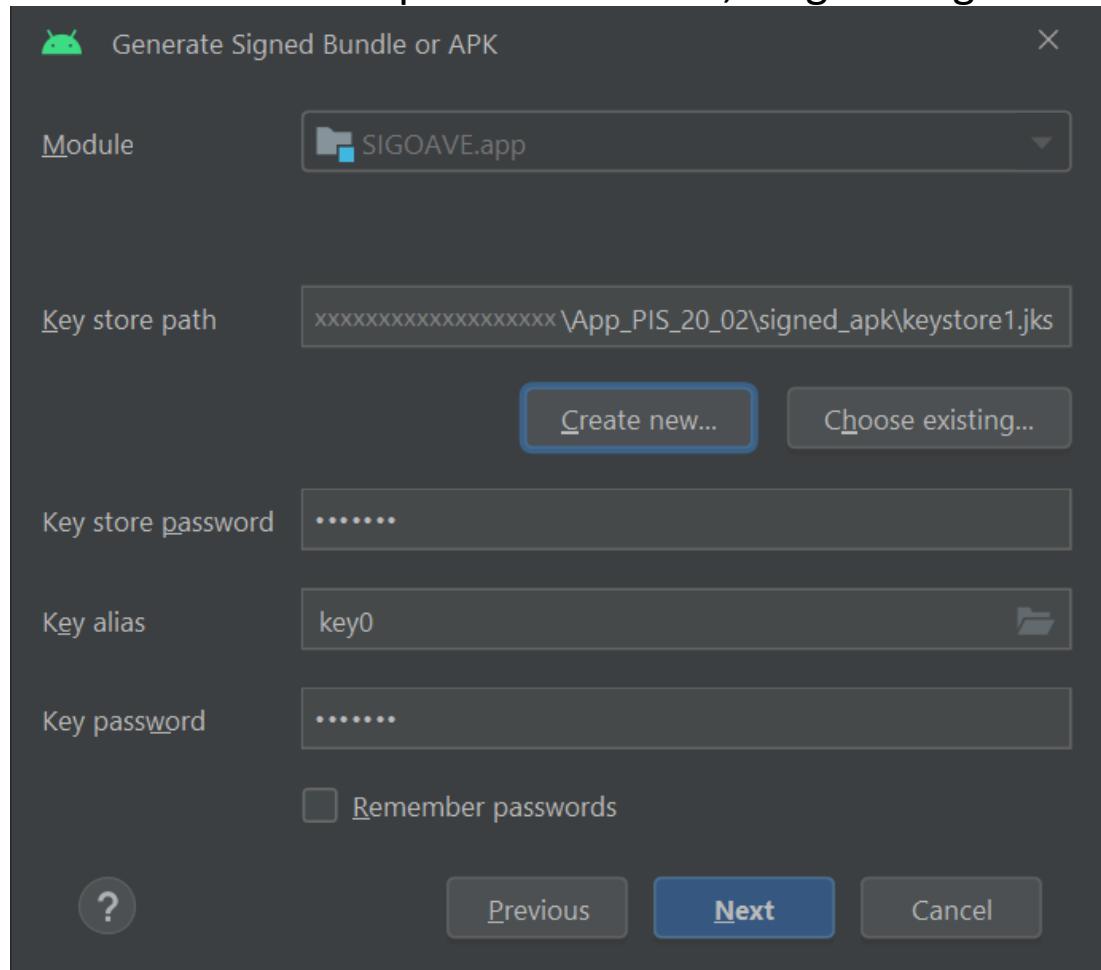
- En la siguiente ventana se debe proporcionar la información solicitada como:
  - Contraseña
  - Confirmar contraseña
  - Alias
  - Contraseña de alias
  - Confirmar contraseña de alias
  - Nombre
  - Apellido
  - Organización
  - Ciudad
  - Provincia
  - Código de país.
- Finalmente se presiona en “OK”.





# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

- En la nueva ventana presionar “Next”, luego escoger “Release” .y presionar en “Create”.





# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

- Finalmente, el sistema informa que se creo la APK.

A screenshot of the Android Studio interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, and SIGOAVE. The Project tool window on the left shows a tree with 'app' selected. The main editor area is dark and displays a message: "Search Everywhere Double Shift", "Go to File Ctrl+Mayús+N", "Recent Files Ctrl+E", "Navigation Bar Alt+Inicio", and "Drop files here to open them". A notification bar at the bottom right shows a green info icon with the text: "Generate Signed APK" and "APK(s) generated successfully for module 'SIGOAVE.app.main' with 1 build variant: Build variant 'debug': locate or analyze the APK.". The bottom navigation bar includes Version Control, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, App Inspection, Layout Inspector, and a status message: "Generate Signed APK: APK(s) generated successfully for module 'SIGOAVE.app.main' with 1 build variant: // Build variant 'debug': locate or analyze the APK. (moments ago)".

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help SIGOAVE

App\_PIS\_20\_02 > app Pixel 3 API 29

Project

Android Scripts

Search Everywhere Double Shift

Go to File Ctrl+Mayús+N

Recent Files Ctrl+E

Navigation Bar Alt+Inicio

Drop files here to open them

Generate Signed APK

APK(s) generated successfully for module 'SIGOAVE.app.main' with 1 build variant: Build variant 'debug': locate or analyze the APK.

Version Control Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector

Generate Signed APK: APK(s) generated successfully for module 'SIGOAVE.app.main' with 1 build variant: // Build variant 'debug': locate or analyze the APK. (moments ago)



# RESULTADOS – PARTE 7 – GENERACIÓN APK FIRMADA

- Se puede buscar la APK en la ubicación creada.

A screenshot of the Windows File Explorer interface. The address bar shows the path: "Este equipo >xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx App\_PIS\_20\_02 > app > debug". The left sidebar shows the navigation tree with "Escritorio" selected. The main pane displays a table with two files:

Nombre	Fecha de modificación	Tipo	Tamaño
app-debug	18/4/2023 14:35	Android app file	12.424 KB
output-metadata.json	18/4/2023 14:35	Archivo JSON	1 KB

2 elementos

# RESULTADOS – PARTE 8

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Corrección de errores.
  - Integración con Amazon DynamoDB de AWS.



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence actualizarUsuario(String puntos, String estado, String user, String telefono, String correo, String edad, String apellido,  
//Crear un nuevo proveedor de credenciales  
dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
// Agregando la lista de atributos a la base de datos  
// La lista de atributos son para actualizar un usuario  
Map<String, AttributeValue> nuevoUsuario = new HashMap<>();  
nuevoUsuario.put( k: "PuntosLicencia", new AttributeValue().withN(puntos));  
nuevoUsuario.put( k: "EstadoLicencia", new AttributeValue(estado));  
nuevoUsuario.put( k: "UserName", new AttributeValue(user));  
nuevoUsuario.put( k: "Telefono", new AttributeValue().withN(telefono));  
nuevoUsuario.put( k: "TipoUsuario", new AttributeValue( s: "Cliente"));  
nuevoUsuario.put( k: "Correo", new AttributeValue(correo));  
nuevoUsuario.put( k: "Edad", new AttributeValue().withN(edad));  
nuevoUsuario.put( k: "Apellido", new AttributeValue(apellido));  
nuevoUsuario.put( k: "Medicas", new AttributeValue(medicas));  
nuevoUsuario.put( k: "Nombre", new AttributeValue(nombre));  
nuevoUsuario.put( k: "Genero", new AttributeValue(genero));  
nuevoUsuario.put( k: "Lentes", new AttributeValue(lentes));  
  
// Realizando la petición a la base de datos para actualizar un elemento  
PutItemRequest peticion = new PutItemRequest();  
// Enviando el nombre de la Tabla de la base de datos  
peticion.withTableName(DYNAMODB_TABLE_DATA).withItem(nuevoUsuario);  
// Se adquiere la respuesta desde la base de datos  
PutItemResult respuesta = dbClient_1.putItem(peticion);
```



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence agregarUsuario(String usernameRecibido, String correoRecibido) {  
    //Crear un nuevo proveedor de credenciales  
    dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
    dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
    // Agregando la lista de atributos a la base de datos  
    // La lista de atributos son por default al crear un nuevo usuario  
    Map<String, AttributeValue> nuevoUsuario = new HashMap<>();  
    nuevoUsuario.put( k: "PuntosLicencia", new AttributeValue().withN( n: "-1" ));  
    nuevoUsuario.put( k: "EstadoLicencia", new AttributeValue( s: "No Ingresado" ));  
    nuevoUsuario.put( k: "UserName", new AttributeValue(usernameRecibido));  
    nuevoUsuario.put( k: "Telefono", new AttributeValue().withN( n: "-1" ));  
    nuevoUsuario.put( k: "TipoUsuario", new AttributeValue( s: "Cliente" ));  
    nuevoUsuario.put( k: "Correo", new AttributeValue(correoRecibido));  
    nuevoUsuario.put( k: "Edad", new AttributeValue().withN( n: "-1" ));  
    nuevoUsuario.put( k: "Apellido", new AttributeValue( s: "No Ingresado" ));  
    nuevoUsuario.put( k: "Medicas", new AttributeValue( s: "No Ingresado" ));  
    nuevoUsuario.put( k: "Nombre", new AttributeValue( s: "No Ingresado" ));  
    nuevoUsuario.put( k: "Genero", new AttributeValue( s: "No Ingresado" ));  
    nuevoUsuario.put( k: "Lentes", new AttributeValue( s: "No Ingresado" ));  
  
    // Realizando la petición a la base de datos para agregar un nuevo elemento  
    PutItemRequest peticion = new PutItemRequest();  
    // Enviando el nombre de la Tabla de la base de datos  
    peticion.withTableName(DYNAMODB_TABLE_DATA).withItem(nuevoUsuario);  
    // Se adquiere la respuesta desde la base de datos  
    PutItemResult respuesta = dbClient_1.putItem(peticion);  
  
    // Enviando al usuario la respuesta de la base de datos  
    return respuesta.toString();
```



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence eliminarUsuario(String usernameRecibido, String correoRecibido) {  
    //Crear un nuevo proveedor de credenciales  
    dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
    dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
    // Agregando la lista de atributos a la base de datos  
    // La lista de atributos son para actualizar un usuario  
    Map<String, AttributeValue> borrarUsuario = new HashMap<>();  
    borrarUsuario.put( k: "UserName", new AttributeValue().withS(usernameRecibido));  
    borrarUsuario.put( k: "Correo", new AttributeValue().withS(correoRecibido));  
  
    // Realizando la petición a la base de datos para actualizar un elemento  
    DeleteItemRequest peticion = new DeleteItemRequest();  
    // Enviando el nombre de la Tabla de la base de datos  
    peticion.withTableName(DYNAMODB_TABLE_DATA).withKey(borrarUsuario);  
    // Se adquiere la respuesta desde la base de datos  
    DeleteItemResult respuesta = dbClient_1.deleteItem(peticion);  
  
    // Enviando al usuario la respuesta de la base de datos  
    return respuesta.toString();  
}
```



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence obtenerUsuario(String usernameRecibido) {  
    //Crear un nuevo proveedor de credenciales  
    dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
    dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
    // Realizando una petición a la base de datos para adquirir los datos del usuario  
    ScanRequest peticion = new ScanRequest(DYNAMODB_TABLE_DATA);  
    // Obteniendo la respuesta desde la base de datos  
    ScanResult respuesta = dbClient_1.scan(peticion);  
    // Agregando los atributos adquiridos desde la base de datos a una lista  
    List<Map<String, AttributeValue>> elementos = respuesta.getItems();  
  
    // Creando una nueva lista para enviar los atributos al usuario  
    List<String> atributos = new ArrayList<>();  
    // Recorriendo la lista obtenida de la base de datos  
    for (Map<String, AttributeValue> map : elementos) {  
        // Creando variables para agregar a la lista de elementos  
        String PuntosLicencia = Objects.requireNonNull(map.get("PuntosLicencia")).getN(); //0  
        String EstadoLicencia = Objects.requireNonNull(map.get("EstadoLicencia")).getS(); //1  
        String TipoUsuario = Objects.requireNonNull(map.get("TipoUsuario")).getS(); //2  
        String Apellido = Objects.requireNonNull(map.get("Apellido")).getS(); //3  
        String Telefono = Objects.requireNonNull(map.get("Telefono")).getN(); //4  
        String UserName = Objects.requireNonNull(map.get("UserName")).getS(); //5  
        String Medicas = Objects.requireNonNull(map.get("Medicas")).getS(); //6  
        String Correo = Objects.requireNonNull(map.get("Correo")).getS(); //7  
        String Nombre = Objects.requireNonNull(map.get("Nombre")).getS(); //8  
        String Genero = Objects.requireNonNull(map.get("Genero")).getS(); //9
```



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence datosDataSetDynamoDB(int identificador) {  
    //Crear un nuevo proveedor de credenciales  
    dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
    dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
    // Realizando una petición a la base de datos para adquirir los datos del usuario  
    ScanRequest peticion = new ScanRequest(DYNAMODB_TABLE_DATASET);  
    // Obteniendo la respuesta desde la base de datos  
    ScanResult respuesta = dbClient_1.scan(peticion);  
    // Agregando los atributos adquiridos desde la base de datos a una lista  
    List<Map<String,AttributeValue>> elementos = respuesta.getItems();  
  
    // Creando una nueva lista para enviar los atributos al usuario  
    List<String> atributos = new ArrayList<>();  
    // Recorriendo la lista obtenida de la base de datos  
    for (Map<String,AttributeValue> map : elementos) {  
        // Creando variables para agregar a la lista de elementos  
        String Id = Objects.requireNonNull(map.get("Id")).getN();  
        String time = Objects.requireNonNull(map.get("time")).getS();  
        String steering_angle = Objects.requireNonNull(map.get("steering_angle")).getN();  
        String speed = Objects.requireNonNull(map.get("speed")).getN();  
        String rpm = Objects.requireNonNull(map.get("rpm")).getN();  
        String acceleration = Objects.requireNonNull(map.get("acceleration")).getN();  
        String throttle_position = Objects.requireNonNull(map.get("throttle_position")).getN();  
        String engine_temperature = Objects.requireNonNull(map.get("engine_temperature")).getN();  
        String system_voltage = Objects.requireNonNull(map.get("system_voltage")).getN();  
        String barometric_pressure = Objects.requireNonNull(map.get("barometric_pressure")).getN();  
        String distance_travelled = Objects.requireNonNull(map.get("distance_travelled")).getN();  
    }  
}
```



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence obtenerAlarma(int identificador) {  
    //Crear un nuevo proveedor de credenciales  
    dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
    dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
    // Realizando una petición a la base de datos para adquirir los datos del usuario  
    ScanRequest peticion = new ScanRequest(DYNAMODB_TABLE_ALERTAS);  
    // Obteniendo la respuesta desde la base de datos  
    ScanResult respuesta = dbClient_1.scan(peticion);  
    // Agregando los atributos adquiridos desde la base de datos a una lista  
    List<Map<String,AttributeValue>> elementos = respuesta.getItems();  
  
    // Creando una nueva lista para enviar los atributos al usuario  
    List<String> atributos = new ArrayList<>();  
    // Recorriendo la lista obtenida de la base de datos  
    for (Map<String,AttributeValue> map : elementos) {  
        // Creando variables para agregar a la lista de elemento  
        String Alerta = Objects.requireNonNull(map.get("Alerta")).getS();  
        String Id = Objects.requireNonNull(map.get("Id")).getN();  
  
        // Agregando los elementos a la lista  
        if (Integer.parseInt(Id) == identificador) {  
            atributos.add(Alerta);  
        }  
    }  
    // Enviando al usuario los atributos  
    return atributos.toString();  
}
```



# RESULTADOS – PARTE 8 – INTEGRACIÓN CON Amazon DynamoDB

```
public CharSequence obtenerId() {  
    //Crear un nuevo proveedor de credenciales  
    dbClient_1 = new AmazonDynamoDBClient(new BasicAWSCredentials(ACCESS_KEY, SECRET_KEY));  
    dbClient_1.setRegion(Region.getRegion(Regions.US_WEST_1));  
  
    // Realizando una petición a la base de datos para adquirir los datos del usuario  
    ScanRequest peticion = new ScanRequest(DYNAMODB_TABLE_DATASET);  
    // Obteniendo la respuesta desde la base de datos  
    ScanResult respuesta = dbClient_1.scan(peticion);  
    // Agregando los atributos adquiridos desde la base de datos a una lista  
    List<Map<String,AttributeValue>> elementos = respuesta.getItems();  
  
    // Creando una nueva lista para enviar los atributos al usuario  
    List<String> atributos = new ArrayList<>();  
    // Recorriendo la lista obtenida de la base de datos  
    for (Map<String,AttributeValue> map : elementos) {  
        // Creando variables para agregar a la lista de elemento  
        String Id = Objects.requireNonNull(map.get("Id")).getN();  
        // Agregando los elementos a la lista  
        atributos.add(Id);  
    }  
  
    // Enviando al usuario los atributos  
    return atributos.toString();  
}
```



# RESULTADOS – PARTE 10

- Pasos a seguir en esta etapa de desarrollo del Agente de Respuesta – SIGOAVE.
  - Corrección de errores.
  - Integración con Amazon DynamoDB de AWS.



# RESULTADOS – PARTE 10 – INTEGRACIÓN CON Amazon DynamoDB

```
public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    //Creación y presentación visual de la actividad
    final View view = inflater.inflate(R.layout.fragmento_lateral_usuario, container, false);
    //Inicializando las variables con los elementos de la actividad
    medicasUsuario = view.findViewById(R.id.respuestaCondicionesMedicas);
    nombreApellidoUsuario = view.findViewById(R.id.txtNombreyApellido);
    licenciaUsuario = view.findViewById(R.id.respuestaEstadoLicencia);
    puntosUsuario = view.findViewById(R.id.respuestaPuntosLicencia);
    nombreUsuario = view.findViewById(R.id.respuestaUsuarioUser);
    telefonoUsuario = view.findViewById(R.id.respuestaTelefono);
    modificarUsuario = view.findViewById(R.id.modificarUsuario);
    eliminarUsuario = view.findViewById(R.id.eliminarUsuario);
    correoUsuario = view.findViewById(R.id.respuestaCorreo);
    generoUsuario = view.findViewById(R.id.respuestaGenero);
    lentesUsuario = view.findViewById(R.id.respuestaLentes);
    edadUsuario = view.findViewById(R.id.respuestaEdad);

    //Segmento de código que permite obtener las credenciales del usuario y establecer conexión con la base de datos
    //Segmento de código utilizado para verificar si el usuario ya había iniciado sesión previamente
    Amplify.Auth.fetchAuthSession(result -> getActivity().runOnUiThread(new Runnable() {
        //Segmento de código que se ejecuta en un hilo secundario para verificar la conexión con los servicios de AWS
        public void run() {
            //Mensaje de consola que informa al usuario que se estableció conexión con los servicios de AWS
            Log.i(tag: "Información AWS: ", result.toString());
            try {
                //Segmento de código que permite obtener de los servicios de AWS el nombre de usuario actualmente logueado
                //Segmento de código que verifica si el usuario ya había iniciado sesión previamente
                Amplify.Auth.getCurrentUser(result -> getActivity().runOnUiThread(new Runnable() {
```



# RESULTADOS – PARTE 10 – INTEGRACIÓN CON Amazon DynamoDB

```
Amplify.Auth.fetchAuthSession(result -> getActivity().runOnUiThread(new Runnable() {...}), error -> getActivity().runOnUiThread(new Runnable() {
    public void run() {
        //Mensaje de consola que informa al usuario que no se pudo obtener el nombre de usuario
        Log.e( tag: "Error AWS: ", error.toString());
    }
}));

//Creando el evento click en un botón inicializado
modificarUsuario.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        //Mensaje emergente de información de usuario
        String mensaje = "Perfil de Usuario";
        Toast.makeText(getActivity(), mensaje, Toast.LENGTH_SHORT).show();
        //Redirigiendo al usuario a la actividad de perfil de usuario
        Intent intent = new Intent(getActivity(), UsuarioActivity.class);
        startActivity(intent);
    }
});

//Creando el evento click en un botón inicializado
eliminarUsuario.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        //Mensaje especial de advertencia de información al usuario con duración de 5 segundos
        String mensaje = "¿Desea Eliminar el Usuario?" + "\n" + "Esta acción no se puede deshacer";
        //Invocando el mensaje de advertencia
        mostrarRojo(mensaje);
    }
});
```



# RESULTADOS – PARTE 10 – INTEGRACIÓN CON Amazon DynamoDB

```
/RegExRedundantEscape/
public class UsuarioActivity extends AppCompatActivity {
    //Inicializando las variables con los elementos de la actividad
    String seleccionado_Genero;
    String seleccionado_Lentes;
    String seleccionado_Estado;
    String seleccionado_Puntos;
    String seleccionado_Medica;
    TextView telefonoUsuario;
    TextView apellidoUsuario;
    String seleccionado_Edad;
    Button modificarUsuario;
    TextView correoUsuario;
    TextView nombreUsuario;
    Spinner spinnerGenero;
    Spinner spinnerLentes;
    Spinner spinnerEstado;
    Spinner spinnerPuntos;
    Spinner spinnerMedica;
    Spinner spinnerEdad;
    TextView userNombre;
```



# RESULTADOS – PARTE 10 – INTEGRACIÓN CON Amazon DynamoDB

```
protected void onCreate(Bundle savedInstanceState) {
    //Creación y presentación visual de la actividad
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_usuario);

    //Inicializando las variables con los elementos de la actividad
    spinnerMedica = findViewById(R.id.respuestaCondicionesMedicas_Perfil);
    spinnerEstado = findViewById(R.id.respuestaEstadoLicencia_Perfil);
    spinnerPuntos = findViewById(R.id.respuestaPuntosLicencia_Perfil);
    modificarUsuario = findViewById(R.id.modificarUsuario_Perfil);
    apellidoUsuario = findViewById(R.id.respuestaApellido_Perfil);
    telefonoUsuario = findViewById(R.id.respuestaTelefono_Perfil);
    userName = findViewById(R.id.respuestaUsuarioUser_Perfil);
    spinnerGenero = findViewById(R.id.respuestaGenero_Perfil);
    spinnerLentes = findViewById(R.id.respuestaLentes_Perfil);
    correoUsuario = findViewById(R.id.respuestaCorreo_Perfil);
    nombreUsuario = findViewById(R.id.respuestaNombre_Perfil);
    spinnerEdad = findViewById(R.id.respuestaEdad_Perfil);

    Amplify.Auth.fetchAuthSession(result -> UsuarioActivity.this.runOnUiThread(new Runnable() {
        //Segmento de código que se ejecuta en un hilo secundario para verificar la conexión con los servicios de AWS
        public void run() {
            //Mensaje de consola que informa al usuario que se estableció conexión con los servicios de AWS
            Log.i( tag: "Información AWS: ", result.toString());
            try {
                //Segmento de código que permite obtener de los servicios de AWS el nombre de usuario actualmente logueado
                //Segmento de código que verifica si el usuario ya había iniciado sesión previamente
                Amplify.Auth.getCurrentUser(result -> UsuarioActivity.this.runOnUiThread(new Runnable() {
                    public void run() {
```



# RESULTADOS – PARTE 10 – INTEGRACIÓN CON Amazon DynamoDB

```
//Llamando a los métodos que permite leer los elementos en el spinner Medica
spinnerMedica.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        //Obteniendo el valor seleccionado de spinner
        seleccionado_Medica = parent.getSelectedItem().toString();
    }

    public void onNothingSelected(AdapterView<?> parent) {
        //TODO
    }
});

//Llamando a los métodos que permite leer los elementos en el spinner Estado
spinnerEstado.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        //Obteniendo el valor seleccionado de spinner
        seleccionado_Estado = parent.getSelectedItem().toString();
    }

    public void onNothingSelected(AdapterView<?> parent) {
        //TODO
    }
});

//Llamando a los métodos que permite leer los elementos en el spinner Puntos
spinnerPuntos.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```