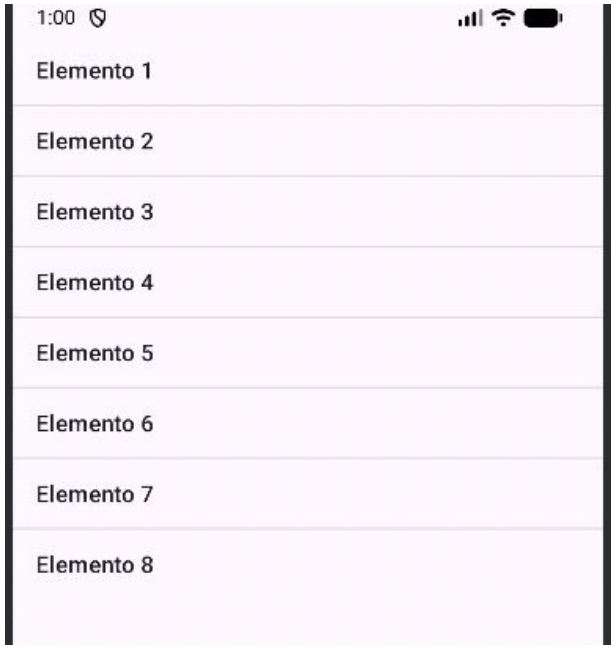
U4. Componentes, Menú y Mensajes



AE U4. Revisión de Componentes y Menú

Para poder comprobar el avance en el módulo, deberás entregar un documento PDF con la siguiente estructura:

1. **Portada**
2. **Índice**
3. **Introducción**
4. **Desarrollo (Con los pantallazos y explicación de cada elemento que revisemos en clase).**
5. **Conclusión**
6. **Webgrafía, Bibliografía e IA**

Valoraré hasta 5 puntos la elaboración del documento con todo los desarrollos realizados, hasta 2 puntos la estructura, legibilidad, maquetación, diseño y originalidad del mismo y hasta 3 puntos de creatividad adicional, utilidad del documento y otros aspectos adicionales a la propuesta.

Los RA y CE que aplican esta actividad son:

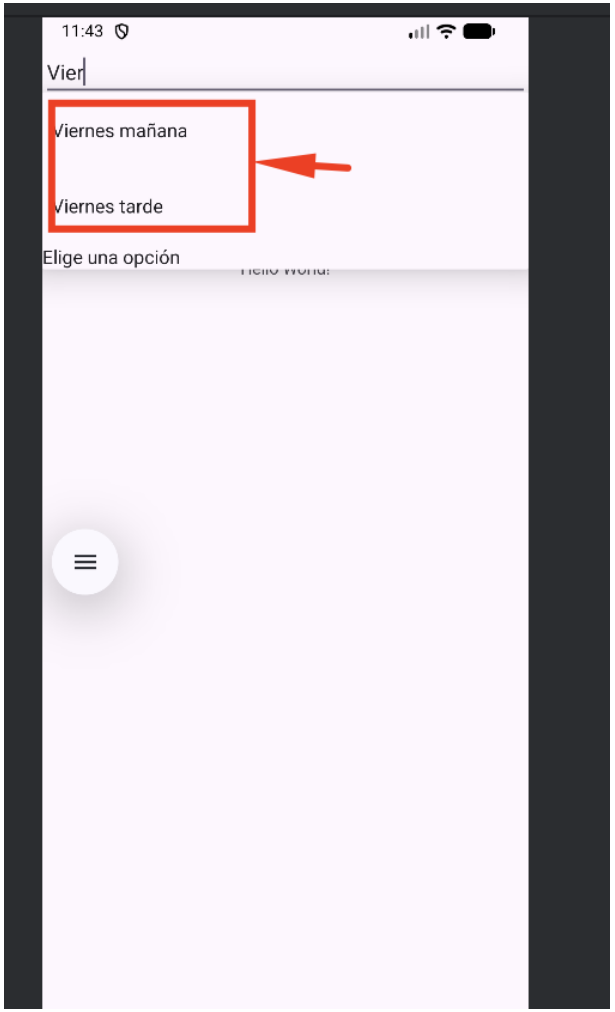
2.b) Se han analizado y utilizado las clases que modelan ventanas, menús, alertas y controles para el desarrollo de aplicaciones gráficas sencillas.
2.d) Se han desarrollado aplicaciones que hacen uso de las funcionalidades proporcionadas por los sensores.
2.f) Se han utilizado las clases necesarias para establecer conexiones con almacenes de datos garantizando la persistencia.
3.a) Se han analizado entornos de desarrollo multimedia

U4.1 Componentes:

Tema 5

+ AutoCompleteTexView
+ SeekBar
+ RatingBar

U4.1 Componentes: AutocompleteTextView



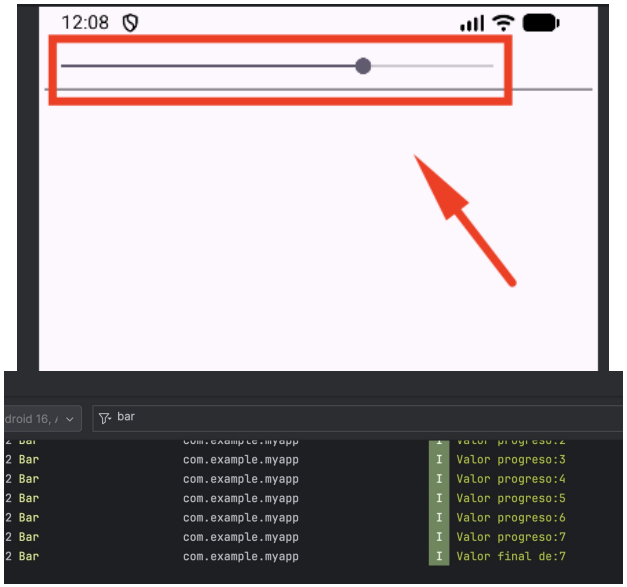
Objeto variante al anterior que añade sugerencia de autocompletado.

```
<AutoCompleteTextView
    android:id="@+id/miTexto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:completionHint="Elige un día de la semana"
    android:completionThreshold="4"
    tools:ignore="MissingConstraints" />
```

```
String[] opciones = { "Lunes", "Martes", "Miércoles", "Jueves", "Viernes
mañana", "Viernes tarde", "Sábado", "Domingo"};
AutoCompleteTextView textoLeido = (AutoCompleteTextView)
findViewById(R.id.miTexto);
ArrayAdapter<String> adapador =  new ArrayAdapter<String> (this,
android.R.layout.simple_dropdown_item_1line, opciones);
textoLeido.setAdapter(adapador);
```

U4.1 Componentes: SeekBar

Se trata de un control de selección que incluye tres eventos: onStartTrackingTouch, onProgressChanged y onStopTrackingTouch.

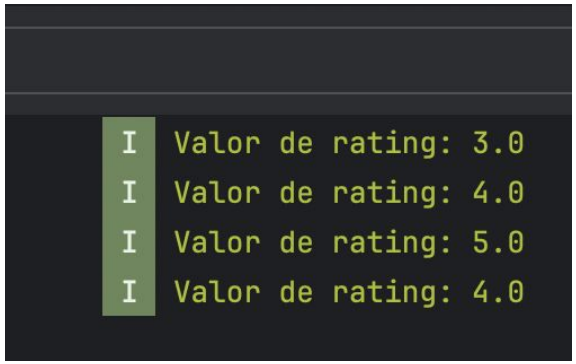



```
<SeekBar
    android:id="@+id/miSeekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:layout_editor_absoluteX="0dp"
    tools:layout_editor_absoluteY="80dp"
    android:max="10"
    android:progress="0"
    tools:ignore="MissingConstraints" />
```

```
SeekBar miControl = (SeekBar) findViewById(R.id.miSeekBar);
miControl.setOnSeekBarChangeListener( new SeekBar.OnSeekBarChangeListener()
{
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {
        Log.i("Bar","Valor progreso:" + seekBar.getProgress());
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        Log.i("Bar","Valor final de:" + seekBar.getProgress());
    }
});
```

U4.1 Componentes: RatingBar

Diseñado para dar una puntuación, su implementación es bastante sencilla:




```
<RatingBar
    android:id="@+id/myRating"
    android:layout_width="250dp"
    android:layout_height="50dp"
    android:numStars="5"
    android:rating="1.0"
    android:stepSize="1.0"
    tools:ignore="MissingConstraints"
    tools:layout_editor_absoluteX="0dp"
    tools:layout_editor_absoluteY="174dp" />
```

```
RatingBar controlRating =  (RatingBar) findViewById(R.id. myRating);
controlRating.setOnRatingBarChangeListener( new
RatingBar.OnRatingBarChangeListener() {
    @Override
    public void onRatingChanged(RatingBar ratingBar,  float rating, boolean
fromUser) {
        Log.i("Rating","Valor de rating: "+rating);
    }
});
```

Tema 5

Lista de componentes a desarrollar en Android:

TextView
Animaciones
Button
ToggleButton
ImageButton
EditText
AutocompleteTextView
Spinner
CheckBox
RadioButton
Switch
SeekBar
RatingBar
ProgressBar
…

U4.2 Listados y Menús:

Tema 6

+ Listados
+ Menús

U4.2 Listados

Tema 6 (Listados):

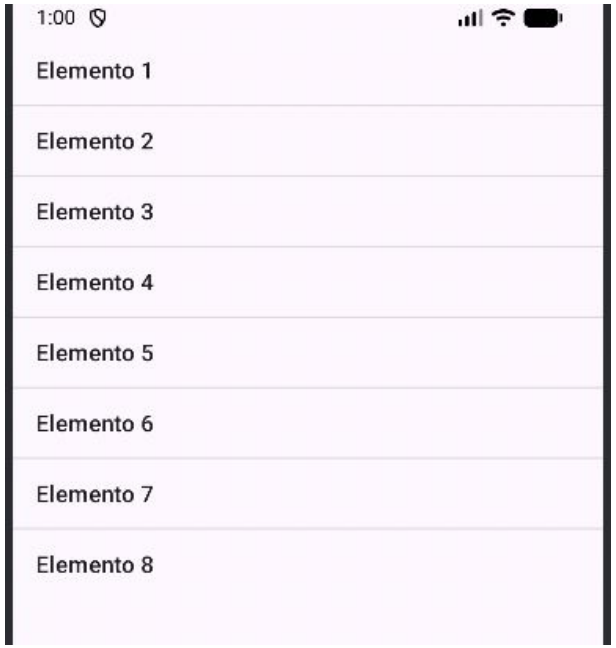
ListView

GridView

Spinner

Adaptadores

U4.2 Listados: List View




```
I  Elemento pulsado: 0
I  Elemento pulsado: Elemento 1
I  Elemento pulsado: 4
I  Elemento pulsado: Elemento 5
I  Elemento pulsado: 7
I  Elemento pulsado: Elemento 8
```

Este elemento permite la visualización de una lista deslizable verticalmente.

```
<ListView
    android:id="@+id/miLista"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:ignore="MissingConstraints" />
```

```
ListView listado = (ListView) findViewById(R.id. miLista);
final String[] datos = new String[]{"Elemento 1", "Elemento 2", "Elemento
3", "Elemento 4", "Elemento 5", "Elemento 6","Elemento 7","Elemento 8"};
ArrayAdapter<String> adaptador = new ArrayAdapter<String>( this,
android.R.layout. simple list item 1,datos);
listado.setAdapter(adaptador);
```

Para obtener información del elemento pulsado:

```
listado.setOnItemClickListener( new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,  int position,
long id) {
        Log.i("Pulsado", "Elemento pulsado: " +position);
Log.i("Pulsado", "Elemento pulsado: " +(String)
parent.getItemAtPosition(position));
    }
});
```

U4.2 Listados: Grid View


```
I  Elemento pulsado: 0
I  Elemento pulsado: Elemento 1
I  Elemento pulsado: 4
I  Elemento pulsado: Elemento 5
I  Elemento pulsado: 6
I  Elemento pulsado: Elemento 7
I  Elemento pulsado: 7
I  Elemento pulsado: Elemento 8
```

Muestra los datos en modo rejilla

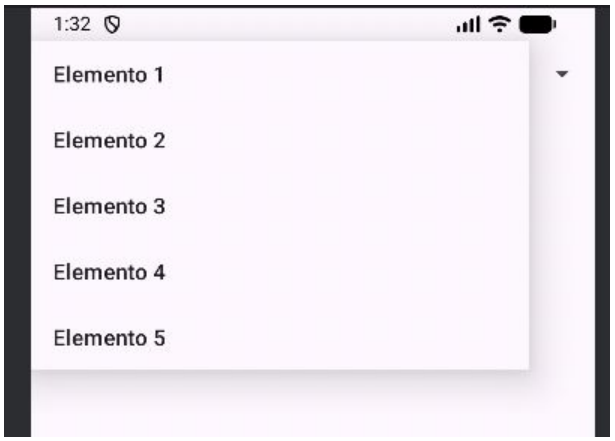
```
<GridView
    android:id="@+id/miGrid"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:numColumns="3"
    tools:ignore="MissingConstraints" />
```

```
GridView listado = (GridView) findViewById(R.id. miGrid);
final String[] datos = new String[]{"Elemento 1", "Elemento 2", "Elemento
3", "Elemento 4", "Elemento 5", "Elemento 6","Elemento 7","Elemento 8"};
ArrayAdapter<String> adaptador = new ArrayAdapter<String>(this,
android.R.layout. simple_list_item_1,datos);
listado.setAdapter(adaptador);
```

Para obtener información del elemento pulsado:

```
listado.setOnItemClickListener( new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,  int position,
long id) {
        Log.i("Pulsado", "Elemento pulsado: " +position);
Log.i("Pulsado", "Elemento pulsado: " +(String)
parent.getItemAtPosition(position));
    }
});
```

U4.2 Listados: Spinner




```
I  Elemento pulsado: 0
I  Elemento pulsado: Elemento 1
I  Elemento pulsado: 2
I  Elemento pulsado: Elemento 3
I  Elemento pulsado: 3
I  Elemento pulsado: Elemento 4
I  Elemento pulsado: 4
I  Elemento pulsado: Elemento 5
```

Spinner muestra una caja configurable con los valores:

```
<Spinner
    android:id="@+id/miSpinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:ignore="MissingConstraints" />
```

```
Spinner listaSpinner = (Spinner) findViewById(R.id. miSpinner);
final String[] datosSpinner = new String[]{"Elemento 1", "Elemento 2",
"Elemento 3", "Elemento 4", "Elemento 5"};
ArrayAdapter<String> adaptadorSpinner = new ArrayAdapter<String>( this,
android.R.layout. simple list item 1, datosSpinner);
listaSpinner.setAdapter(adaptadorSpinner);
```

Para obtener información del elemento pulsado:

```
listaSpinner.setOnItemSelectedListener( new
AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
        Log.i("Pulsado", "Elemento pulsado: " +position);
        Log.i("Pulsado", "Elemento pulsado: " +(String)
parent.getItemAtPosition(position));
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }
});
```

U4.2 Listados: Trabajando con Adaptadores

Anteriormente, hemos trabajado con simple_list_item_1 como adaptador de un elemento, pero podemos trabajar con adaptadores más complejos.

Para crear estos adaptadores complejos, seguiremos los siguientes pasos:

1. Crearemos la clase Datos que dotará de información al adaptador

```
public class Datos {
    private String texto1;
    private String texto2;
    public Datos (String texto1,
String texto2){
        texto1 = texto1;
        texto2 = texto2;
    }
    public String getTexto1(){
        return texto1;
    }
    public String getTexto2(){
        return texto2;
    }
}
```

2. Ahora crearemos la vista con el diseño del elemento: elemento.xml en res/layout.

```
<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/miTexto1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:textSize="18sp" />
        <TextView
            android:id="@+id/miTexto2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textStyle="normal"
            android:textSize="12sp" />
    </LinearLayout>
```

U4.2 Listados: Trabajando con Adaptadores

3. Procedemos a crear el adaptador, que podrá ser de dos tipos: ArrayAdapter y BaseAdapter:

3.1 El ArrayAdapter es el más sencillo ya que convierte a un ArrayList de objetos en vistas. Para ello crearemos una clase independiente que extiende de ArrayAdapter:

```
public class Adaptador extends ArrayAdapter<Datos>
{
   private Datos[] datos;
   public Adaptador(Context context, Datos[]
datos){
       super(context, R.layout.elemento, datos);
       this.datos = datos;
   }
   public View getView(int position, View
convertView, ViewGroup parent){
       LayoutInflater mostrado =
LayoutInflater.from(getContext());
       View elemento =
mostrado.inflate(R.layout.elemento, parent, false);
       TextView texto1 = (TextView)
elemento.findViewById(R.id.miTexto1);
       texto1.setText(datos[position].getTexto1());
       TextView texto2 = (TextView)
elemento.findViewById(R.id.miTexto2);
       texto1.setText(datos[position].getTexto2());
       return elemento;
   }
}
```

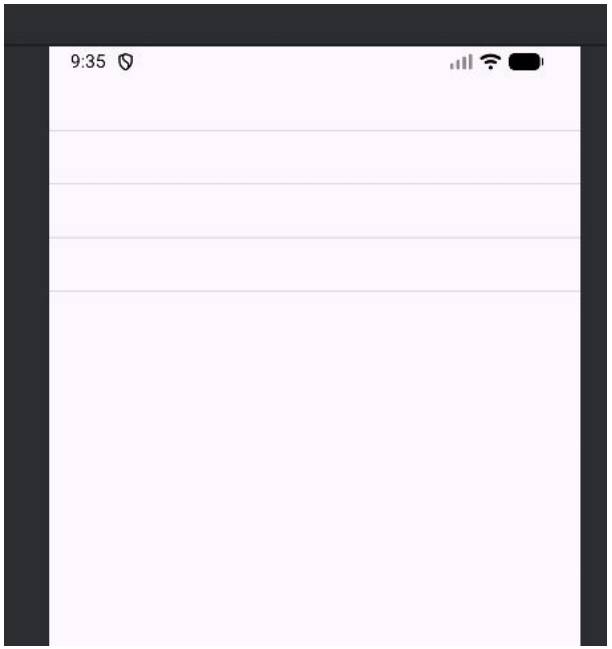
Ya solo quedaría la llamada desde el Java Principal:

```
Datos[] datos = new Datos[]{
       new Datos("Linea superior 1", "Linea inferior 1"),
       new Datos("Linea superior 2", "Linea inferior 2"),
       new Datos("Linea superior 3", "Linea inferior 3"),
       new Datos("Linea superior 4", "Linea inferior 4"),
       new Datos("Linea superior 5", "Linea inferior 5")
};
ListView listado = (ListView) findViewById(R.id.miLista);
Adaptador miAdaptador = new Adaptador(this, datos);
listado.setAdapter(miAdaptador);
```

U4.2 Listados:
Trabajando con Adaptadores

Ejercicios:

¿Por qué sale de este modo el ListView?



¿Cómo podemos hacer para cuando pulsemos en un elemento obtener toda la información?

¿Qué otras mejoras podríamos añadir?

U4.2 Listados:
Trabajando con Adaptadores

3.2 BaseAdapter. Su proceso de creación es similar al ArrayAdapter sólo que la clase que se extiende será BaseAdapter y se dispone de los siguientes métodos:

- int getCount() - N. total de elementos
- Object getItem(int position) - Elemento en la posición
- long getItemId(int position) - ID del elemento en la posición
- View getView(int position, View convertView, ViewGroup parent) - Se usa para mostrar un elemento.

U4.2 Listados: Trabajando con Adaptadores

Código de la Clase Adaptador

```
public class Adaptador extends BaseAdapter {
    private ArrayList<Datos> datos;
    private Context contexto;
    public Adaptador(Context contexto, ArrayList<Datos> datos){
        super();
        this.contexto = contexto;
        this.datos = datos;
    }
    @Override
    public int getCount() {
        return datos.size();
    }
    @Override
    public Object getItem(int position) {
        return datos.get(position);
    }
    @Override
    public long getItemId(int position) {
        return position;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater mostrado = LayoutInflater. from(contexto);
        View elemento = mostrado.inflate(R.layout. elemento, parent, false);
        TextView texto1 = (TextView) elemento.findViewById(R.id. miTexto1);
        texto1.setText( datos.get(position).getTexto1());
        TextView texto2 = (TextView) elemento.findViewById(R.id. miTexto2);
        texto2.setText( datos.get(position).getTexto2());
        return elemento;
    }
}
```

U4.2 Listados: Trabajando con Adaptadores

Código de la Clase Main

```
ArrayList<Datos> datos =  new ArrayList<Datos>();
datos.add(new Datos("Linea superior 1", "Linea inferior 1"));
datos.add(new Datos("Linea superior 2", "Linea inferior 2"));
datos.add(new Datos("Linea superior 3", "Linea inferior 3"));

ListView listado = (ListView) findViewById(R.id. miLista);
Adaptador miAdaptador =  new Adaptador(this, datos);
listado.setAdapter(miAdaptador);
```

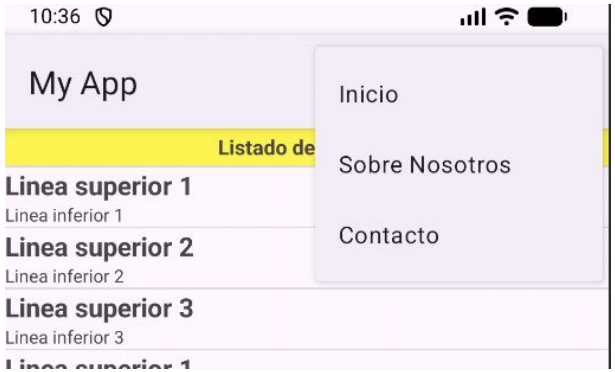
Tema 6 (Menús):

+OptionsMenu

+SubMenu

+Menú Contextual

U4.2 Menús: OptionsMenu I



Se trata del menú más simple y clásico. Creamos un archivo res/menu/mainmenu.xml

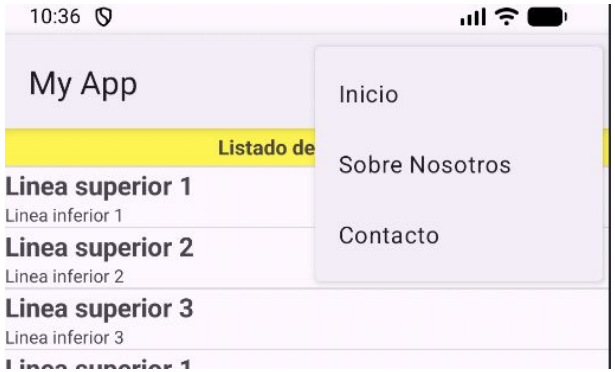
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
   <item android:id="@+id/home"
        android:title="Inicio" />
   <item android:id="@+id/about"
       android:title="Sobre Nosotros" />
   <item android:id="@+id/contact"
       android:title="Contacto" />
</menu>
```

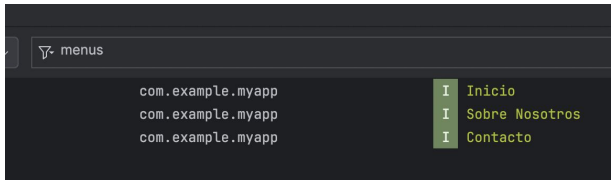
Ahora “inflamos” nuestra app añadiendo el menú y sobrescribiendo este método fuera en el ActivityMain:

```
@Override
public boolean onCreateOptionsMenu (Menu menu) {
   getMenuInflater().inflate(R.menu. mainmenu, menu);
   return true;
}
```

¡OJO! no aparece por defecto, por la configuración del tema de nuestra aplicación, para ello nos vamos a values/themes/themes.xml y sustituimos la opción Theme.Material3.DayNight.NoActionBar por Theme.Material3.DayNight

U4.2 Menús: OptionsMenu II





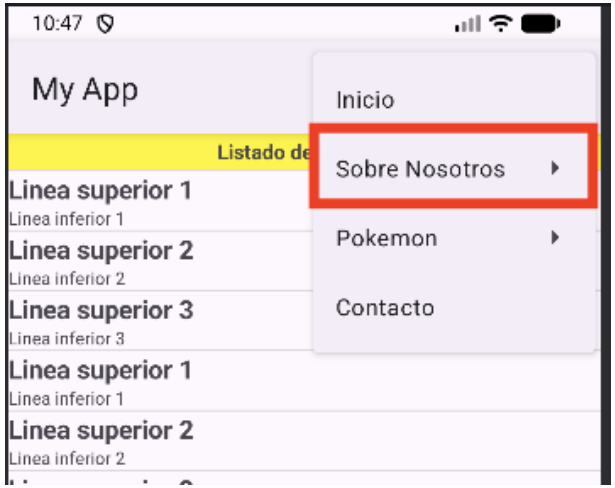
Ahora que ya aparece nuestro menú, sólo quedaría capturar los clicks:

Para ello, igualmente sobreescribimos las funcionalidades por defecto del método:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    Log.i("menus",item.toString());
    return super.onOptionsItemSelected(item);
}
```

Justo debajo del anterior.

U4.2 Menús: Submenu

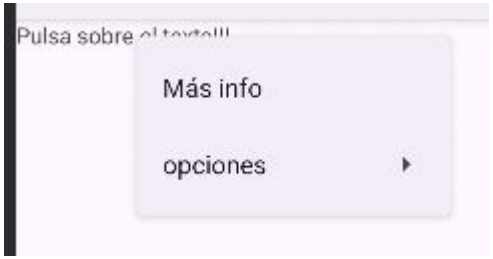


¿Cómo agregamos profundidad a nuestro menú?

Para ello, sólo tenemos que añadir un nuevo menú dentro del item correspondiente:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/home"
          android:title="Inicio"/>
    <item android:id="@+id/about"
        android:title="Sobre Nosotros">
    <menu>
        <item android:id="@+id/empresa"
            android:title="Empresa" />
        <item android:id="@+id/emprendimiento"
            android:title="Emprendimiento" />
    </menu>
    </item>
    <item android:id="@+id/pokemon"
        android:title="Pokemon">
        <menu>
            <item android:id="@+id/listado pokemon"
                android:title="Lisado de Pokemon" />
            <item android:id="@+id/crear pokemon"
                android:title="Crear Pokemon" />
        </menu>
    </item>
    <item android:id="@+id/contact"
        android:title="Contacto"/>
</menu>
```

U4.2 Menús:
Contextual Menú I



¿Podemos añadir Menú a un elemento determinado?
Comenzamos limpiando nuestra vista principal y añadiendo un texto simple:

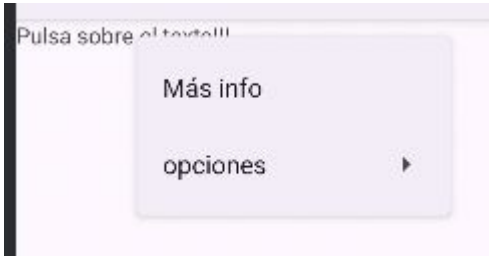
```
<TextView
    android:id="@+id/texto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Pulsa sobre el texto!!!"
    tools:layout_editor_absoluteX="179dp"
    tools:layout_editor_absoluteY="305dp"
    tools:ignore="MissingConstraints" />
```

Nos creamos el nuevo menuelemento.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/info"
        android:title="Más info"/>
    <item android:id="@+id/opciones"
        android:title="opciones">
        <menu>
            <item android:id="@+id/editar"
                android:title="Editar" />
            <item android:id="@+id/eliminar"
                android:title="Eliminar" />
        </menu>
    </item>

</menu>
```

U4.2 Menús:
Contextual Menú II



1º Siguiendo los pasos del [manual de Android], registramos la acción contextual de ese elemento:

```
TextView elemento = (TextView) findViewById(R.id. texto);
registerForContextMenu(elemento);
```

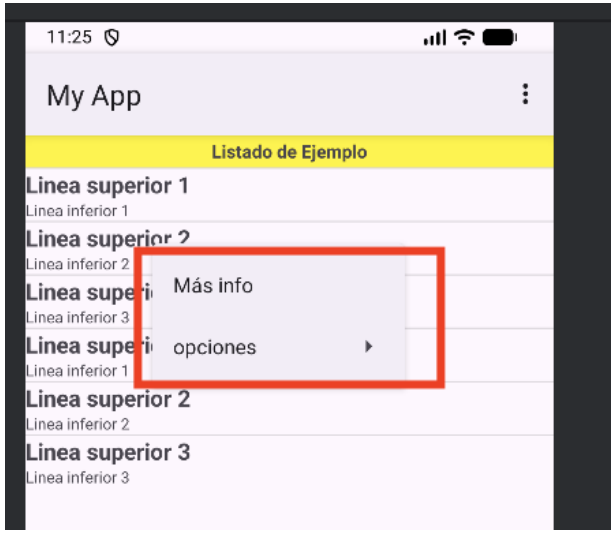
2º Reescribimos el método `onCreateContextMenu`

```
@Override
public void onCreateContextMenu (ContextMenu menu, View v,
                                 ContextMenu.ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu. menuelemento, menu);
}
```

3º Capturamos las opciones

```
@Override
public boolean onContextItemSelected (MenuItem item) {
   AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
   Log.i("menus",item.toString());
            return super.onContextItemSelected(item);
   }
```

U4.2 Menús:
Contextual Menú III (listas)



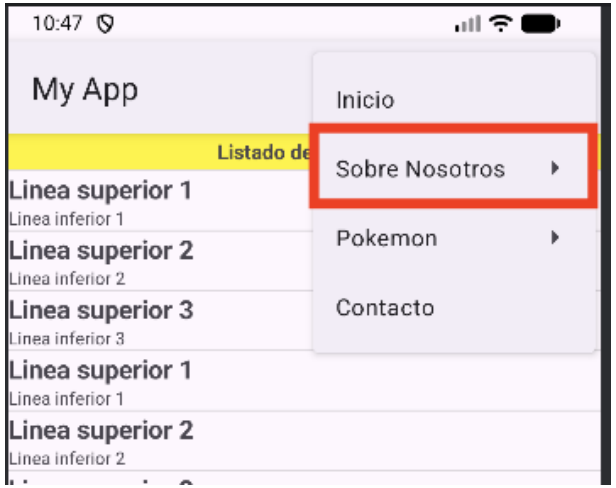
¿Podemos añadir Menú contextual a un elemento de una lista?

Para ello recuperamos el ejemplo de ArrayList<Datos> y registramos listado:

```
listado.setAdapter(miAdaptador);
registerForContextMenu(listado);
```

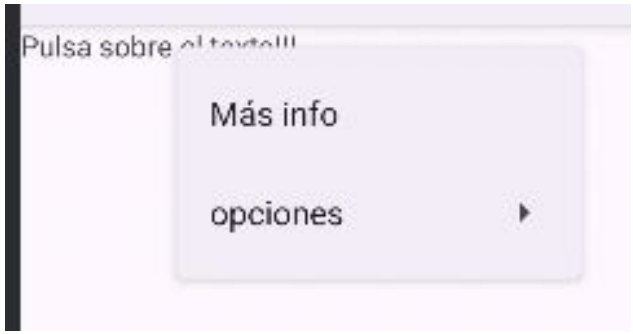
Toda la lógica se mantiene ¿Por qué?

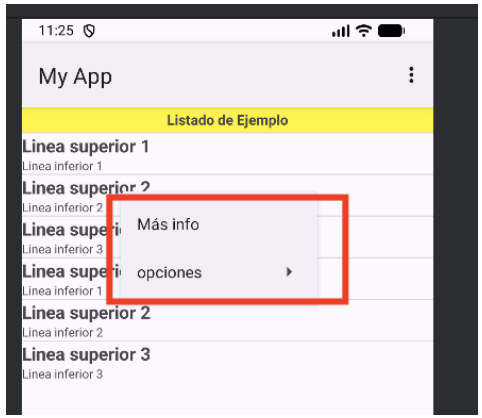
U4.2 Menús: Ejercicios



Ejercicios:

1. ¿Cuál serían los menú de tu aplicación dividido por roles?
2. Añádelo a la entrega que estamos desarrollando y empieza a implementar las vistas básicas.
3. ¿Qué menús contextuales tendrá tu aplicación?
4. ¿Y si quisiéramos crear diferentes menú para diferentes elementos de la lista?: Por ejemplo para el elemento 1 el menuelemento y para el resto de elementos el menurestoelementos





AE U4.1-2 Revisión de Componentes, Menú y Mensajes

Para poder comprobar el avance en el módulo, deberás entregar un documento PDF con la siguiente estructura:

1. **Portada**
2. **Índice**
3. **Introducción**
4. **Desarrollo (Con los pantallazos y explicación de cada elemento que revisemos en clase).**
5. **Conclusión**
6. **Webgrafía, Bibliografía e IA**

Valoraré hasta 5 puntos la elaboración del documento con todo los desarrollos realizados, hasta 2 puntos la estructura, legibilidad, maquetación, diseño y originalidad del mismo y hasta 3 puntos de creatividad adicional, utilidad del documento y otros aspectos adicionales a la propuesta.

Los RA y CE que aplican esta actividad son:

2.b) Se han analizado y utilizado las clases que modelan ventanas, menús, alertas y controles para el desarrollo de aplicaciones gráficas sencillas.
2.d) Se han desarrollado aplicaciones que hacen uso de las funcionalidades proporcionadas por los sensores.
2.f) Se han utilizado las clases necesarias para establecer conexiones con almacenes de datos garantizando la persistencia.
2.i) Se han documentado los procesos necesarios para el desarrollo de las aplicaciones
3.a) Se han analizado entornos de desarrollo multimedia

Resumen U4.

U4.1 Componentes

TextView
Animaciones
Button
ToggleButton
ImageButton
EditText
AutocompleteTextView
Spinner
CheckBox
RadioButton
Switch
SeekBar
RatingBar
ProgressBar
…

U4.2 Listados y Menús

+ListView
+GridView
+Spinner
+Adaptadores
+OptionsMenu
+SubMenu
+Menú Contextual