PROGRAMACIÓN EN MÓVILES

LABORATORIO N° 07

Uso de Menus y RecyclerView en Android

Alumno(s):	*Carlos Chullunquia Téllez*	***Nota***	
Grupo:		***Ciclo: III***	

Criterio de Evaluación	*Excelente (4pts)*	*Bueno (3pts)*	*Requiere mejora (2pts)*	*No acept. (0pts)*	*Puntaje Logrado*
Identifica y configura correctamente menus principales y contextuales en Android					*3*
Identifica el uso del componente RecyclerView en Android					*3*
Desarrolla adecuadamente los ejercicios propuestos					*6*
Realiza observaciones y conclusiones que aporten un opinión crítica y técnica					*3*
Es puntual y redacta el informe adecuadamente sin copias de otros autores					*2*
Evidencia avance en laboratorio					*3*

Laboratorio 7: Uso de Menus y RecyclerView en Android

Alumno: _____ **Nota:** _____

Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:

- Implementar sistemas de menu en una aplicación Android y crear objetos Intent para su respectiva interaccion.

Seguridad:

- Ubicar maletines y/o mochilas en el gabinete al final de aula de Laboratorio o en los casilleros asignados al estudiante.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

Equipos y Materiales:

- Una computadora con:
  - Windows 7 o superior
  - VMware Workstation 10+ o VMware Player 7+
  - Conexión a la red del laboratorio
- Máquinas virtuales:
  - Máquina Virtual con el software Android Studio instalado
- Dispositivo android:
  - 1 Teléfono o Tableta con sistema operativo Android y cable USB para conectar a la PC

Procedimiento:

Lab Setup

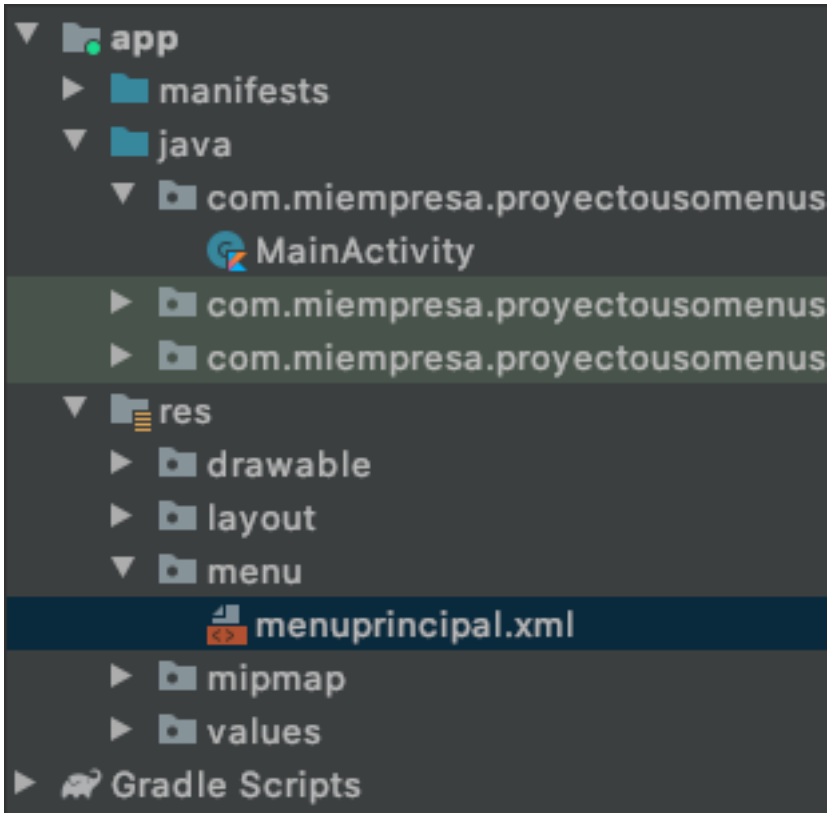
1. Creacion de Proyecto

1.1. Crear un proyecto nuevo en Android Studio
- 1.1.1. Plataforma: **Phone and Tablet**
- 1.1.2. Seleccione **Empty Activity** y clic en **Next**
- 1.1.3. En la siguiente pantalla configuraremos nuestro proyecto. Configure como se indica:
  - 1.1.3.1. **Name: ProyectoUsoMenusV4**
  - 1.1.3.2. **Package name:** com.miempresa.proyectousomenusv4
  - 1.1.3.3. **Save location:** dejar la ruta sugerida
  - 1.1.3.4. **Language:** kotlin
  - 1.1.3.5. **Minimun SDK:** API 21 Android 5.0(Lollipop)

2. Creacion de Menus

2.1. Haga clic derecho sobre la carpeta general del proyecto **APP→new→android resource file**
- 2.1.1.File name: **menuprincipal**
- 2.1.2.Resource type: **Menu**

2.2. En la estructura del proyecto debe mostrarle la carpeta **menu**



2.3. Dirjase al archivo **menuprincipal.xml** y modifique el codigo tal como se muestra en la imagen

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity">
    <item android:id="@+id/configuracion"
        android:title="Configuracion"
        android:icon="@android:drawable/ic_menu_preferences"
        app:showAsAction="never" />
    <item android:id="@+id/informacion"
        android:title="Informacion"
        android:icon="@android:drawable/ic_menu_info_details"
        app:showAsAction="ifRoom"
        android:orderInCategory="2"/>
    <item android:id="@+id/otros"
        android:title="otros"
        android:icon="@android:drawable/ic_menu_day"
        app:showAsAction="always"
        android:orderInCategory="1"/>
</menu>
```

2.4. Explique la utilidad de las propiedades: showAsAction, orderInCategory

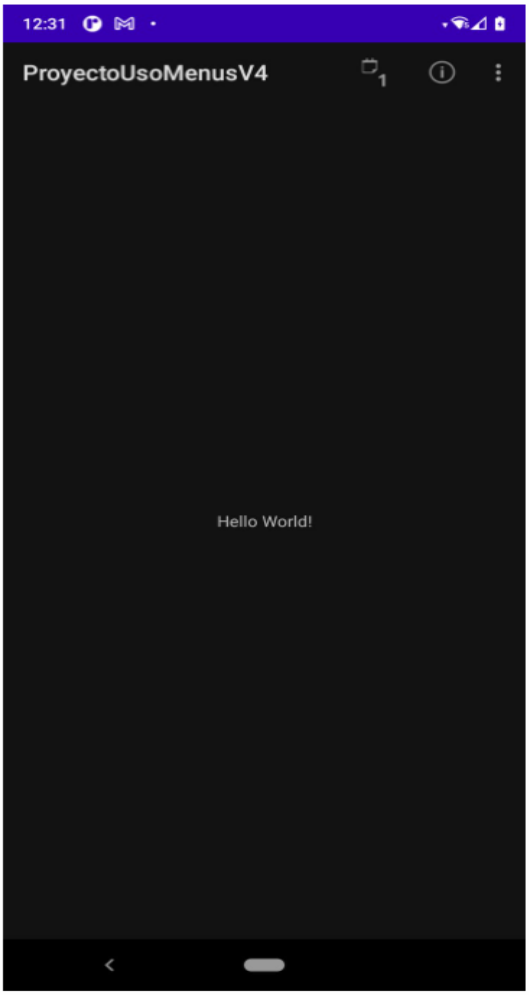
ShowAsAction: permite establecer cuando y donde veremos nuestros botones de acción. Las opciones más usadas son ifRoom y never. La primera indica que si existe espacio disponible para nuestro ítem, entonces se visualizará. La segunda indica que nunca debe aparecer en la Action Bar, solamente en el despliegue de botones de poco uso

OrderInCategory: Es un valor entero que establece la preponderancia que tiene un ítem con respecto a otro. Lo que quiere decir que un ítem con categoría 1 es más prioritario que uno en categoría 2.

2.5. Habra el archivo **MainActivity.kt, importe la clase import kotlinx.android.synthetic.main.activity_main.*** e implemente el metodo **onCreateOptionsMenu** y modifique tal como muestra la imagen.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        menuInflater.inflate(R.menu.menuprincipal, menu)
        return true
    }
}
```

2.6. Ejecute la aplicación y verifique que le muestre un menu de opciones



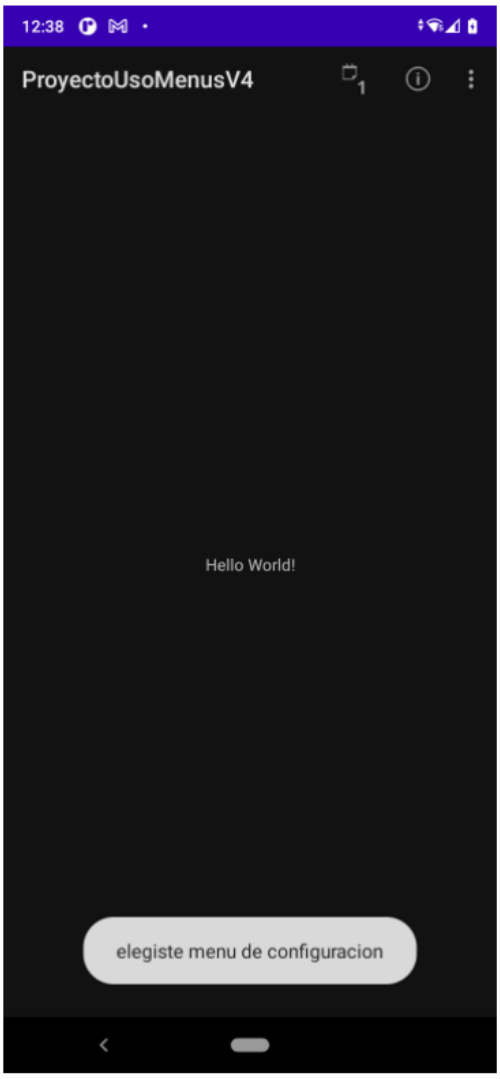
2.7. Fijese que se creo un menu de configuracion pero todavia no se ha programado las acciones a realizar cuando se elija un elemento.

2.8. Procedemos a crear el metodo que nos permitira capturar que elemento del menu se eligio

2.9. Habra el archivo **MainActivity.kt** e implemente el metodo **onOptionsItemSelected** y modifique tal como muestra la imagen(lineas 19 a 30

```
10  class MainActivity : AppCompatActivity() {
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          setContentView(R.layout.activity_main)
14      }
15      override fun onCreateOptionsMenu(menu: Menu?): Boolean {
16          menuInflater.inflate(R.menu.menuprincipal, menu)
17          return true
18      }
19      override fun onOptionsItemSelected(item: MenuItem): Boolean {
20          val id: Int = item.getItemId()
21          if (id == R.id.configuracion) {
22              Toast.makeText( context: this,  text: "Elegiste menu configuracion", Toast.LENGTH_LONG).show();
23              return true
24          }
25          if (id == R.id.informacion) {
26              Toast.makeText( context: this,  text: "Elegiste menu informacion", Toast.LENGTH_LONG).show();
27              return true
28          }
29          return super.onOptionsItemSelected(item)
30      }
31  }
```

2.10. Ejecute el proyecto y verifique el funcionamiento de la aplicación



2.11. Proceda a crear dos actividades: C**onfiguracion** y **Informacion** para que al elegir un elemento del menu creado anteriormente, nos permita aperturar actividades independientemente

2.11.1. Click derecho en carpeta del proyecto principal **APP→new→Activity→Empty Activity**

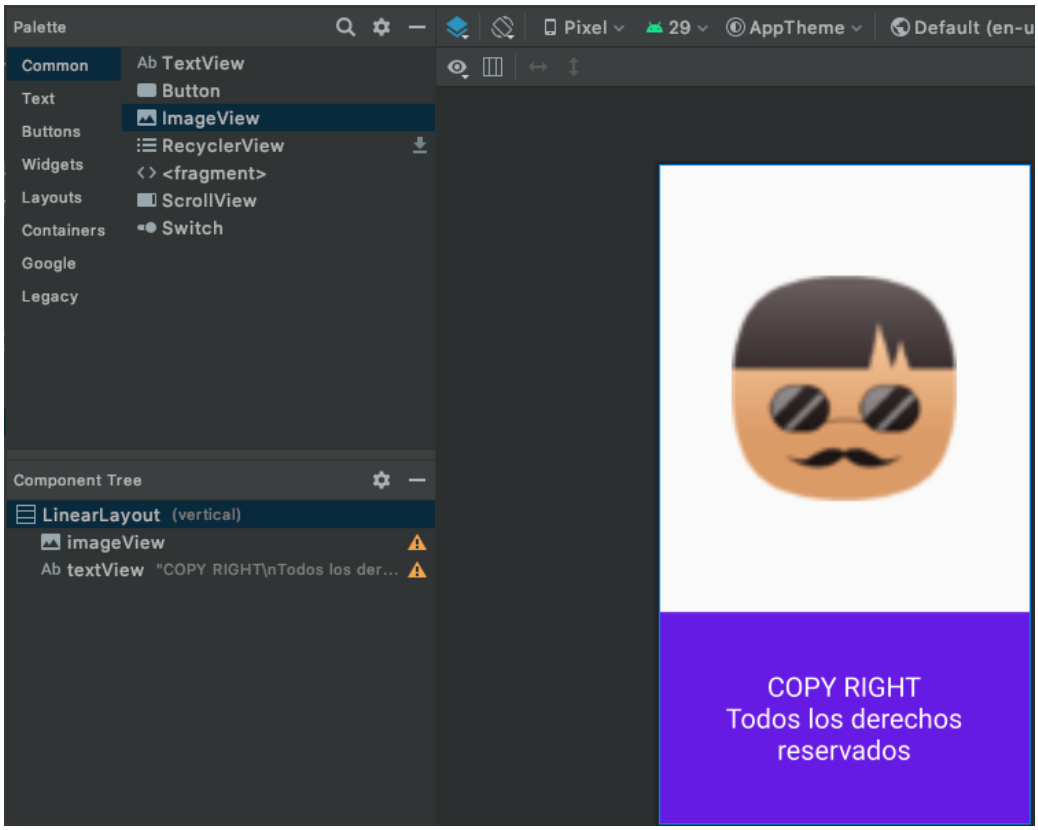
2.11.2. Name Activity: **Configuracion**

2.11.3. Name Activity: **Informacion**

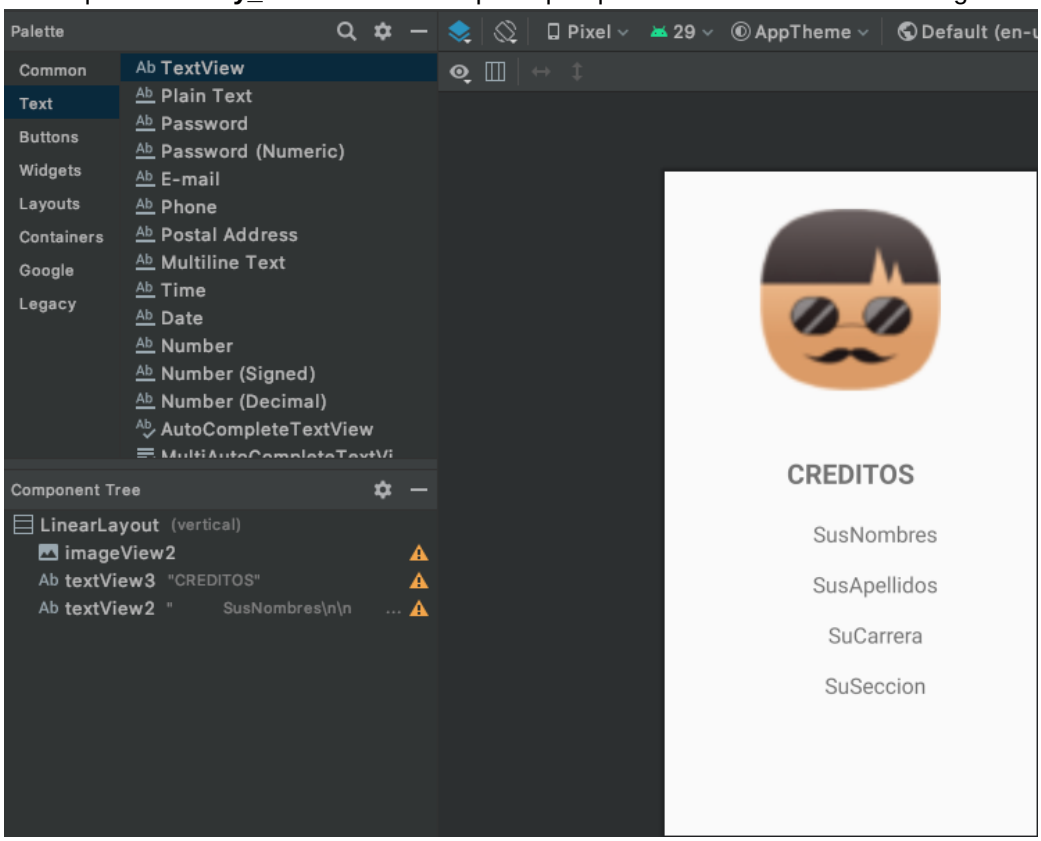
2.11.4. Deje todas las opciones por defecto

2.12. Agrege una imagen al directorio **drawable** llamado i**mgconfiguracion.png** la cual se usara para mostrarla en la actividad a crear

2.13. Modifique la **activity_configuracion.xml** para que quede tal como muestra la imagen



2.14. Modifique la **activity_informacion.xml** para que quede tal como muestra la imagen



2.15. Modifique el archivo **MainActivity.kt** el metodo **onOptionsItemSelected** tal como se muestra(lineas 23,24 y 29,30)

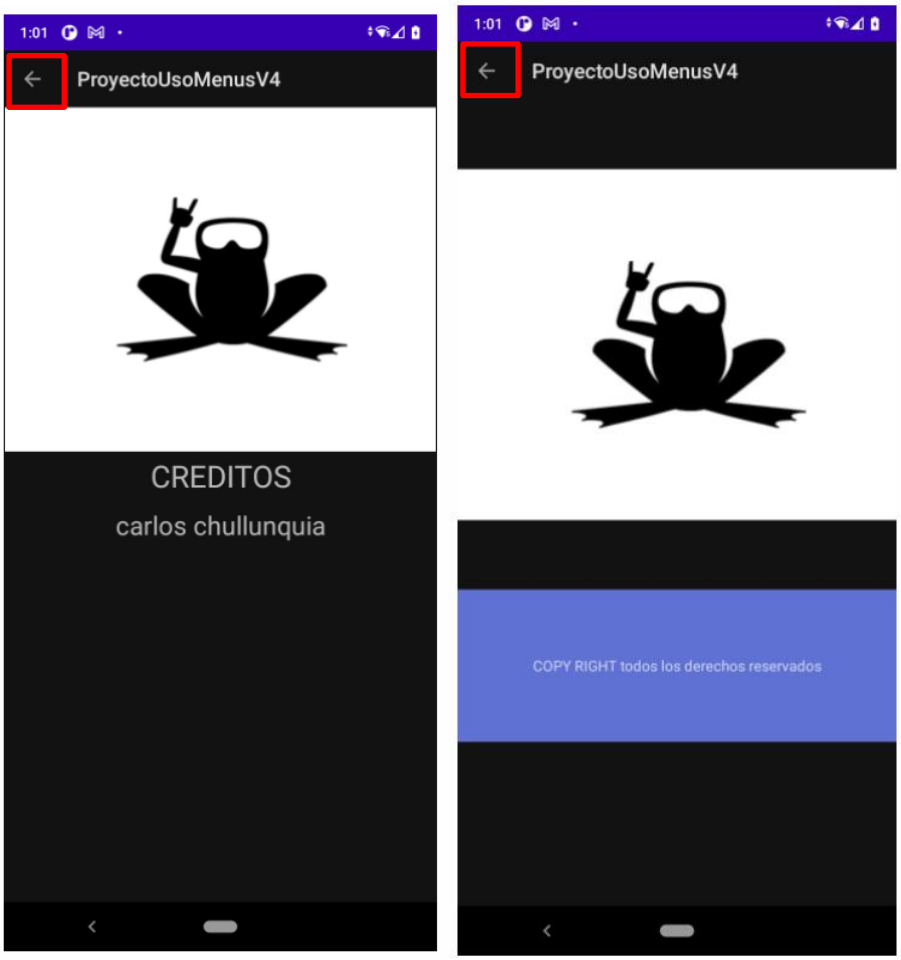
```
19    override fun onOptionsItemSelected(item: MenuItem): Boolean {
20        val id: Int = item.getItemId()
21        if (id == R.id.configuracion) {
22            //Toast.makeText(this, "Elegiste menu configuracion", Toast.LENGTH_LONG).show();
23            val llamaractividad = Intent(applicationContext, Configuracion::class.java)
24            startActivity(llamaractividad)
25            return true
26        }
27        if (id == R.id.informacion) {
28            //Toast.makeText(this, "Elegiste menu informacion", Toast.LENGTH_LONG).show();
29            val llamaractividad = Intent(applicationContext, Informacion::class.java)
30            startActivity(llamaractividad)
31            return true
32        }
33        return super.onOptionsItemSelected(item)
34    }
```

2.16. Abra **AndroidManifest.xml** y se agregar la propiedad **parentActivityName** a la actividad **Informacion** para que permita una vez visualizada la actividad llamada, mostrar una flecha para volver a la actividad padre de donde se las invoco(linea 15)

```
11            android:theme="@style/Theme.ProyectoUsoMenusV4">
12            <activity
13                android:name=".Informacion"
14                android:exported="true"
15                android:parentActivityName=".MainActivity"
16                />
17            <activity
18                android:name=".Configuracion"
```

2.17. Repita el paso anterior para la actividad **Configuracion**

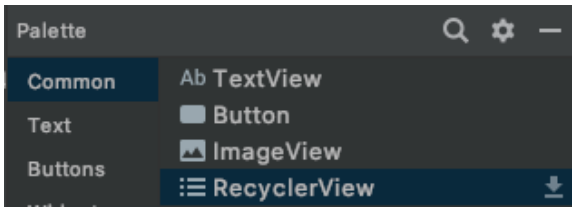
2.18. Ejecute el proyecto y verifique que al seleccionar un elemento del menu este invoca a la actividad previamente configurada.


1:01
ProyectoUsoMenusV4
CREDITOS
carlos chullunquia
1:01
ProyectoUsoMenusV4
COPY RIGHT todos los derechos reservados

3. Uso de RecyclerView

3.1. Utilizaremos el objeto RecyclerView para mostrar listas con imágenes.

3.2. Abra el archivo **activity_main.xml,** en **Palette→common→RecyclerView** haga clic sobre el icono de descarga para instalar el complemento



3.3. En **activity_main.xml** agregue el componente reciclerview y coloque como ID = lista (lineas 10 a 13)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout 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      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      tools:context=".MainActivity" >
9
10     <androidx.recyclerview.widget.RecyclerView
11         android:id="@+id/lista"
12         android:layout_width="match_parent"
13         android:layout_height="match_parent"/>
14
15 </LinearLayout>
```

3.4. En la carpeta **layout** cree un nuevo recurso llamado **elementoslista.xml** con el diseño que determine como se mostraran los elementos en la lista(fijese en los ID colocados al ImageView y TextView)

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_margin="2dp"
5      android:layout_height="wrap_content">
6      <ImageView
7          android:id="@+id/elemento_imagen"
8          android:src="@drawable/ic_launcher_background"
9          android:paddingLeft="40dp"
10         android:layout_width="100dp"
11         android:layout_height="100dp" />
12     <TextView
13         android:id="@+id/elemento_texto"
14         android:layout_width="match_parent"
15         android:layout_centerVertical="true"
16         android:text="ProyectoUsoMenusV4"
17         android:padding="20dp"
18         android:textColor="#000"
19         android:textSize="20sp"
20         android:layout_height="wrap_content" />
21 </LinearLayout>
```

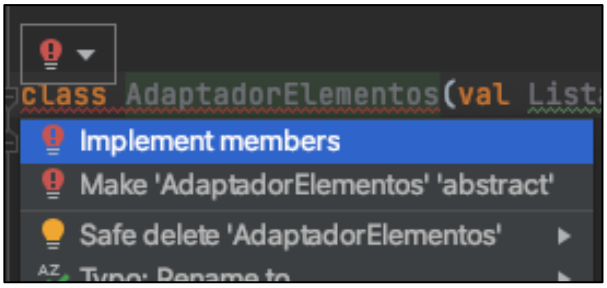
3.5. En el paquete **com.miempresa.proyeousomenusv4** cree una clase kotlin que servira como modelo de del texto e imagen que tendra nuestra lista(**clic derecho→new→kotlin File/class**. Coloque el nombre y despues elija Class) Nombre de clase: **Elementos.kt**

```
1   package com.miempresa.proyectousomenusv4
2
3   import android.graphics.Bitmap
4
5   data class Elementos(val texto:String, val imagen:Bitmap);
```

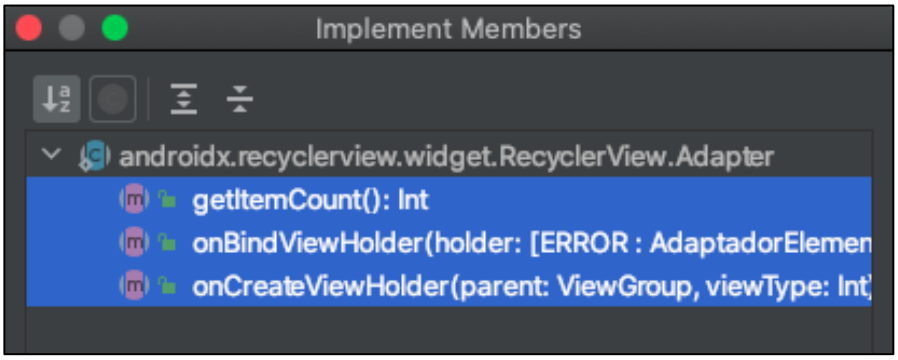
3.6. Dentro del mismo paquete cree otra clase kotlin llamada **AdaptadorElementos.kt** . Esta clase heredara de la clase **recyclerView(**una vez aplicada la herencia solo implemente los metodos sugeridos por el editor de Android Studio**)**

```
class AdaptadorElementos(val ListaElementos:ArrayList<Elementos>): RecyclerView.Adapter<AdaptadorElementos.ViewHolder>() {
```

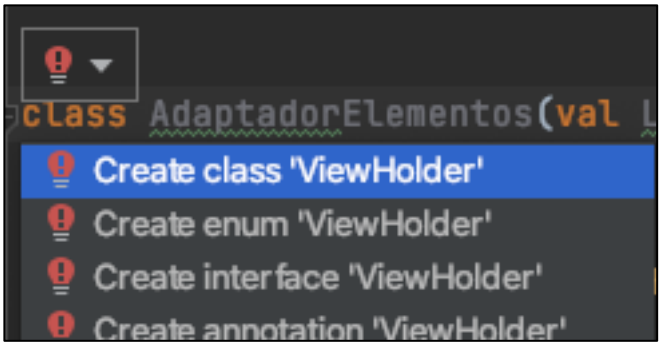
- Clic en el nombre **AdaptadorElementos,** clic en simbolo de foquito rojo y elija I**mplements members**



- Elija los tres métodos sugeridos y clic en OK



- Clic en nombre **ViewHolder**, clic en el foquito rojo y elija **Create class 'ViewHolder'** y si le solicita donde crearlo, determine que se genere dentro de la misma clase **AdaptadorElementos**



- Si hasta aquí sigue marcando errores en algunos nombres, ignore los mensajes, ya que falta aun configurar la clase

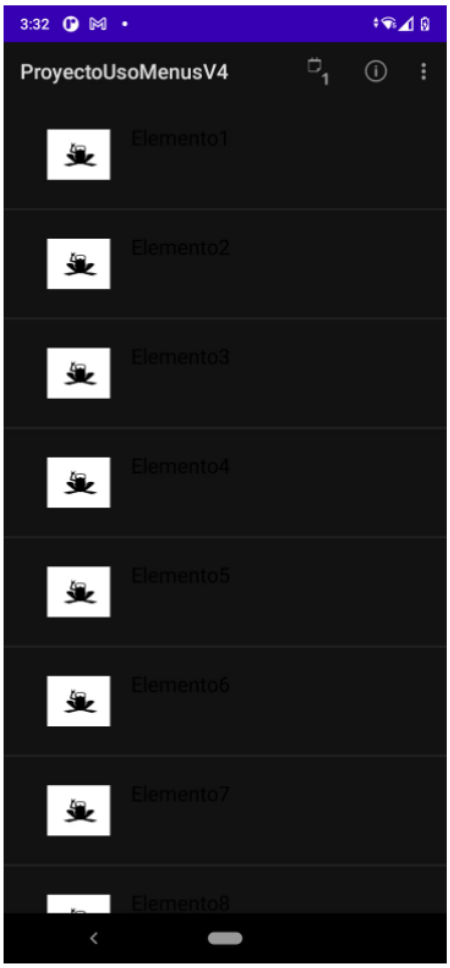
- Modifique el contenido de los métodos generados como se indica en la imagen

```
class AdaptadorElementos(val ListaElementos:ArrayList<Elementos>): RecyclerView.Adapter<AdaptadorElementos.ViewHolder>() {
    class ViewHolder(itemView: View):RecyclerView.ViewHolder(itemView) {
        val fTexto = itemView.findViewById<TextView>(R.id.elemento_texto)
        val fImagen = itemView.findViewById<ImageView>(R.id.elemento_imagen);
    }

    override fun onCreateViewHolder(
        parent: ViewGroup,
        viewType: Int
    ): AdaptadorElementos.ViewHolder {
        val v = LayoutInflater.from(parent?.context).inflate(R.layout.elementoslista, parent,  attachToRoot: false);
        return ViewHolder(v);
    }

    override fun onBindViewHolder(holder: AdaptadorElementos.ViewHolder, position: Int) {
        holder?.fTexto?.text=ListaElementos[position].texto
        holder?.fImagen?.setImageBitmap(ListaElementos[position].imagen)
    }

    override fun getItemCount(): Int {
        return ListaElementos.size;
    }
}
```

3.7. Ahora dirijase a **MainActivity.kt** y se procedera a rellenar la lista de manera estatica por el momento. Se creara un bucle que rellene elementos en la lista. Agrege en el metodo **onCreate**(lineas 23 a 33)

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        lista.addItemDecoration(DividerItemDecoration( context: this, DividerItemDecoration.VERTICAL))
        lista.layoutManager = LinearLayoutManager( context: this);

        var llenarLista = ArrayList<Elementos>()
        for (i in 1 until 20) {
            llenarLista.add(Elementos( texto: "Elemento "+i,
                BitmapFactory.decodeResource(resources, R.drawable.imgconfiguracion)))
        }

        val adapter = AdaptadorElementos(llenarLista)
        lista.adapter = adapter
    }
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
```

3.8. Ejecute el aplicativo y verifique los cambios realizados



4. Menus Contextuales

4.1. Cree un nuevo menu denominado: **menucontextual.xml** que tenga los siguientes elementos:

4.1.1. Ver Nombres
4.1.2. Ver Carrera
4.1.3. Ver Semestre
4.1.4. Ver Email

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".MainActivity">
    <item android:id="@+id/nombre"
        android:title="Ver Nombre"
        android:icon="@android:drawable/ic_menu_preferences"
        app:showAsAction="ifRoom" />
    <item android:id="@+id/carrera"
        android:title="Ver Carrera"
        android:icon="@android:drawable/ic_menu_info_details"
        app:showAsAction="ifRoom"
        />
    <item android:id="@+id/semestre"
        android:title="Ver Semestre"
        android:icon="@android:drawable/ic_menu_day"
        app:showAsAction="ifRoom"
        />
    <item android:id="@+id/email"
        android:title="Ver Email"
        android:icon="@android:drawable/ic_dialog_email"
        app:showAsAction="ifRoom"
        />
</menu>
```

4.2. Modifique el archivo **MainActivity.kt** para llenar con valores la lista insertada y para permitir mostrar el menu contextual(**registerForContextMenu**) cuando se utilice la lista. Modifique tal como se muestra al final del metodo **onCreate** (linea 34)

```
31              val adapter = AdaptadorElementos(llenarLista)
32              lista.adapter = adapter
33
34              registerForContextMenu(lista)
35          }
```

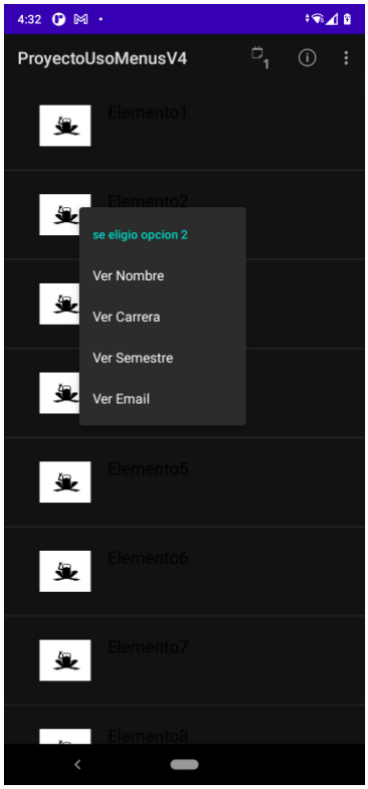
4.3. Creamos el metodo que permita invocar al menu contextual. Despues de la llave de cierre del metodo onCreate, agrege el metodo **onCreateContextMenu**(lineas 36 a 44)

```
36      override fun onCreateContextMenu(
37              menu: ContextMenu?,
38              v: View?,
39              menuInfo: ContextMenu.ContextMenuInfo?
40          ) {
41              super.onCreateContextMenu(menu, v, menuInfo)
42              val inflater:MenuInflater = menuInflater
43              inflater.inflate(R.menu.menucontextual,menu)
44          }
45      override fun onCreateOptionsMenu(menu: Menu?): Boolean {
```

4.4. En la clase **AdaptadorElementos.kt** agregaremos el soporte de recyclerView para controlar acciones de clic largos. En el metodo **onBindViewHolder** agrege(lineas 25 a 28)

```
21      override fun onBindViewHolder(holder: ViewHolder, position: Int) {
22          holder?.fTexto?.text=ListaElementos[position].texto
23          holder?.fImagen?.setImageBitmap(ListaElementos[position].imagen)
24
25          holder.itemView.setOnCreateContextMenuListener { contextMenu, _, _ ->
26              contextMenu.setHeaderTitle("Se eligio opcion " + (position+1))
27              true
28          }
29      }
```

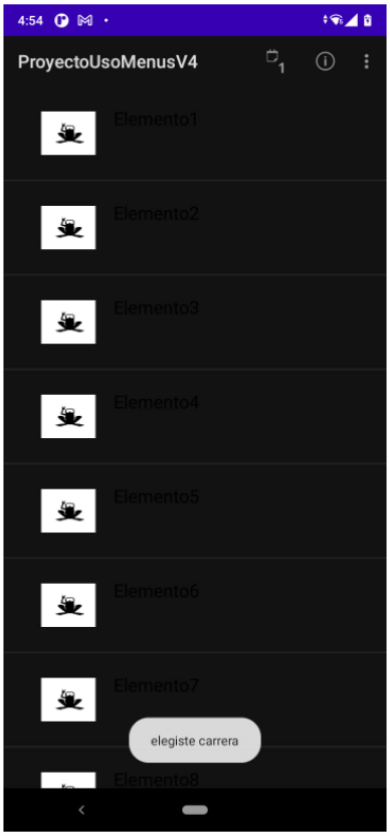
4.5. Ejecute y verifique el funcionamiento de la aplicación. Mantenga presionado un elemento de la lista



4.6. En **MainACtivity.kt** creamos el metodo(**onContextItemSelected**) que permita determinar que elemento del **menucontextua.xmll** se eligio

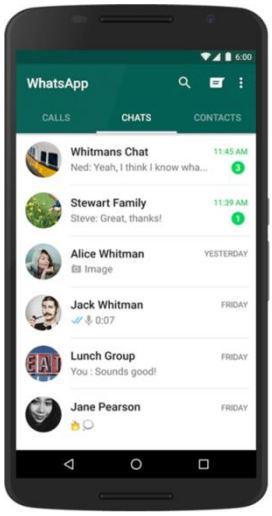
```
override fun onContextItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.nombre -> {
            Toast.makeText( context: this,  text: "Elejiste nombre", Toast.LENGTH_LONG).show()
        }
        R.id.semestre -> {
            Toast.makeText( context: this,  text: "Elejiste semestre", Toast.LENGTH_LONG).show()
        }
        R.id.email -> {
            Toast.makeText( context: this,  text: "Elejiste email", Toast.LENGTH_LONG).show()
        }
        R.id.carrera -> {
            Toast.makeText( context: this,  text: "Elejiste carrera", Toast.LENGTH_LONG).show()
        }
    }
    return true
}
```

4.7. Ejecute y verifique el funcionamiento de la aplicación

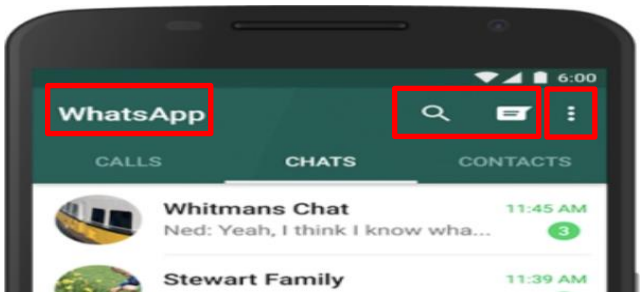

4:54
ProyectoUsoMenusV4
Elemento1
Elemento2
Elemento3
Elemento4
Elemento5
Elemento6
Elemento7
elegiste carrera
Elemento8

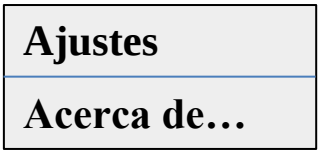
Tarea:

1. Cree un nuevo proyecto denominado **DemoWhatsApp**
2. Aplocando lo aprendido, desarrolle una aplicación que simule la lista de CHATS de WhatApss(no debe tener funcionalidad de chats, solo implemente el diseño)



3. Que muestre tal como se ve en la imagen El titulo de la aplicación **WhatApp** , listar iconos de dos actividades , y que al hacer clic en los 3 puntos muestre la opcion de **Ajustes** y **Acerca de..**





4. Investigue la librería https://github.com/hdodenhof/CircleImageView para mostrar iconos redondeados

OBSERVACIONES (5 mínimo):
(Las observaciones son las notas aclaratorias, objeciones y problemas que se pudo presentar en el desarrollo del laboratorio)

CONCLUSIONES (5 mínimo):

(Las conclusiones son una opinión personal sobre tu trabajo, explicar como resolviste las dudas o problemas presentados en el laboratorio. Ademas de aportar una opinión critica de lo realizado)