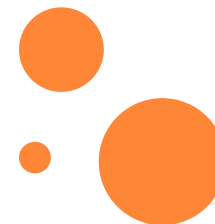


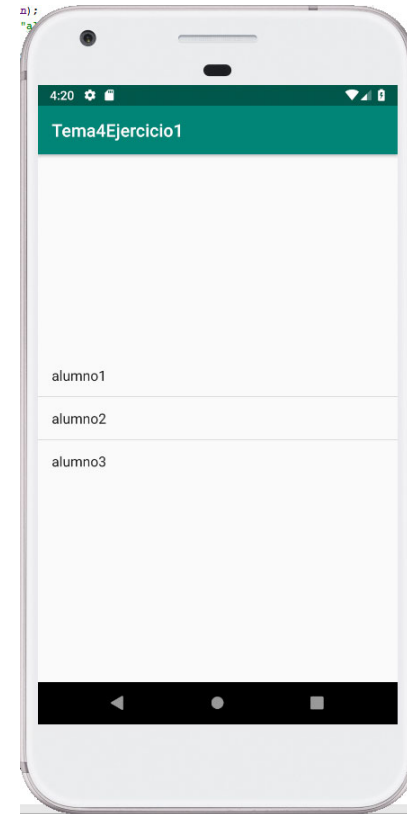
LISTAS



LISTAS

○ ListView

- En Android, los *ListView*s son uno de los *views* más usados, porque permite organizar elementos en una lista vertical.
- Hay dos tipos de *ListView*s:
 - Básico: cada *ListView* está compuesto de ítems simples. Los elementos se introducen en el *ListView* mediante un “adaptador” (*ArrayAdapter*).
 - Personalizado: hay que crear una vista para los ítems, y una clase para un “adaptador” personalizado.



LISTAS

○ ListView

- ListView básico

- Hay que añadir un elemento de tipo ListView al fichero xml del layout (está en la categoría Legacy)
- Generar un adaptador e indicarle qué datos debe mostrar (en forma de array) y cómo debe mostrarlos (aspecto)

```
String[] arraydedatos={"alumno1","alumno2","alumno3"};  
ArrayAdapter eladaptador =  
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,arraydedatos);  
ListView lalista = (ListView) findViewById(R.id.lista);  
lalista.setAdapter(eladaptador);
```

Identificador del
ListView en el layout

Aspecto a usar para los
elementos

Contenido



LISTAS

○ ListView

- ListView básico
 - Se puede añadir un listener para indicar qué hacer cuando se seleccione algún elemento del ListView

```
lalista.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        Log.i("etiqueta", ((TextView)view).getText().toString()+" "+position+" "+id);  
    }  
});
```

Contenido del
elemento pulsado

Posición del elemento
pulsado (empieza en 0)

Identificador del elemento
pulsado. En ArrayAdapter es
igual a la posición



LISTAS

○ ListView

- Layout de los ítems
 - Android proporciona básicamente 2 layouts para dar formato a los ítems:
 - *simple_list_item_1* para elementos de una única línea

```
ArrayAdapter eladaptador =  
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,arraydedatos);
```

| |
|---------|
| alumno1 |
| alumno2 |
| alumno3 |
| |



LISTAS

○ ListView

- Layout de los ítems

- Android proporciona básicamente 2 layouts para dar formato a los ítems:
 - *simple_list_item_2* para elementos de dos líneas

Hay que indicarle uno de los textview que contiene (text1 o text2) y sobrescribir el método getView

```
final String[] arraydedatos={"alumno1","alumno2","alumno3"};
final Integer[] arrayedades={22,23,24};
ArrayAdapter eladaptador =
    new ArrayAdapter<String>(this, android.R.layout.simple_list_item_2, android.R.id.text1, arraydedatos) {

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View vista= super.getView(position, convertView, parent);
        TextView lineaprincipal=(TextView) vista.findViewById(android.R.id.text1);
        TextView lineasecundaria=(TextView) vista.findViewById(android.R.id.text2);
        lineaprincipal.setText(arraydedatos[position]);
        lineasecundaria.setText(arrayedades[position].toString());
        return vista;
    }
};

ListView lalista = findViewById(R.id.list);
lalista.setAdapter(eladaptador);
```

Indicar el contenido de cada línea

| |
|---------|
| alumno1 |
| 22 |
| alumno2 |
| 23 |
| alumno3 |
| 24 |



LISTAS

○ ListView

- Layout de los ítems

- Al usar un listener para el layout *simple_list_item_2* hay que indicara cuál de las dos etiquetas queremos acceder

```
lalista.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
        Log.d("etiqueta", ((TextView) view.findViewById(android.R.id.text1)).getText().toString());  
    }  
});
```

Queremos acceder al contenido de la etiqueta text1 (la superior)



LISTAS

○ ListView

- Layout de los ítems

- Existen otros layouts

- *simple_list_item_single_choice* muestra un radiobutton en cada elemento.

| | |
|----|----------------------------------|
| 22 | <input checked="" type="radio"/> |
| 21 | <input type="radio"/> |
| 23 | <input type="radio"/> |

- *simple_list_item_multiple_choice* muestra un checkbox en cada elemento

| | |
|----|-------------------------------------|
| 22 | <input type="checkbox"/> |
| 21 | <input type="checkbox"/> |
| 23 | <input checked="" type="checkbox"/> |

- *simple_list_item_checked* muestra un tick en cada elemento

| | |
|----|---|
| 22 | ✓ |
| 21 | ✓ |
| 23 | ✓ |



LISTAS

○ ListView

- Layout de los ítems
 - Existen otros layouts
 - En todos ellos se debe añadir el atributo choiceMode al elemento ListView del xml en el layout
 - singleChoice: solo se puede marcar uno de los elementos
 - multipleChoice: se pueden marcar varios de los elementos

```
<ListView  
...  
    android:choiceMode="singleChoice"/>
```



LISTAS

○ ListView

- Layout de los ítems
 - El ListView tiene operaciones que permiten acceder a las posiciones seleccionadas, a los identificadores, etc.

Un array con true si está seleccionada esa posición y false en caso contrario

```
String texto="Los elegidos son: ";
SparseBooleanArray elegidos= lalista.getCheckedItemPositions();
for(int i=0;i<elegidos.size();i++){
    if(elegidos.valueAt(i)==true){
        String s = ((TextView) lalista.getChildAt(i)).getText().toString();
        texto = texto+s+" ";
    }
}
Log.d("etiqueta", texto);
```



LISTAS

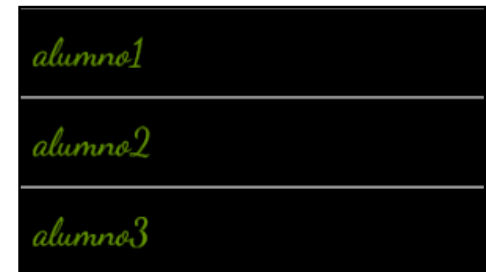
○ ListView:

- ListView básico

- También podemos crear nuestro propio layout con un elemento TextView y personalizarlo en cuanto a tamaño, color, etc.
- Hay que crear un fichero xml en la carpeta *layout* con el aspecto que queramos dar a nuestra etiqueta

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    android:textSize="28dp"
    android:fontFamily="cursive"
    android:background="@android:color/black"
    android:textColor="@android:color/holo_green_dark"/>
```

Sólo puede contener
un TextView



```
ArrayAdapter eladaptador=new ArrayAdapter (this,R.layout.fila,arraydedatos);
```

Nombre del fichero xml



LISTAS

- **Ejercicio 1:** Cread una actividad con un ListView y que cada vez que pulséis un elemento os aparezca en el log un mensaje indicando qué elemento habéis pulsado.



LISTAS

○ ListView

- ListView personalizado
 - Para poder mostrar en cada elemento de la lista todo lo que se quiera.
 - Hay que crear un layout con el aspecto deseado para cada fila del ListView

```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
...>

<ImageView
    android:id="@+id/imagen"
.../>

<TextView
    android:id="@+id/etiqueta"
... />

<RatingBar
    android:id="@+id/barra"
.../>

</android.support.constraint.ConstraintLayout>
```

Una imagen

Un texto

Una barra de puntuaciones



LISTAS

○ ListView

- ListView personalizado

- Hay que crear una nueva clase que extienda de *BaseAdapter*

```
public class AdaptadorListView extends BaseAdapter {...}
```

- Hay que definir los datos que se quieren mostrar y la constructora correspondiente

```
private Context contexto;  
private LayoutInflater inflater;  
private String[] datos;  
private int[] imagenes;  
private double[] puntuaciones;
```

El contexto y el inflater son necesarios.
El resto son los datos que queremos mostrar

```
public AdaptadorListView(Context pcontext, String[] pdatos, int[] pimágenes, double[] ppuntuaciones)  
{  
    contexto = pcontext;  
    datos = pdatos;  
    imagenes=pimágenes;  
    puntuaciones=ppuntuaciones;  
    inflater = (LayoutInflater) contexto.getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
}
```

Se realiza la asignación y se obtiene el inflater



LISTAS

○ ListView

- ListView personalizado

- Al extender de *BaseAdapter* aparecen una serie de métodos que hay que sobrescribir

```
@Override  
public int getCount() {  
    return datos.length;  
}
```

El número de elementos

```
@Override  
public Object getItem(int i) {  
    return datos[i];  
}
```

El elemento i

```
@Override  
public long getItemId(int i) {  
    return i;  
}
```

El identificador del elemento i



LISTAS

○ ListView

- ListView personalizado

- Al extender de *BaseAdapter* aparecen una serie de métodos que hay que reescribir

```
@Override
public View getView(int i, View view, ViewGroup viewGroup) {
    view=inflater.inflate(R.layout.fila,null);
    TextView nombre= (TextView) view.findViewById(R.id.etiqueta);
    ImageView img=(ImageView) view.findViewById(R.id.imagen);
    RatingBar barra= (RatingBar) view.findViewById(R.id.barra);

    nombre.setText(datos[i]);
    img.setImageResource(imagenes[i]);
    barra.setRating((float)puntuaciones[i]);

    return view;
}
```

Cómo se visualiza un elemento

Se indica el xml con el layout para cada elemento

Se recogen los elementos del layout en variables

Se asigna a cada variable el contenido que se quiere mostrar en ese elemento



LISTAS

○ ListView

- ListView personalizado

- En la actividad que contiene el ListView se genera la instancia del adaptador customizado con los datos que se quieran mostrar y se asigna al ListView de la interfaz

```
protected void onCreate(Bundle savedInstanceState) {  
    int[] personajes={R.drawable.bart,R.drawable.edna,R.drawable.homer,R.drawable.lisa,R.drawable.skinner};  
    String[] nombres={"Bart Simpson","Edna Krabappel","Homer Simpson","Lisa Simpson","Seymour Skinner"};  
    double[] valoracion={3.2,2.4,4.6,4.9,3.0};  
  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    ListView simpsons= (ListView) findViewById(R.id.lista);  
    AdaptadorListView eladap= new AdaptadorListView(getApplicationContext(),nombres,personajes,valoracion);  
    simpsons.setAdapter(eladap);  
}
```

Arrays con la información

ListView en el layout de la actividad

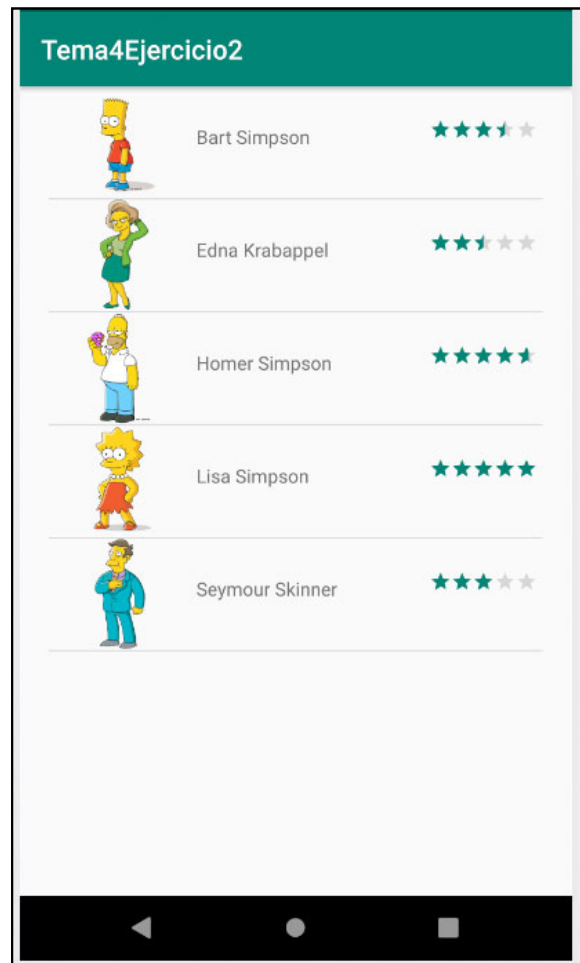
Instancia de la clase adaptador que se ha personalizado

Asignación del adaptador al ListView



LISTAS

- ListView
 - ListView personalizado



LISTAS

○ ListView

- ListView personalizado

- Al definir el listener para el ListView personalizado se recibe como View todo el layout correspondiente a una fila y hay que acceder al elemento que se desee

```
simpsons.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id){  
        Log.d("etiqueta", ((TextView)view.findViewById(R.id.etiqueta)).getText().toString());  
    }  
});
```

El id del TextView en el layout del elemento



LISTAS

○ ListView

- ListView personalizado

- Si en el layout de cada fila hay algún elemento que pueda obtener el focus (como un botón), el listener no funcionará.
- Hay que evitar que los elementos de la fila puedan coger el focus
- En el elemento raíz del layout de cada fila hay que poner el atributo **descendantFocusability** a "blocksDescendants"

```
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
...
    android:descendantFocusability="blocksDescendants"
>

...

</android.support.constraint.ConstraintLayout>
```



LISTAS

○ ListView

- ListView personalizado

- Si en el layout de cada fila hay algún elemento que pueda obtener el focus (como un botón) es probable que queramos definir un listener para él
- El listener para un elemento de la fila se crea en el método getView() del adaptador

```
public View getView(int i, View view, ViewGroup viewGroup) {  
    final int fila=i;  
    ...  
    Button elboton= (Button) view.findViewById(R.id.boton);  
    elboton.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            Log.i("etiqueta", "Se ha pulsado el botón de la fila:"+fila);  
        }  
    });  
    ...  
    return view;  
}
```



LISTAS

- **Ejercicio 2:** Cread una aplicación que muestre un ListView personalizado donde cada elemento tenga una imagen, un texto y un botón.
 - Cada vez que se pulse el ListView se abrirá una actividad con una caja de texto y un botón.
 - En la caja de texto se escribirá el código hexadecimal de un color.
 - Al pulsar el botón se volverá a la actividad del ListView, donde se le pondrá de fondo el color introducido por el usuario.

El ListView

```
simpsons.setBackgroundColor(Color.parseColor("#00ff00"));
```

- Cada vez que se pulse el botón se abrirá una actividad cuyo color de fondo será el mismo que el del ListView y donde se mostrará la imagen correspondiente a esa fila y un botón para volver a la actividad del ListView.
 - Pasad un código y con un if decidid qué imagen mostrar



CARDVIEW

- El CardView es un contenedor de elementos gráficos (Views)
 - Los muestra con el aspecto de una tarjeta



CARDVIEW

- Añadir en el layout el elemento CardView (categoría Containers) y su contenido (los Views que se deseen)

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout ...>

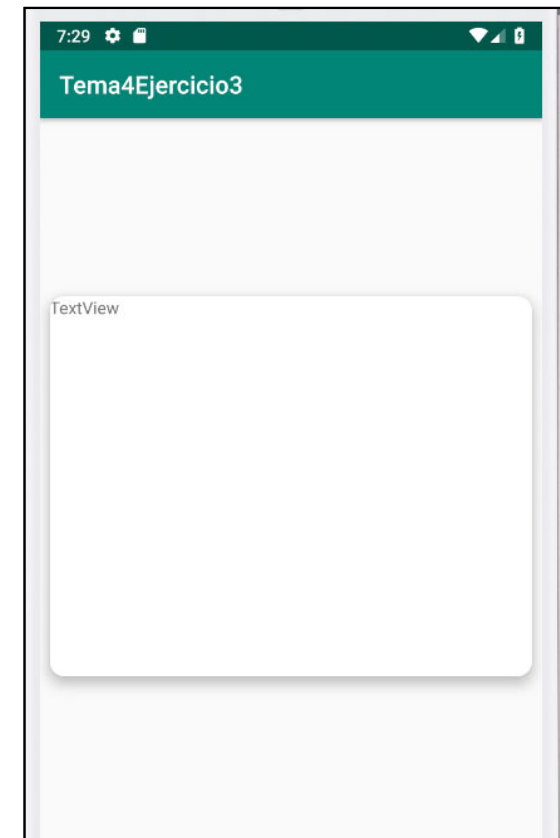
    <androidx.cardview.widget.CardView
        android:layout_width="395dp"
        android:layout_height="312dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:cardElevation="8dp"
        app:cardCornerRadius="12dp">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView" />

    </androidx.cardview.widget.CardView>
</android.support.constraint.ConstraintLayout>
```

Efecto de elevación

Radio de las esquinas



RECYCLERVIEW

- El RecyclerView es “similar” a un ListView pero gestiona mejor la memoria
 - Es recomendable su uso cuando hay que listar más elementos de los que entran en la pantalla del dispositivo
 - Se usa mucho en combinación con CardViews
- Se añade igual que un ListView
 - Se encuentra en la categoría Common del editor.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout ...>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/elreciclerview"
        android:scrollbars="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</android.support.constraint.ConstraintLayout>
```



RECYCLERVIEW

- Hay que definir el layout que tendrá cada elemento de RecyclerView.
 - Por ejemplo, usando CardView

```
<androidx.cardview.widget.CardView
...>
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <ImageView
            android:id="@+id/foto"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@drawable/ic_launcher_background"
            android:scaleType="centerCrop" />
        <TextView
            android:id="@+id/texto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center" />
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```



RECYCLERVIEW

- Hay que definir una clase que extienda a ViewHolder
 - Puede ser una clase externa o interna
 - En el constructor de la clase se hace la asociación entre los campos de la clase y los elementos gráficos del layout

```
public class ElViewHolder extends RecyclerView.ViewHolder {  
    public TextView eltexto;  
    public ImageView laimagen;  
  
    public ElViewHolder (@NonNull View itemView){  
        super(itemView);  
        eltexto=itemView.findViewById(R.id.texto);  
        laimagen=itemView.findViewById(R.id.foto);  
    }  
}
```

Los elementos definidos en el layout para un elemento

Los atributos definidos para la clase



RECYCLERVIEW

- Crear una clase que extienda a la clase genérica RecyclerView.Adapter
 - Se usa la clase que extiende a ViewHolder
 - En el constructor
 - Recibir los datos que se quieren mostrar en la lista
 - Asignarlos a atributos de la clase

```
public class ElAdaptadorRecycler extends RecyclerView.Adapter<ElViewHolder> {  
  
    private String[] losnombres;  
    private int[] lasimagenes;  
  
    public ElAdaptadorRecycler (String[] nombres, int[] imagenes)  
    {  
        losnombres=nombres;  
        lasimagenes=imagenes;  
    }  
  
    ...  
}
```

La información que se quiere mostrar

La clase que extiende ViewHolder



RECYCLERVIEW

- Crear una clase que extienda a RecyclerView.Adapter
 - Hay que sobrescribir el método *onCreateViewHolder*
 - “Infla” el layout definido para cada elemento y crea y devuelve una instancia de la clase que extiende a ViewHolder

```
public class ElAdaptadorRecycler extends RecyclerView.Adapter<ElViewHolder>{  
  
...  
public ElViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
  
    View elLayoutDeCadaItem= LayoutInflater.from(parent.getContext()).inflate(R.layout.item_layout,null);  
    ElViewHolder evh = new ElViewHolder(elLayoutDeCadaItem);  
    return evh;  
}  
...  
}
```

La clase que extiende a
ViewHolder

El xml con el layout para
un elemento



RECYCLERVIEW

- Crear una clase que extienda a RecyclerView.Adapter
 - Hay que sobrescribir los métodos:
 - *onBindViewHolder* que asigna a los atributos del ViewHolder los valores a mostrar para una posición concreta
 - *getItemCount* que devuelve la cantidad de elementos mostrar

```
public class ElAdaptadorRecycler extends RecyclerView.Adapter<ElViewHolder>

...
@Override
public void onBindViewHolder(@NonNull ElViewHolder holder, int position) {
    holder.eltexto.setText(losnombres[position]);
    holder.laimagen.setImageResource(lasimagenes[position]);
}

@Override
public int getItemCount() {
    return losnombres.length;
}
}
```

Asignar a los atributos del ViewHolder la información a mostrar

La cantidad de datos total a mostrar



RECYCLERVIEW

- En la actividad
 - Se recoge el elemento de la interfaz gráfica

```
RecyclerView lalista= findViewById(R.id.elreciclerview);
```

- Se crea el adaptador con los datos a mostrar y se asigna al RecyclerView

```
int[] personajes= {R.drawable.bart, R.drawable.edna, R.drawable.homer, R.drawable.lisa,  
                  R.drawable.skinner};  
String[] nombres={"Bart Simpson", "Edna Krabappel", "Homer Simpson", "Lisa Simpson",  
                 "Seymour Skinner"};  
ElAdaptadorRecycler eladaptador = new ElAdaptadorRecycler(nombres, personajes);  
lalista.setAdapter(eladaptador);
```

Los datos a mostrar

Crear una instancia del adaptador y asignarlo



RECYCLERVIEW

- En la actividad

- Hay que establecer cómo se desea que se organicen los elementos dentro del RecyclerView
 - LinearLayoutManager: los elementos se muestran de forma lineal (vertical u horizontal)
 - GridLayoutManager: Los elementos se muestran en forma de rejilla (todos el mismo tamaño)
 - StaggeredGridLayoutManager: Los elementos se muestran en forma de rejilla (cada elemento puede tener un tamaño distinto)



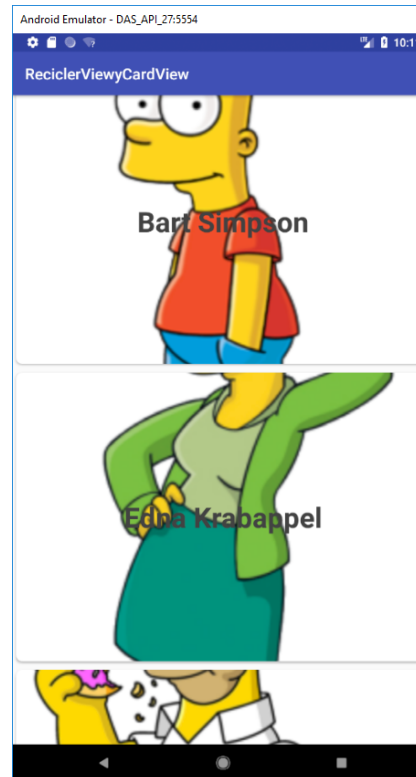
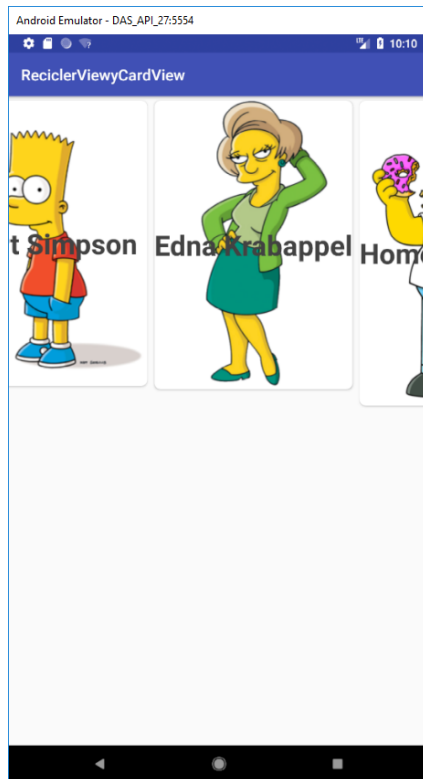
RECYCLERVIEW

- En la actividad
 - LinearLayoutManager

Si los datos se muestran en orden inverso

```
LinearLayoutManager elLayoutLineal= new LinearLayoutManager(this,LinearLayoutManager.HORIZONTAL,false);  
lalista.setLayoutManager(elLayoutLineal);
```

Orientación



RECYCLERVIEW

- En la actividad
 - GridLayoutManager

Número de columnas

Si los datos se muestran en orden inverso

```
GridLayoutManager elLayoutRejillaIgual= new GridLayoutManager(this,2,GridLayoutManager.VERTICAL,false);  
lista.setLayoutManager(elLayoutRejillaIgual);
```

Orientación



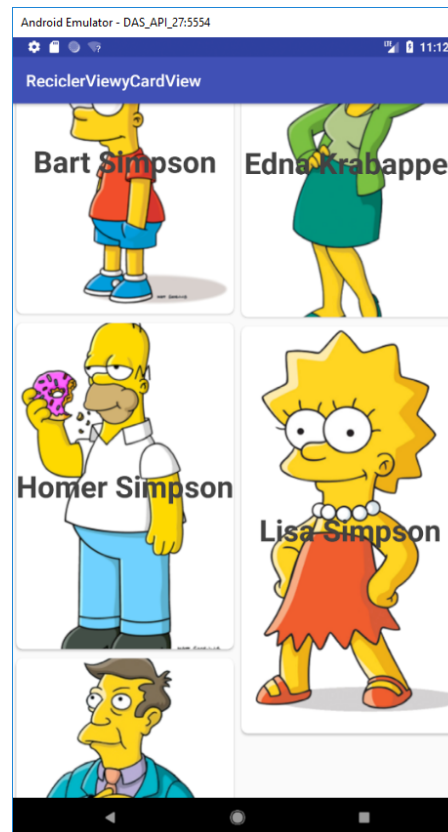
RECYCLERVIEW

- En la actividad
 - StaggeredGridLayoutManager

```
StaggeredGridLayoutManager elLayoutRejillaDesigual =  
    new StaggeredGridLayoutManager(2, GridLayoutManager.VERTICAL);  
lalista.setLayoutManager(elLayoutRejillaDesigual);
```

Número de columnas

Orientación



RECYCLERVIEW

- Para gestionar la interacción del usuario
 - Definir en la clase del adaptador alguna manera de controlar qué elementos del RecyclerView han sido seleccionados

```
public class ElAdaptadorRecycler extends RecyclerView.Adapter <ElViewHolder>{  
  
    private String[] losnombres;  
    private int[] lasimagenes;  
    private boolean[] seleccionados;  
  
    public ElAdaptadorRecycler (String[] nombres, int[] imagenes){  
        losnombres=nombres;  
        lasimagenes=imagenes;  
        seleccionados=new boolean[nombres.length];  
    }  
  
    ...  
}
```

Array de booleanos para indicar qué elementos se han elegido

Inicializar a false (nada elegido) un array de tantas posiciones como elementos se muestran



RECYCLERVIEW

- Para gestionar la interacción del usuario
 - Hay que poder acceder a ese control de la selección desde la clase que extiende ViewHolder
 - Si la clase es interna, se puede acceder a las variables definidas como static

```
public class ElAdaptadorRecycler extends RecyclerView.Adapter <ElViewHolder>{  
  
    private String[] losnombres;  
    private int[] lasimágenes;  
    private static boolean[] seleccionados;  
    ...  
}
```

Si la clase que extiende a ViewHolder es interna



RECYCLERVIEW

- Para gestionar la interacción del usuario
 - Hay que poder acceder a ese control de la selección desde la clase que extiende ViewHolder
 - Si la clase es externa, hay que hacerle llegar el atributo del adaptador que sirve como control

```
public class ElViewHolder extends RecyclerView.ViewHolder {  
    public TextView eltexto;  
    public ImageView laimagen;  
    public boolean[] seleccion;  
    ...  
}
```

Atributo que recogerá el control

```
public class ElAdaptadorRecycler extends RecyclerView.Adapter<ElViewHolder>  
{  
    ...  
    @NonNull  
    @Override  
    public ElViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
        View elLayoutDeCadaItem= LayoutInflater.from(parent.getContext()).inflate(R.layout.item_layout,null);  
        ElViewHolder evh = new ElViewHolder(elLayoutDeCadaItem);  
        evh.seleccion = seleccionados;  
        return evh;  
    }  
    ...  
}
```

"Pasar" el atributo del adaptador al atributo del ViewHolder

RECYCLERVIEW

- Para gestionar la interacción del usuario
 - En la clase que extiende a ViewHolder: definir un listener para cada View y gestionar qué se quiere hacer al seleccionar cada elemento

```
public class ElViewHolder extends RecyclerView.ViewHolder {  
    ...  
    public ElViewHolder (@NonNull View itemView) {  
        super(itemView);  
        eltexto=itemView.findViewById(R.id.texto);  
        laimagen=itemView.findViewById(R.id.foto);  
        itemView.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                if (seleccion[getAdapterPosition()]==true) {  
                    seleccion[getAdapterPosition()]=false;  
                    laimagen.setColorFilter(null);  
                }  
                else {  
                    seleccion[getAdapterPosition()]=true;  
                    laimagen.setColorFilter(Color.BLACK);  
                }  
            }  
        });  
    }  
}
```

El listener

Qué hacer al
marcar/desmarcar
un elemento

La posición del elemento
seleccionado

Resultado



RECYCLERVIEW

○ Resumen

```
nombres={ "Homer", ...}  
...  
findViewById(R.id.rv)  
new ElAdaptadorRecycler()  
setAdapter()  
new XLayoutManager()  
setLayoutManager()
```

Actividad.java

```
class ... extends  
RecyclerView.Adapter  
<EVH>{  
    onCreateViewHolder() {...}  
    onBindViewHolder() {...}  
    getItemCount() {...}  
}
```

ElAdaptadorRecycler.java

```
class EVH extends  
RecyclerView.ViewHolder{  
    TextView texto;  
    public EVH(){  
        texto =  
            findViewById(R.id.txt);  
    }  
}
```

ElViewHolder.java

```
<ConstraintLayout...  
  
<RecyclerView ... id="@+id/rv"
```

Layout_actividad.xml

```
<CardView ...  
    <RelativeLayout ...  
        <ImageView ... id="@+id/img"  
        <TextView ... id="@+id/txt"
```

item_layout.xml



LISTAS

- Ejercicio 3:
- Cread una aplicación que muestre un RecyclerView donde cada elemento sea un CardView
- Cuando el usuario pulse un elemento, su fondo se debe volver azul.
 - Si se vuelve a pulsar el elemento, que el fondo se ponga blanco.

