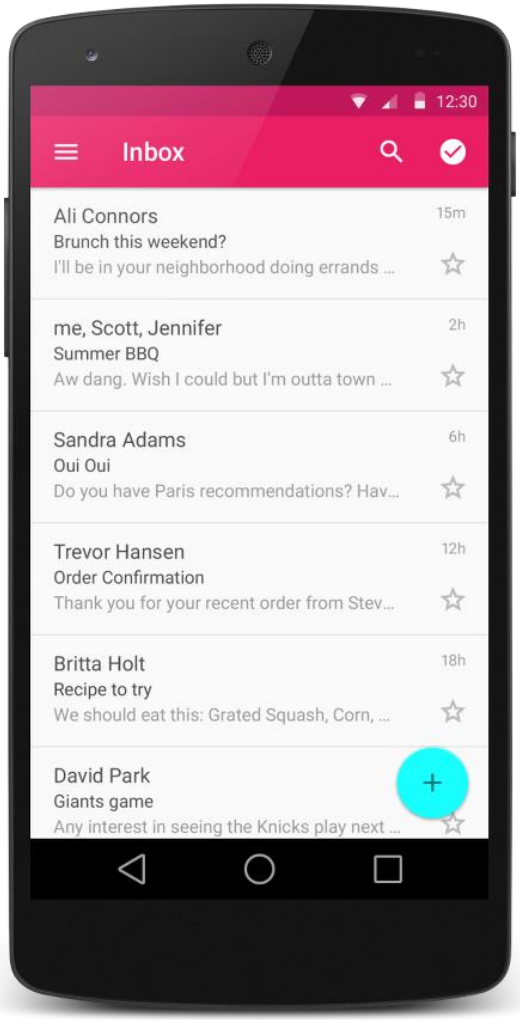
Interfaz de usuario:
Controles de selección (III):
"RecyclerView"

Mª Alejandra Tomás Fernández

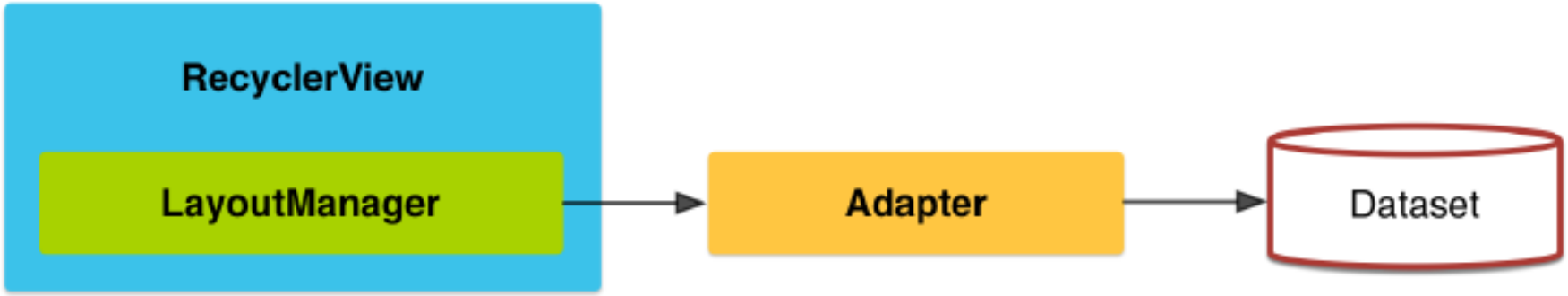
Control RecyclerView

- Permite crear **listas o cuadrículas deslizables.**
- Es una versión más avanzada y potente que sustituye a:
  - ListView
  - GridView
- Incluye entre sus mejoras:
  - Reciclado de vistas (usando ViewHolder)
  - Distribución de vistas configurable
  - Animaciones Automáticas
  - Separador de elementos
  - Trabaja conjuntamente con otros widgets de Material Design



- Aparece en la versión 5.0, pero está en la librería de compatibilidad v7. Por tanto puede usarse desde SDK 7.

Control RecyclerView
Elementos de un RecyclerView

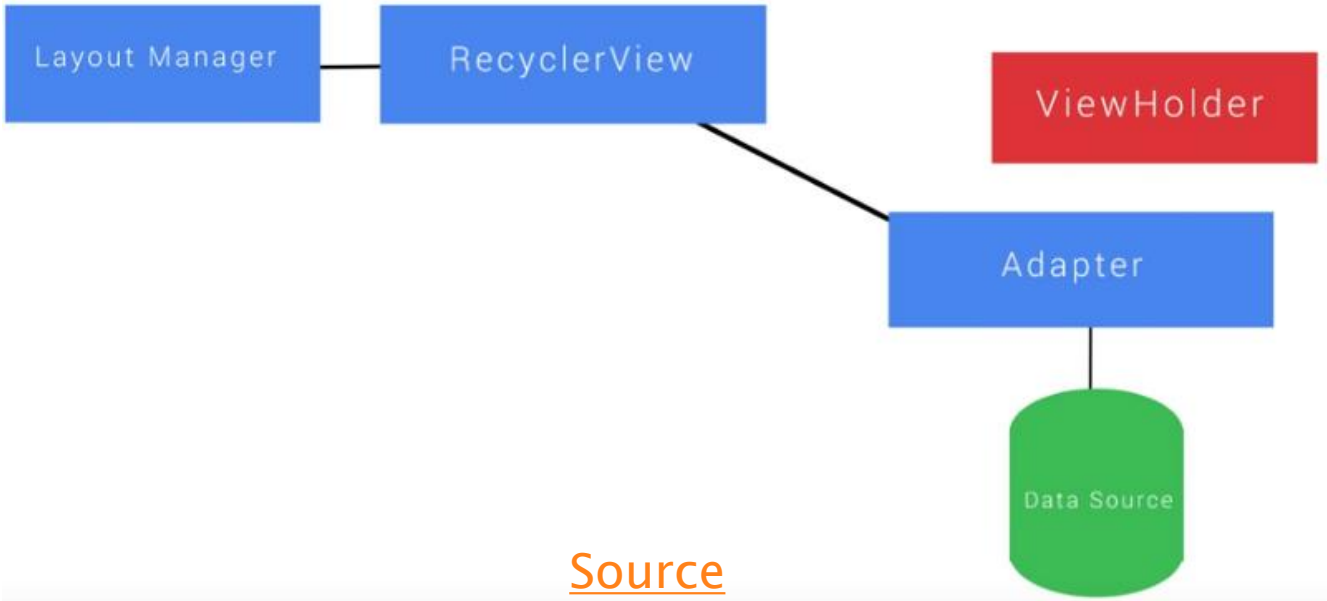


1. RecyclerView.Adapter
2. RecyclerView.LayoutManager
3. RecyclerView.ItemAnimator
4. RecyclerView.ViewHolder

VEREMOS CADA UNO DE ELLOS CON MÁS DETALLE A CONTINUACIÓN

Control RecyclerView
RecyclerView.Adapter

- Su tarea es **coger los datos desde un origen de datos: BD, array, rellenar cada una de las vistas (VIEW)… y pasárselas de una en una al LayoutManager**.
- El LayoutManager tiene que presentar esos datos al usuario.
- **Pasa DATOS DE UN PUNTO A OTRO.**



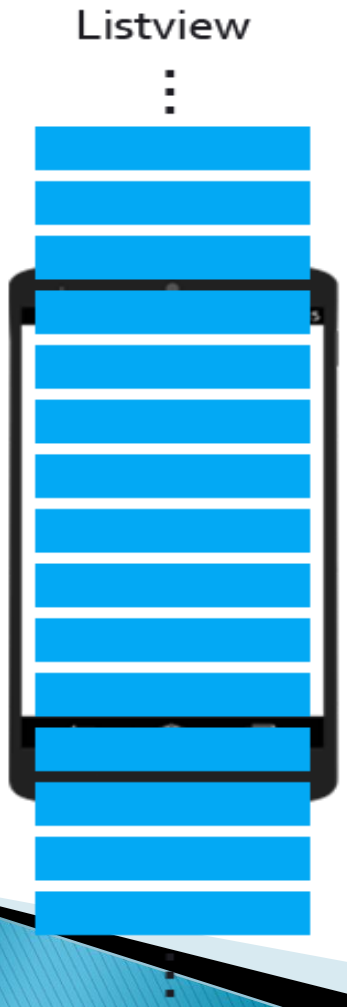
Source

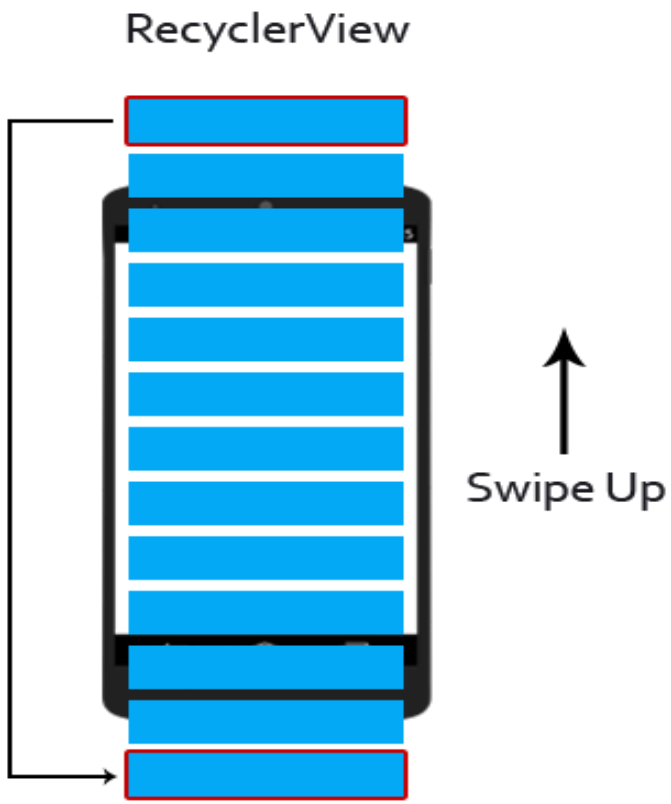
Control RecyclerView
RecyclerView.Adapter

- Pero RecyclerView.Adapter es un **adaptador especial**:

Control RecyclerView
RecyclerView.ViewHolder

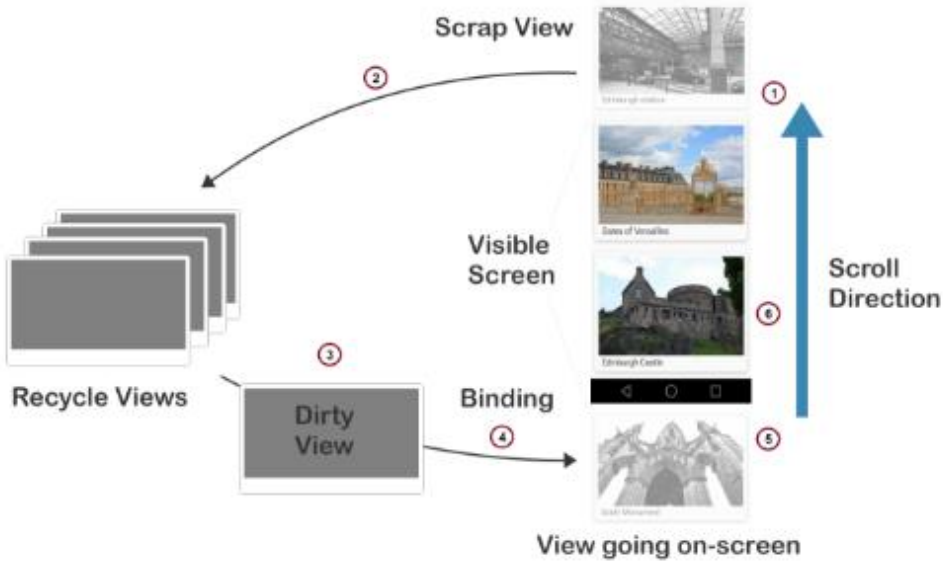
ENLACE: Muy buen ejemplo usando PIZZAS de por qué hay que usar ViewHolder





Control RecyclerView
RecyclerView.ViewHolder

- Cada VIEW está unida a un VIEWHOLDER
  - *Para intentar crear las menos vistas posibles*
- El ViewHolder almacenará la **caché** de un objeto VIEW
  - *Cuando una VIEW se sale de la pantalla al hacer Scroll, se guarda su estado (CACHEA en el ViewHolder) de forma que si el usuario vuelve a hacer scroll hacia abajo, se le muestra la vista automáticamente.*



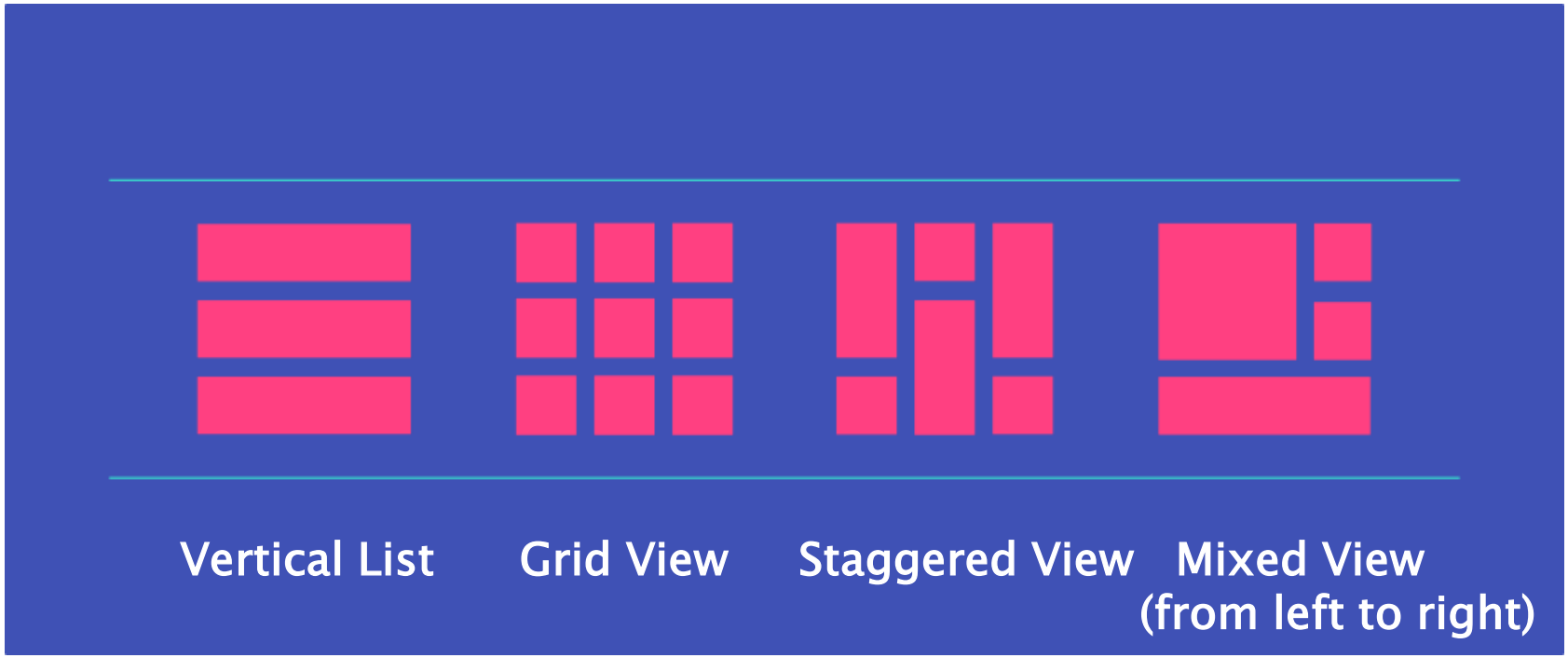
Control RecyclerView
RESUMEN: RecyclerView.Adapter y RecyclerView.ViewHolder

- El **principal trabajo del** **RecyclerView.Adapter** es:

1. Crear el ViewHolder y unirlo a las vistas (VIEWS)
2. Resto de tareas lo hace el VIEWHOLDER.

Control RecyclerView
RecyclerView.LayoutManager

- Ayuda a tomar las VIEWs desde el ADAPTER y **organiza** los datos en ellas en una determinada posición y posiciona **las VIEWs en pantalla de una determianda forma: lista, cuadrícula,…**



Control RecyclerView
Gestión evento onItemClick()

- **no incluye un evento onItemClick()** como ocurría en el caso de ListView.
- RecyclerView delega esta tarea a otro componente:
  - la propia vista que conforma cada elemento de la colección.
  - **Tendremos que asociar a esa vista el código a ejecutar como respuesta al evento click.**
  - **Hay varias formas de hacerlo. Veremos una de las que menos recursos consume (no usa objetos anónimos)**

.

Control RecyclerView
Gestión evento onItemClick()

1. Incluiremos el listener correspondiente como atributo del adaptador y lo asignamos con un setter:

```
public class AdaptadorUsuarios
       extends RecyclerView.Adapter<AdaptadoUsuarios.UsuariosViewHolder>
        {

    //...
    private View.OnClickListener listener;

    //…
   public void setItemClickListener(View.OnClickListener clickListener) {
            this.listener = clickListener;
   }
```

Control RecyclerView
Gestión evento onItemClick()

2. asignar el escuchador a cada una de las vistas creadas:

```
public class AdaptadorUsuarios
        extends RecyclerView.Adapter<AdaptadorUsuarios.UsuariosViewHolder> {

    //...
     @Override
    public UsuariosViewHolder onCreateViewHolder(ViewGroup viewGroup, int
viewType) {
        View itemView = LayoutInflater.from(viewGroup.getContext())
                .inflate(R.layout.listitem_usuario, viewGroup, false);

        itemView.setOnClickListener(listener);
        UsuariosViewHolder uvh = new UsuariosViewHolder(itemView);
         return uvh;
    }
}
```

Control RecyclerView
Gestión evento onItemClick()

3. El método adicional de nuestro adaptador, ese setter, setOnClickListener() nos servirá para asociar el listener *real* a nuestro adaptador en el momento de crearlo.

Veamos cómo quedaría nuestra actividad principal con este cambio:

```
public class  MainActivity extends AppCompatActivity {
private RecyclerView recyclerView;
private List<Usuario> listaUsuarios;
private AdaptadorUsuarios adaptadorUsuarios;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
                //…
        //inicializar recyclerview
        recyclerView=findViewById(R.id.rvListaUsuarios);
        adaptadorUsuarios=new AdaptadorUsuarios(listaUsuarios);
        adaptadorUsuarios.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(MainActivity.this, "Has pulsado el elemento en la posición: "+
                        recyclerView.getChildAdapterPosition(view), Toast.LENGTH_SHORT).show();
            }
        });
        recyclerView.setAdapter(adaptadorUsuarios);
        recyclerView.setLayoutManager(new
LinearLayoutManager(this,LinearLayoutManager.VERTICAL,false));
    }
}
```

Control RecyclerView
EFECTO VISUAL AL HACER CLICK

- Por defecto, al hacer click sobre uno de los elementos de la lista, éste no muestra ningún cambio visual.
- Si queremos añadir un efecto al pulsarlo:
  - En el elemento root del layout que representa cada fila de la lista (en nuestro ejemplo: item_layout.xml, ala propiedad "android:background" le damos el valor:

```
android:background="?android:attr/selectableItemBackground"
```

  - En nuestro ejemplo:

```
<android.support.constraint.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_width="match_parent"
    android:background="?android:attr/selectableItemBackground"
    android:minHeight="?android:attr/listPreferredItemHeightLarge"
    android:layout_height="wrap_content">
```

Control RecyclerView
PASOS PARA CREAR UN RECYCLERVIEW

1. **Añadir la librería de soporte "recyclerview-v7"** (en build.gradle)

   implementation **'com.android.support:recyclerview-v7:27.1.1'**

2. **Diseñar un layout** ***(para la Activity,…)*** **que contenga al RecyclerView y otro layout para cada elemento de la lista (item_layout)**
3. **Crear el RecyclerView.Adapter**
   - ***Definir el RecyclerView.ViewHolder dentro del adaptador***
4. En la Activity, donde se vaya a usar el RecyclerView:
   - **Obtener la referencia al RecyclerView**
   - **Asignarle el adaptador :** **recycler.setAdapter(xxxx);**
   - **Asociarle al RecyclerView un LayoutManager** que posicione las vistas adecuadamente

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

2. Diseñar un layout *(para la Activity,…)* que contenga al RecyclerView

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

2. Diseñar un layout *(para la Activity,…)* que contenga al RecyclerView

```
<android.support.v7.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:scrollbars="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

2. Diseñar el layout de cada elemento de la lista (item_layout.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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_width="match_parent"
    android:layout_height="?attr/listPreferredItemHeight">

    <TextView
        android:id="@+id/tvNombre"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Nombre"
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

2. Diseñar el layout de cada elemento de la lista (item_layout.xml):

```
    android:textSize="14sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/tvApellidos"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="Apellidos"
    android:textSize="14sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

2. Diseñar el layout de cada elemento de la lista (item_layout.xml):

```
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvNombre" />

<TextView
    android:id="@+id/tvDireccion"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:gravity="center"
    android:text="Dirección"
    android:textSize="14sp"
    android:textStyle="italic"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.51"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tvApellidos" />
</android.support.constraint.ConstraintLayout>
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

3. Crear el Recycler.Adapter:

- Este adaptador deberá extender a la clase RecyclerView.Adapter
- Tendremos que sobreescribir 3 métodos:

  - **onCreateViewHolder()**. Encargado de crear los nuevos objetos ViewHolder necesarios para los elementos de la colección.
  - **onBindViewHolder().** Encargado de actualizar los datos de un ViewHolder ya existente.
  - **getItemCount().** Indica el número de elementos de la colección de datos.

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

De momento sólo hacemos el esqueleto de la clase Adapter:

```
public class AdaptadorUsuarios extends
RecyclerView.Adapter<AdaptadorUsuarios.UsuariosViewHolder> {

@Override
    public UsuariosViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        return null;
    }

    @Override
    public void onBindViewHolder(@NonNull UsuariosViewHolder holder, int
position) {
    }

    @Override
    public int getItemCount() {
        return 0;
    }
}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

Antes de seguir, con este adaptador, necesitamos definir su UsuariosViewHolder (que será nuestro ViewHolder):

- Lo definiremos como una **clase interna y estática y que extiende de RecyclerView.VieHolder.**
- Debemos incluir en él **como atributos, las referencias a los controles que lleva cada layout item_layout de la lista** (en mi ejemplo, son 3 TextView: el nombre, apellidos y dirección).

Nos quedará:

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

```
public class AdaptadorUsuarios extends
RecyclerView.Adapter<AdaptadorUsuarios.UsuariosViewHolder> {
//...
    public static class UsuariosViewHolder extends RecyclerView.ViewHolder {
        private TextView tvNombre, tvApellidos,tvDireccion;

        public UsuariosViewHolder(View itemView) {
            super(itemView);

            tvNombre=itemView.findViewById(R.id.tvNombre);
            tvApellidos=itemView.findViewById(R.id.tvApellidos);
            tvDireccion=itemView.findViewById(R.id.tvDireccion);
        }


        public void bindUsuario(Usuario usuario){
            tvNombre.setText(usuario.getNombre());
            tvApellidos.setText(usuario.getApellidos());
            tvDireccion.setText(usuario.getDirección());
        }
    }
}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

Una vez hecho el ViewHolder, ya podemos seguir con el adaptador, sobreescribiendo los métodos del esqueleto anterior:

1. En el método **onCreateViewHolder()** nos limitaremos a:
   - **inflar una vista** a partir del layout correspondiente a los elementos de la lista (*item_layout*)
   - **crear y devolver un nuevo ViewHolder**
     - ✓ llamando al constructor de nuestra clase UsuariosViewHolder pasándole dicha vista como parámetro.

```
@Override
public UsuariosViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
    View itemView = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.item_layout, parent, false);

    UsuariosViewHolder usuariosViewHolder=new UsuariosViewHolder(itemView);
    return usuariosViewHolder;
}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

2. Necesitamos un atributo en el adaptador que sea la lista de datos (nuestro array) y creamos un constructor para inicializarlos:

```
public class AdaptadorUsuarios extends
RecyclerView.Adapter<AdaptadorUsuarios.UsuariosViewHolder> {
    List<Usuario> listaUsuarios;


    public AdaptadorUsuarios(List<Usuario> listaUsuarios) {
        this.listaUsuarios = listaUsuarios;
    }

//...

}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

3. En el método **onBindViewHolder()** nos limitaremos a:
   - **recuperar el objeto Usuario correspondiente a la posición recibida** como parámetro
   - **asignar sus datos sobre el ViewHolder** también recibido como parámetro

```
@Override
public void onBindViewHolder(@NonNull UsuariosViewHolder holder, int
position) {
        Usuario usuario=listaUsuarios.get(position);
        holder.bindUsuario(usuario);
}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

4. En el método **getItemCount()** nos limitaremos a devolver el tamaño de la lista de datos (usuarios):

```
@Override
public int getItemCount() {
    return listaUsuarios.size();
}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

Estaríamos en el paso 4, en el que en el MainActivity, tendríamos que:

- Obtener la referencia al RecyclerView
- Asignarle el adaptador : recycler.setAdapter(xxxx);
- Asociarle al RecyclerView un LayoutManager que posicione las vistas adecuadamente

```
public class   MainActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private List<Usuario> listaUsuarios;
    private AdaptadorUsuarios adaptadorUsuarios;

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

            initDatosEjemplo();

            //inicializar recyclerview
            recyclerView=findViewById(R.id.rvListaUsuarios);
            adaptadorUsuarios=new AdaptadorUsuarios(listaUsuarios);
            recyclerView.setAdapter(adaptadorUsuarios);
            recyclerView.setLayoutManager(new
    LinearLayoutManager(this,LinearLayoutManager.VERTICAL,false));

        }
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

```
    private void initDatosEjemplo() {
        listaUsuarios=new ArrayList<>();
        for(int i=0;i<15; i++){
            listaUsuarios.add(new Usuario("Nombre "+i, "Apellidos "+i, "Dirección "+i,0,0,0);
        }
    }
}
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

Nos faltaría implementar el onItemClickListener. Para ello en el adaptador escribimos:

```
public class AdaptadorUsuarios extends
RecyclerView.Adapter<AdaptadorUsuarios.UsuariosViewHolder> implements View.OnClickListener
{
    private List<Usuario> listaUsuarios;
    private View.OnClickListener listener;
//…
    @Override
    public UsuariosViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
//…
        itemView.setOnClickListener(this);
        //…
        return usuariosViewHolder;
    }

//…

    public void setOnClickListener(View.OnClickListener listener){
        this.listener=listener;
    }
```

Control RecyclerView
EJEMPLO de una LISTA DE USUARIOS

Y en el MainActivity:

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

    initDatosEjemplo();

    //inicializar recyclerview
    recyclerView=findViewById(R.id.rvListaUsuarios);
    adaptadorUsuarios=new AdaptadorUsuarios(listaUsuarios);
    adaptadorUsuarios.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(MainActivity.this, "Has pulsado el elemento en la posición:
"+recyclerView.getChildAdapterPosition(view), Toast.LENGTH_SHORT).show();
        }
    });
    recyclerView.setAdapter(adaptadorUsuarios);
    recyclerView.setLayoutManager(new
LinearLayoutManager(this,LinearLayoutManager.VERTICAL,false));

}
```