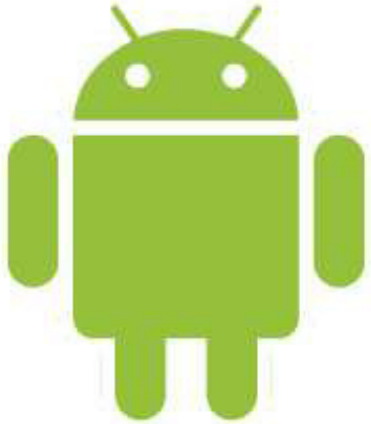
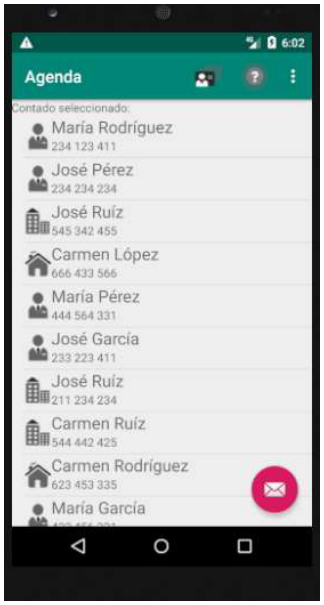
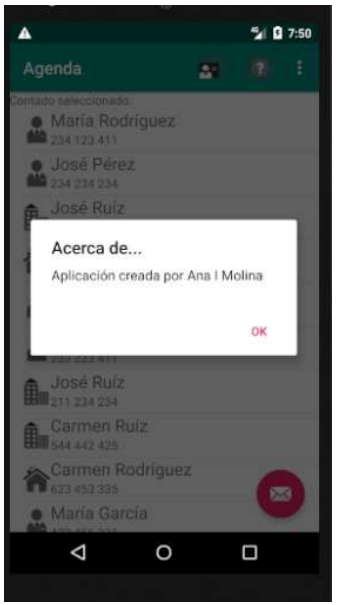
Introducción al desarrollo de aplicaciones móviles con Android (II)

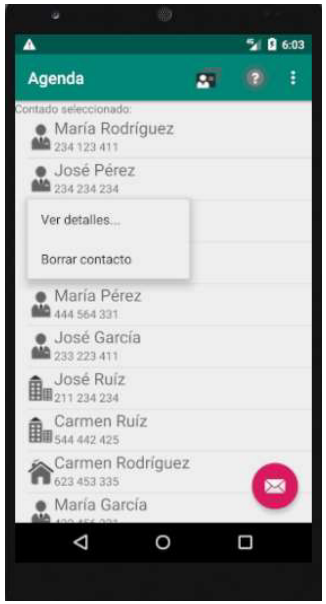


En este **segundo seminario** aprenderemos…

- Uso y personalización del componente para visualizar listas (*RecyclerView*)
  - Contenido dinámico
  - Borrado y modificación de elementos
- Creación y personalización de menús (*Action Bar*)
- Creación de menús contextuales
- Más técnicas de notificación en Android
  - Diálogos
- Más *layouts* y controles
  - *TableLayout*
  - *Spinner*

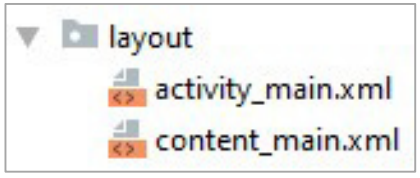


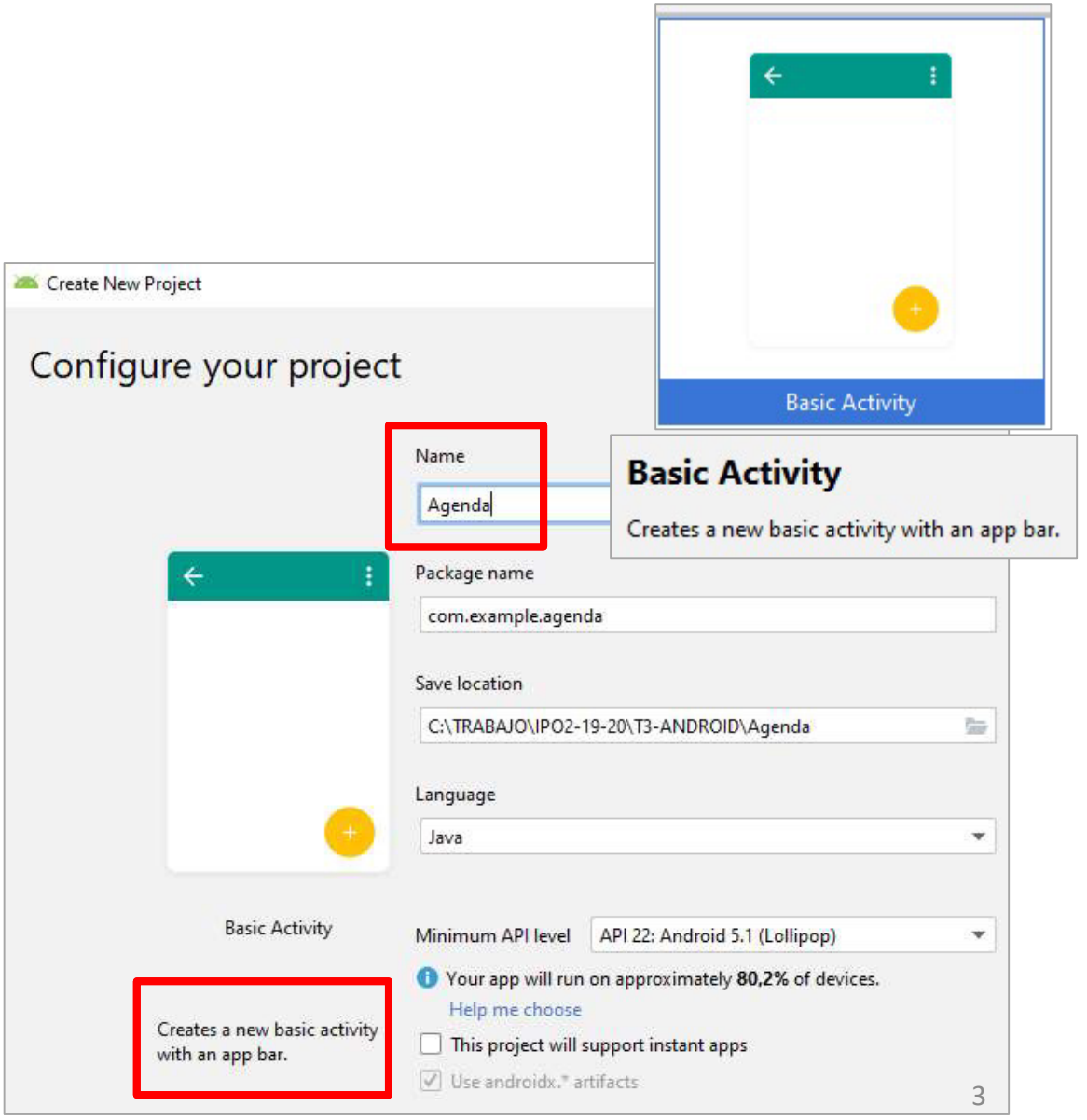




Aplicación Agenda (Lista de contactos)

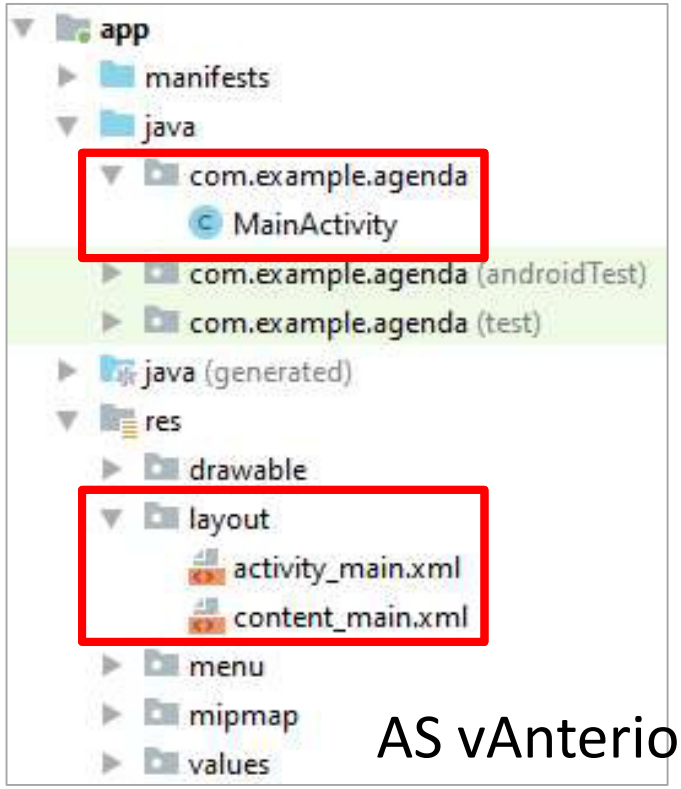
- Creamos un nuevo proyecto, al que llamaremos **Agenda**
- Mininum SDK: API 23 (84.9%)
- Crear la actividad principal
  - *Activity Name*: MainActivity
- En este caso se generan dos ficheros xml.
- Trabajaremos sobre ***content_main.xml***
- En el fichero ***activity_main.xml*** están definidos el *FloatingActionButton* y la *Toolbar.*



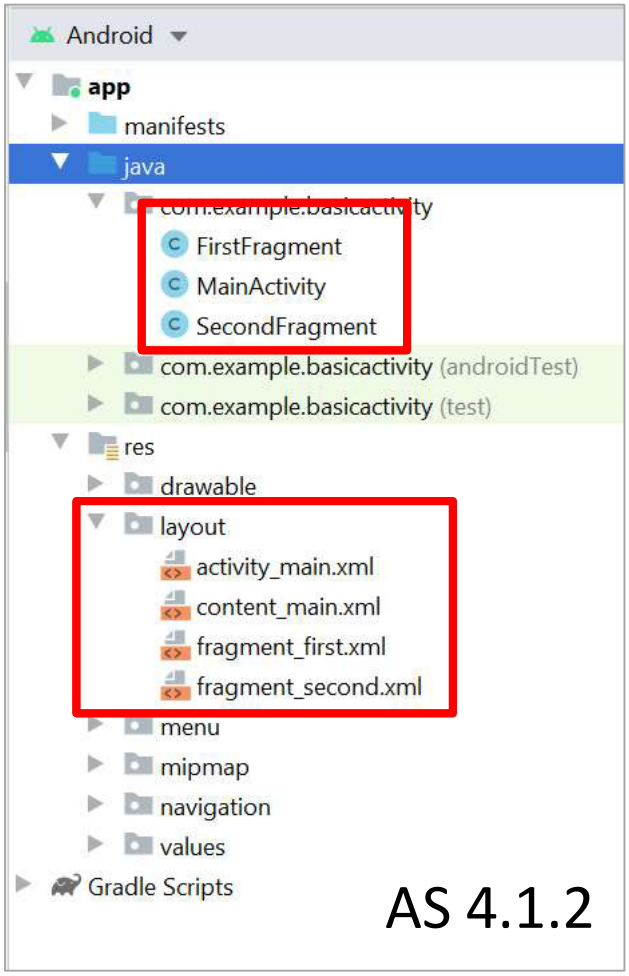


Cambios en AS 4.1.2

- Se genera el proyecto con algunos cambios:



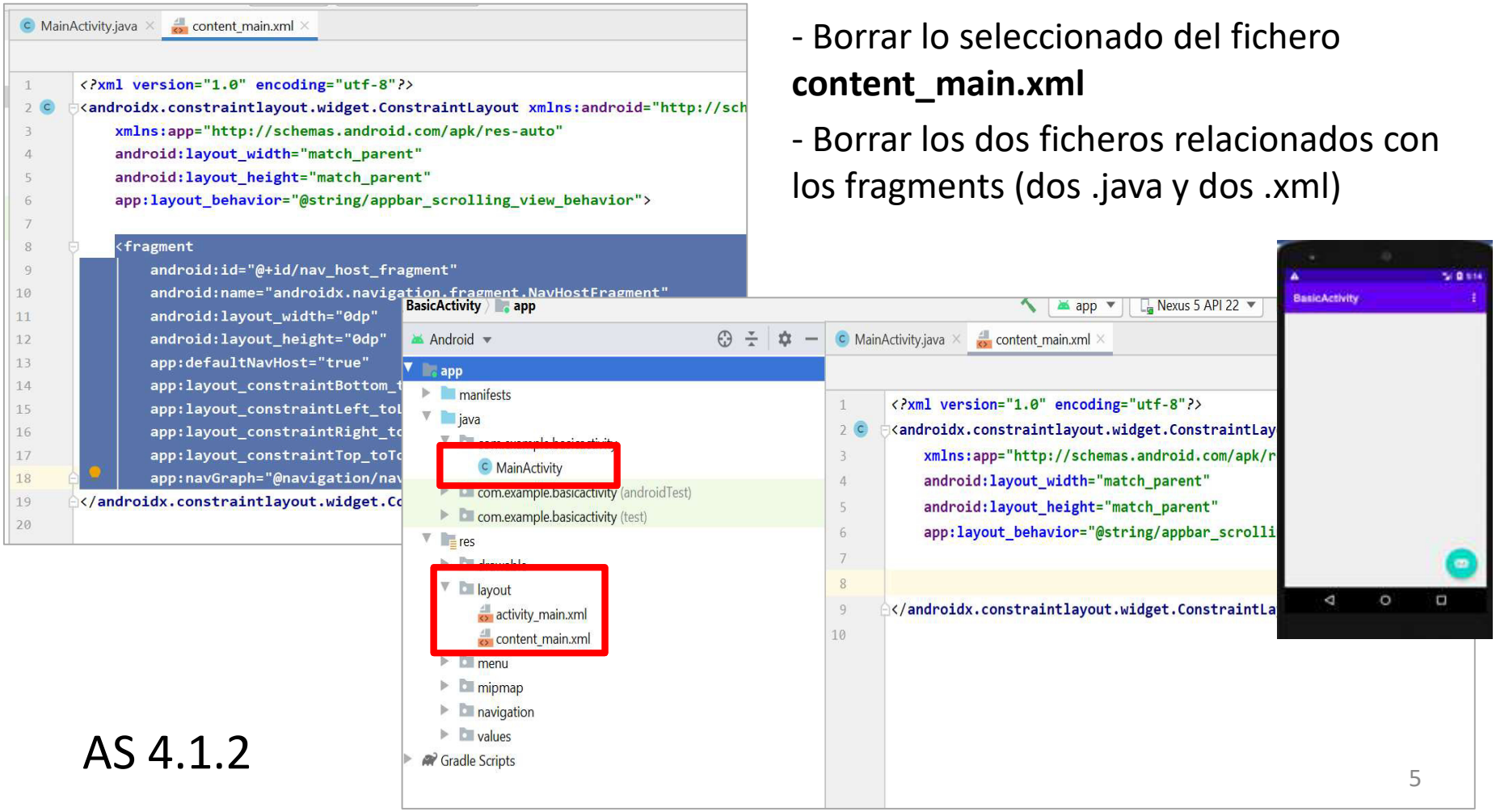
AS vAnterior



AS 4.1.2

Cambios en AS 4.1.2

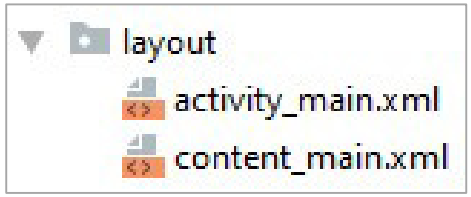
- Se genera el proyecto con algunos cambios:



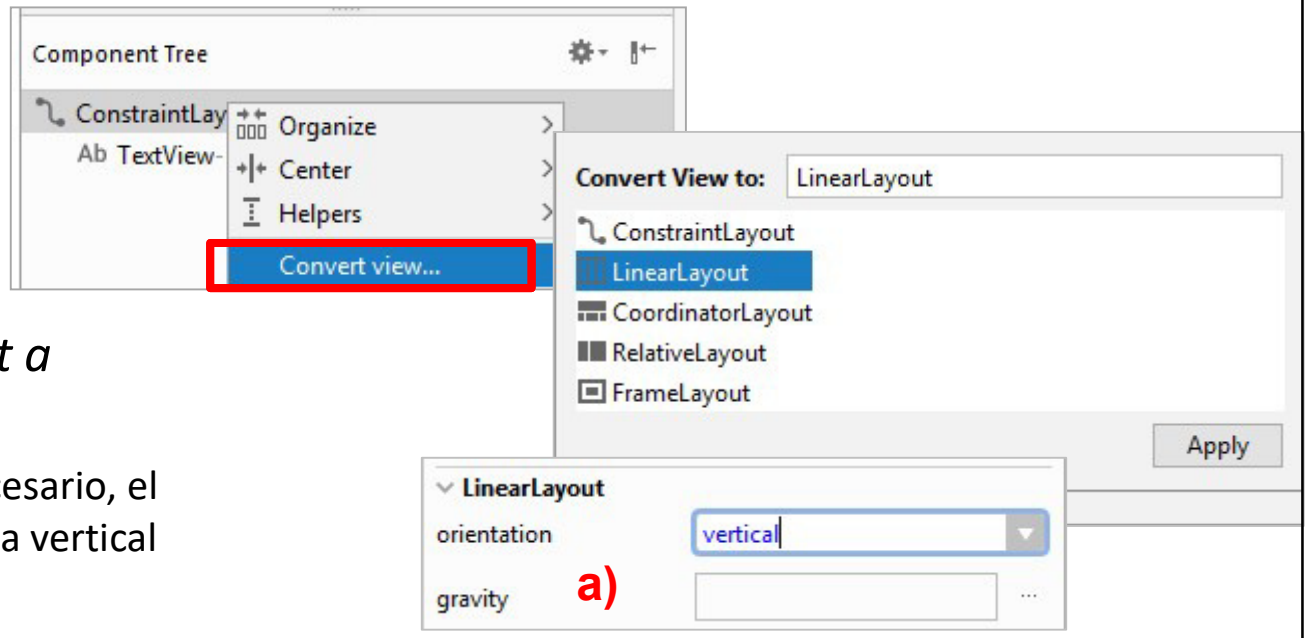
- Borrar lo seleccionado del fichero **content_main.xml**
- Borrar los dos ficheros relacionados con los fragments (dos .java y dos .xml)

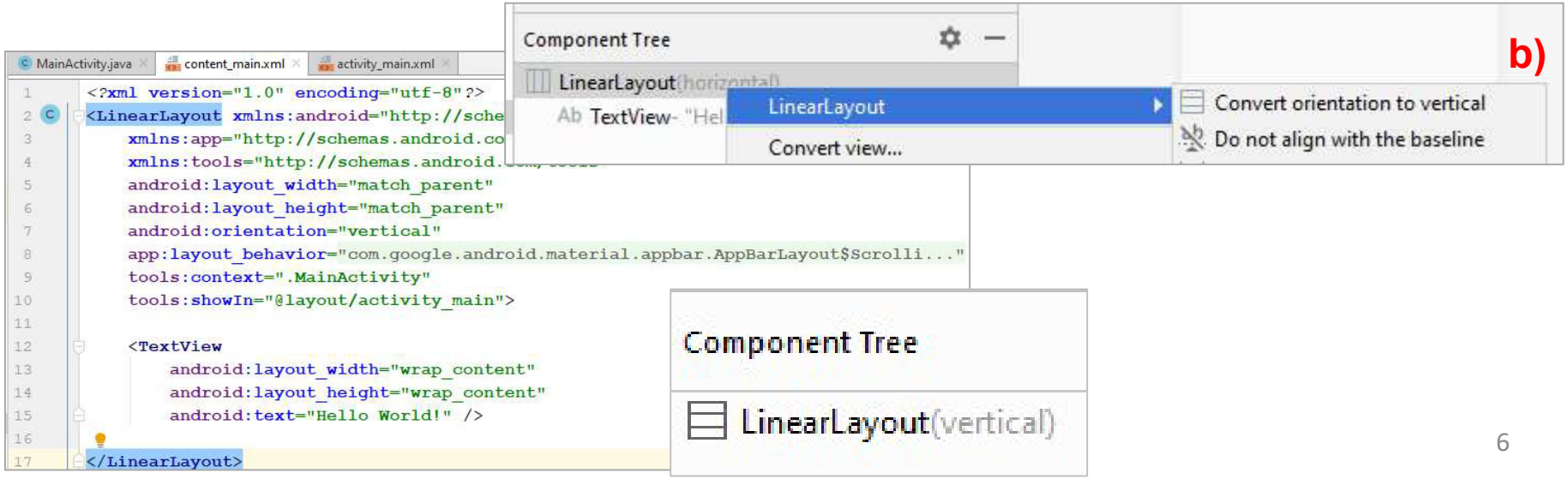
AS 4.1.2

Aplicación Agenda (Lista de contactos)



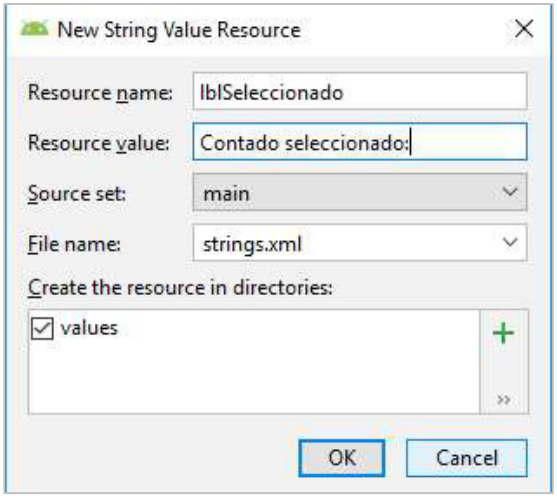
- Trabajaremos sobre ***content_main.xml***
  - Cambiamos el *layout a* ***LinearLayout***.
    - Cambiar, si fuera necesario, el atributo **orientation** a vertical

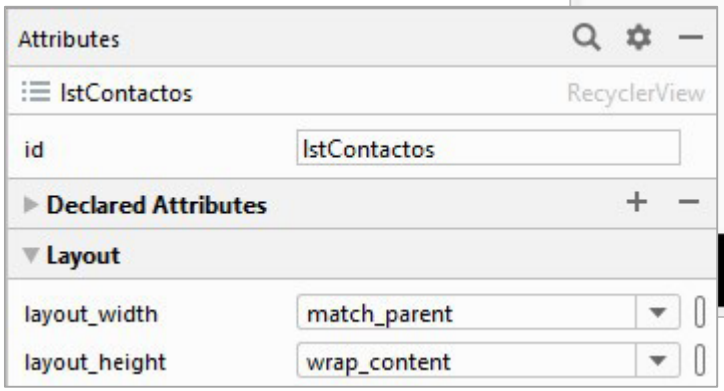


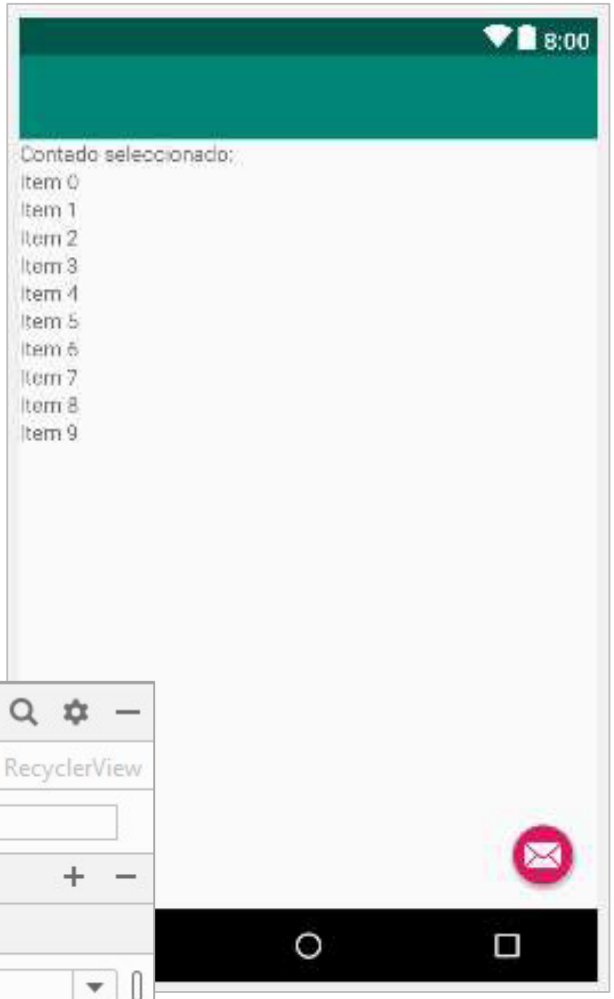


Aplicación Agenda (Lista de contactos)

- Interfaz de usuario
  - **Text/Common:** *TextView* (ya aparece una por defecto)
  - **Containers/Common:** *RecyclerView*



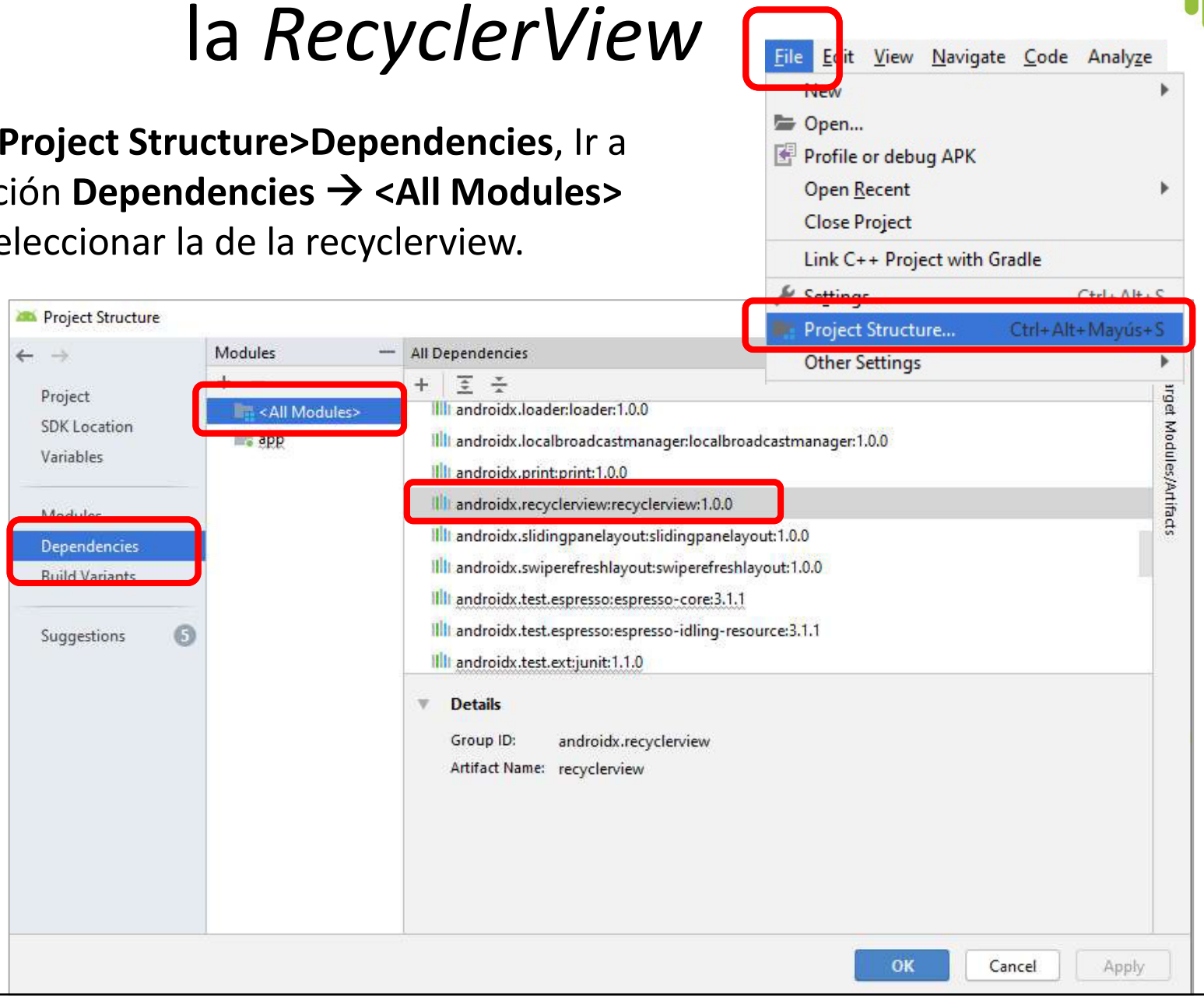




- Ejecuta la aplicación…

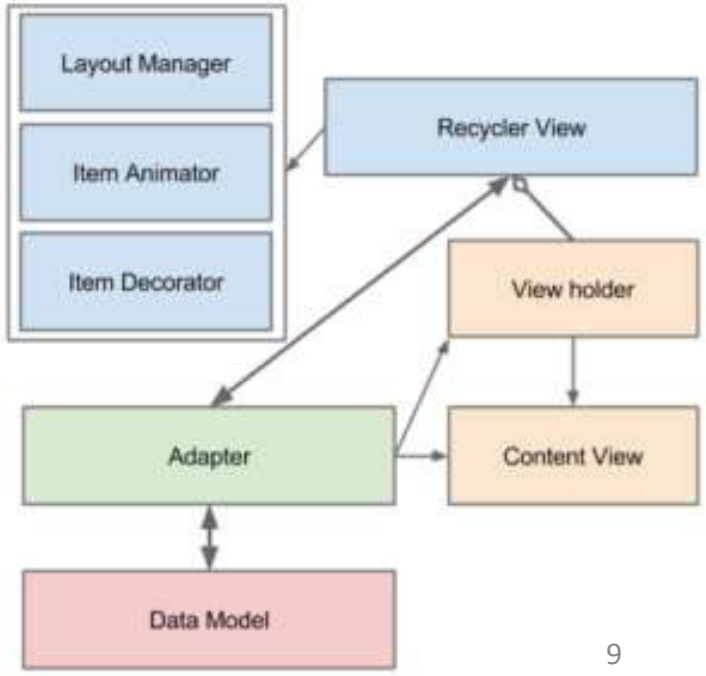
Añadir dependencias para usar la *RecyclerView*

- **File > Project Structure>Dependencies**, Ir a la sección **Dependencies → <All Modules>** → y seleccionar la de la recyclerview.

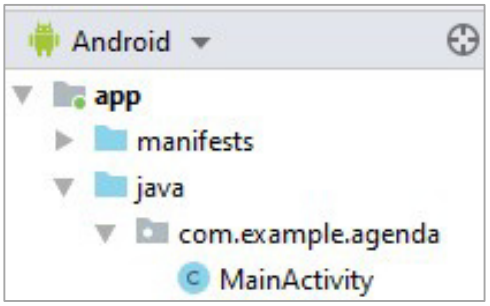


El componente **RecyclerView**

- ***RecyclerView*** (al igual que su antecesor *ListView*) permite mostrar en pantalla colecciones de datos.
  - *RecyclerView* no va a hacer "casi nada" por sí mismo, sino que se va a sustentar sobre otros componentes complementarios para determinar cómo acceder a los datos y cómo mostrarlos.
  - Los más importantes serán los siguientes:
    - `RecyclerView.Adapter`
    - `RecyclerView.ViewHolder`
    - `LayoutManager`
    - `ItemDecoration`
    - `ItemAnimator`
- Una vista de tipo *RecyclerView* no determina por sí sola la forma en que se van a mostrar en pantalla los elementos de nuestra colección, sino que va a delegar esa tarea a otro componente llamado ***LayoutManager***.
  - El SDK incorpora de serie tres *LayoutManager* para las tres representaciones más habituales: lista vertical u horizontal (*LinearLayoutManager*), tabla tradicional (*GridLayoutManager*) y tabla apilada o de celdas no alineadas (*StaggeredGridLayoutManager*).



Añadir **contenido** a la lista de contactos

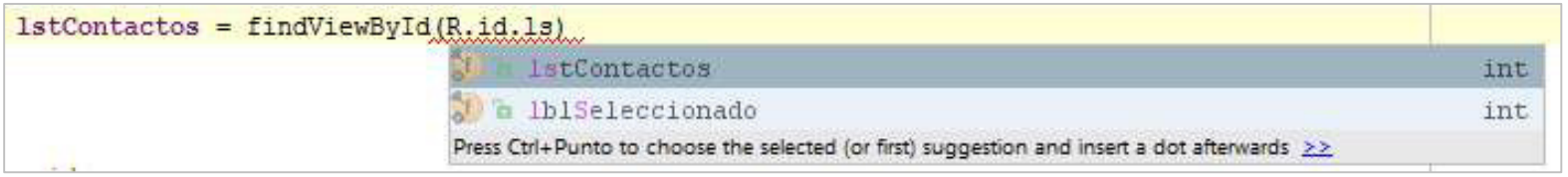


- Añadimos un atributo privado de tipo *ArrayList* que contendrá los datos de los contactos (**MainActivity.java**):

```
private ArrayList<Contacto> contactos;
```

- Necesitamos obtener una referencia al objeto *lstContactos* (definido en el fichero de interfaz; *xml*) en el fichero **MainActivity.java**
  - Creamos un **atributo privado en la clase**
  - Usamos el método ***findViewById*** para obtener dicha referencia en el método **onCreate**
  - Hacer los **imports** necesarios **(Alt+Enter)**

```
private RecyclerView lstContactos;
```

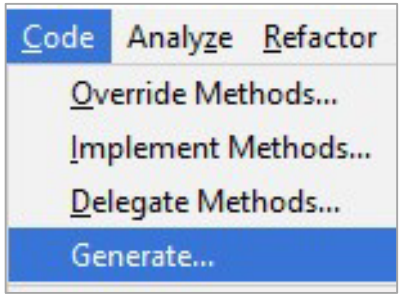



```
lstContactos = findViewById(R.id.lstContactos);
```

Añadir **contenido (personalizado y dinámico)** a la lista de contactos

- Tal y como se ha indicado, se va a usar *estructura dinámica (ArrayList)*, en cada uno de cuyos elementos se añadirá una instancia de la clase **Contacto** que deberemos crear.
  - New→ Java Class

- Para generar el constructor los métodos get y set se puede usar **Code>Generate...**




```
package com.example.agenda;

public class Contacto {

	private String nombre;
	private String telefono;
	private int tipo; //0:familia; 1:amigo;
2:trabajo
	private String email;
	private String direccion;

	public Contacto(String nom, String tel, int
tip, String em, String dir){
		nombre= nom;
		telefono= tel;
		tipo=tip;
		email=em;
		direccion=dir;

	}
	public String getNombre(){
		return nombre;
	}
	public String getTelefono(){
		return telefono;
	}
	public int getTipo(){
		return tipo;
	}
	public String getEmail(){
		return email;
	}
	public String getDireccion(){
		return direccion;
	}
}
```

Añadir **contenido (personalizado y dinámico)** a la lista de contactos

- En el método **onCreate** inicializamos la lista de contactos con algunos datos de prueba (se podrían tomar de un fichero de texto, un xml, una BD). Para este ejemplo añadiremos una serie de datos de prueba al array (crearemos un método para ello):

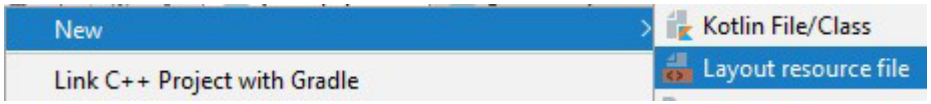
```
//Obtener una referencia a la lista gráfica
lstContactos = findViewById(R.id.lstContactos);
//Crear la lista de contactos y añadir algunos datos de prueba
contactos = new ArrayList<Contacto>();
//Método que rellena el array con datos de prueba
rellenarDatosPrueba();
```

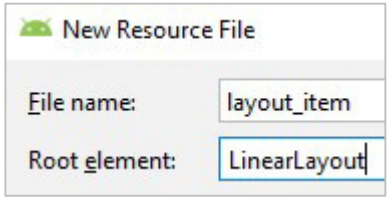
```
public void rellenarDatosPrueba()
{
      contactos.add(new Contacto("María Rodríguez", "234 123 411",1, "mariarodriguez@correo.com", "C/Ronda, 10"));
      contactos.add(new Contacto("José Pérez", "234 234 234",1, "joseperez@correo.com", "C/Huertas, 1"));
      contactos.add(new Contacto("José Ruíz", "545 342 455",2, "joseruiz@correo.com", "C/Ancha, 7"));
      contactos.add(new Contacto("Carmen López", "666 433 566",0, "carmenlopez@correo.com", "C/Luz, 12"));
      contactos.add(new Contacto("María Pérez", "444 564 331",1, "mariapezar@correo.com", "C/Ciudad, 11"));
      contactos.add(new Contacto("José García", "233 223 411",1, "josegarcia@correo.com", "C/Darro, 4"));
      contactos.add(new Contacto("José Ruíz", "211 234 234",2, "joseruiz@correo.com", "C/Ronda, 6"));
      contactos.add(new Contacto("Carmen Ruíz", "544 442 425",2, "carmenrodriguez@correo.com", "C/Principal, 13"));
      contactos.add(new Contacto("Carmen Rodríguez", "623 453 335",0, "carmenrodriguez@correo.com", "C/Rodero, 5"));
      contactos.add(new Contacto("María García", "432 456 331",1, "mariagarcia@correo.com", "C/Paseo, 8"));
}
```

- Tenemos que conectar la vista (el objeto *lstContactos*) con los nuevos datos a mostrar (contactos): **Adaptador**
- En este caso **crearemos una clase adaptadora personalizada**, que permita mostrar en cada ítem de la lista:
  - nombre, teléfono y una imagen identificativa del tipo de contacto (familia, amigo, trabajo).

Definir el **aspecto gráfico** de cada uno de los **ítems de la lista**

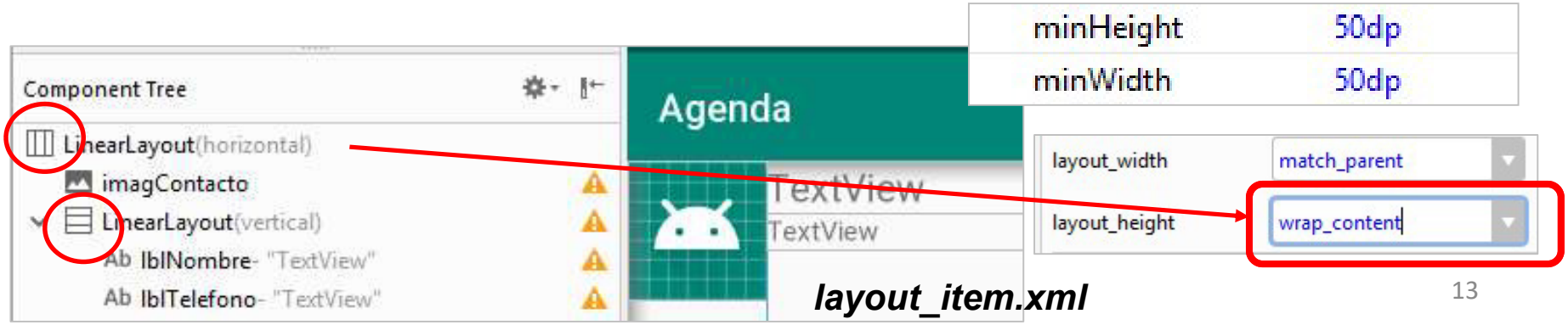
- Primero creamos el **aspecto gráfico** que tendrá **cada uno de los ítems de la lista**
- Tendremos que crear un **nuevo fichero de *layout***
  - En la carpeta **/res/layout/** creamos un nuevo fichero *xml* (***layout_item.xml***)
  - El *layout* será un *LinearLayout* horizontal





  - Añadir un *TextView* (**lblNombre)** a 2**0sp**
  - Quitar el valor de la propiedad **layout_weight**

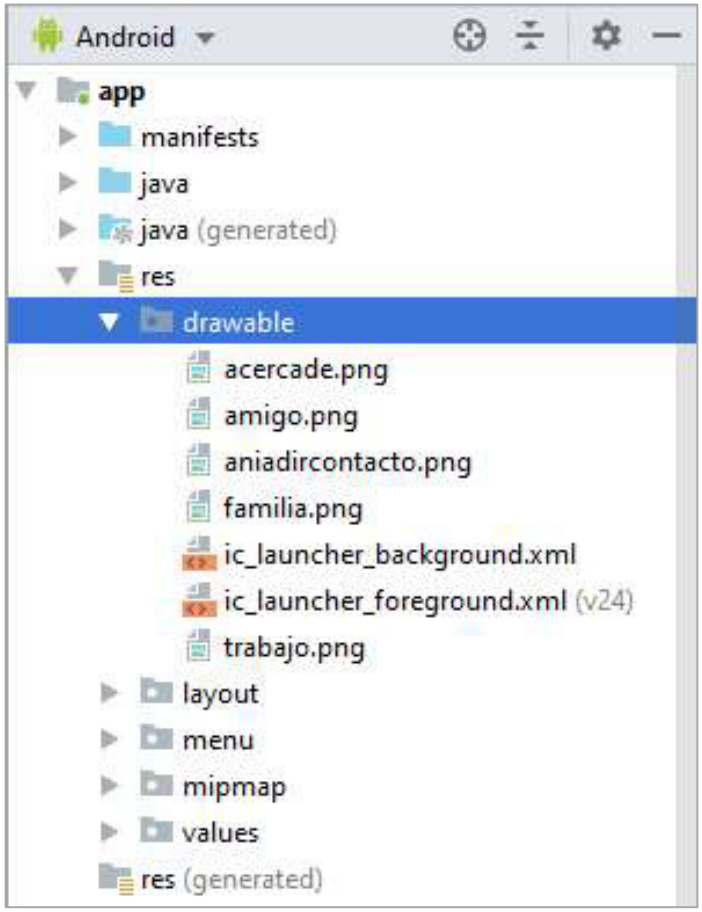
  - Especificar el tamaño de la *ImageView* (**imagContacto**)
    - **MinHeight: 50dp** y **MinWidth: 50dp**



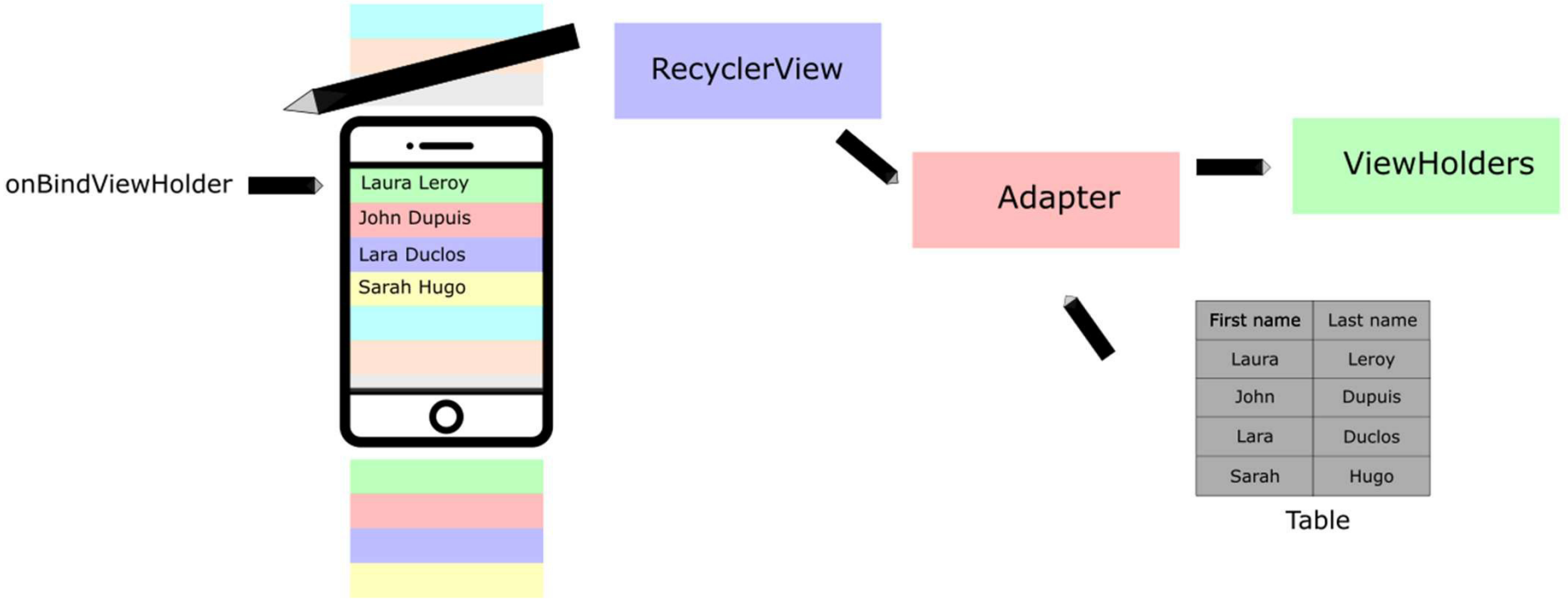
layout_item.xml

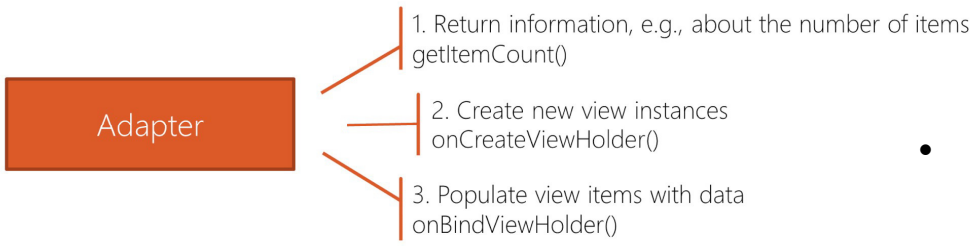
Definir el **aspecto gráfico** de cada uno de los **ítems de la lista**

- Incluir las **imágenes** que se usarán para renderizar cada uno de los ítems según su tipo (familia, amigo, trabajo).
  - **/res/drawable/**
    - Botón derecho>Show in Explorer
    - Copiar las imágenes en la carpeta
  - O directamete arrastrarlas a la carpeta



Creamos una **clase Adaptadora**



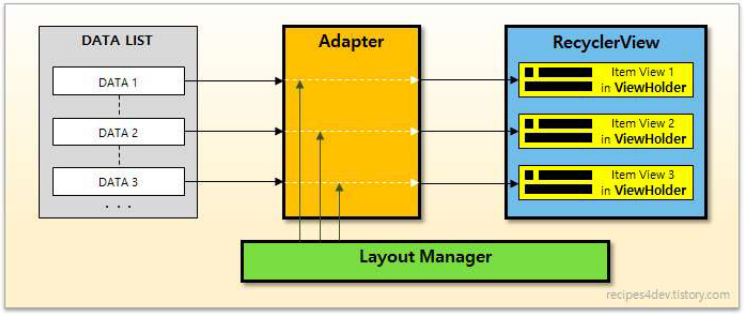


- **onCreateViewHolder().** Encargado de crear los nuevos objetos **ViewHolder** necesarios para los elementos de la colección.
- **onBindViewHolder()**. Encargado de actualizar los datos de un **ViewHolder** ya existente.
- **onItemCount()**. Indica el número de elementos de la colección de datos.

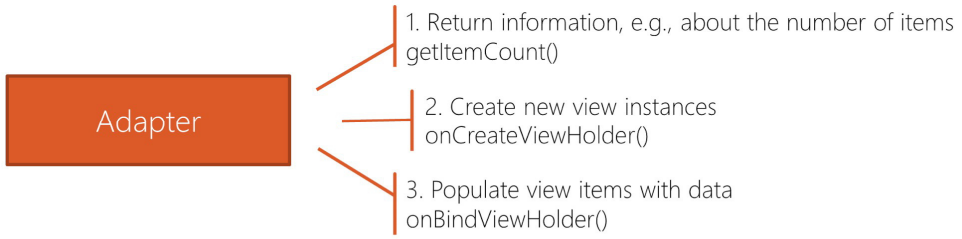
Creamos una **clase Adaptadora**

- New → Java Class
- La nueva clase (a la que llamaremos **AdaptadorLista.java**) **extenderá de la clase *RecyclerView.Adapter***

```
public class AdaptadorLista extends
        RecyclerView.Adapter<AdaptadorLista.ViewHolder> {

    private ArrayList<Contacto> contactos;

    class ViewHolder extends RecyclerView.ViewHolder {

        private TextView lblNombre;
        private TextView lblTelefono;
        private ImageView imagContacto;

        ViewHolder(View view) {
            super(view);

            lblNombre = view.findViewById(R.id.lblNombre);
            lblTelefono = view.findViewById(R.id.lblTelefono);
            imagContacto = view.findViewById(R.id.imagContacto);
        }
    }

    public AdaptadorLista(ArrayList<Contacto> contactos){
        this.contactos = contactos;
    }

...
```



```
...

    @Override
    public AdaptadorLista.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.layout_item, parent, false);

        return new ViewHolder(v);
    }

    @Override
    public void onBindViewHolder(AdaptadorLista.ViewHolder holder, int position) {
        Contacto contacto = contactos.get(position);
        holder.lblNombre.setText(contactos.get(position).getNombre());
        holder.lblTelefono.setText(contactos.get(position).getTelefono());
        switch (contactos.get(position).getTipo())
        {
            case 0: //Cargar imagen de contactos tipo "familia"
                holder.imagContacto.setImageResource(R.drawable.familia);
                break;
            case 1: //Cargar imagen de los contactos tipo "amigos"
                holder.imagContacto.setImageResource(R.drawable.amigo);
                break;
            case 2: //Cargar imagen de los contactos tipo "trabajo"
                holder.imagContacto.setImageResource(R.drawable.trabajo);
        }
    }

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



Añadir contenido a la lista de contactos: Asociar la *RecyclerView* con su **Adaptador**

- Asociar al objeto *RecyclerView* el adaptador (en **MainActivity.java**)
  - Crear un atributo privado a nivel de clase

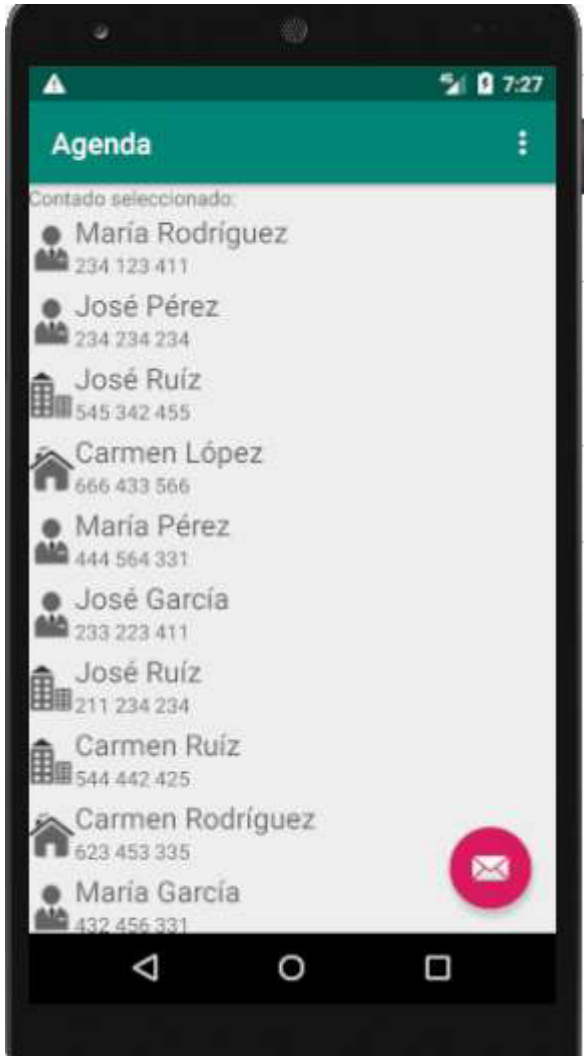
```
private AdaptadorLista adaptador;
```

  - Añadir las siguientes instrucciones detrás de la inicialización del *ArrayList* contactos.

```
RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(getApplicationContext());

lstContactos.setLayoutManager(mLayoutManager);

adaptador = new AdaptadorLista(contactos);
lstContactos.setAdapter(adaptador);
```

Añadir contenido a la lista de contactos: **Adaptador**

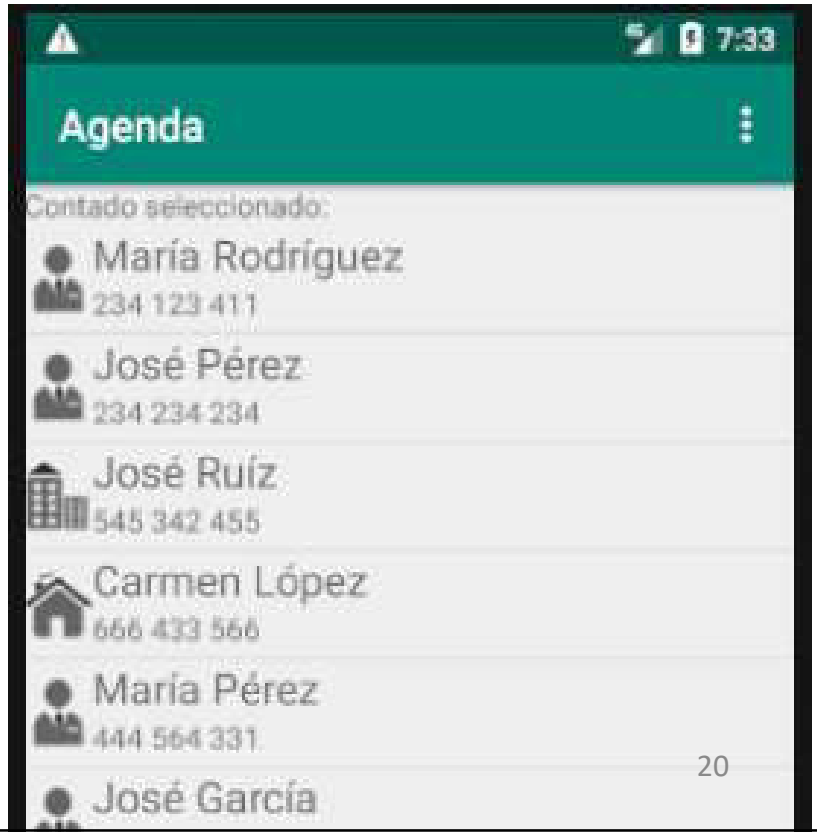
- Ejecuta la aplicación….



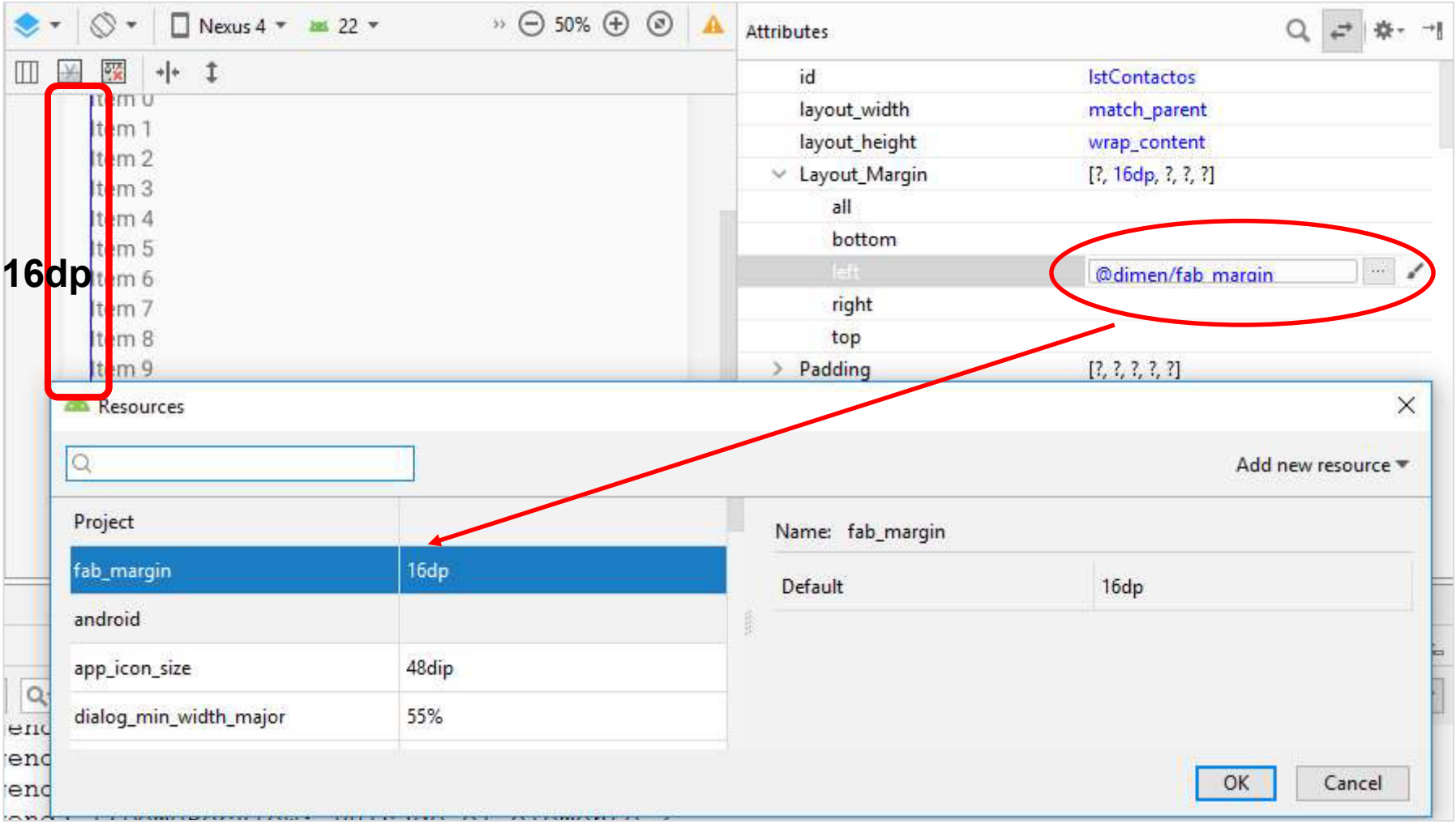
Añadir un separador de ítems

- Hacer uso del método `addItemDecoration` del `RecyclerView`

```
lstContactos.addItemDecoration(new DividerItemDecoration(this,
LinearLayoutManager.VERTICAL));
```

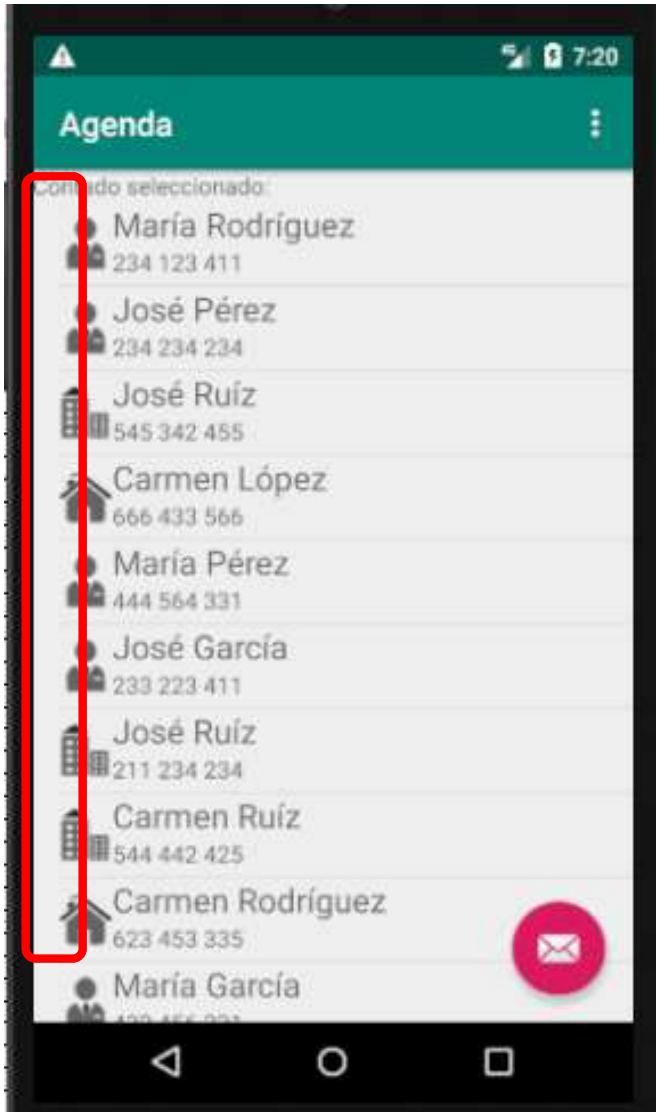


Modificar el aspecto de la interfaz



Añadir contenido a la lista de contactos: **Adaptador**

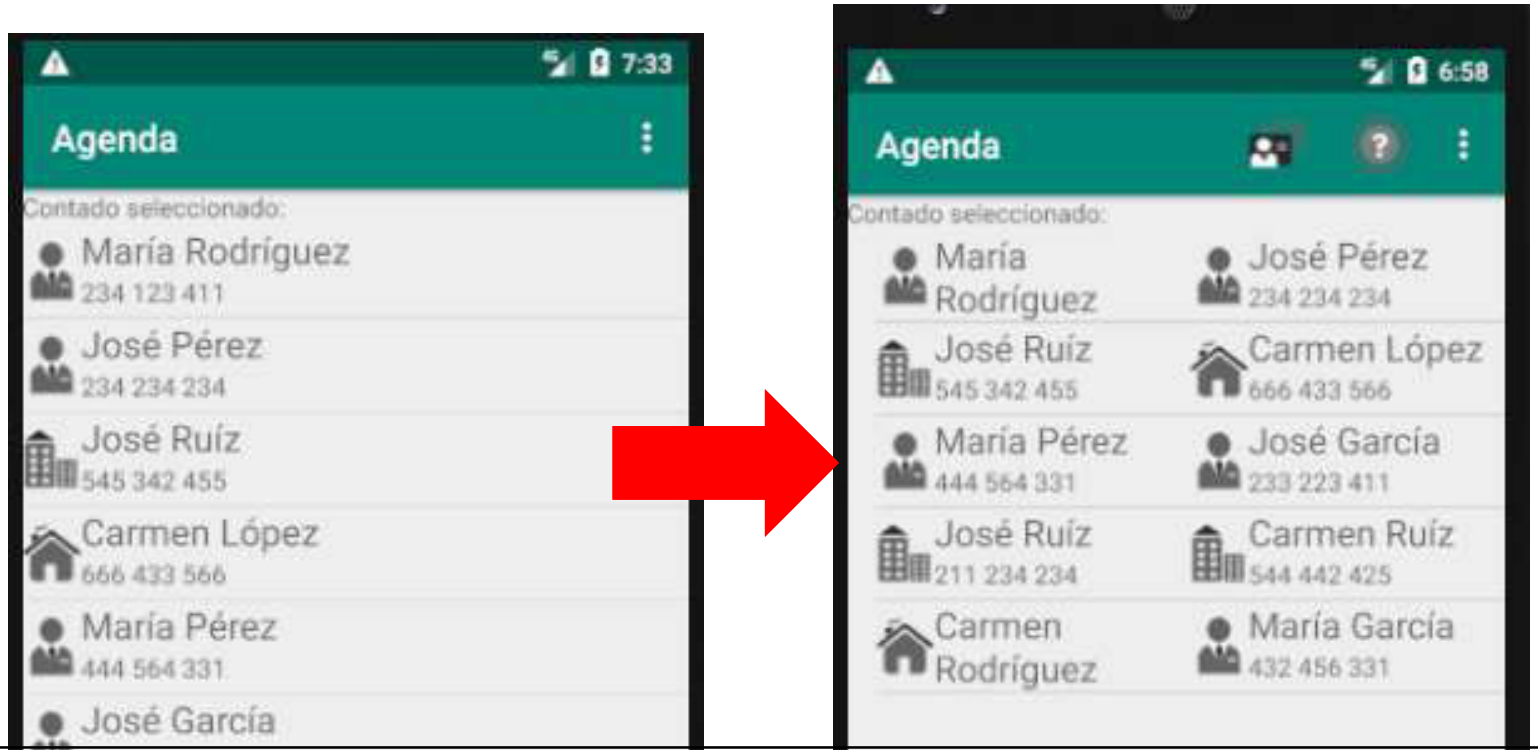
- Ejecuta la aplicación….



Cambiar el Layout a Grid

- Probar a cambiar el *LayoutManager* del *RecyclerView*

```
//RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(getApplicationContext());
RecyclerView.LayoutManager mLayoutManager = new
GridLayoutManager(this,2);
```



Añadimos un **oyente de selección de ítems** de la lista

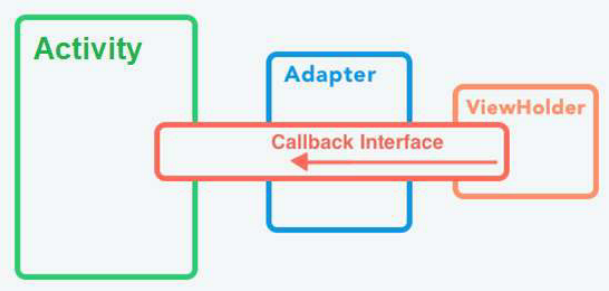
- Añadimos un **oyente de selección de ítems** asociado a la *RecyclerView.*
  - Hará que en *lblSeleccionado* aparezca el nombre del contacto seleccionado por el usuario en la lista
  - Para ello hay que tomar una referencia a la *TextView* (findViewById), al igual que se ha hecho con la *RecyclerView*
  - Hacer los **imports** correspondientes

```
private TextView lblSeleccionado;
```

```
//Obtener una referencia a la etiqueta en la que se mostrará el ítem
//seleccionado
lblSeleccionado = findViewById(R.id.lblSeleccionado);
```

- la clase *RecyclerView* no incluye un evento *onItemClick()* como ocurría con su predecesora (*ListView*)
  - *RecyclerView* delegará esta tarea a otro componente, en este caso a la propia vista que representa a cada elemento de la colección (el ***ViewHolder***).
  - Hay varias formas de implementar el oyente. Usaremos una de ellas, pero hay otras formas de crearlo y asociarlo.

Añadimos un **oyente de selección de ítems** de la lista



Interfaz

```
OnItemSelectedListener
- onContactoSeleccionado()
```

implements

MainActivity

```
adaptador.setItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onContactoSeleccionado(int posicion) {
        lblSeleccionado.setText("Contacto seleccionado: "+
                contactos.get(posicion).getNombre());
    }
});
```

AdaptadorLista

```
private OnItemSelectedListener itemSelectedListener;

public void setItemSelectedListener
        (OnItemSelectedListener itemSelectedListener) {
    this.itemSelectedListener = itemSelectedListener;
}
```

ViewHolder

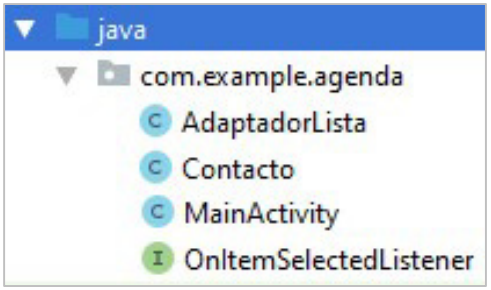
```
view.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int posicion = getAdapterPosition();
        if (itemSelectedListener != null) {
            itemSelectedListener.onContactoSeleccionado(posicion);
        }
    }
});
```

Añadimos un **oyente de selección de ítems** de la lista

- PASOS:

1. Creamos la interface **OnItemSelectedListener**:

```
public interface OnItemSelectedListener {
    void onContactoSeleccionado(int posicion);
}
```



2. En la clase **AdaptadorLista** declaramos un atributo privado (listener) que tendrá como tipo la interface recién creada (**OnItemSelectedListener**) :

```
private OnItemSelectedListener itemSelectedListener;
```

3. Creamos (o generamos-**Code>Generate>Setter…**) el método que permite registrarse como oyentes de la clase Adaptadora a otras clases (**setItemSelectedListener**):

```
public void setItemSelectedListener(OnItemSelectedListener itemSelectedListener)
{
    this.itemSelectedListener = itemSelectedListener;
}
```

Añadimos un **oyente de selección de ítems** de la lista

4. La clase **ActivityMain** querrá registrarse como oyente de los eventos de selección que se den en la clase Adaptadora; por tanto hará uso del método **setItemSelectedListener** (recién creado en AdaptadorLista.java).

```
adaptador = new AdaptadorLista(contactos);
lstContactos.setAdapter(adaptador);
adaptador.setItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onContactoSeleccionado(int posicion) {
            lblSeleccionado.setText("Contacto seleccionado: "+
                contactos.get(posicion).getNombre());
    }
});
```

Añadimos un **oyente de selección de ítems** de la lista

5. El elemento que responderá a eventos de tipo **OnClick** será la **View** asociada al ***ViewHolder.***

Pasará la posición del elemento seleccionado en la lista a la MainActivity.

```
view.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int posicion = getAdapterPosition();
        if (itemSelectedListener != null) {
            itemSelectedListener.onContactoSeleccionado(posicion);
        }
    }
});
```

Añadimos un **oyente de selección de ítems** de la lista

- Ejecuta la aplicación…