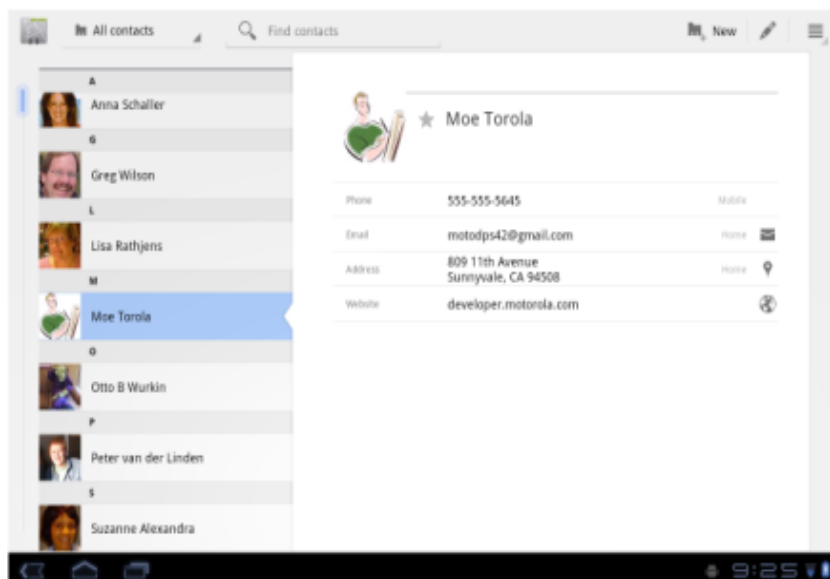




Fragmentos-II





Contenidos

Usando el asistente de creación de fragmentos (P_55_Fragmentos_6)	3
Plantilla Master/Detail flow (P_58_Fragmentos_7).....	14



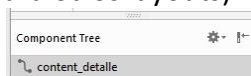
Usando el asistente de creación de fragmentos (P_53_Fragmentos_6)

Deseamos una aplicación semejante a P_44_Fragmentos_5 pero en la que la elección de la provincia se realice a través de una lista (la plataforma recomienda RecyclerView en vez de ListView y AS sigue el consejo). Como trabajar con fragmentos que contienen este tipo de vista es muy usual, AS proporciona un asistente de creación de este tipo de fragmentos.

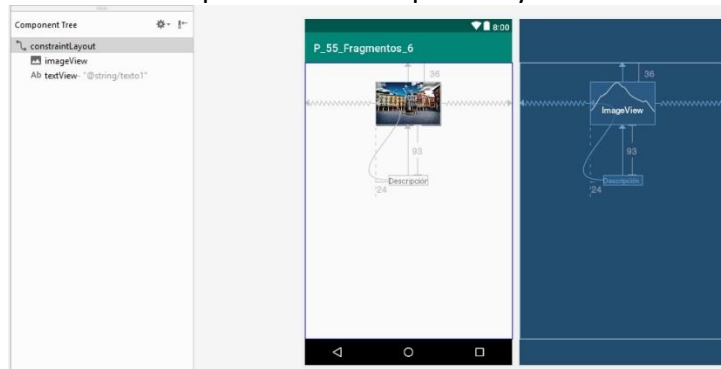
Creación de MainActivity, DetalleActivity y DetalleFragmento:

Para tener las imágenes necesarias copia a la carpeta drawable, las que están en la carpeta recursos (recuerda obtener las de las distintas densidades).

Creamos nuestro proyecto como siempre, le añadimos la segunda actividad llamada DetalleActivity (recuerda fijar Hierarchy Parent cuando la creas!) y fijamos un identificador para su layout vacío, por ejemplo content_detalle (recuérdalo, usarás este nombre muchas veces más para otros layouts):



Creamos el fragmento DetalleFragment con las mismas características que en el proyecto Fragmentos_05 (con una imagen y un texto en su layout, para que no se "amontonen" es conveniente que cambies el tipo de layout a ConstraintLayout).



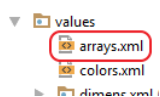
Trabajando con arrays:

En el código de nuestras clases los datos de la ciudad (nombre, descripción y foto) que deben mostrarse dependen de la posición del RecyclerView seleccionada, para el primer ítem serán los de la primera ciudad, para el segundo los de la segunda, etc. Lo más útil será trabajar con arrays. Para los strings podemos usar el fichero strings.xml:

```
<string-array name="ciudades">
  <item>Zaragoza</item>
  <item>Huesca</item>
  <item>Teruel</item>
</string-array>
<array name="descripciones">
  <item>Zaragoza, ciudad bañada por el Ebro,\nbla, bla, bla\nbla, bla, bla</item>
  <item>Huesca, cercana a los Pirineos,\nbla, bla, bla\nbla, bla, bla</item>
  <item>Teruel, capital del bajo Aragón,\nbla, bla, bla\nbla, bla, bla</item>
</array>
```

Para las imágenes necesitamos crear el fichero de recursos de tipo values llamado arrays.xml y cuyo contenido sea:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <array name="fotos">
    <item>@drawable/zaragoza</item>
    <item>@drawable/huesca</item>
    <item>@drawable/teruel</item>
  </array>
</resources>
```



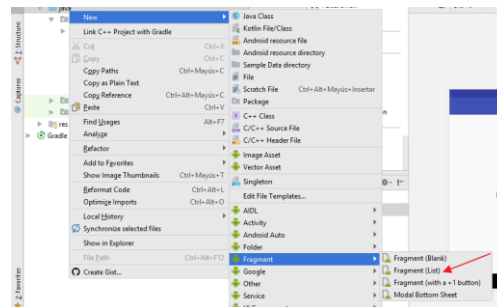


Creación del fragmento con la lista:

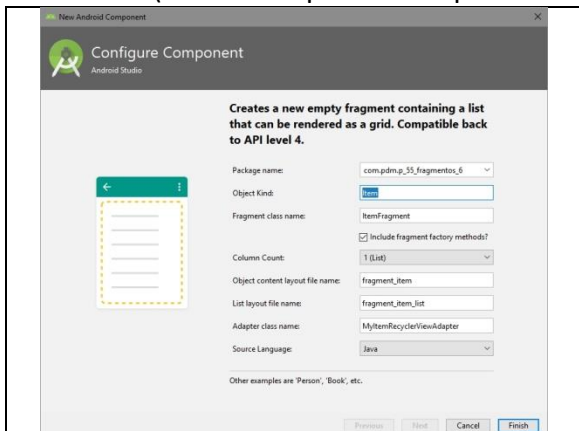
Como ya conocemos los principales conceptos del uso de fragmentos, haremos uso de la ayuda que nos proporciona el asistente de creación de fragmentos tanto para la creación de los layouts, los métodos de vida del fragmento y la declaración de la interfaz.

Importante: El asistente de creación de fragmentos aquí explicado es válido para **Android Studio 3.5.2**, nuevas versiones del IDE seguro que tienen asistentes distintos. Si más adelante actualizas AS el proyecto obviamente funcionará bien.

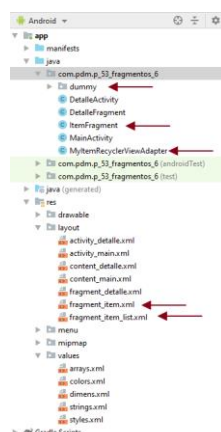
Esta "graciosa" característica de AS es la que me hace aconsejaros no utilizar siempre asistentes de creación y hacer las tareas "desde cero", de manera que sepas lo que haces sin necesidad de acudir a ayudas cambiantes.



El asistente además de ofrecernos la posibilidad de fijar el nombre de la clase y de añadir los métodos del ciclo de vida nos ofrece la posibilidad de cambiar el número de columnas (es decir si queremos tipo lista o tipo tabla).

	<p>Consejos:</p> <ol style="list-style-type: none">1. No cambies los nombres propuestos (Item, fragment_item, etc.) o la plantilla fallará (???)2. Puedes escoger el nº de columnas que tendrá el RecyclerView que generará la plantilla
---	---

Trabajo realizado por el asistente:





El asistente nos ha creado los dos layout necesarios: `fragment_item_list` y `fragment_item` que son respectivamente el listado de ítem y el contenido de cada ítem.

fragment_item_list	fragment_item
<p>El propio layout es un RecyclerView identificado como list.</p>	<p>El layout es un LinearLayout con dos TextView, uno identificado como item_number y el otro como content.</p> <p>Como solo queremos ver el nombre de la ciudad, podemos borrar uno de ellos (por ejemplo <code>item_number</code>). Observa los cambios en el otro layout!</p> <p>Recuerda que el alto del propio layout debe ser <code>?android:attr/listPreferredItemHeight</code> y que es conveniente que el ancho sea <code>match_parent</code>.</p>

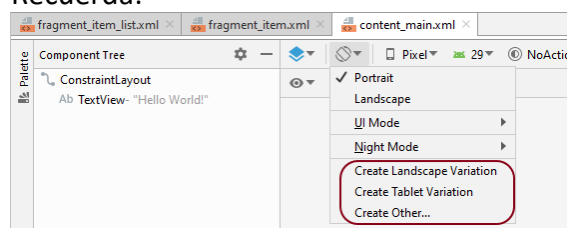
Modificaciones en layout content_main

Para trabajar con otros tipos de configuraciones y no siempre con horizontal/vertical, nos interesa:

- para dispositivos de tamaño normal (teléfono) y en orientación vertical: trabajar con dos actividades como lo hacíamos en el ejemplo anterior cada una con sus respectivos fragmentos.
- para dispositivos de tamaño grande y en orientación vertical: una actividad con dos fragmentos en vertical
- para dispositivos en orientación horizontal: una actividad con dos fragmentos en horizontal.

(Cómo crearemos las variaciones?

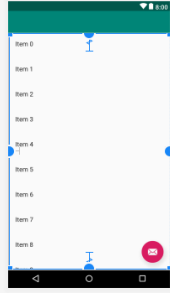

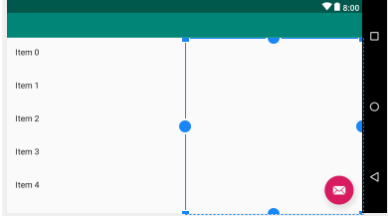
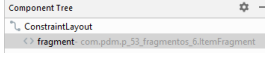
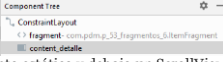

Recuerda:



Para crear la variación para tamaño grande y horizontal, escogeremos Other y en la siguiente pantalla fijaremos tamaño como `xlarge` y orientación como `landscape`).



En la configuración normal de content_main, eliminamos el TextView por defecto y añadimos el fragmento estático ItemFragment. Creamos la variación para layout large, para layout-land y la variación para layout-large-land (lo más importante es fijar bien el identificador del scrollview que debe ser igual que el del layout de la DetalleActivity, es decir **content_detalle**):

layout/content_main	large/content_main	land\content_main large-land\content_main
		
	 Fragmento estático y debajo un ScrollView (puedes borrar su contenido) identificado como content_detalle	 Fragmento estático y al lado un ScrollView (puedes borrar su contenido) identificado como content_detalle

Creación de la clase para el modelo de datos

Necesitamos una clase POJO con el modelo de datos que tiene cada ítem de la lista, la llamaremos, por ejemplo, Item (te aconsejo este nombre genérico en vez de por ejemplo Ciudad, para que cuando hagas "recorta y pega" en otros proyectos te pueda servir igual y no tengas que hacer muchas modificaciones!)

```
public class Item {
    String texto;
}
```

Con la herramienta "generate" añadimos el constructor y los "getters":

```
public class Item {
    String texto;

    public Item(String texto) {
        this.texto = texto;
    }

    public String getTexto() {
        return texto;
    }
}
```

Carpeta dummy

Observa que se ha creado una carpeta dummy, que contiene la clase DummyContent que sólo es un ejemplo de uso y que podemos borrar (a menos que nos dediquemos algo de tiempo al concepto de Map, que por ahora no es el caso!).



Modificaciones en la clase ItemFragment:

El asistente nos ha creado la clase que hereda de Fragment con sus métodos de vida básicos, la interfaz que responde al click en la list (OnListFragmentInteractionListener) y nos advierte con comentarios (`* Activities containing this fragment MUST implement the (@Link OnListFragmentInteractionListener)`, `// TODO: Customize parameter argument names`, etc.) los cambios que debemos realizar.

Cambiamos el parámetro del método de la interfaz (está al final de la clase) para que sea un entero que representa la posición del ítem seleccionado:

```
public interface OnListFragmentInteractionListener {  
    void onListFragmentInteraction(int id);  
}
```

El código del método onCreate() hace referencia al RecyclerView, ajustando su número de columnas y lo más importante su adaptador que tendremos que cambiar:

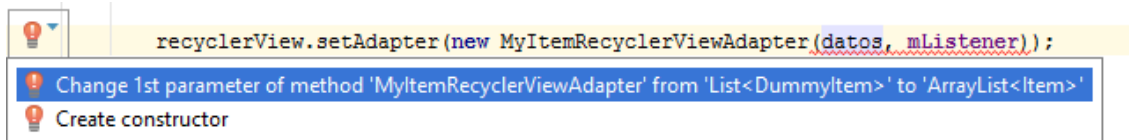
```
recyclerView.setAdapter(new MyItemRecyclerViewAdapter(DummyContent.ITEMS, mListener));
```

Necesitamos crear el adapter correcto con los datos del array de ciudades:

```
String[] ciudades = getResources().getStringArray(R.array.ciudades);  
ArrayList<Item> datos = new ArrayList<>();  
for(int i=0; i<3; i++)  
    datos.add(new Item(ciudades[i]));
```

```
recyclerView.setAdapter(new MyItemRecyclerViewAdapter(datos, mListener));
```

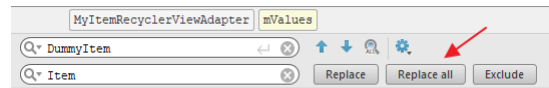
Y tal como nos marca AS modificar la clase correspondiente:





Modificaciones en la clase MyItemRecyclerViewAdapter

Hay que cambiar las referencias a los viejos DummyItem por nuestros Item, para más rapidez: Edit->Find->Replace:



Borraremos también las referencias al textview item_number que hemos borrado. Y en el manejador de la respuesta al click cambiaremos el parámetro.

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

    private final List<Item> mValues;
    private final OnListFragmentInteractionListener mListener;

    public MyItemRecyclerViewAdapter(ArrayList<Item> items,
OnListFragmentInteractionListener listener) {
        mValues = items;
        mListener = listener;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.fragment_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder holder, int position) {
        holder.mItem = mValues.get(position);
        holder.mContentView.setText(mValues.get(position).getTexto());

        holder.mView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (null != mListener) {
                    // Notify the active callbacks interface (the activity, if the
                    // fragment is attached to one) that an item has been selected.
                    mListener.onListFragmentInteraction(holder.getLayoutPosition());
                }
            }
        });
    }

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

    public class ViewHolder extends RecyclerView.ViewHolder {
        public final View mView;
        public final TextView mContentView;
        public Item mItem;

        public ViewHolder(View view) {
            super(view);
            mView = view;
            mContentView = (TextView) view.findViewById(R.id.content);
        }

        @Override
        public String toString() {
            return super.toString() + " '" + mContentView.getText() + "'";
        }
    }
}
```

(Observa que respondemos al evento en el método onBindViewHolder(), en el proyecto de RecyclerView anterior lo hacíamos en la clase ViewHolder; recuerda que ya comentamos que había varias maneras de responder. Esta es más lenta.)



Modificaciones en la clase MainActivity

Igual que el proyecto Fragmentos_05 hay que implementar el listener correctamente y cambiar el nombre de su método correspondiente. También debemos saber si hay 1 o 2 fragmentos para crear el fragmento o lanzar la actividad:

```
public class MainActivity extends AppCompatActivity implements
ItemFragment.OnListFragmentInteractionListener{

    private static final String ID_SELECCIONADO = "seleccionado";
    private boolean dosFragmentos;

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

        ...

        if (findViewById(R.id.content_detalle) != null)
            dosFragmentos=true;

    }

    ...

    @Override
    public void onListFragmentInteraction(int id) {
        Bundle bundle=new Bundle();
        bundle.putInt(ID_SELECCIONADO, id);
        if (dosFragmentos){
            DetalleFragment detalleFragment=new DetalleFragment();
            detalleFragment.setArguments(bundle);

            getSupportFragmentManager().beginTransaction().replace(R.id.content_detalle,
            detalleFragment).commit();

        }
        else{
            Intent intent=new Intent(this, DetalleActivity.class);
            intent.putExtras(bundle);
            startActivity(intent);
        }
    }
}
```

Modificaciones en la clase DetalleFragment

Los únicos cambios son:

- crear las variables globales necesarias:

```
private int seleccionado;
private static final String ID_SELECCIONADO = "seleccionado";
```

- obtener el entero pasado al crear el fragmento método onCreate:

```
@Override
public void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    Bundle recibidos=getArguments();
    seleccionado=recibidos.getInt(ID_SELECCIONADO);
}
```

- obtener el texto y la imagen usando los arrays creados en recursos en el método onCreateView():

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_detalle, container, false);
    TextView textView = view.findViewById(R.id.textView);

    textView.setText(getResources().getStringArray(R.array.descripciones)[seleccionado]);
    ImageView imageView = view.findViewById(R.id.imageView);

    imageView.setImageDrawable(getResources().obtainTypedArray(R.array.fotos).getDrawable(seleccionado));
    return view;
}
```



Modificaciones en la clase DetalleActivity

El código es muy similar al del proyecto Fragmentos_05:

```
public class DetalleActivity extends AppCompatActivity {
    private static final String ID_SELECCIONADO = "seleccionado";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getResources().getConfiguration().orientation ==
            Configuration.ORIENTATION_LANDSCAPE) {
            finish();
            return;
        }

        if ((getResources().getConfiguration().screenLayout &
            Configuration.SCREENLAYOUT_SIZE_MASK) == Configuration.SCREENLAYOUT_SIZE_LARGE) {
            finish();
            return;
        }

        setContentView(R.layout.activity_detalle);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action",
                    Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });

        Bundle bundle = getIntent().getExtras();
        int seleccionado = bundle.getInt(ID_SELECCIONADO);
        String texto = getResources().getStringArray(R.array.ciudades)[seleccionado];
        if (getSupportActionBar() != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setTitle(texto);
        }

        DetalleFragment detalleFragment = new DetalleFragment();
        detalleFragment.setArguments(bundle);
        getSupportFragmentManager().beginTransaction().replace(R.id.content_detalle,
            detalleFragment).commit();
    }
}
```

Explicación:

- Si la actividad se ha iniciado desde una configuración de orientación horizontal o de tamaño grande, se finaliza.
- Como su contenido es el DetalleFragmento, reemplazamos su layout por dicho fragmento.

Prueba la app.

Funciona pero podemos mejorarla, para que el ítem seleccionado sea más destacado en orientación horizontal.



Color de fondo en el ítem seleccionado.

Nuestras últimas modificaciones:

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

    private final List<Item> mValues;
    private final OnListFragmentInteractionListener mListener;
    //Posición inicial seleccionada ninguna
    private int posicionSeleccionada = -1;

    public MyItemRecyclerViewAdapter(ArrayList<Item> items,
OnListFragmentInteractionListener listener) {
        mValues = items;
        mListener = listener;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.fragment_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(final ViewHolder holder, int position) {
        holder.mItem = mValues.get(position);
        if (posicionSeleccionada == position) {
            holder.mView.setBackgroundColor(ContextCompat.getColor(holder.mView.getContext(),
R.color.colorAccent));
        } else {
            holder.mView.setBackgroundColor(Color.TRANSPARENT);
        }
        holder.mContentView.setText(mValues.get(position).getTexto());

        holder.mView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (null != mListener) {
                    /// Actualización de posiciones nuevas y antiguas
                    notifyItemChanged(posicionSeleccionada);
                    posicionSeleccionada = holder.getLayoutPosition();
                    notifyItemChanged(posicionSeleccionada);

                    mListener.onListFragmentInteraction(holder.getLayoutPosition());
                }
            }
        });
    }

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

    public class ViewHolder extends RecyclerView.ViewHolder {
        public final View mView;
        public final TextView mContentView;
        public Item mItem;

        public ViewHolder(View view) {
            super(view);
            mView = view;
            mContentView = (TextView) view.findViewById(R.id.content);
        }

        @Override
        public String toString() {
            return super.toString() + " '" + mContentView.getText() + "'";
        }
    }
}
```



Prueba la app y realiza varios giros: por ejemplo, partiendo de horizontal elige Teruel, vuelve a vertical y elige Huesca, vuelve a horizontal, no está seleccionada nada pero se ve el detalle de Teruel (que fue el último content_detalle en horizontal que se reemplazó por detalleFragmento).

Soluciones (elige la que prefieras):

1. La fácil (y por tanto, menos elegante): cada vez que se crea MainActivity después de un cambio de orientación, queremos que se olvide de los posibles fragmentos anteriores, la manera de conseguirlo es:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(null);
    ...
```

Es el proyecto P_53_Fragmentos_06.

2. La intermedia: Cada vez que se crea MainActivity después de un cambio de orientación, queremos que se visualice como escogida Zaragoza (la primera de la lista).

Los cambios en MainActivity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ...
    posicionEscogida=0;
    if (findViewById(R.id.content_detalle) != null) {
        dosFragmentos=true;
        Bundle bundle=new Bundle();
        bundle.putInt(ID_SELECCIONADO, posicionEscogida);
        DetalleFragment detalleFragment=new DetalleFragment();
        detalleFragment.setArguments(bundle);

        getSupportFragmentManager().beginTransaction().replace(R.id.content_detalle, detalleFragment).commit();
    }
}
```

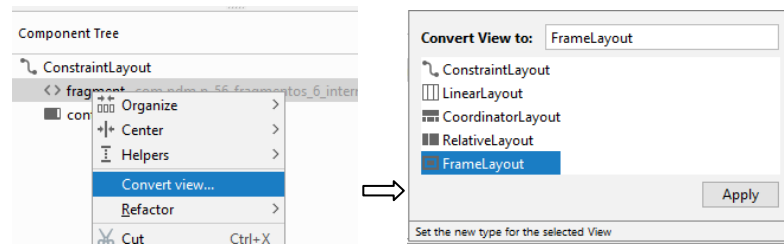
Los cambios en MyItemRecyclerViewAdapter:

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

    private final List<Item> mValues;
    private final OnListFragmentInteractionListener mListener;
    //Posición inicial seleccionada Zaragoza
    private int posicionSeleccionada=0 ;
    ...
```

Es el P_54_Fragmentos_06_intermedia

3. La mejor (obviamente, la que más trabajo nos va a dar ☺!): queremos que en MainActivity recuerde el fragmento del que venía. Debemos:
 - a. Cambiar los fragmentos estáticos por dinámicos pero que sigan teniendo como identificador fragment:
 - i. La manera más sencilla es convertir el fragmento estático a FrameLayot (recuerda que hay que hacerlo en cuatro layouts!):



- b. En MainActivity, sobrescribir el método onSaveInstanceState para guardar la posición escogida y en el método onCreate recuperarla haciendo uso de savedInstanceState y cambiar los 2 fragmentos según esa posición.

```
public class MainActivity extends AppCompatActivity implements ItemFragment.OnListFragmentInteractionListener{

    private static final String ID_SELECCIONADO = "seleccionado";
    private boolean dosFragmentos;
    private int posicionEscogida;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        ...

        if (savedInstanceState != null) {
            posicionEscogida = savedInstanceState.getInt(ID_SELECCIONADO);
        } else {
            posicionEscogida=0;
        }

        Bundle bundle=new Bundle();
        bundle.putInt(ID_SELECCIONADO, posicionEscogida);
        ItemFragment itemFragment=new ItemFragment();
        itemFragment.setArguments(bundle);
        getSupportFragmentManager().beginTransaction().replace(R.id.fragment, itemFragment).commit();

        if (findViewById(R.id.content_detalle)!=null) {
            dosFragmentos=true;
            DetalleFragment detalleFragment=new DetalleFragment();
            detalleFragment.setArguments(bundle);
            getSupportFragmentManager().beginTransaction().replace(R.id.content_detalle, detalleFragment).commit();
        }

        ...

        @Override
        public void onListFragmentInteraction(int id) {
            posicionEscogida=id;
            Bundle bundle=new Bundle();
            bundle.putInt(ID_SELECCIONADO, id);
            if (dosFragmentos) {
                DetalleFragment detalleFragment=new DetalleFragment();
                detalleFragment.setArguments(bundle);
                getSupportFragmentManager().beginTransaction().replace(R.id.content_detalle, detalleFragment).commit();
            }
            else {
                Intent intent=new Intent(this, DetalleActivity.class);
                intent.putExtras(bundle);
                startActivity(intent);
            }
        }

        @Override
        public void onSaveInstanceState(Bundle savedInstanceState) {
            super.onSaveInstanceState(savedInstanceState); // Always call the superclass first
            // Save values
            savedInstanceState.putInt(ID_SELECCIONADO, posicionEscogida);
        }
    }
}
```

- c. En ItemFragment, recuperar la posición en el método onCreate() y enviarla al adapter del recycler:

```
public ItemFragment() {
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    if (getArguments() != null) {
        posicionEnviada = getArguments().getInt(ID_SELECCIONADO);
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_item_list, container, false);

    if (view instanceof RecyclerView) {
        ...
        recyclerView.setAdapter(new MyItemRecyclerViewAdapter(datos, mListener, posicionEnviada));
    }
    return view;
}
```

