24. Fragmentos

Informalmente, puedes imaginar un fragmento como una subactividad, con su propia interfaz, comportamiento y ciclo de vida, que puedes reutilizar en distintas actividades. A pesar de su carácter modular, un fragmento siempre debe estar ligado a una actividad y su ciclo de vida depende del de la actividad a la que está ligado. Por ejemplo, cuando la actividad se destruye mediante una llamada a su método callback `onDestroy()`, el fragmento también se destruye.

Al romper el diseño de una actividad en fragmentos, su apariencia se puede modificar dinámicamente cuando sea necesario, sin la necesidad de cambios complejos de la jerarquía de vistas. Por ejemplo, un juego puede utilizar un fragmento (fragmento 1) para mostrar un menú de acciones a la izquierda y otro, fragmento 2, para mostrar la actividad correspondiente a la derecha. En una pantalla grande, ambos fragmentos pueden formar parte de la misma actividad, actividad 1. Sin embargo, en un teléfono más pequeño, necesitaremos dos actividades, una ligada al fragmento 1 del menú, y otra al 2, que se arrancará cada vez que pulsemos algún botón del fragmento 1.

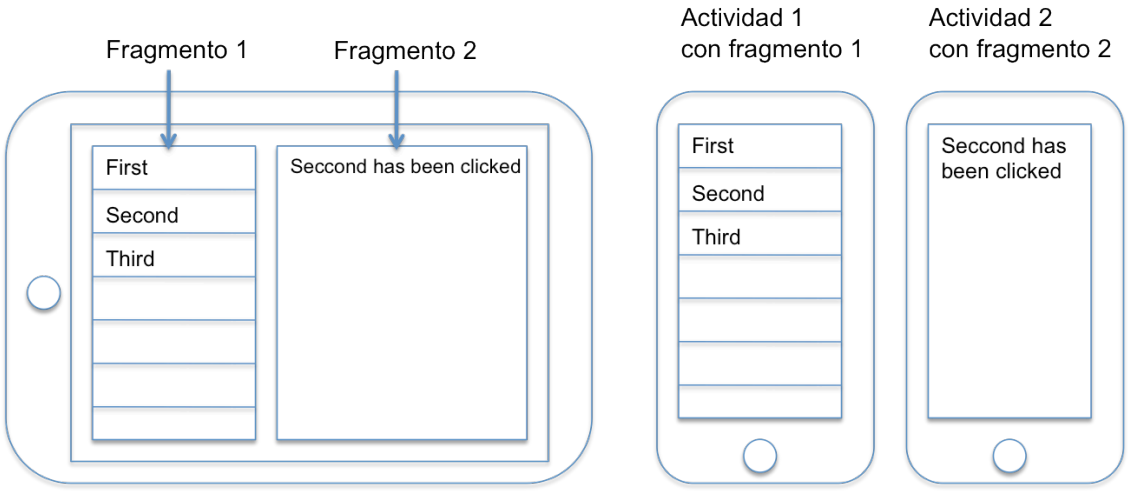


Figura 1. Una única actividad en una pantalla grande o dos en una pantalla más pequeña.

24.1 La biblioteca de apoyo

Android introdujo los fragmentos en Android 3.0 (API 11) principalmente para flexibilizar el diseño de interfaces gráficas en pantallas grandes como las de las tabletas. Sin embargo, puedes utilizarlos incluso aunque el nivel mínimo del API de tu aplicación sea inferior a 11. En este caso es necesario recurrir a la biblioteca de apoyo.

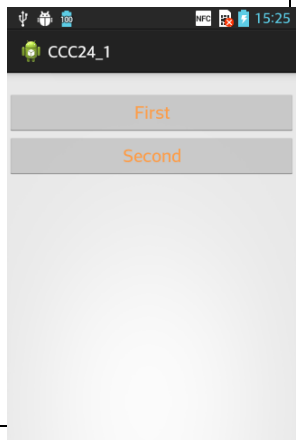
Esta biblioteca utiliza un paquete específico con algunos nombres ligeramente diferentes a los utilizados en la plataforma. Para instalar esta biblioteca debes arrancar el gestor del SDK, localizar la carpeta `Extras` al final del listado de paquetes, seleccionar `Android Support Library`, y pulsar `install package`. Debes asegurarte de no utilizar accidentalmente APIs nuevos en tu proyecto comprobando que tanto tus fragmentos como tus actividades provienen del paquete de apoyo:

```
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
```

Para ilustrar la utilización de la biblioteca de apoyo veamos un ejemplo, `CCC24_1`, con un nivel API mínimo igual a 8 y con un solo fragmento. Al igual que las actividades, los fragmentos necesitan de un fichero de diseño y otro Java. El fichero de diseño de nuestro primer fragmento, `fragment1.xml`, especifica dos botones llamados First y Second:

`/res/layout/fragment1.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/fragment1Button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textColor="#FF9E49"
        android:text="@string/first_button_text"
        android:textSize="20sp" />
    <Button
        android:id="@+id/fragment1Button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#FF9E49"
        android:text="@string/second_button_text"
        android:textSize="20sp" />
</LinearLayout>
```



Los recursos `first_button_text` y `second_button_text` están definidos en `strings.xml`:

`/res/values/strings.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">CCC24_1</string>
    ...
    <string name="first_button_text">First</string>
    <string name="second_button_text">Second</string>
</resources>
```

La clase del fragmento debe extender `Fragment` y, como mínimo, sobrescribir el método `onCreateView()`, que se llama cuando llega el momento de dibujar el layout del fragmento. Este método debe devolver a la actividad asociada una vista que será la raiz del diseño del fragmento. Solo existe una excepción: cuando el fragmento extiende `ListFragment`, la implementación por defecto devuelve un

objeto de tipo `ListView` y no es necesario implementar `onCreateView()`. También existe un método `onCreate()` pero, a diferencia de las actividades, no se utiliza para inflar la interfaz sino para configurar el fragmento.

Para implementar el ejemplo de esta sección, asegúrate de incluir la clase `Fragment` de la biblioteca de apoyo:

/src/Fragment1.java

```
package es.uam.eps.android.CCC24_1;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment1 extends Fragment {
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
                             savedInstanceState){
        View view = inflater.inflate(R.layout.fragment1, container);
        return view;
    }
}
```

Como hemos mencionado anteriormente, el método `onCreateView()` devuelve la vista correspondiente al fragmento. El método `inflate()` de `LayoutInflater` acepta dos argumentos: el identificador del diseño del fragmento que se desea inflar (`R.layout.fragment1`) y la vista en la que se insertará el fragmento, argumento pasado por `onCreateView()`. El método `onCreateView()`, como el resto de los del ciclo de vida de los fragmentos, es público para poder ser invocado por cualquier actividad a la que se ligue el fragmento. Esto les diferencia de los métodos del ciclo de vida de la actividad.

Si bien los fragmentos pueden ser utilizados por varias actividades, siempre deben ligarse a una de ellas. Un fragmento no puede por si solo colocar una vista en la pantalla si previamente no se le ha asignado una zona de la jerarquía de vistas de una actividad. Para ligar un fragmento a una actividad podemos utilizar el fichero XML de diseño de la actividad, o bien, código Java, como veremos más adelante. En nuestro sencillo ejemplo, `MainActivity` incluye el fragmento `Fragment1` en su fichero de diseño mediante el atributo `class` de su elemento `<fragment>`:

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:baselineAligned="false">
    <fragment
        android:id="@+id/fragment1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="es.uam.eps.android.CCC24_1.Fragment1" />
</LinearLayout>
```

Cuando el sistema infla este diseño, instancia el fragmento especificado, llama a su método `onCreateView()` e inserta la vista devuelta en el lugar indicado por el elemento `<fragment>` del archivo de diseño de la actividad. Asegúrate de extender `FragmentActivity` en lugar de `Activity`, cuando utilices la biblioteca de apoyo, pues las actividades anteriores al API 11 no saben gestionar fragmentos:

/src/MainActivity.java

```
package es.uam.eps.android.CCC24_1;

import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

24.2 Añadiendo fragmentos a una actividad en su fichero XML

El ejemplo de esta sección utiliza dos fragmentos y prescinde de la biblioteca de apoyo. El primer fragmento, el de los dos botones, va a estar ligado a la actividad principal, y el segundo se va a ligar a una nueva actividad llamada `Detail`.

El fichero de diseño del primer fragmento, `fragment1.xml`, es el mismo que el de la sección anterior, es decir, especifica dos botones: First y Second. También reutilizaremos el fichero `Fragment1.java`, esta vez utilizando la clase `Fragment` original en lugar de la clase de la biblioteca de apoyo:

/src/Fragment1.java

```
package es.uam.eps.android.CCC24_2;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment1 extends Fragment {
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
                             savedInstanceState){
        View view = inflater.inflate(R.layout.fragment1, container);
        return view;
    }
}
```

El fichero de diseño del segundo fragmento, `Fragment2`, cuenta con un elemento de tipo `TextView` sobre un fondo de color naranja:

`/res/layout/fragment2.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FF9E49"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/fragment2TextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:textSize="20sp" />
</LinearLayout>
```

El fichero `Fragment2.java` es equivalente al del otro fragmento:

`src/Fragment2.java`

```
package es.uam.eps.android.CCC24_2;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment2, container);
        return view;
    }
}
```

La actividad `MainActivity` incluye el fragmento `Fragment1` en su fichero de diseño mediante el atributo `class`:

`/res/layout/activity_main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:baselineAligned="false">
    <fragment
        android:id="@+id/fragment1"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        class="es.uam.eps.android.CCC24_2.Fragment1" />
</LinearLayout>
```

El fichero de diseño de la segunda actividad, `Detail`, cuenta con un elemento `fragment` para el segundo fragmento:

/res/layout/detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:baselineAligned="false"
    android:orientation="horizontal" >
    <fragment
        android:id="@+id/fragment2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class="es.uam.eps.android.CCC24_2.Fragment2" />

</LinearLayout>
```

La actividad `MainActivity` infla el fichero `activity_main.xml` y extiende la clase `Activity` en lugar de la clase `FragmentActivity`. Además, implementa `OnClickListener` para dar funcionalidad a los dos botones del fragmento que aloja. Al pulsarlos se arranca la actividad `Detail` a la que además se pasa un mensaje indicando qué botón se ha pulsado:

/src/MainActivity.java

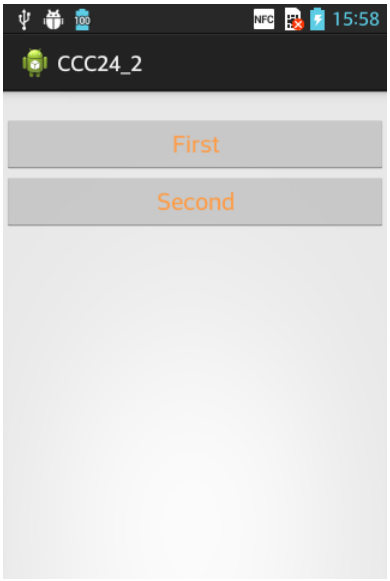
```
package es.uam.eps.android.CCC24_2;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.app.Activity;
import android.content.Intent;

public class MainActivity extends Activity implements OnClickListener{
    Fragment1 fragment1;
    Fragment2 fragment2;
    TextView textView;

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

         Button button1 = (Button) findViewById(R.id.fragment1Button1);
         Button button2 = (Button) findViewById(R.id.fragment1Button2);

         button1.setOnClickListener(this);
         button2.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        String str;

        if (v.getId() == R.id.fragment1Button1)
            str = "Fist button clicked";
        else
            str = "Second button clicked";

         Intent intent = new Intent (getApplicationContext(), Detail.class);
         intent.putExtra("message", str);
         startActivity(intent);
    }
}
```

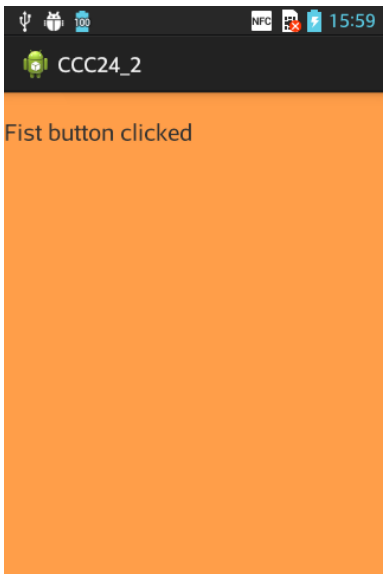
Finalmente, la actividad `Detail`, después de inflar su fichero de diseño, extrae el texto con etiqueta `message` del objeto de tipo `Intent` creado en `MainActivity` y, si no es nulo, lo muestra en la vista de tipo `TextView` de su fragmento:

`/src/Detail.java`

```
package es.uam.eps.android.CCC24_2;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Detail extends Activity {
    public void onCreate (Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        setContentView(R.layout.detail);
        Bundle bundle = getIntent().getExtras();
        if (bundle != null){
            String str = bundle.getString("message");
            TextView textView = (TextView) findViewById(R.id.fragment2TextView);
            textView.setText(str);
        }
    }
}
```

No olvides añadir un elemento `activity` para la actividad `Detail` en el fichero de manifiesto. El resultado es el siguiente antes y después de pulsar el botón First:





24.3 Añadiendo fragmentos a una actividad durante su ejecución

Los fragmentos especificados en el fichero XML de la actividad no se pueden eliminar durante la ejecución de la app. Para crear interfaces dinámicas de este tipo es necesario añadir los fragmentos dinámicamente. En esta sección vamos a generar una interfaz similar a la de la sección anterior creando los fragmentos con código Java.

Podemos reutilizar sin cambios los ficheros XML de los dos fragmentos de la sección anterior. En los ficheros Java debemos añadir un tercer argumento al método `inflate()`, que indica si la vista inflada debe añadirse al padre (`true`) o no (`false`). Como vamos a hacerlo desde el código Java, debemos pasar `false` como tercer argumento. Por ejemplo, el primer fragmento queda así:

/src/Fragment1.java

```
package es.uam.eps.android.CCC24_3;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class Fragment1 extends Fragment {
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
                             savedInstanceState){
        View view = inflater.inflate(R.layout.fragment1, container, false);
        return view;
    }
}
```

El fichero de diseño de la clase `MainActivity` ya no especifica el elemento de tipo `fragment`, sino simplemente un elemento de tipo `LinearLayout` vacío:

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:baselineAligned="false">
    <LinearLayout
        android:id="@+id/fragment1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"/>
</LinearLayout>
```

El método `onCreate()` de `MainActivity` va a ser el encargado de crear el fragmento. Primero conseguimos una referencia al gestor de fragmentos que Android añadió a las actividades en el API 11 y que, entre otras cosas, mantiene:

- una lista de fragmentos identificados por los recursos id de sus contenedores: `R.id.fragment1` y `R.id.fragment2` en nuestra app.
- una pila de fragmentos (*back stack*) ligados a la actividad.

Para acceder al gestor llamamos al método `getFragmentManager()` o, si utilizas la biblioteca de apoyo, `getSupportFragmentManager()`:

```
FragmentManager fm = getFragmentManager();
```

Y a partir del gestor, conseguimos una referencia a `FragmentTransaction`:

```
FragmentTransaction ft = fragmentManager.beginTransaction();
```

A continuación se instancia el fragmento y se añade mediante el método `add()`, que tiene dos argumentos: el identificador del contenedor del fragmento (`R.id.fragment1`) y el fragmento que se desea añadir (`fragment1`):

```
Fragment1 fragment1 = new Fragment1();
ft.add(R.id.fragment1, fragment1);
```

Para que los cambios tengan lugar es necesario ejecutar el método `commit()` de la transacción:

```
ft.commit();
```

Antes de añadir los fragmentos conviene comprobar si ya se encuentran en la pila. Cuando la actividad se destruye, por rotación por ejemplo, la lista de fragmentos se guarda para ser recreados cuando se recree la actividad. Esta es una razón por la que ya podríamos tener a los fragmentos en la pila. Por lo tanto, para evitar instanciar fragmentos innecesariamente, utilizaremos el siguiente código:

```
FragmentManager fm = getFragmentManager();
if (fm.findFragmentById(R.id.fragment1) == null) {
    Fragment1 fragment1 = new Fragment1();
    fm.beginTransaction().add(R.id.fragment1, fragment1).commit();
}
```

Hemos utilizado el hecho de que el método `add()` devuelve una referencia a la transacción. El fichero `MainActivity.java` es el siguiente:

/src/MainActivity.java

```
package es.uam.eps.android.CCC24_3;

import android.app.Activity;
import android.app.FragmentManager;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MainActivity extends Activity implements OnClickListener{

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

        FragmentManager fm = getFragmentManager();
        if (fm.findFragmentById(R.id.fragment1) == null) {
            Fragment1 fragment1 = new Fragment1();
            fm.beginTransaction().add(R.id.fragment1, fragment1).commit();
        }
    }

    @Override
    protected void onStart(){
        super.onStart();

        Fragment1 fragment = (Fragment1)
            getFragmentManager().findFragmentById(R.id.fragment1);

        Button button1 = (Button)
            fragment.getView().findViewById(R.id.fragment1Button1);
        Button button2 = (Button)
            fragment.getView().findViewById(R.id.fragment1Button2);
```

```
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        String str;

        if (v.getId() == R.id.fragment1Button1)
            str = "Fist button clicked";
        else
            str = "Second button clicked";

        Intent intent = new Intent ("es.uam.eps.android.CCC24_3.DETAIL");
        intent.putExtra("message", str);
        startActivity(intent);
    }
}
```

En el método `onStart()` es donde se captura una referencia al primer fragmento mediante el método `findFragmentById()`:

```
Fragment1 fragment = (Fragment1) getFragmentManager().findFragmentById(R.id.fragment1);
```

A continuación se recuperan sendas referencias a sus dos botones, para registrar la actividad como escuchador. Fíjate en que los fragmentos no cuentan con un método `findViewById()` sino que debemos utilizar el de la clase `View`. Para ello primero recuperamos la vista del fragmento con el método `getView()`. También podemos acceder a la actividad ligada al fragmento mediante el método `getActivity()` y así, por ejemplo, acceder a una vista determinada de su diseño:

```
TextView textView = (TextView) getActivity().findViewById(R.id.textView);
```

Finalmente, el método `onClick()` lleva a cabo la misma tarea que el método `onClick()` del ejemplo anterior, es decir, arrancar la actividad `Detail`.

La actividad `Detail`, por su parte, crea el fragmento número 2 como sigue:

`/src/Detail.java`

```
package es.uam.eps.android.CCC24_3;

import android.app.Activity;
import android.app.FragmentManager;
import android.os.Bundle;
import android.widget.TextView;

public class Detail extends Activity {
    public void onCreate (Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.detail);

        FragmentManager fm = getFragmentManager();
        if (fm.findFragmentById(R.id.fragment2) == null) {
            Fragment2 fragment2 = new Fragment2();
            fm.beginTransaction().add(R.id.fragment2, fragment2).commit();
        }
    }

    @Override
    protected void onStart (){
        super.onStart();

        Bundle bundle = getIntent().getExtras();
        if (bundle != null){
```

```
            String str = bundle.getString("message");
            TextView textView = (TextView) findViewById(R.id.fragment2TextView);
            textView.setText(str);
        }
    }
}
```

Además del método `add()`, también podemos utilizar los métodos `replace()` y `remove()`. El método `addToBackStack()` permite añadir la transacción a la pila de transacciones (*back stack*). La actividad, a través del gestor de fragmentos, gestiona esta pila que permite al usuario volver a estados anteriores pulsando el botón back del dispositivo. Los fragmentos de las transacciones de la *back stack* se detienen y luego reanudan al pulsar el botón back, en lugar de destruirse directamente.

24.4 Comunicación con la actividad a través de una interfaz

Veamos una forma de conseguir el mismo resultado de la sección anterior sin que los fragmentos se comuniquen directamente, manteniendo así su independencia. El proyecto `CCC24_4` comienza modificando la clase `Fragment2.java` al añadir el método `showText()`, que se encarga de mostrar la cadena pasada como argumento en el `TextView` del segundo fragmento:

`/src/Fragment2.java`

```
package es.uam.eps.android.CCC24_4;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

public class Fragment2 extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment2, container, false);
        return view;
    }

    public void showText(String text) {
        TextView view = (TextView) getView().findViewById(R.id.fragment2TextView);
        view.setText(text);
    }
}
```

El primer fragmento define la interfaz `onButtonSelectedListener`

```
public interface OnButtonSelectedListener {
    public void onButtonSelected(String link);
}
```

La interfaz contiene un único método, `onButtonSelected()`, que tendrá que implementar la actividad en la que se incruste el fragmento. Esta es una solución que evita que los fragmentos se comuniquen directamente, manteniendo así su independencia:

/src/Fragment1.java

```
package es.uam.eps.android.CCC24_4;

import android.app.Activity;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.Button;

public class Fragment1 extends Fragment {
    private OnButtonSelectedListener listener;

    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
                             savedInstanceState){
        View view = inflater.inflate(R.layout.fragment1, container, false);

        Button button1 = (Button)view.findViewById(R.id.fragment1Button1);
        Button button2 = (Button)view.findViewById(R.id.fragment1Button2);

        button1.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                listener.onButtonSelected("First button clicked");

            }
        });

        button2.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                listener.onButtonSelected("Second button clicked");

            }
        });

        return view;
    }

    public interface OnButtonSelectedListener {
        public void onButtonSelected (String str);
    }

    @Override
    public void onAttach (Activity activity){
        super.onAttach(activity);
        if (activity instanceof OnButtonSelectedListener)
            listener = (OnButtonSelectedListener) activity;
        else {
           throw new ClassCastException(activity.toString() +
                       " does not implement OnButtonSelectedListener");
        }
    }

    @Override
    public void onDetach(){
        super.onDetach();
        listener = null;
    }
}
```

Asignamos la actividad como escuchador en el método que se ejecuta cuando el fragmento se liga a la actividad: `onAttach()`. Este método lanza una excepción si la actividad pasada como argumento no implementa la interfaz. En el método `onDettach()` se asigna el escuchador a `null`.

La clase `MainActivity.java` se encarga de crear el primer fragmento con una transacción y de implementar la interfaz definida en el primer fragmento, es decir, implementar el método `onButtonSelected()`. Este método arranca la actividad `Detail` con un extra de clave `message`:

/src/MainActivity.java

```
package es.uam.eps.android.CCC24_4;

import android.app.Activity;
import android.app.FragmentManager;
import android.content.Intent;
import android.os.Bundle;

import es.uam.eps.android.CCC24_4.Fragment1.OnButtonSelectedListener;

public class MainActivity extends Activity implements OnButtonSelectedListener {

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

        FragmentManager fm = getFragmentManager();
        if (fm.findFragmentById(R.id.fragment1) == null) {
            Fragment1 fragment1 = new Fragment1();
            fm.beginTransaction().add(R.id.fragment1, fragment1).commit();
        }
    }

    @Override
    public void onButtonSelected(String str) {
        Intent intent = new Intent(getApplicationContext(), Detail.class);
        intent.putExtra("message", str);
        startActivity(intent);
    }
}
```

La actividad `Detail` crea el fragmento número 2 y, en el método `onStart()`, captura una referencia al fragmento y pasa al método `showText()` el mensaje extraído del objeto de tipo `Intent`:

/src/Detail.java

```
package es.uam.eps.android.CCC24_4;

import android.app.Activity;
import android.app.FragmentManager;
import android.os.Bundle;

public class Detail extends Activity {
    public void onCreate (Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.detail);

        FragmentManager fm = getFragmentManager();
        if (fm.findFragmentById(R.id.fragment2) == null) {
            Fragment2 fragment2 = new Fragment2();
            fm.beginTransaction().add(R.id.fragment2, fragment2).commit();
        }
    }

    @Override
    protected void onStart (){
        super.onStart();

        Bundle bundle = getIntent().getExtras();
```

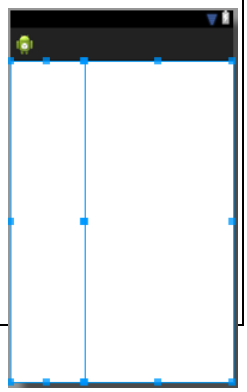
```
        if (bundle != null){
             String str = bundle.getString("message");
             Fragment2 fragment2 = (Fragment2)
                                   getFragmentManager().findFragmentById(R.id.fragment2);
             fragment2.showText(str);
        }
    }
}
```

24.5 Diseño para tabletas

Hasta ahora, nuestra app muestra cada fragmento en una actividad diferente: `MainActivity` y `Detail`. En una tableta ambos fragmentos se pueden visualizar simultáneamente ligándolos a la actividad principal. Vamos a crear un último proyecto, `CCC24_5`, con un fichero de diseño con dos contenedores para fragmentos:

`/res/layout/activity_tablet.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal"
    android:baselineAligned="false">
    <LinearLayout
        android:id="@+id/fragment1"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:orientation="vertical" />
    <LinearLayout
        android:id="@+id/fragment2"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:orientation="vertical" />
</LinearLayout>
```



Además, añadiremos un recurso alias de nombre `activity_master_detail` y tipo `layout` que inflará la actividad principal. Un recurso alias es un recurso que apunta a otro recurso. En este caso el alias apuntará a `activity_main.xml` en teléfonos o `activity_tablet.xml` en tabletas. Los recursos alias se colocan en la carpeta `/res/values/` dentro del fichero `refs.xml`:

`/res/values/refs.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_master_detail" type="layout">@layout/activity_main</item>
</resources>
```

Para que el recurso apunte a `activity_tablet` en tabletas, crearemos una versión alternativa de `refs.xml` en la carpeta `/res/values-sw600dp`:

/res/values-sw600dp/refs.xml

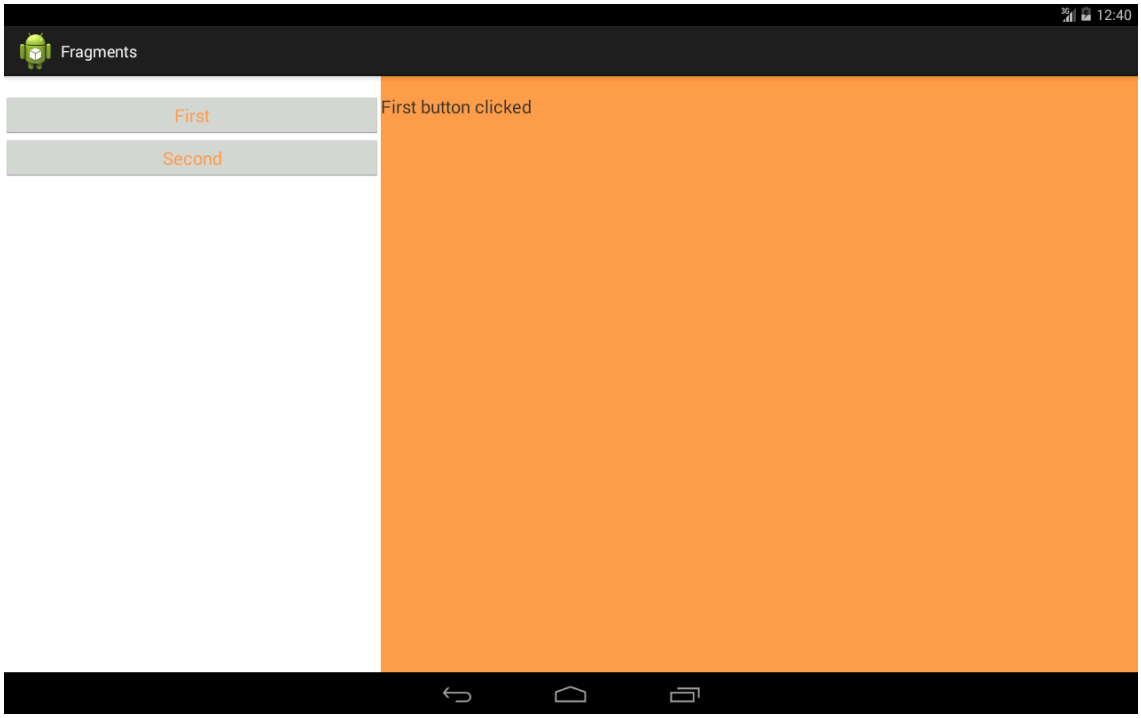
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_master_detail" type="layout">@layout/activity_tablet</item>
</resources>
```

La actividad principal infla el fichero apuntado por el recurso alias de nombre `activity_master_detail.xml` y, además, antes de crear el fragmento de tipo `Fragment2`, comprueba que la anchura en dps es superior a 600, de tal forma que solo se instancie en tabletas:

/src/MainActivity.java

```
package es.uam.eps.android.CCC24_5;

import android.app.Activity;
import android.app.FragmentManager;
import android.content.Intent;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.Display;

import es.uam.eps.android.CCC24_5.Fragment1.OnButtonSelectedListener;

public class MainActivity extends Activity implements OnButtonSelectedListener {

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

        Display display = getWindowManager().getDefaultDisplay();
        DisplayMetrics outMetrics = new DisplayMetrics ();
        display.getMetrics(outMetrics);
        float density  = getResources().getDisplayMetrics().density;
        float dpWidth  = outMetrics.widthPixels / density;

        FragmentManager fm = getFragmentManager();
        if (fm.findFragmentById(R.id.fragment1) == null) {
            Fragment1 fragment1 = new Fragment1();
            fm.beginTransaction().add(R.id.fragment1, fragment1).commit();
        }

        if (dpWidth > 600){
            if (fm.findFragmentById(R.id.fragment2) == null) {
                Fragment2 fragment2 = new Fragment2();
                fm.beginTransaction().add(R.id.fragment2, fragment2).commit();
            }
        }
    }

    @Override
    public void onButtonSelected(String str) {
        Fragment2 fragment = (Fragment2) getFragmentManager().findFragmentById(
                        R.id.fragment2);
        if (fragment != null) {
            fragment.showText(str);
        } else {
            Intent intent = new Intent(getApplicationContext(), Detail.class);
            intent.putExtra("message", str);
            startActivity(intent);
        }
    }
}
```

Ahora, este es el resultado en una tableta:


12:40
Fragments
First
Second
First button clicked