FRAGMENTOS DINÁMICOS

Este ejemplo demuestra como tener un actividad en la cargar un fragmento u otro en función de alguna acción. En otros ejemplos en la Moodle vimos cómo cambiar según se pulse un botón u otro. En este caso se hará pulsando el botón flotante que se genera al usar un Basic Activity.

Un Basic Activity viene con un menú y un botón flotante, o Tab.

Veamos cómo desarrollar este proyecto paso a paso:

Paso 1:

En el MainActivity.java, los métodos referentes al menú vamos a borrarlos ya que no vamos a usarlos en esta aplicación.

```
public boolean onCreateOptionsMenu(Menu menu) {
public boolean onOptionsItemSelected(MenuItem item) {
```

Fíjense que en el onCreate aparecen definidos:

- el Toolbar: en el resto de aplicaciones aparecían definidos directamente en la interface de usuario y no en el código.
- y el FloatingActionButoon: botón flotante que por defecto están en la parte inferior derecha.
- La implementación gráfica de ambos se encuentra en la carpeta Layout

Paso 2: implementar modificaciones en la interfaz de usuario

En este caso vemos que se han creado en la carpeta res/layout dos archivos, activity_main.xml y content_main.xml. El primero define la pantalla principal en tres elementos, el Toolbar (o cabecera principal), el botón flotante y en el centro del código podemos ver que incluye el código

```
<include layout="@layout/content_main" />
```

que representa la funcionalidad de esta pantalla. En nuestro caso vamos a implementar este content_main.xml como un FrameLayout donde se visualizarán o cargaran los fragmentos que implementemos. Para ello, eliminamos el TextView y definir el contenedor como un FrameLayout y asignarle, muy importante, un nombre para poder referenciarlo a la hora de cargarle un fragmento **android:id="@+id/container"**

Podemos modificar el botón flotante:

1. La imagen: descargo una nueva imagen: new→ Vector Assert y selecciona la que me interese.
   La añado en el archivo activity_main:

   ```
   app:srcCompat="@drawable/ic_refresh_white_24dp" />
   ```

   Para modificiarle el color, directamente abro el archivo que la contiene y modifico dicho color
2. Le modifico el color de fondo: por defecto tiene asignado el **colorAccent, a**sí que acceder al archivo color.xml y modificarlo por el que queramos

Paso 3: crear dos fragmentos, y a cada uno de ellos cambiarles el mensaje de texto y añadir a las propiedades del fragmento un color diferente. Recuerda desmarcar las dos casillas al crear los fragmentos ya que no vamos a hacer implementaciones con ellos y así evitamos código de más que no vamos a utilizar.

```
android:background="@android:color/holo_green_light"
```

Paso 4: comenzamos con la implementación en el MainActivity.java

- Rescatamos el contenedor y le asignamos uno de los fragmentos por defecto

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

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener((view) → {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGT
                        .setAction("Action", null).show();
        });

        // Rescatamos el contenedor y le vamos a cargar un fragmento

        getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.container,new PrimerFragment())
                .commit();


    }
```

- Vamos a hacer que al hacer click en el botón flotante, por defecto está cargado el primer fragmento, pues al pulsarlo lo cambiará por el segundo.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    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) {
            // Modificar el Fragmento que sale en la pantalla
            // dentro del container
            getSupportFragmentManager()
                    .beginTransaction()
                    .replace(R.id.container,new SegundoFragment())
                    .commit();
        }
    });

    // Rescatamos el contenedor y le vamos a cargar un fragmento
    getSupportFragmentManager()
            .beginTransaction()
            .add(R.id.container,new PrimerFragment())
```

- Para que cada vez que al clickear vaya modificando por el fragmento que en ese momento no esté cargado haremos los siguientes cambios.
  Declaro en la clase una variable booleana

```
boolean cargarFragmento2 = true;
```

  Implementar un condicional en el click para que se cambie entre frragmentos:

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Modificar el Fragmento que sale en la pantalla
        // dentro del container

        if(cargarFragmento2) {
            getSupportFragmentManager()
                    .beginTransaction()
                    .replace(R.id.container, new SegundoFragment())
                    .commit();
        } else {
            getSupportFragmentManager()
                    .beginTransaction()
                    .replace(R.id.container, new PrimerFragment())
                    .commit();
        }

        cargarFragmento2 = !cargarFragmento2;

    }
```

- Y el código optimizado quedaría como en el ejemplo implementado en la Moodle.

Esto parece muy teórico pero veremos que esto tiene una importancia y una utilidad cuando se trabaje con Navegación por menús, pestañas, navigation drawer, … ya que son usados estos fragmentos.