# Práctica #1

Creación del proyecto "Empty activity"

## Pantalla completa

Añadir la anotación "@Fullscreen"

Construir app

Se genera un error: debido a que el manifest no está la ".MainActivity_"

Generar nuevamente

Se genera un error: no se ha puesto la anotación "@EActivity". Ésta es esencial para la generación de AndroidAnnotations

Generar nuevamente y desplegar en el emulador

Se mostrará la app a pantalla completa pero con el títtulo de la app

## Extra - Pantalla completa

Si queremos quitar el título de la app deberemos añadir la anotación

```
@WindowFeature({ Window.FEATURE_NO_TITLE })
Adicionalmente si quieres añadir el gesto para salir de la app
    @WindowFeature({ Window.FEATURE_NO_TITLE, Window.FEATURE_SWIPE_T

    // En mi caso he añadido he añadido, desde código andoid la ocul
    private void hideNavigationBar() {
       View decorView = getWindow().getDecorView();
       int uiOptions = View.SYSTEM_UI_FLAG_HIDE_NAVIGATION | View.SY
       decorView.setSystemUiVisibility(uiOptions);
    }
```

## Objeto de UI personalizado

El objeto personalizado se podría hacer de dos maneras diferentes, como se ha explicado en la clase "Anotación EView" es decir poner la anotación en "@EView" en la respectiva clase y crear el respectivo objeto de UI, por lo que habría que hacerlo tanto para el Text View como para el Switch.

Otra posibilidad es creando la anotación "@EViewGroup" y posteriormente hacer un objecto personalizado que contenga tanto el text view como el switch, nos decantaremos por este segundo, ya que el EView ya lo hemos visto.

### (1) Crear nuevo laout: custom_marvel_element_list.xml

Se deberá crear un nuevo layout en en los resources. Por ejemplo "custom_marvel_element_list.xml". El contenedor de los objetos de UI será un "merge":

https://developer.android.com/training/improving-layouts/reusing-layouts#Merge

The tag helps eliminate redundant view groups in your view hierarchy when including one layout within another. For example, if your main layout is a vertical LinearLayout in which two consecutive views can be re-used in multiple layouts, then the re-usable layout in which you place the two views requires its own root view. However, using another LinearLayout as the root for the re-usable layout would result in a vertical LinearLayout inside a vertical LinearLayout. The nested LinearLayout serves no real purpose other than to slow down your UI performance. A continuación se deberá crear el Text View, en mi caso yo he creado dos, uno para el título del objeto personalizado y otro para la descripción.

**(2) Crear la clase con la anotación "@EViewGroup": MarvelElementList**

https://github.com/androidannotations/androidannotations/wiki/Enhance-custom-views#custom-viewgroups-with-eviewgroup

Ahí se deberán crear los respectivos métodos para inicialización y para la lógica de negocio de la app. Por ejemplo, para hacer los sets de las labels definiremos el método "setTexts" que inicializa los Text View.

Para instanciar hemos empleado el "@ViewById", esta anotación se explicará más adelante, no obstante se emplea para instanciar objetos de UI).

La anotación "@EViewGroup" deberá hacer referencia al layout que hemos creado en el punto anterior ("@EViewGroup(R.layout.custom_marvel_element_list)").

**(3) Invocación al custom object en la UI: activity_main.xml**

Abriremos "activity_main.xml" y añadiremos el objeto de UI que definimos en la clase que contiene la anotación "@EViewGroup". Definir los márgenes, tamaños...

```
<com.example.openwebinar.fullscreen.views.MarvelElementList_
    android:id="@+id/firstMarvelElementList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.09"
    android:layout_margin="15dp" />
```

**(4) Instanciación en la clase Java: MainActivity**

Nuevamente emplearemos la anotación "@ViewById" (proximamente la explicaremos).

Posteriormente se deberá inicializar los objetos empleando el método "setTexts" anteriormente mencionado.

## Completar el ejercicio

### string.xml

A continuación se deberá añadir el siguiente texto en el string.xml

```
<string name="mavel_avengers">SÚPER HEROES</string>
```

### activity_main.xml

Ahora en el layout principal añadiremos un string a modo de título, por ejemplo:

```
<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="10dp"
android:layout_marginTop="5dp"
android:text="@string/mavel_avengers"
android:textSize="45sp"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

Se puede poner tantos objetos personalizados como se desee. Si se ponen más habrá que instanciar e inicializar en la "MainActivity".

Ejecutar la app en el emulador.