

ESCUELA SUPERIOR DE INGENIERÍA

Programación en Internet

Grado en Ingeniería Informática

Aplicación Android

Autores:

José Manuel Periñán Freire

Iñaki Urrutia Sanchez

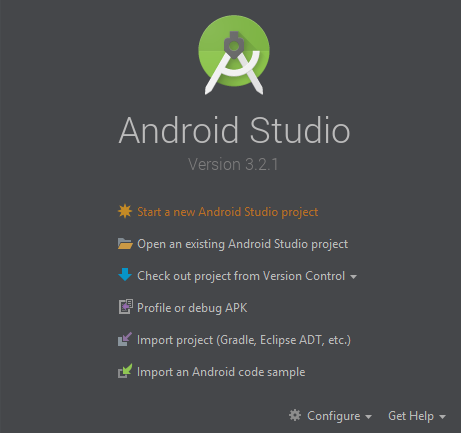
Supervisores:

David Corral Plaza

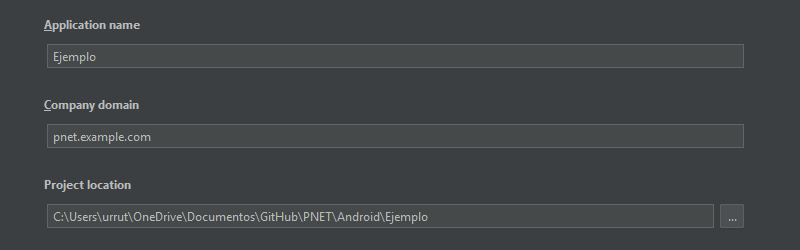
Cádiz, 14 de diciembre de 2019

1. **Creación del proyecto**

Para crear el proyecto de nuestra aplicación, abrimos Android Studio, y damos click en “Start a new Android Studio project”.



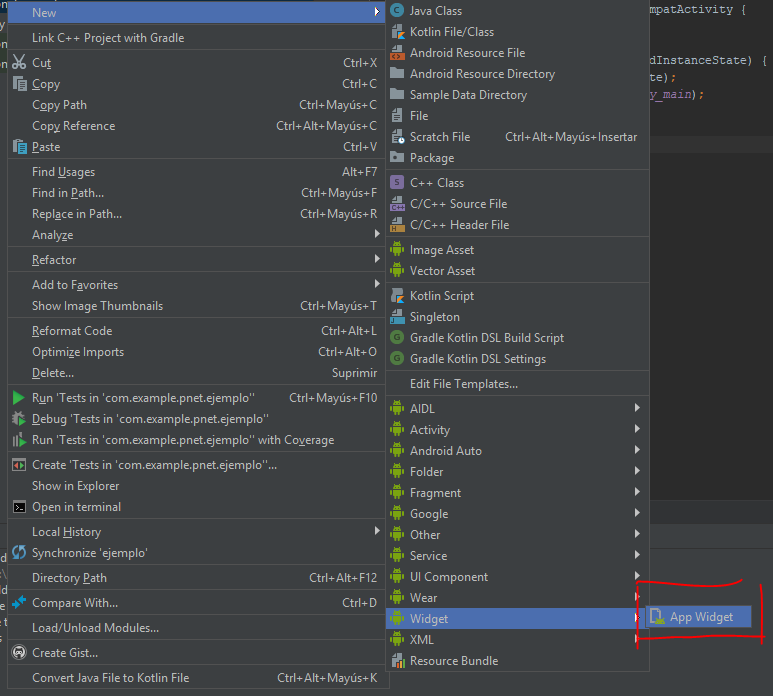
Ahora especificamos el nombre de nuestra aplicación, el nombre de nuestra empresa y la localización del proyecto en el que vamos a trabajar. En este ejemplo los campos quedarían rellenos de la siguiente manera:



Pulsamos next, y a continuación marcamos la casilla *Phone and Tablet* . En el desplegable que está justo abajo seleccionamos la API mínima compatible con nuestra aplicación. Es recomendable escoger una que incluya el mayor número de dispositivos posible, como la API 15. Le damos a next, y elegimos una *Empty Activity*. Dejamos las opciones por defecto y pulsamos *Finish* para crear nuestro proyecto.

1. **Apartado 2**
2. **Apartado 3**
3. **Apartado 4**
4. **Apartado 5**
5. **Apartado 6**
6. **Apartado 7**
7. **Apartado 8**
8. **Apartado 9**
9. **Widgets**

Para crear un widget seleccionamos *New* > *Widget* > *App Widget* tal y como se muestra en la siguiente imagen:



Indicamos el nombre de la clase que va a controlar su comportamiento, su tamaño mínimo, controlamos su posible colocación, seleccionamos Java como lenguaje del código fuente y pulsamos en *Finish*.

Una vez hagamos esto, Android Studio nos creará los correspondientes archivos .xml y .java para programar nuestro widget.

En este ejemplo vamos a hacer que nuestro widget abra nuestro *Main Activity*, y también vamos a personalizar su color. Para lo primero, antes de nada, tenemos que añadir un id único al layout que nos ha creado Android Studio:

<RelativeLayout mlns:android="http://schemas.android.com/apk/res/android"

**android:id="@+id/widget"**

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#09C"

android:padding="@dimen/widget\_margin">

<TextView

android:id="@+id/appwidget\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:layout\_margin="8dp"

android:background="#09C"

android:contentDescription="@string/appwidget\_text"

android:text="@string/appwidget\_text"

android:textColor="#ffffff"

android:textSize="24sp"

android:textStyle="bold|italic" />

</RelativeLayout>

Ahora nos vamos a la clase Java de nuestro widget, y sustituimos el método onUpdate(…) por la siguiente versión:

public void onUpdate(Context context, AppWidgetManager appWidgetManager,

int[] appWidgetIds) {

// There may be multiple widgets active, so update all of them

for (int appWidgetId : appWidgetIds) {

Intent intent = new Intent(context, MainActivity.class);

PendingIntent pendingIntent =

PendingIntent.getActivity(context, 0, intent, 0);

RemoteViews views = new RemoteViews(context.getPackageName(),

R.layout.new\_app\_widget);

views.setOnClickPendingIntent(R.id.widget, pendingIntent);

appWidgetManager.updateAppWidget(appWidgetId, views); }

}

Con el código anterior:

* Creamos un *Intent*  que abre la *Activity* que queramos de nuestra aplicación, y la incluye en un *PendingIntent.*
* Creamos un *RemoteViews* que establece un canal de comunicación entre nuestra aplicación y nuestro Widget.
* Indicamos que al pulsar el Widget se lance el *Intent.*
* A través de *AppWidgetMannager*, actualizamos nuestro Widget.

Ahora vamos a personalizar sus colores, para ello nos vamos al xml y asignamos a los dos ***android:background*** el color que queramos, en este caso el verde (#00ff00).

…

…

…  
 android:background="#00ff00"  
  
 <TextView  
 android:id="@+id/appwidget\_text"  
 android:background="#00ff00"

…

…

…  
</RelativeLayout>

Así quedaría nuestro Widget:

