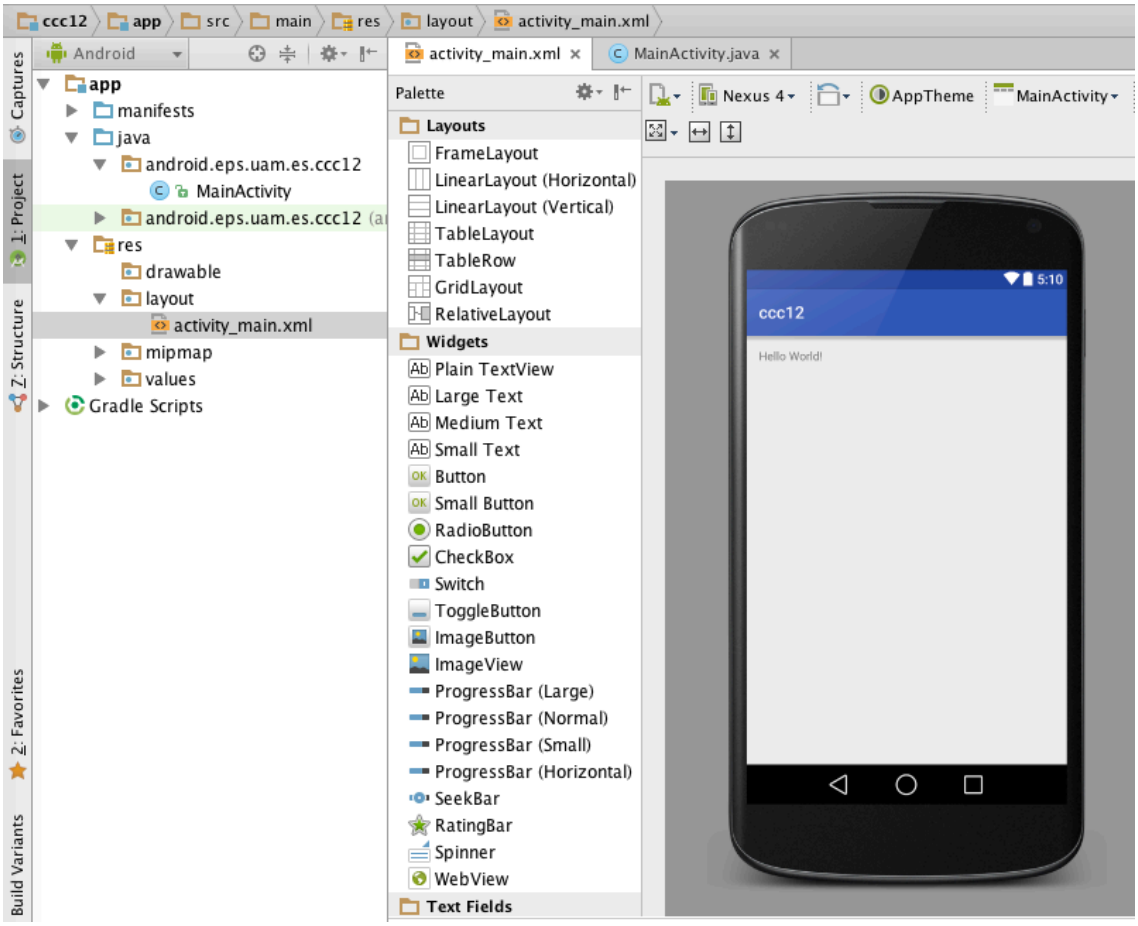
## 12. Otras vistas

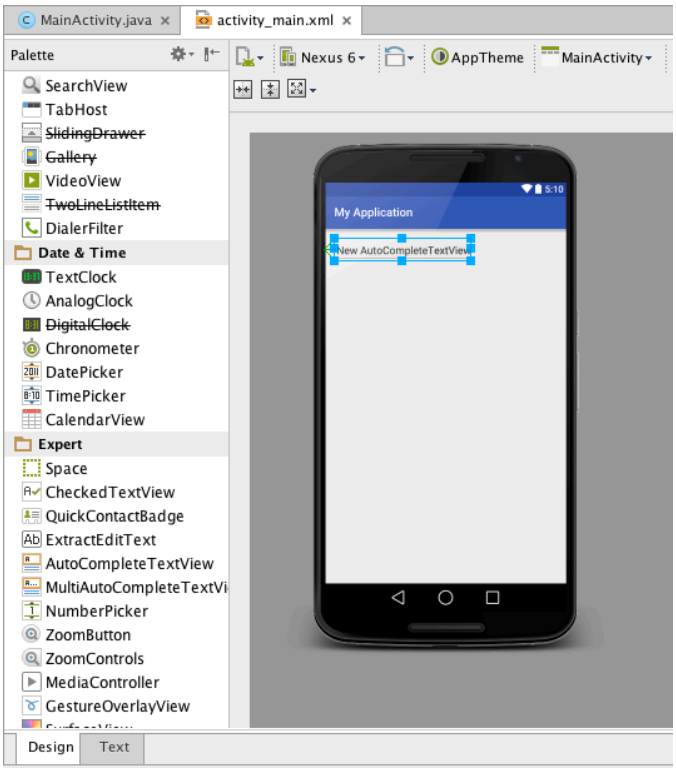
En las unidades anteriores hemos utilizado algunas vistas de Android como `Button` y `RadioButton`. Como puedes imaginar existen otras muchas vistas en Android, algunas de las cuales utilizaremos a lo largo del desarrollo de nuestro proyecto. En esta unidad vamos a ver una forma sencilla de experimentar con otras vistas en Android Studio.

Para ello crea un nuevo proyecto de nombre ccc12 y selecciona el fichero de diseño `activity_main.xml` de la carpeta `res` de la vista `Android`:

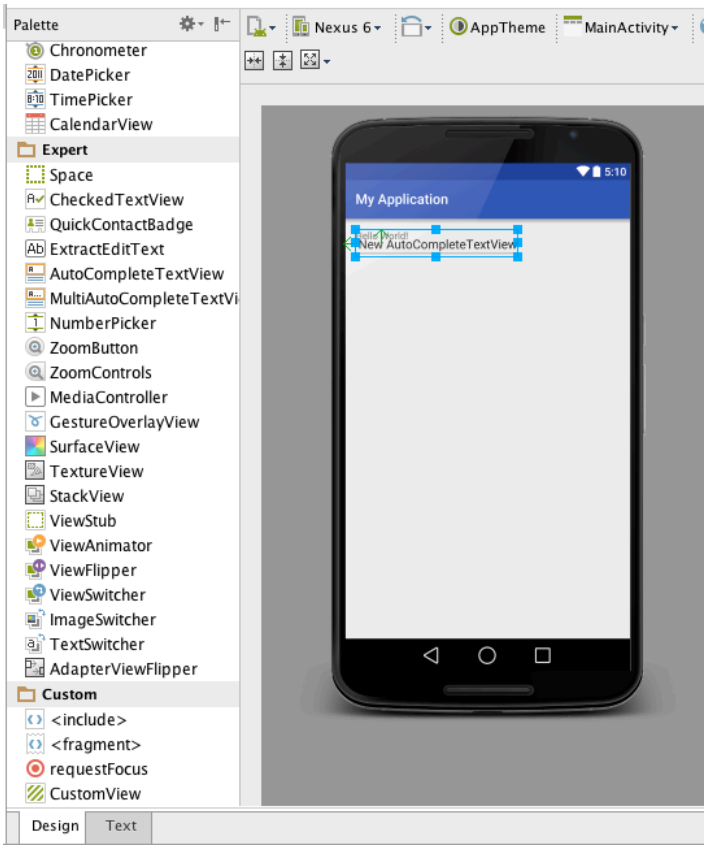



En la columna de nombre `Palette` tienes clasificadas las vistas y contenedores por categorías. Por ejemplo, dentro de la carpeta `Expert`, (tendrás que desplazarte hacia abajo en la columna `Palette`) nos encontramos con elementos como `AutoCompleteTextView`, que es una vista de texto que muestra sugerencias a medida que el usuario escribe en ella.

Antes de añadir gráficamente este elemento a la interfaz, vamos a eliminar la vista de tipo `TextView` que ha generado el entorno automáticamente. Para ello pulsa con el botón derecho del ratón y selecciona `Delete`:




A continuación, añade la nueva vista arrastrándola y dejándola caer sobre la pantalla del dispositivo. El resultado es el siguiente:



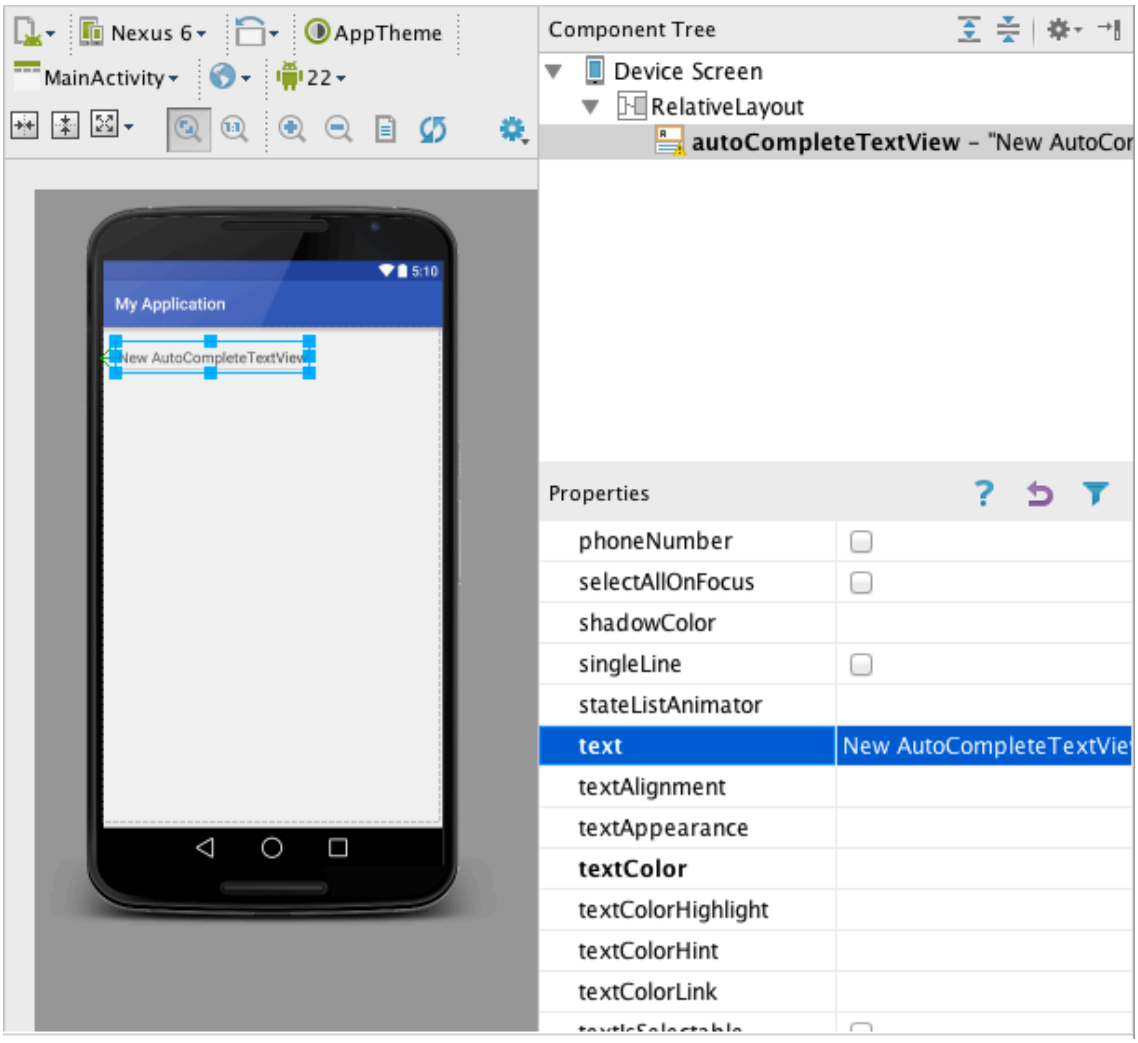
Android Studio añade automáticamente el elemento a nuestro fichero de diseño XML:

/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <AutoCompleteTextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New AutoCompleteTextView"
        android:id="@+id/autoCompleteTextView"
        android:layout_alignTop="@+id/textView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

Puedes consultar y actualizar las propiedades de la vista en la ventana de la derecha de nombre `Properties`:



Por ejemplo, puedes eliminar el valor que el atributo `android:text` tiene por defecto marcando y borrando el texto a la derecha de la propiedad `text`.

Las sugerencias que propone la vista a medida que se escribe se han de suministrar en el código como muestra la actividad `MainActivity.java`:

`/src/MainActivity.java`

```
package es.uam.eps.dadm.ccc12;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class MainActivity extends Activity {
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.activity_main);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line, CIUDADES);

        AutoCompleteTextView textView = (AutoCompleteTextView)
            findViewById(R.id.autoCompleteTextView);

        textView.setAdapter(adapter);
    }

    private static final String[] CIUDADES = new String[] {
        "Burgos", "Soria", "Barcelona", "Sevilla", "Santander"
    };
}
```

Las sugerencias están contenidas en el array `CIUDADES` que se pasa como tercer argumento al constructor del adaptador de arrays de nombre `adapter`:

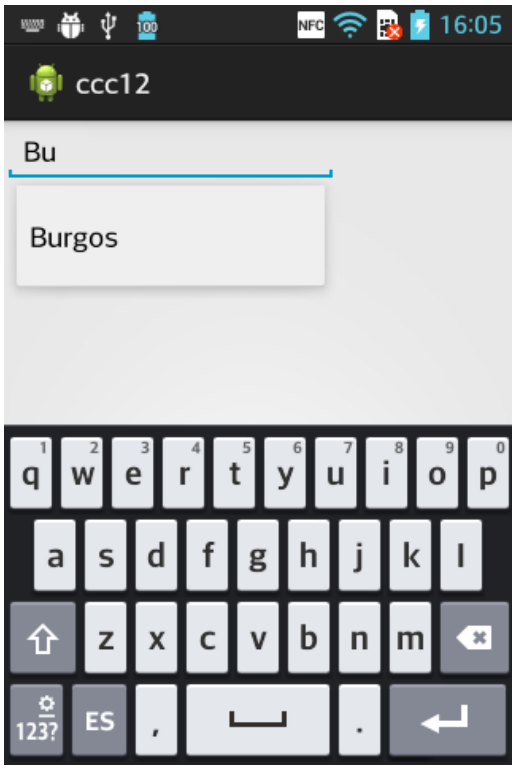
```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
     android.R.layout.simple_dropdown_item_1line, CIUDADES);
```

Un adaptador es una especie de intermediario entre unos datos que deseamos mostrar y la vista que los ha de representar. El primer argumento del constructor es el contexto, la actividad en este caso. El segundo es el identificador de recurso del fichero de diseño en el que mostrar los datos. En este caso utilizamos un diseño predefinido consistente en un `TextView`. Podemos suministrar otro fichero de diseño siempre que tenga como raíz un elemento de tipo `TextView`.

El adaptador se ajusta al objeto de tipo `AutoCompleteTextView` mediante la llamada al método `setAdapter()`:

```
textView.setAdapter(adapter);
```

La actividad resultante mejora la experiencia del usuario frente a un simple elemento `EditText` gracias a las sugerencias:


16:05
ccc12
Bu
Burgos
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123?
ES