

# Interfaz de usuario: Controles de entrada *(inputs)*:

“Texto (EditText)”

# Control EditText (I)

(ver su API)

- ▶ Es un **subtipo de TextView**.
- ▶ Permite que el usuario **introduzca texto** en la aplicación.
- ▶ Pueden ser de **una sola línea** o **multilínea**.
- ▶ **Tocar** un **campo de texto**, automáticamente **saca el teclado** y **coloca el cursor** en el campo de texto.
- ▶ Además de esto, **permiten selección de texto** (*para operaciones de “portapapeles”*) y **búsqueda de datos mediante autocompletado**.
- ▶ Para añadir un campo de texto se puede usar el elemento **<EditText>** en el layout **XML**.

# Control EditText (II)

## (ver su API)

### ► Propiedades más usadas:

#### 📁 android:text

- Permite **indicar el texto inicial** que contendrá el EditText.
- Si no queremos que aparezca inicializado, **no incluimos esta propiedad.**

#### 📁 android:inputType

- Permite **indicar el tipo de contenido** que se va a **introducir** en el cuadro de texto. Por ejemplo, *direcciones email, números de teléfono,...*
- ***Lo veremos a continuación más detallado.***

# Control EditText (III)

## Especificando el Tipo de Teclado (I)

- ▶ Para **especificar el tipo de teclado** que queremos para el **EditText**, usamos el atributo:






**android:inputType**

- ▶ *Por ejemplo*, si el usuario debe introducir un **e-mail**, se usa el tipo **“textEmailAddress”**

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

# Control EditText (IV)

## Especificando el Tipo de Teclado (II)

- ▶ Existen **diferentes tipos** disponibles para **diferentes situaciones**.
- ▶ Valores más usados para “**android:inputType**”:
  -  “**text**”: teclado normal de texto
  -  “**textEmailAddress**”: teclado normal de texto con el caracter arroba.
  -  “**textUri**”: Para direcciones Web. Es texto normal con el carácter “/”
  -  “**number**”: Teclado numérico básico
  -  “**phone**”: Teclado estilo teléfono

*Ver todos los valores posibles en la documentación.*

# Control EditText (V)






## Comportamientos de teclado con "android:inputType" (I)

- ▶ El atributo **android:inputType** permite también especificar algunos **comportamientos de teclado**, como *por ejemplo* :
  - 📄 poner en **mayúsculas** todas las **iniciales** de **nuevas palabras**
  - 📄 **autocompletar palabras**
  - 📄 **ocultar** el **texto** cuando son **contraseñas**
- ▶ **"android:inputType"** permite **combinaciones** a nivel de bits (usando el carácter "|"):
  - 📄 Así que, **se puede especificar un tipo de teclado y un comportamiento o más comportamientos**.
  - 📄 **Ejemplo:**

**android:inputType**="textPostalAddress | textCapWords |  
textNoSuggestions" → no sugerencias

# Control EditText (VI)

## Comportamientos de teclado con “android:inputType” (II)

- ▶ Valores más usados para “**android:inputType**” que indiquen un **comportamiento de teclado** son:
-  “**textCapSentences**”: pone en mayúscula la primera letra de cada frase.
-  “**textCapWords**”: se pone en mayúscula la primera letra de cada palabra
-  “**textAutoCorrect**”: corrige automáticamente palabras mal escritas.
-  “**textPassword**”: los caracteres introducidos se visualizan como puntos.
-  “**textMultiLine**”: Permite introducir textos largos incluyendo saltos de línea.

# Control EditText (VII)

## Comportamientos de teclado con "android:inputType" (III)

### ► Ejemplo de uso:

```
<EditText
    android:id="@+id/postal_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/postal_address_hint"
    android:inputType="textPostalAddress|
                    textCapWords|
                    textNoSuggestions" />
```



# Control EditText (VIII)

## Autocompletar (Sugerencias mientras escribes) (I)

- ▶ Si queremos **proporcionar sugerencias** a los usuarios mientras escriben, se puede usar una **subclase de EditText** llamada **AutoCompleteTextView**.
- ▶ Para implementar el auto-completar, se debe **especificar un adaptador que provea las sugerencias**.
- ▶ Hay **varios tipos de adaptadores** disponibles, dependiendo de si los datos vienen de una fuente u otra (*los adaptadores se verán más adelante*):

 de una base de datos o

 de un array,...

# Control EditText (IX)

## Autocompletar (Sugerencias mientras escribes) (II)

- ▶ Ejemplo de cómo usar un autoCompleteTextView que obtiene los datos de un array (usa ArrayAdapter):
  1. Añadir el autoCompleteTextView al layout:

```
<?xml version="1.0" encoding="utf-8"?>
<AutoCompleteTextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/autocomplete_country"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

# Control EditText (XIX)

## Autocompletar (Sugerencias mientras escribes) (III)

2. Definir el array que contiene las sugerencias. Por *ejemplo*, un array con nombres de países definido en un **fichero XML** en la carpeta **res/values/strings.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries_array">
        <item>Afghanistan</item>
        <item>Albania</item>
        <item>Algeria</item>
        <item>American Samoa</item>
        <item>Andorra</item>
        <item>Angola</item>
        <item>Anguilla</item>
        <item>Antarctica</item>
        ...
    </string-array>
</resources>
```

# Control EditText (XX)

## Autocompletar (Sugerencias mientras escribes) (IV)

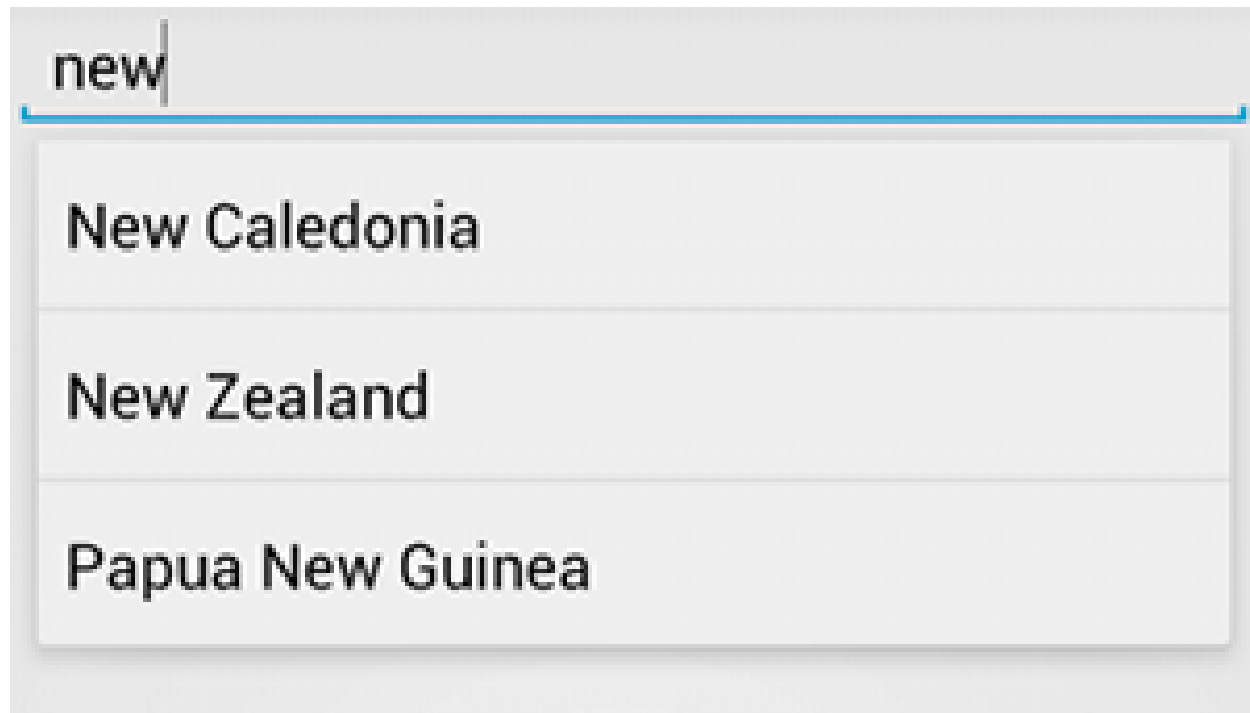
3. En nuestro **Activiy** usamos el siguiente código para **especificar el adaptador**:

```
// Get a reference to the AutoCompleteTextView in the layout
AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.autocomplete_country);
// Get the string array
String[] countries =
    getResources().getStringArray(R.array.countries_array);
// Create the adapter and set it to the AutoCompleteTextView
ArrayAdapter<String> adapter =
    new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1, countries);
textView.setAdapter(adapter);
```

# Control EditText (XXI)

## Autocompletar (Sugerencias mientras escribes) (V)

4. El resultado es:



## Control EditText (XXII)

### ¿Cómo responder ante cambios en el texto del EditText? (I)

- ▶ Los **editText** incorporan un método **addTextChangedListener()**
- ▶ Este método **permite incorporar un escuchador al EditText que reaccionará ante cambios en el texto del EditText.**
- ▶ Así, ese método, **podemos usarlo para responder a eventos de cambio de texto.**
- ▶ Para usarlo, **debemos pasarle un objeto** que implemente la **interfaz TextWatcher** o **crear una instancia anónima e implementar sus tres métodos.**
- ▶ Obviamente **los métodos pueden implementarse pero dejarse vacíos.**

# Control EditText (XXIII)

## ¿Cómo responder ante cambios en el texto del EditText? (II)

► Los **tres métodos** que incorpora el **interfaz TextWatcher** son:

- **beforeTextChanged()**

Este método es usado cuando los caracteres van a ser remplazados con algún nuevo texto.

En este caso, **EL TEXTO ES INEDITABLE.**

- **onTextChanged()**

Este método es usado cuando los cambios han tenido lugar y los caracteres han sido reemplazados.

En este caso, también, **EL TEXTO ES INEDITABLE.**

- **afterTextChanged()**

Este método es similar al anterior, excepto **que EL TEXTO ES EDITABLE.**

# Control EditText (XXIV)

## ¿Cómo responder ante cambios en el texto del EditText? (III)

- Ejemplo de uso creando una instancia anónima de **TextWatcher**:

```
EditText mPasswordLength;  
mPasswordLength = (EditText)findViewById(R.id.password_length);  
mPasswordLength.addTextChangedListener(new TextWatcher() {  
    public void afterTextChanged(Editable s)  
    {  
    }  
  
    public void beforeTextChanged(CharSequence s, int start, int count, int after)  
    {  
    }  
  
    public void onTextChanged(CharSequence s, int start, int before, int count)  
    {  
    }  
});
```



# Control EditText (XXV)

¿Cómo responder ante cambios en el texto del EditText? (IV)

## ▶ Ejemplo de uso implementando la interfaz **TextWatcher**:

1. Para **implementar el interfaz TextWatcher**, en el MainActivity ponemos:

```
public class MainActivity extends Activity implements TextWatcher { }
```

2. **Sobreescribir los 3 métodos** de TextWatcher en el MainActivity.
3. **Añadir este código** en el método onCreate() de MainActivity:

```
et=(EditText)findViewById(R.id.edittext1);  
et.addTextChangedListener(this);
```