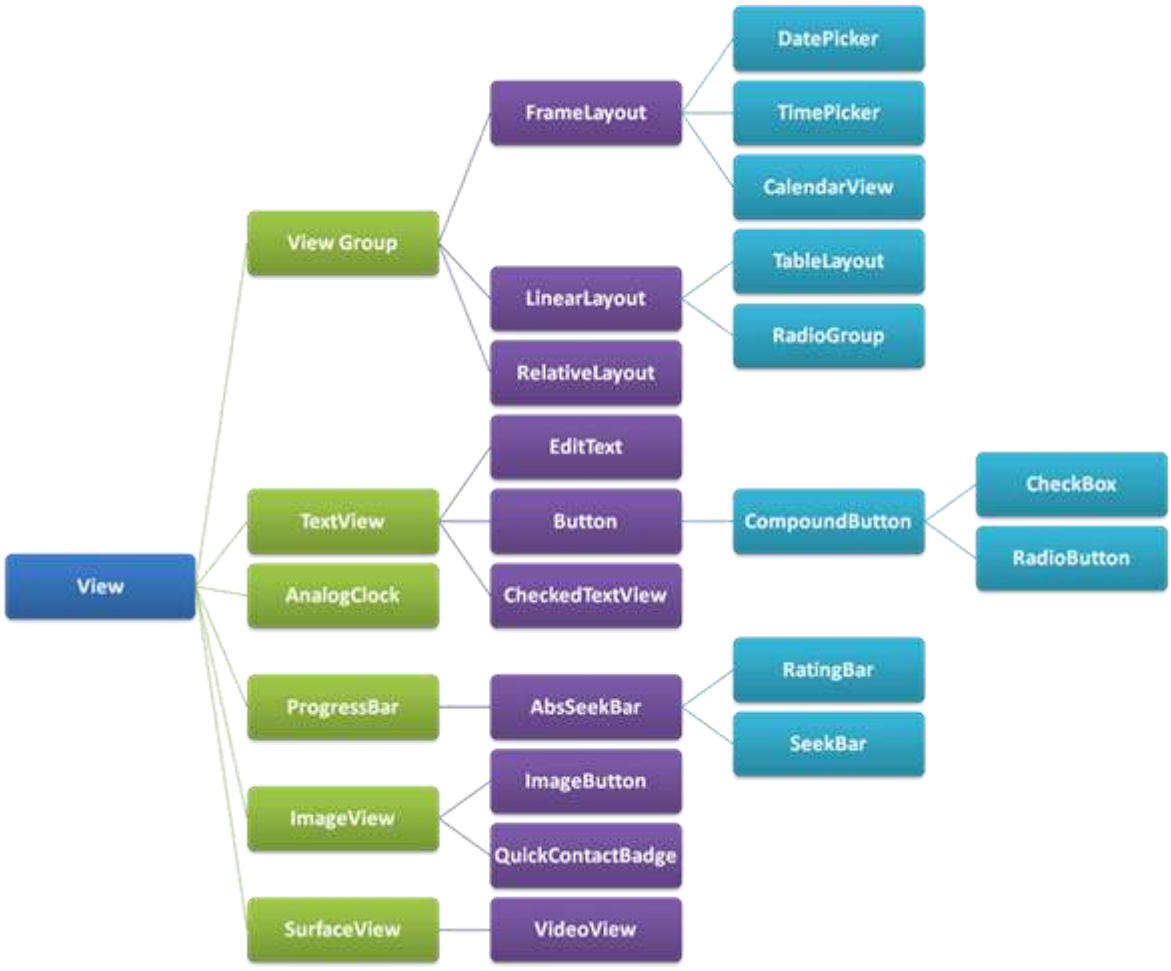
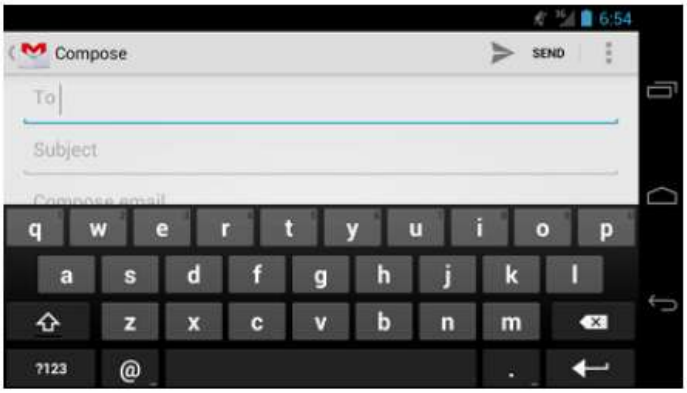
INDICE

1.1. Introducción

- El control **EditText** se usa para ingresar y editar texto por el usuario de una aplicación de Android.
- Este control es una subclase de la clase TextView.



- La forma de agregar un EditText en XML en un diseño es a través del componente: <EditText />
- Una propiedad importante es: android: inputType
  - ✓ Solo permitirá ingresar números, texto, contraseñas, teléfonos, etc.
  - ✓ También indicará si el control es de una sola línea o de varias líneas.
  - ✓ Haz que no complete las palabras, etc.
- También le permiten copiar, cortar y pegar.



- Cuando se entra en un cuadro de texto, en un dispositivo que no tenga un teclado físico conectado, se abrirá en la pantalla un **teclado virtual** (también llamado light/soft) que se adaptará a la propiedad de **android:inputType** como veremos a continuación.
- Finalmente, la tecla **Enter** en el teclado virtual puede realizar diferentes acciones, ya sea de forma predeterminada o explícitamente con la propiedad de **Android:imeOptions** (IME: Input Method Editor)
- Referencias:
  - ✓ El control EditText: https://developer.android.com/reference/android/widget/EditText
  - ✓ Campos de texto: https://material.io/components/text-fields
  - ✓ Campos de texto para programadores: https://developer.android.com/training/keyboard-input/style
  - ✓ Filtros de entrada (android:inputType): https://developer.android.com/reference/android/widget/TextView.html#attr_android:inputType
  - ✓ IME: https://developer.android.com/guide/topics/text/creating-input-method.html
  - ✓ android:imeOptions: https://developer.android.com/reference/android/widget/TextView.html#attr_android:imeOptions

1.2. Casos prácticos

- Partimos del proyecto inicial que hemos creado en unidades anteriores.
- Recordar que si se está utilizando bibliotecas de compatibilidad (predeterminadas), **vuestra activity se derivará de AppCompactActivity** y no de Activity.
- En los siguientes ejemplos, toda activity deriva de Activity, pero lo normal es que se utilice AppCompactActivity (cuidado al copiar los ejemplos).
- Si no lo tenemos creado antes, crear un paquete llamado **UI** como un subpaquete de su paquete principal.
- Dentro del paquete de IU, crearemos un nuevo paquete llamado: **EditTexto**.

- Dentro del paquete **EditTexto** crea una nueva ' *Empty Activity*' llamada: **UD02_01_EditText** de tipo Launcher y sin compatibilidad.
  - Modifique el archivo **AndroidManifiest.xml** y añadir una label a la activity.

```
<activity android:name=".UI.EditTexto.UD02_01_EditText"
    android:label="UD02_01_EditText">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

1.2.1.Distintos tipos de datos: la propiedad android:inputType

- Más información sobre todos los posibles valores InputType en este enlace.
- La imagen muestra un AVD con diferentes tipos de campos y con una leyenda interna que indica qué tipo de datos se deben ingresar en ese campo.

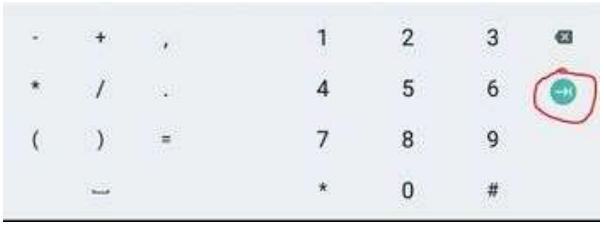
- El diseño xml que define en esa pantalla es el siguiente. Observar como siempre las líneas marcadas.

```
<LinearLayout 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:orientation="vertical" >

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe un número entero"
        android:inputType="numberSigned" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe un número decimal"
        android:inputType="numberSigned|numberDecimal" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe palabras...(saldrá la 1ª letra en mayúsculas)"
        android:inputType="textCapWords|textMultiLine" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Dirección e-mail. Esta línea sale por fuera de la pantalla"
        android:inputType="textEmailAddress" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe la clave."
        android:inputType="textPassword" />

</LinearLayout>
```

- Propiedad **android:hint**: indica el texto que se va a mostrar en el control cuando este vacío.
- Propiedad **android:inputType:** indica el tipo de datos que se pueden introducir en el campo.
  - ✓ Si no se pone la propiedad, el campo acepta toda la combinación de caracteres.
  - ✓ Se pueden combinar los filtros haciendo uso de "**|**".
- Obviamente los valores de los Hints es mejor tenerlos declarados en constantes en recursos XML.

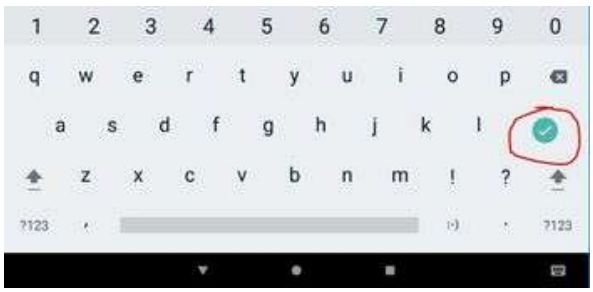
- *NOTA Edición 2015:* ***En el caso de usar un AVD con API 21****,* ***el botón*** *Next* ***es substituido por*** ">", ***el botón*** *Done* (Hecho) por una marca de verificación.

Botones teclado



En el caso de varios campos en el formulario, este botón indica **Next**, ir al siguiente si se presiona.

En el caso de un campo multilínea, este icono indica salto de línea o **Enter**.



Cuando llegamos al último campo del formulario, aparece el icono **Done**.

Distintos teclados

Cuando nos posicionamos en el primer campo, como es para enteros con signo (inputType), proporciona un teclado con solo números, signo "**-**" y "**.**" o "**,**".

Probar a intentar introducir un número decimal.

Tener en cuenta también que la tecla **Enter** ha sido reemplazada por **Next**, para pasar al siguiente campo. Pronto estudiaremos esta tecla.

El segundo campo ya admite números decimales y con signo. La tecla **Enter** continúa marcando **Next**.

El teclado ya contempla las letras y así sucesivamente. Observe cómo pone en mayúscula automáticamente la primera letra de cada palabra.

Como el tercer campo es de varias líneas, la tecla **Enter** ahora es un retorno de carro para que pueda seguir escribiendo varias líneas en el campo.

Colocamos nuestro dedo (o ratón) en el siguiente campo. Ahora el teclado ya ofrece @ para el correo. La tecla **Enter** sigue siendo **Next**.

Una vez que se presiona la tecla **Done**, el teclado desaparece.

Y ahora es el momento de ingresar la contraseña. Observar cómo la tecla **Enter** es ahora **Done** (hecho). Porque este campo es el último y no hay más a continuación.

1.2.2.Diferentes acciones de la tecla "Enter": android: imeOptions

- La tecla **Enter** del teclado virtual puede realizar una serie de acciones:
  - ✓ Por defecto:
    - el sistema determina si hay otro elemento al que puede dirigirse el foco, en cuyo caso la función de la tecla es **Next**.
    - Si no lo encuentra, es la acción de la tecla **Done**.
    - Excepto en algún **inputType** como multilínea.
- Podemos asignar explícitamente qué acción queremos que realice cada cuadro de texto. Para eso está el atributo de **android: imeOptions**.
  - ✓ Que de un campo vaya al anterior, que envíe el texto, que busques el texto, etc. En las referencias hay un enlace a posibles acciones.

- El siguiente XML va a la misma pantalla anterior, solo se han cambiado las acciones de la tecla **Enter** para algunos **EditText**. Observar las acciones definidas en las líneas marcadas.

```
<LinearLayout 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:orientation="vertical" >

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe un número entero"
        android:imeOptions="actionDone"
        android:inputType="numberSigned" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe un número decimal"
        android:imeOptions="actionPrevious"
        android:inputType="numberSigned|numberDecimal" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe palabras...(saldrá la 1ª letra en mayúsculas)"
        android:inputType="textCapWords|textMultiLine" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Dirección e-mail. Esta línea sale por fuera de la pantalla"
        android:imeOptions="actionSend"
        android:inputType="textEmailAddress" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Escribe la clave."
        android:inputType="textPassword" />

</LinearLayout>
```

Introducir datos

El primer campo tiene la acción **Done** definida para la tecla **Enter**.

El segundo campo la acción pasa al campo anterior.

El campo de mail ha definido la acción **Send** que se capturará en Java y realizará una acción en consecuencia.

1.2.3.Foco

- De forma predeterminada, siempre se centrará en el primer edittext que se encuentre en la ventana ComponentTree del diseñador de layout.
- Cuando cargamos un layout podemos indicar cuál será la View por defecto a la que irá el foco saltándonos la regla anterior.

  Para ello debemos utilizar la etiqueta: <requestFocus /> dentro de la View al que queremos que vaya el foco al iniciar la activity.

```
<EditText
    android:id="@+id/evEmail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Dirección e-mail. Esta línea sale por fuera de la pantalla"
    android:imeOptions="actionSend"
    android:inputType="textEmailAddress">

    <requestFocus/>

</EditText>
```

  Fijarse que dicha etiqueta no está dentro de la etiqueta <EditText ....> como el resto de los atributos y tuvimos que eliminar el final de la etiqueta (/) al final para poner una etiqueta </EditText>

- En caso de que queramos hacerlo por programación tendríamos que llamar al método **requestFocus()** de la forma:

```
public class UD02_01_EditText extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d02_01__edit_text);

        EditText email = findViewById(evEmail);
        email.requestFocus();
    }
}
```

Cuando tenemos EditText, normalmente el foco pasa al primero de la lista y aparece el teclado virtual predeterminado.

En algunos sitios indica que si ponemos el foco sobre o view diferente a un EditText, el teclado desaparece, pero haciendo pruebas, a veces funciona y otras no.

La forma que encontré para ocultar el teclado virtual al iniciar la actividad es colocando el método *onCreate*:

getWindow().setSoftInputMode(WindowManager.LayoutParams.***SOFT_INPUT_STATE_HIDDEN***);

- Para hacer que el control pase de una view a otra haciendo Tab en el teclado virtual, tendríamos que usar en el diseño la propiedad nextFocusForward de la forma: android: **`nextFocusForward = "@id/etNome"`**
- Para hacer que el control pase de una view a otra haciendo clic en la flecha del teclado virtual, tendríamos que usar en el diseño la propiedad nextFocusDown de la forma: android: **`nextFocusDown = "@id/etNome"`**

1.3. Accesibilidad

- Podemos activar el modo TalkBack siguiendo las instrucciones de este enlace.

  Este modo "leerá" la descripción de cada elemento.

- Como regla general podemos dar las siguientes pautas:

  Un <TextView> debe tener el atributo **'android: hint'** como se muestra en el siguiente ejemplo:

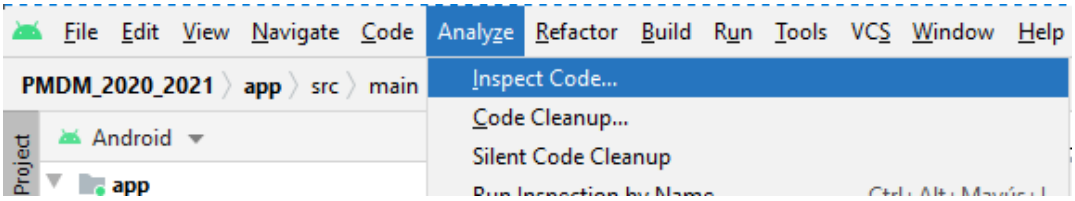
```
<EditText
    android:id="@+id/etNome_UD03_01_Intents"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:inputType="textPersonName"
    android:hint="Nombre a enviar"
    />
```

- En caso de que EditText esté asociado con un TextView, podemos reemplazar este atributo por el atributo: **android:labelFor** del TextView que indica cuál es el EditText asociado con el TextView.

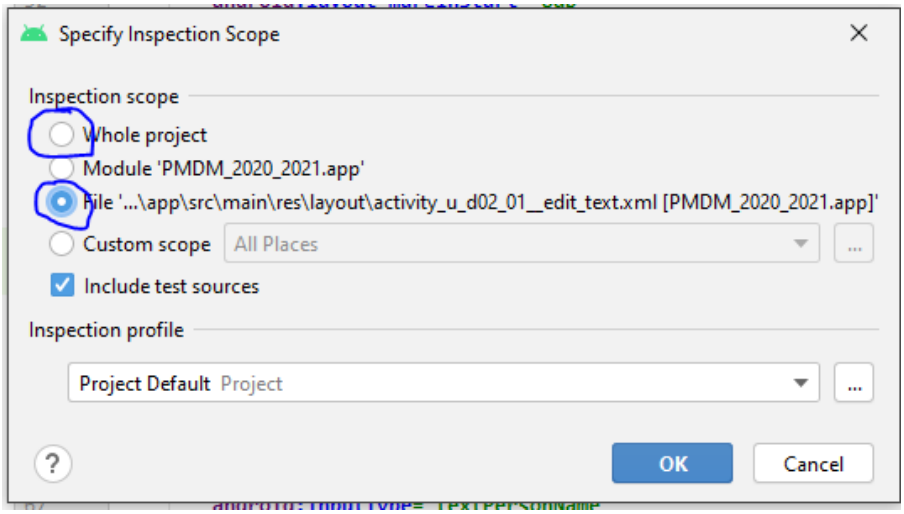
```
<TextView
    android:id="@+id/textView7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:text="Dime tu nombre:"
    android:textSize="18sp"
    android:labelFor="@id/etNome_UD03_01_Intents"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="@+id/etNome_UD03_01_Intents" />

<EditText
    android:id="@+id/etNome_UD03_01_Intents"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintStart_toEndOf="@+id/textView7"
    app:layout_constraintTop_toTopOf="parent" />
```

- Para comprobar que no tenemos problemas de accesibilidad:

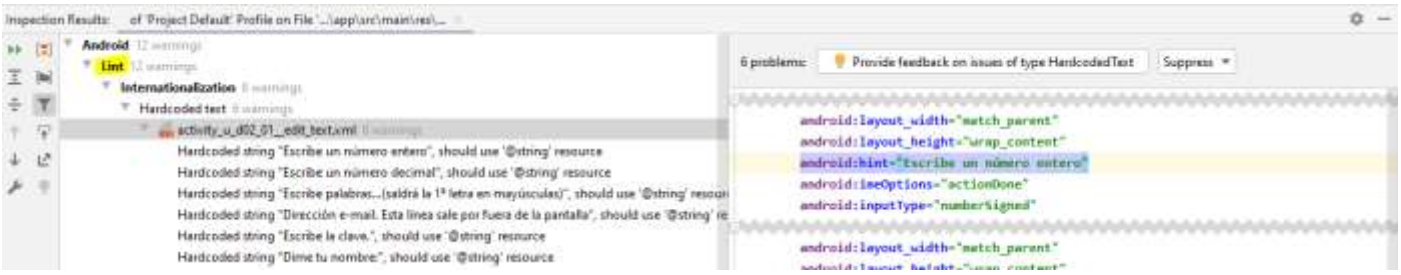
Comprobando la accesibilidad



Vamos al menú de AndroidStudio y escogemos la opción **Analyce → Inspect Code**



Indicamos si queremos analizar el proyecto completo o la activity actual.



Si hay algún problema tenemos que ir a la sección de **Lint** y ver los errores que aparecen.

1.4. Atributos a conocer

- Se dispone de una lista de atributos en: https://developer.android.com/reference/android/widget/EditText.html
- Dentro de este control tendréis que conocer y saber utilizar los siguientes atributos:
  - id
  - layout_width, layout_height
  - margin
  - padding
  - gravity
  - contentDescription
  - text
  - hint
  - inputType
  - maxLines
  - imeoptions
  - textXXXX (Size, Color, Style, Alignment)
  - fontFamily
  - typeFace
  - visibility
  - ems
  - maxLength

1.5. Métodos a conocer en el manejo de los EditText´s

- Referenciar a un EditText con el método findViewById.
- Recupera el contenido del texto.
- Cambia el contenido del texto, pudiendo utilizar recursos guardados en **/res/values**.
- Agrega texto nuevo a uno existente.
- Modificar propiedades básicas, como color, tamaño, visibilidad, ... (y métodos getZZZZZ para obtener sus valores)
- Cambiar o foco a un view EditText concreto.

Ejemplos:

```
EditText et = findViewById(R.id.etvTextoEditText);          // Referenciamos a un EditText que este en la Activity.
String cadeaet = et.getText().toString();                   // Fijarse que getText() devuelve un objeto Editable, no un String.
et.setText("Caeblamos o texto directamente");               // Caeblamos el contenido por una cadena concreta
String textoRes = getResources().getString(R.string.app_name);
et.setText(textoRes + " texto que ven da BD");              // Si queremos concatenar texto una base de datos con un texto que pueda ser traducido

et.append(" engadimos novo texto");                         // append para añadir

et.setTextSize(20);                                         // Ajustar tamaño

et.setTextColor(Color.BLUE);                                // Cambia el color

et.setVisibility(View.VISIBLE);                             // Las constantes aparecen en Android Studio al teclear. GONE hace que no ocupe espacio en el Layout.


EditText et2 = findViewById(R.id.etvTextoEditText2);        // Referenciamos al otro EditText que este en la Activity.
et2.requestFocus();                                         // Cambiamos el foco para ese EditText
```