

# Tema 6. Interfaz de Usuario II

[Descargar estos apuntes](#)

## Índice

1. [Sliders](#)
2. [Chips](#)
  1. [Input chips](#)
  2. [Choice Chip](#)
  3. [Filter Chips](#)
  4. [Action Chips](#)
3. [Diálogos](#)
  1. [Alert dialog](#)
  2. [Simple dialog](#)
  3. [Confirmation dialog](#)
  4. [Full-screen dialog](#)
4. [DataPickerDialog](#)
5. [ProgressBar](#)

# Sliders

## Chips

Uno de los componentes más atractivos de la librería **Material Design** es el **Chip**. Existen cuatro tipos: de entrada, de filtro, de elección y de acción.

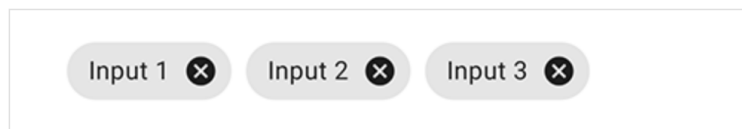
Los **chips** se utilizan habitualmente agrupados. Para poder hacerlo de forma eficiente se recomienda la utilización del componente **ChipGroup** que permite patrones de comportamiento sobre la vista.

Los cambios de un chip se pueden observar así:

```
chip.setOnClickListener {  
    // Responds to chip click  
}  
  
chip.setOnCloseIconClickListener {  
    // Responds to chip's close icon click if one is present  
}  
  
chip.setOnCheckedChangeListener { chip, isChecked ->  
    // Responds to chip checked/unchecked  
}
```

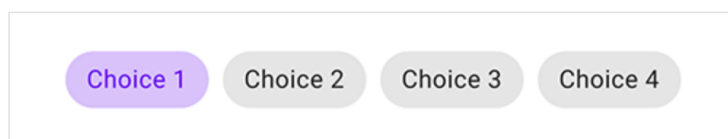
## Input chips

Como características generales pueden contener un icono de chip opcional, un icono de cierre opcional y opcionalmente se pueden marcar.



## Choice Chip

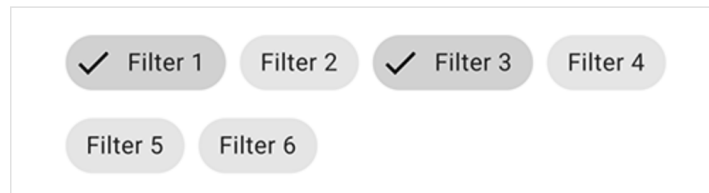
Permiten la selección de un único elemento de las opciones de **Chip** existentes. Cuando seleccionamos un **Chip** automáticamente se desmarca el que estuviera seleccionado.



## Filter Chips

Los chips de filtro utilizan etiquetas o palabras descriptivas para filtrar el contenido.

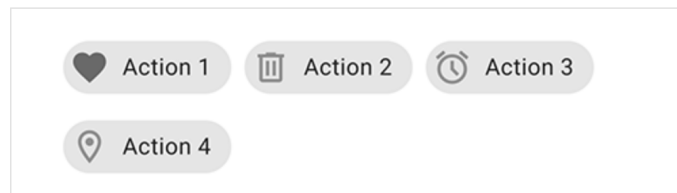
Los chips de filtro delimitan y muestran claramente las opciones en un área compacta. Son una buena alternativa para alternar botones o casillas de verificación.



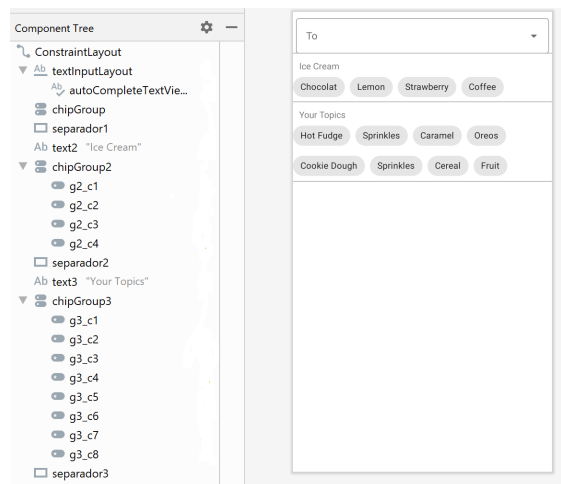
## Action Chips

Los chips de acción ofrecen acciones relacionadas con el contenido principal. Deben aparecer de forma dinámica y contextual en una interfaz de usuario.

Una alternativa a los chips de acción son los botones, que deben aparecer de manera persistente y consistente.



Veamos un diseño donde implementar todas estas vistas:



Colocaremos un control **autocompleteTextView** que nos permitirá ir seleccionando nombres de una lista y los añadiremos como **InputChip** dentro de un **chipGroup** de forma dinámica (posteriormente modificaremos esta parte para incluir el chip seleccionado dentro del **autocompleteTextView**).

También tendremos otro **chipGroup** para agrupar los **chipChoice** y otro **chipGroup** para los **chipFilter**. Cada uno de esas vistas van separadas por un delimitador **view**.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDropdownMe
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:hint="To"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <com.google.android.material.textfield.MaterialAutoCompleteTextView
      android:id="@+id/autoCompleteTextView"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:singleline="true"
      android:completionThreshold="2"/>
  </com.google.android.material.textfield.TextInputLayout>

  <com.google.android.material.chip.ChipGroup
    android:id="@+id/chipGroup"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:singleLine="false"
    style="@style/Widget.MaterialComponents.ChipGroup"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/textInputLayout">
  </com.google.android.material.chip.ChipGroup>

  <View
    android:id="@+id/separador1"
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:background="@android:color/darker_gray"
    app:layout_constraintBottom_toBottomOf="@id/chipGroup"
    app:layout_constraintLeft_toLeftOf="parent" />

  <TextView
    android:id="@+id/text2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Ice Cream"
    app:layout_constraintLeft_toLeftOf="parent"

```

```

        app:layout_constraintTop_toBottomOf="@id/separador1" />

<com.google.android.material.chip.ChipGroup
    android:id="@+id/chipGroup2"
    app:selection="true"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:singleLine="true"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text2">

    <com.google.android.material.chip.Chip
        android:id="@+id/g2_c1"
        style="@style/Widget.MaterialComponents.Chip.Choice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Chocolat" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g2_c2"
        style="@style/Widget.MaterialComponents.Chip.Choice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Lemon" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g2_c3"
        style="@style/Widget.MaterialComponents.Chip.Choice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Strawberry" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g2_c4"
        style="@style/Widget.MaterialComponents.Chip.Choice"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Coffee" />
</com.google.android.material.chip.ChipGroup>

<View
    android:id="@+id/separador2"
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:background="@android:color/darker_gray"
    app:layout_constraintBottom_toBottomOf="@id/chipGroup2"
    app:layout_constraintLeft_toLeftOf="parent" />

<TextView
    android:id="@+id/text3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

        android:layout_margin="10dp"
        android:text="Your Topics"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@id/separador2" />

<com.google.android.material.chip.ChipGroup
    android:id="@+id/chipGroup3"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text3">

    <com.google.android.material.chip.Chip
        android:id="@+id/g3_c1"
        style="@style/Widget.MaterialComponents.Chip.Filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hot Fudge" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g3_c2"
        style="@style/Widget.MaterialComponents.Chip.Filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Sprinkles" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g3_c3"
        style="@style/Widget.MaterialComponents.Chip.Filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Caramel" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g3_c4"
        style="@style/Widget.MaterialComponents.Chip.Filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Oreos" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g3_c5"
        style="@style/Widget.MaterialComponents.Chip.Filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cookie Dough" />

    <com.google.android.material.chip.Chip
        android:id="@+id/g3_c6"
        style="@style/Widget.MaterialComponents.Chip.Filter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

```

```

        android:text="Sprinkles" />

        <com.google.android.material.chip.Chip
            android:id="@+id/g3_c7"
            style="@style/Widget.MaterialComponents.Chip.Filter"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cereal" />

        <com.google.android.material.chip.Chip
            android:id="@+id/g3_c8"
            style="@style/Widget.MaterialComponents.Chip.Filter"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Fruit" />
    </com.google.android.material.chip.ChipGroup>

    <View
        android:id="@+id/separador3"
        android:layout_width="match_parent"
        android:layout_height="1dp"
        android:background="@android:color/darker_gray"
        app:layout_constraintTop_toBottomOf="@id/chipGroup3"
        app:layout_constraintLeft_toLeftOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

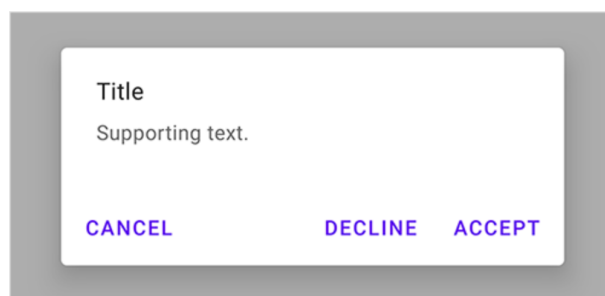
```

## Diálogos

Un cuadro de diálogo es un tipo de ventana modal que aparece por encima del contenido de la aplicación para proporcionar información crítica o solicitar una acción. Los cuadros de diálogo deshabilitan todas las funciones de la aplicación cuando aparecen y permanecen en la pantalla hasta que se confirman, se descartan o se toma una acción requerida.

### Alert dialog

El siguiente ejemplo muestra un cuadro de diálogo de alerta.



Para gestionar los eventos sobre el mismo:

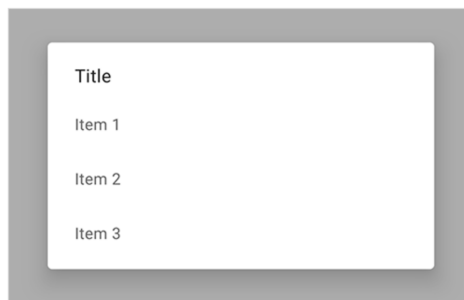
```

MaterialAlertDialogBuilder(context)
    .setTitle(resources.getString(R.string.title))
    .setMessage(resources.getString(R.string.supporting_text))
    .setNeutralButton(resources.getString(R.string.cancel)) { dialog, which ->
        // Respond to neutral button press
    }
    .setNegativeButton(resources.getString(R.string.decline)) { dialog, which ->
        // Respond to negative button press
    }
    .setPositiveButton(resources.getString(R.string.accept)) { dialog, which ->
        // Respond to positive button press
    }
    .show()kotlin

```

## Simple dialog

Muestran los elementos de una lista, que son inmediatamente procesables cuando se seleccionan. No tienen botones de acción.



Para gestionar los eventos sobre el mismo:

```

val items = arrayOf("Item 1", "Item 2", "Item 3")

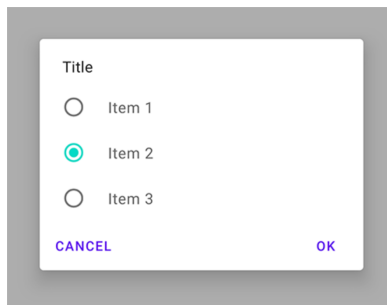
MaterialAlertDialogBuilder(context)
    .setTitle(resources.getString(R.string.title))
    .setItems(items) { dialog, which ->
        // Respond to item chosen
    }
    .show()

```

## Confirmation dialog

Los diálogos de confirmación brindan a los usuarios la posibilidad de proporcionar una confirmación final de una elección antes de comprometerse con ella, para que tengan la oportunidad de cambiar de opinión si es necesario.





Para gestionar los eventos sobre el mismo:

```
val singleItems = arrayOf("Item 1", "Item 2", "Item 3")
val checkedItem = 1

MaterialAlertDialogBuilder(context)
    .setTitle(resources.getString(R.string.title))
    .setNegativeButton(resources.getString(R.string.cancel)) { dialog, which ->
        // Respond to neutral button press
    }
    .setPositiveButton(resources.getString(R.string.ok)) { dialog, which ->
        // Respond to positive button press
    }
    // Single-choice items (initialized with checked item)
    .setSingleChoiceItems(singleItems, checkedItem) { dialog, which ->
        // Respond to item chosen
    }
    .show()
```

Es posible también seleccionar más un elemento de los presentados en el diálogo. Para implementar este tipo de diálogo:

```
val multiItems = arrayOf("Item 1", "Item 2", "Item 3")
val checkedItems = booleanArrayOf(true, false, false, false)

MaterialAlertDialogBuilder(context)
    ...
    //Multi-choice items (initialized with checked items)
    .setMultiChoiceItems(multiItems, checkedItems) { dialog, which, checked ->
        // Respond to item chosen
    }
    .show()
```

## Full-screen dialog


Los cuadros de diálogo de pantalla completa son los únicos cuadros de diálogo sobre los que pueden aparecer otros cuadros de diálogo.

No existe una implementación de **Material Design** específica de un diálogo de pantalla completa. Podemos implementarlo usando un **DialogFragment**.

## DataPickerDialog


## ProgressBar




 Aclaraciones:

- **ImageView** líneas 10..16:



 Aquí tenemos un enlace donde se encuentra información sobre la gestión de colores que ofrece Material Design

 EjercicioPropuestoCardView