

Apunts

[Descarregar aquests apunts](#)

Tema 6. Interfície d'Usuari II

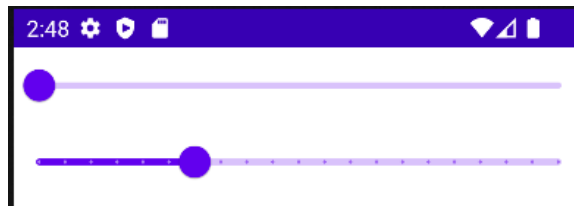
Índex

1. [Sliders](#)
2. [Chips](#)
 1. [Input chips](#)
 2. [Choice Chip](#)
 3. [Filter Chips](#)
 4. [Action Chips](#)
3. [Diàlegs](#)
 1. [Alert dialog](#)
 2. [Simple dialog](#)
 3. [Confirmation dialog](#)
 4. [Full-screen dialog](#)
4. [DataPicker](#)
5. [Progress indicators](#)

Sliders

Permeten als usuaris veure i seleccionar un valor (o rang) del rang al llarg d'una barra. Són ideals per a ajustar configuracions com el volum i la lluentor (brillo), o per a aplicar filtres d'imatge. En interactuar amb un control lliscant (deslizante), els canvis han de reflectir-se immediatament a l'usuari.

Els controls lliscants poden usar icones en tots dos extrems de la barra per a representar una escala numèrica o relativa. El rang de valors o la naturalesa dels valors, com el canvi de volum, es poden indicar amb icones.



Poden ser continus (permeten seleccionar un valor aproximat subjectiu) o discrets (permeten seleccionar un valor exacte).

```
<!-- Continue slider -->
<com.google.android.material.slider.Slider
    android:id="@+id/continueSlider"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
6    android:valueFrom="0.0"
7    android:valueTo="100.0"/>
```

```
<!-- Discrete slider -->
<com.google.android.material.slider.Slider
    android:id="@+id/discreteSlider"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:valueFrom="0.0"
    android:valueTo="100.0"
16    android:stepSize="5"
17    android:value="30"/>
```

📌 Aclariments:

- **Línia 6** amb l'atribut `android:valueFrom` establim el valor mínim del `slider`
- **Línia 7** amb l'atribut `android:valueTo` establim el valor màxim.
- **Línia 16** amb l'atribut `android:stepSize` establim el valor del increment del `slider` discret.

- **Línia 17** amb l'atribut `android:value` podem establir un valor inicial al `slider` ja siga aquest continu o discret.

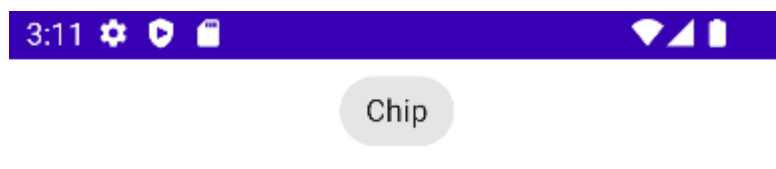
Podem atendre els canvis sobre el `slider` amb els listeners:

```
slider.addOnSliderTouchListener(object : Slider.OnSliderTouchListener {  
    override fun onStartTrackingTouch(slider: Slider) {  
        // Responds to when slider's touch event is being started  
    }  
  
    override fun onStopTrackingTouch(slider: Slider) {  
        // Responds to when slider's touch event is being stopped  
    }  
})  
  
slider.addChangeListener { slider, value, fromUser ->  
    // Responds to when slider's value is changed  
}
```

👉 Practica realitzant un nou projecte que permeta modificar el color de fons del nostre `layout` a través de tres `slider` que permeten variar el **RGB** del color d'aquest.

Chips

Un dels components més atractius de la llibreria `Material Design` és el `Chip`.



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

Existeixen quatre tipus: d'entrada, de filtre, d'elecció i d'acció.

Els `chips` s'utilitzen habitualment agrupats. Per a poder fer-ho de manera eficient es recomana la utilització del component `ChipGroup` que permet patrons de comportament sobre la vista.

Els canvis d'un chip es poden observar així:

```

chip.setOnClickListener {
    // Responds to chip click
}

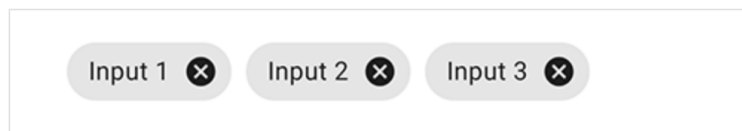
chip.setOnCloseIconClickListener {
    // Responds to chip's close icon click if one is present
}

chip.setOnCheckedChangeListener { chip, isChecked ->
    // Responds to chip checked/unchecked
}

```

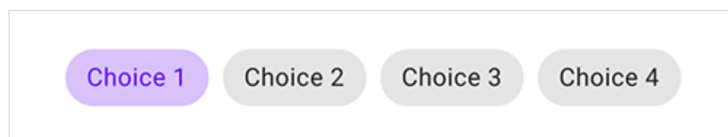
Input chips

Com caraterístiques generals poden contindre una icona de chip opcional, una icona de tancament opcional i opcionalment es poden marcar.



Choice Chip

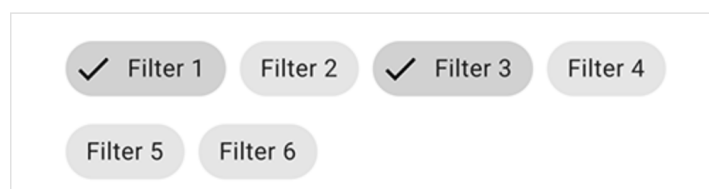
Permeten la selecció d'un únic element de les opcions de **chip** existents. Quan seleccionem un **chip** automàticament es desmarca el que estiguera seleccionat.



Filter Chips

Els chips de filtre utilitzen etiquetes o paraules descriptives per a filtrar el contingut.

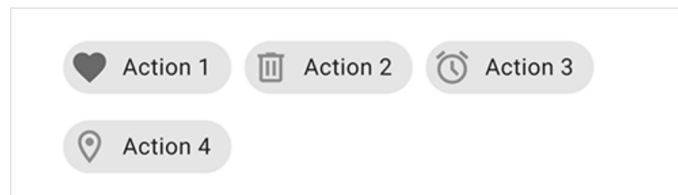
Els chips de filtre delineen i mostren clarament les opcions en una àrea compacta. Són una bona alternativa per a alternar botons o caselles de verificació.



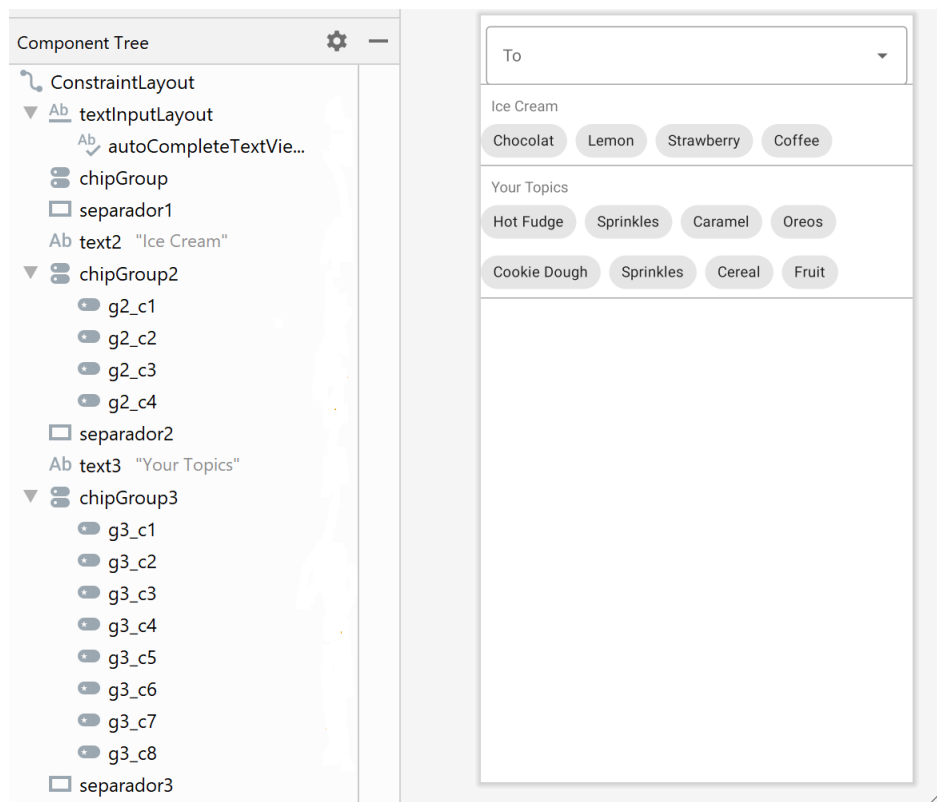
Action Chips

Els chips d'acció ofereixen accions relacionades amb el contingut principal. Han d'aparèixer de manera dinàmica i contextual en una interfície d'usuari.

Una alternativa als chips d'acció són els botons, que han d'aparèixer de manera persistent i consistent.



Vegem un disseny on implementar totes aquestes vistes:



Col·locarem un control `autoCompleteTextView` que ens permetrà anar seleccionant noms d'una llista i els afegirem com `InputChip` dins d'un `chipGroup` de manera dinàmica (posteriorment modificarem aquesta part per a incloure el chip seleccionat dins del `autoCompleteTextView`).

També tindrem un altre `chipGroup` per a agrupar els `chipChoice` i un altre `chipGroup` per als `chipFilter`. Cadascun d'aqueixes vistes van separades per 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>

```

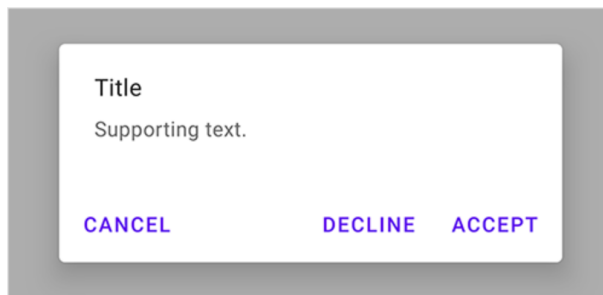
👉 Practica modificant l'exercici anterior de manera que s'incorporen els **Chip** al camp **AutoCompleteTextView** . Cal aconseguir que el botó d'eliminació del **Chip** estiga actiu i utilitzable.

Diàlegs

Un quadre de diàleg és un tipus de finestra modal que apareix per damunt del contingut de l'aplicació per a proporcionar informació crítica o sol·licitar una acció. Els quadres de diàleg deshabiliten totes les funcions de l'aplicació quan apareixen i romanen en la pantalla fins que es confirmen, es descarten o es pren una acció requerida.

Alert dialog

El següent exemple mostra un quadre de diàleg d'alerta.

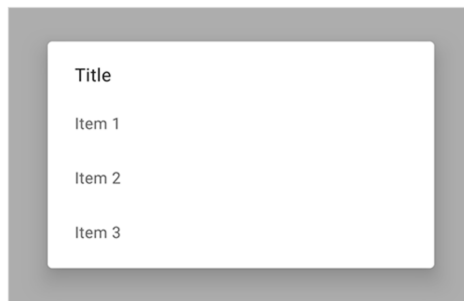


Per a gestionar els esdeveniments sobre aquest:

```
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

Mostren els elements d'una llista, que es poden processar immediatament quan se seleccionen. No tenen botons d'acció.



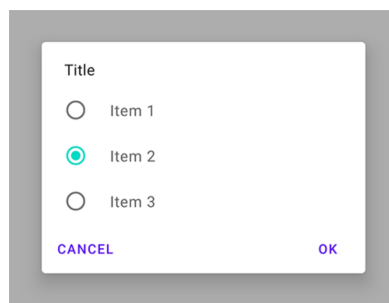
Per a gestionar els esdeveniments sobre aquest:

```
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

Els diàlegs de confirmació brinden als usuaris la possibilitat de proporcionar una confirmació final d'una elecció abans de comprometre's amb ella, perquè tinguin l'oportunitat de canviar d'opinió si és necessari.



Per a gestionar els esdeveniments sobre aquest:

```

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

MaterialAlertDialogBuilder(context)
    .setTitle(resources.getString(R.string.title))
    .setNeutralButton(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()

```

És possible també seleccionar més un element dels presentats en el diàleg. Per a implementar aquest tipus de diàleg:

```

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

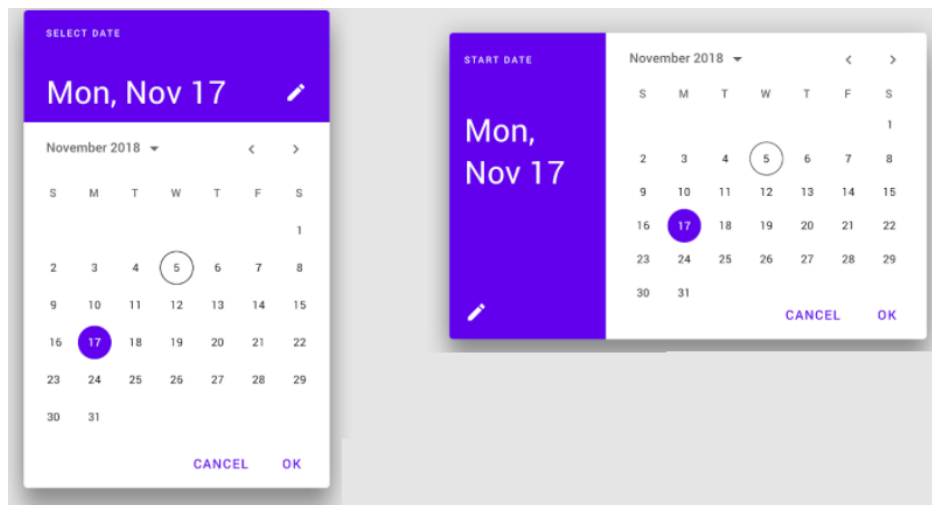
Els quadres de diàleg de pantalla completa són els únics quadres de diàleg sobre els quals poden aparèixer altres quadres de diàleg.

No existeix una implementació de **Material Design** específica d'un diàleg de pantalla completa. Podem implementar-ho usant un **DialogFragment**.

 [Exercici resolt Dialogue_Personalitzat](#)

DataPicker

Permeten als usuaris seleccionar una data o un rang de dates. Els `dataPicker` s'adapten a l'orientació del dispositiu:



El més freqüent és que aquests controls no apareguen directament en la interfície, sinó que quan premem un camp d'edició associat a una data, ens aparega un diàleg (o `fragmentDialog`) que inclou el `dataPicker`.

Crearem el projecte **EjemploDataPicker** i col·loquem en l'activitat principal un `TextInputEditText` en el listener corresponent al `onClick` activarem el `dataPicker` i sobre ell visualitzarem l'entrada realitzada en el `dataPicker` quan aquest es tanque.



És important que el nostre `TextInputEditText` tinga les següents propietats `android:clickable="false"` i `android:focusable="false"` que eviten que el nostre camp d'edició es pugui seleccionar o prémer, evitant d'aquesta manera que aparega el teclat, solament es mostrarà el `dataPicker`.

Vegem el codi:

```

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        binding.dateText.setOnClickListener {
            val datePicker = MaterialDatePicker.Builder.datePicker()
                .setTitleText("Fecha de nacimiento")
                .setSelection(MaterialDatePicker.todayInUtcMilliseconds())
                .build()
            datePicker.addOnPositiveButtonClickListener {
                binding.dateText.setText(datePicker.headerText.toString())
            }
            datePicker.show(supportFragmentManager, "")
        }
    }
}

```

💡 De vegades serà necessari que el DatePicker iniciï amb una data establida per l'aplicació, l'usuari, etc. per a això es necessitarà usar el següent codi que ens permet inicialitzar amb un string una instància de tipus Date a un format especificat, per a posteriorment assignar-la a la propietat **setSelection** de DatePicker en milisegons:

```

val format = SimpleDateFormat("dd/MM/yyyy")
val fecha = format.parse("14/03/2020")
val fechaInicio = Calendar.getInstance()
fechaInicio.time = fecha
val datePicker = MaterialDatePicker.Builder.datePicker()
    .setTitleText("Fecha de nacimiento")
    .setSelection(fechaInicio.timeInMillis)
...

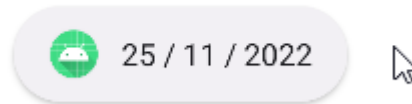
```

💡 També podem necessitar realitzar l'operació contrària, és a dir extraure la data seleccionada per l'usuari en el DatePicker però traent-la amb format de data, per a això podríem fer el codi següent, en este cas hem compost una cadena amb les dades per a mostrar el Toast com el que es veu baix.

```

datePicker.addOnPositiveButtonClickListener {
    val date = Date(it)
    val calendar=Calendar.getInstance()
    calendar.time=date
    var dia = calendar.get(Calendar.DAY_OF_MONTH)
    var mes = calendar.get(Calendar.MONTH)
    var anyo = calendar.get(Calendar.YEAR)
    var fechaSeleccionada="$dia / $mes / $anyo"
    Toast.makeText(this,fechaSeleccionada, Toast.LENGTH_SHORT).show()
    ...
}

```



👉 Practica modificant l'exercici anterior de manera que si el camp d'edició té una data, la mostra en obrir el `dataPicker` .

- Afegir dos camps de data més al disseny amb esquema **Data d'Entrada** i **Data d'eixida** similar als utilitzats en una reserva d'hotel, usant un `MaterialDatePicker.Builder.dateRangePicker` per a la seua implementació.

Progress indicators

Els **Progress indicators** informen els usuaris sobre l'estat d'un procés en execució, com carregar una aplicació, enviar un formulari o guardar actualitzacions, utilitzen animacions per a captar l'atenció. Mai són simplement decoratius.

Material Design ofereix dos tipus d'indicadors de progrés visualment diferents: indicadors de progrés lineals i circulars. Convé utilitzar sempre el mateix tipus per a la mateixa acció.


Els indicadors de progrés poden ser determinats (quan es coneix o es pot calcular la finalització del procés) o indeterminats (no es coneix la fi de la tasca o no és necessari indicar quant temps durarà la mateixa).

A mesura que es disposa de més informació sobre un procés, un **Progress indicator** pot passar d'un estat indeterminat a un determinat.

```
<!-- Linear progress indicator -->
<com.google.android.material.progressindicator.LinearProgressIndicator
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
<!-- Circular progress indicator -->
<com.google.android.material.progressindicator.CircularProgressIndicator
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

A qualsevol dels dos tipus se li pot dir que és indeterminat amb la propietat

`android:indeterminate="true"` , per defecte és determinat.

 Per ara utilitzarem Progress Indicators Indeterminats fins que ens ensenyem a usar corrutines, més endavant.