Apunts

Descarregar aquests apunts

Tema 5. Interfície d'Usuari I

Índex

ViewBinding

Si ja eres coneixedor d'Android, coneixeràs findViewById, que permet vincular les vistes de l'aplicació amb les classes, i així poder accedir a determinades propietats dels widgets de les vistes.

Des de la versió 3.6 d'Android Studio, és possible utilitzar View Binding per a enllaçar els elements dels layouts amb les classes.

Si estem treballant amb una versió d'Android Studio superior a la 4.0 la configuració del archiu `build.gradle` a nivell de `Module:app` és la següent:

```
    android {
        ...
        viewBinding {
            enabled = true
        }
    }
```

Una vegada habilitada la vinculació de vista per a un projecte, per cada arxiu xml es generarà una classe de vinculació a aquest, que serà utilitzat per a fer referencies a les vistes d'aquest.

Si el nostre arxiu s'anomena `activity_secundaria.axml` la classe de vinculació generada es dirà ActivitySecundariaBinding.

Per a poder utilitzar la vinculació de vistes cal fer el següent en el mètode `onCreate()` de l'activitat:

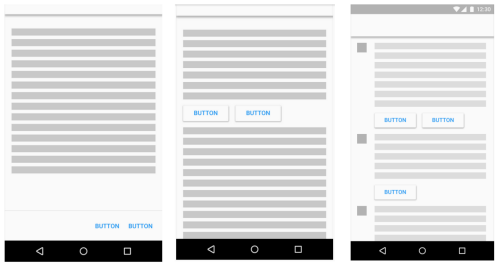
```
private lateinit var binding: ActivitySecundariaBinding

    override fun onCreate(savedInstanceState: Bundle) {
        super.onCreate(savedInstanceState)
        binding = ActivitySecundariaBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)
    }
```

A partir d'ara podem fer referència a les vistes del nostre xml. Ho veurem a continuació tal com anem veient diferents components.

Botons

Els botons formen una part funcional molt important en qualsevol aplicació. Existeixen diversos tipus estàndard de botons: `Floating Action Button` (botó circular amb una acció molt concreta en la nostra aplicació), `Filled, elevated button` (botó amb relleu amb efecte de pulsació i fons color), `Filled, unelevated button` (botó amb relleu sense efecte de pulsació i color fons), `Outlined button` (botó fons transparent i bord), `Text button` (té un fons transparent amb text en color) i `Icon button` (incorpora una icona al botó, pot ser amb text i sense text).

L'elecció d'un botó o un altre dependrà de la importància que se li vulga donar al mateix i de la disposició d'aquest.



OK OK OK

Ací tenim un enllaç on es troba informació sobre aquest component que ofereix Material Design.

Per a definir un botó d'un tipus o un altre assignarem un estil al botó.

Crearem un projecte nou que anomenarem **EjemploBotones** i on anirem incorporant els diferents components que veurem a continuació.

Filled, elevated button

Botó elevat amb fons i efecte de pulsació. S'utilitza per a accions finals del tipus **Guardar** o **Confirmar**. Si no s'especifica cap atribut d'estil per a aquest element, aquest és l'estil que s'utilitzarà per defecte.

Utilitzarem com a contenidor principal un `LinearLayout` amb orientació `vertical`.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    android:orientation="vertical">

    <com.google.android.material.button.MaterialButton
        android:id="@+id/material_button_salir"
12      style="@style/Widget.MaterialComponents.Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_label_salir"
        android:layout_gravity="center"/>
</LinearLayout>
```

> 📌 El `colorPrimary` del seu tema proporciona el fons predeterminat per al botó.

Filled, unelevated button

```
<com.google.android.material.button.MaterialButton
        android:id="@+id/material_button_flat"
3       style="@style/Widget.MaterialComponents.Button.UnelevatedButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_label_flat"
        android:layout_gravity="center"/>
```

Outlined button

```
<com.google.android.material.button.MaterialButton
        android:id="@+id/material_button_outlined"
        style="@style/Widget.MaterialComponents.Button.OutlinedButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_label_outlined"
        android:layout_gravity="center"/>
```

Text button

```
<com.google.android.material.button.MaterialButton
        android:id="@+id/material_button_text"
        style = "@style/Widget.MaterialComponents.Button.TextButton"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "@string/button_label_text"
        android:layout_gravity="center"/>
```

Icon button

```
<com.google.android.material.button.MaterialButton
        android:id="@+id/material_button_icon"
        style = "@style/Widget.MaterialComponents.Button.Icon"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        app:icon = "@drawable/ic_email_black_24dp"
        android:layout_gravity="center"/>
```

> 📌 Amb l'atribut **`app:icon`** (línia 6) indiquem el recurs que hi ha que carregar en el botó. Hi ha una sèrie de propietats que podem usar quan usem **`IconButton`**. El **`app:iconSize="15dp"`** per a ample i alt de la icona. Podem agregar l'espai entre la icona i el text amb **`app:iconPadding="10dp"`**. També hi ha una opció per a canviar el color de la icona amb l'atribut **`iconTint`**. Per a canviar la posició de la icona, utilitze l'atribut **`iconGravity`**. També podem aplicar un efecte d'arrodoniment a les vores del botó (menys a **`TextButton`**) amb la propietat **`app:cornerRadius`**.

[Ací tenim un enllaç on descarregar icones que ofereix Google.]
(https://fonts.google.com/icons?icon.query=mail)

Per a incorporar text al botó amb icona:

```
    <com.google.android.material.button.MaterialButton
           android:id="@+id/material_button_icon_text"
           style = "@style/Widget.MaterialComponents.Button.TextButton.Icon"
           android:layout_width = "wrap_content"
           android:layout_height = "wrap_content"
           app:icon = "@drawable/ic_email_black_24dp"
           android:text = "@string/button_label_icon"
           android:layout_gravity="center"
9          app:iconGravity="end"/>
```

📌 Amb l'atribut **`app:iconGravity`** (línia 9) indiquem la posició de la icona respecte al text.

Manejar esdeveniments botó

Per a gestionar la pulsació realitzada sobre un botó tenim diverses possibilitats: usar atribut `onClick` del botó, un listener anònim o implementar la interfície `View.OnClickListener`. El més eficient és implementar la interfície, encara que trobarem molta documentació que implementa el listener anònim.

Vegem el codi de la nostra activitat:

```
class MainActivity : AppCompatActivity(), View.OnClickListener {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        binding.materialButtonFlat.setOnClickListener(){
            Toast.makeText(applicationContext,"Pulsaste el botón FLAT",Toast.LENGTH_LONG)
        }

        binding.materialButtonOutlined.setOnClickListener(this)
        binding.materialButtonText.setOnClickListener(this)
        binding.materialButtonIcon.setOnClickListener(this)
        binding.materialButtonIconText.setOnClickListener(this)
    }

    override fun onClick(v: View?) {
        when (v) {
            binding.materialButtonOutlined-> visualizarToast("OUTLINED")
            binding.materialButtonText-> visualizarToast("TEXT")
            binding.materialButtonIcon-> visualizarToast("ICON")
            binding.materialButtonIconText-> visualizarToast("ICON CON TEXT")
            else -> {
                visualizarToast("OTRA COSA MARIPOSA")
            }
        }
    }

    fun visualizarToast(view: View) {
        Toast.makeText(this,"Pulsaste el botón RAISED",Toast.LENGTH_LONG).show()
    }

    fun visualizarToast(s: String) {
        Toast.makeText(this,"Pulsaste el botón "+ s,Toast.LENGTH_LONG).show()
    }
}
```

📌 **Manejar esdeveniments botó**:

- `android:onClick` . Aquest atribut incorporat en la definició d'un botó en el XML permet assignar un mètode públic que s'executarà quan aquest es prema (línia 32). Ho aplicarem en el nostre primer botó `raised_button` i visualitzarem un `toast` (línia 33) indicant la pulsació d'aquest:

```
    <com.google.android.material.button.MaterialButton
        android:id="@+id/material_button_raised"
        style="@style/Widget.MaterialComponents.Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_label_raised"
        android:layout_gravity="center"
8       android:onClick="visualizarToast"/>
```

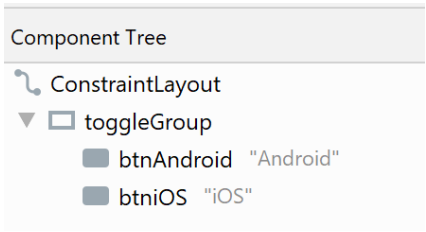
- `listener anònim` . Programem el listener anònim directament sobre la vista i implementem el mètode `setOnClickListener()` (línia 10). Incloem aquest codi en el `onCreate()` de l'activitat:
- implementant interfície `setOnclickListener` en l'activitat (línia 1), registrar les vistes que seran objecte d'implementació a través de la interfície (línies 14-17) i gestionar la pulsació (línies 20-30 per a la resta de botons). La línia 36 és una funció per a visualitzar un `string` en un `toast` , de manera que per a cada botó puguem traure un text personalitzat del botó premut.

Toggle Button

No és més que un conjunt de botons amb un mateixos estil (`MaterialComponents`). En principi poden seleccionar-se més d'un botó, però podem fer que solament hi haja un seleccionat amb la propietat `app:singleSelection` .

l L'aspecte d'un `ToggleButton` amb dos botons és el següent:

Per a implementar aquesta vista hem d'incorporar els següents elements:



El xml corresponent a aquesta activitat seria el següent:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.
    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">

9   <com.google.android.material.button.MaterialButtonToggleGroup
        android:id="@+id/toggleGroup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
14      app:checkedButton="@id/btnAndroid"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
19      app:singleSelection="true">

        <Button
            android:id="@+id/btnAndroid"
            style="@style/Widget.MaterialComponents.Button.OutlinedButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Android" />

        <Button
            android:id="@+id/btniOS"
            style="@style/Widget.MaterialComponents.Button.OutlinedButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="iOS" />
    </com.google.android.material.button.MaterialButtonToggleGroup>
</androidx.constraintlayout.widget.ConstraintLayout>
```

> 📌 **Aclariments**:
>
> - **Línia 9**. Definició del `MaterialButtonToggleGroup`
> - **Línies 14**. Especifiquem quin botó dels quals defineixen el `ToggleButton` es ressaltarà com marcat.

- **Línia 19** amb aquesta propietat `app:singleSelection` amb valor `true` indiquem que únicament pot prémer-se un botó simultàniament.

El listener que controla aquesta vista és `addButtonChekdelListener()` (línia 10) encara que també podem col·locar el listener sobre cada botó (línia 23) accedint prèviament a la comprovació de quin identificador del `ToggleButton` s'ha premut (línia 21). Vegem totes dues formes en el següent 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)

10      binding.toggleGroup.addOnButtonCheckedListener { group, checkedId, isChecked ->
            if (isChecked) {
                if (checkedId == R.id.btnAndroid) {
                    Toast.makeText(applicationContext, "Pulsaste ANDROID", Toast.LENGTH_S
                }
                /*if (checkedId == R.id.btniOS) {
                    Toast.makeText(applicationContext, "Pulsaste IOS", Toast.LENGTH_SHORT
                }*/
            }
        }

21      val buttonId = binding.toggleGroup.checkedButtonId
        //listener del botón
23      binding.btniOS.setOnClickListener {
            Toast.makeText(applicationContext, "Pulsaste IOS", Toast.LENGTH_SHORT).show()
        }
    }
}
```

Floating Action Button (FAB)

Un botó d'acció flotant (FAB) executa l'acció principal o més comuna en l'activitat. Es caracteritza per tindre una forma circular amb una icona en el centre normalment animat. Els FAB venen en tres tipus: regulars, mini i estesos (amb una etiqueta de text).

Creem un projecte que es cride **EjemploFAB** i editem el xml de l'activitat:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.
   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.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:src="@drawable/ic_add"
        app:fabSize="normal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

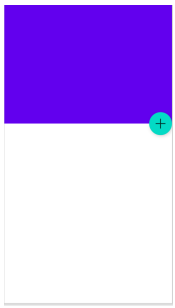
📌 **Aclariments**:

- **Línia 9**. Definició del `FAB`
- **Línies 13-16**. Especifiquem els marges del `FAB` .
- **Línia 17** especifiquem la icona que tindrà el `FAB` .
- **Línia 18** especifiquem el tipus.
- **Línies 19-21**. ancoratges del `FAB` respecte al contenidor.

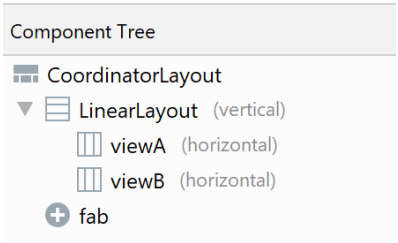
És molt interessant l'aplicar animacions al `FAB`, per exemple una rotació de 45 en cada pulsació. Vegem com quedaria el codi complet d'aquest projecte:

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    var click = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        binding.fab.setOnClickListener { view ->
            click = !click
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                val interpolador = AnimationUtils.loadInterpolator(baseContext,
                    android.R.interpolator.fast_out_slow_in)
                view.animate()
                    .rotation(if (click) 45f else 0f)
                    .setInterpolator(interpolador)
                    .start()
                Toast.makeText(applicationContext, "Pulsaste FAB", Toast.LENGTH_SHORT).sh
            }
        }
    }
}
```

El `FAB` també es pot utilitzar com un element diferenciador entre dues vistes:



Per a implementar aquesta vista hem d'incorporar els següents elements:



El xml corresponent a aquesta activitat seria el següent:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/viewA"
            android:layout_weight="0.6"
            android:background="@color/purple_500"
            android:orientation="horizontal"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/viewB"
            android:layout_weight="0.4"
            android:background="@android:color/white"
            android:orientation="horizontal"/>
    </LinearLayout>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:clickable="true"
        android:layout_marginBottom="16dp"
        android:src="@drawable/ic_add"
        app:fabSize="normal"
        android:layout_margin="20dp"
        app:layout_anchor="@id/viewB"
        app:layout_anchorGravity="top|end"/>
</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

📌 **Aclariments**:

- **Línia 2**. El contenidor `CoordinatorLayout` ens facilita la superposició de elements en la vista i desplaçaments automàtics dels elements quan això succeeix (quan la `snacbar` interfereix amb el `FAB`).
- **Línies 13 i 21**. Tenim dues vistes en l'activitat identificades com `viewA` i `viewB` .

- **Línia 38** especifiquem amb l'atribut `app:layout_anchor` que vista s'ancorarà el `FAB` .
- **Línia 39** amb l'atribut `app:layout_anchorGravity` especifiquem la posició del `FAB` respecte a la vista amb la qual s'ha ancorat.

📌 És possible transformar un FAB en un menú d'accions.

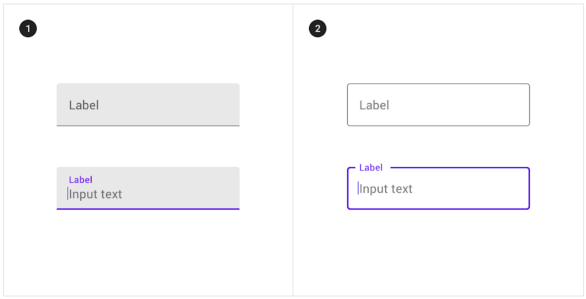
✋ EjercicioResueltoFABExtended

Ací tenim un enllaç on es troba informació sobre FAB i transicions.

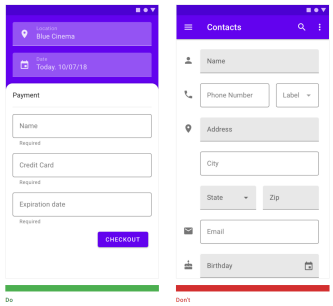
TextInputLayout etiquetes flotants

La introducció de text es realitza amb el control `TextInputEditText`, les propietats bàsiques són: `hint` que permet que aparega un text dins del control que desapareixerà quan es punxe sobre el mateix i també és possible incloure una icona en el control.

Posteriorment s'ha introduït un altre control que permet interactuar amb un `TextInputEditText` de manera que el `hint` del mateix en comptes de desaparéixer es desplaça automàticament a la part superior de la vista. Aquest control es diu `TextInputLayout` i pot tindre dos aspectes diferents que es definen amb un estil `FilledBox` (1) i `OutlinedBox` (2).



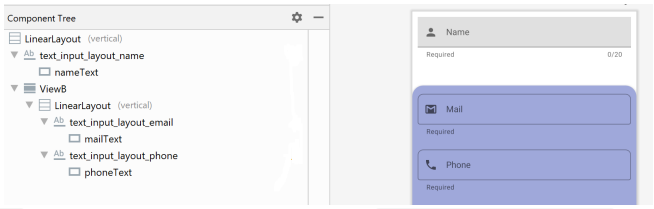
No convé utilitzar tots dos en la mateixa vista almenys que hi haja un element separador entre ells:



Ací tenim un enllaç on es troba informació sobre Text Fields que ofereix Material Design.

Crearem un projecte nou que anomenarem **EjemploTextFields** on incorporarem els diferents vistes que anirem veient.

La vista que volem obtindre i els components utilitzats es mostren en la imatge següent:



Utilitzarem un `LinearLayout` que contindrà el primer `TextInputLatout` amb el primer `TextInputEditText` corresponent al `name` i posteriorment inclourem un `CardView` amb els altres dos elements d'edició (`Mail` i `Phone`) que estan dins d'un altre `LinearLayout`.

Vegem l'arxiu xml que defineix aquesta vista i comentem els aspectes més rellevants:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    android:orientation="vertical">

    <com.google.android.material.textfield.TextInputLayout
        style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"
        android:id="@+id/text_input_layout_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:layout_marginStart="10dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="10dp"
        android:hint="Name"
        app:startIconDrawable="@drawable/account"
        app:helperText="Required"
        android:layout_weight="1"
        app:counterEnabled="true"
        app:counterMaxLength="20">

        <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/nameText"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="textPersonName"
                android:singleLine="true" />
        </com.google.android.material.textfield.TextInputLayout>

    <androidx.cardview.widget.CardView
        android:id="@+id/ViewB"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="50dp"
        app:cardBackgroundColor="#9FA8DA"
        android:layout_weight="10"
        app:cardCornerRadius="20dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/text_input_layout_email"
                style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginTop="10dp"
                android:layout_marginEnd="10dp"
                android:layout_marginBottom="10dp"
                android:hint="Mail"
                app:boxCornerRadiusTopEnd="10dp"
                app:boxCornerRadiusTopStart="10dp"
                app:helperText="Required"
                app:startIconDrawable="@drawable/gmail">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/mailText"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textEmailAddress"
                    android:singleLine="true"/>
            </com.google.android.material.textfield.TextInputLayout>

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/text_input_layout_phone"
                style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginTop="10dp"
                android:layout_marginEnd="10dp"
                android:layout_marginBottom="10dp"
                android:hint="Phone"
                app:boxCornerRadiusTopEnd="10dp"
                app:boxCornerRadiusTopStart="10dp"
                app:helperText="Required"
                app:startIconDrawable="@drawable/phone">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/phoneText"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textEmailAddress"
                    android:singleLine="true"/>
            </com.google.android.material.textfield.TextInputLayout>

            <LinearLayout android:id="@+id/area_nombre"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentTop="true"
                android:orientation="horizontal">

                <ImageView android:id="@+id/img_cliente"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_vertical"
```

```
                android:src="@drawable/outline_credit_card_24" />

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/text_input_layout_card"
                style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginTop="10dp"
                android:layout_marginEnd="10dp"
                android:layout_marginBottom="10dp"
                android:hint="Card"
                app:boxCornerRadiusTopEnd="10dp"
                app:boxCornerRadiusTopStart="10dp">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/cardText"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textEmailAddress"
                    android:singleLine="true"/>
                </com.google.android.material.textfield.TextInputLayout>
            </LinearLayout>
        </LinearLayout>
    </androidx.cardview.widget.CardView>
</LinearLayout>
```

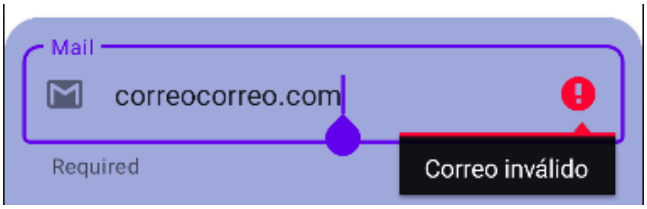
📌 Aclariments:

- **Línia 11** definim l'estil del nostre control com `FilledBox`
- **Línies 15-18** establim els marges per a aconseguir l'efecte de separació desitjat.
- **Línia 19** amb la propietat `android:hint` establim el text que servirà per a definir la dada que es vol introduir.
- **Línia 20** amb la propietat `app:startIconDrawable` definim la icona que apareixerà dins de la vista i la seua posició (`app:endIconDrawable` si ho volem al final).
- **Línia 21** amb `app:helperText` establim un text aclaridor davall de la vista
- **Línies 23-24** amb `app:counterEnabled="true"` permetem que aparega un comptador de caràcters associat a la vista i amb `app:counterMaxLength="20"` definim el màxim de caràcters a introduir, una vegada superats el control indica l'error per superar aquest màxim.
- **Línies 28-32** es defineix el `TextInputEditText`
- **Línia 50** definim l'estil de l'el nostre control com `OutlinedBox`
- **Línies 58-59** amb aquestes propietats definim les vores del control `app:boxCornerRadiusTopEnd ...`

- Es possible afegir un botó d'acció a la vista `app:endIconMode="clear_text"` , un altre valor possible és `password_toggle` no permetent que es veja l'entrada de text.

Les guies de Material Design per a camps d'edició permeten també que es visualitze una icona al costat del `TextInputLayout` però la icona deu canviar de color quan el `TextInputEditText` rep el focus, per a això s'haurà d'usar DrawableCompat.setTint sobre la icona.

Este control oferix també la possibilitat de mostrar errors, molt útil per exemple per a mostrar errors de validació.



Per a això usarem el mètode `setError()` del control `TextInputEditText`, que ens servirà per a indicar el text de l'error o per a eliminar-ho si passem null com a paràmetre. Podria ser un mètode com el següent a què es cridaria al voler validar les dades:

```
fun validarMail():Boolean
{
   if(!Patterns.EMAIL_ADDRESS.matcher(binding.mailText.text.toString()).matches())
   {
       binding.mailText.setError("Correo inválido")
       return false
   }
   else
   {
       binding.mailText.setError(null)
       return true
   }
}
```

📌 Aclariments: Per a validar el text ho farem a través de la classe **Patterns**, la qual conté mètodes per a l'ús d'expressions regulars.

AutoComplete TextView

Un control de text editable que mostra suggeriments mentre l'usuari està escrivint. La llista de suggeriments es mostra en un desplegable en el qual l'usuari pot triar un element per a emplenar el contingut del quadre d'edició.

La llista de suggeriments s'obté d'un adaptador de dades i apareix només després d'un nombre determinat de caràcters definit per el `completionThreshold` (línia 20). Com veiem en l'exemple, pot

estar embolicat en un `TextInputLatout` amb un estil `ExposedDropdownMenu` que permet amb l'aparició d'una icona fletxa el desplegar la llista (línia2).

```
    <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/text_input_layout_pais"
3               style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox.ExposedDr
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="10dp"
                android:layout_marginTop="10dp"
                android:layout_marginEnd="10dp"
                android:layout_marginBottom="10dp"
                android:hint="Pais"
                app:boxCornerRadiusTopEnd="10dp"
                app:boxCornerRadiusTopStart="10dp">

                <com.google.android.material.textfield.MaterialAutoCompleteTextView
                    android:id="@+id/paisText"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textAutoComplete"
                    android:singleLine="true"
20                  android:completionThreshold="2"/>
        </com.google.android.material.textfield.TextInputLayout>
```

Per a poder emplenar la llista de selecció per al control `AutoCompleteTextView`, podem fer-ho de dos formes:

- Per mitjà de Xml, per a això necessitem definir un arxiu de recursos en `res/values/` que cridem `country.xml` on definirem els valors de la llista:"

  ```
  <?xml version="1.0" encoding="utf-8"?>
  <resources>
      <string-array name="paises">
          <item></item>
          <item>España</item>
          <item>Ecuador</item>
          <item>Estados Unidos</item>
          <item>Eslovaquia</item>
          <item>Eslovenia</item>
          <item>Islandia</item>
          <item>Islas Marshall</item>
          <item>Islas Salomon</item>
      </string-array>
  </resources>
  ```

  I en el mètode `onCreate()` de l'activitat, crearem l'adaptador referenciant a les dades definits en el XML:"

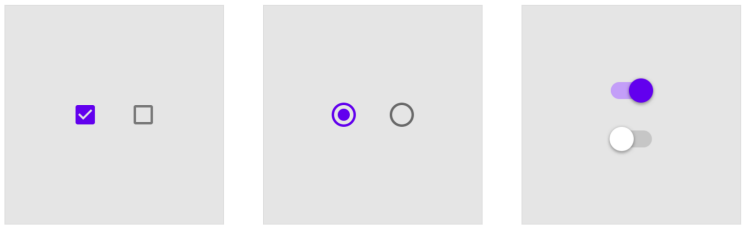
```
    val adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1,
                            resources.getStringArray(R.array.paises))
    binding.paisText.setAdapter(adapter)
```

- Creant una llista de dades per mitjà de codi, esta forma és més flexible per a afegir dades noves en cas de ser necessari:

```
var lista= listOf<String>("España","Ecuador", "Estados Unidos",
                            "Eslovaquia","Eslovenia","Islandia",
                            "Islas Marshall","Islas Salomon")
   val adapter= ArrayAdapter(this,android.R.layout.simple_list_item_1,lista)
   binding.paisText.setAdapter(adapter)
```

Controls de selecció

Són controls que permeten seleccionar un element (`RadioButton`) o varis (`CheckButton`) o entre dos estats (`Switch`) per a realitzar una entrada de dades.



Checkboxes Radio buttons Switches

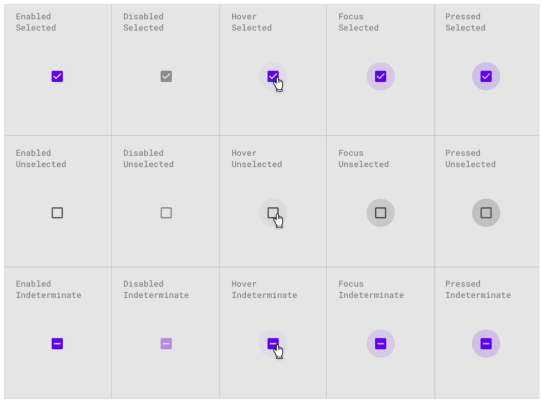
CheckBox

Els `CheckBox` permeten als usuaris seleccionar un o més elements d'un conjunt. Els `CheckBox` poden activar o desactivar una opció.

Les caselles de verificació poden tindre una relació pare-fill amb altres caselles de verificació.

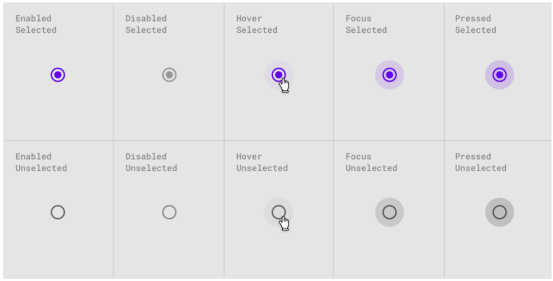
- Quan la casella de verificació principal està marcada, totes les caselles de verificació secundàries estan marcades.
- Si una casella de verificació principal està desmarcada, totes les caselles de verificació secundàries estan desmarcades.
- Si algunes caselles de verificació secundàries, però no totes, estan marcades, la casella de verificació principal es converteix en una casella de verificació indeterminada.

Les caselles de verificació poden estar seleccionades, no seleccionades o indeterminades. Les caselles de verificació tenen estats habilitat, deshabilitat, flotant, enfocat i pressionat.



RadioButton

Els botons d'opció permeten als usuaris seleccionar una opció d'un conjunt.



Switches

Els interruptors activen o desactiven l'estat d'un sol element.

> ✋EjercicioResueltoControlesSeleccion

SnackBar

La `snackBar` informen els usuaris sobre un procés que una aplicació ha realitzat o realitzarà. Apareixen temporalment en la part inferior de la pantalla. No han d'interrompre l'experiència de l'usuari i no requereixen l'acció de l'usuari per a desaparéixer. Són similars a un `Toast`. Només es pot mostrar un `snackbar` alhora.

Els ajustos preestablits de duració disponibles són:

- `LENGTH_INDEFINITE` (Mostra la `snackbar` fins que es descarta o fins que es mostra una altra)
- `LENGTH_LONG` (Mostra la `snackbar` durant un període llarg de temps)

- `LENGTH_SHORT` (Mostre la `snackbar` per un període de temps curt)
-

```
Snackbar.make(findViewById(R.id.constraintLayout),"La acción seleccionada se ha realizado cor
```

SnackBar con acción

Per a agregar una acció, use el mètode `setAction`. Les `snackBar``** es descarten automàticament quan es fa clic en l'acció.

```
Snackbar.make(findViewById(R.id.constraintLayout),"La acción seleccionada se ha realizado cor
    .setAction("ACEPTAR") {
    // Responds to click on the action
    }
    .show()
```

Desenvolupar un exercici que es cride **EjemploSnackBar** i provar aquests elements. Recomane que col·loqueu una activitat amb tres botons (tres tipus de `snackBar`) i un `FAB`.

Què succeeix quan ho executeu? Se us superposen les `snackBar` al `FAB`? Vegem com resoldre-ho...

Descartar SnackBar

Perquè la `snackBar` aparega per davall del `FAB` i aquest tinga un desplaçament cap amunt, és necessari que el contenidor principal siga un `CoordinatorLayout`, ja utilitzat en temes anteriors.

A més el fet que utilitzem aquest element ens proporcionarà també una acció addicional sobre la `snackBar`, concretament el poder descartar-la amb un gest. Aquestes interaccions i animacions automàtiques són un element molt potent visualment parlant.