Disseny de la interfície d'usuari

Projecte Integrat de Software (PIS)

Universitat de Barcelona

victor.campello@ub.edu
carlos.martinisla@ub.edu

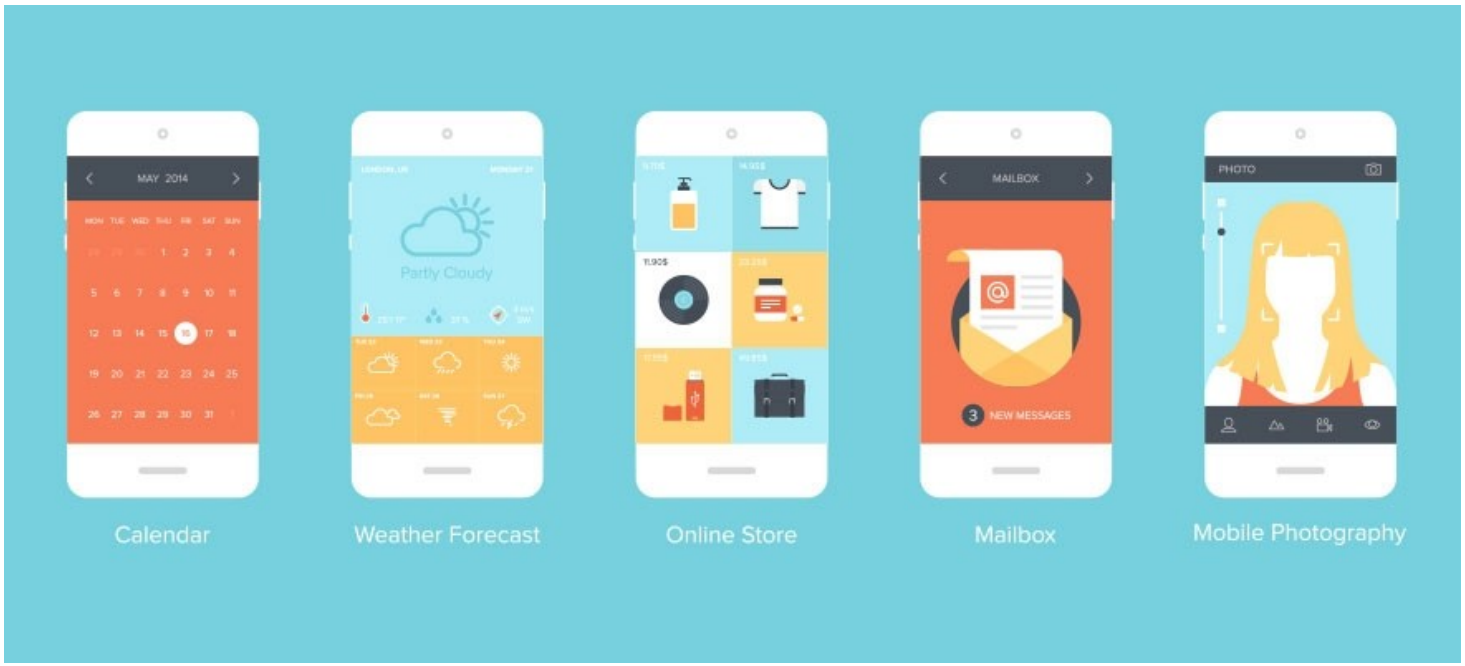
22-24 de Febrer de 2022

Overview

1. Disseny de l'aplicació - Interfície d'usuari
   - Layouts
   - Widgets
   - Estils i temes
   - Material
   - Gradle
2. Intents
   - Cicle de transmissió dels Intents
   - Com crear un Intent

Interfície - *User Interface* (UI)

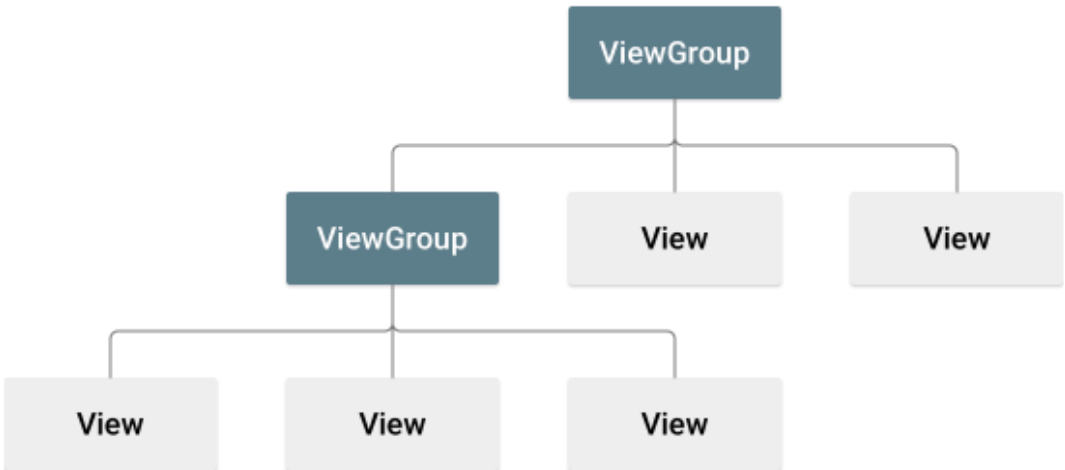
- La interfície d'usuari de la App és tot allò que l'usuari pot veure i amb el que interactua. Android ofereix una varietat de components de UI pre-dissenyats. Nosaltres veurem un subconjunt d'aquests.
- Teniu en compte que crear una UI que l'usuari entengui i disfruti és una crucial.



Classe View

L'estructura de la interfície utilitza la següent jerarquia d'objectes:

- **ViewGroup:** Contenidor invisible que defineix el tamany i posició d'altres *ViewGroup* o *View*. Els *ViewGroup* centrats en posicionar altres *View* s'anomenen "dissenys" o "layouts".
- **View:** Principal classe que defineix tot allò amb que l'usuari pot interactuar (inclós el *ViewGroup*). Es denominen "controls" o "widget".



Aquests elements es construeixen de forma habitual amb recursos XML.

Assignació d'una interfície a una activitat

- La interfície d'una activitat és inicialment buida i se li pot associar un disseny amb la funció *setContentView*.

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

- Utilitzant els recusos XML dissociem la lògica del disseny.
- Una vegada definit, es pot accedir als *View* definits mitjançant la funció *findViewById*.

```
TextView myTV = findViewById(R.id.myTextView);
```

Layouts

- L'SDK d'Android ofereix un seguit de diferents layouts que es poden combinar per crear una interfície tan complexa com vulgueu.

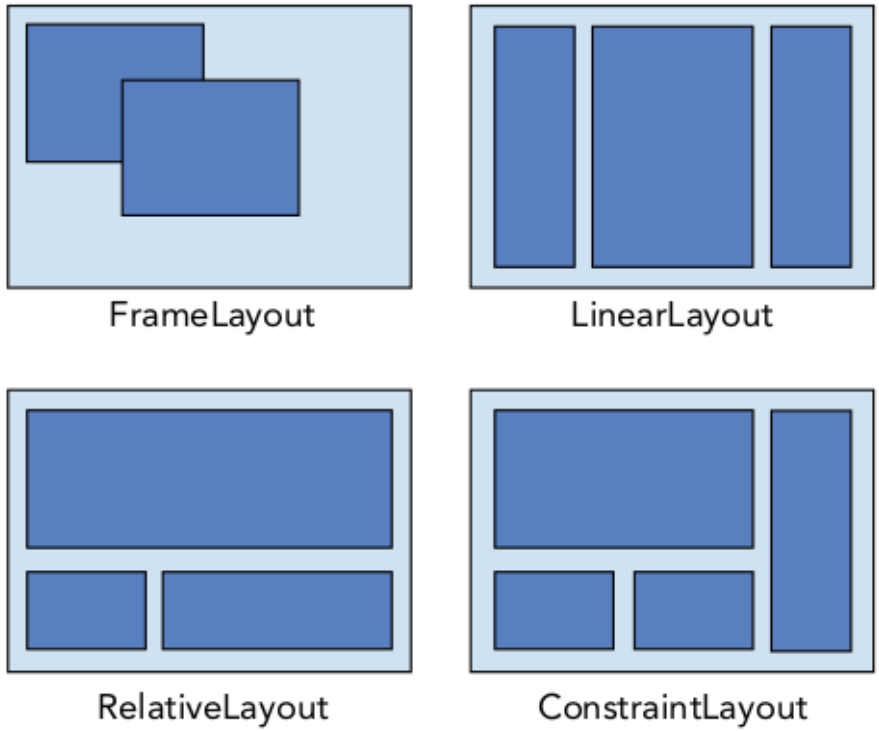
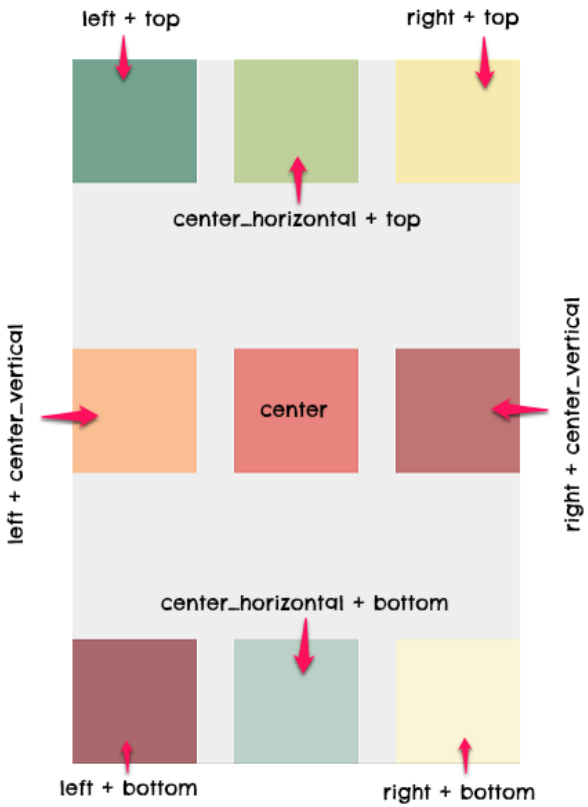


Figure: Source: *Professional Android*, Meier & Lake.

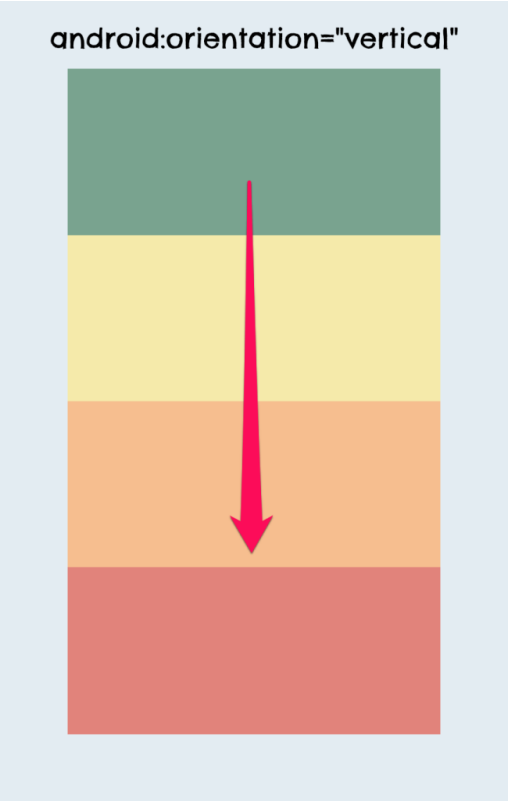
Layouts

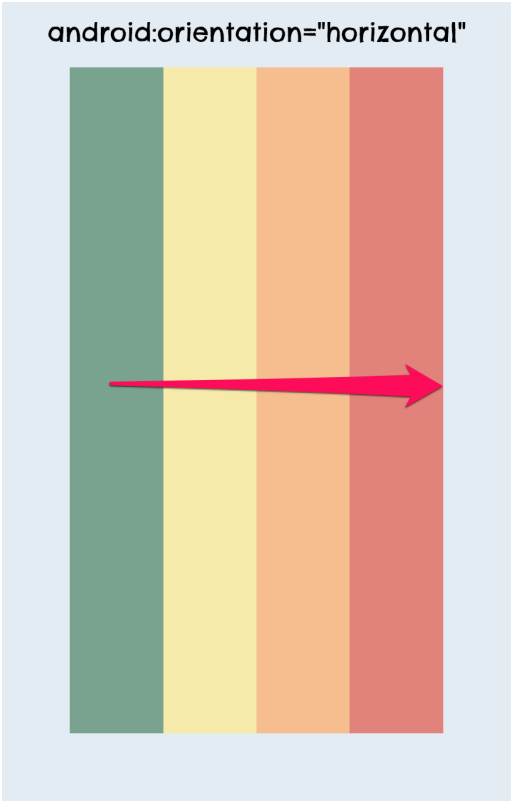
- *FrameLayout*: El més simple. Situa cada View dins del seu marc. Està pensat per un únic objecte així que cada nou element es situa a sobre de l'anterior. L'atribut *layout_gravity* ens permet posicionar els elements.

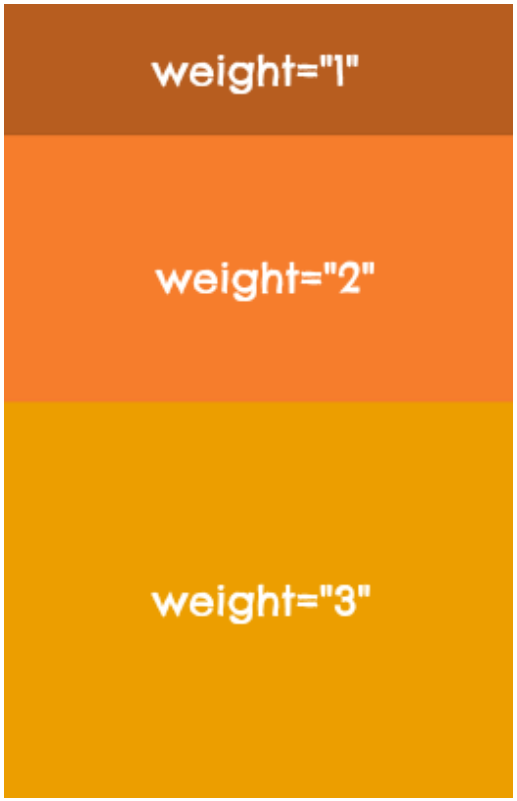


Layouts

- *LinearLayout*: Alinea els elements de forma vertical o horitzontal. Te l'atribute *layout_weight* que controla el pes relatiu de cada element per modificar el seu tamany.

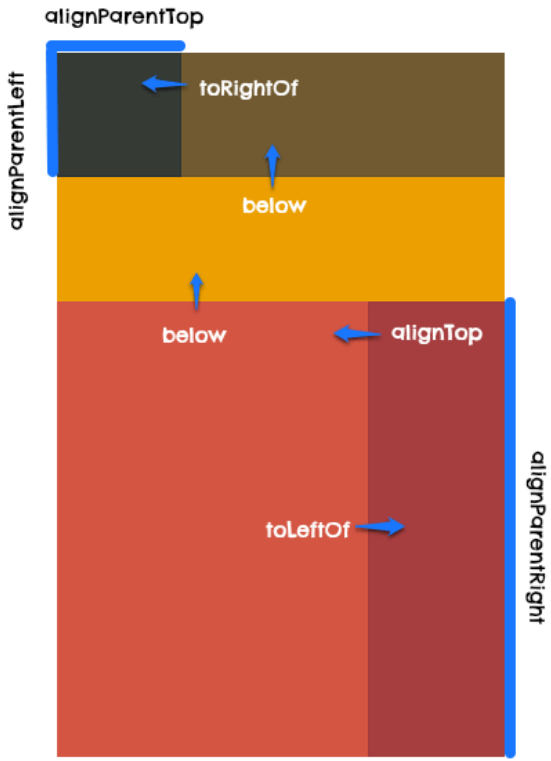






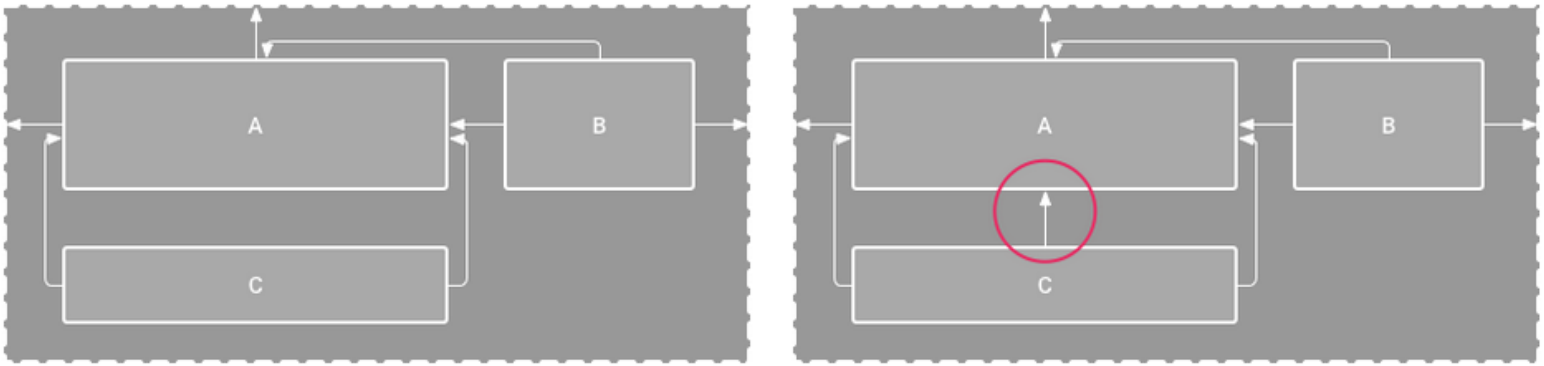
Layouts

- *RelativeLayout*: Un dels més flexibles. Permet definir els Views en relació a altres elements o a les fronteres del layout.



Layouts

- *ContraintLayout*: El més recent (i recomanat). Està pensat per dissenyar interfícies complexes sense necessitar d'utilitzar múltiples layouts. Defineix les posicions dels elements amb lligams.
  - developer.android.com/training/constraint-layout
  - Google Codelab (tutorial)



Edició del Layout

Per editar un layout, hi ha dos mètodes:

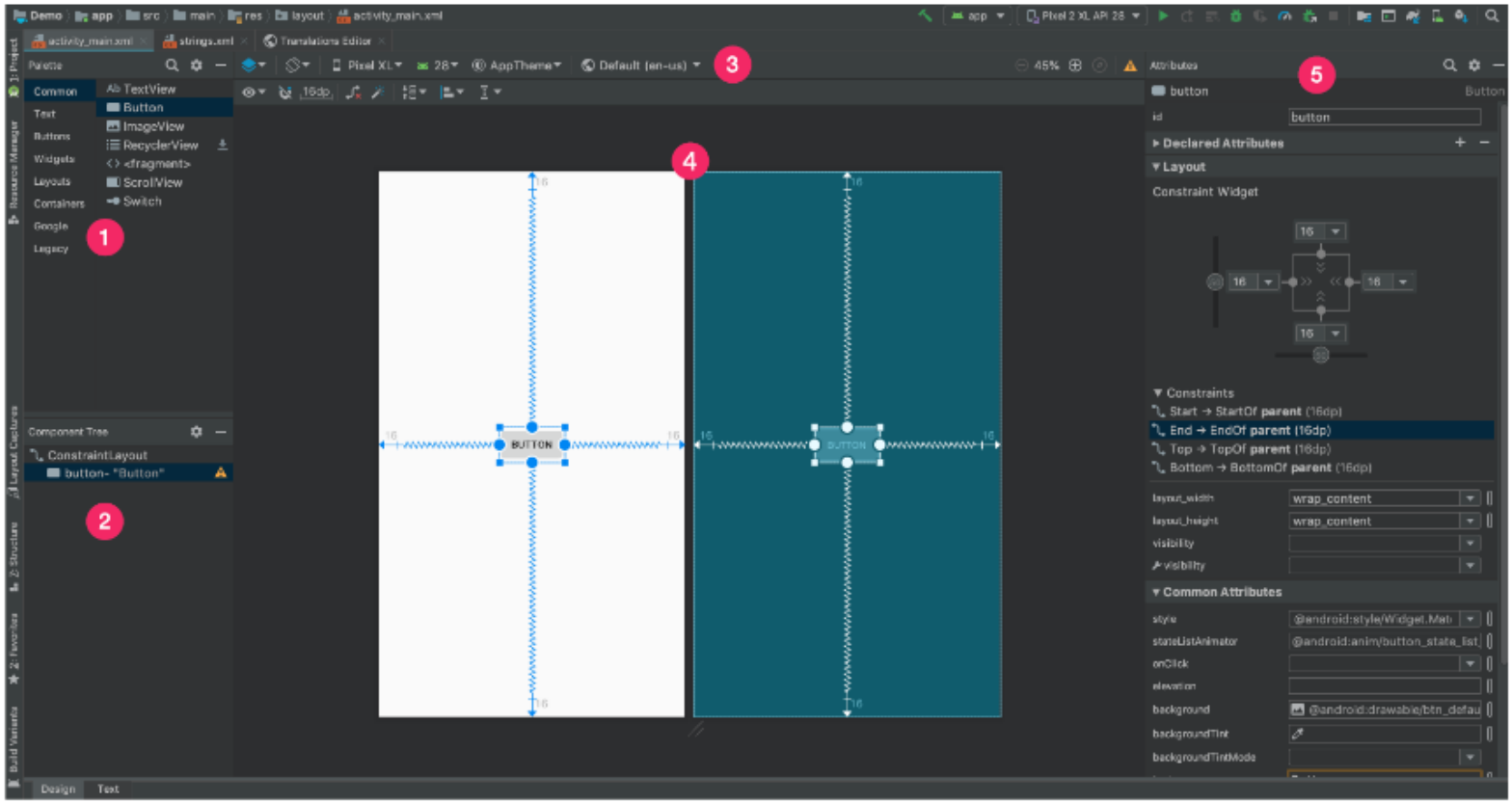
- Amb el codi XML definint directament els elements amb la funció d'autocompletat

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

- Amb el *Layout editor*, un editor per dissenyar de forma interactiva i que genera automàticament el codi XML.

Layout Editor

1. **Palette**: Es la lista de vistas y grupos de vistas que puedes arrastrar a tu diseño.
2. **Component Tree**: Corresponde a la jerarquía de vistas para tu diseño.
3. **Toolbar**: Tiene los botones para configurar la apariencia de tu diseño en el editor y cambiar algunos atributos de diseño.
4. **Design editor**: Corresponde al diseño en la vista Design o Blueprint, o ambas.
5. **Attributes**: Contiene controles para los atributos de las vistas seleccionadas.



Paràmetres importants als layouts

Als diferents layouts definits existeixen paràmetres bàsics que heu de conèixer:

- **android:id="@+id/name"** Identificador únic per a cada element. El símbol @ farà que l'analitzador de XML processi la cadena de caràcters posteriors. Si afegim +, significa que l'id no està creat i es crearà automàticament a l'arxiu R.strings.
- **layout_width, layout_height** Defineixen l'ample i l'alçada de qualsevol *View*. Si fem servir "wrap_content", el tamany s'ajusta al mínim per a una correcta visualització. Si, pel contrari, utilitzem "match_parent", s'ajusta al màxim permès pel pare ("layout").

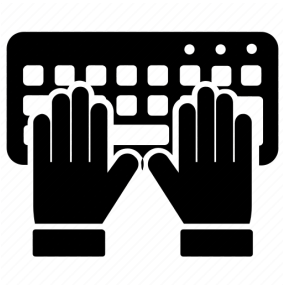
Widgets

Android te una sèrie de widget predefinits. Alguns dels més comuns són:

- *TextView*: per mostrar text.
- *EditText*: per mostrar text editable.
- *ImageView*: per mostrat una única imatge.
- *Toolbar*: mostra un títol i accions freqüents.
- *Button*: element interactiu de pulsació.
- *RecyclerView*: serveix per mostrar una gran quantitat de Views amb desplaçament.
- *CheckBox*: un requadre clickable amb dos estats.
- *RadioButton*: un grup d'opcions seleccionables on només una pot estar activa.
- *ViewPager*: permet passar de forma horitzontal entre diferents Views.

Exercici pràctic

- Proveu a dissenyar la interfície d'una calculadora. Ha de tenir els botons numèrics i les opcions per a cada operació.



Referències

Per realitzar el vostre projecte teniu una sèrie de materials que poden ser molt útils:

- Documentació d'Android: developer.android.com/docs
- Repositori oficial: github.com/android
  - Exemples amb views i widgets.
  - Exemples de diferents components d'arquitectura.
- Tutorials interactius: codelabs.developers.google.com
  - Un exemple per crear un xat. En Kotlin!

Action Bar

Per a agregar una barra d'activitat, s'ha de comprovar primer si existeix la llibreria de compatibilitat

```
class MyActivity : AppCompatActivity() {
  // ...
}
```

S'ha d'evitar que el layout utilitzi la barra d'acció per defecte. Al manifest s'ha de modificar el següent i afegir la ToolBar al diseny de l'aplicació.

```
<application
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"
    />
```

```
<android.support.v7.widget.Toolbar
   android:id="@+id/my_toolbar"
   android:layout_width="match_parent"
   android:layout_height="?attr/actionBarSize"
   android:background="?attr/colorPrimary"
   android:elevation="4dp"
   android:theme="@style/ThemeOverlay.AppCompat.ActionBar"
   app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>
```

Estils i temes

- L'**estil** es defineix com una col·lecció d'atributs d'aparença en una sola View (una activitat concreta).
- El **tema** és un estil aplicat a tota l'aplicació i a totes les vistes.

A continuació es mostra com crear i aplicar un estil.

Crear un estil

Para crear un nuevo estilo o tema, abre el archivo `res/values/styles.xml` de tu proyecto. Para cada estilo que desees crear, sigue estos pasos:

1. Agrega un elemento `<style>` con un nombre que identifique el estilo de forma exclusiva.
2. Agrega un elemento `<item>` para cada atributo de estilo que quieras definir.

   El `name` en cada elemento especifica un atributo que de otro modo usarías como un atributo XML en tu diseño. El valor del elemento `<item>` es el valor de ese atributo.

Por ejemplo, si defines el siguiente estilo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

Puedes aplicar el estilo a una vista de la siguiente manera:

```xml
<TextView
    style="@style/GreenText"
    ... />
```

Aplicar un tema

Puedes crear un tema de la misma manera en que creas estilos. La diferencia es cómo lo aplicas: en vez de aplicar un estilo con el atributo `style` en una vista, aplica un tema con el atributo `android:theme` en la etiqueta `<application>` o en una etiqueta `<activity>` en el archivo `AndroidManifest.xml`.

Por ejemplo, aquí se muestra cómo aplicar el tema "oscuro" de material design de la biblioteca de compatibilidad de Android a toda la app:

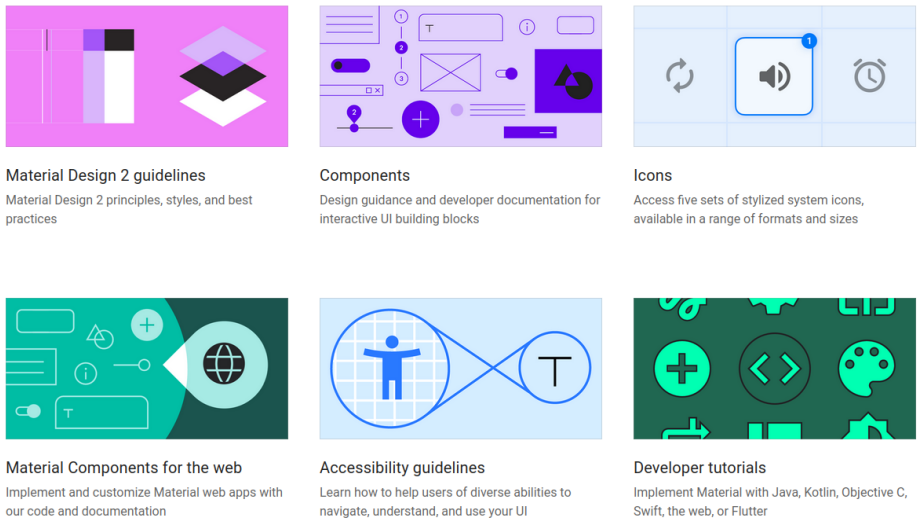
```
<manifest ... >
    <application android:theme="@style/Theme.AppCompat" ... >
    </application>
</manifest>
```

Aquí se muestra cómo aplicar el tema "claro" a una sola actividad:

```
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.AppCompat.Light" ... >
        </activity>
    </application>
</manifest>
```

Material design

- A l'hora de dissenyar interfícies, existeix una guia amb recomanacions i bones pràctiques anomenada **Material** (material.io). S'utilitza en general per disseny web i només per a aplicacions.



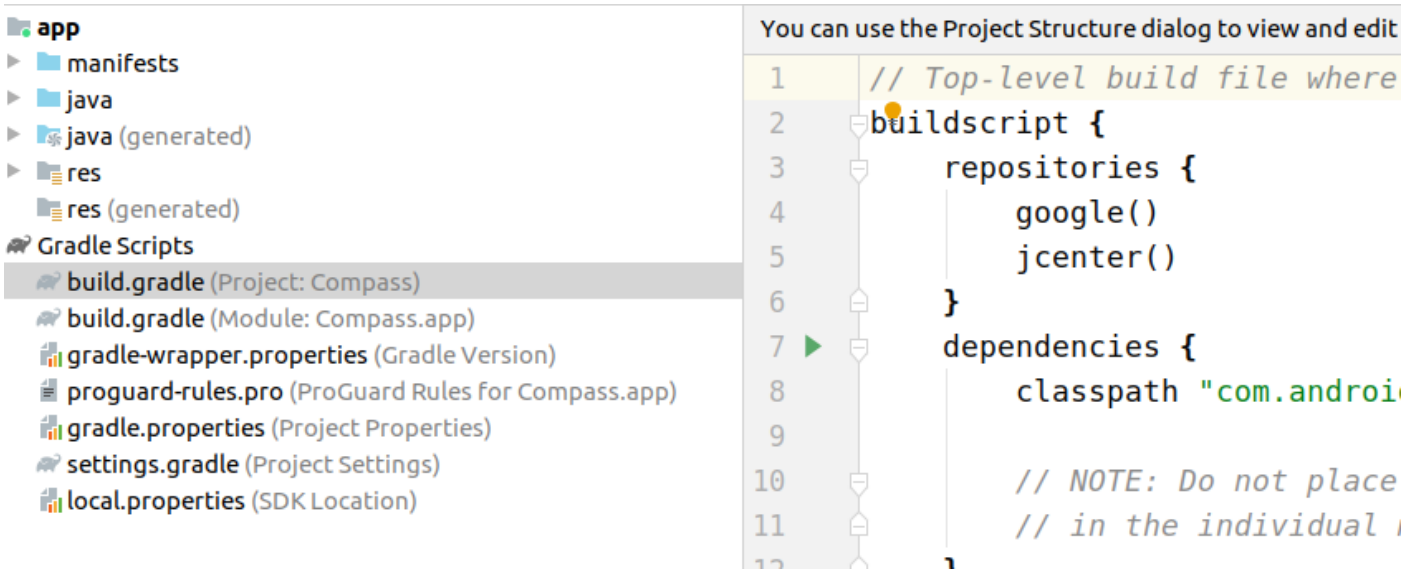
- Per incorporar llibreries com Material a l'aplicació s'ha d'utilitzar el *Gradle*. A l'Android Studio, **Gradle Scripts** > **build.gradle**.

Gradle

Per afegir una llibreria necessitem dues coses:

- Afegir el repositori d'origen de la llibreria.
- Escollir la llibreria en qüestió i la versió d'interés.

Seguint les instruccions de la llibreria **Material**, el repositori necessari és *google()* i s'ha d'afegir al *build.gradle* del projecte. Per defecte, ja hi serà a les versions més modernes d'Android Studio.

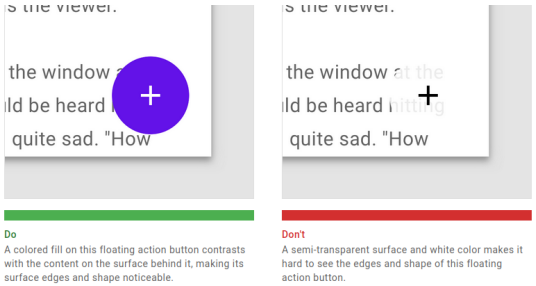


Gradle

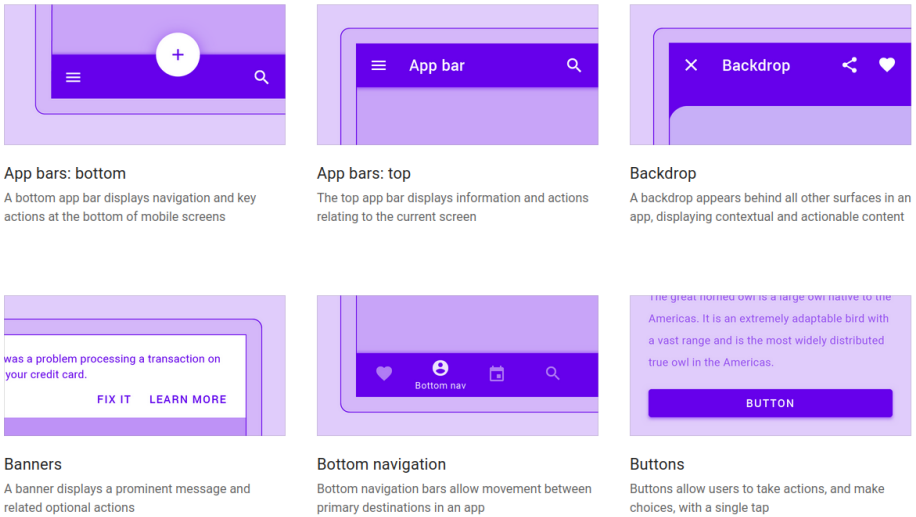
A continuació només cal afegir el nom que ens indica la llibreria al *build.gradle* del mòdul, i sincronitzar el projecte de nou per a afegir la llibreria (des del pop-up o a *File* > *Sync Project with Gradle Files*).

Més "Material"

- Podeu aprendre sobre com dissenyar les vostres interfícies per a que siguin atractives i fàcils d'utilitzar.



- Material te un seguit de instruccions per implementar diferents components de la UI.



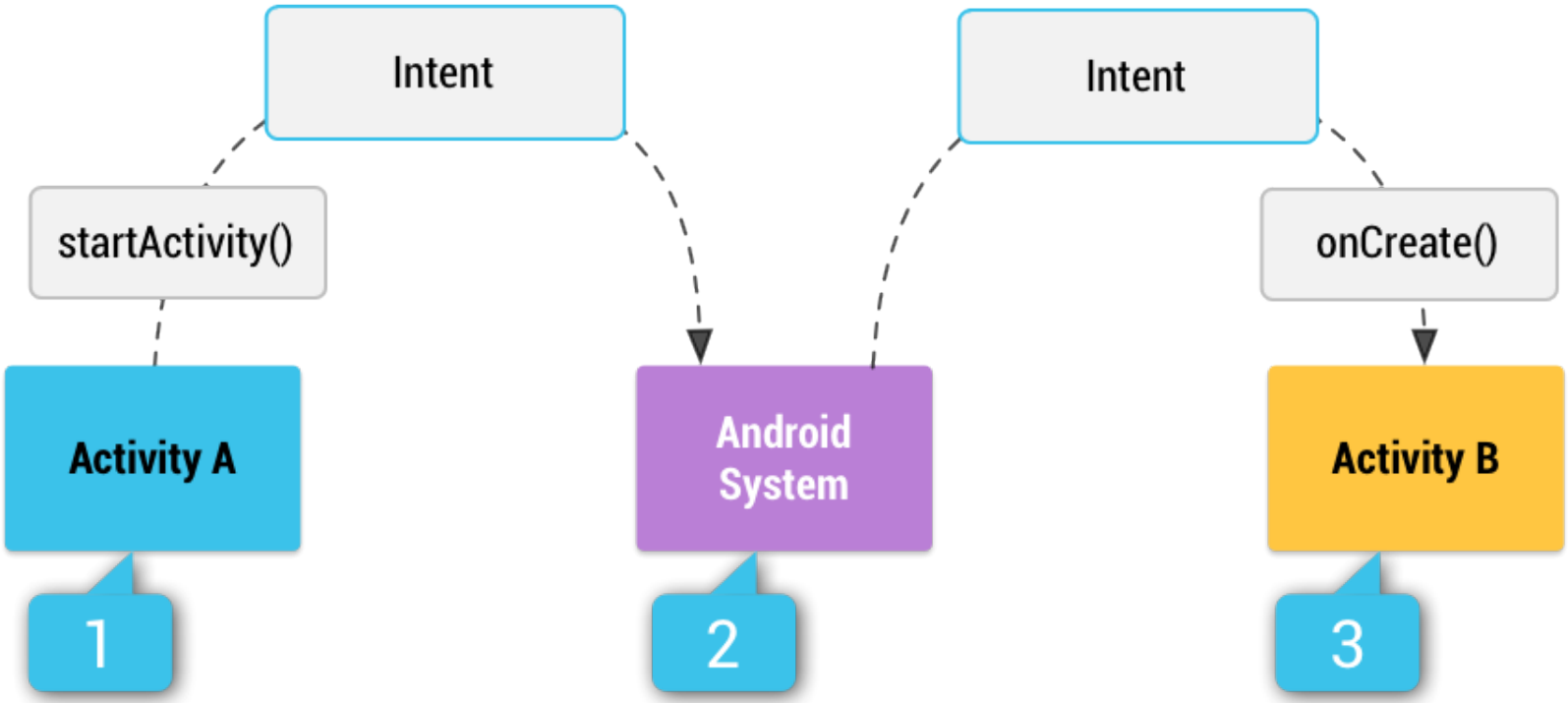
Intents

Un **intent** és un objecte de missatgeria que podeu utilitzar per sol·licitar una acció d'un altre component d'una app.
Dos tipus d'*intents*:

- **Intents explícits:** Especifiquen quina aplicació administra (rep) l'intent. S'utilitzen normalment per iniciar activitats dins d'una mateixa app, pel simple fet que el propi intent es pot determinar amb el nom (conegut) de la classe o activitat que la crida.
- **Intents implícits:** No nomenen cap component o classe específics, així que es pot utilitzar (interpretar) per altres apps del sistema. Exemple: quan una app necessita obrir el mapa, l'intent es trasllada a una altra aplicació que permeti obrir mapes. En aquest cas, al ser implícit, Google Maps pot llegir la informació i executar l'acció.

Intent específic vs Intent genèric

Intent - Cicle de transmissió



Com crear un Intent

Un objecte **Intent** te informació que el sistema Android utilitza per determinar quin component ha d'inicialitzar (com el nom exacte del component o la categoria que ha de rebre l'intent) i informació del component receptor que s'utilitza per realitzar l'acció correctament (per exemple, l'acció que s'ha d'efectuar i les dades sobre les que ha d'actuar). Els camps a crear per definir un intent són els següents:

1. **Nom del component**. Component a iniciar. Opcional. És el que defineix un intent com explícit.
2. **Acció**. String que defineix l'acció genèrica a realitzar (View, Send, etc).
3. **Dades**. Tipus de dades a manipular per l'apliació que genera i/o rep.
4. **Categoria**. String amb informació sobre les dades a manipular.

Informació addicional dels Intents

Existeix informació extra que pot ser associada a un Intent. Per a més informació, podeu consultar la documentació oficial: `https://developer.android.com/guide/components/intents-filters`

Exemples d'Intents

1. Intent explícit

```
// Executed in an Activity, so 'this' is the Context
// The fileUrl is a string URL, such as "http://www.example.com/image.png"
val downloadIntent = Intent(this, DownloadService::class.java).apply {
    data = Uri.parse(fileUrl)
}
startService(downloadIntent)
```

2. Intent implícit

```
// Create the text message with a string
val sendIntent = Intent().apply {
    action = Intent.ACTION_SEND
    putExtra(Intent.EXTRA_TEXT, textMessage)
    type = "text/plain"
}

// Verify that the intent will resolve to an activity
if (sendIntent.resolveActivity(packageManager) != null) {
    startActivity(sendIntent)
}
```

Selector de apps

Quan s'envia un intent implícit, és possible que diverses aplicacions puguin executar-lo. Per poder sel·leccionar una aplicació necessitem un selector com el següent:

```
val sendIntent = Intent(Intent.ACTION_SEND)
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
val title: String = resources.getString(R.string.chooser_title)
// Create intent to show the chooser dialog
val chooser: Intent = Intent.createChooser(sendIntent, title)

// Verify the original intent will resolve to at least one activity
if (sendIntent.resolveActivity(packageManager) != null) {
    startActivity(chooser)
}
```

Filtres

En el procés de rebre un Intent, s'ha de definir quins paràmetres pot rebre i en quina activitat. Per fer-ho es dissenyen filtres amb un element <intent-filter> declarats al *Manifest.xml*. Un component d'aplicació ha de declarar filtres independents per cada tasca única que pot fer. S'han d'especificar aquests elements:

- <**action**>
- <**data**>
- <**category**>

Exemples de filtres

```
<activity android:name="MainActivity">
    <!-- This activity is the main entry, should appear in app launcher -->
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<activity android:name="ShareActivity">
    <!-- This activity handles "SEND" actions with text data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
    <!-- This activity also handles "SEND" and "SEND_MULTIPLE" with media data -->
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <action android:name="android.intent.action.SEND_MULTIPLE"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="application/vnd.google.panorama360+jpg"/>
        <data android:mimeType="image/*"/>
        <data android:mimeType="video/*"/>
    </intent-filter>
</activity>
```

Exercici pràctic

- Afegiu un intent a l'aplicació de la calculadora per mostrar el resultat de l'operació en una nova Activity.