Android Essentials

UI

Ciclo de vida de la aplicación

FOLDER IT

En el mejor de los casos todas las aplicaciones iniciadas van a quedar en memoria.
Lo que hace que el cambio entre aplicaciones sea más rápido.
Pero en la realidad la memoria es limitada
Para administrar la memoria el sistema puede terminar procesos en cualquier momento

Cada proceso tiene una prioridad, si el sistema necesita memoria se terminan los procesos siguiendo las prioridades hasta tener la cantidad de memoria necesaria. Se usa **LRU** como estrategia de selección

Posibles estados

- **Prioridad 1 -** Foreground (primer plano), el usuario está usando la aplicación
- **Prioridad 2 -** Visible, aplicación parcialmente visible pero no en uso.
- **Prioridad 3 -** Service, de una aplicación que no califica como prioridad 1 o 2
- **Prioridad 4 -** Background (segundo plano), app pausada sin servicios o receivers escuchando
- **Prioridad 5 -** Empty, app sin componentes activos

Métodos del Ciclo de vida de la aplicación

FOLDER IT

`onCreate()` - llamado antes de que inicie el primer componente de la app

`onLowMemory()` - llamado cuando el sistema pide que la app libere memoria

`onTrimMemory()` **`(API 14+)`** llamado cuando el sistema ha determinado que es un buen momento para que la app libere memoria que no necesita

`onTerminate()` - solo para testing, no se usa en producción

`onConfigurationChanged()` - llamado cuando la configuración cambia

Activity

FOLDER IT

Una Activity es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción.

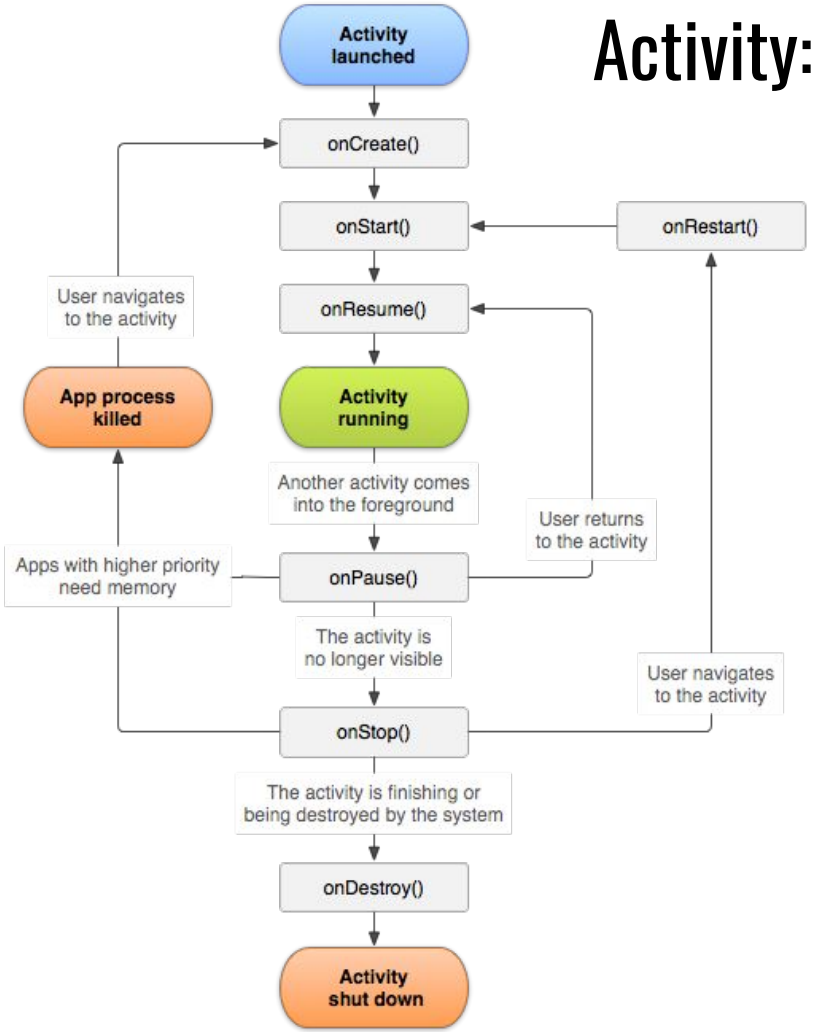
A cada actividad se le asigna una ventana en la que se puede dibujar su interfaz de usuario. La ventana generalmente abarca toda la pantalla, pero en ocasiones puede ser más pequeña que esta y quedar "flotando" encima de otras ventanas.

Una aplicación generalmente consiste en múltiples actividades
Cada actividad puede a su vez iniciar otra actividad para poder realizar diferentes acciones.

Cada vez que se inicia una actividad nueva, se detiene la actividad anterior, pero el sistema conserva la actividad en una pila (la "pila de actividades").

Cuando se inicia una actividad nueva, se la incluye en la pila de actividades y capta el foco del usuario.

Activity: ciclo de vida



Estados:

- Running
- Paused
- Stopped
- Killed

Cuando una actividad entra y sale de los diferentes estados que se describieron más arriba, esto se notifica a través de diferentes callbacks.

Activity: ciclo de vida

FOLDER IT

```java
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) { // la actividad está siendo creada
        super.onCreate(savedInstanceState);
    }
    @Override
    protected void onStart() { // la actividad está por mostrarse
        super.onStart();
    }
    @Override
    protected void onResume() { // la actividad es visible
        super.onResume();
    }
    @Override
    protected void onPause() { // la actividad está por pasar a segundo plano
        super.onPause();
    }
    @Override
    protected void onStop() { // la actividad no es más visible
        super.onStop();
    }
    @Override
    protected void onDestroy() { // la actividad está por ser destruida
        super.onDestroy();
    }
}
```

Interfaces

Tenemos un archivo XML donde se encuentra el diseño puramente visual de la pantalla
Y un archivo Java, donde se encuentra el código fuente que determina la lógica de la pantalla desde el cual, podremos tener acceso a los elementos visuales definidos en el XML

Android implementa este mecanismo de vinculación mediante el uso de un ID para cada elemento visual.

XML para la vista

Java para la lógica

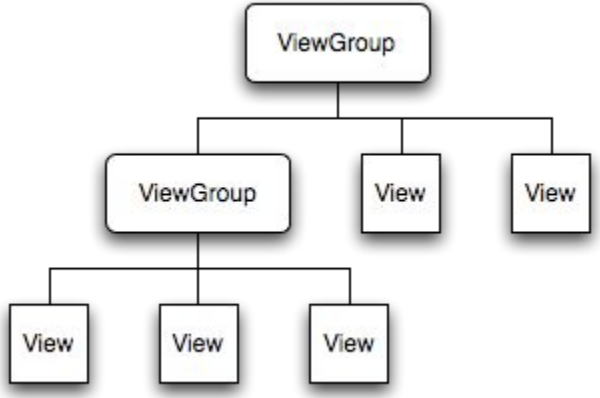
Interfaces

FOLDER IT

En Android las interfaces se construyen utilizando una jerarquía de objetos del tipo `View` y `ViewGroup`.

Las `View` son componentes visuales (`Widgets`) (`TextView`, `Button`, `EditText`).

Las `ViewGroup` son controles más complejos (`Layout`, `ListView`), y no son visuales.



Layouts

FOLDER IT

Definen la estructura visual de una interface

Son elementos no visibles, que sirven para controlar la distribución, posición y dimensiones de los elementos en su interior.

Se puede declarar un layout de dos formas diferentes:

- ✓ Declarar los elementos visuales en un XML, junto con sus propiedades y comportamiento.
- ✓ Instanciar los elementos visuales desde nuestro archivo Java, podemos crear objetos del tipo `View` y `ViewGroup` y manipular sus propiedades y comportamiento de manera dinámica.

Layouts

FOLDER IT

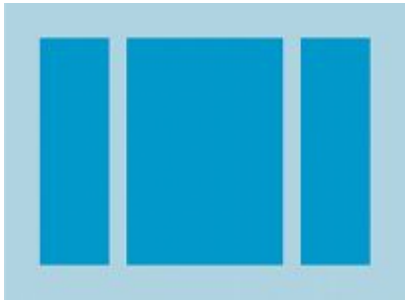
```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="folderit.net.clase1.MainActivity">
   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/titulo"
       android:id="@+id/textView" />
   <Button
       android:text="Button"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/textView"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="45dp"
       android:id="@+id/my_button" />
</RelativeLayout>
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Button mButton = (Button) findViewById(R.id.my_button);
   TextView mTextView = (TextView) findViewById(R.id.my_textView);
}
```

Layouts comunes

FOLDER IT

Linear Layout



Organiza sus elementos secundarios en una sola fila horizontal o vertical.
Si la longitud de la ventana supera la longitud de la pantalla, crea una barra de desplazamiento.

Relative Layout

Te permite especificar la ubicación de los objetos secundarios en función de ellos mismos (el objeto secundario A a la izquierda del objeto secundario B) o en función del elemento primario (alineado con la parte superior del elemento primario).

Web View

Muestra páginas web.

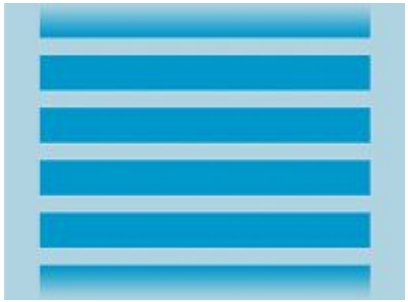
Layouts con adaptadores

FOLDER IT

Cuando el contenido de tu diseño sea dinámico o no sea predeterminado, puedes usar un diseño con la subclase `AdapterView` para completar el diseño con vistas durante el tiempo de ejecución.

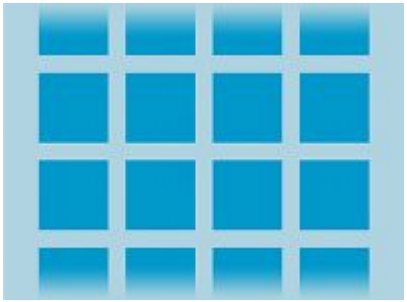
Una subclase de la clase `AdapterView` usa un `Adapter` para enlazar datos con su diseño. El `Adapter` se comporta como intermediario entre la fuente de datos y el diseño `AdapterView`; el `Adapter` recupera los datos (de una fuente como una matriz o una consulta a la base de datos) y convierte cada entrada en una vista que puedes agregar al diseño `AdapterView`.

`ListView`



Muestra una sola lista de columnas desplazable.

`GridView`



Muestra una cuadrícula desplazable de columnas y filas.

Otros Layouts

FOLDER IT

`FrameLayout`

Es el más simple de todos los layouts de Android.
Coloca todos sus controles hijos alineados con su esquina superior izquierda, de forma que cada control quedará oculto por el control siguiente
Esto provoca que sólo se lo utilice para mostrar un único control en su interior, a modo de contenedor (placeholder) sencillo para un sólo elemento sustituible, por ejemplo una imagen.

`TableLayout`

Este layout permite distribuir sus elementos hijos de forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla.
La estructura de la tabla se define indicando las filas que compondrán la tabla (objetos TableRow), y dentro de cada fila las columnas necesarias, no existen objetos del tipo TableColumn, directamente insertamos los controles necesarios dentro del TableRow y cada componente insertado corresponderá a una columna de la tabla.

Sintaxis

FOLDER IT

Cuando en el archivo que describe la interfaz, creamos un Nuevo elemento (widget), y necesitamos que se agregue al archivo R.java, entonces debemos indicarlo con la sintaxis “`@+id/`”.

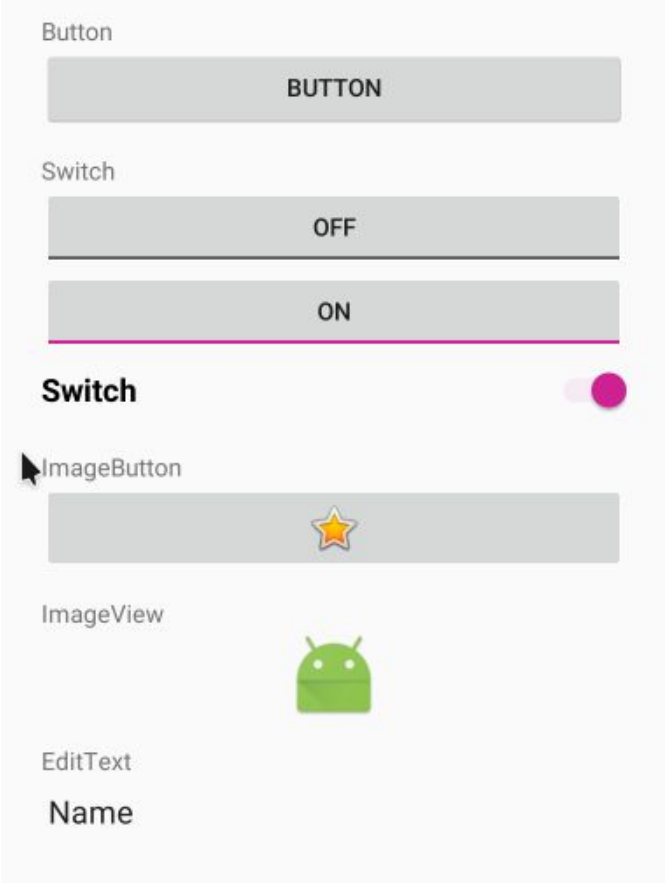
Así la primer ocurrencia de un valor “id” debe incluir el signo “+”. (`@+id/m_button`). Luego las siguientes veces que se lo referencie puede omitirse.

La ausencia del signo “+” indica que se debe usar un recurso ya existente.
Al poner un “+” en la primer ocurrencia, estamos indicando al compilador que genere el recurso, al no ponerlo en las siguientes, estamos indicando que utilice un recurso creado.

Componentes

FOLDER IT

- Button
- ToggleButton
- Switch
- ImageButton
- ImageView
- TextView
- EditText

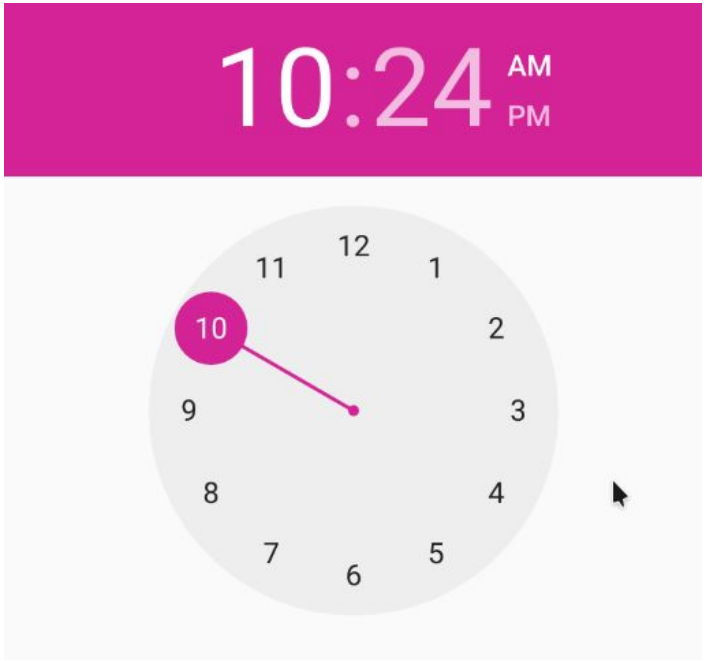


Componentes

FOLDER IT

DatePicker

TimePicker



Componentes

FOLDER IT

CheckBox

RadioButton

RadioGroup

Retomando

FOLDER IT

Con lo visto hasta aquí podemos definir que la UI de una aplicación Android está formada por 3 elementos claves

`View` representan un elemento de la GUI

`Container` es una vista que contiene otras vistas

`Layout` permite acomodar vistas y contenedores dentro de la interface.

Repaso: Patrón Adapter

FOLDER IT

Convierte la interfaz de una clase en otra interfaz que el cliente espera.
Adapter permite a las clases trabajar juntas, lo que de otra manera no podría hacerlo debido a sus interfaces incompatibles.

Cliente - colabora con objetos adaptables a la interfaz Objetivo

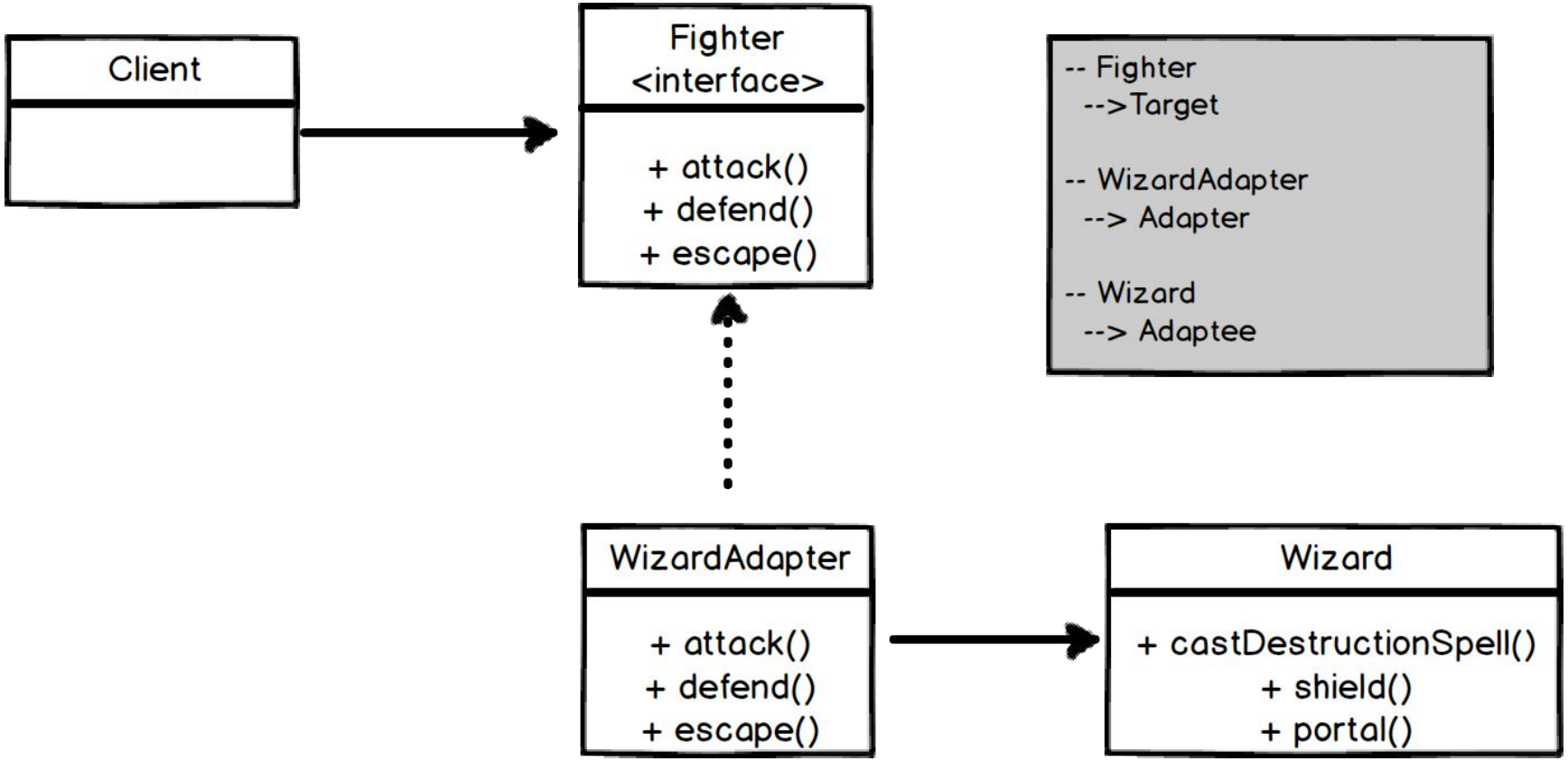
Objetivo o Target - define el dominio específico de la Interfaz que el Cliente usa

Adaptador o Adapter - adapta la Interfaz del “Adaptado” a la interfaz Objetivo

Adaptado o Adaptee - define una interfaz existente que se necesita adaptar

Patrón Adapter

FOLDER IT



Spinner

FOLDER IT

Es una lista desplegable

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```


```
Spinner spinner = (Spinner) findViewById(R.id.spinner); // Cliente

final ArrayList<String> mArray = new ArrayList<>(); // Adaptado

ArrayAdapter<String> adapter = // adaptador
       new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, mArray);

spinner.setAdapter(adapter); // la interfaz objetivo esta definida en la clase padre
```

Listas

FOLDER IT