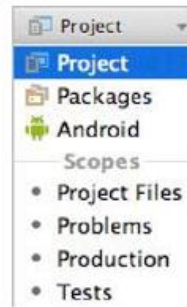


■ **menu**: contiene las barras de acciones de aplicación.

- **Gradle Scripts**: contiene los scripts y las configuraciones Gradle de proyecto.

Estas carpetas se corresponden con la vista **Android** del proyecto, pero puede cambiar el tipo de visualización del proyecto mediante el menú situado sobre su arquitectura de proyecto.



## 03 de Marzo de 2023

+ Cargar un proyecto en Android.doc

+ Creación de interfaces sencillas.

+ Objetivo:

➤ Hacer aplicaciones con interfaces de usuario:

+ Aplicaciones: Botones, menus, CheckBox, Radiobotons, Campos de Texto:

+ => Estos componentes se programan en el Layout (XML)

+ => Colocar los componentes en el layout

+ Y Como se ponen o se colocan ¿?

+ Se usan los layout's o machotes o plantillas

+ => Son formatos que especifican donde colocar los componentes

En un layout, dicen como organizar una pantalla

+ => Los componente tienen atributos: Color, tamaño, tipo de letra, etc. Posición en la ventana. Cada Layout tiene atributos, ancho, alto, espacio y márgenes

+ Programación de eventos y captura de datos de los componentes

+ => Programarlo en el programa principal de java

+ => Conocer el nombre del componente en Java para programarlo






Después vienen varias acciones:

+ Guardar los datos capturados en una BD

+ Transferirlo a otra vista

+ *Layouts, interfaces y eventos de interface.doc*

Ver los siguientes programas:

-  Ch5\_LinearLayout
-  Ch5\_TableLayout
-  Ch5\_RelativeLayout
-  Ch5\_GridLayout
-  Ch5\_ScrollView.doc

**10 de Marzo de 2023**

Programación de interfaces de usuario.

Declarar los componentes de la vista de usuario en el layout (xml) con un identificador: +@id

```
id="@+id/TxtInput
```

Con la posición y tamaño y atributos.

En el programa en Java:

1. Declarar variables del tipo de componente correspondiente de la vista, variables de tipo botón, Texto, Checkbox, radiobottons, etc.

```
private EditText txtTexto;  
private Button btnNegrita;  
private Button btnSetText;  
  
private TextInputLayout txtInputLayout;  
private EditText txtInput;  
private Button btnComprobar;  
  
private CheckBox cbMarcame;  
private TextView lblMensaje;  
private RadioGroup rgOpciones;  
private RadioButton rbOpcion1;  
private RadioButton rbOpcion2;
```

2. Asociar estas variables con los componentes de la vista mediante la clase R y el identificador +@id declarado en el layout (xml), mediante el método findViewById:

```
txtTexto = (EditText) findViewById(R.id.TxtBasico);  
btnNegrita = (Button) findViewById(R.id.BtnNegrita);  
cbMarcame = (CheckBox) findViewById(R.id.ChkMarcame);  
  
lblMensaje = (TextView) findViewById(R.id.LblSeleccion);  
  
rbOpcion1 = (RadioButton) findViewById(R.id.RbOpcion1);  
rbOpcion2 = (RadioButton) findViewById(R.id.RbOpcion2);
```

3. La clase R es una clase auxiliar que me ayuda para comunicarme entre los componentes de la vista y el programa en java.
4. Y con esto las variables de tipo botón, Texto, Checkbox, radiobuttons, etc. a través de sus métodos, pueden conocer los atributos de los componentes de interface: valores que se teclearon, que botón se hizo click, el valor de los radiobuttons, checkbox, etc. y además pueden modificar los componentes en la interface: color, tipo de letra, habilitarlo, tamaño, posición, etc.:

```
Spannable texto = txtTexto.getText();  
txtTexto.setText(str);
```

La programación de eventos:

Programar la clase listener View.OnClickListener del componente correspondiente y sobreponer el método de acción `onClick`, el cual se va a ejecutar al momento de hacer un click, o un cambio en el componente visual (Checkbox, radiobuttons, por ejemplo). Y asociar este listener al componente declarado en la clase en java (`setOnClickListener`):

Para el caso del botón:

```
btnNegrita.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View arg0)  
    {  
        ....  
    }  
});
```

Para el caso del CheckBox:

```
cbMarcame.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        boolean isChecked = ((CheckBox)view).isChecked();  
  
        if (isChecked) {  
            cbMarcame.setText("Checkbox marcado!");  
        }  
        else {  
            cbMarcame.setText("Checkbox desmarcado!");  
        }  
    }  
});
```

Para el caso del RadioButton:

```
View.OnClickListener list = new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        String opcion = "";  
        switch(view.getId()) {  
            case R.id.RbOpcion1:  
                opcion = "opción 1";  
                break;  
            // ...  
        }  
    }  
};
```

```

        break;
    case R.id.RbOpcion2:
        opcion = "opción 2";
        break;
    }

    lblMensaje.setText("ID opción seleccionada: " + opcion);
}
};




```

Y el evento `OnClickListener` se asocia al componente gráfico:

```

rbOpcion1.setOnClickListener(list);
rbOpcion2.setOnClickListener(list);

```

-  android-imagen-texto
-  android-check-radio
-  android-botones

Ver las notas:

 *Programación de eventos de componentes de interface de usuario.doc*

## 13 de Marzo de 2023

Examen miércoles 22 de Marzo a las 6:00 PM por mail. Duración 2 horas. Abarca desde el inicio hasta la programación de interfaces. Y ese día no habría clases. Es individual.

## 17 de Marzo de 2023

Creación de interfaces sencillas en Android

**Creación de interfaces sencillas en Android: Listas**

Ver Notas:

*Creación de interfaces sencillas en Android – Listas.PDF*

En Android, una lista como componente visual (interface de usuario predefinido), representa un conjunto de elementos que se muestran unos a continuación de los otros.

Algunas aplicaciones de las listas son por ejemplo en agendas, lista de contactos, lista de precios, lista de llamadas, lista de música, lista de mensajes, lista de fotos e imágenes, etc.