

# Módulo 7

Código: 488

## Desarrollo de Interfaces

Técnico Superior en Desarrollo de  
Aplicaciones Multiplataforma



# UNIDAD 1

## Confección de interfaces de usuario

Guía didáctica profesor



1. Objetivos generales de la unidad
2. Competencias y contribución en la unidad
3. Contenidos conceptuales y procedimentales
4. Evaluación
5. Distribución de los contenidos
6. Sesiones

# 1. Objetivos generales de la unidad

## Objetivos curriculares

---

1. Creación de una interfaz gráfica utilizando los asistentes de un editor visual.


2. Utilización de las funciones del editor para localizar los componentes de la interfaz.

3. Modificación de las propiedades de los componentes para adecuar a las necesidades de la aplicación.

4. Análisis y modificación del código generado por el editor visual.

5. Asociación de los eventos a las acciones correspondientes.

## 2. Competencias y contribución en la unidad



Generar interfaces gráficas de usuario mediante editores visuales utilizando las funcionalidades del editor y adaptando el código generado.

	Emplear herramientas y lenguajes específicos, siguiendo las especificaciones, para desarrollar componentes multimedia.	
--	--	--

# 3. Contenidos

CONTENIDOS CONCEPTUALES	CONTENIDOS PROCEDIMENTALES
<ul style="list-style-type: none"><li>• Lenguajes de Programación.</li><li>• Herramientas propietarias y libres de edición de interfaces.</li><li>• Librerías de componentes disponibles para diferentes sistemas.</li><li>• Clases, propiedades, métodos.</li><li>• Componentes, características y campo de aplicación.</li><li>• Enlace de componentes a orígenes de datos.</li><li>• Eventos; escuchadores.</li><li>• Asociación de acciones a eventos.</li></ul>	<ul style="list-style-type: none"><li>• Edición y análisis del código generado por la herramienta de diseño.</li></ul>

## 4. Evaluación

Se ha creado un interfaz gráfico utilizando los asistentes de un editor visual.

Se han utilizado las funciones del editor para ubicar los componentes del interfaz.

Se han modificado las propiedades de los componentes para adecuarlas a las necesidades de la aplicación.

Se ha analizado el código generado por el editor visual.

## 4. Evaluación

Se ha modificado el código  
generado por el editor visual.

Se han asociado a los eventos  
las acciones correspondientes.

Se ha desarrollado una  
aplicación que incluye la interfaz  
gráfica obtenida.



## 1.1. LENGUAJE DE PROGRAMACIÓN

Definiremos lenguaje de programación como un lenguaje formal que a través de un serie de instrucciones, compuestas por un conjunto de símbolos, reglas semánticas y sintácticas, permite escribir secuencias de órdenes y algoritmos para controlar el comportamiento físico y lógico de una máquina.



## 1.1. TIPOS DE LENGUAJES DE PROGRAMACIÓN

### LENGUAJES DE BAJO NIVEL

#### CÓDIGO MÁQUINA

```
:03000000020100FA1001000075813F
7590FFB29012010D80F97A1479D40
90110003278589EAF3698E8EB25B
A585FEA2569AD96E6D8FED9FAD
AF6DD00000001FF255AFED589EA
F3698E8EB25BA585FEA2569AD96
DAC59700D00000278E6D8FED9FA
DAF6DD0000001FF255AFED8FED
9FADAF6DD000F7590FFB29013278
E6D8FED9FADAF6DD00000001FF2
55AFED589EAF3698E8EB25BA585
FEA2569AD96DAC59D9FADAF6D
D00000001FF255AFED8FED9FADA
F6DD000F7590FFB29013278E6D82
78E6D8FED9FA589EAF3698E8EB2
5BA585FEA2569AD96DAF6DD000
00001FF2DAF6DD00000001FF255A
ADAF6DD00000001FF255AFED8FE
D9FA
```

#### ENSAMBLADOR

```
;Fichero printx.asm

[BITS 16]
[GLOBAL Printx]
[SEGMENT CODIGO]

    mov esi,0b800h        ; Segmento donde comienza
    mov es,esi            ; la pantalla en modo texto
    mov esi,80            ; Anchura pantalla
    imul esi,ebx
    add esi,eax

@otro: mov al,[ds:edx]
    cmp al,0
    jz @fin
    mov [es:esi],al        ; Copia un carácter
    inc si
    mov [es:esi],cl        ; Copia el atributo de color
    inc esi
    inc edx
    jmp @otro

@fin: retf                ; Retorno far
```

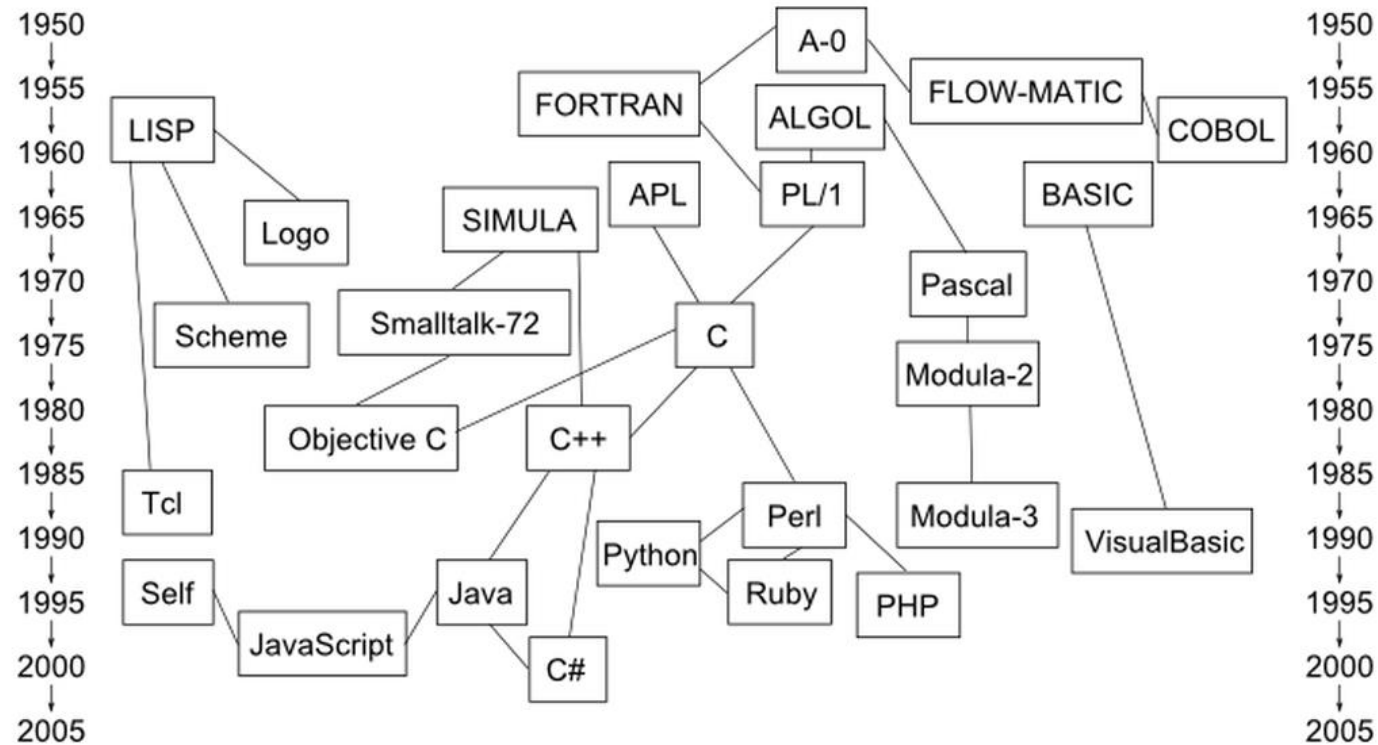
## 1.1. TIPOS DE LENGUAJES DE PROGRAMACIÓN

### LENGUAJES DE ALTO NIVEL

```
}  
  
/**  
 * @param args the command line arguments  
 */  
public void Sorteo() {  
    HashSet<Integer> numerosSorteo = new HashSet<Integer>();  
    int numero;  
    for (int i = 0; i < 6; i++) {  
        numero = (int) (Math.random() * bombo.size())+1;  
        System.out.println("Números: " + numero);  
        numerosSorteo.add(numero);  
        bombo.remove(numero);  
        //numero = (int) (Math.random()*49)+1;  
    }  
}
```

## 1.1. TIPOS DE LENGUAJES DE PROGRAMACIÓN

### GENERACIONES

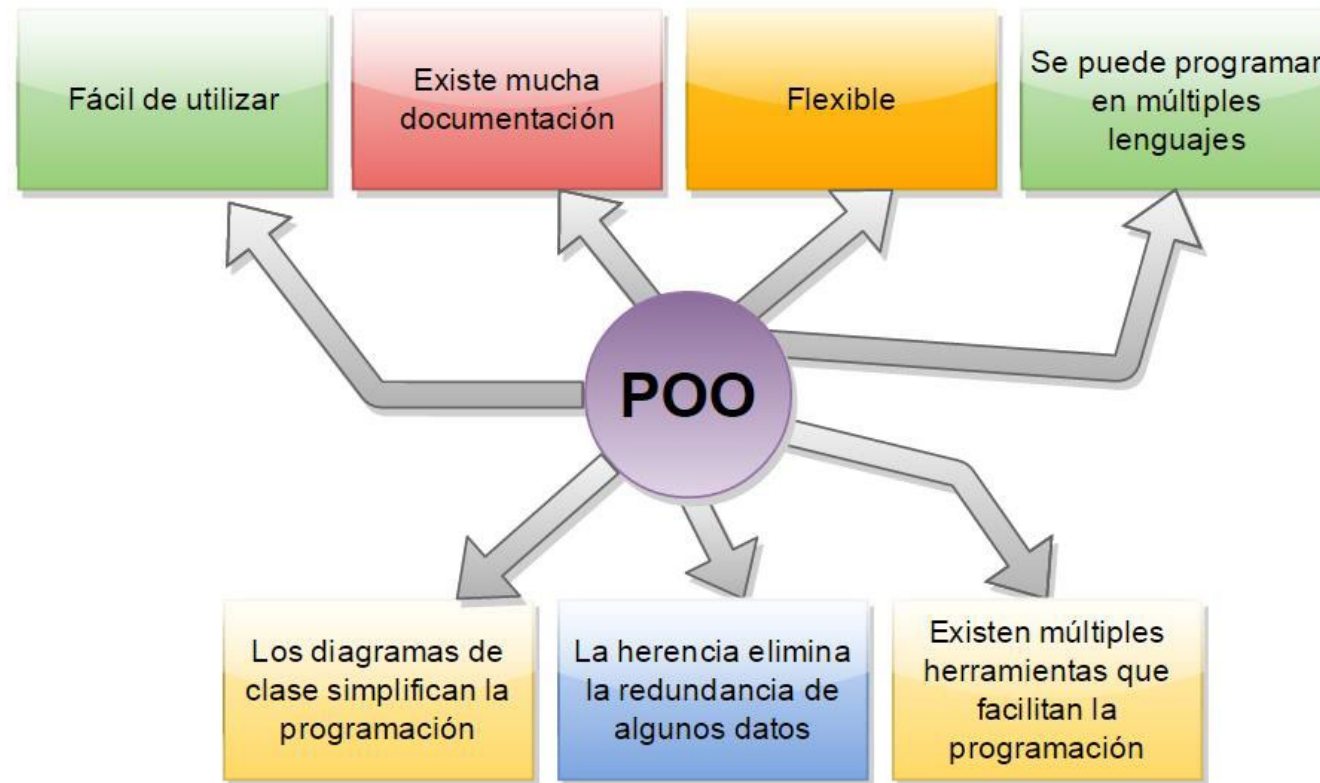


## 1.2. PARADIGMAS DE PROGRAMACIÓN

Definiremos paradigma de programación como un modelo básico de diseño y desarrollo de programas. Los cuales pueden ser considerados patrones de pensamiento para resolver problemas. Entre los tipos de paradigmas de programación más comunes encontramos:

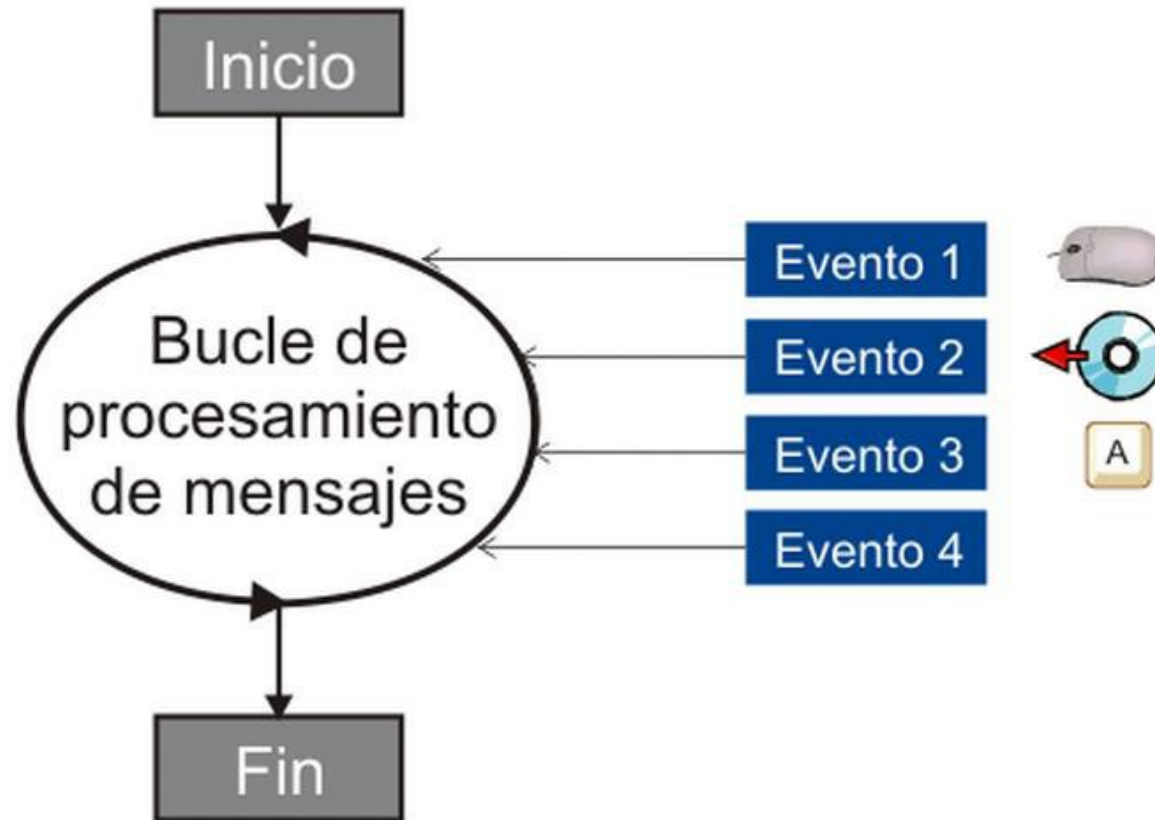
- Imperativo o por procedimientos: es considerado el más común y está representado, por ejemplo, por C, BASIC o Pascal.
- Funcional: está representado por Scheme o Haskell. Este es un caso del paradigma declarativo.
- Lógico: está representado por Prolog. Este es otro caso del paradigma declarativo.
- Declarativo: por ejemplo la programación funcional, la programación lógica, o la combinación lógico-funcional.
- Orientado a objetos: está representado por Smalltalk, un lenguaje completamente orientado a objetos.
- Programación dinámica: está definida como el proceso de romper problemas en partes pequeñas para analizarlos.

## 1.2.1. PROGRAMACIÓN ORIENTADA A OBJETOS





## 1.2.2. PROGRAMACIÓN DIRIGIDA POR EVENTOS



## 1.2.3. PROGRAMACIÓN BASADA EN COMPONENTES

La programación orientada a componentes se basa en la descomposición de sistemas ya conformados en componentes funcionales o lógicos con interfaces bien definidas usadas para la comunicación entre componentes.

Un componente de software es un elemento de un sistema que ofrece un servicio predefinido, y es capaz de comunicarse con otros componentes.



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

VISUAL STUDIO, soporta varios lenguajes de programación:

- VISUAL C++
- VISUAL C#
- VISUAL J#
- VISUAL BASIC .NET
- ASP.NET



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

GAMBAS, Genera código BASIC, similar a Visual Basic.



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

GLADE. Genera código XML.



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

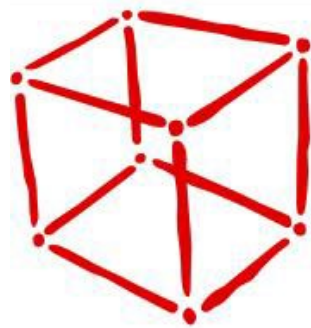
EMBARCADERO DELPHI, genera código Object Pascal.



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

NETBEANS, entre otras da soporte a las siguientes tecnologías:

- JAVA
- PHP
- GROOVY
- C
- C++
- HTML5



# NetBeans

## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

ECLIPSE: Entre otros de soporte a:

- JAVA
- C
- C++
- PYTHON

Y lenguajes de script no tipados como:

- PHP
- JAVASCRIPT



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

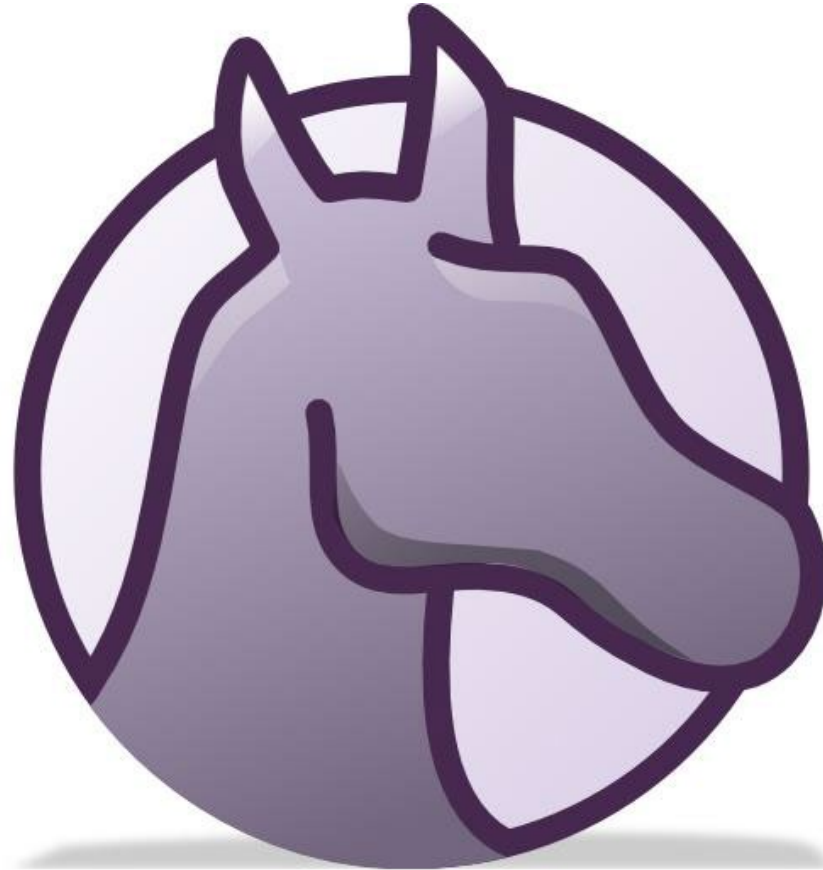
ORACLE DATABASE, permite crear bases de datos y con la herramienta Oracle Designer permite crear interfaces para acceder a ellas.



## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

ANJUTA. Lenguajes soportados:

- C
- C++
- JAVA
- PYTHON
- VALA





## 1.3. HERRAMIENTAS DE EDICIÓN DE INTERFACES

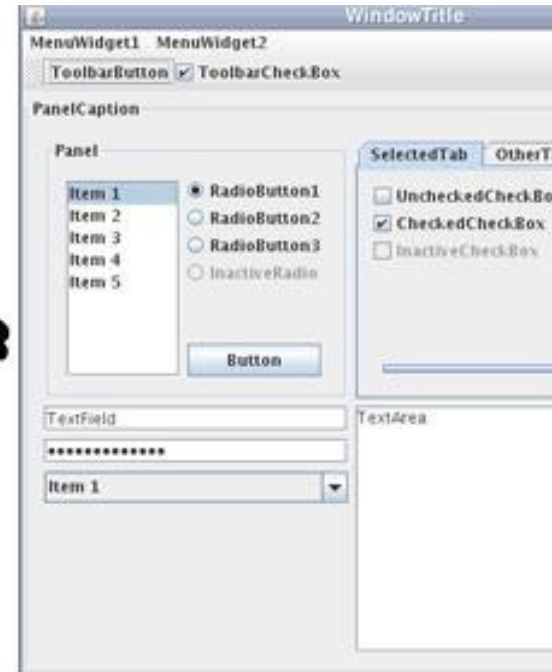
DREAMWEAVER. Lenguajes soportados:

- HTML
- XHTML
- CSS
- JAVASCRIPT
- COLDFUSION MARKUP LANGUAGE CFML
- VBSCRIPT (ASP)
- C#
- VISUAL BASIC (ASP.NET)
- JSP
- PHP



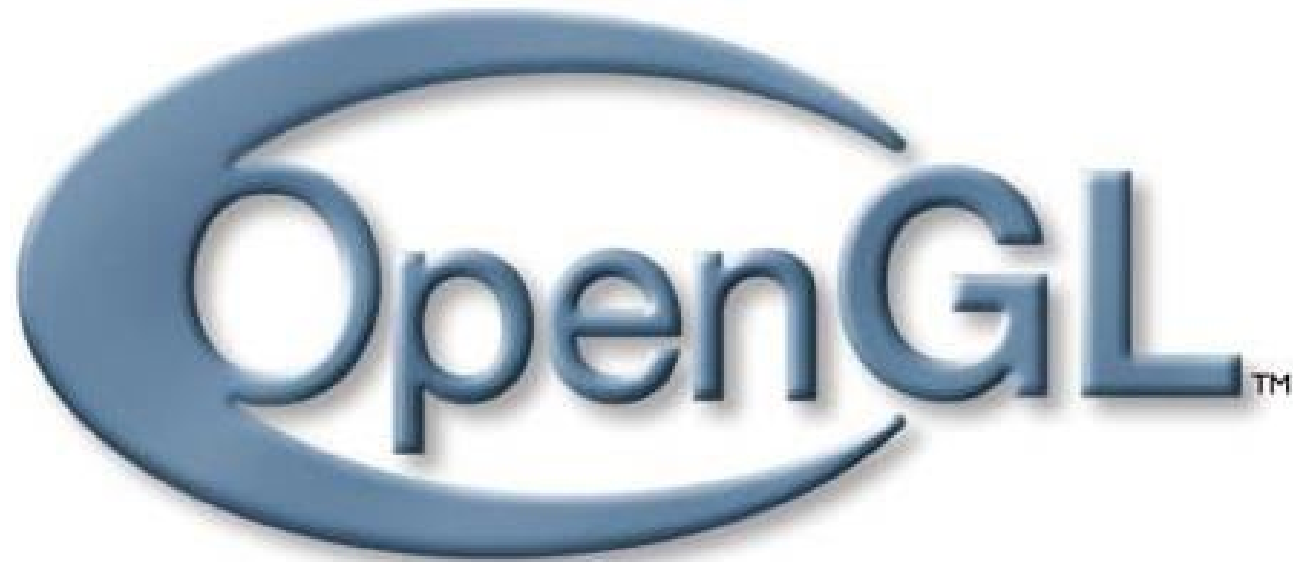
## 1.4. BIBLIOTECAS DE COMPONENTES

SWING, versión más moderna del API de Java, proporcionan una serie de clases e interfaces que la amplían.



## 1.4. BIBLIOTECAS DE COMPONENTES

OpenGL, especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.



## 1.4. BIBLIOTECAS DE COMPONENTES

DIRECTX, colección de API para tareas relacionadas con multimedia, especialmente programación de video juegos en la plataforma Microsoft.

Microsoft®  
DirectX®

## 1.4. BIBLIOTECAS DE COMPONENTES

GTK (GIMP TOOL KIT), biblioteca del equipo GTK+ que contiene objetos y funciones para crear la interfaz gráfica del usuario.

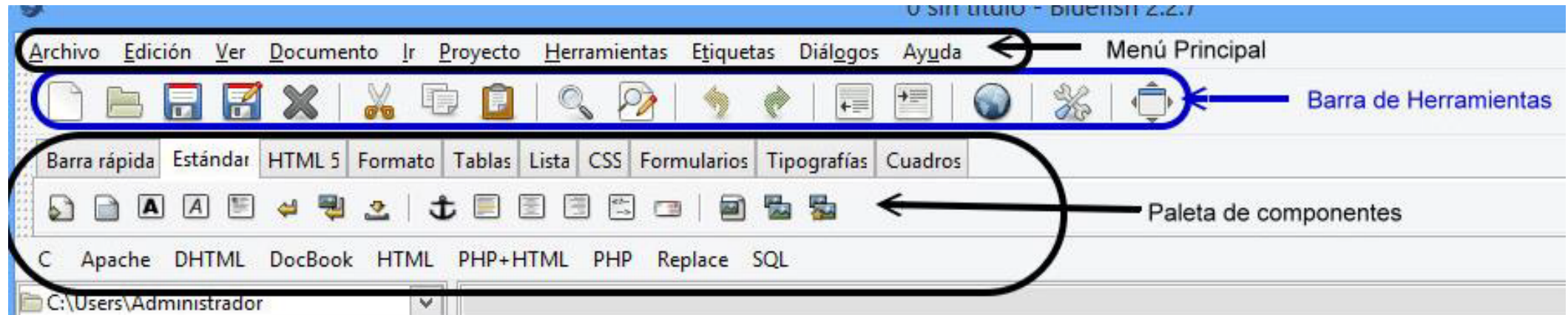


## 1.4. BIBLIOTECAS DE COMPONENTES

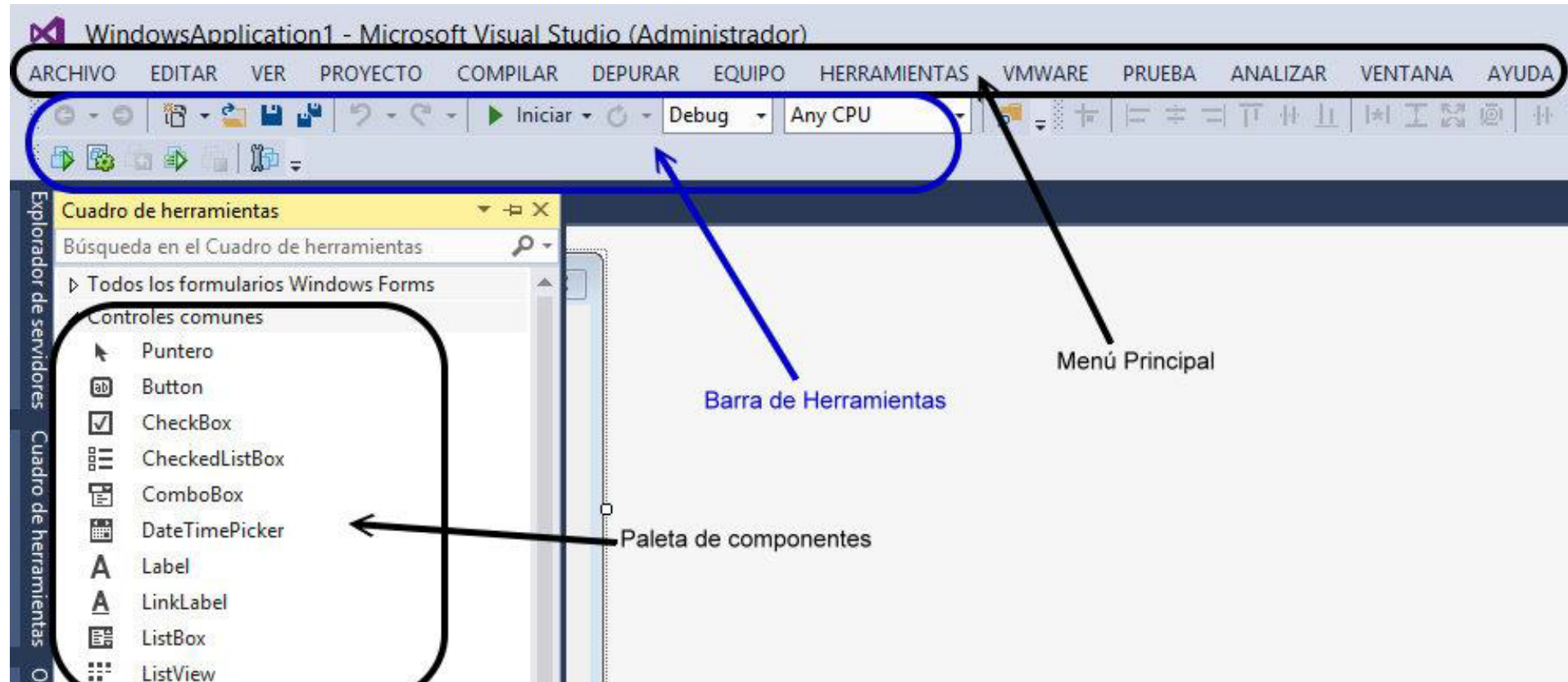
QT, biblioteca multiplataforma para el desarrollo de aplicaciones con interfaz gráfica y sin interfaz como herramientas de línea de comandos y consolas para servidores.



## 1.5. AREA DE DISEÑO, PALETA, EDITOR

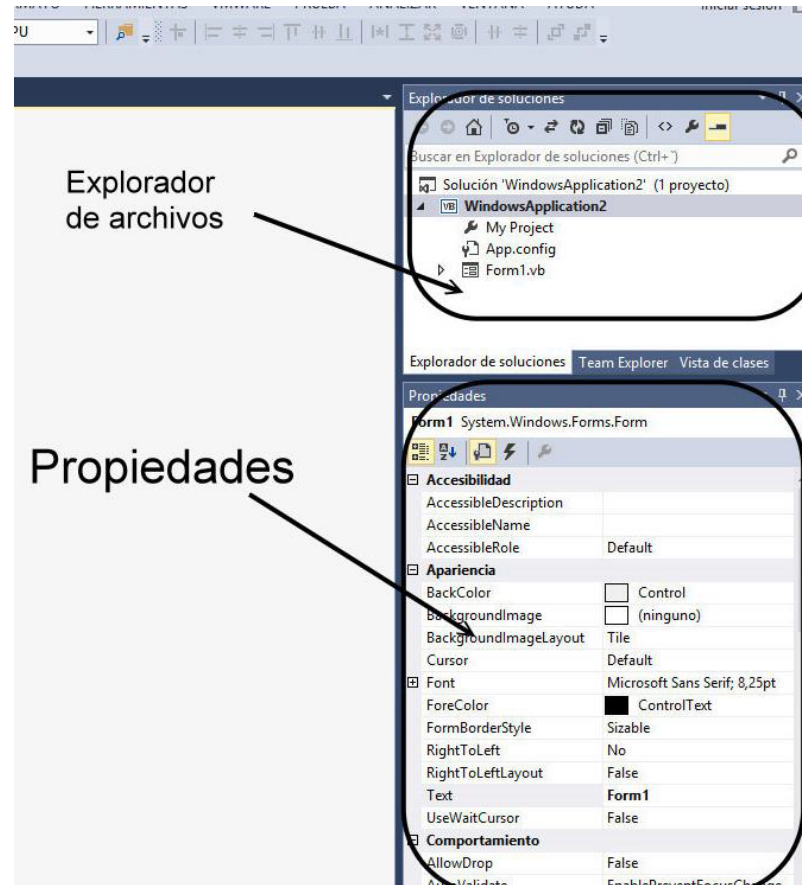


## 1.5. AREA DE DISEÑO, PALETA, EDITOR

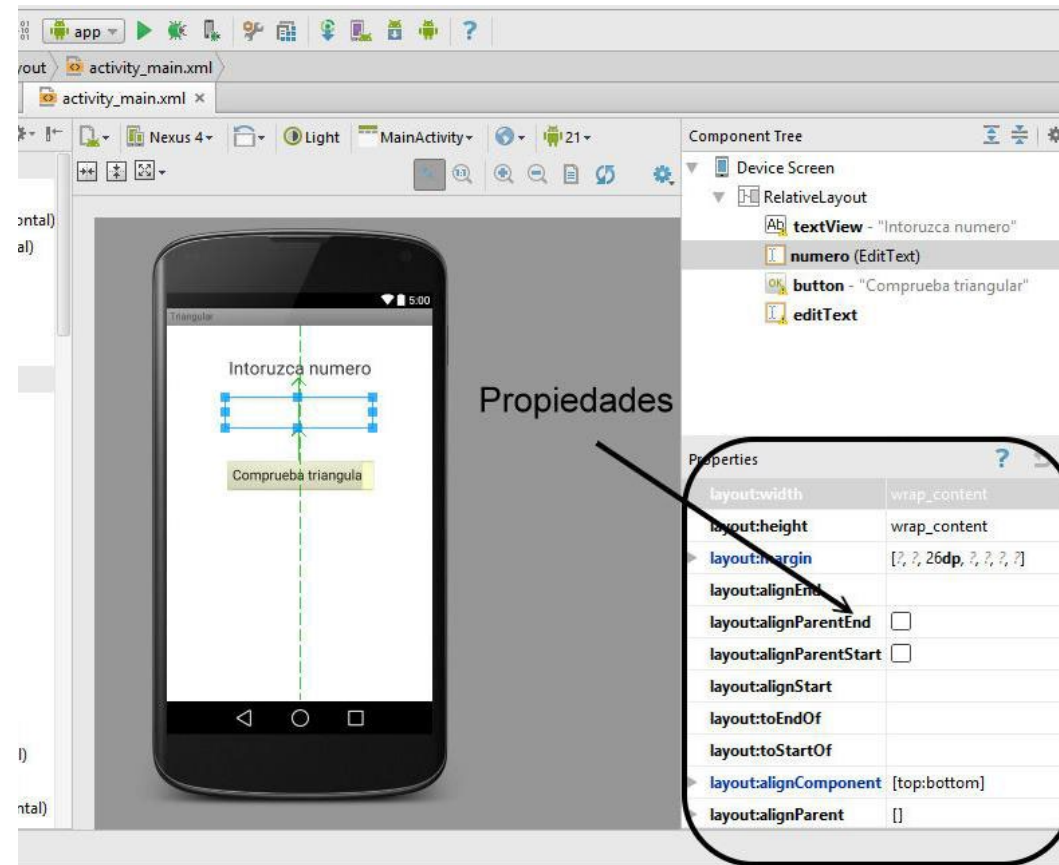




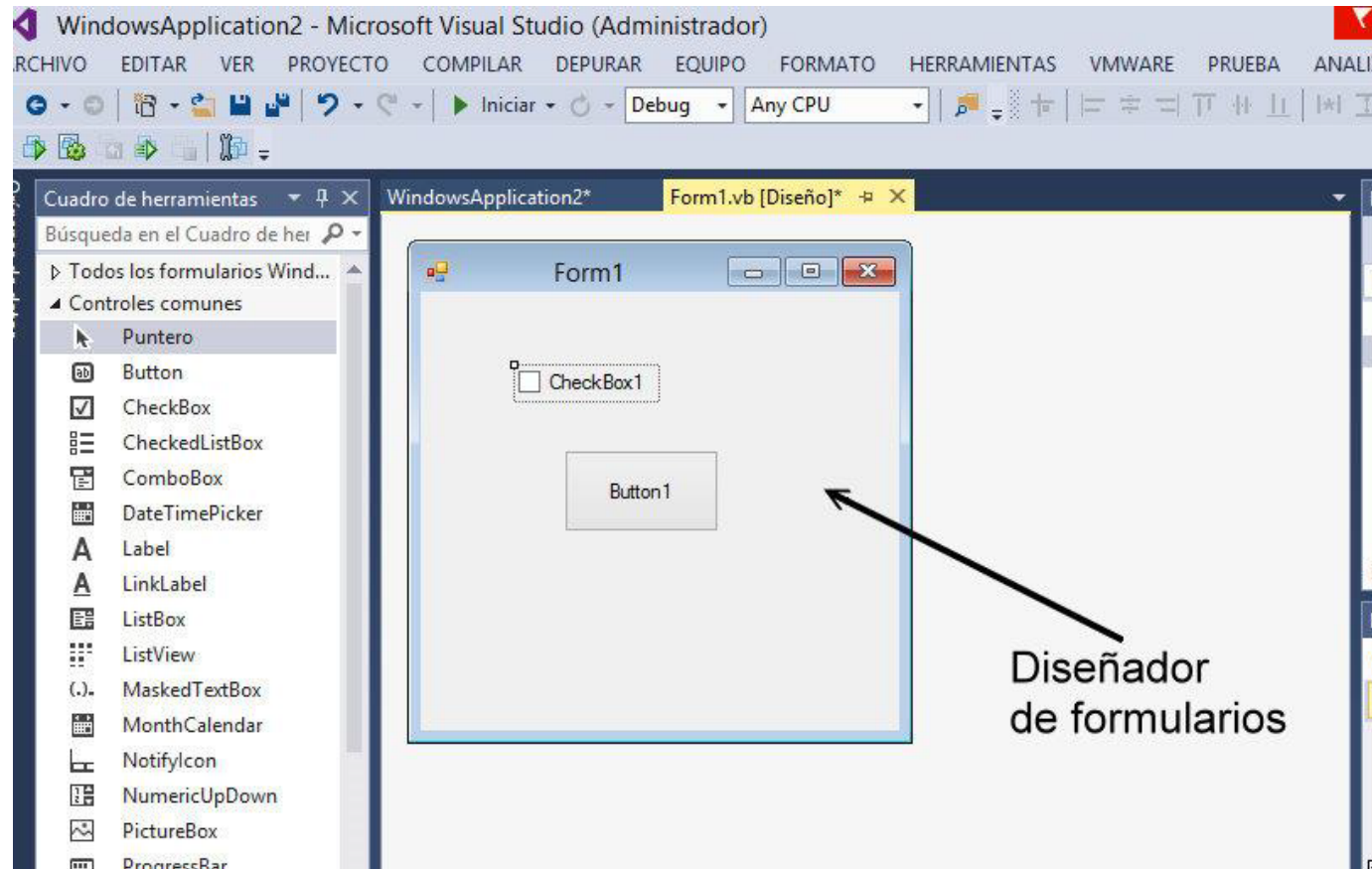
## 1.5. AREA DE DISEÑO, PALETA, EDITOR



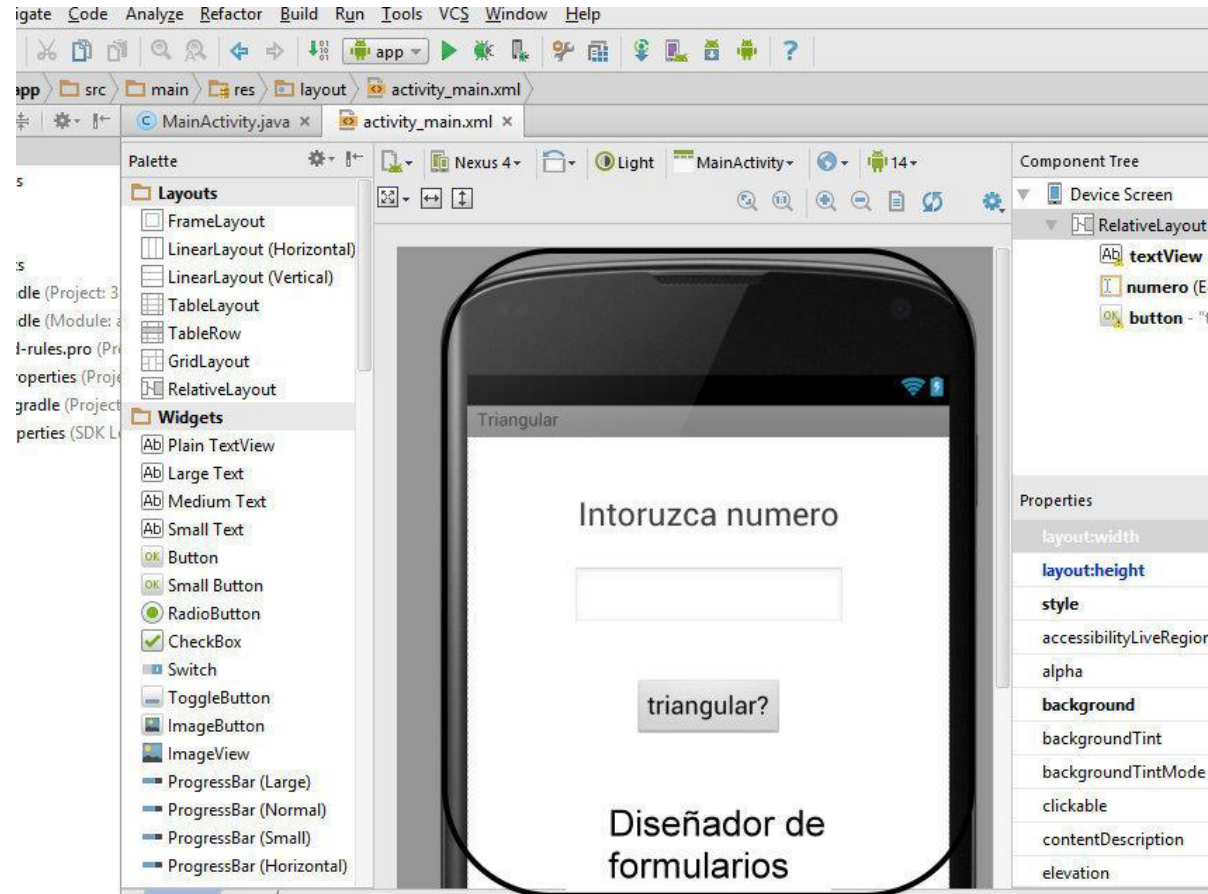
## 1.5. AREA DE DISEÑO, PALETA, EDITOR



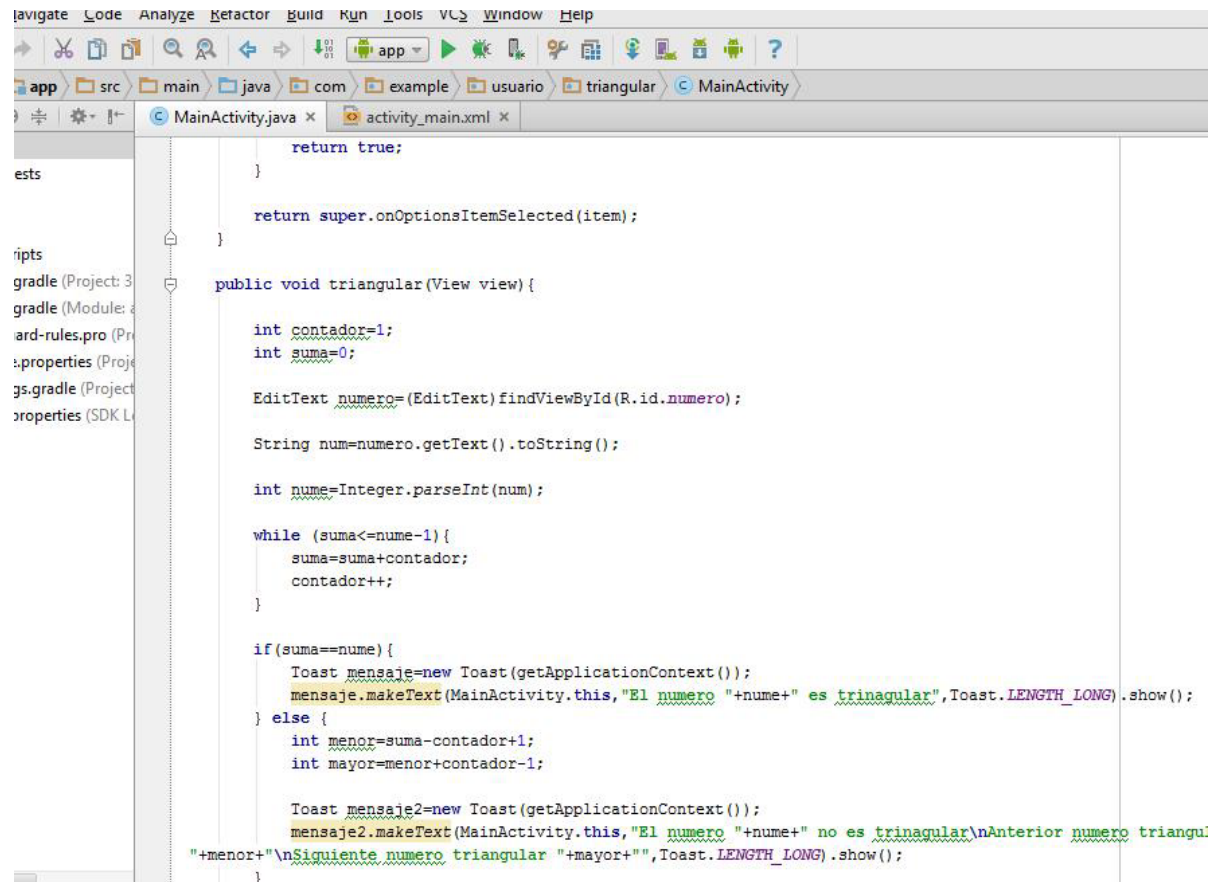
## 1.5. AREA DE DISEÑO, PALETA, EDITOR



## 1.5. AREA DE DISEÑO, PALETA, EDITOR



## 1.5. AREA DE DISEÑO, PALETA, EDITOR



```
return true;
}

return super.onOptionsItemSelected(item);
}

public void triangular(View view) {

    int contador=1;
    int suma=0;

    EditText numero=(EditText) findViewById(R.id.numero);

    String num=numero.getText().toString();

    int nume=Integer.parseInt(num);

    while (suma<=nume-1){
        suma=suma+contador;
        contador++;
    }

    if(suma==nume){
        Toast mensaje=new Toast(getApplicationContext());
        mensaje.makeText(MainActivity.this,"El numero "+nume+" es triangular",Toast.LENGTH_LONG).show();
    } else {
        int menor=suma-contador+1;
        int mayor=menor+contador-1;

        Toast mensaje2=new Toast(getApplicationContext());
        mensaje2.makeText(MainActivity.this,"El numero "+nume+" no es triangular\nAnterior numero triangul
"+menor+"\nSiguiente numero triangular "+mayor+"", Toast.LENGTH_LONG).show();
    }
}
```

## 1.6. CLASES, PROPIEDADES, MÉTODOS

Una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Cada clase es un modelo que define un conjunto de variables y métodos apropiados para operar con dichos datos.

Cada objeto creado a partir de la clase se denomina instancia de la clase.

Las clases se componen de elementos, llamados genéricamente miembros, de varios tipos:

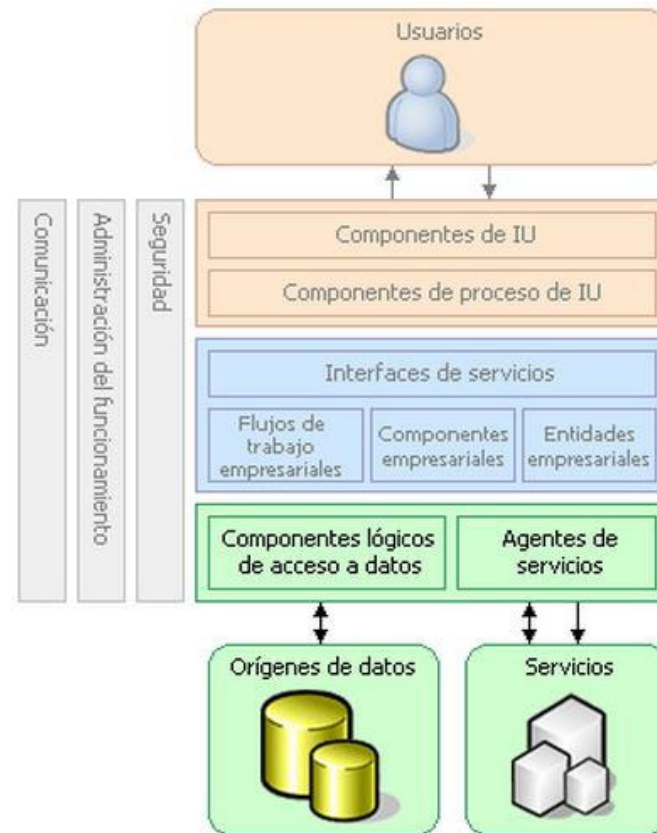
- **Campos de datos:** Los campos de datos se utilizan para contener datos que reflejan el estado de la clase. Los datos pueden estar almacenados en variables, o estructuras más complejas.
- **Métodos:** Los métodos implementan la funcionalidad asociada al objeto. Los métodos son el equivalente a las funciones en programación estructurada.
- Ciertos lenguajes permiten un tercer tipo de miembro: **las propiedades**. Las propiedades son un tipo especial de métodos.

## 1.7. COMPONENTES

Se considera que el nivel de abstracción de los componentes es más alto que el de los objetos y por lo tanto no comparten un estado y se comunican intercambiando mensajes que contienen datos.

## 1.8. ENLACE DE COMPONENTES A ORÍGENES DE DATOS

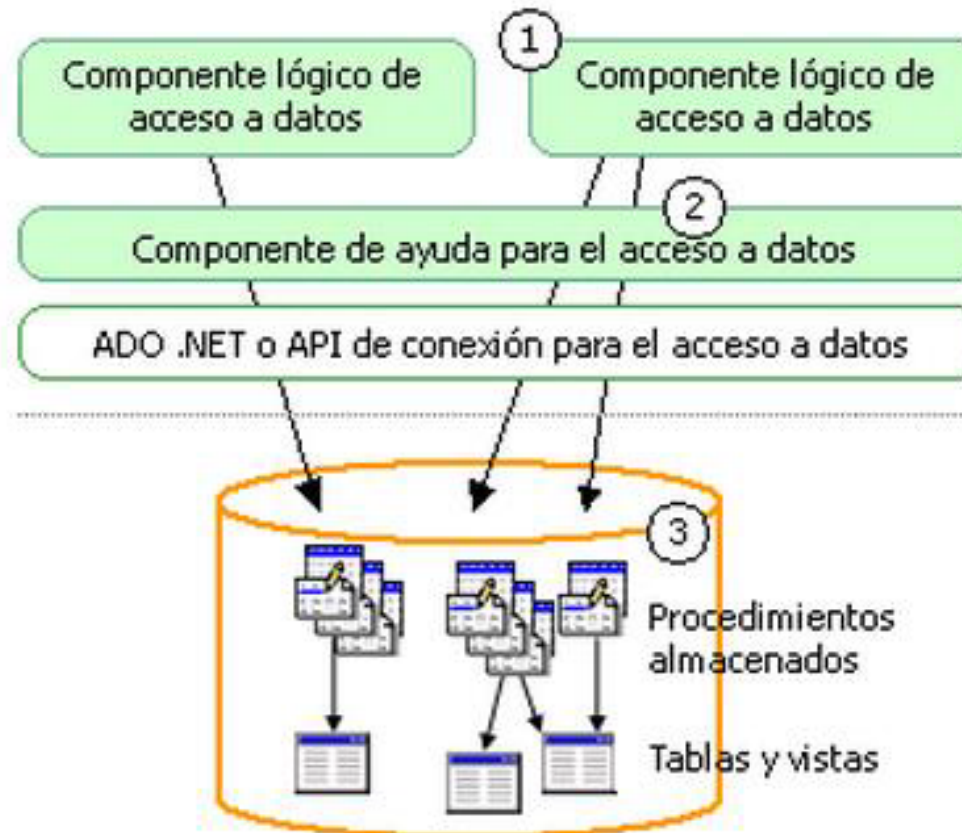
Capa de acceso a los datos y los componentes de lógica a datos.





## 1.8. ENLACE DE COMPONENTES A ORÍGENES DE DATOS

Componentes lógicos de acceso de datos.



## 1.9. INTERFACES RELACIONADAS CON EL ENLACE DE DATOS

Con ADO.NET podemos crear muchas estructuras de datos distintas para satisfacer las necesidades de enlace de la aplicación y de los datos con los que trabaja. ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para programadores de .NET Framework, ofreciendo una gran cantidad de componentes para la creación de aplicaciones de uso compartido de datos distribuidos.

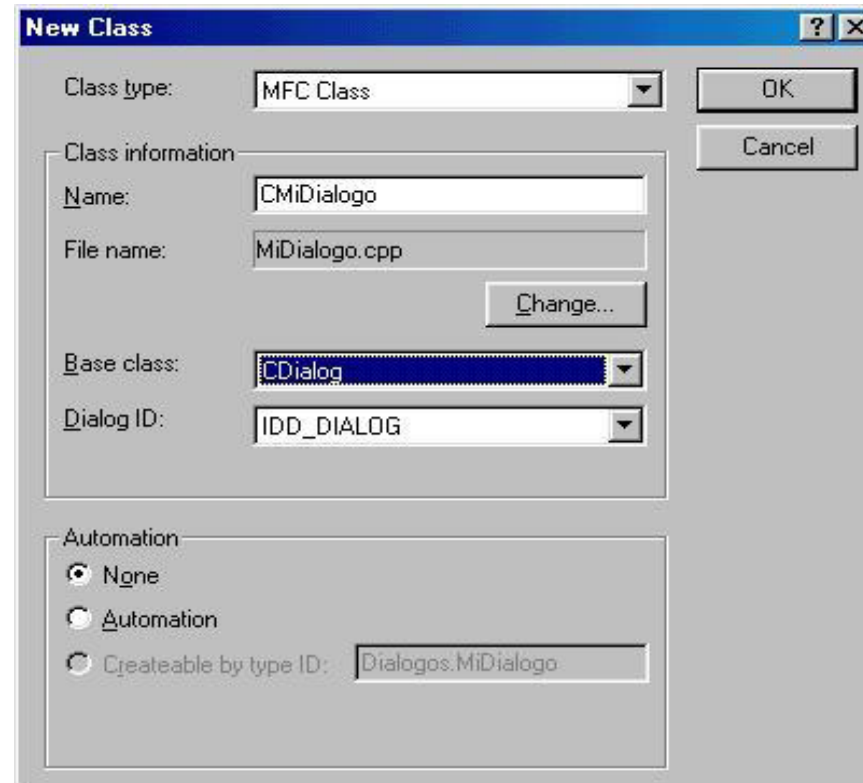
## 1.10. CUADROS DE DIÁLOGO: DIÁLOGOS MODALES Y NO MODALES

Los cuadros de diálogo modales suelen mostrar información crítica ante eventos peligrosos y acciones irreversibles. Requieren que el usuario responda antes de poder continuar con el programa.



## 1.10. CUADROS DE DIÁLOGO: DIÁLOGOS MODALES Y NO MODALES

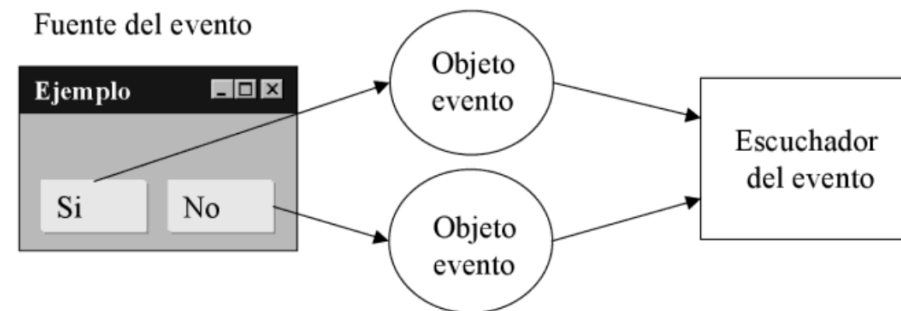
Los cuadros de diálogo no modales permanecen en la pantalla y están disponibles para uso en cualquier momento, permitiendo otras actividades del usuario.



## 1.11. EVENTOS. ESCUCHADORES

La programación dirigida por eventos es un paradigma de programación en el que tanto la estructura como la ejecución van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o provocados por ellos mismos. Por ello es la base de lo que llamamos interfaz de usuario, aunque puede usarse para desarrollar interfaces entre componentes de software o módulos del núcleo.

Un escuchador de eventos (event listener) es un objeto que se registra con un tipo de evento para un objeto. En ellos el programador incorpora el código para responder a los eventos del usuario. Cuando ocurre un evento en un objeto, se ejecuta el método apropiado del escuchador asociado a ese evento y objeto.



## 1.12. EDICIÓN Y ANÁLISIS DEL CÓDIGO GENERADO POR LA HERRAMIENTA DE DISEÑO

Al desarrollar una interfaz a través de una herramienta gráfica de diseño, esta generando código del lenguaje de programación sobre el que se basa dicha herramienta y, por tanto, podemos analizar y ver el código generado por dicha aplicación.

Los IDE más destacados son:

- MICROSOFT VISUAL ESTUDIO
- GAMBAS
- GLADE
- EMBARCADERO DELPHI
- NETBEANS
- ECLIPSE
- ANJUTA
- DREAMWEAVER