

27/02/2025

COMPONENTES Y LIBRERIAS



¿QUÉ ES UNA LIBRERIA?

Conjuntos de clases, métodos, interfaces y funciones empaquetadas en archivos .dll (Dynamic Link Library). Proveen funcionalidades específicas sin ser ejecutables directamente.



¿CÓMO FUNCIONAN LAS LIBRERÍAS ?

- Creación: Se escriben como proyectos de librerías de clases, donde se definen métodos, clases e interfaces reutilizables.

- Compilación: Se compilan en archivos .dll, listos para ser usados en otros proyectos.

- Referencia: En el proyecto donde se necesite, se agrega la librería como una referencia.

- Uso: Se invocan las clases y métodos de la librería como si fueran parte del proyecto, ahorrando tiempo y esfuerzo.

¿PARA QUÉ SIRVEN LAS LIBRERÍAS?

- Reutilización de código: Evitan escribir las mismas funciones una y otra vez.
- Modularidad: Facilitan la organización del proyecto en partes independientes.
- Mantenimiento eficiente: Los cambios en la librería se reflejan en todos los proyectos que la usan.
- Facilitan el trabajo en equipo: Diferentes programadores pueden trabajar en distintas librerías.
- Acceso a soluciones avanzadas: Librerías de terceros ofrecen herramientas especializadas como manipulación de JSON, bases de datos, o interfaces gráficas



RAMAS RELACIONADAS

- Gestión de Paquetes: Uso de plataformas como NuGet para importar librerías externas.
- Patrones de Diseño: Como el patrón de repositorio, que organiza el acceso a datos a través de librerías.
- Desarrollo Orientado a Servicios: Donde librerías encapsulan servicios reutilizables.

COMO FUNCIONA UNA LIBRERIA

- Creación: Se escriben como proyectos de librerías de clases, donde se definen métodos, clases e interfaces reutilizables.

- Compilación: Se compilan en archivos .dll, listos para ser usados en otros proyectos.

- Referencia: En el proyecto donde se necesite, se agrega la librería como una referencia.

- Uso: Se invocan las clases y métodos de la librería como si fueran parte del proyecto, ahorrando tiempo y esfuerzo.



QUE SON LOS COMPONENTES

Un componente es una parte modular y reutilizable de una aplicación que encapsula una funcionalidad específica.

Los componentes pueden ser visuales (como botones, cuadros de texto) o no visuales (como temporizadores o conectores de base de datos). Son esenciales para el desarrollo rápido y eficiente de aplicaciones de escritorio, ya que simplifican la creación de interfaces y la implementación de lógica

CARACTERÍSTICAS DE

LOS COMPONENTES:

- Reutilizables: Pueden usarse en diferentes partes de una misma aplicación o en proyectos distintos.
- Encapsulados: Ocultan su funcionamiento interno, exponiendo solo lo necesario a través de propiedades y métodos.
- Configurables: A través de sus propiedades (como tamaño, color, texto, etc.).
- Eventos Asociados: Permiten ejecutar acciones cuando ocurre una interacción, como un clic o el cambio de un valor.
- Visuales o No Visuales: Algunos se muestran en la interfaz gráfica, otros funcionan en segundo plano.

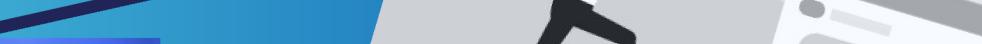
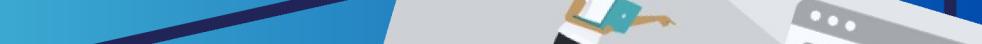
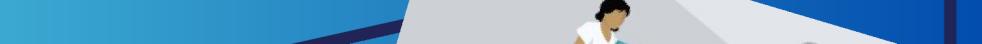
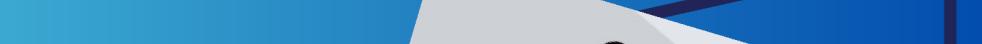
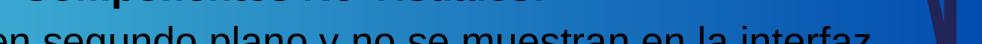
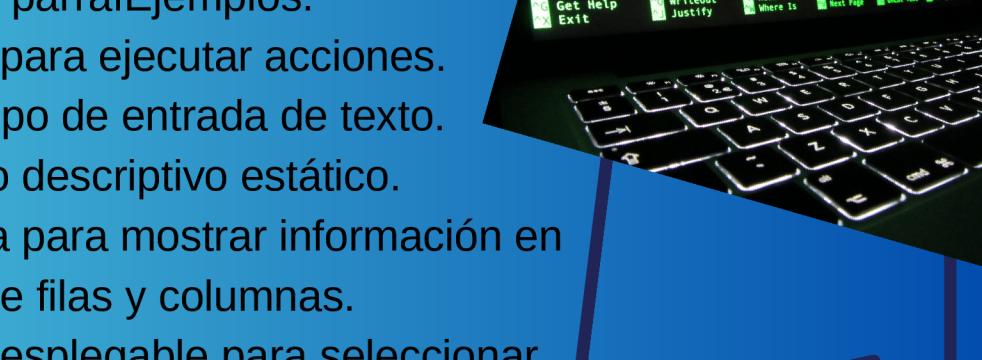
TIPOS DE COMPONENTES

componentes visuales:

El texto del párraf

Ejemplos:

- Button: Botón para ejecutar acciones.
- TextBox: Campo de entrada de texto.
- Label: Texto descriptivo estático.
- DataGridView: Tabla para mostrar información en formato de filas y columnas.
- ComboBox: Lista desplegable para seleccionar opciones.



CICLO DE VIDA DE UN COMPONENTE

1. Creación: Se instancia el componente en el código o desde el diseñador de Visual Studio.
2. Configuración: Se ajustan propiedades como tamaño, color, texto, eventos, etc.
3. Incorporación: Se agrega al formulario o contenedor correspondiente.
4. Ejecución: Responde a eventos y realiza sus funciones según la lógica implementada.
5. Destrucción: Se elimina cuando ya no es necesario, liberando recursos.



EVENTOS COMUNES EN COMPONENTES VISUALES:

- Click: Cuando se hace clic en el componente (ej.: Button).
- TextChanged: Cuando cambia el texto de un componente (ej.: TextBox).
- SelectedIndexChanged: Cuando cambia la selección de un componente (ej.: ComboBox).
- CellValueChanged: Cuando se modifica un valor en una celda (ej.: DataGridView).



VENTAJAS DEL USO DE COMPONENTES:

- Ahorro de Tiempo: Los componentes preconstruidos aceleran el desarrollo.
- Modularidad: Separan funcionalidades específicas en elementos independientes.
- Reutilización: Un componente puede usarse en múltiples proyectos.
- Facilidad de Mantenimiento: Los cambios en un componente se reflejan en todas sus instancias.



DESVENTAJAS:

- Sobrecarga: Usar demasiados componentes visuales puede afectar el rendimiento.
- Curva de Aprendizaje: Crear componentes personalizados complejos requiere experiencia.
- Dependencia: Cambios en componentes externos pueden afectar el proyecto.

