# Mostrar un formulario (Show y ShowDialog)

En C#, existen dos maneras principales de mostrar un formulario: usando Show() y ShowDialog().

* **Show()** permite abrir un formulario de forma **no modal**, es decir, el usuario puede seguir interactuando con otras ventanas de la aplicación mientras ese formulario está abierto. Es ideal cuando queremos que el formulario secundario complemente al principal sin bloquearlo (por ejemplo, un panel de información adicional o una ventana de ayuda).
* **ShowDialog()**, en cambio, abre el formulario de manera **modal**, bloqueando la interacción con el resto de la aplicación hasta que esa ventana se cierre. Se usa, por ejemplo, en formularios de configuración o ingreso de datos donde necesitamos forzar al usuario a terminar esa tarea antes de continuar.  
  Ambas opciones instancian el formulario en memoria y lo muestran, y su uso adecuado depende del flujo deseado de interacción entre ventanas.

# Ocultar un formulario (Hide)

La función Hide() oculta un formulario de la vista del usuario pero **no lo elimina de la memoria**. Esto significa que todas las variables, controles y datos del formulario siguen cargados y disponibles. Esta opción es muy útil cuando se necesita reutilizar el formulario más adelante sin tener que volver a cargarlo o reconfigurarlo. Por ejemplo, en una aplicación con múltiples pantallas navegables (como un sistema de menús), podríamos ocultar un formulario para mostrar otro, y luego volver a mostrar el anterior sin pérdida de estado. Sin embargo, hay que tener en cuenta que este enfoque **consume más memoria**, ya que los formularios ocultos siguen ocupando recursos del sistema.

# Cerrar un formulario (Close)

Cuando se llama a Close(), el formulario se **cierra y se libera de la memoria**, lo que implica que se destruye el objeto y sus datos. Si más adelante queremos volver a mostrar ese mismo formulario, será necesario **crear una nueva instancia**. Esto es útil en situaciones donde sabemos que ya no vamos a necesitar ese formulario nuevamente, o cuando queremos liberar recursos para optimizar el rendimiento de la aplicación. Por ejemplo, en formularios que se usan una sola vez durante la ejecución, como un formulario de bienvenida o de carga inicial. La ventaja de Close() es que evita la acumulación innecesaria de memoria, pero tiene la desventaja de perder el estado, a diferencia de Hide().

## Comparación y buenas prácticas

* Usá **Show()** cuando querés mostrar un formulario y permitir al usuario seguir usando la aplicación.
* Usá **ShowDialog()** cuando necesitás que el usuario complete algo antes de continuar.
* Usá **Hide()** si pensás reutilizar el formulario más adelante y no querés perder su estado.
* Usá **Close()** si ya no necesitás el formulario y querés liberar memoria.

Combinar correctamente estas funciones es clave para lograr una buena experiencia de usuario y eficiencia en el uso de recursos.

📌 **Ejemplo:**

formulario que muestra y oculta al “formularioSecundario”

**public** partial class Form1 : Form  
{

    // Se mantiene una instancia de Form2 para reutilizarla  
    **private** Form2 formularioSecundario;

**private** void btnMostrar\_Click(object sender, EventArgs e)  
    {  
        // Si no existe o ya fue cerrado, se crea una nueva instancia  
        **if** (formularioSecundario == **null** || formularioSecundario.**IsDisposed**)  
        {

// Se crea la instancia del formulario que queremos mostrar   
            formularioSecundario = **new** Form2();  
        }  
        formularioSecundario.Show(); // Mostrar (no modal)  
    }  
  
    **private** void btnOcultar\_Click(object sender, EventArgs e)  
    {

// Si esta creado el formulario se intenta ocultar  
        **if** (formularioSecundario != **null** && !formularioSecundario.**IsDisposed**)  
        {  
            formularioSecundario.Hide(); // Ocultar el formulario  
        }  
    }

    }

**Informacion:** Se utiliza **formularioSecundario.IsDisposed** para saber si el formulario se encuentra o no cerrado

**📝 Ejercitación**

Desarrollar una aplicación de escritorio en C# llamada **"Panel de Control de Usuario"**, que tenga al menos **dos formularios**:

* FormLogin: simula un formulario de inicio de sesión.
* FormPanel: simula el panel principal de la aplicación.

**Requisitos funcionales:**

1. En FormLogin, incluir:
   * Un campo de texto para el "Usuario" (TextBox).
   * Un botón llamado "Ingresar" que al hacer clic **oculte** el formulario de login y **muestre** el formulario FormPanel.
2. En FormPanel, incluir:
   * Una etiqueta que diga "Bienvenido “+**NOMBRE**+” al Panel de Usuario".
   * Un botón llamado "Cerrar sesión" que al hacer clic **oculte** el formulario del panel y **vuelva a mostrar** el formulario FormLogin.