

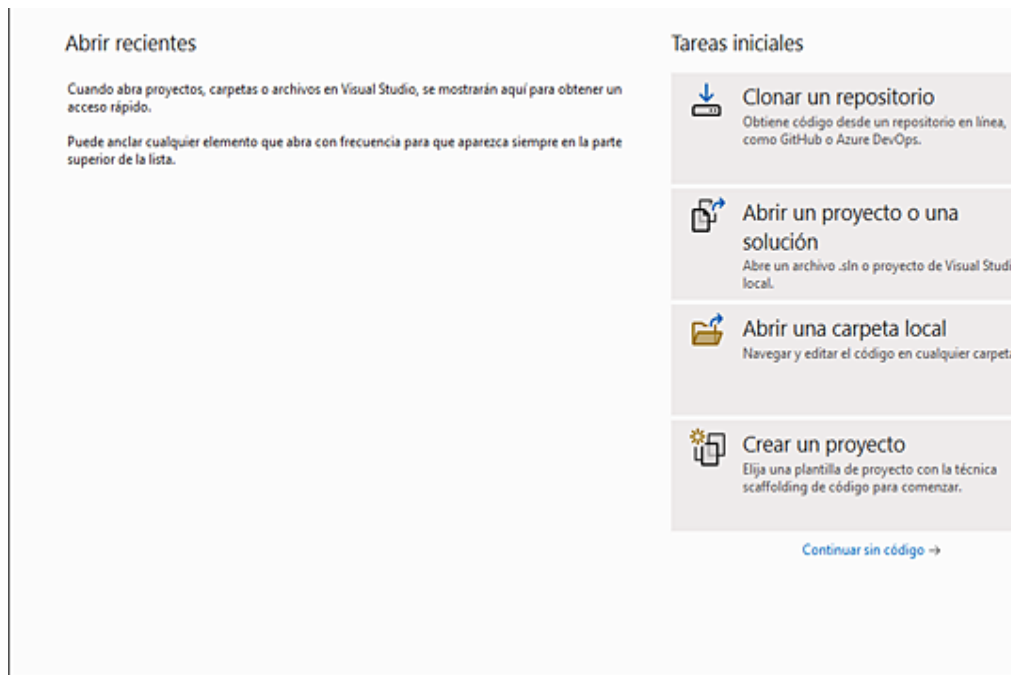
Visual Studio 2022

Visual Studio es la interfaz de desarrollo de Microsoft. Se compone de un conjunto de herramientas que permiten a los desarrolladores crear aplicaciones para las plataformas .NET. Visual Studio 2022 se distribuye en varias ediciones:

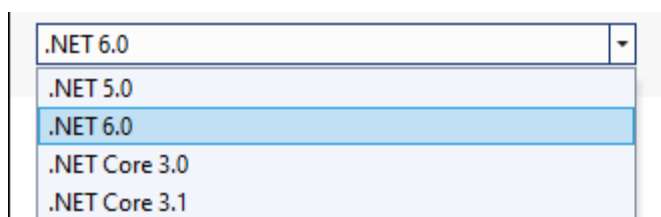
- **Community:** Microsoft proporciona gratuitamente esta edición de Visual Studio 2022. Su objetivo es servir en la formación de los estudiantes, desarrolladores open source (código abierto) y desarrolladores particulares. Reúne todas las funcionalidades básicas para la creación de proyectos. Esta edición contiene todas las herramientas de desarrollo multiplataforma para las aplicaciones móviles Windows, iOS y Android. La galería Visual Studio permite acceder a numerosas herramientas, modelos y controles para acelerar el desarrollo.
- **Professional:** edición dirigida a desarrolladores profesionales individuales o equipos pequeños de trabajo. Las funcionalidades son las mismas que para la edición Community, pero destinada a las empresas.
- **Enterprise:** para los equipos profesionales que trabajan en proyectos que necesitan más interacción entre sus miembros.

La interfaz de desarrollo

La página de inicio se ha rediseñado para mejorar la experiencia del usuario y ayudar a los desarrolladores que no están familiarizados con Visual Studio a iniciar rápidamente un nuevo proyecto o clonar un repositorio remoto:



Las aplicaciones pueden especificar las versiones del Framework .NET que se deben usar. En función de las opciones de instalación elegidas, Visual Studio 2022 propone las versiones del Framework .NET instaladas en su máquina y ofrece la instalación de nuevas versiones del Framework, si es necesario.



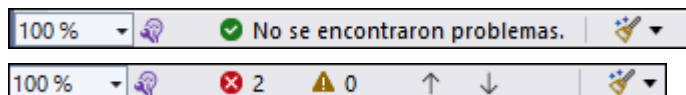
La interfaz de Visual Studio tiene varias herramientas imprescindibles para el desarrollo de aplicaciones.

1. El editor de texto

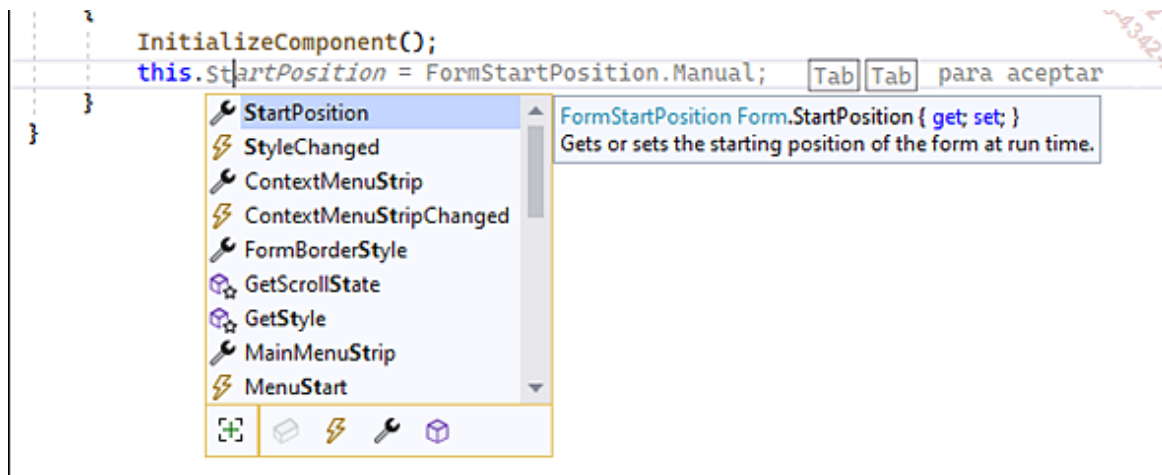
El editor de texto de Visual Studio es una potente herramienta que permite introducir el código de la aplicación. Las palabras clave y los tipos de datos se colorean. Esto permite facilitar la lectura y la comprensión del código. A medida que se escribe el código, el editor evalúa los errores de sintaxis, las variables declaradas que no se usan en el código y muestra IntelliSense.

La interfaz de usuario se ha actualizado para dejar más espacio para el editor de código.

La calidad del documento que se está viendo se anuncia utilizando el icono en la parte inferior del documento para indicar si contiene sugerencias, advertencias o errores. Puede navegar a través de las correcciones propuestas gracias al menú contextual, un doble clic para cambiar de una a la otra y un [Mayús] clic para volver a la anterior.



IntelliSense es una funcionalidad que permite mostrar las clases y sus miembros en relación con el código introducido, así como los argumentos y las posibles sobrecargas para los métodos:



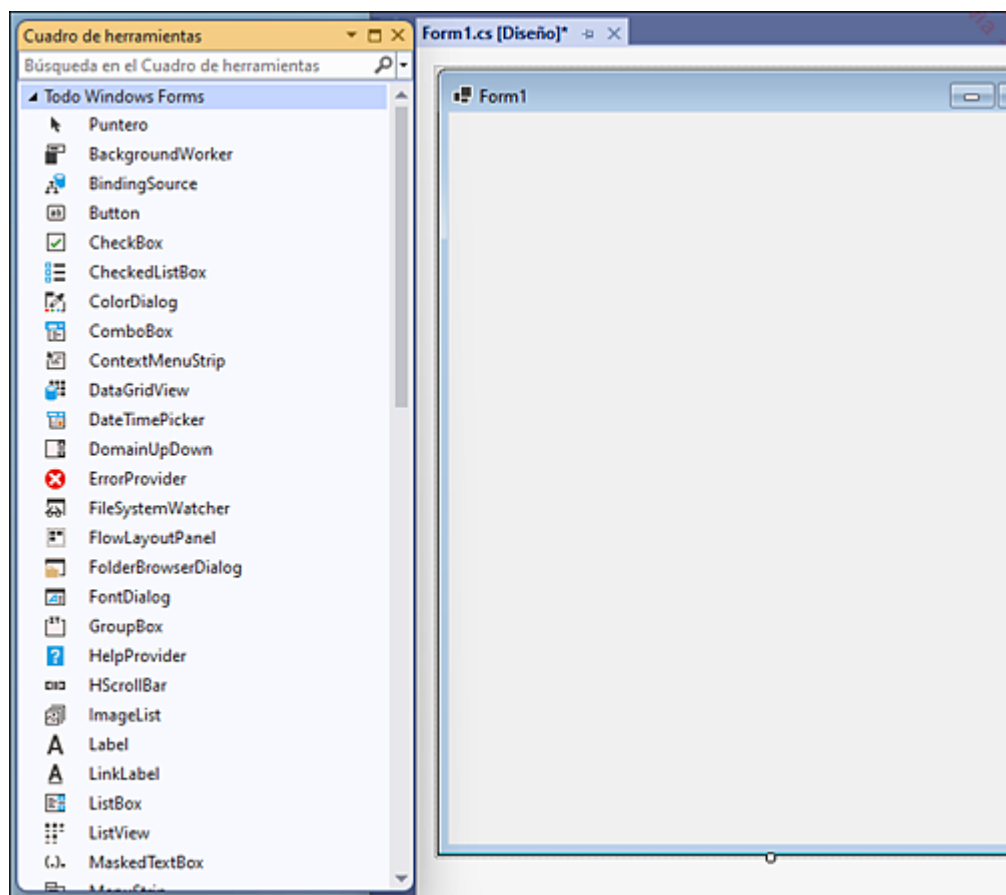
Los iconos situados a la izquierda de los nombres de los miembros permiten definir visualmente si el miembro es un método, una propiedad, una variable, un evento u otro tipo de miembro. Cuando un miembro se destaca, aparece un tooltip (descripción emergente) explicativo que muestra una presentación del método, los argumentos que se esperan e incluso los tipos de excepciones que se pueden producir.

Las bombillas aparecen en el margen del editor y permiten acceder a todas las acciones rápidas, la corrección de problemas de código habituales y la refactorización del código. Cuando Visual Studio considera que es necesario, aparece una bombilla en el margen y puede mostrar las acciones y sugerencias con el ratón o mediante el atajo de teclado [Ctrl][.].

La función de búsqueda global (accesible mediante el atajo [Ctrl] **T** o vía el menú **Edición e Ir a**), que permite buscar tipos, métodos u otros elementos en los archivos tiene en cuenta las correspondencias en caso de error ortográfico al introducir los términos de búsqueda:

2. El diseñador de vistas

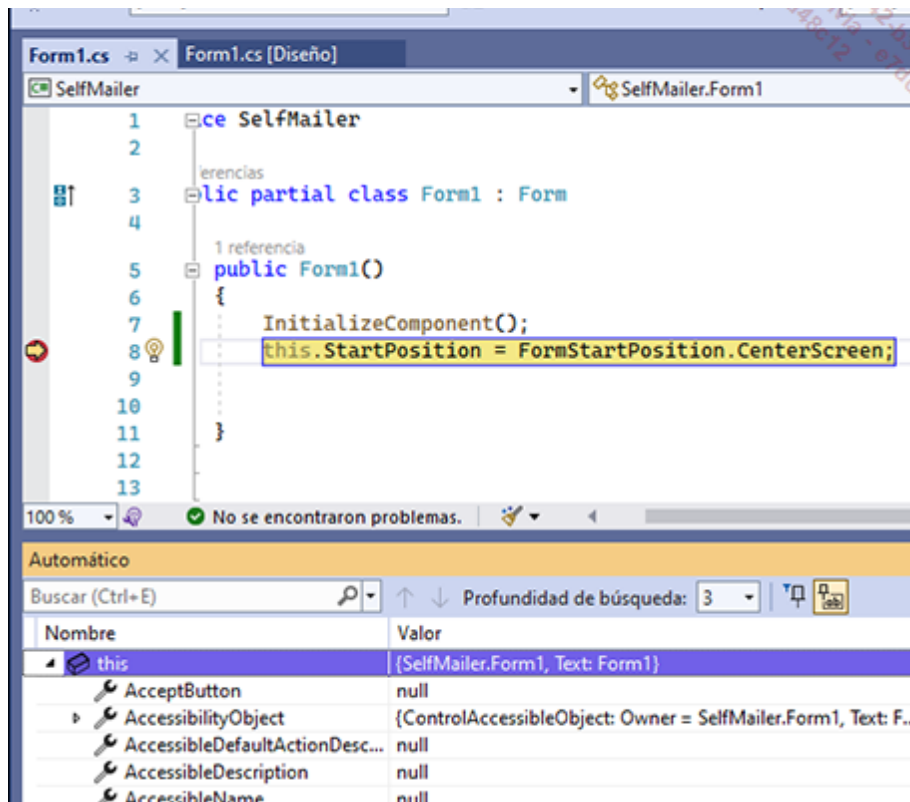
El diseñador de vistas permite ubicar gráficamente los elementos en los formularios, arrastrándolos desde la caja de herramientas:



Visual Studio genera automáticamente el código necesario para instanciar los controles y ubicarlos. El diseñador de vistas es, en cierto modo, una herramienta que interpreta el código para darle una representación visual.

3. El depurador integrado

El depurador integrado de Visual Studio permite probar y trazar la ejecución de la aplicación, tener puntos de interrupción y seguir paso a paso la ejecución del código, monitorizar y modificar manualmente las variables del código en ejecución.



Cuando la aplicación se ejecuta desde Visual Studio con la tecla [F5] (**Iniciar depuración**), Visual Studio añade su depurador a los procesos de la aplicación y permite controlarlos.

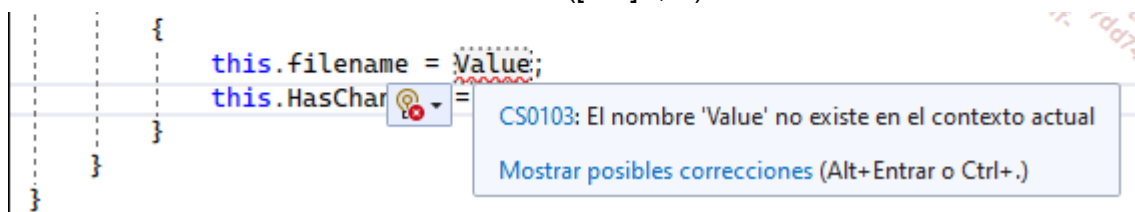
Los tipos de errores

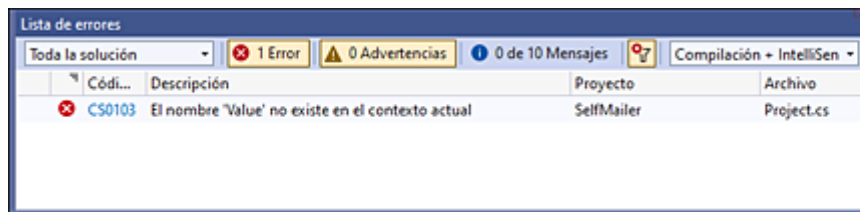
Sea cual sea la experiencia de un desarrollador, los errores son inevitables. La depuración permite localizarlos y explicarlos para poder corregirlos. Visual Studio incluye herramientas que permiten trazar los errores a lo largo del ciclo de desarrollo de una aplicación.

Existen tres tipos de errores principales durante el diseño de una aplicación: los errores de sintaxis, que corresponden al código que el compilador no puede interpretar, los errores de ejecución, que se producen durante la ejecución de la aplicación y los errores de lógica, que producen resultados no deseados.

1. Los errores de sintaxis

Visual Studio traza los errores de sintaxis en el editor de texto antes de la compilación. Se puede tratar de un error en el nombre de una variable o una instrucción no válida. Visual Studio facilita la identificación y localización de esos errores, destacándose en el código y añadiéndolos a la ventana Lista de errores ([Ctrl] °, E).





La ventana Lista de errores enumera todos los errores de la aplicación, dando a cada uno de ellos una descripción, el archivo en el que se encuentra la instrucción, así como su línea. Haciendo doble clic en un error de la lista, se abre el archivo relacionado y se sitúa el cursor en la instrucción que ha causado el error. Los errores se clasifican en tres categorías: errores, advertencias y mensajes. Los errores impiden la compilación de la aplicación, a diferencia de las advertencias y los mensajes.

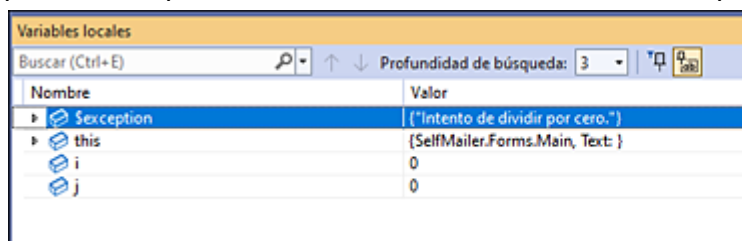
2. Los errores de ejecución

Los errores de ejecución se producen cuando la aplicación intenta hacer una operación inválida, teniendo en cuenta el valor de las variables y el contexto de ejecución. Por ejemplo, la división por cero:

```
int i = 0;
int j = 10 / i;
```

La compilación se realiza con éxito, pero durante la ejecución se genera una excepción, ya que la división por cero está prohibida.

Tan pronto como se produce una excepción no gestionada, la ejecución de la aplicación se interrumpe y Visual Studio muestra una descripción del error. La ventana Variables locales permite comprobar el valor de las variables cuando se produce el error:



Es posible modificar esos valores y retomar, a continuación, la ejecución de la aplicación. Haciendo clic en el valor de la variable *i* en la ventana Variables locales y modificando el valor por 2 podemos retomar la ejecución de la aplicación ([F5]). La ejecución ya no devuelve el error y se asigna el valor 5 a la variable *j*.

La modificación de los valores de las variables locales tiene como objetivo realizar ensayos para identificar un error complejo. El código original no se modifica en ningún caso.

Para facilitar la depuración, la ventana Variables locales tiene una barra de búsqueda con la posibilidad de elegir la profundidad de búsqueda en los diferentes objetos monitoreados.

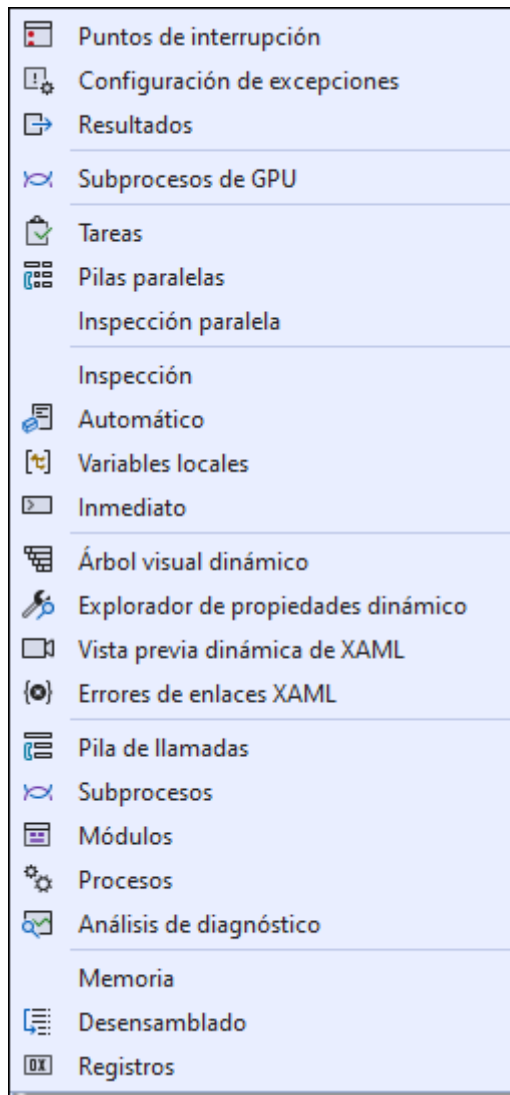
3. Los errores de lógica

La aplicación se ejecuta sin errores y, por tanto, el resultado no es el esperado. Los errores de lógica son los más difíciles de detectar y corregir, ya que no tendrá ninguna pista sobre

su origen. Visual Studio también tiene herramientas para ayudar a la localización de esos errores, como el depurador y las pruebas unitarias.

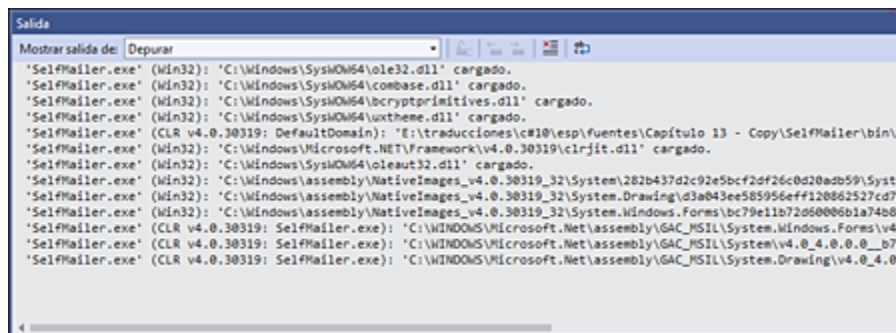
Las ventanas

El menú Depurar - Ventanas contiene varias herramientas que facilitan la depuración del código. Algunas de estas ventanas están disponibles en modo diseño y depuración, pero a la mayor parte solo se puede acceder en modo depuración.



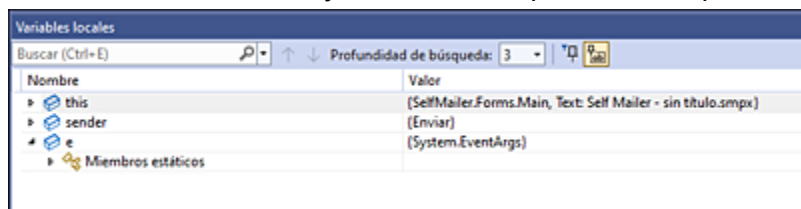
1. La ventana Salida (Resultados)

La ventana Salida muestra el conjunto de información de compilación y ejecución de la aplicación. Esto incluye la generación de la aplicación, así como la salida de las instrucciones Debug y Trace:



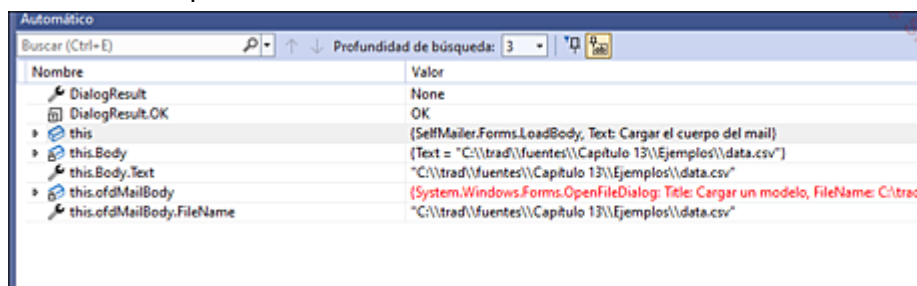
2. La ventana Variables locales

La ventana Variables locales ([Ctrl][Alt] V, L) muestra las variables del método actual. Las variables se presentan en una tabla con su nombre, valor y tipo. Para los objetos complejos, el signo + situado a la izquierda del nombre de la variable permite desplegar el árbol de los miembros para mostrar su valor. El valor de una variable aparece en rojo cuando se acaba de modificar durante la ejecución de la aplicación o a partir de la propia ventana:



3. La ventana Automático

La ventana Automático ([Ctrl][Alt] V, A) se parece a la ventana Variables locales, con la diferencia de que solo se muestran las variables de las instrucciones actual y anterior:







4. La ventana Inspección

La ventana Inspección ([Ctrl][Alt] W) se presenta de la misma manera que las ventanas Variables locales y Automático, y permite monitorizar las variables de su elección, incluso si están fuera de ámbito.

Para añadir una variable a esta ventana, seleccione una variable o una expresión y abra el menú contextual. Seleccione la opción Añadir inspección.

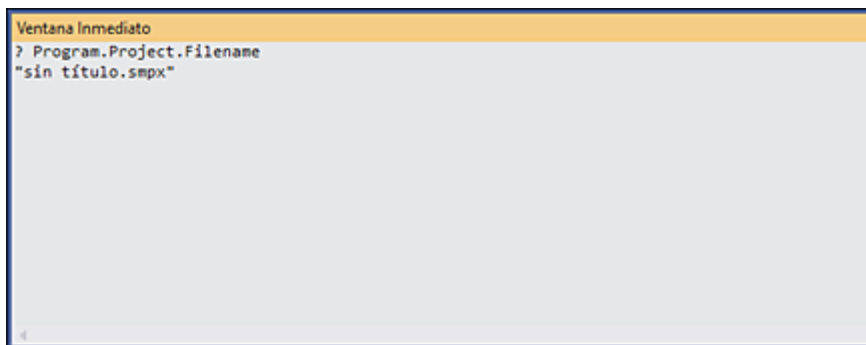
Visual Studio ofrece cuatro ventanas de Inspección, que permiten organizar las variables de diferentes maneras:

	Inspección 1	Ctrl+Alt+W, 1
	Inspección 2	Ctrl+Alt+W, 2
	Inspección 3	Ctrl+Alt+W, 3
	Inspección 4	Ctrl+Alt+W, 4

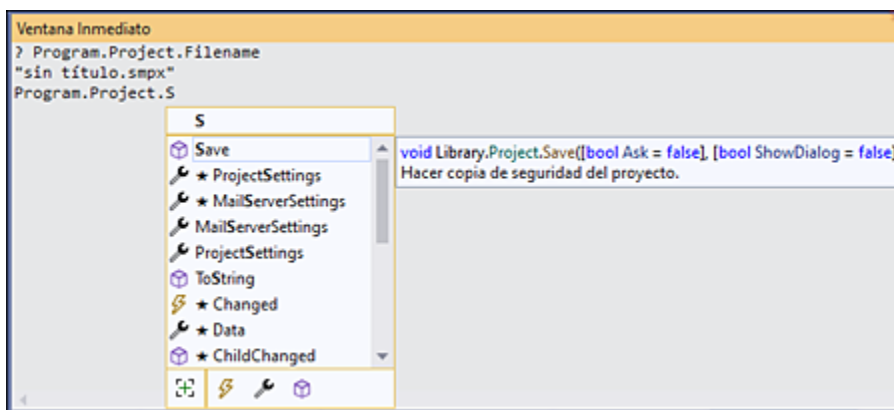
5. La ventana Inmediato

La ventana Inmediato ([Ctrl][Alt] I) permite ejecutar las instrucciones, evaluar las expresiones o modificar el valor de las variables en la línea de comandos.

Para mostrar el valor de una variable, comience la instrucción con el carácter ?:

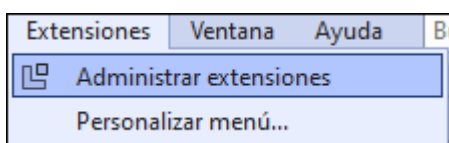


Para ejecutar un método o modificar el valor de una variable, escriba la instrucción directamente en la ventana Inmediato. IntelliSense está disponible en esta ventana, pero los errores de sintaxis no se resaltan:

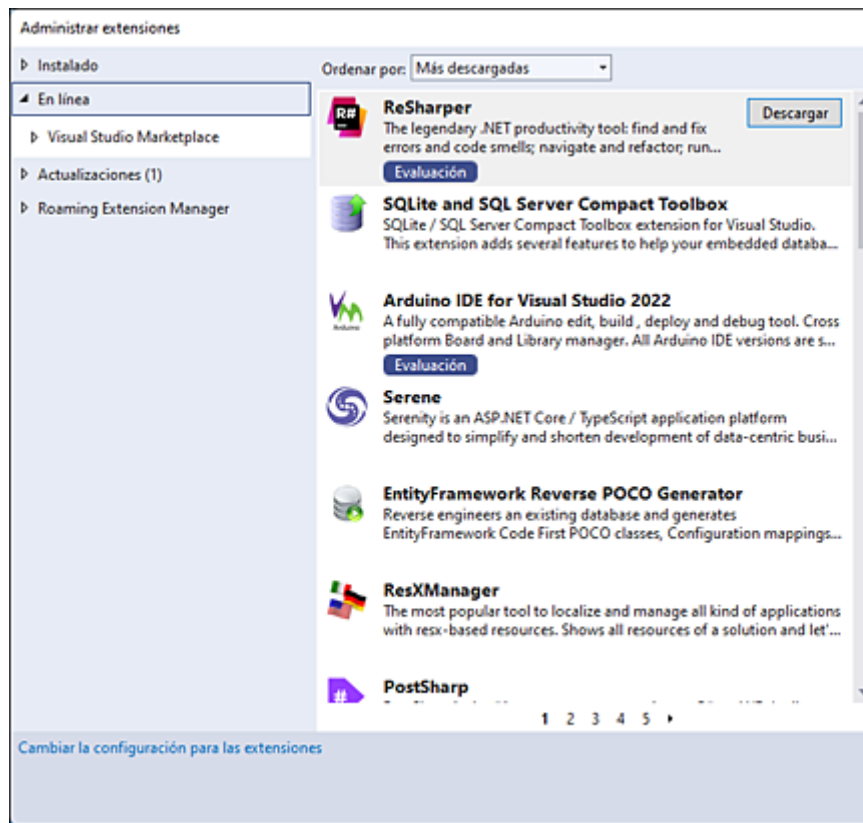


4. El administrador de extensiones

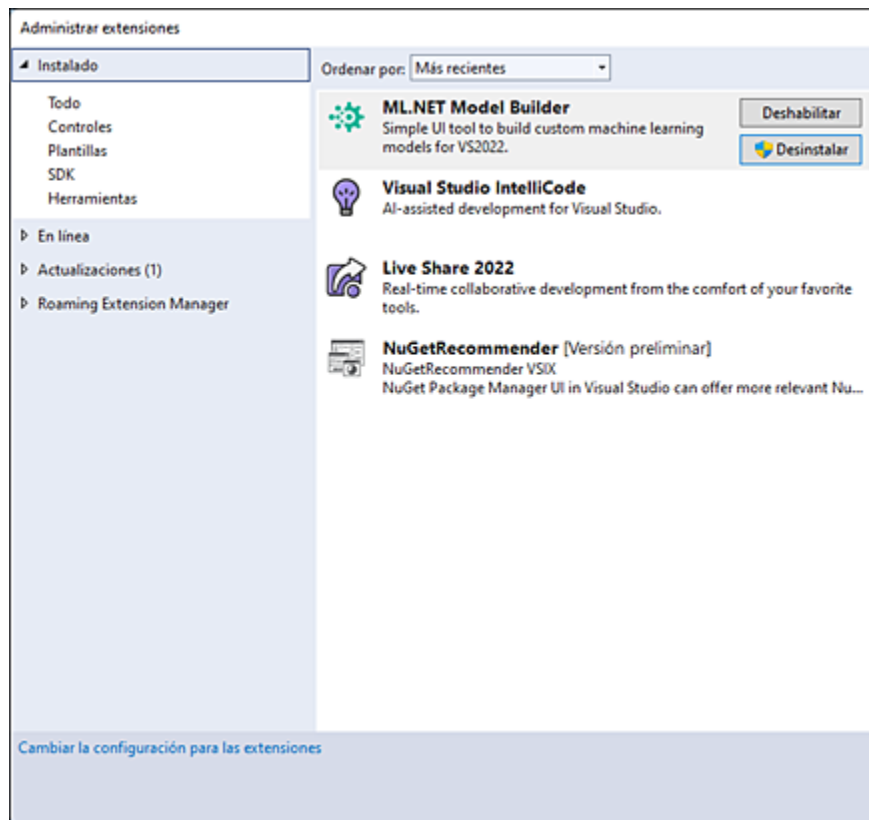
El administrador de extensiones permite añadir funcionalidades a Visual Studio desde una galería online. Se puede acceder al administrador de extensiones desde el menú **Extensiones** seleccionando, a continuación, **Administrar extensiones**:



La pestaña **En línea** ofrece multitud de extensiones. Ya sean controles, modelos o herramientas, basta con recorrer la galería y pulsar en el botón **Descargar**.

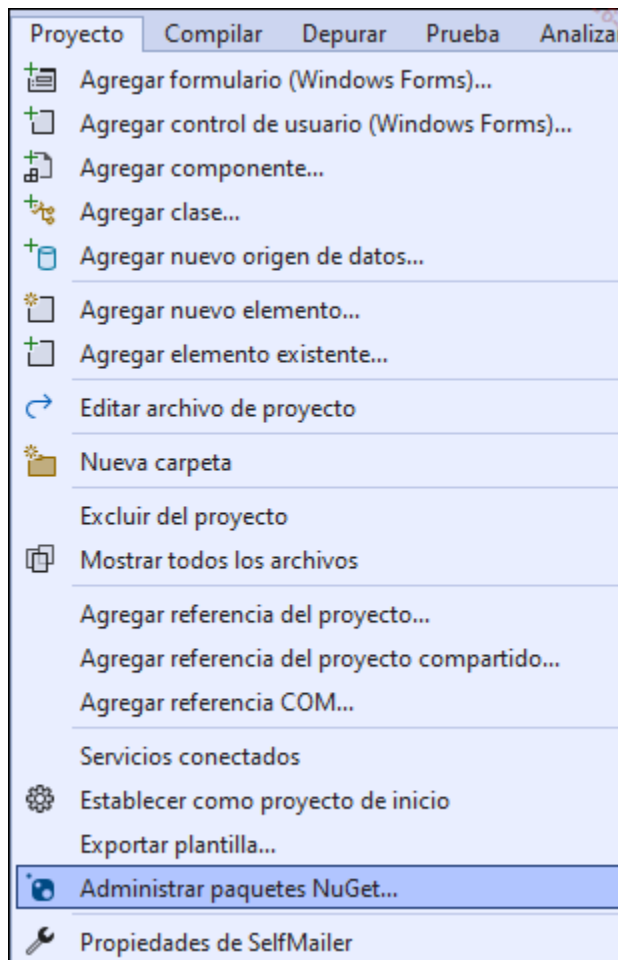


La pestaña **Instalado** permite eliminar la instalación o deshabilitar las extensiones descargadas.



5. NuGet

Visual Studio integra NuGet, que se basa en el principio del administrador de extensiones, cuyos módulos actuarán a nivel del proyecto, y no de Visual Studio. Se puede acceder al administrador de módulos NuGet desde el menú **Proyecto** y después **Administrar paquetes NuGet...**:



De esta manera, tendrá acceso a un gran número de módulos, tan fáciles de instalar como las extensiones.

La ventaja de añadir un componente a su proyecto usando el administrador de módulos NuGet es que podrá obtener la última versión en línea de estos. Además, el administrador gestiona las dependencias entre módulos. Si el módulo que desea utilizar necesita otro módulo, este último también se añadirá a su solución. Las dependencias entre módulos se muestran en la parte inferior de la columna de la derecha, que describe el módulo.

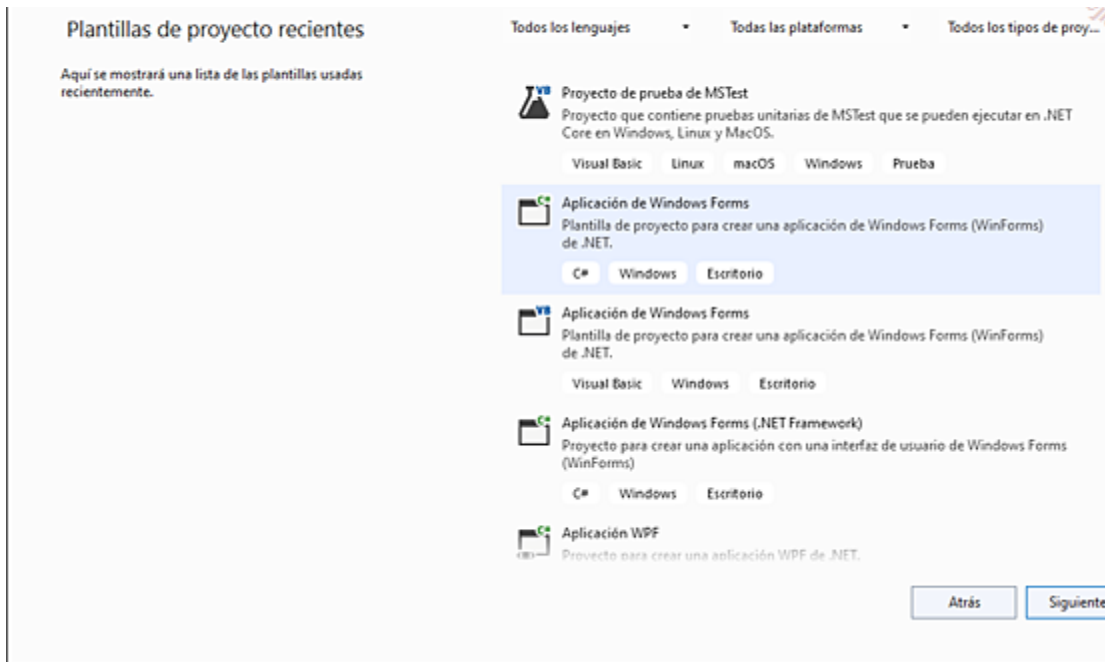
La presencia de la pestaña **Actualización** arriba de la ventana permite comprobar si hay versiones más recientes de los módulos ya instalados. Si existen, la actualización es tan sencilla como la instalación, ya que solo es necesario pulsar un botón para comenzar la actualización.

La creación de soluciones

Trabajando con Visual Studio es raro comenzar a partir de una solución vacía. Visual Studio ofrece plantillas de proyecto. Estas plantillas contienen los elementos predeterminados, las referencias y configuraciones para el tipo deseado.

La selección del tipo de proyecto se hace durante su creación. El siguiente cuadro de diálogo permite realizar la selección:

Archivo - Nuevo - Proyecto...



Vamos a crear un tipo de proyecto **Aplicación de Windows Forms**.

Tan pronto como se produce la validación de la elección, Visual Studio abre la solución con los archivos básicos, un formulario **Form1.cs**, así como un archivo que es el punto de entrada principal de la aplicación, llamado **Program.cs**.

De momento este proyecto no tiene ninguna funcionalidad. Cuando se ejecuta con [F5], aparece el formulario **Form1** vacío. Visual Studio ya ha hecho la integración básica durante la creación del proyecto, insertando el siguiente código en el archivo **Program.cs**:

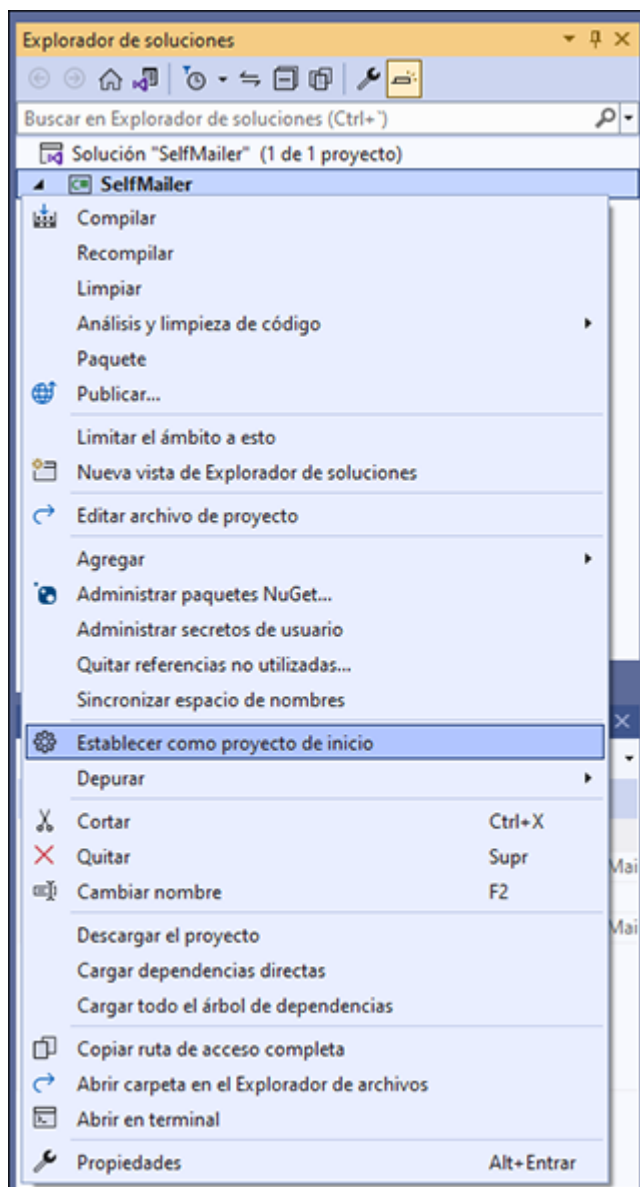
```
namespace WinFormsApp1
{
    0 referencias
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        0 referencias
        static void Main()
        {
            // To customize application configuration such as set high DPI settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}
```

1. Definir el punto de entrada

Cada aplicación ejecutable, a diferencia de lo que sucede con las bibliotecas de clases, deben tener un punto de entrada. Este punto de entrada es el **método Main** presentado en el ejemplo de la sección anterior, que permite administrar el formulario.

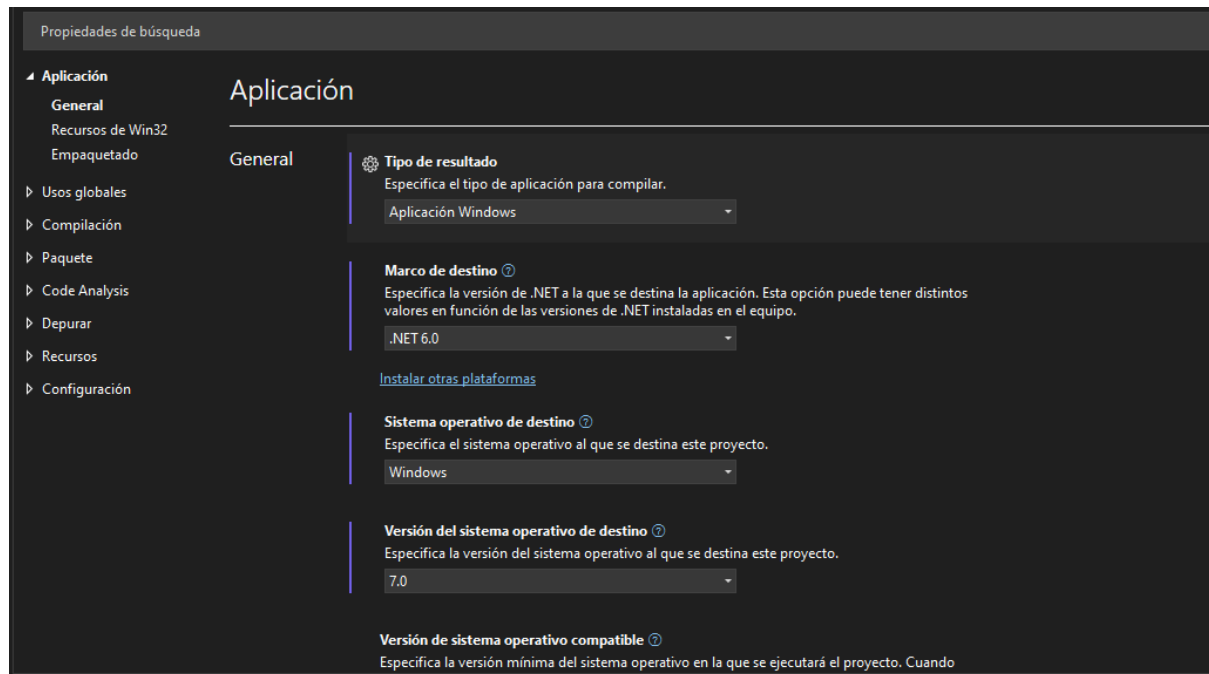
2. La diferencia entre proyectos y soluciones

Un proyecto es un conjunto de archivos que se compilarán en un único ensamblado. Una solución es un conjunto de uno o varios proyectos. De la misma manera que un proyecto tiene un punto de entrada, una solución tiene un proyecto de inicio. Este proyecto se identifica en el explorador de soluciones porque su nombre está en negrita. Para modificar esta propiedad, haga clic con el botón derecho del ratón sobre el proyecto. Esto le permitirá seleccionar la opción **Establecer como proyecto de inicio** en el menú contextual.



3. Configurar el proyecto

Para configurar el proyecto, haga clic con el botón derecho del ratón sobre el proyecto en el explorador de soluciones y después seleccione la opción **Propiedades** del menú contextual.



Esta ventana contiene las pestañas que permiten configurar las diferentes partes del proyecto.

La pestaña **Aplicación** aplica a las propiedades generales de la aplicación:

- **Nombre de ensamblado:** puede especificar el nombre del ensamblado que se generará y se añadirá la extensión **.exe** o **.dll**, dependiendo del tipo de proyecto.
- **Espacio de nombres predeterminado:** define el espacio de nombres que se usará cuando se cree una nueva clase en el proyecto:
- **Plataforma de destino:** permite especificar qué versión del Framework .NET se usará para administrar la solución.
- **Tipo de salida:** este tipo se elige en función del tipo de aplicación que se va a generar. Si bien esta propiedad se puede modificar después de la creación del proyecto, es preciso aportar los cambios correspondientes al punto de entrada de la aplicación y al método `Main()`, así como a las referencias del proyecto.
- **Objeto de inicio:** en caso de que haya varios métodos `Main()` en el proyecto, esta propiedad permite especificar cuál será el punto de entrada.