Imagen que contiene interior, mamífero, gato, viendo

Descripción generada automáticamente

**Tema**

PLATAFORMA .NET

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS.MDU.CCNA.CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

01/02/2023

**PLATAFORMA** .**NET**

[PLATAFORMA .NET 4](#_Toc125935374)

[INTRODUCCIÓN 4](#_Toc125935375)

[OBJETIVOS 4](#_Toc125935376)

[OBJETIVO GENERAL 4](#_Toc125935377)

[OBJETIVOS ESPECIFICOS 4](#_Toc125935378)

[MARCO TEÓRICO 4](#_Toc125935379)

[PLATAFORMA .NET 4](#_Toc125935380)

[ARQUITECTURA .NET 5](#_Toc125935381)

[PCL (BIBLIOTECA DE CLASES PORTABLE) 6](#_Toc125935382)

[ENTORNO DE TIEMPO DE EJECUCIÓN 7](#_Toc125935383)

[INFRAESTRUCTURA COMÚN 7](#_Toc125935384)

[HERRAMIENTAS 7](#_Toc125935385)

[LENGUAJES DE PROGRAMACIÓN 8](#_Toc125935386)

[DOCUMENTACIÓN 10](#_Toc125935387)

[INTEROPERABILIAD DE LENGUAJES DE PROGRAMACIÓN 10](#_Toc125935388)

[PARTE PRÁCTICA 11](#_Toc125935389)

[EJECUCIÓN DEL PROYECTO 47](#_Toc125935390)

[ESTRUCTURA DE LA APLICACIÓN 49](#_Toc125935391)

[CONCLUSIONES 49](#_Toc125935392)

[RECOMENDACIONES 49](#_Toc125935393)

[BIBLIOGRAFÍA 49](#_Toc125935394)

**INDICE DE IMÁGENES**

[Figura 1. Logotipo Identificativo de Microsoft .NET 5](#_Toc125935157)

[Figura 2. Arquitectura de Microsoft .NET 5](#_Toc125935158)

**ÍNDICE DE TABLAS**

[Tabla 1. Resumen comandos útiles dentro de Java 43](#_Toc118731587)

# PLATAFORMA .NET

# INTRODUCCIÓN

# OBJETIVOS

## OBJETIVO GENERAL

Identificar la infraestructura y herramientas de la plataforma .NET, también comparar el lenguaje de programación C# con otros lenguajes y crear una aplicación en el que se use la interoperabilidad de lenguajes de programación C#, F# y Visual Basic.

## OBJETIVOS ESPECIFICOS

* Conocer

# MARCO TEÓRICO

## PLATAFORMA .NET

Las personas no saben que es en realidad .NET o varias piensan que .NET es lo mismo que .NET Framework, a inicios del año 2002 fue así, pero ha ido evolucionando al pasar del tiempo y esto va mucho más allá. Para ser un programador de .NET debes saber los fundamentos arquitectura y los componentes, la verdad es que .NET es todo un mundo y tomaría años en aprenderla a nivel profesional y de forma completa.

.NET es una plataforma de desarrollo, es decir un ambiente o entorno de software común en el que se puede crear aplicaciones por medio de la programación, APIs y bibliotecas. Para entender mejor se puede suponer que la plataforma .NET es igual a una mesa de costura, entonces la mesa sería la plataforma, pero con tener la mesa de costura no se puede hacer vestidos, se necesita una máquina de coser, hilos, tijeras, cinta de medida, etcétera. Al tener estos componentes o herramientas se puede construir el vestido, dicho esto se puede decir que en la plataforma de desarrollo de .NET se dispone de una serie de lenguajes, implementaciones bibliotecas y herramientas. Con todo ello podemos desarrollar diversas aplicaciones en diferentes lenguajes de programación e implementaciones .NET.



Figura 1. Logotipo Identificativo de Microsoft .NET

La plataforma .NET puede usarse para construir muchos tipos de aplicaciones, como web, móvil, de escritorio, juegos, IoT (internet de las cosas), aquí se encuentran todos los dispositivos que se puede conectar a internet como Smart TV impresora refrigeradoras relojes etcétera. por último, podemos crear aplicaciones de Machine Learning, estos son sistemas de inteligencia artificial, como ejemplo básico se puede mencionar a Cortana de Microsoft o Siri de iOS. algunas de estas aplicaciones son multiplataformas y otras son específicas para un sistema operativo.

## ARQUITECTURA .NET

La arquitectura de .NET se caracteriza principalmente por sus implementaciones, el primero y el que la gran mayoría conoce es .NET Framework. Esta es una plataforma de desarrollo para compilar aplicaciones web con ASP.NET y con WPF Windows Forms, también se puede hacer aplicaciones en la plataforma Windows, Windows Mobile, Windows Server y Azure. Esta contiene una biblioteca de clases base como por ejemplo .NET SqlConnection, DataTable, etcétera.

Graphical user interface, application

Description automatically generated

Figura . Arquitectura de Microsoft .NET

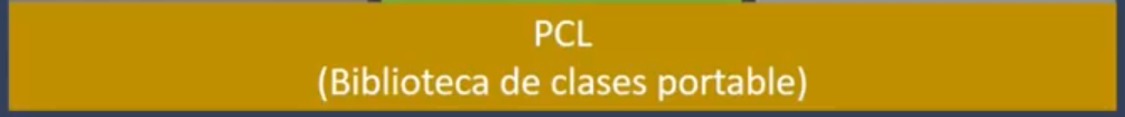
La segunda implementación es .NET CORE, aquí se puede desarrollar aplicaciones para la plataforma universal de Windows, por ejemplo, PC, Móviles, IoT, Xbox y Surface. .NET CORE también sirve para desarrollar aplicaciones en la nube y es compatible con Windows, Mac OS, Linux. de igual manera esta su biblioteca CORE FX.

Como última implementación esta MONO/XAMARIN, con esta herramienta se puede crear aplicaciones móviles de plataforma cruzada para Mac OS, Android, Windows móvil y todo escrito en el lenguaje C#. Esto es una ventaja ya que no se requiere aprender lenguajes para Android o iOS completamente, de igual manera se tiene su biblioteca de clases MONO. También están PCL que son bibliotecas de clases portátil, con ella se puede compartir el código entre varios proyectos específicos de la plataforma, tanto a Android, Windows o Windows Phone.

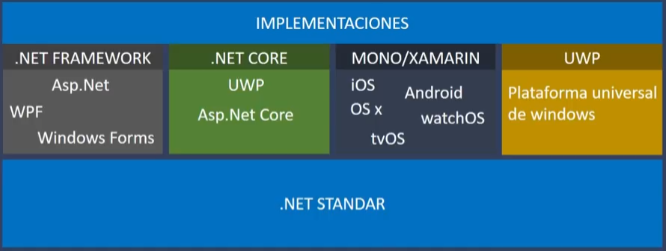


## PCL (BIBLIOTECA DE CLASES PORTABLE)

Se puede leer una pequeña descripción y con qué plataformas será compatible. En estos proyectos se pueden crear clases, métodos y poder usarla en cualquier plataforma que se haya seleccionado, esto ahorraría mucho tiempo ya que no se tendría que hacer código para cada plataforma, pero la PCL presentan muchas desventajas. Si la clase usa una clase que es netamente de .NET Framework, esta no podrá ser usada en .NET CORE. Para esto se tendría que reemplazar con uno equivalente en .NET CORE.



Para solucionar esto, la fundación .NET creó una API .NET Standard, se podría decir que es una fusión de las bibliotecas de clases base y las bibliotecas de clases portable, esto quiere decir que es compatible con todas las implementaciones, por ejemplo, la clase DataSet puede ser usada por .NET CORE, MONO, o .NET Framework, pero esto depende de las versiones. En la última versión de .NET estándar 2.0 se puede usar casi todas las clases en todas las plataformas, esto quiere decir que podemos tener una sola clase para todas las plataformas, así que no se tendría que escribir código esta forma ahorrando mucho tiempo, aunque en la última versión de Visual Studio, la PCL quedó obsoleta y borrada de .NET.



## ENTORNO DE TIEMPO DE EJECUCIÓN

Aun hay APIs adicionales que son específicas de los sistemas operativos en los que se ejecuta, por ejemplo, .NET Framework para Windows incluye una app para acceder al registro de Windows y así sucede en cada plataforma. Existen APIs específicas para cada plataforma que no pueden ser usadas por otros, se tienen los entornos de ejecución que es en donde se ejecuta el programa administrador para .NET Framework CLR (Common Language Runtime), para .NET CORE (CoreCommon Language Runtime) y para UWP.NET nativo usa una pequeña parte de la biblioteca CORE de .NET CORE por el cual muchos lo consideran dentro de la implementación de .NET CORE, pero sería en sí una implementación más

Graphical user interface

Description automatically generated

## INFRAESTRUCTURA COMÚN

Por último, está la infraestructura común, los programadores tienen tres lenguajes de programación, C#, F# y Visual Basic, luego esta MS BUILD que es el motor de compilación para los proyectos de punto .NET.



## HERRAMIENTAS

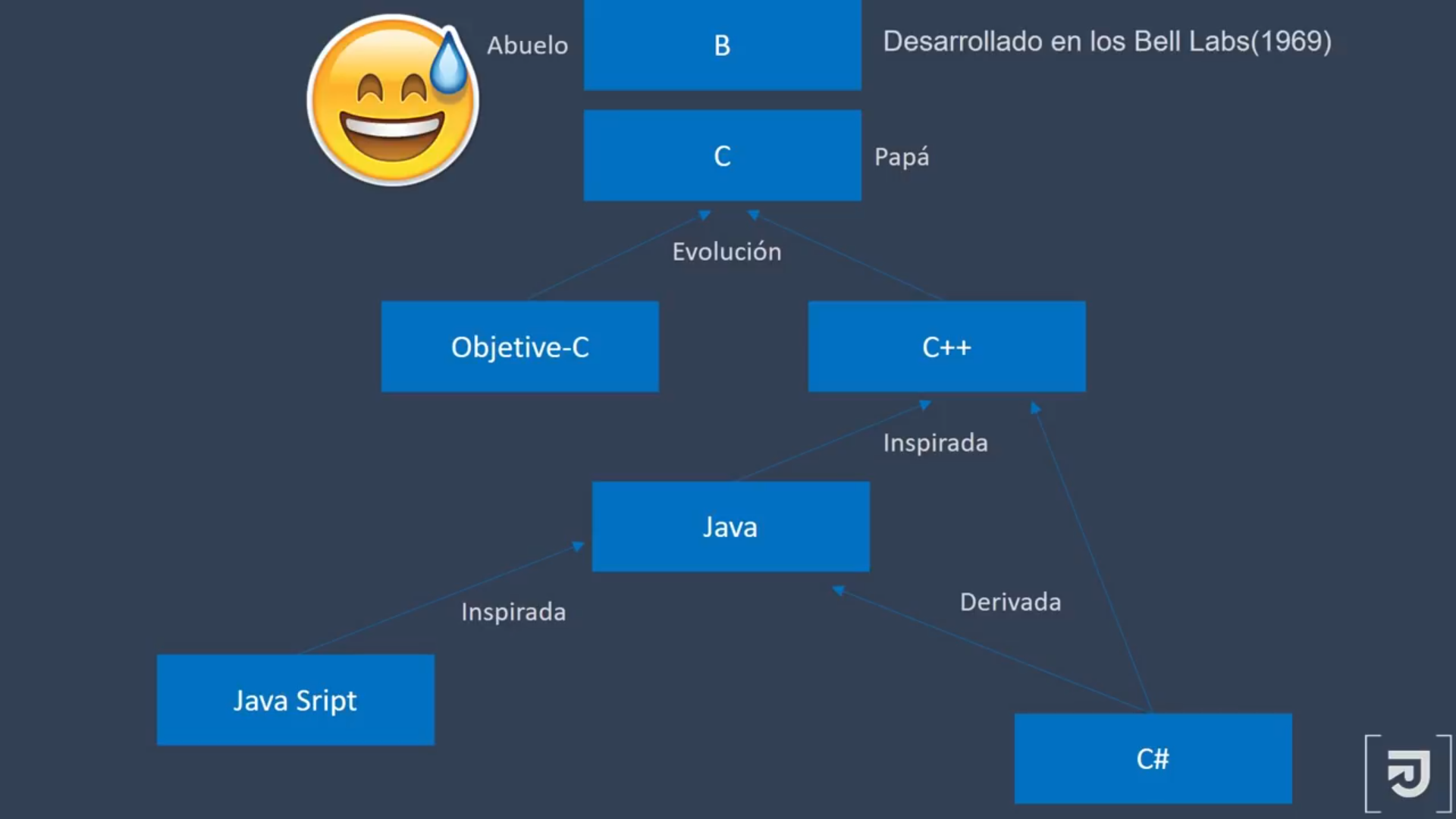
Nugget es un administrador de paquetes de Microsoft para .NET, con esta herramienta se puede descargar diferentes bibliotecas que contiene un código compilado como archivos DLL, por ejemplo, archivos EntityFramework. Se puede descargar desde un navegador o el propio Visual Studio. Los IDEs como Visual Studio, Xamarin Studio, Visual Studio para Mac, Jet Brains Rider, Visual Studio Code y el plugin OmniSharp que se puede integrar a los editores de código y empezar a programar en .NET.



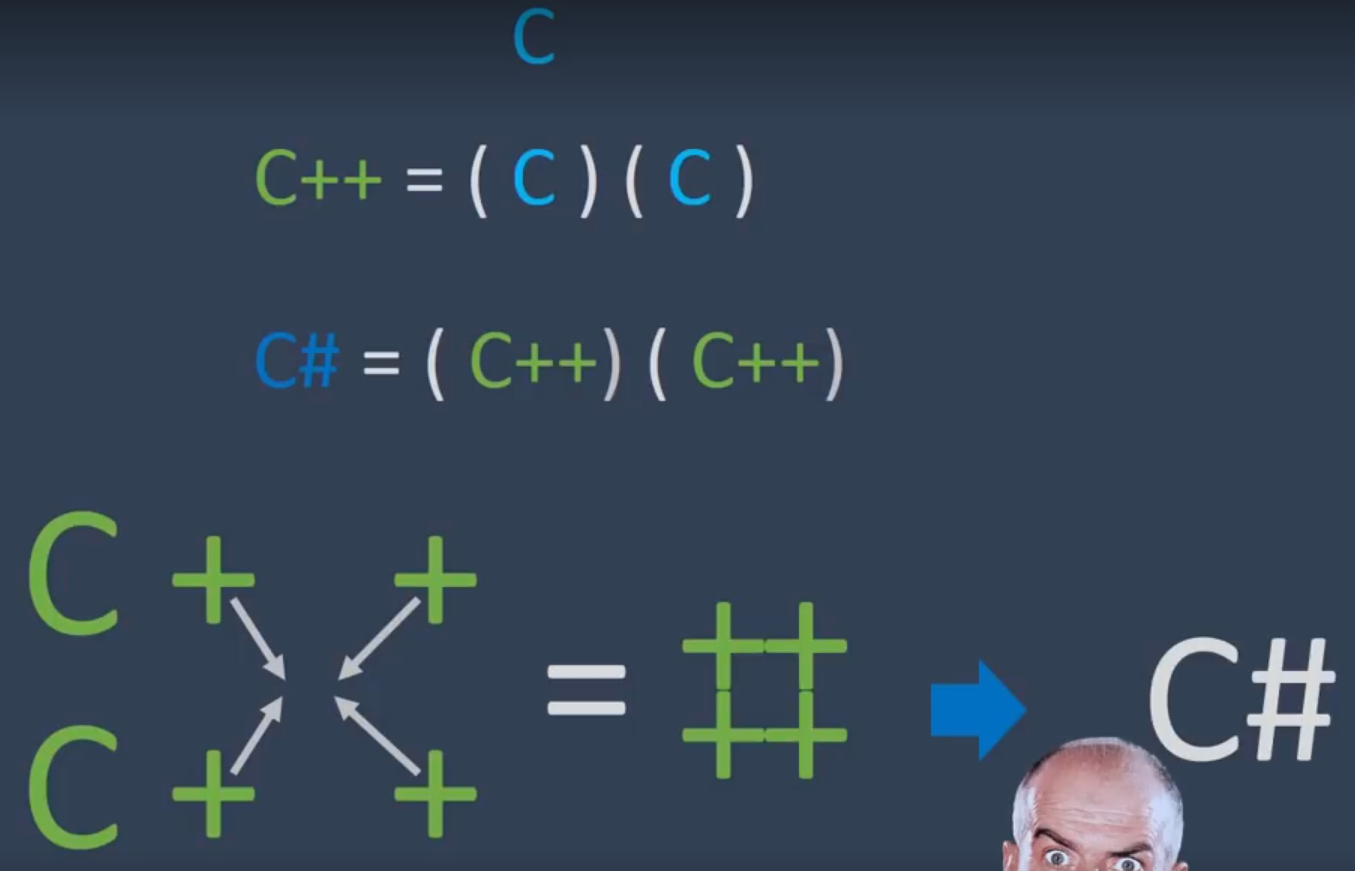
### LENGUAJES DE PROGRAMACIÓN

Microsoft.net desarrolla activamente tres lenguajes de programación: C#, F# y Visual Basic. C# es un lenguaje de programación sencillo moderno orientado a objetos y con seguridad de tipos, C# tiene sus raíces en la familia del lenguaje C y eso facilitara mucho más rápido el aprendizaje con C++, Java y JavaScript.

Estos lenguajes son una evolución del lenguaje C, esto tampoco va al tema, pero es bueno saberlo. Se podría decir que C es el papá, de ahí nacen Objetive-C y C++, luego nace Java, un lenguaje inspirado en C++, luego nace JavaScript, un lenguaje inspirado en Java y luego nace C#, un lenguaje inspirado en Java y C++. Es por esta razón que las sintaxis son similares, si se conoce uno de estos lenguajes se puede adaptar muy rápido a los demás lenguajes mencionados, también cabe decir que el lenguaje B es el abuelo de todos estos.



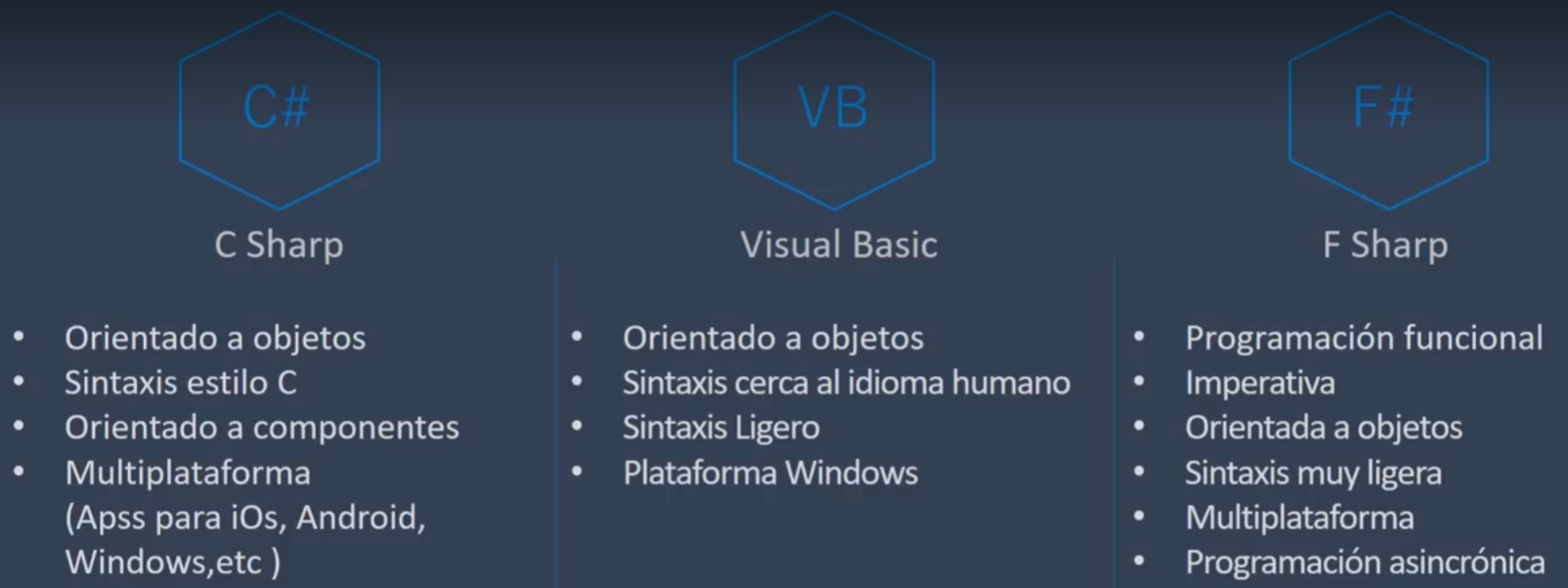
la otra historia es que C++ es igual a dos veces C, de ahí viene los dos signos más y C# es igual a dos veces C++, de igual manera si se juntan los dos signos de C++ se formaría el signo numeral o michi, algo muy curioso de verdad.



C# también incluye compatibilidad para programación orientada a componentes multiplataforma, varias características de C# permiten construir aplicaciones sólidas y duraderas gracias a que recolectan automáticamente los elementos no utilizados de la memoria. También está el control de excepciones que proporciona un enfoque estructurado y extensible para la detección de errores y la recuperación.

Visual Basic está diseñado para crear de manera productiva aplicaciones con seguridad de tipos orientadas a objetos, además es un lenguaje fácil de aprender ya que usa una sintaxis más cercana al idioma normal, lo que a menudo facilita el trabajo a las personas sin experiencia en desarrollo de software, además ofrece una manera rápida y sencilla de crear aplicaciones basadas en .NET Framework. Visual Basic solo es para la plataforma Windows, pero si se puede integrar clases portables a proyectos XAMARIN.

F# es un lenguaje de programación principalmente funcional, implica definir tipos y funciones que deduce y generaliza automáticamente, esto permite que el programador se enfoque en el dominio del problema y pueda manipular los datos en lugar de prestarle atención a los detalles de la programación. También admite la programación tradicional imperativa y orientada a objetos, resulta muy fácil escribir código correcto y fácil de mantener. C# también es multiplataforma, la programación asincrónica de F# es superior a la de C# y Visual Basic.



### DOCUMENTACIÓN

Es recomendable leer la documentación que nos ofrece Microsoft, proporciona toda la información y algunos tutoriales que sirven para entender mejor y tener conceptos más sólidos sobre .NET.

* <https://learn.microsoft.com/en-us/>

En este enlace se puede ver toda la documentación de Microsoft, pero por el momento lo más importante es .NET, aquí están todos los componentes, está la guía para nuevos desarrolladores en C# y para los que ya tienen experiencia. También es recomendable estar al tanto de las novedades y así estar al día con los cambios del lenguaje o nuevas funciones de la última versión de C#.

## INTEROPERABILIAD DE LENGUAJES DE PROGRAMACIÓN

Se puede crear software más robusto, eficiente y rápido gracias a que estos tres lenguajes son interoperables entre sí, pues son compatibles y puede coexistir en una sola aplicación, por ejemplo, se puede crear una aplicación usando C# y F# aplicando dos paradigmas programación, orientada a objetos y programación funcional. Entonces se puede aprovechar al máximo las ventajas de cada uno, de hecho, ahora los desarrolladores pueden crear la interfaz gráfica y acceso a datos en C# y en F# se puede realizar la lógica de las operaciones complejas y la programación asincrónica.

La programación asincrónica en F# es superior, tiene más ventajas sobre C# y Visual Basic, entonces para ser más grandes en desarrollo de software, porque no aplicar todos los paradigmas que admiten estos lenguajes, crear una aplicación orientada a objetos, funcional, imperativa y con componentes.

A picture containing chart

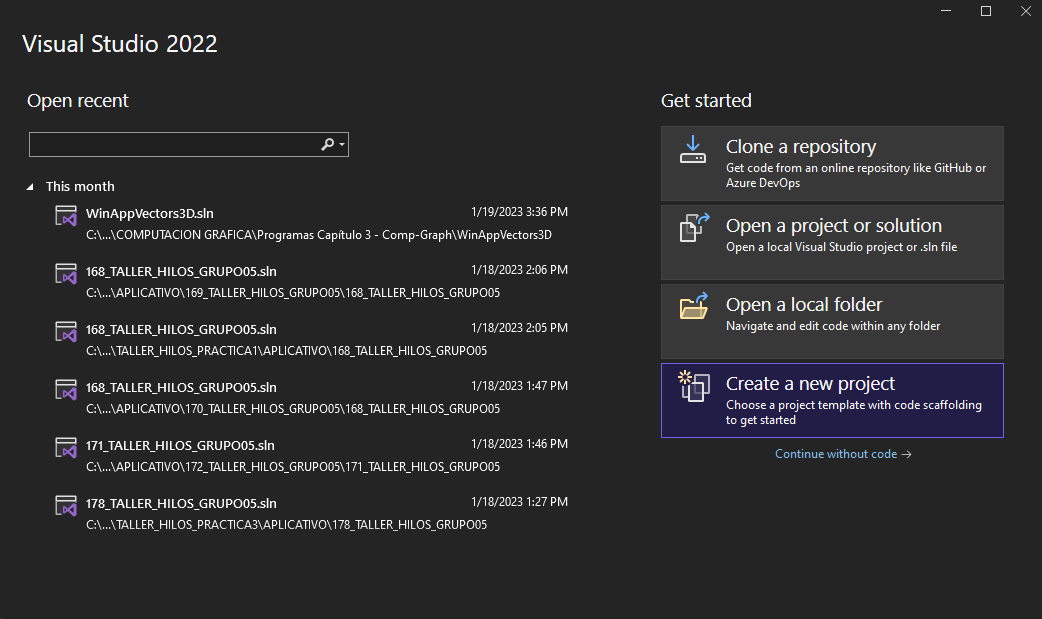
Description automatically generated

# PARTE PRÁCTICA

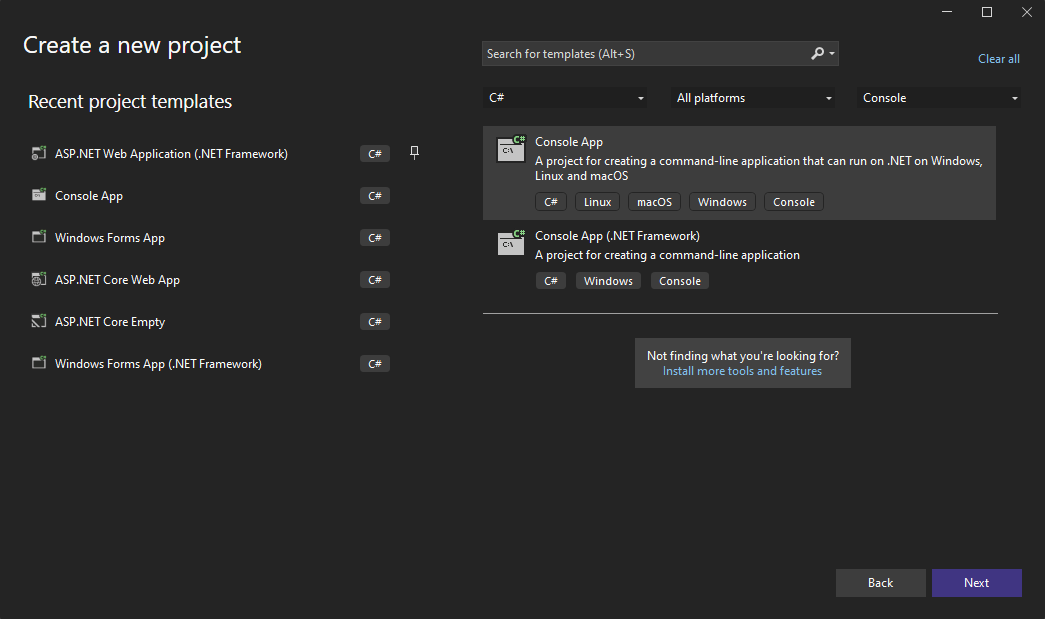
## PRÁCTICA PARA COMPARAR LA SINTAXIS DE LENGUAJES DE PROGRAMACIÓN

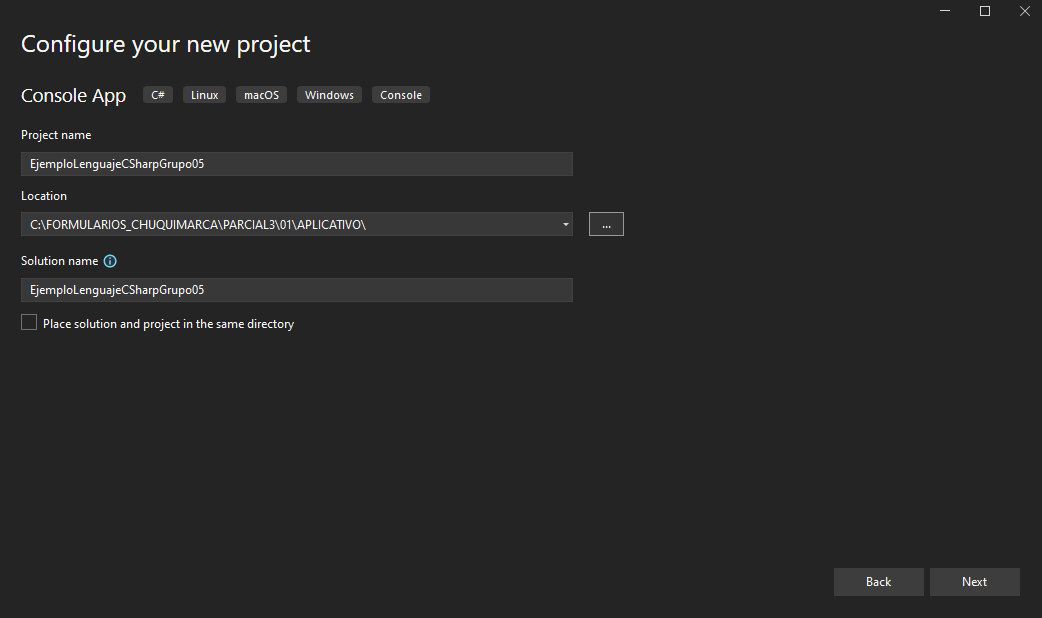
Para esta parte de la práctica, se compara la sintaxis de los lenguajes de programación C#, F#, Visual Basic, Java y C++ con un ejemplo corto y simple.

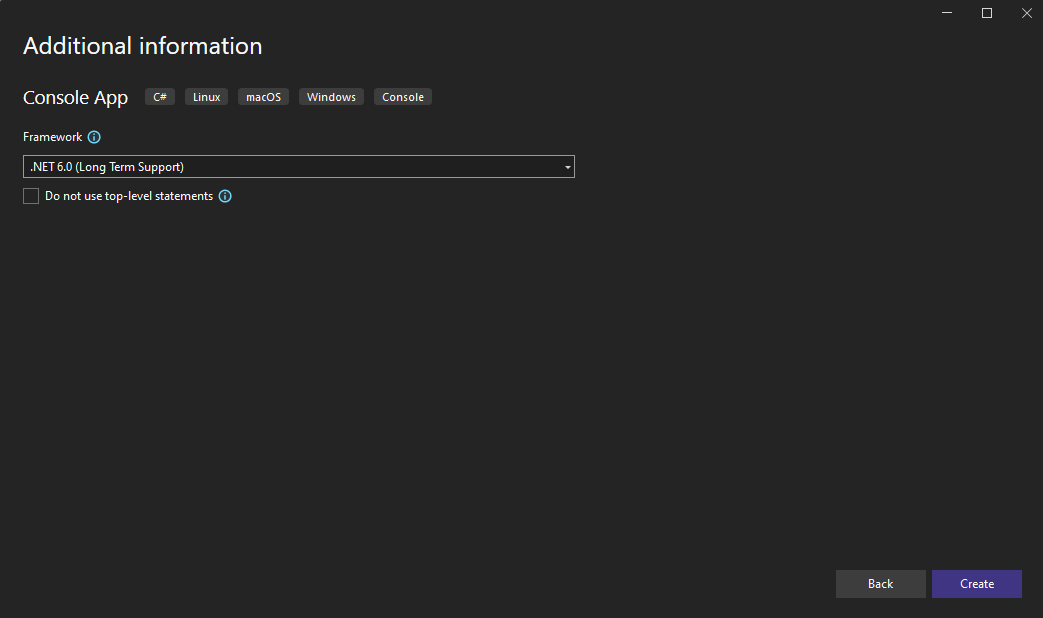
1. Primero, se crea un proyecto en Visual Studio, para esto abrir el IDE y clic en “Create a new project”.



1. En la siguiente ventana, seleccionar el tipo de proyecto Console App y clic en Next.

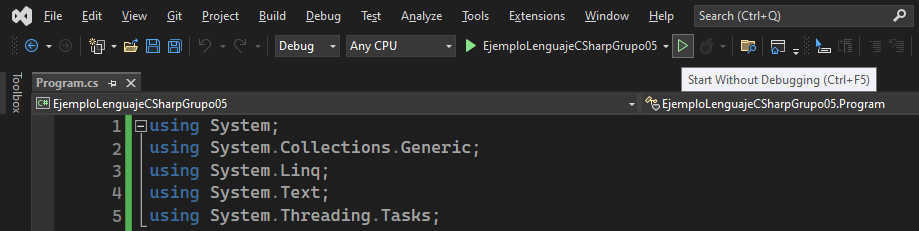






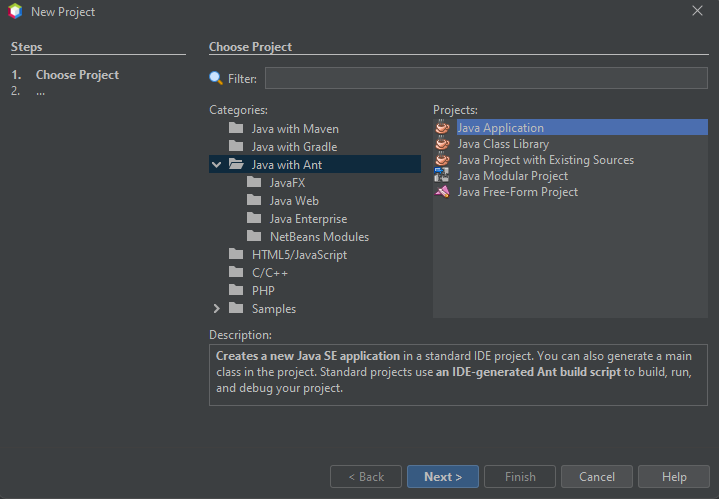
Text

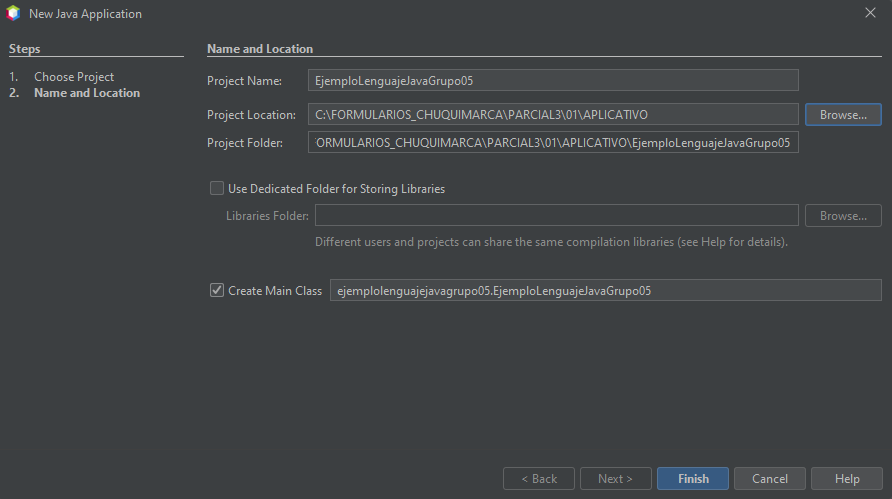
Description automatically generated



Graphical user interface, website

Description automatically generated





Text

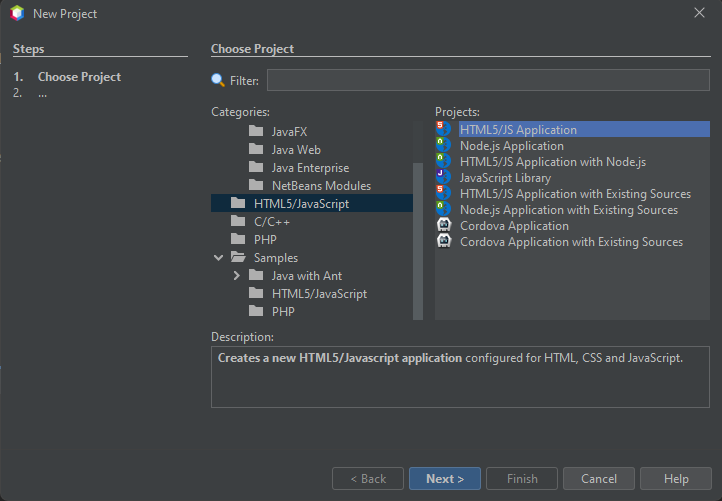
Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated



Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, chat or text message

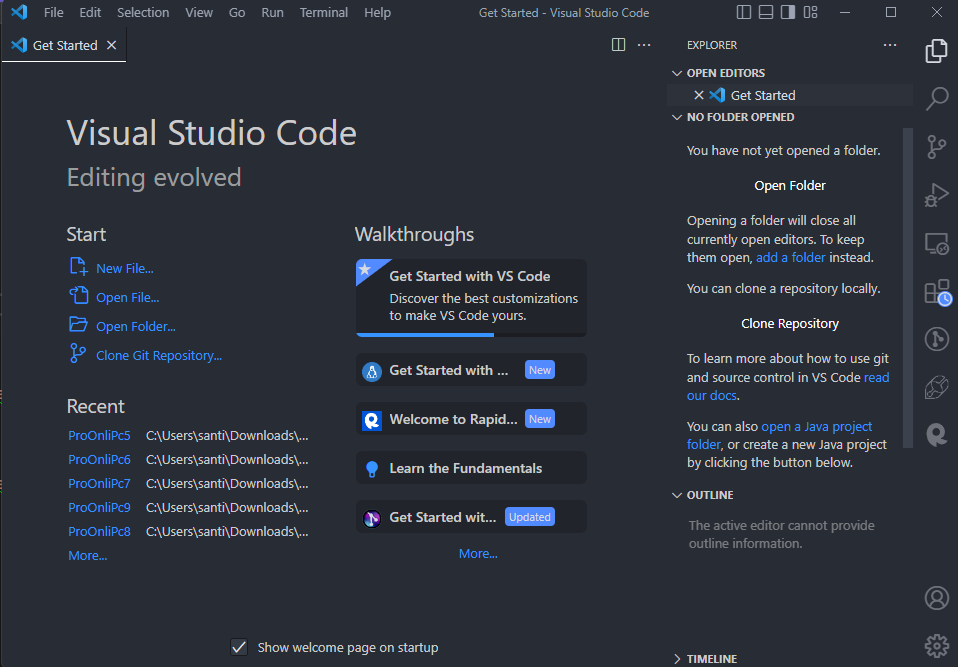
Description automatically generated

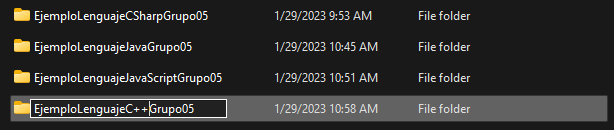
Graphical user interface, text, application, website

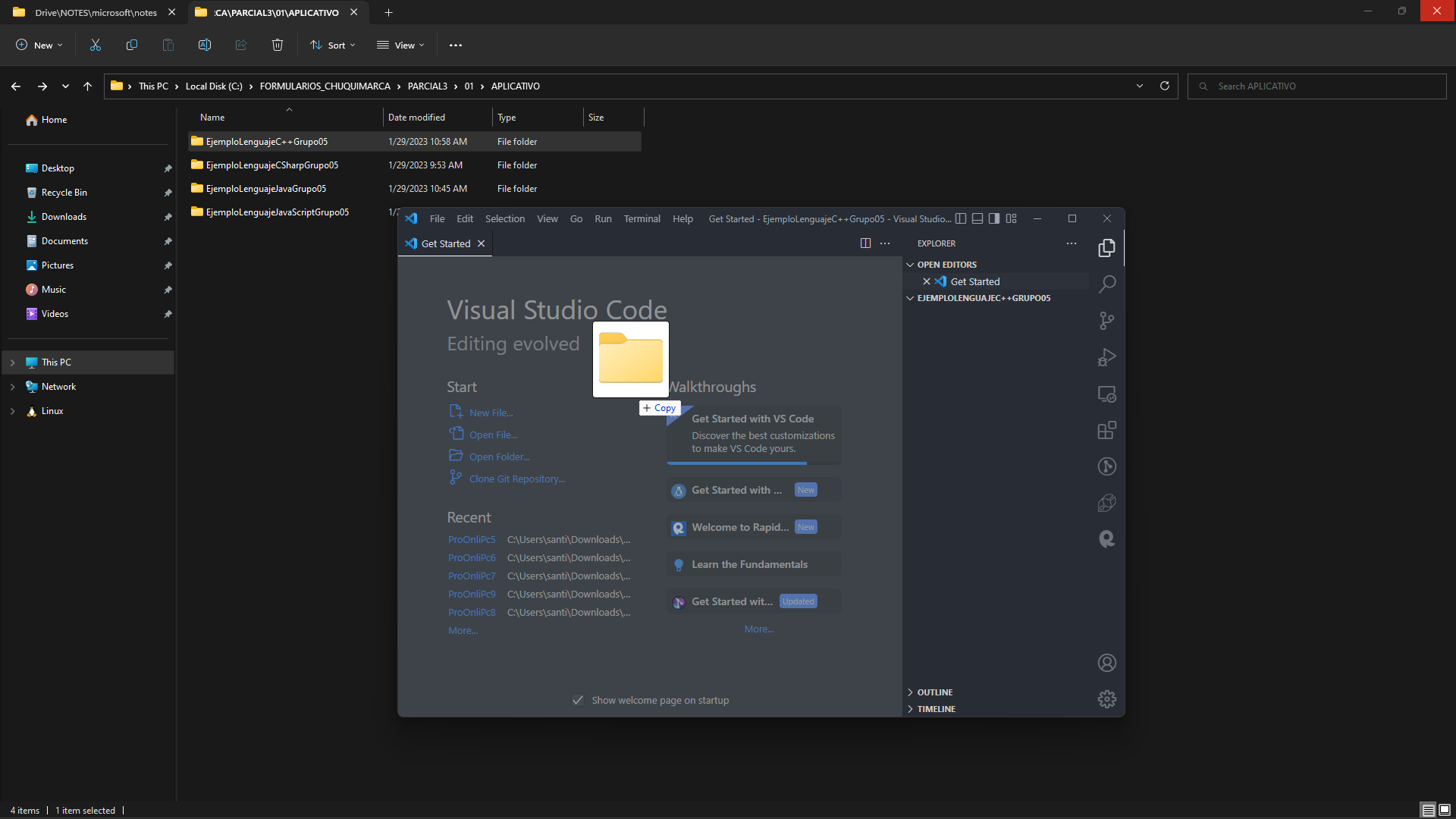
Description automatically generated

Graphical user interface, text, application

Description automatically generated

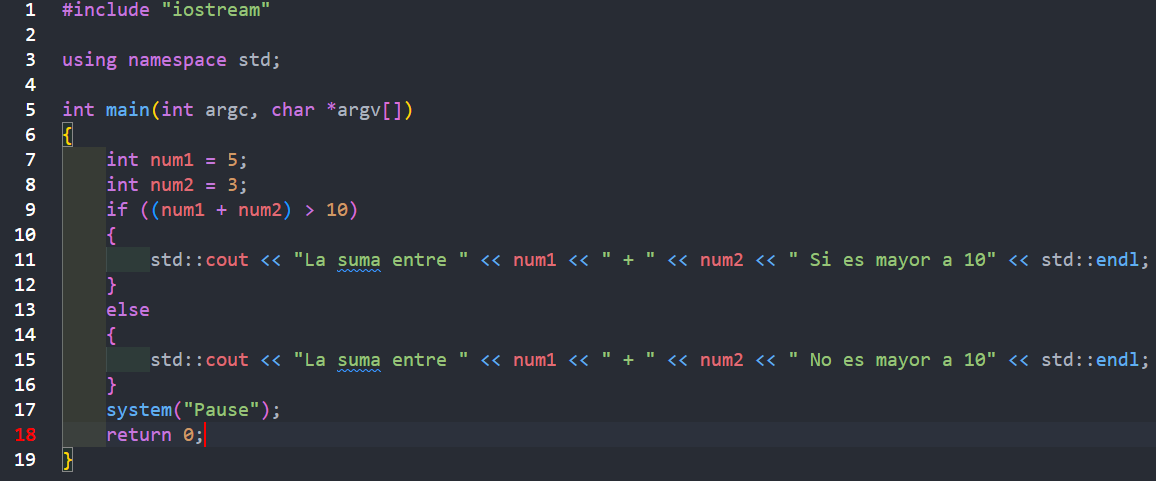






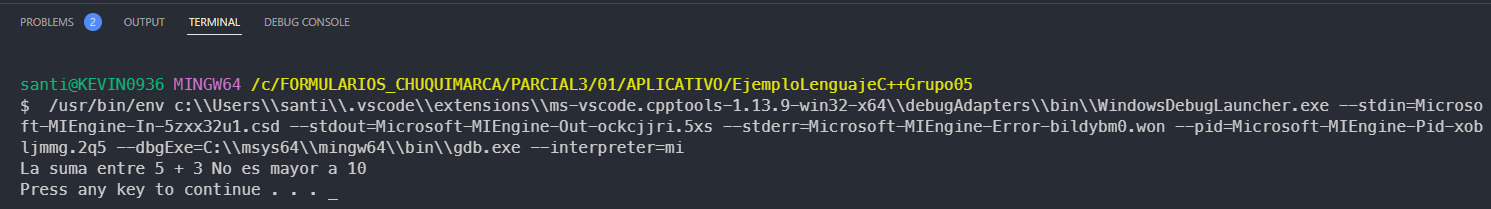
Graphical user interface, text

Description automatically generated



Graphical user interface, application

Description automatically generated



Practica de código con VB

Graphical user interface, text, application

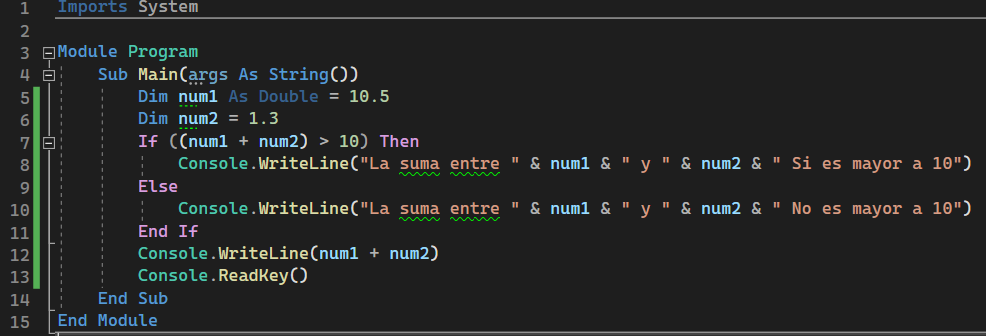
Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated



Graphical user interface, text, application

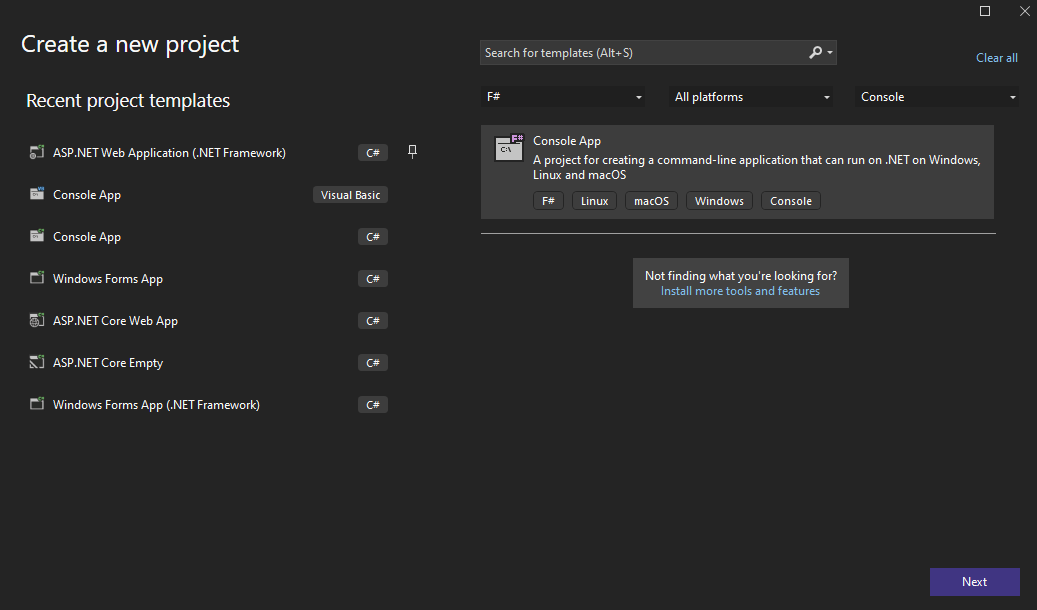
Description automatically generated

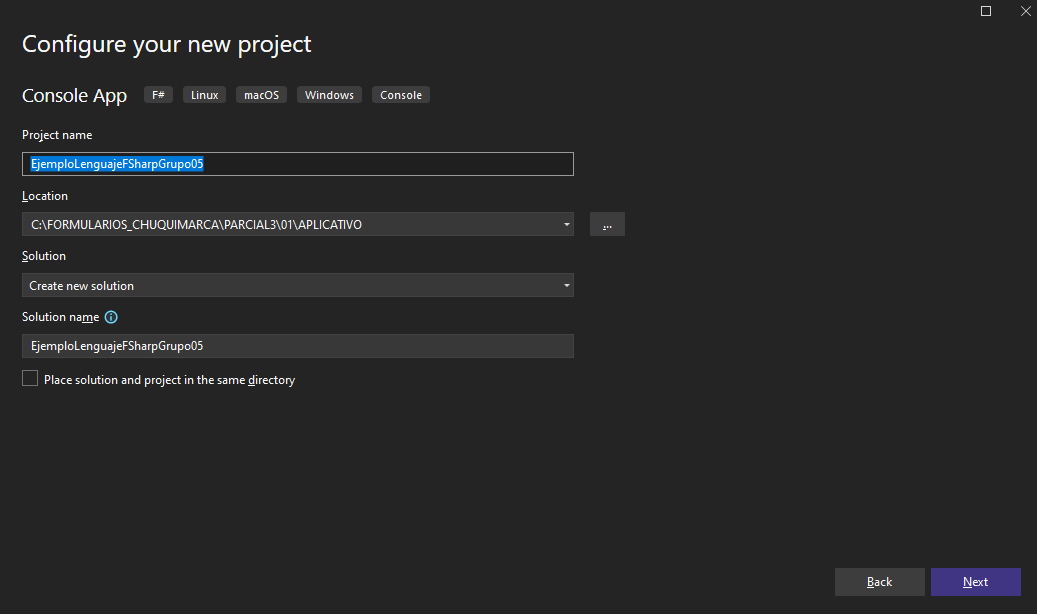
Graphical user interface, website

Description automatically generated

Tipo de dato la suma fue satisfactorio, pero sí hay ocasiones que debemos especificar el tipo de dato

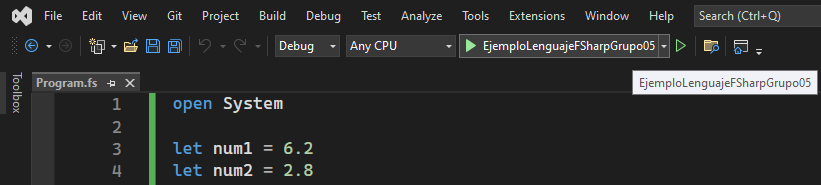
Practica con F#





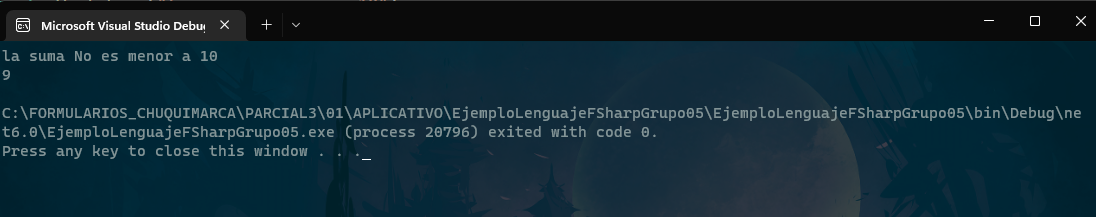
Graphical user interface, text, application

Description automatically generated

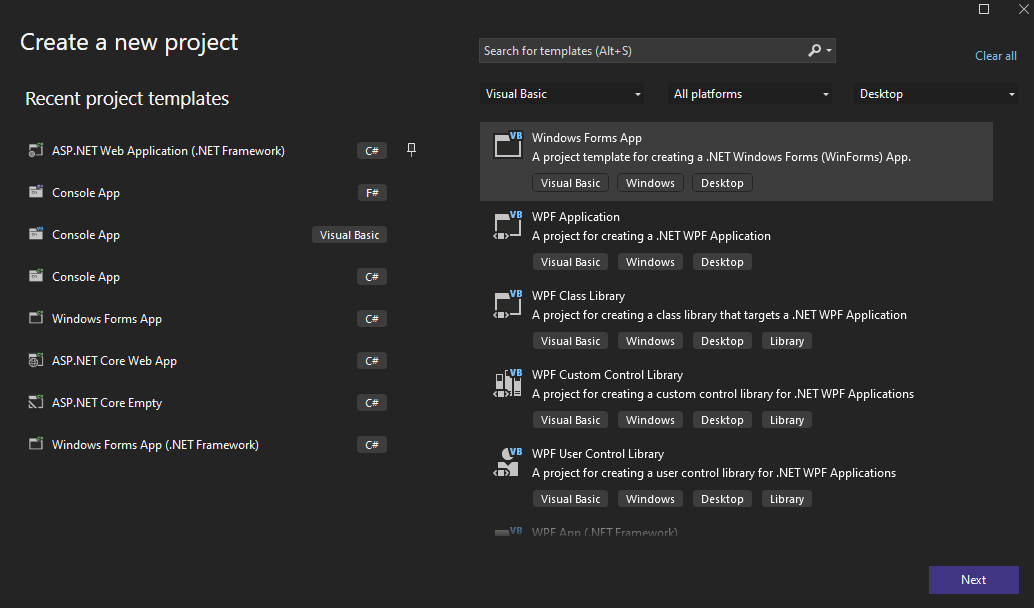


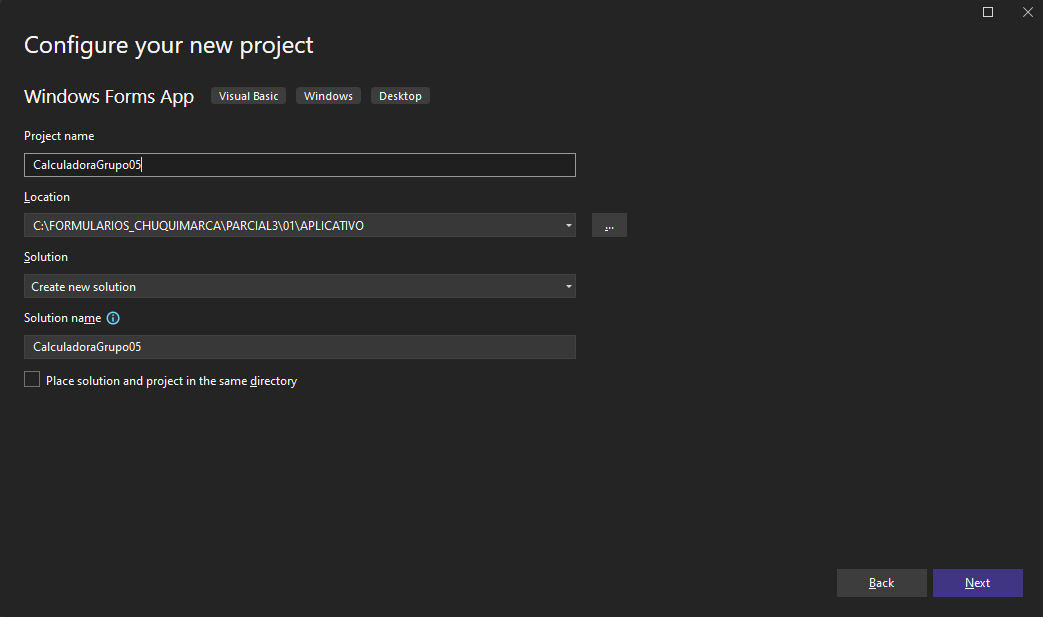
Text

Description automatically generated



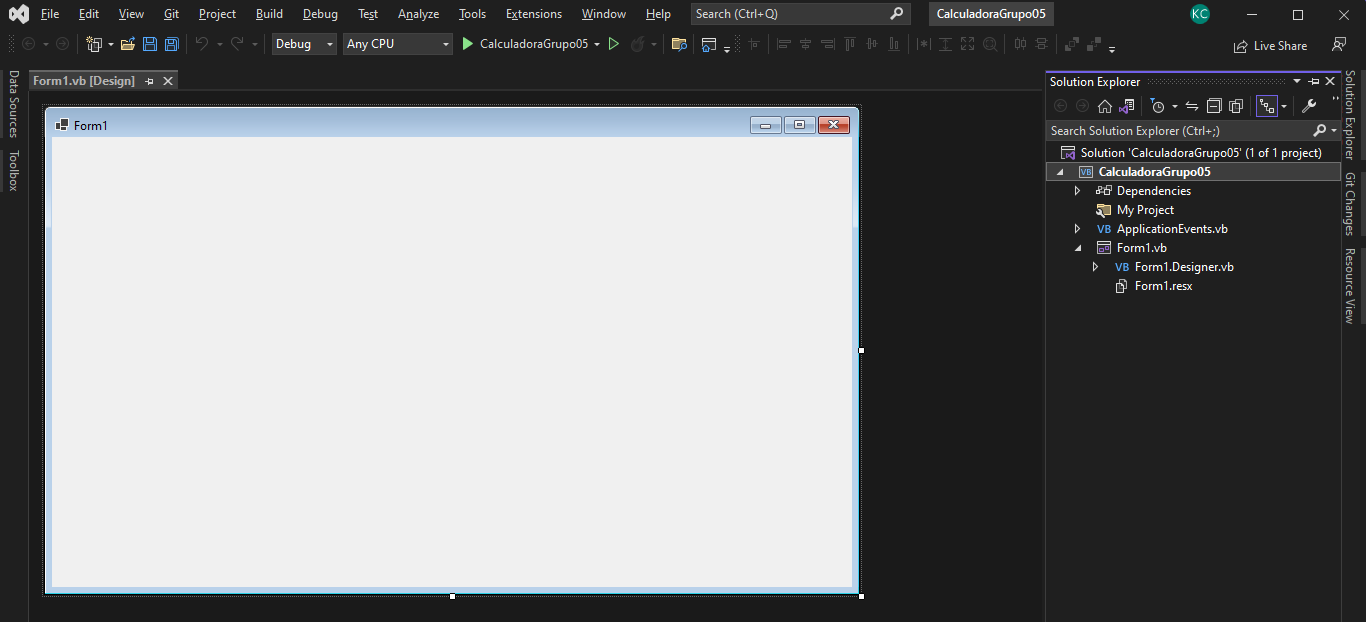
Video 3 solo practica (Interoperable de lenguajes .NET)



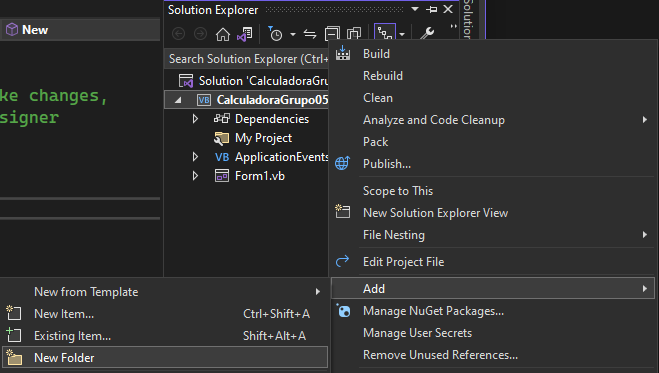


Graphical user interface, text, application

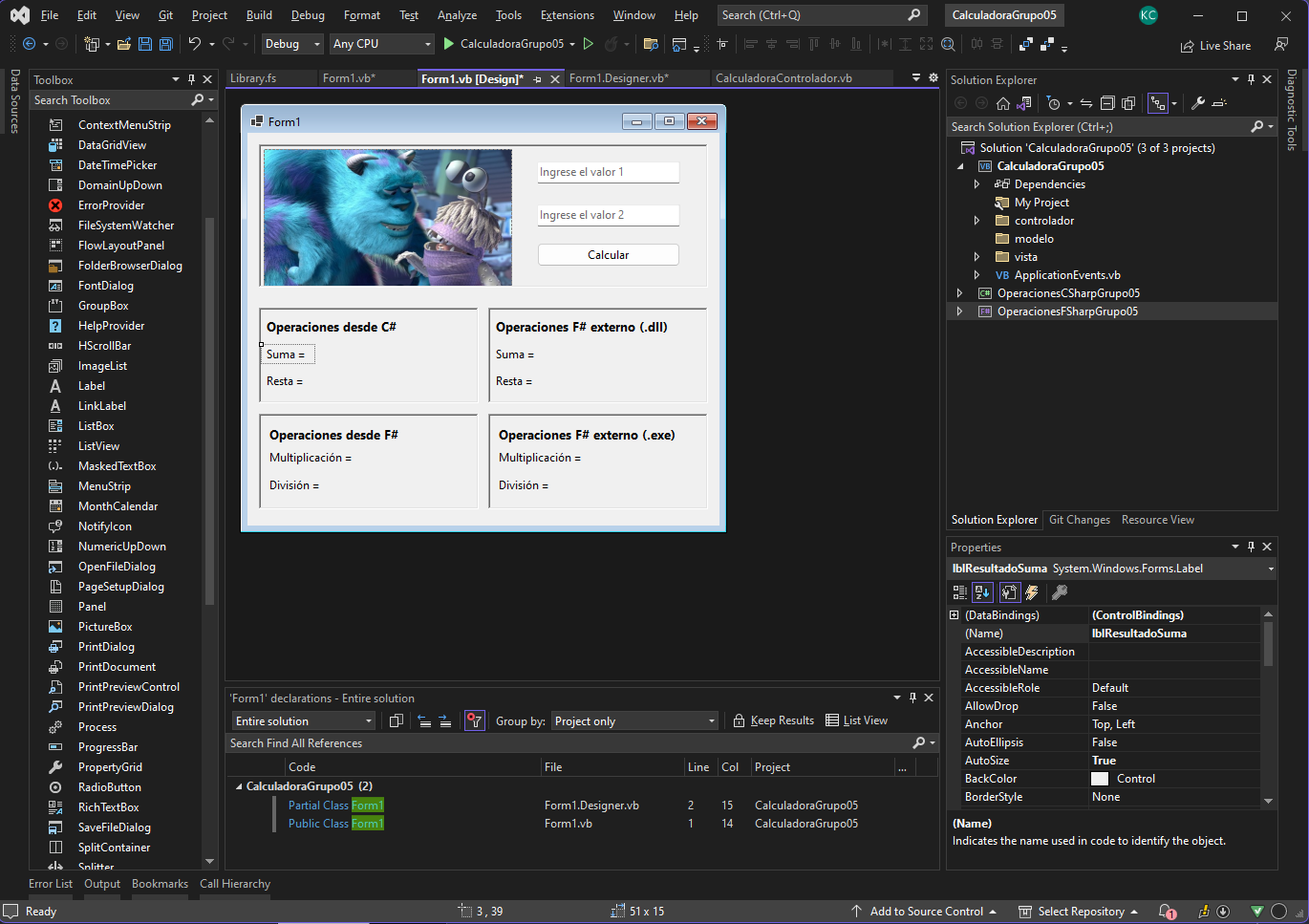
Description automatically generated

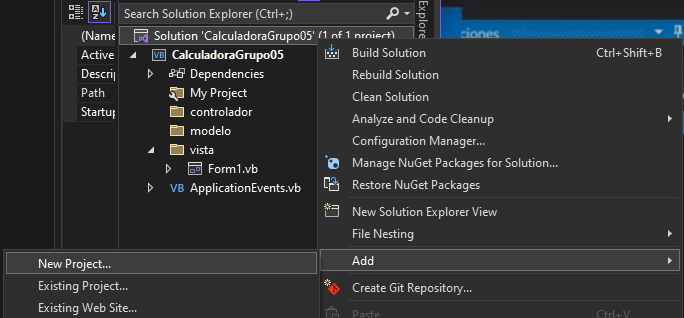


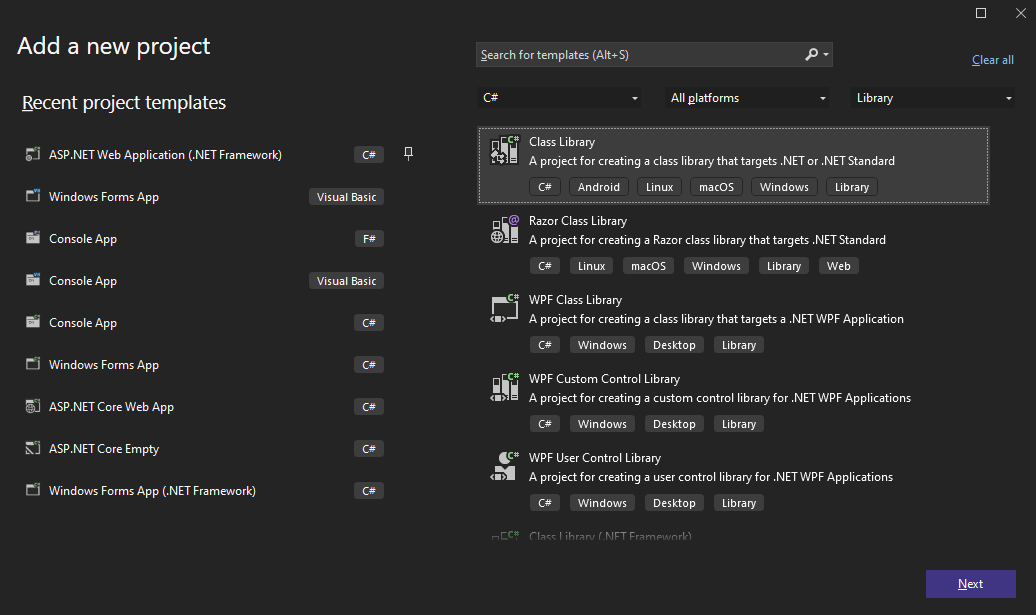
Arquitectura MVC











Graphical user interface, text

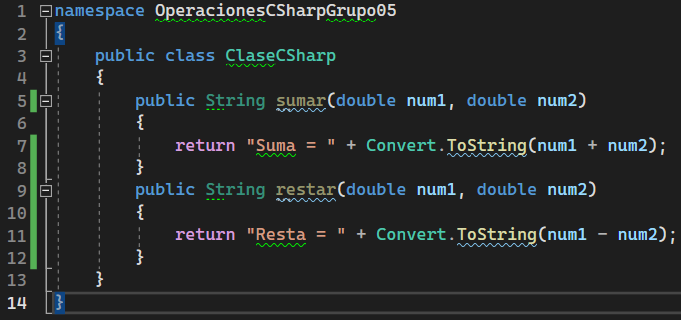
Description automatically generated

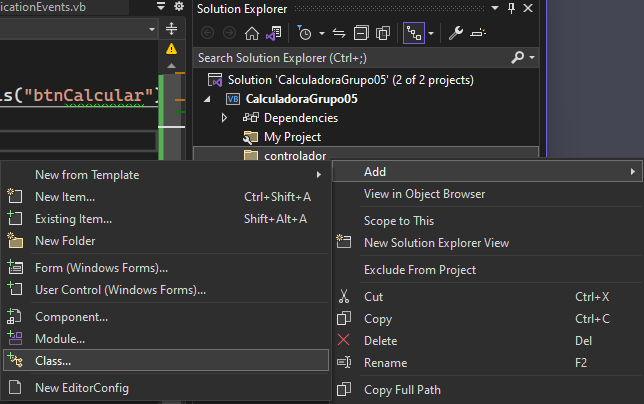
Graphical user interface, text, application

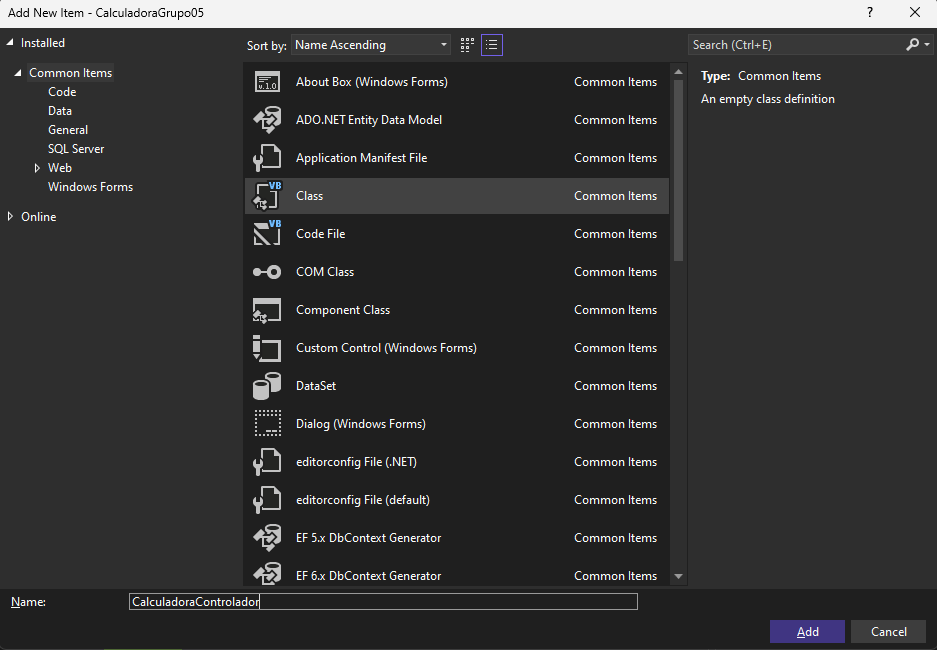
Description automatically generated

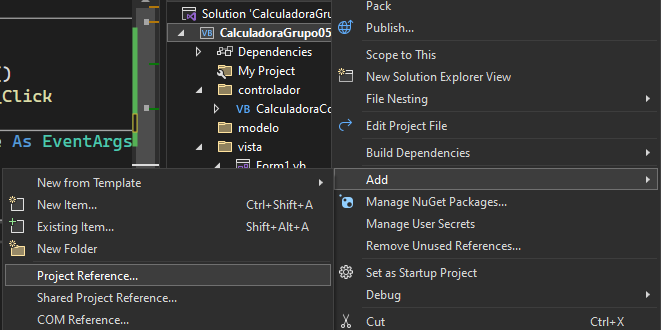
Graphical user interface, text

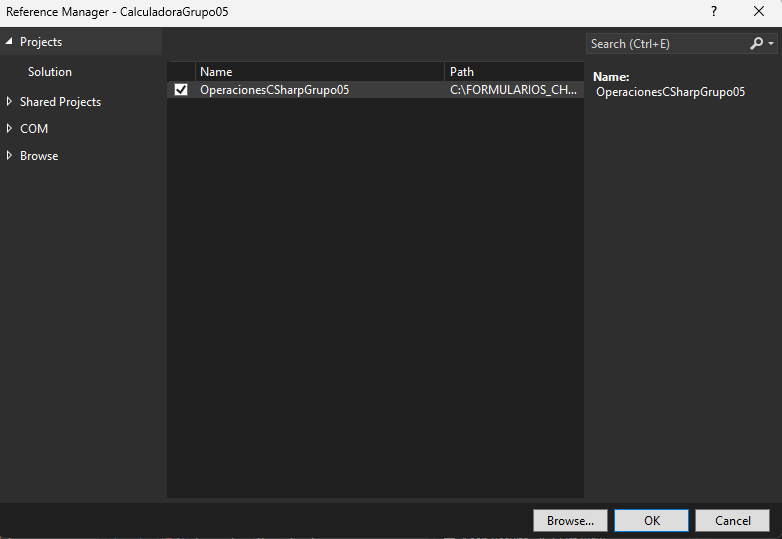
Description automatically generated

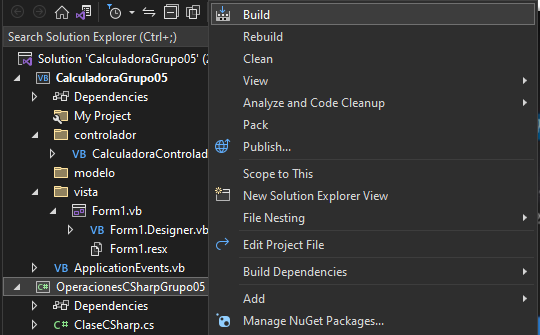


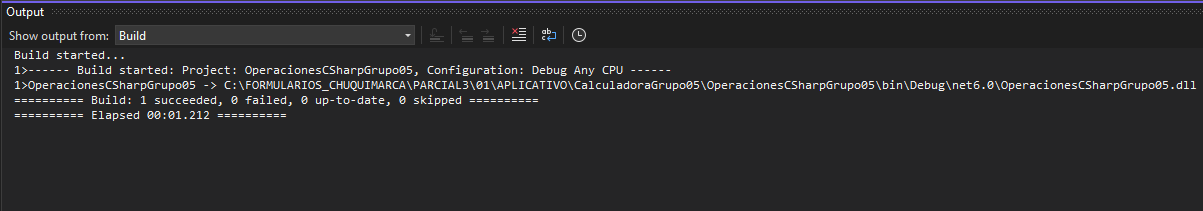


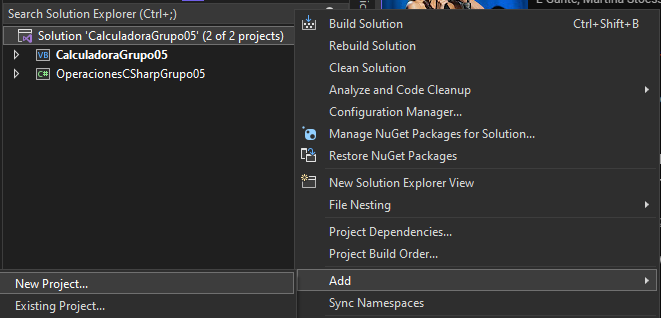












A screenshot of a computer

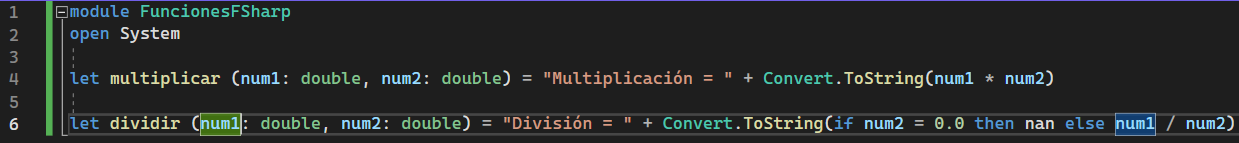
Description automatically generated with medium confidence

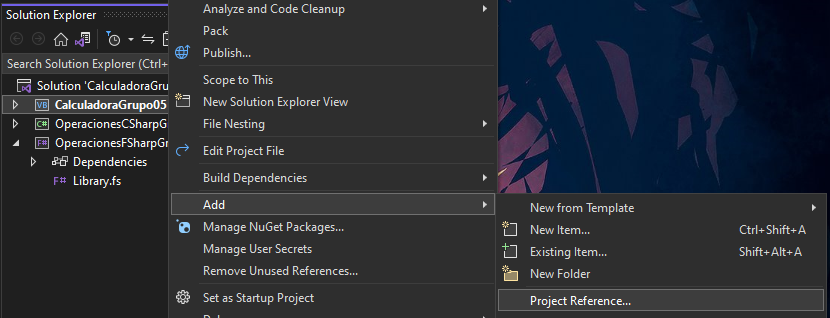
Graphical user interface, text, application, Teams

Description automatically generated

Graphical user interface, text, application

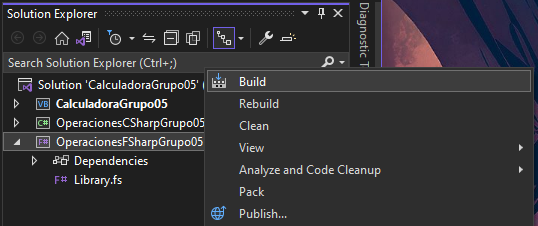
Description automatically generated



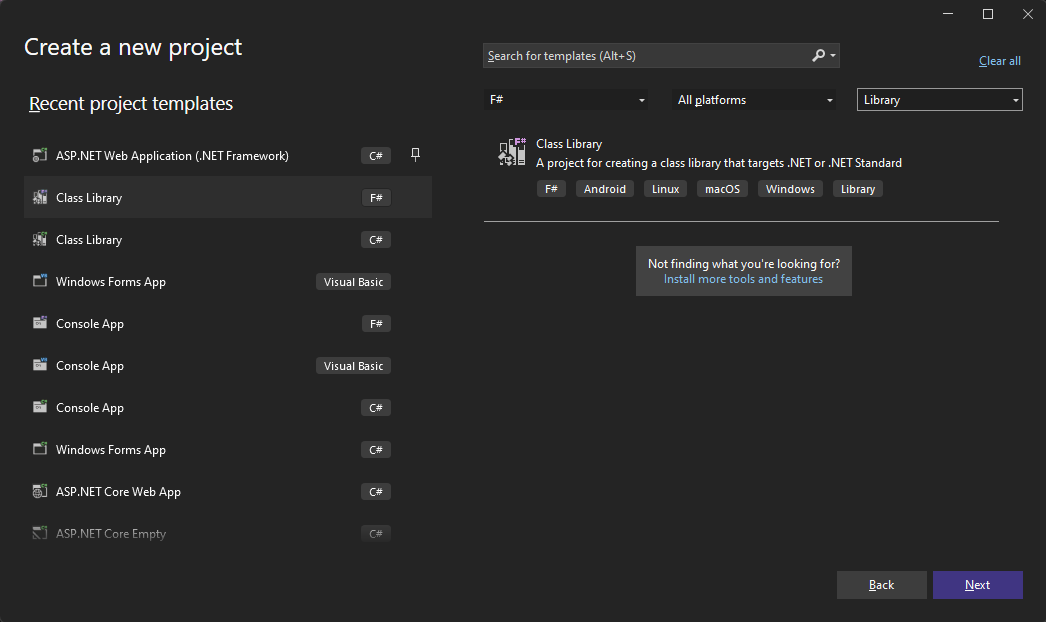


Graphical user interface, text

Description automatically generated



Agregar referencia de proyectos F# externos



Graphical user interface, text, application, Teams

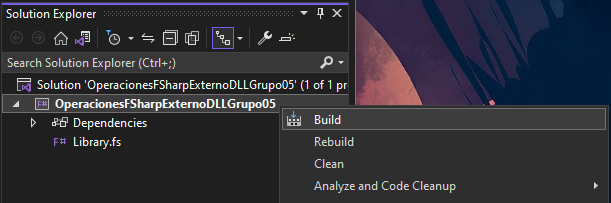
Description automatically generated

Graphical user interface, text, application

Description automatically generated

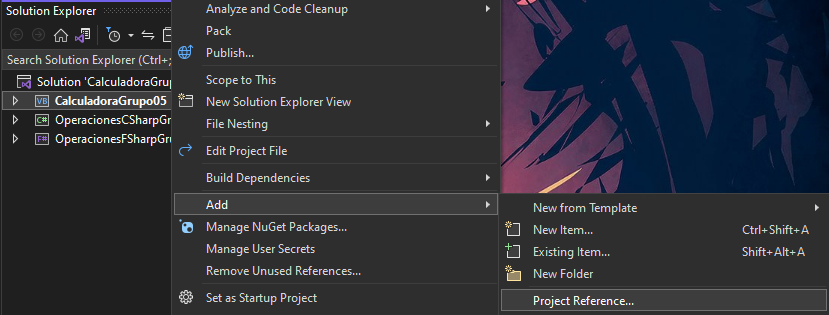
A screenshot of a computer

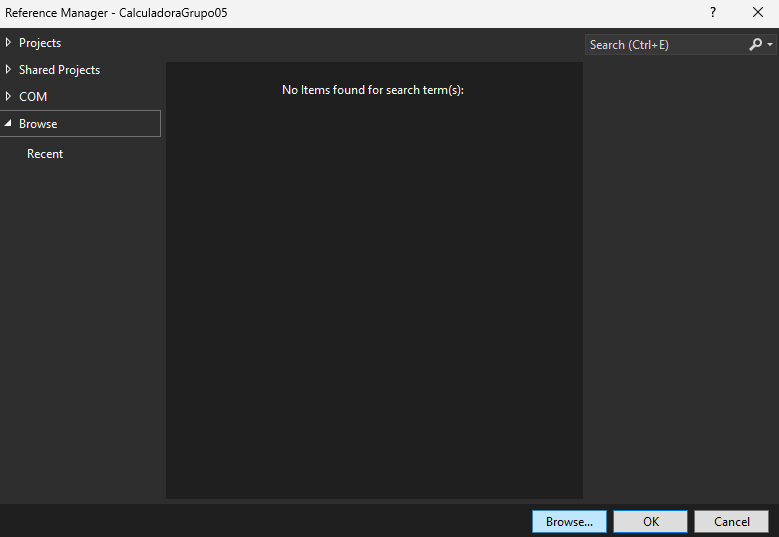
Description automatically generated with medium confidence

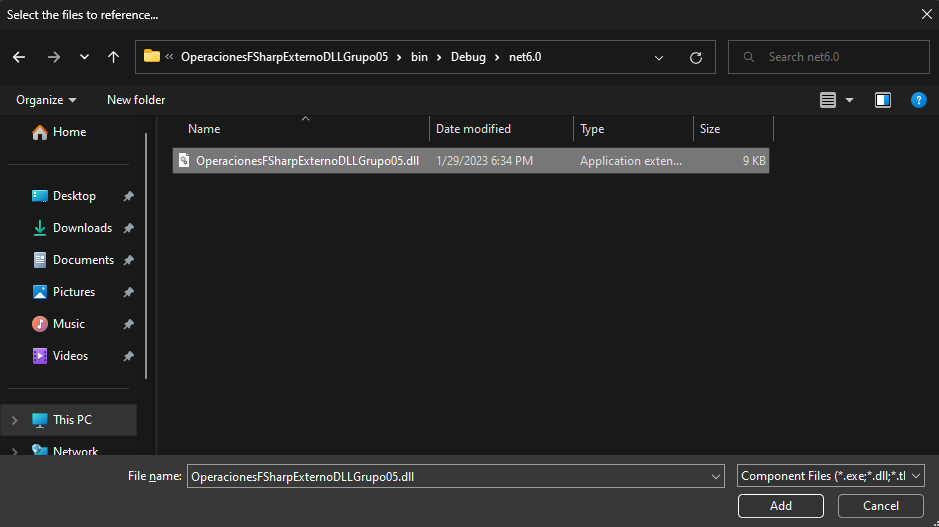


Graphical user interface, text

Description automatically generated

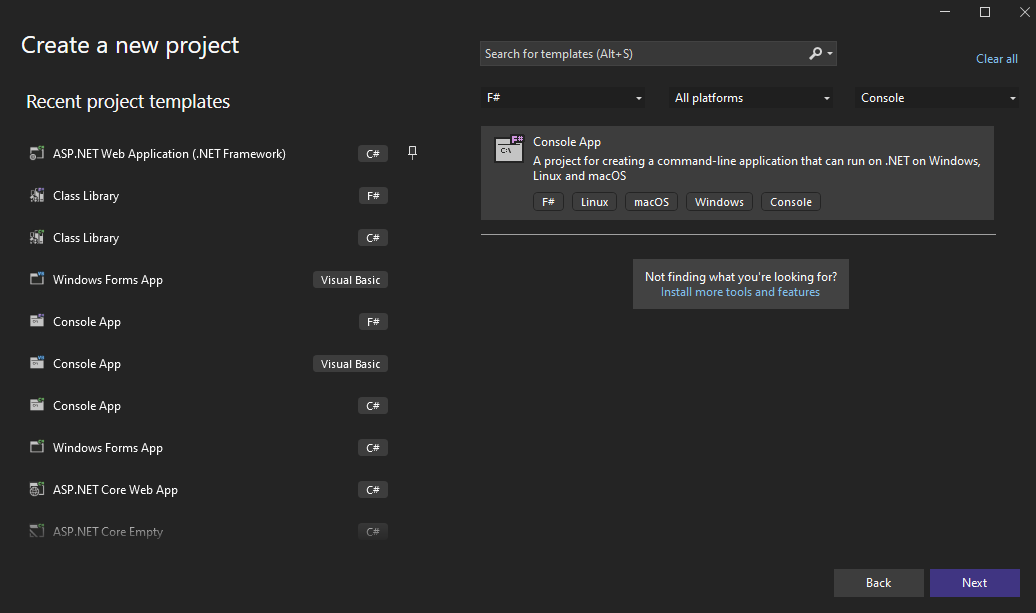






Text

Description automatically generated



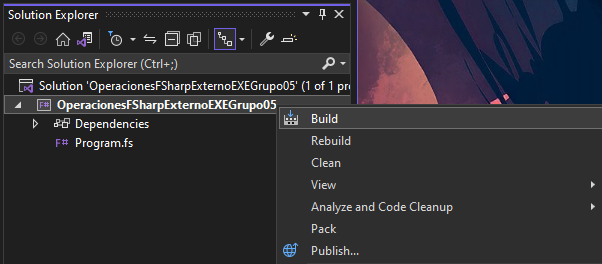
Graphical user interface, text, application, Teams

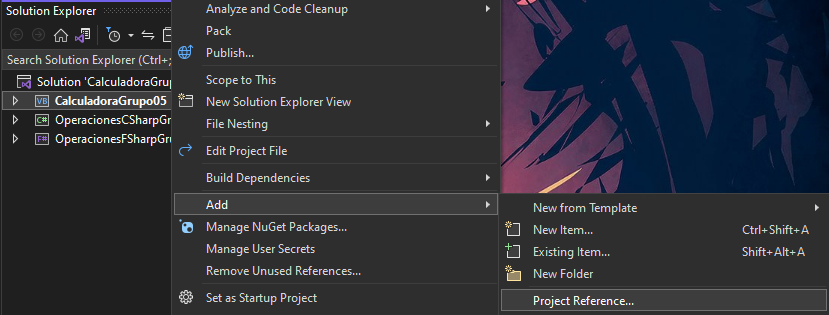
Description automatically generated

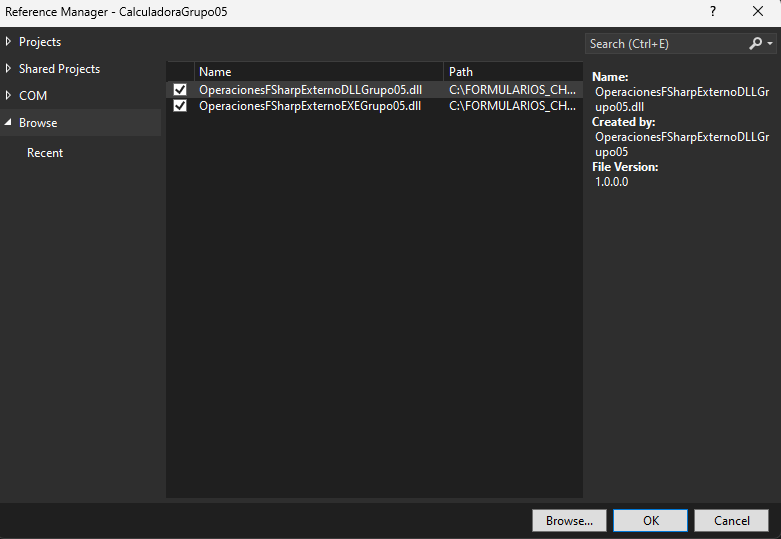
Graphical user interface, text, application

Description automatically generated









Codificación vista

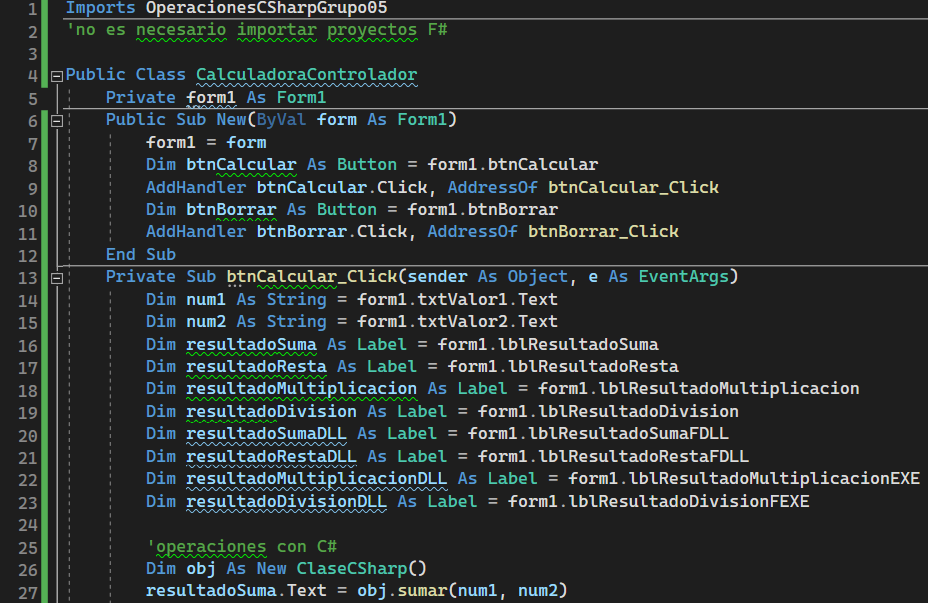
Text

Description automatically generated

Graphical user interface, website

Description automatically generated

Clase controlador



# EJECUCIÓN DEL PROYECTO

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

# ESTRUCTURA DE LA APLICACIÓN

Text

Description automatically generated

# CONCLUSIONES

* El servidor de aplicaciones GlassFish dentro del desarrollo en Java, ha sido reemplazado por Payara, ofreciendo mayor robustez, soporte y actualizaciones, permitiendo así desarrollar y testear aplicativos eficientemente.
* Java EE en sus versiones más recientes busca simplificar el desarrollo en J2EE que tradicionalmente ha sido muy complejo.

# RECOMENDACIONES

* Investigar sobre las características y compatibilidad de cada herramienta en sus nuevas versiones, en Apache NetBeans algunos procesos pueden llegar a cambiar respecto a versiones anteriores de Java EE.
* Para empezar con el desarrollo de aplicaciones utilizando Java EE se debe tener una introducción a las principales herramientas como son los: servidores de aplicaciones, EJB, Servlets, JPA, JPF, entre otros, los cuales son fundamentales para el desarrollo de aplicaciones web empresariales.

# **BIBLIOGRAFÍA**

**There are no sources in the current document.**