UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

VB.Net

Modulo 6

unidos por un cambio transformador

VIVE LA FIME

Objetivo

- En este curso aprenderemos a crear diferentes tipos de aplicaciones utilizando Visual Studio y el lenguaje de Programacion Visual Basic.
- Veremos a detalle los componentes que utilizaremos para crear las aplicaciones.
- También veremos una arquitectura de desarrollo que nos permita crear aplicaciones de manera mas eficiente.

Historia de Visual Basic

- En 1964, John Kemeny y Thomas Kurtz se propusieron crear una variante del lenguaje de programación denominado BASIC (Beginners All-Purpose Symbolic Instruction Code, Código de instrucciones simbólicas para principiantes orientado a todo propósito). Si bien ya había en el mercado varias versiones de BASIC, la generada en Dartmouth College por Kemeny y Kurtz fue la que más se popularizó.
- BASIC llegó para reducir de una manera notable los tiempos de aprendizaje y de escritura de un programa para computadoras. Así, gracias a la visión de futuro de sus propulsores, muchos estudiantes pudieron desarrollar aplicaciones en tiempo récord, algo muy valioso para esa época.

Historia de Visual Basic

- Al momento de diseñar el lenguaje, sus creadores tomaron como filosofía ocho principios que debían destacar a BASIC por sobre los demás:
- 1. Ser fácil de usar.
- 2. Ser un lenguaje de propósito general.
- 3. Permitir la incorporación de características avanzadas por expertos, priorizando su facilidad para principiantes.
- 4. Gozar de interactividad.
- 5. Ofrecer claros mensajes de error.
- 6. Brindar rápida respuesta en programas pequeños.
- 7. No requerir que los usuarios tengan conocimientos sobre hardware.
- 8. Alejar al usuario de la complejidad del sistema operativo.

Historia de Visual Basic

- Hacia 1975, la empresa fundada por **Bill Gates** y **Paul Allen** lanzó su propia versión de BASIC denominada **Altair BASIC**.
- Para fines de la década del 70, apareció la primera adaptada a la plataforma Apple II.
- En 1979, Microsoft negoció vender su licencia de BASIC a varias compañías incluida IBM.
- Años más tarde, Microsoft siguió distribuyendo una versión reducida de BASIC junto a su popular sistema operativo **MS DOS.**
- Luego la firma lanzó al mercado **Visual Basic 1.0** un entorno de desarrollo que facilitaba la creación de aplicaciones con menús, ventanas y botones, pero que aún corría bajo DOS.
- En la version 2.0, desarrollada para **Windows 3.0/3.1**, Microsoft dejó de lado el entorno gráfico construido mediante caracteres **ASCII**, para dar inicio a una era distinta: la era **RAD** de desarrollo de aplicaciones para Windows.

Historia de Visual Basic

- Desde **Visual Basic 3.0**, Microsoft fortaleció el desarrollo de aplicaciones RAD orientado a bases de datos y dio un gran soporte al lenguaje para conectarse a cualquier base entre las más populares del mercado **(Dbase, Paradox, Fox Pro**)
- La versión 4.0 llegó al mercado casi al mismo tiempo que **Windows 95**, con lo cual se lanzó una edición doble, para 16 y 32 bits
- **Visual Basic 5.0** contó con una versión lite denominada **CCE** (*Control Creation Edition*), en la que no solo era posible crear librerías **DLL** y archivos ejecutables, sino que también se habilitaba a los programadores a generar controles personalizados, combinando dos o más controles **ActiveX** existentes.
- En 1998 Microsoft introdujo **Visual Basic 6.0**

Historia de Visual Basic

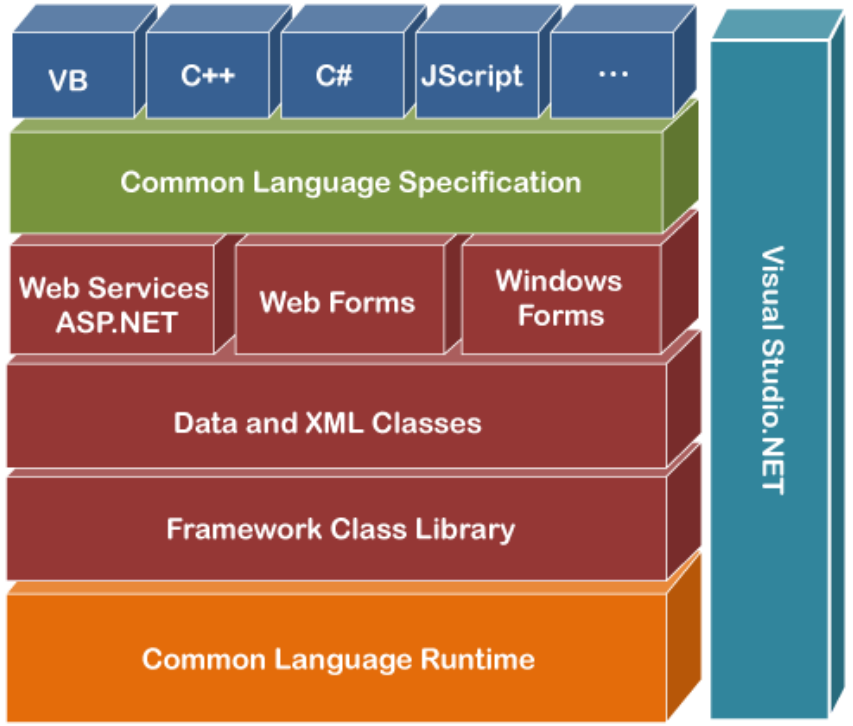
- Microsoft Visual Studio es una plataforma para desarrollo de software integrada por varios lenguajes de programación:
- **Visual C++**,
- **Visual C#**,
- **ASP.NET**
- **Visual Basic.NET**

Historia de Visual Basic

- La plataforma .NET de Microsoft es un componente de software que puede ser añadido al sistema operativo Windows. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de los programas escritos específicamente con la plataforma. Esta solución es el producto principal en la oferta de Microsoft, y pretende ser utilizada por la mayoría de las aplicaciones creadas para la plataforma Windows.

Historia de Visual Basic

Número de versión	Versión de CLR	Lanzamiento	Incluido en Windows
1.0 SP2	1.0	07-08-2002	XP SP1
1.1 SP1	1.1	30-08-2004	XP SP2 y SP3
2.0	2.0	27-10-2005	N/A
3.0	2.0	06-11-2006	Vista
3.5 SP1	2.0	11-08-2008	7 SP1
4.0	4	12-04-2010	N/A
4.5	4	15-08-2012	8
4.5.1	4	17-10-2013	8.1
4.5.2	4	05-05-2014	N/A
4.6	4	20-07-2015	10 v1507
4.6.1	4	30-11-2015	10 v1511
4.6.2	4	02-08-2016	10 v1607
4.7	4	05-04-2017	10 v1703
4.7.1	4	13-10-2017	10 v1709
4.7.2	4	30-04-2018	10 v1803 y 10 v1809
4.8	4	18-04-2019	10 v1903, 10 v1909, 10 v2004, 10 v20H2 10 v21H1 y Windows 11



Programación Orientada a Objetos (POO/OOP)

- La POO es una técnica para desarrollar soluciones computacionales utilizando componentes de *software* (objetos de *software*).
- **Clase/Objeto**
- **Campo**
- **Método**
- **Modularidad**

Programación Orientada a Objetos (POO/OOP)

- **Reutilización**
    - **Polimorfismo**
    - **Genericidad**
    - **Operaciones Relacionadas**
    - **Caja negra**
    - **Sobrecarga**

Programación Orientada a Objetos (POO/OOP)

- **Abstracción**
    - **Encapsulación de información**
    - **Herencia**

Palabras Reservadas

PALABRAS RESERVADAS DEL LENGUAJE			
AddHandler	AddressOf	Alias	And
AndAlso	As	Boolean	ByRef
Byte	ByVal	Call	Case
Catch	CBool	CByte	CChar
CDate	CDec	CDbl	Char
CInt	Class	CLng	CObj
Const	Continue	CSByte	CShort
CSng	CStr	CType	CUInt
CULng	CUShort	Date	Decimal
Declare	Default	Delegate	Dim
DirectCast	Do	Double	Each
Else	ElseIf	End	EndIf
Enum	Erase	Error	Event
Exit	False	Finally	For
Friend	Function	Get	GetType
GetXMLNamespace	Global	GoSub	GoTo
Handles	If	If()	Implements
Imports (.NET)	Imports (XML)	In	Inherits
Integer	Interface	Is	IsNot
Let	Lib	Like	Long
Loop	Me	Mod	Module

PALABRAS RESERVADAS DEL LENGUAJE			
MustInherit	MustOverride	MyBase	MyClass
Namespace	Narrowing	New	Next
Not	Nothing	NotInheritable	NotOverridable
Object	Of	On	Operator
Option	Optional	Or	OrElse
Overloads	Overridable	Overrides	ParamArray
Partial	Private	Property	Protected
Public	RaiseEvent	ReadOnly	ReDim
REM	RemoveHandler	Resume	Return
SByte	Select	Set	Shadows
Shared	Short	Single	Static
Step	Stop	String	Structure
Sub	SyncLock	Then	Throw
To	True	Try	TryCast
TypeOf	Variant	Wend	UInteger
ULong	UShort	Using	When
While	Widening	With	WithEvents
WriteOnly	Xor	#Const	#Else
#ElseIf	#End	#If	=
&	&=	*	*=
/	/=	\	\=
^	^=	+	+=
-	-=	>> (Operador)	>>= (Operador)
<<	<<=		

unidos por un cambio transformador

VIVE LA FIME

Palabras No Reservadas

PALABRAS NO RESERVADAS			
Add	Ansi	Assembly	Auto
Binary	Compare	Custom	Distinct
Equals	Explicit	From	Group By
Group Join	Into	IsFalse	IsTrue
Join	Key	Mid	Off
Order By	Preserve	Skip	Skip While
Strict	Take	Take While	Text
Unicode	Until	Dónde	#ExternalSource
#Region			

Tipos de Datos

Visual Basic clasifica los tipos de datos en dos grupos principales: los **Tipos por valor** y los **Tipos por referencia**.

Tipo	Valores	Tamaño
Boolean	Representa un valor verdadero (true) o falso (false)	2 bytes
Byte	Representa un valor de 8 bits en un rango entre 0 y 255	1 byte
Char	Representa un carácter de 16 bits	2 bytes
DateTime	Representa un valor de fecha y hora	8 bytes
Decimal	Representa un valor de 28 dígitos significativos	12 bytes
Double	Representa un valor en coma flotante de 64 bits	8 bytes
Integer	Representa un valor entre un rango de +-2,147,483,698	4 Bytes
Long	Representa un valor entre un rango de +-9.223.372.036.854.775.807	8 Bytes
Short	Representa un valor entre un rango de +-32.677	2 Bytes
String	Cadena de caracteres	0 a 2 billones de caracteres

Variables/Constantes

Espacio de memoria que permite almacenar un tipo de dato

Variables Locales
Dim <nombre de variable> **as** <Tipo de Dato>

Variables Globales
Global<nombre de variable> **as** <Tipo de Dato>

Constantes
Variables a las que solamente se les puede asignar una vez el valor y permanece constante a lo largo del programa

Const <nombre de variable> = <valor constante>

Operadores

Operadores Aritméticos

Operadores de Visual Basic .NET	Operador	Expresión
Suma	+	numero1 +numero2
Resta	-	numero1 – numero2
Multiplicación	*	Numero1 * numero2
División	/	Numero1 / numero2
División entera	\	Numero1 \ numero2
Residuo	mod	numero1 mod numero2
Exponenciación	^	Numero1 ^ numero2

Operadores Relacionales

Operadores Relacionales	Operador	Expresión
Mayor que	>	variable1>variable2
Mayor o igual que	>=	variable1>=variable2
Menor que	<	variable1<variable2
Menor o igual que	<=	variable1<=variable2
Diferente	<>	variable1<>variable2
Igual a	=	variable1=variable2

Operadores Lógicos

Operadores lógicos	Operador	Expresión
Y	And	Es verdadero, si al evaluar cada uno de los operandos el resultado es verdadero, si uno de los operandos es falso el resultado será falso.
También Y	AndAlso	Es falso, si al evaluar el primer operando el resultado es falso, el segundo operando no es evaluado.
O	Or	Es falso, si al evaluar cada uno de los operandos el resultado es falso, si uno de los operandos es verdadero el resultado será verdadero.
También O	OrAlso	Es verdadero, si al evaluar el primer operando el resultado es verdadero, el segundo operando no es evaluado.
Negación	Not	El resultado de aplicar este operando es falso si al evaluar su operando el resultado es verdadero, y verdadero en caso contrario.
	Xor	Da como resultado verdadero, si al evaluar cada uno de los operando uno de ellos es verdadero y el otro falso, caso contrario es falso.

Sentencias de control


```
If (condición) Then
      Instrucción(es) a ejecutarse si la condición es verdadera
Else
      Instrucción(es) a ejecutarse si la condición es falsa
Endif
```

```
While (condición)
   Instruccion(es)
End While
Resto del programa
```

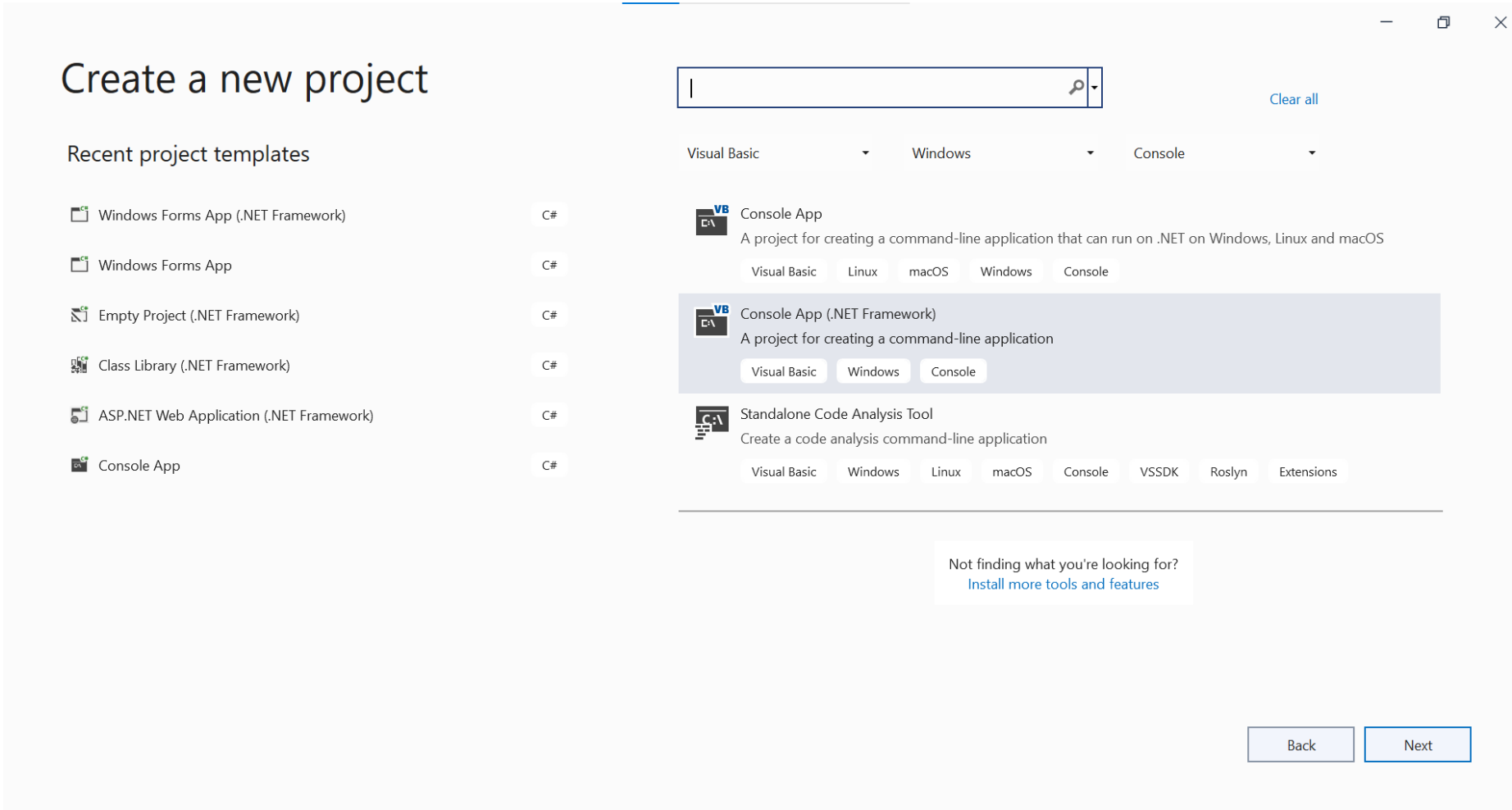
```
Do
   Instrucción(es)
Loop While (condición)
```

```
For variable =expresion1 To expresión2 Step expresion3
    Instruccion(es)
Next
```

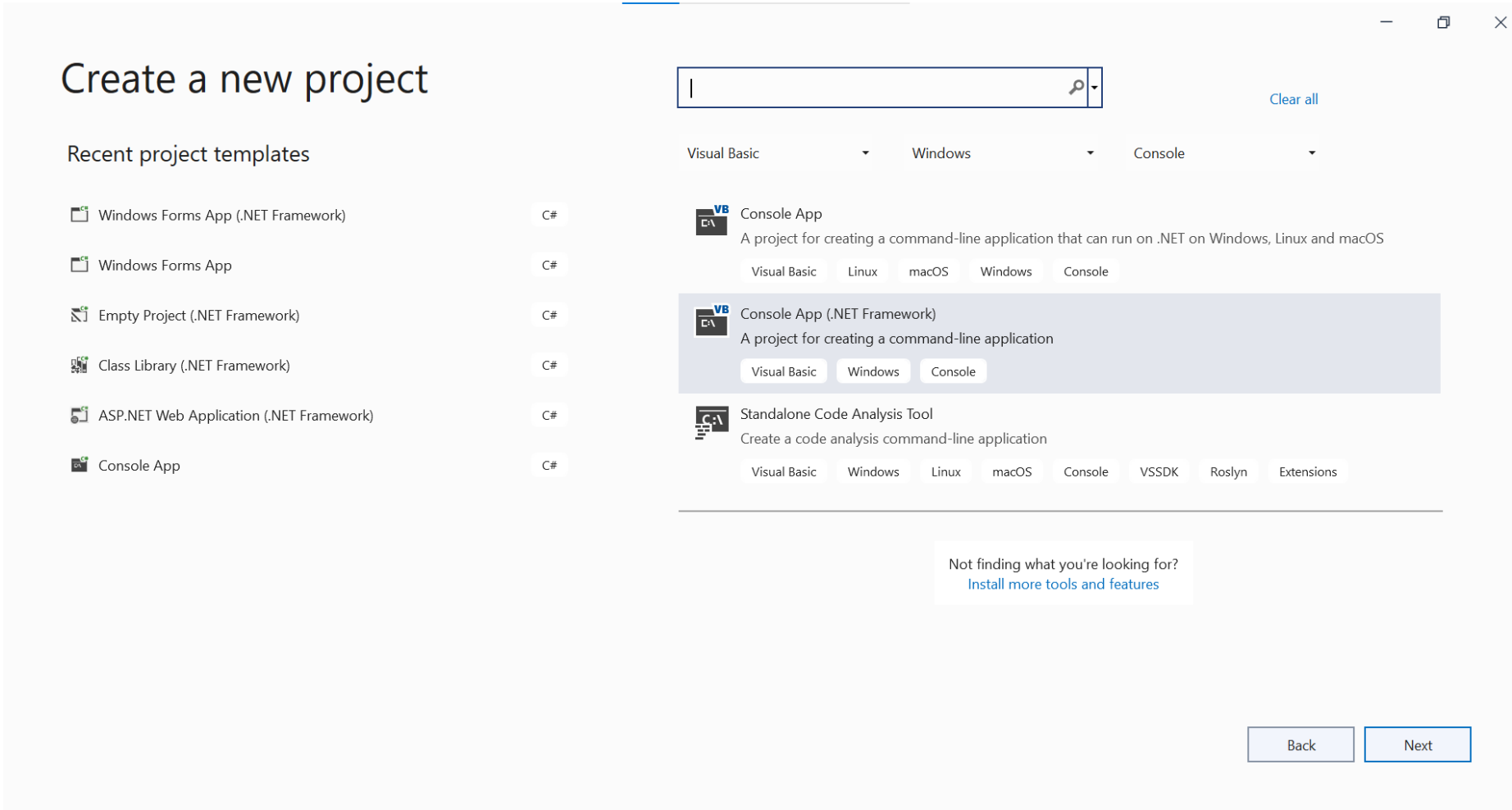
```
Select (variable)
        Case expresion1
            instrucciones1
        Case expresion2
            instrucciones2
        ...
         ...
        Case Else:
           instruccionesN
EndSelect
```

```
Case Is <y        ‘ variable < y
Case 3            ‘ variable =3
Case y to 10      ‘ variable = y, y+1,…..,10
Case 3, x         ‘ variable = 3, x
Case -5, w To 5           ‘ variable = -1, w, w+1….,5
Case “dato”, “DATO”       ‘ variable =”dato”, “DATO”
Case Is >=200             ‘ variable >=200
```

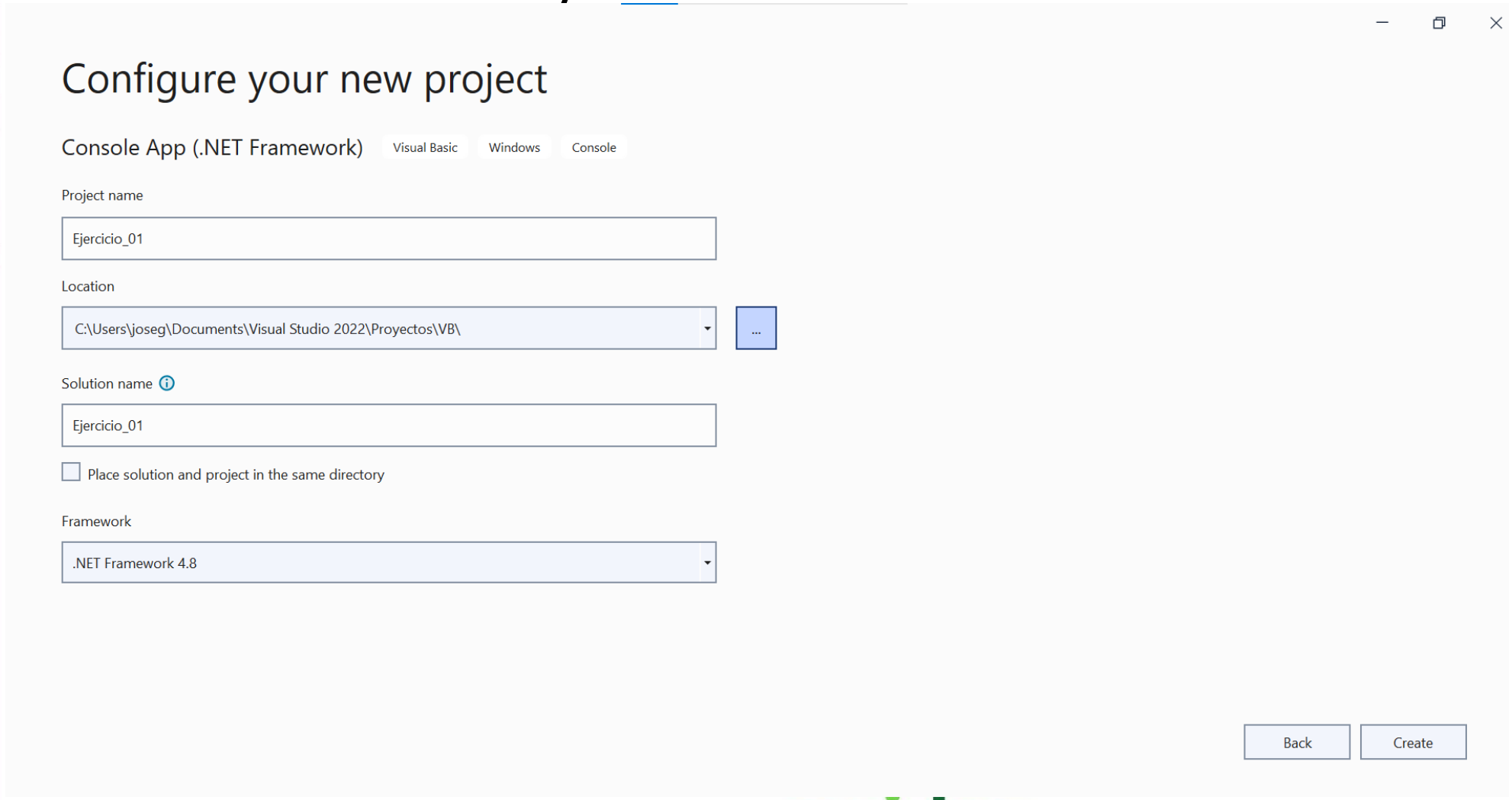
Creación de Proyecto Consola



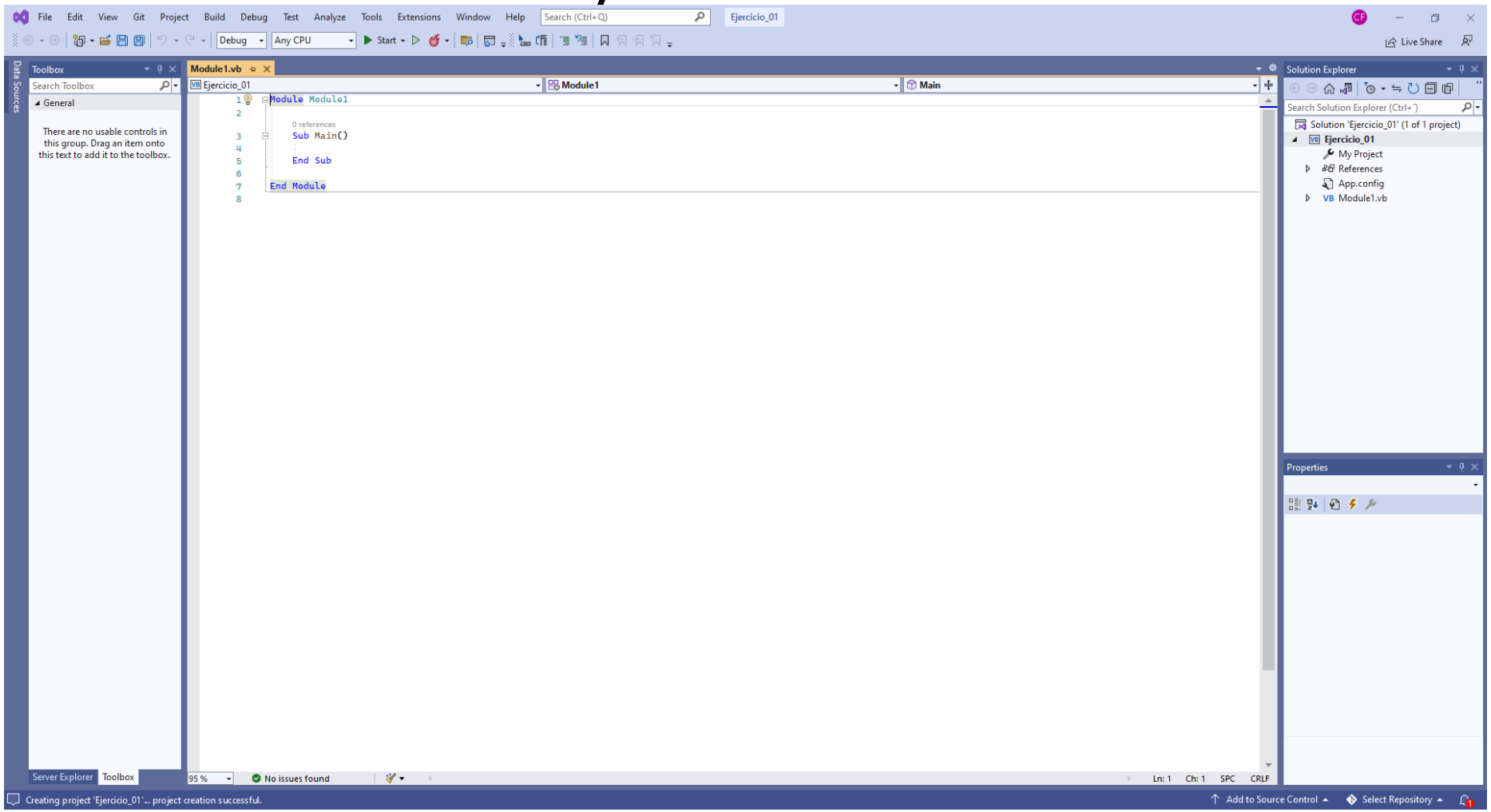
Creación de Proyecto Consola



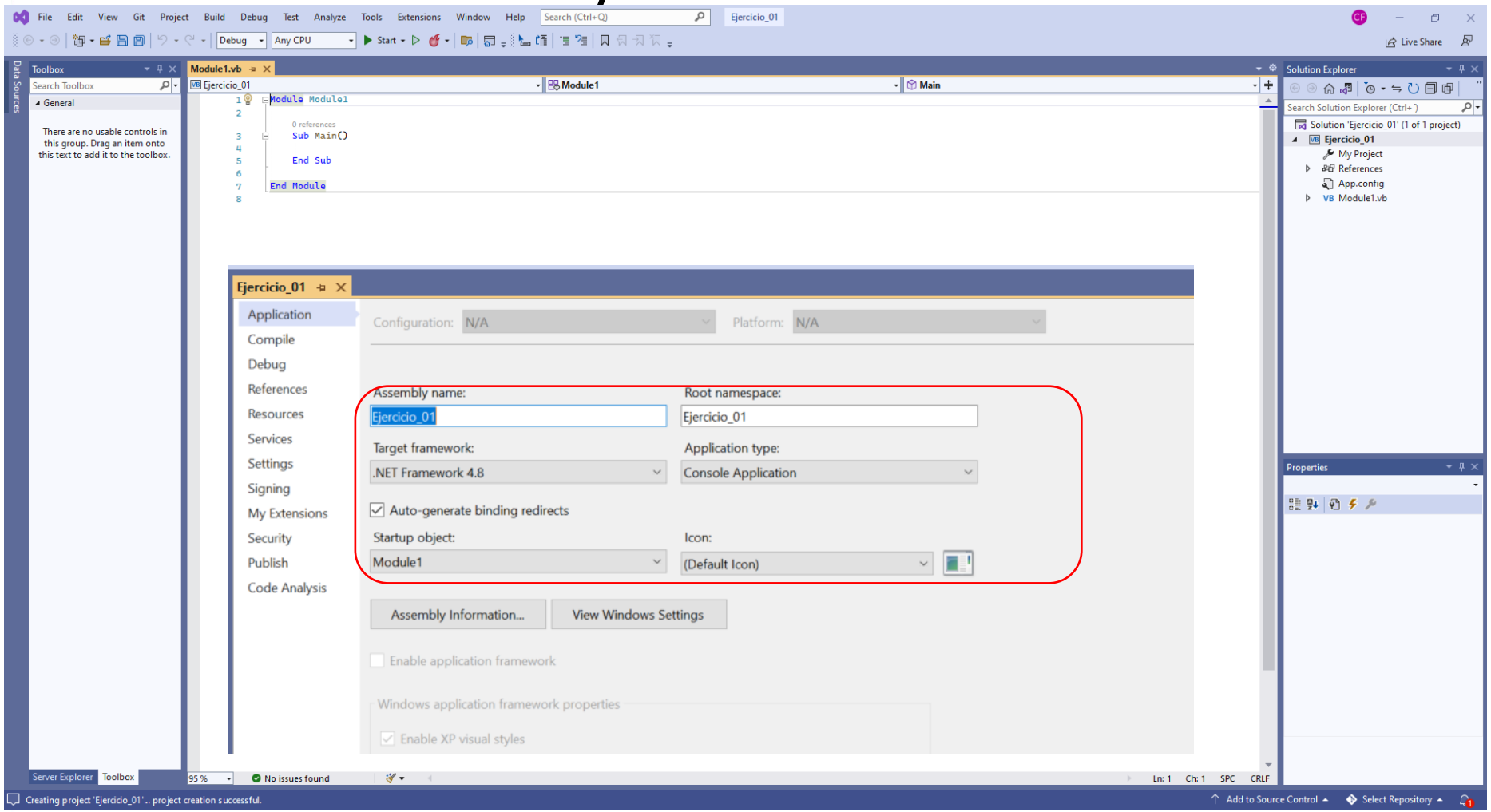
Creación de Proyecto Consola



Creación de Proyecto Consola



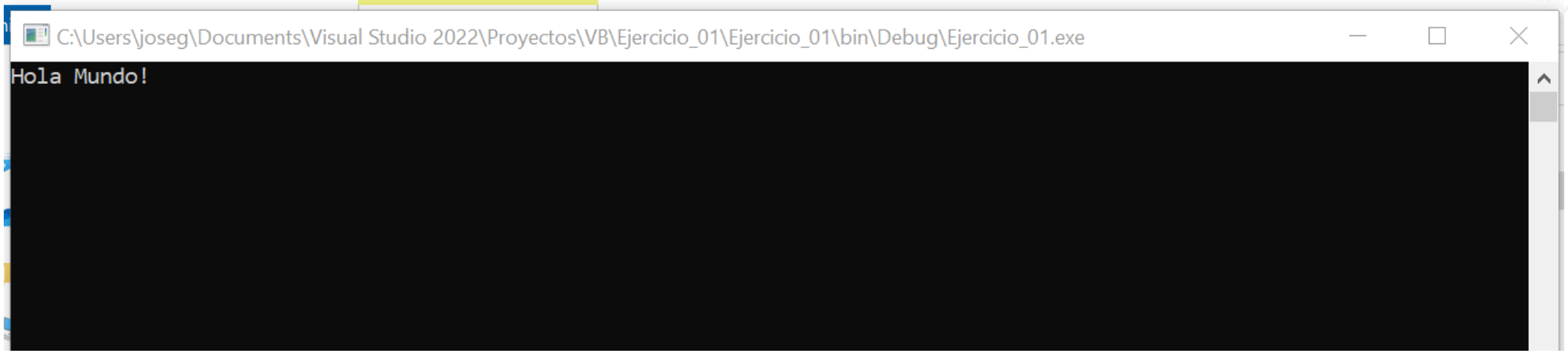
Creación de Proyecto Consola



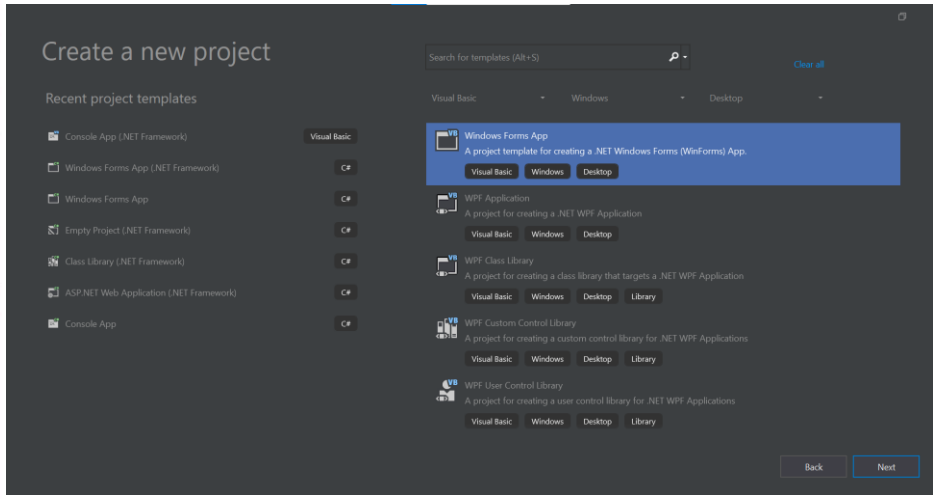
unidos por un cambio transformador

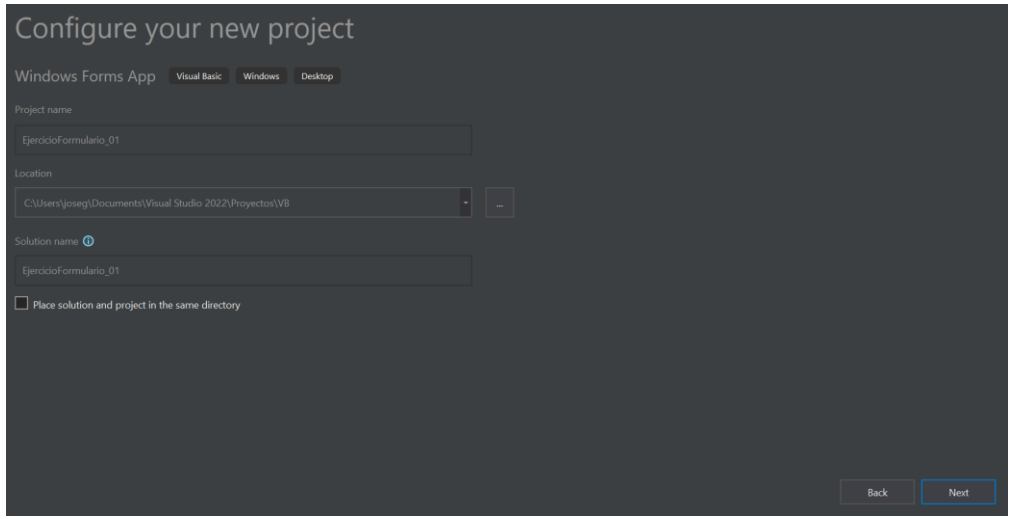
VIVE LA FIME

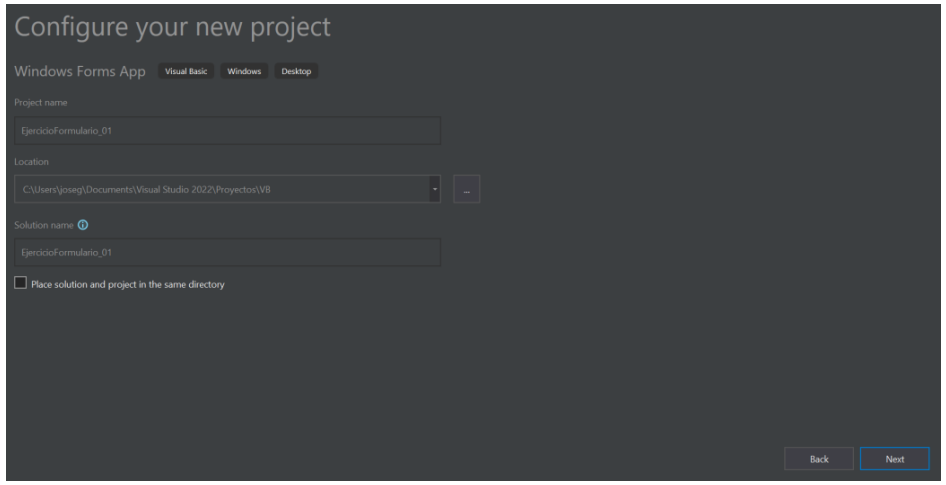
Creación de Proyecto Consola

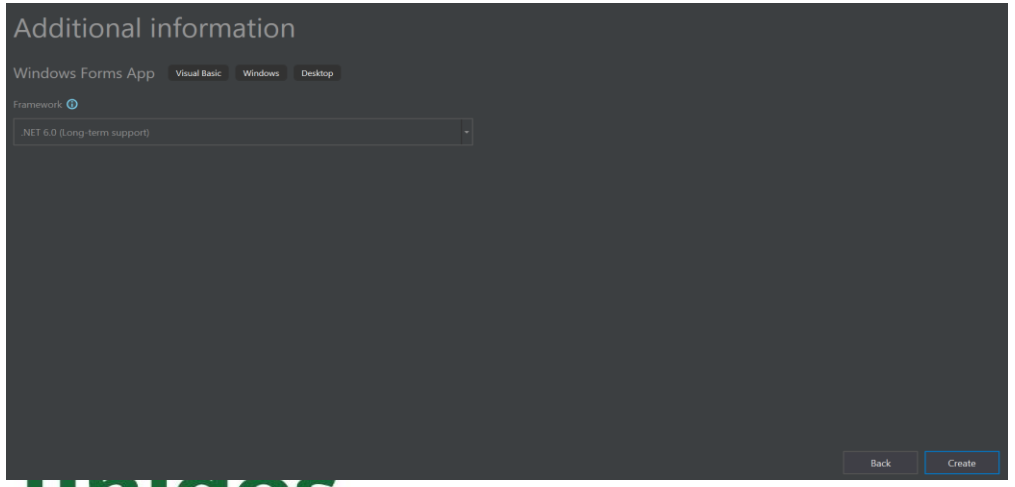


Creación de Proyecto Escritorio

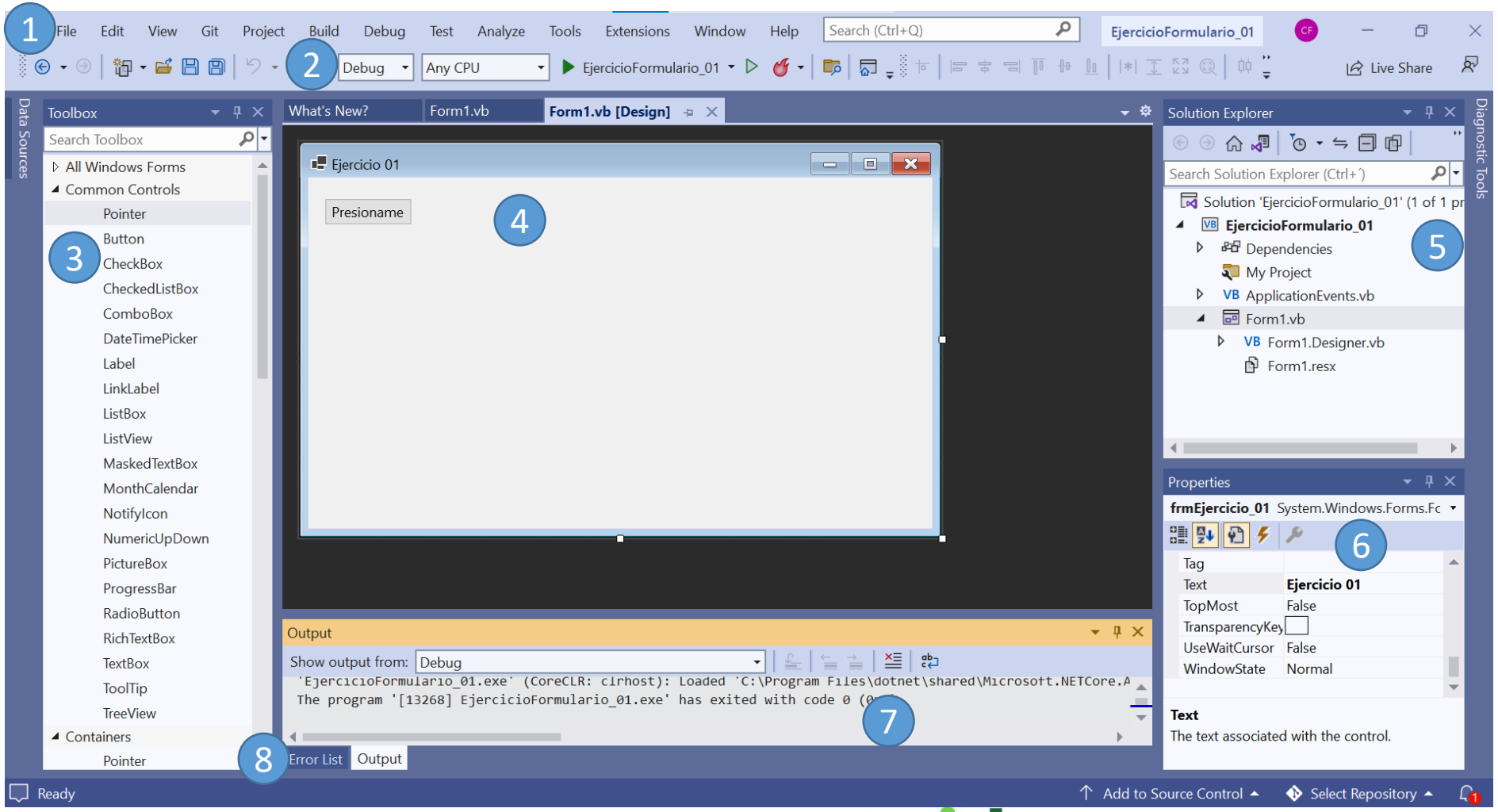








Entorno de Desarrollo



unidos por un cambio transformador

VIVE LA FIME

Entorno de Desarrollo

1. Menús: agrupa las funciones de personalización del IDE, la ayuda, la visualización de las ventanas contenedoras de herramientas, el acceso a propiedades, y la función de compilación de proyectos.
2. Barra de herramientas: en ella se pueden agrupar los principales botones equivalentes a las funciones más utilizadas de los menús. La barra principal incluye las funciones de iniciar un nuevo proyecto, guardar, agregar uno o más objetos al proyecto, compilar y depurar, entre otras.
3. Cuadro de herramientas: agrupa los controles que permiten diseñar la interfaz gráfica de las aplicaciones (botones, cajas de texto, grilla de datos, etc.). Algunos controles pueden no estar visibles al momento de ejecutar nuestro software.
4. Editor de código: es el área de trabajo donde escribimos las sentencias que ejecutará el programa para funcionar.

Entorno de Desarrollo

5. Explorador de soluciones: en él se listarán todos los archivos y recursos internos y externos que componen un proyecto.
6. Ventana de propiedades: desde ella podemos asignar el nombre a formularios y controles, y ajustar las propiedades de cada uno de ellos, la posición en pantalla, las fuentes y los colores, entre otras características.
7. Ventana de inmediato: aquí veremos en modo depuración los valores asignados a variables o a las propiedades de nuestros controles y objetos.
8. Lista de errores: la lista de errores, advertencias y mensajes nos mantendrá informados sobre las equivocaciones cometidas al escribir el código y que el motor de depuración detecte. También permite evaluar datos de variables o espacios de nombre no declarados, entre otras advertencias.