



NOTAS - GITHUB - COPILOT

Resumen

Este documento contiene notas de pruebas realizadas en GitHub Copilot en Visual Studio (VS) y Visual Studio Code (VSC).

Luis Carlos Salas Villalobos

e: lusalas16@gmail.com

w: <https://luiscasalas16.github.io>

1 CONTENIDO

2	Explicar código	2
3	Documentar código	3
4	Optimizar código	4
5	Corregir errores	5
6	Explicar Errores	6
7	Consultas en ventana	8
8	Consultas en contexto	9
9	Generar código en tiempo real	11
10	Generar código desde comentario.....	11
11	Generar código desde consulta.....	12
12	Generar código por comando	13
13	Generar casos de prueba	14
14	Consultas a archivos	15
15	Consultas a IDE	15

2 EXPLICAR CÓDIGO

VS

```
//calculate fibonacci number

/explain

public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
//calculate fibonacci number

Ask Copilot

/explain

The selected code is a method called Fibonacci that calculates the Fibonacci number for a given input n. The Fibonacci sequence is a series of numbers in which each number is the sum of the two preceding ones, usually starting with 0 and 1.

public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

VSC

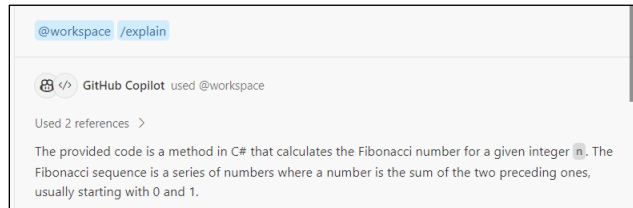
```
/explain

You can also type / for commands

//calculate fibonacci number
int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

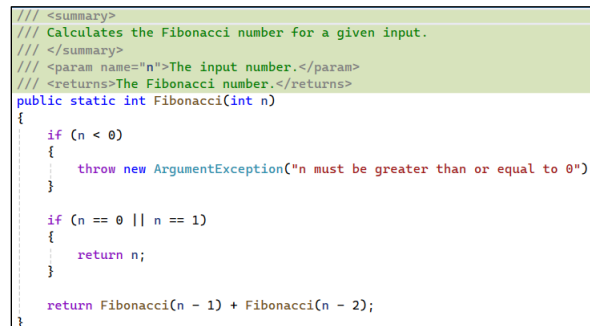
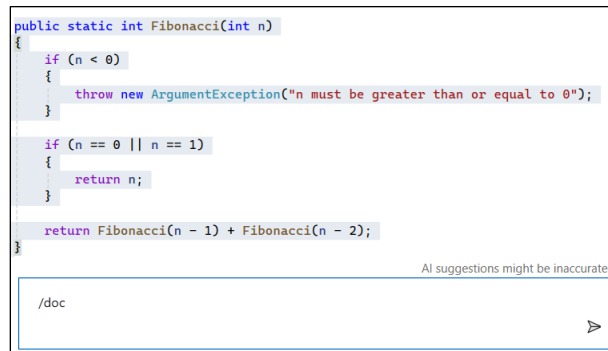
    if (n == 0 || n == 1)
    {
        return 0;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

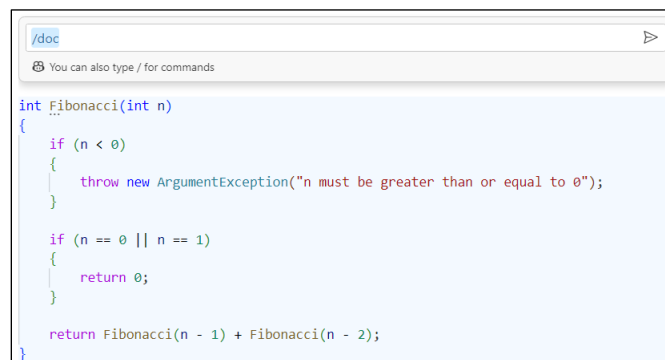


3 DOCUMENTAR CÓDIGO

VS



VSC



```

/// <summary>
/// Calculates the Fibonacci number at the given index.
/// </summary>
/// <param name="n">The index of the Fibonacci number to calculate.</param>
/// <returns>The Fibonacci number at the given index.</returns>
int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return 0;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}

```

4 OPTIMIZAR CÓDIGO

VS

```

public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}

```

AI suggestions might be inaccurate.

/optimize

```

private static Dictionary<int, int> memo = new Dictionary<int, int>();

public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    if (memo.ContainsKey(n))
    {
        return memo[n];
    }

    int result = Fibonacci(n - 1) + Fibonacci(n - 2);
    memo[n] = result;

    return result;
}

```

VSC

```
/optimize >
You can also type / for commands

int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }
    if (n == 0 || n == 1)
    {
        return 0;
    }
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
Dictionary<int, int> memo = new Dictionary<int, int>();

int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    if (memo.ContainsKey(n))
    {
        return memo[n];
    }

    int result = Fibonacci(n - 1) + Fibonacci(n - 2);
    memo[n] = result;
    return result;
}
```

5 CORREGIR ERRORES

1 VS

```
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}

AI suggestions might be inaccurate.

/fix >
```

```
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0)
    {
        return 0;
    }

    if (n == 1)
    {
        return 1;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

1 VSC

```
static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

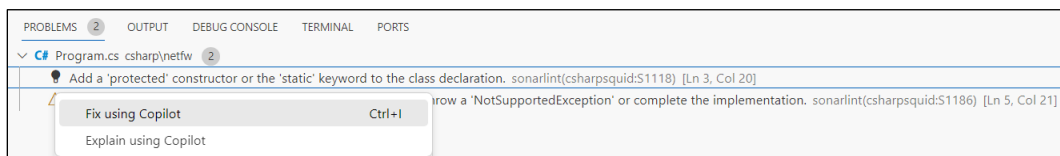
    if (n == 0)
    {
        return 0;
    }

    if (n == 1)
    {
        return 1;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

2 VSC

```
namespace netfw
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```



```
namespace netfw
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```

Accept Discard Changed 1 line

```
internal class Program → 3+ internal static class Program
{
    0 references
    static void Main(string[] args)
    {
    }
}
```

6 EXPLICAR ERRORES

VS

Error List - Current Document (Program.cs)	
Current Document	0 Errors 2 Warnings 0 of 1 Message Build + IntelliSense
Code	Description
S1118	Add a 'protected' constructor or the 'static' keyword to the class declaration.
S1186	Add a nested comment explaining why this method is empty, throw a 'NotSupportedException' or complete the implementation.

C# Program.cs /explain Add a 'protected' constructor or the 'static' keyword to the class declaration.
References

GitHub Copilot

In C#, the **protected** keyword is used to specify that access is limited to the containing class or types derived from the containing class. A **protected** constructor in a class means that the class cannot be instantiated from outside, but can be extended, and the derived classes can call the base class constructor.

On the other hand, the **static** keyword on a class means that the class cannot be instantiated or extended. It can only contain static members. This is useful when you want a class to just provide some functionality, but it doesn't have or need any state.

Here's how you can add a **protected** constructor to your class:

```
namespace TestCopilot
{
    internal class Program
    {
        protected Program()
        {
            // constructor logic here
        }

        static void Main(string[] args)
        {
        }
    }
}
```

Insert in new file
Preview

And here's how you can make your class **static**:

```
namespace TestCopilot
{
    internal static class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Insert in new file
Preview

Remember, if you make your class **static**, you cannot have instance constructors (like a **protected** constructor) in it.

- What is the purpose of a 'protected' constructor in a base class?
- Can a 'static' class have instance methods or properties?

References

VSC

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

C# Program.cs csharp\netw 2

Add a 'protected' constructor or the 'static' keyword to the class declaration. sonarlint(csharpquid:S1118) [Ln 3, Col 20]

Add a nested comment explaining why this method is empty, throw a 'NotSupportedException' or complete the implementation. sonarlint(csharpquid:S1186) [Ln 5, Col 21]

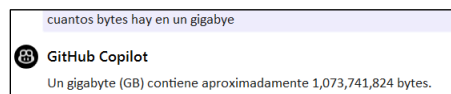
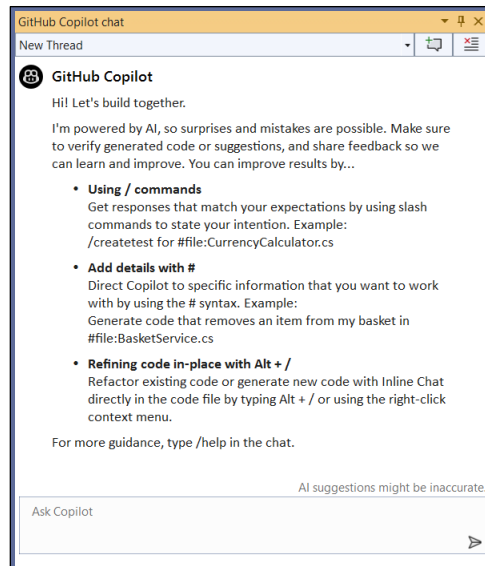
Fix using Copilot Ctrl+I

Explain using Copilot



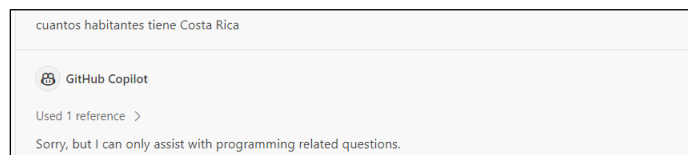
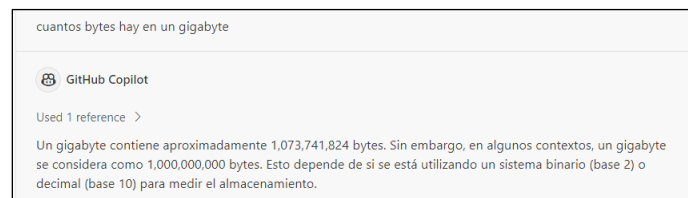
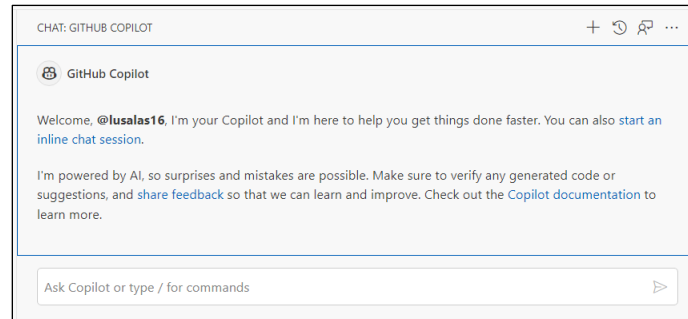
7 CONSULTAS EN VENTANA

VS



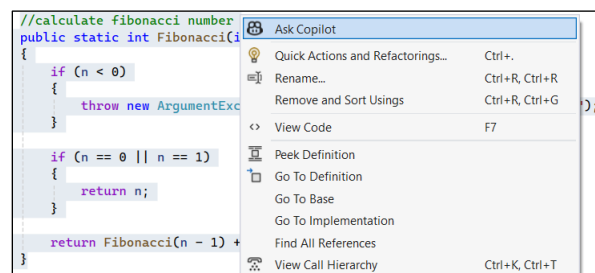


VSC



8 CONSULTAS EN CONTEXTO

VS



```
//calculate fibonacci number

que hace este código
```

```
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
//calculate fibonacci number

Ask Copilot
```

que hace este código

Este código implementa una función llamada `Fibonacci` que calcula el número de Fibonacci para un número dado `n`. La secuencia de Fibonacci es una serie de números en la que cada número es la suma de los dos números anteriores.

```
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

VSC

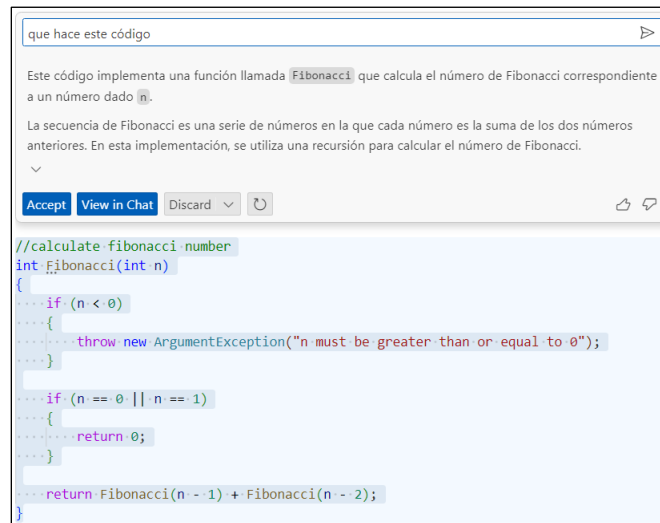
```
que hace este código
```

You can also type / for commands

```
//calculate fibonacci number
int fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

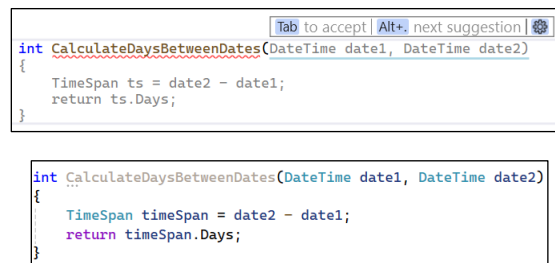
    if (n == 0 || n == 1)
    {
        return 0;
    }

    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

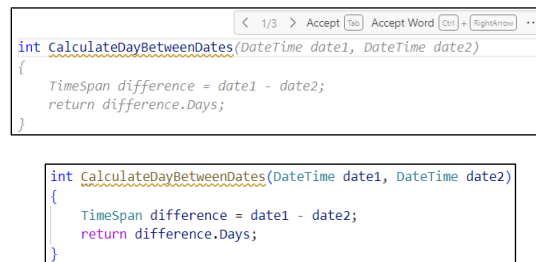


9 GENERAR CÓDIGO EN TIEMPO REAL

VS



VSC



10 GENERAR CÓDIGO DESDE COMENTARIO

VS

```
//calculate fibonacci number
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
//calculate fibonacci number
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

VSC

```
//calculate fibonacci number
int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0)
    {
        return 0;
    }

    if (n == 1)
    {
        return 1;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
//calculate fibonacci number
int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

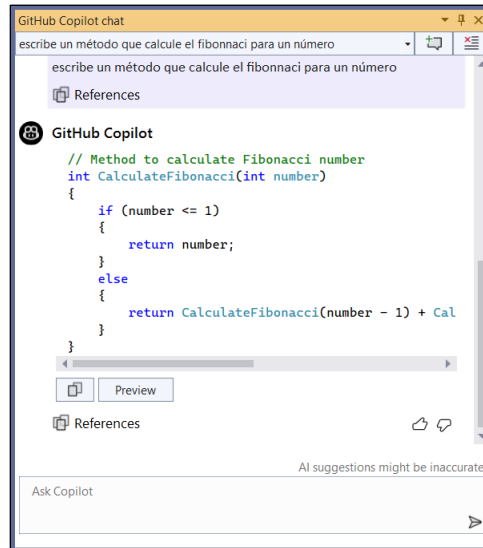
    if (n == 0)
    {
        return 0;
    }

    if (n == 1)
    {
        return 1;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

11 GENERAR CÓDIGO DESDE CONSULTA

VS

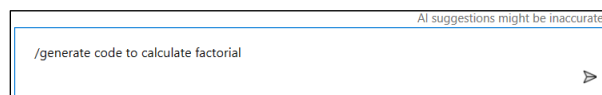


VSC



12 GENERAR CÓDIGO POR COMANDO

VS



```
static int CalculateFactorial(int n)
{
    if (n == 0)
        return 1;
    else
        return n * CalculateFactorial(n - 1);
}
```

VSC

You can also select code to make an edit

```
int CalculateFactorial(int number)
{
    if (number == 0)
        return 1;

    int factorial = 1;
    for (int i = 1; i <= number; i++)
    {
        factorial *= i;
    }

    return factorial;
}
```

13 GENERAR CASOS DE PRUEBA

VS

```
//calculate fibonacci number
public static int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return n;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

AI suggestions might be inaccurate.

```
+using NUnit.Framework;
+
+namespace TestCopilot.Tests
+{
+    public class ProgramTests
+    {
+        [Test]
+        public void Fibonacci_ReturnsCorrectResult()
+        {
+            // Arrange
+            int n = 5;
+            int expected = 5;
+
+            // Act
+            int result = Program.Fibonacci(n);
+
+            // Assert
+            Assert.AreEqual(expected, result);
+        }
+    }
+}
```

VSC

```
/tests
You can also type / for commands

//calculate fibonacci number
int Fibonacci(int n)
{
    if (n < 0)
    {
        throw new ArgumentException("n must be greater than or equal to 0");
    }

    if (n == 0 || n == 1)
    {
        return 0;
    }

    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

```
[Fact]
0 references
public void TestFibonacci()
{
    // Arrange
    int n = 6;
    int expectedFibonacci = 5;

    // Act
    int actualFibonacci = Fibonacci(n);

    // Assert
    Assert.Equal(expectedFibonacci, actualFibonacci);
}
```

14 CONSULTAS A ARCHIVOS

VS

#file:Program.cs cuantos métodos tiene ?

References

GitHub Copilot

El archivo **Program.cs** en el contexto proporcionado tiene dos métodos. Estos son:

1. **Main(string[] args)**: Este es el método principal que se ejecuta cuando se inicia el programa.
2. **Fibonacci(int n)**: Este es un método que calcula el número de Fibonacci en la posición **n**.

VSC

#file:Program.cs cuantos métodos tiene ?

GitHub Copilot

Used 1 reference >


The file **Program.cs** contains one method. The method is named **Fibonacci** and it takes an integer **n** as an argument. This method is used to calculate the Fibonacci number at the position **n**.

15 CONSULTAS A IDE

VS

/askvs How do I connect Visual Studio to a Git Repository?

References

 **GitHub Copilot**



To connect Visual Studio to a Git repository, follow these steps:

1. Open Visual Studio.
2. Click on the **Team Explorer** tab. If it's not visible, you can open it by going to **View > Team Explorer**.
3. In the **Team Explorer** window, click on the **Manage Connections** button.
4. Click on the **Connect** button under the **Local Git Repositories** section.
5. Click on the **Clone** button.
6. Enter the URL of the Git repository you want to connect to in the **URL** field.
7. Specify the local path where you want to clone the repository in the **Path** field.
8. Click on the **Clone** button to clone the repository and connect Visual Studio to it.

Please note that you need to have Git installed on your machine to use Git features in Visual Studio.


[How do I commit changes in Visual Studio?](#)

[How do I push changes to the remote repository from Visual Studio?](#)

References  

VSC

@vscode How do I connect Visual Studio Code to a Git Repository?

 **GitHub Copilot** used @vscode

To connect Visual Studio Code to a Git repository, you can use the **Git: Open Repository** command (`git.openRepository`). This command will prompt you to select a local directory which is a Git repository and open it in VS Code.

If you want to initialize a new Git repository in your current workspace, you can use the **Git: Initialize Repository** command (`git.init`). This command will create a new Git repository in your workspace.

Show "Git: Open Repository"

Show "Git: Initialize Repository"