

SESSÃO 3

Primeiros Passos com Git

Objetivos:

Trabalhar com o *GitHub Desktop*:

- configurar
- adicionar um repositório existente
- visualizar e analisar o histórico de *commits*
- criar *commits*
- publicar um repositório
- clonar um repositório

Aprofundar os conhecimentos da aplicação:

- criar e configurar um novo repositório,
- ignorar ficheiros;
- efetuar *commit* seletivo de ficheiros;

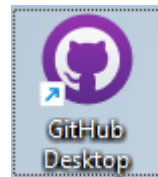
Aplicar regras e recomendações para a escrita de mensagens de *commit*

Criar e editar o ficheiro README.md

Transcrição

Vídeo 6

<https://www.youtube.com/watch?v=zv15DIBzmiE>

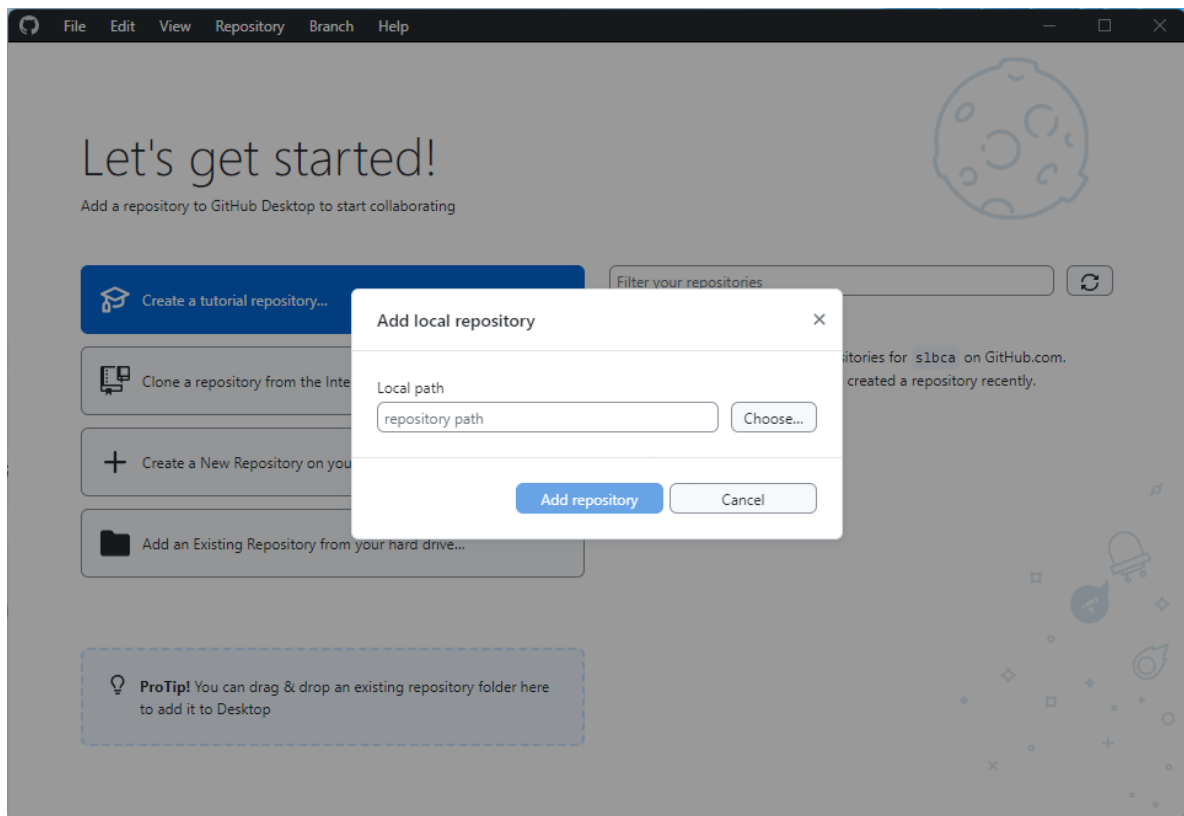


1. Verificar se a aplicação está configurada para utilizar a conta do GitHub para isso executar os seguintes passos:

- menu **File** comando **Options...** e no separador **Accounts** verificar a presença da conta.
- caso a sessão não esteja iniciada clicar em **sign in**, na caixa seguinte **Continue with browser**, introduzir os dados de acesso.

2. Utilizar um repositório existente

- menu **File** comando **Add local repository...** e na caixa aberta selecionar a localização do repositório

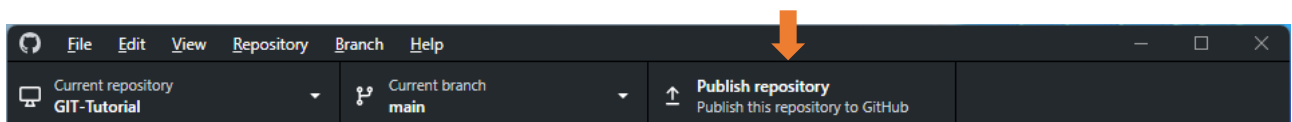


3. Analisar a janela do repositório aberta

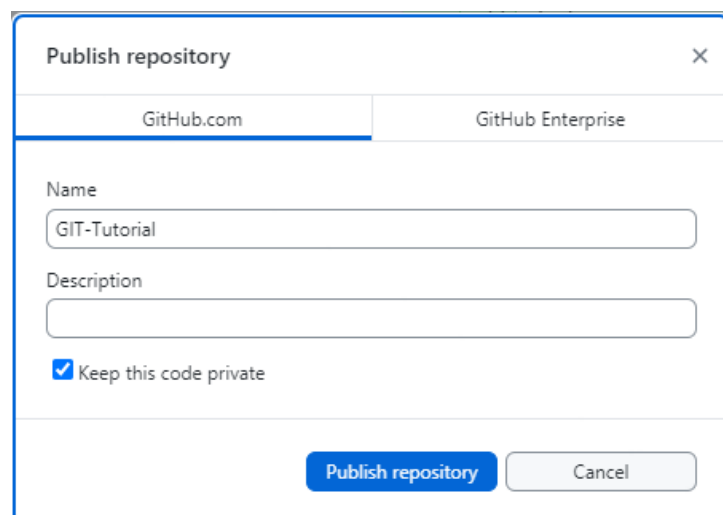
- a) clicar no separador **History** para verificar os **commits** realizados e respectivas alterações.

Commit early, commit often

4. Publicar o repositório



- a) clicar em **Publish repository** e preencher a janela



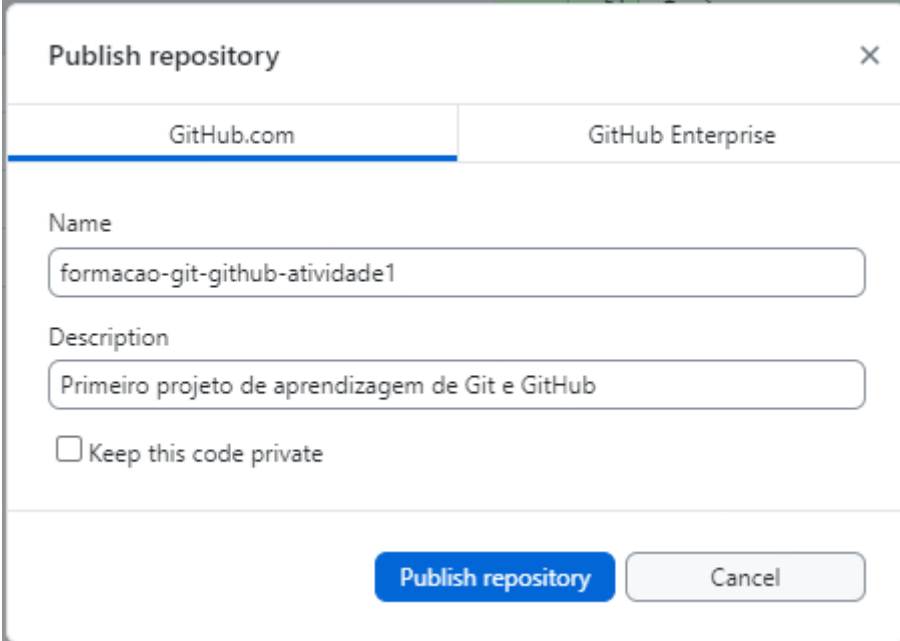
Name: formacao-git-github-atividade1

Nota: no nome dos ficheiros não são aceites cedilhas, caracteres acentuados, espaços; antes de continuar serão apresentadas abaixo as alterações.

Description: formacao-git-github-atividade1

Keep this code private – para manter o repositório privado; retirar o visto para tornar o repositório público.

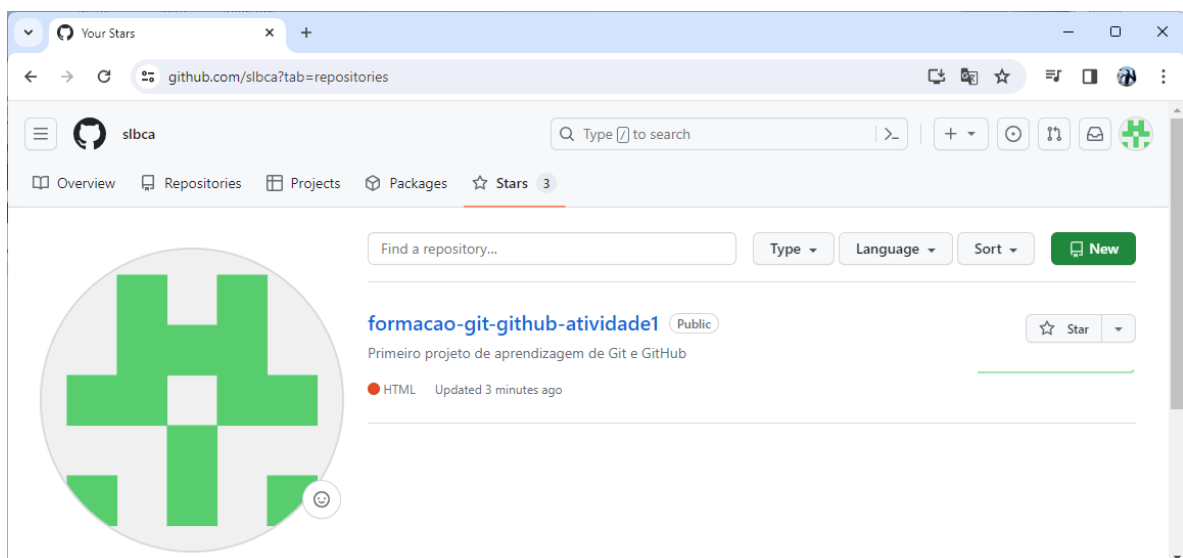
Organização – só se estivermos inseridos numa organização; não é o caso.



b) fazer as alterações e clicar em **Publish repository**

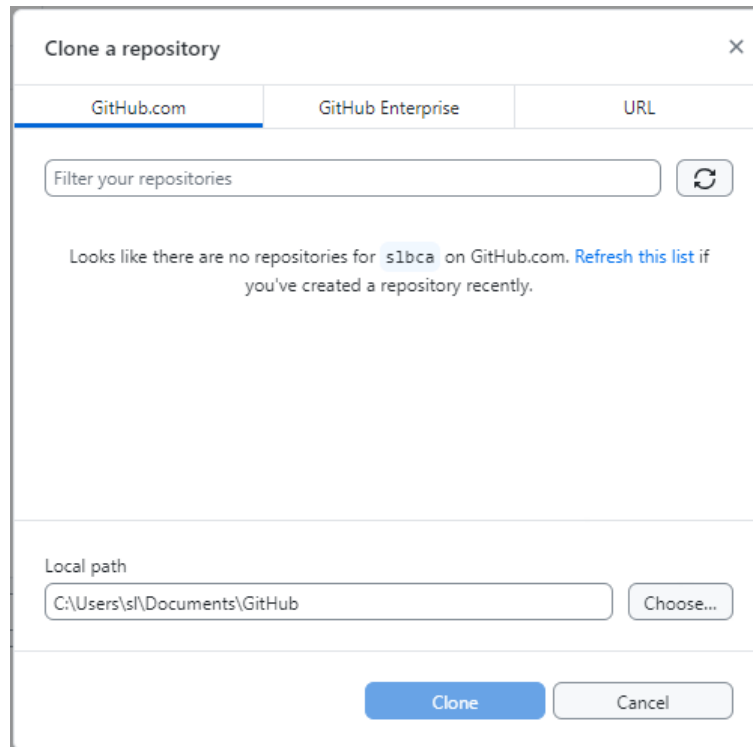
5. Aceder ao repositório publicado

a) a partir do **browser** e na conta pessoal no **Git** visualizar o repositório criado, no separador **Repositories**

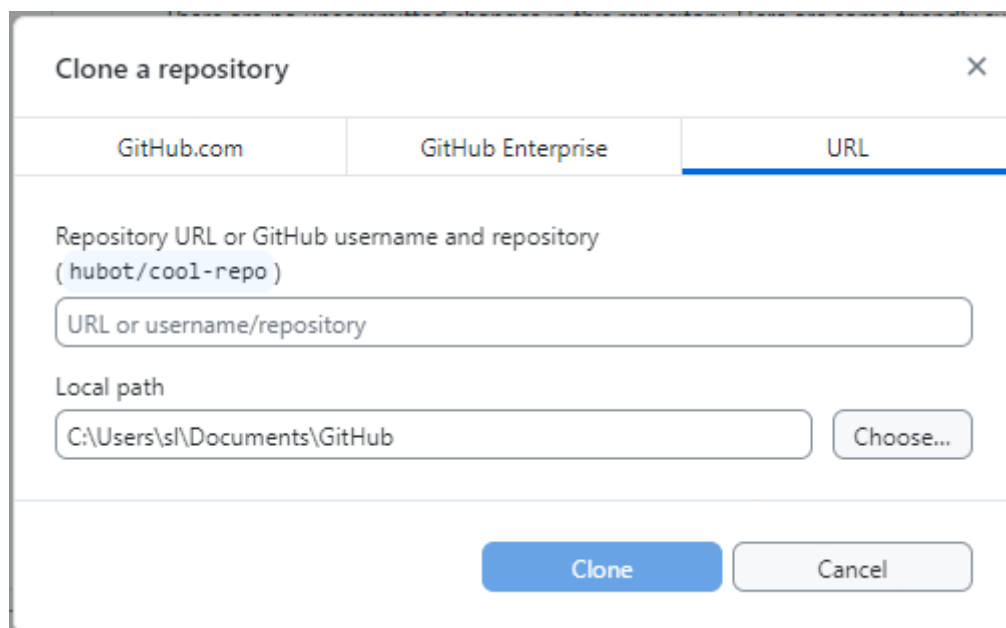


6. Clonar um repositório com o GitHub Desktop

a) menu **File** comando **Clone repository...**



b) clicar no **separador URL**



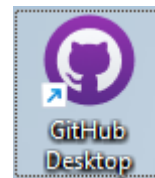
c) colar na caixa **URL** o endereço de um repositório, por exemplo,

<https://github.com/williammalone/Simple-HTML5-Drawing-App.git>

d) definir um local para a colocação do clone através da opção **Local path**

e) clicar no botão **Clone**.

<https://www.youtube.com/watch?v=Cf99zYp2MTs>



1. Criar um novo repositório com o Git Desktop – aplicação C# para efetuar cálculos

a) menu **File** comando **New repository...**

This is a screenshot of the "Create a new repository" dialog box in Git Desktop. The dialog has a title bar with a close button (X). It contains several input fields: "Name" with the placeholder text "repository name", "Description" (empty), "Local path" with the text "C:\Users\sl\Desktop" and a "Choose..." button, a checkbox for "Initialize this repository with a README" which is currently unchecked, a "Git ignore" dropdown menu set to "None", and a "License" dropdown menu also set to "None". At the bottom, there are two buttons: "Create repository" (highlighted in blue) and "Cancel".

b) Preencher os diferentes campos da janela acima.

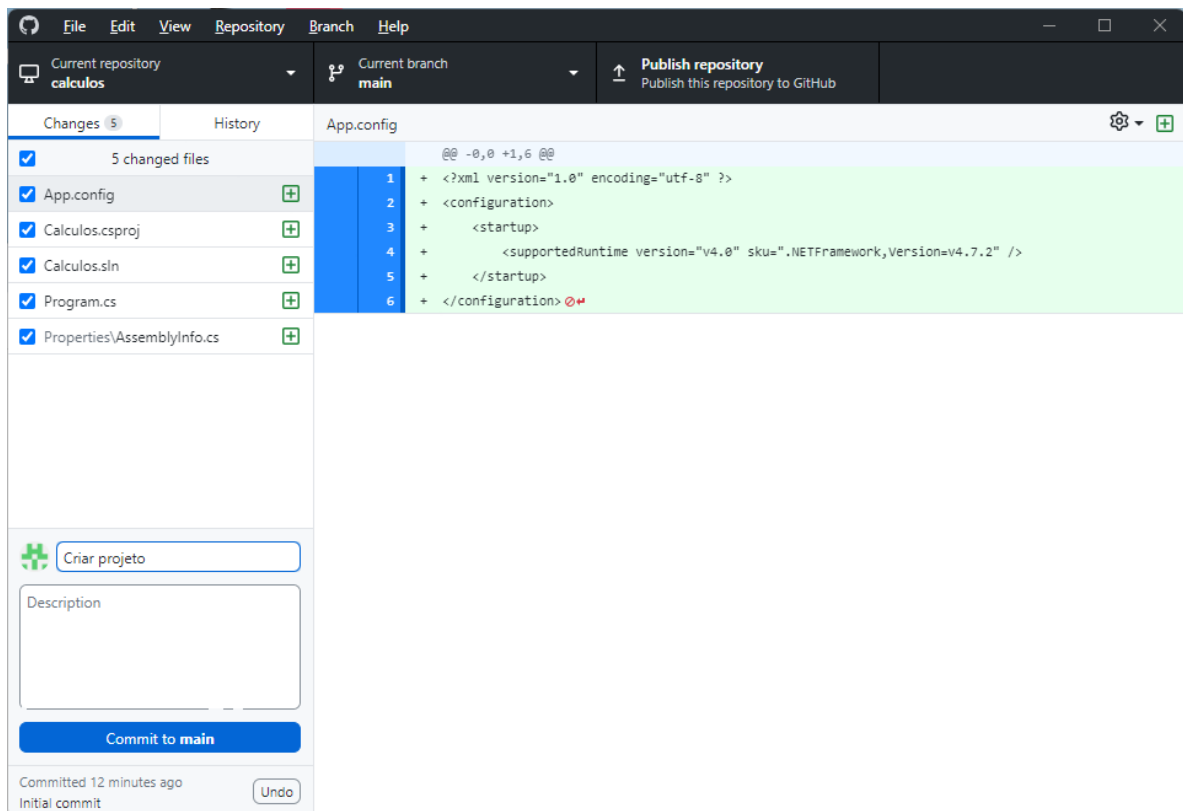
This is a second screenshot of the "Create a new repository" dialog box, showing it after the fields have been filled. The "Name" field now contains "calculos". The "Description" field contains "Aplicação C# para efetuar diversos tipos de cálculos". The "Local path" field remains "C:\Users\sl\Desktop". The "Initialize this repository with a README" checkbox is now checked. The "Git ignore" dropdown menu is set to "VisualStudio". The "License" dropdown menu is set to "MIT License". The "Create repository" button remains highlighted in blue, and the "Cancel" button is also visible.

c) Abrir a pasta do repositório criada anteriormente, no ambiente de trabalho.

| | |
|----------------|--|
| .git | indica que temos um repositório |
| .gitattributes | ficheiro com um conjunto de definições a aplicar a ficheiros ou conjunto de ficheiros |
| .gitignore | permite ignorar certos tipos de ficheiros e pastas deve ser colocado na raiz do repositório |
| LICENSE | texto da licença |
| README.md | Descrição do projeto |

d) Descompactar a pasta **calculos.zip** para a pasta **calculos** do repositório.

e) Definir um segundo **commit** com a mensagem **criar projeto**.



2. GITIGNORE

a) Serve para colocar ficheiros que devem ser ignorados pelo **git**, por exemplo, ficheiros com informação sensível.

b) No git existe um conjunto de **templates** para o **gitignore**, no exemplo será utilizado o associado ao **visual studio**.

<https://github.com/github/gitignore>

c) Clicar com o **botão direito do rato** sobre o ficheiro **.gitignore** e pedir para abrir com o **Visual Studio Code**.

.gitignore

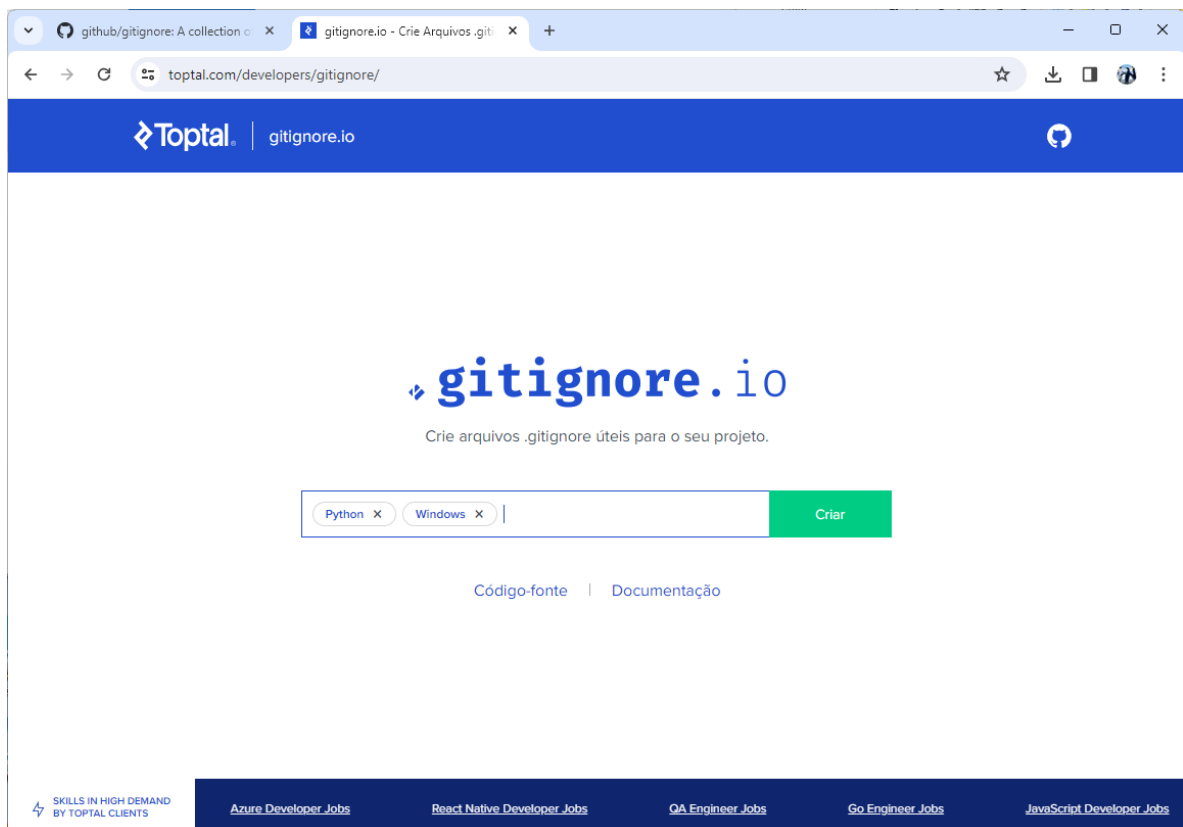
| opção | descrição |
|-----------------|--|
| # | comentário |
| / | pasta |
| ! | negação |
| * | qualquer conjunto de caracteres (exceto /) |
| ? | qualquer caracter (exceto /) |
| [] | <i>range</i> : qualquer um dos caracteres especificados |
| <i>exemplos</i> | |
| *.pdf | ignorar todos os ficheiros .pdf |
| .vs/ | ignorar a pasta .vs |
| [Dd]atabase | ignorar os ficheiros Database e database |
| *[0..9]*.txt | ignorar todos os ficheiros de texto cujo nome contenha números |

Nota: a página

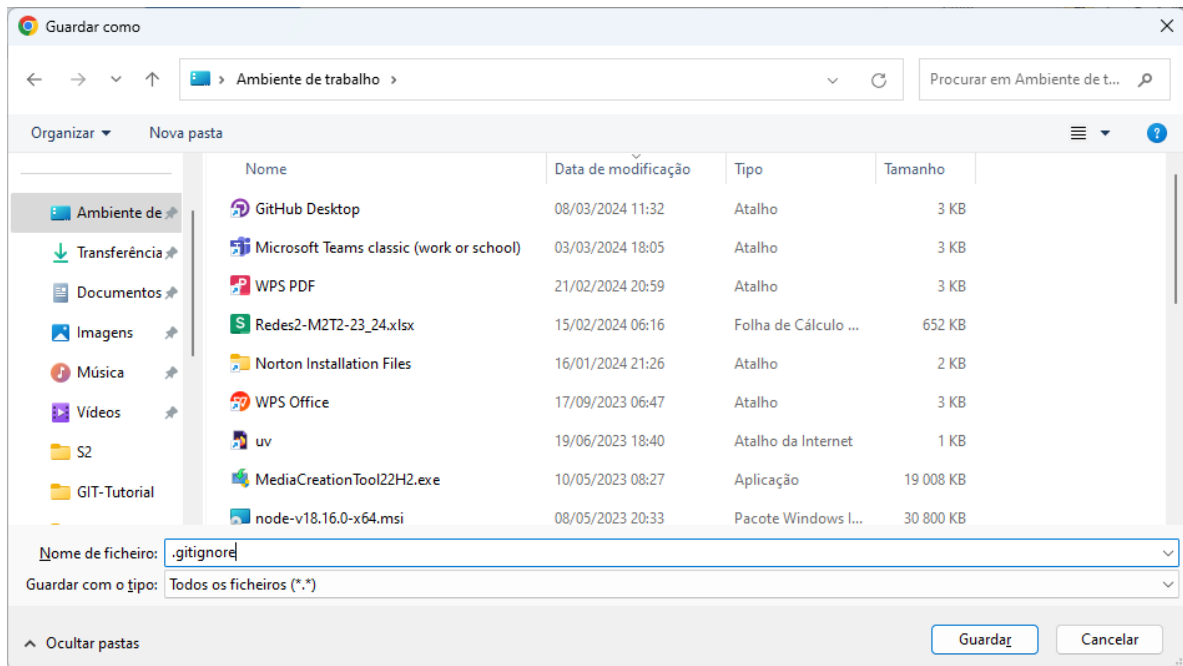
<https://www.toptal.com/developers/gitignore/>

Permite criar um ficheiro *gitignore* com base num ficheiro inserido pelo utilizador.

3. Criar o ficheiro **gitignore** na página <https://www.toptal.com/developers/gitignore/> com base nas seguintes palavras Python e Windows



- a) Clicar em **Criar**.
- b) **CTRL + S** para gravar. Selecionar o ambiente de trabalho, extensão **Todos os ficheiros (*.*)** e no nome **.gitignore**.



4. Criar um ficheiro **.gitignore** sem o **Git Desktop**.

- a) Criar a pasta **NovoProjeto** no ambiente de trabalho.
- b) Abrir a **Git Bash** e digitar os comandos

```

MINGW64:~
$ cd "C:\Users\s1\Desktop\NovoProjeto"

MINGW64 ~/Desktop/NovoProjeto
$ git init
Initialized empty Git repository in C:/Users/s1/Desktop/NovoProjeto/.git/

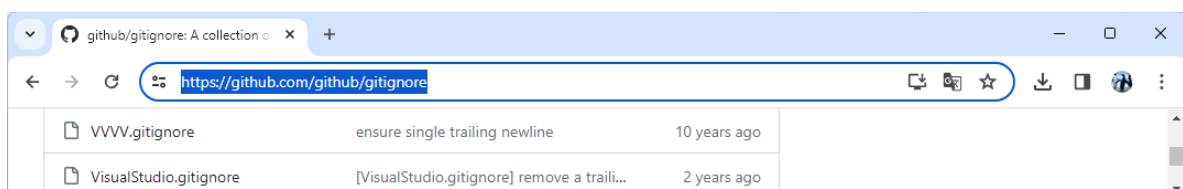
MINGW64 ~/Desktop/NovoProjeto (main)
$

```

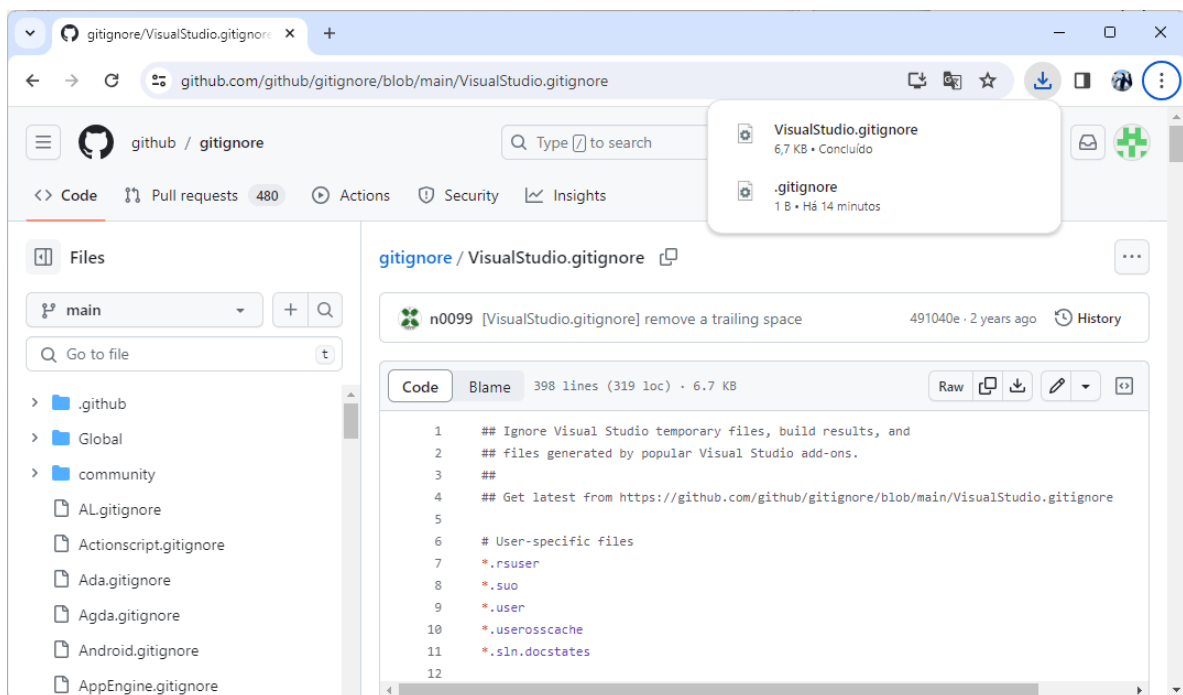
- c) Através do site

<https://github.com/github/gitignore>

identificar e selecionar o ficheiro **VisualStudio.gitignore**



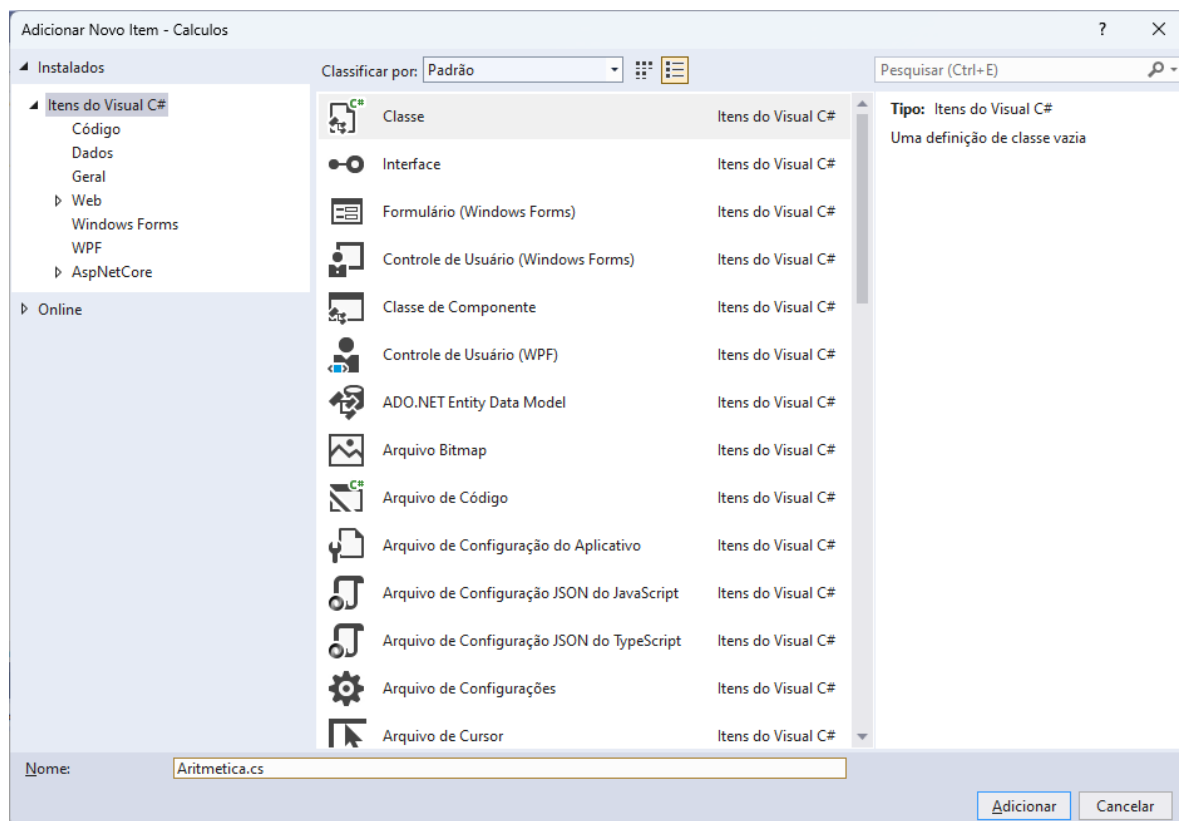
d) Fazer o *download* do ficheiro.



e) Gravar o ficheiro na pasta do **Novo Projeto** com o nome **.gitignore**

5. Na pasta **calculos**, localizar o ficheiro **Calculos.sln** e duplo clique para abrir.

a) Criar uma nova classe através de botão direito do rato sobre **Calculos**, comando **Adicionar** e novo submenu **Classe...Gravar** com o nome **Aritmetica.cs**



b) Implementar o método somar

```

namespace Calculos
{
    /// <summary>
    /// Implementa operações aritméticas
    /// </summary>
    /// 0 referências
    internal class Aritmetica
    {
        /// 0 referências
        public int Somar(int x, int y)
        {
            return x + y;
        }
    }
}

```

c) Gravar o ficheiro.

d) No **Git Desktop**, separador **Changes** seleccionar o ficheiro **Aritmetica.cs** e adicionar commit com o seguinte texto:

Criar classe Aritmetica e adicionar método Somar

Clicar em **Commit to main**

e) Na classe **Aritmetica** do **Visual Studio** adicionar o método subtrair e gravar.

```

namespace Calculos
{
    /// <summary>
    /// Implementa operações aritméticas
    /// </summary>
    /// 0 referências
    internal class Aritmetica
    {
        /// 0 referências
        public int Somar(int x, int y)
        {
            return x + y;
        }

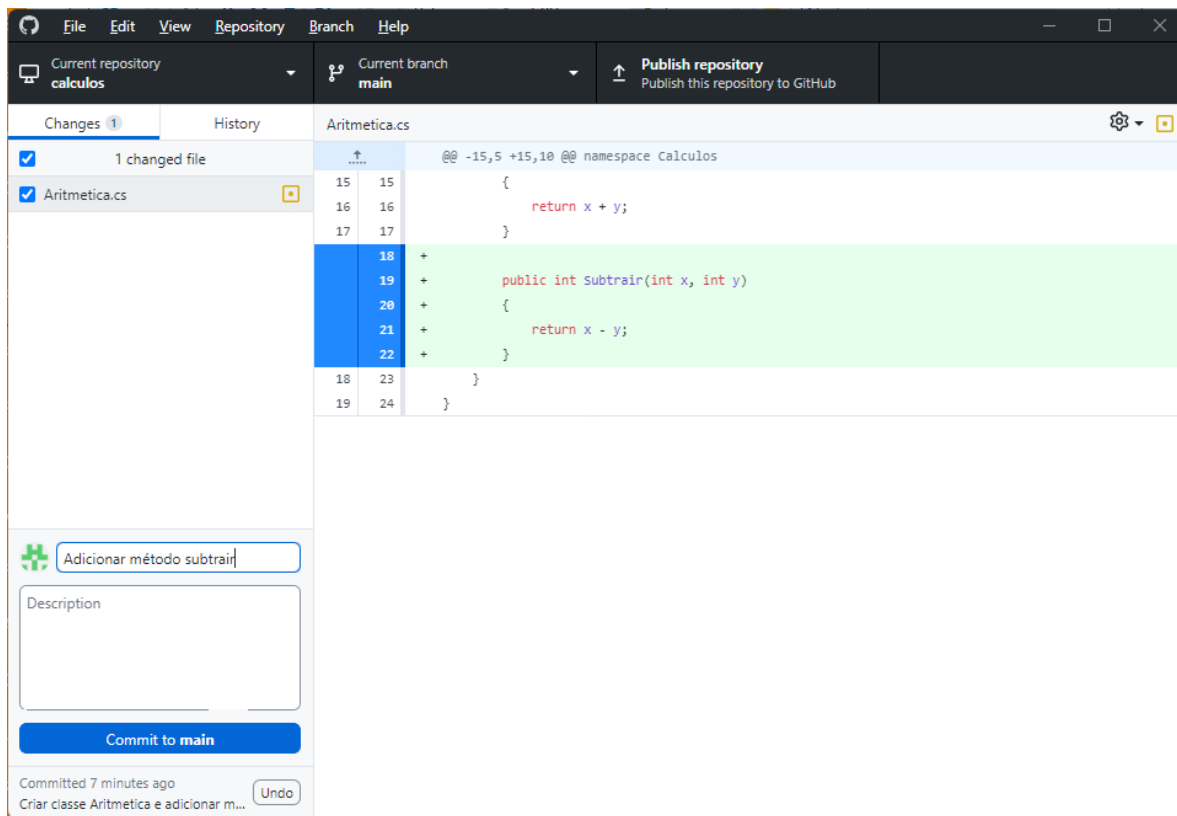
        /// 0 referências
        public int Subtrair(int x, int y)
        {
            return x - y;
        }
    }
}

```

f) No **Git Desktop**, separador **Changes** seleccionar o ficheiro **Aritmetica.cs** e adicionar commit com o seguinte texto:

Criar método Subtrair

Clicar em **Commit to main**

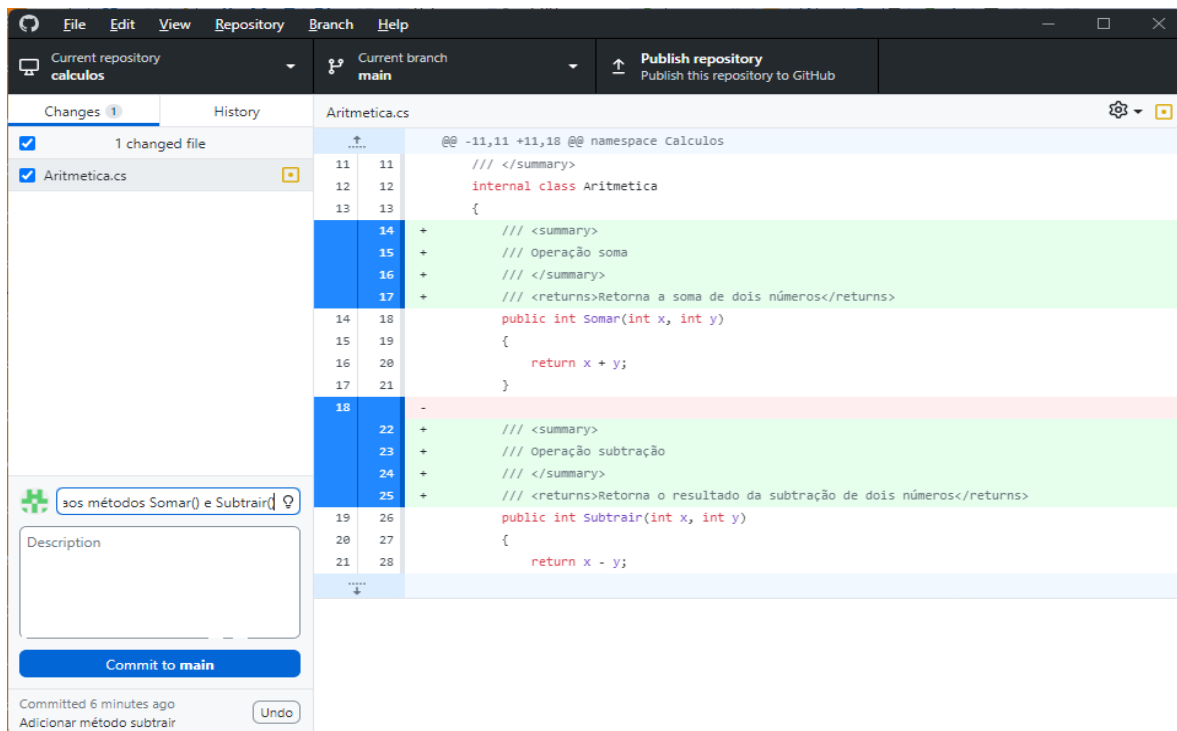


g) Adicionar comentário XML no VS

```
internal class Aritmetica
{
    /// <summary>
    /// Operação soma
    /// </summary>
    /// <returns>Retorna a soma de dois números</returns>
    /// Referências
    public int Somar(int x, int y)
    {
        return x + y;
    }
    /// <summary>
    /// Operação subtração
    /// </summary>
    /// <returns>Retorna o resultado da subtração de dois números</returns>
    /// Referências
    public int Subtrair(int x, int y)
    {
        return x - y;
    }
}
```

h) Commit no Git desktop

Adicionar comentários XML aos métodos Somar() e Subtrair()



i) Adicionar método Converter Temperatura

```

internal class Aritmetica
{
    /// <summary>
    /// Tipo de conversão de temperatura a executar
    /// </summary>
    3 referências
    public enum ConversaoTemperatura
    {
        Nulo = 0,
        CelsiusFahrenheit,
        FahrenheitCelsius
    }

    /// <summary>
    /// Operação soma
    /// </summary>
    /// <returns>Retorna a soma de dois números</returns>
    0 referências
    public int Somar(int x, int y)
    {
        return x + y;
    }

    /// <summary>
    /// Operação subtração
    /// </summary>
    /// <returns>Retorna o resultado da subtração de dois números</returns>
    0 referências
    public int Subtrair(int x, int y)
    {
        return x - y;
    }

    0 referências
    public double ConverterTemperatura(ConversaoTemperatura conversao, double temperatura)
    {
        if (conversao == ConversaoTemperatura.CelsiusFahrenheit)
        {
            return (temperatura * 1.8000 + 32);
        } else if (conversao == ConversaoTemperatura.FahrenheitCelsius){
            return ((temperatura - 32) / 1.8000);
        }

        return -1;
    }
}
  
```

j) Adicionar commit

Adicionar método ConverterTemperatura()

k) Adicionar método Converter Temperatura

```
internal class Aritmetica
{
    /// <summary>
    /// Tipo de conversão de temperatura a executar
    /// </summary>
    3 referências
    public enum ConversaoTemperatura
    {
        Nulo = 0,
        CelsiusFahrenheit,
        FahrenheitCelsius
    }

    /// <summary>
    /// Operação soma
    /// </summary>
    /// <returns>Retorna a soma de dois números</returns>
    0 referências
    public int Somar(int x, int y)
    {
        return x + y;
    }

    /// <summary>
    /// Operação subtração
    /// </summary>
    /// <returns>Retorna o resultado da subtração de dois números</returns>
    0 referências
    public int Subtrair(int x, int y)
    {
        return x - y;
    }

    0 referências
    public double ConverterTemperatura(ConversaoTemperatura conversao, double temperatura)
    {
        if (conversao == ConversaoTemperatura.CelsiusFahrenheit)
        {
            return (temperatura * 1.8000 + 32);
        } else if (conversao == ConversaoTemperatura.FahrenheitCelsius){
            return ((temperatura - 32) / 1.8000);
        }

        return -1;
    }
}
```

l) Substituir **internal class Aritmetica** por **internal static class Aritmetica**

m) Métodos também passam a ser estáticos

```

internal static class Aritmetica
{
    /// <summary>
    /// Tipo de conversão de temperatura a executar
    /// </summary>
    3 referências
    public enum ConversaoTemperatura
    {
        Nulo = 0,
        CelsiusFahrenheit,
        FahrenheitCelsius
    }

    /// <summary>
    /// Operação soma
    /// </summary>
    /// <returns>Retorna a soma de dois números</returns>
    0 referências
    public static int Somar(int x, int y)
    {
        return x + y;
    }

    /// <summary>
    /// Operação subtração
    /// </summary>
    /// <returns>Retorna o resultado da subtração de dois números</returns>
    0 referências
    public static int Subtrair(int x, int y)
    {
        return x - y;
    }

    0 referências
    public static double ConverterTemperatura(ConversaoTemperatura conversao, double temperatura)
    {
        if (conversao == ConversaoTemperatura.CelsiusFahrenheit)
        {
            return (temperatura * 1.8000 + 32);
        } else if (conversao == ConversaoTemperatura.FahrenheitCelsius){
            return ((temperatura - 32) / 1.8000);
        }

        return -1;
    }
}

```

n) Adicionar commit

Tornar a classe Aritmetica numa classe estática

o) Escrever as linhas para efetuar os cálculos

```

namespace Calculos
{
    //referências
    internal class Program
    {
        //referências
        static void Main(string[] args)
        {
            Console.WriteLine("Aplicação Cálculos");

            Console.WriteLine($"4 + 2 = {Aritmetica.Somar(4,2)}");
            Console.WriteLine($"4 - 2 = {Aritmetica.Subtrair(4, 2)}");

            Console.ReadKey();
        }
    }
}

```

- p) **Criar uma classe** para a **conversão da temperatura**. Retirar o código da classe **Aritmetica** para a classe **Conversoes**

```

namespace Calculos
{
    /// <summary>
    /// Implementa operações de conversão
    /// </summary>
    //referências
    internal static class Conversoes
    {
        /// <summary>
        /// Tipo de conversão de temperatura a executar
        /// </summary>
        //referências
        public enum ConversaoTemperatura
        {
            Nulo = 0,
            CelsiusFahrenheit,
            FahrenheitCelsius
        }

        //referências
        public static double ConverterTemperatura(ConversaoTemperatura conversao, double temperatura)
        {
            if (conversao == ConversaoTemperatura.CelsiusFahrenheit)
            {
                return (temperatura * 1.8000 + 32);
            }
            else if (conversao == ConversaoTemperatura.FahrenheitCelsius)
            {
                return ((temperatura - 32) / 1.8000);
            }

            return -1;
        }
    }
}

```

- q) Adicionar comentários

```

namespace Calculos
{
    /// <summary>
    /// Implementa operações de conversão
    /// </summary>
    0 referências
    internal static class Conversoes
    {
        /// <summary>
        /// Tipo de conversão de temperatura a executar
        /// </summary>
        3 referências
        public enum ConversaoTemperatura
        {
            Nulo = 0,
            CelsiusFahrenheit,
            FahrenheitCelsius
        }

        /// <summary>
        /// Conversão de temperaturas
        /// </summary>
        /// <param name="conversao">a conversão a efetuar</param>
        /// <param name="temperatura">a temperatura a converter</param>
        /// <returns>Retorna o resultado da conversão</returns>
        0 referências
        public static double ConverterTemperatura(ConversaoTemperatura conversao, double temperatura)
        {
            if (conversao == ConversaoTemperatura.CelsiusFahrenheit)
            {
                return (temperatura * 1.8000 + 32);
            }
            else if (conversao == ConversaoTemperatura.FahrenheitCelsius)
            {
                return ((temperatura - 32) / 1.8000);
            }

            return -1;
        }
    }
}

```

r) Adicionar a mensagem de commit

Criar a classe Conversoes e mover método

s) Para alterar o **commit** (apenas o último **commit** efetuado) No separador **History** clicar com o botão direito do rato sobre o último **commit** e no submenu aberto clicar em **Amend commit...**

t) Colocar o texto seguinte na **descrição** do **commit**:

O método ConverterTemperatura() foi removido da classe Aritmetica e colocado na classe Conversoes uma vez que implementa uma operação que não se adequa à classe.

u) Alterar o ficheiro Calculos


```

namespace Calculos
{
    // Referências
    internal class Program
    {
        // Referências
        static void Main(string[] args)
        {
            Console.WriteLine("Aplicação Cálculos");

            Console.WriteLine($"4 + 2 = {Aritmetica.Somar(4,2)}");
            Console.WriteLine($"4 - 2 = {Aritmetica.Subtrair(4, 2)}");

            double t1 = Conversoes.ConverterTemperatura(Conversoes.ConversaoTemperatura.CelsiusFahrenheit, 36);
            double t2 = Conversoes.ConverterTemperatura(Conversoes.ConversaoTemperatura.FahrenheitCelsius, 100);

            Console.WriteLine($"36º Celsius = {t1}º Fahrenheit");
            Console.WriteLine($"100º Fahrenheit = {t2}º Celsius");
            Console.ReadKey();
        }
    }
}

```

v) Acrescentar um **commit** com o texto **Acrescentar código para testar**



https://imgs.xkcd.com/comics/git_commit.png

Algumas considerações:

- 📁 **commit early, commit often**
- 📁 Não é possível alterar o código do **commit**
- 📁 Não podemos alterar a mensagem de **commits** antigos apenas o último **commit**.

Nota: para alterar o último **commit** na linha de comandos usar o seguinte código:

git commit -- amend -m "nova mensagem do commit"

Recomendações para mensagens de *commit*

- 📖 Devem ser informativas mas sucintas
- 📖 Deve ser possível perceber rapidamente quais as alterações efetuadas
- 📖 Devem ter um título e, opcionalmente, uma descrição
- 📖 O título e a descrição devem ser separados por uma linha em branco
- 📖 O título não deve acabar com ponto de final

Recomendações para mensagens de *commit*

(em formato de mensagem de *commit*)

- 📖 O título não deve exceder 50 caracteres
- 📖 A descrição não é obrigatória mas se existir deve oferecer uma explicação mais detalhada acerca do *commit*.
 - 📖 Separar o título da descrição por uma linha em branco.
 - 📖 O corpo da descrição deve conter um máximo de 72 caracteres por linha.
- Podem ser usados *bullet points*
- Explicar o **porquê** do código e **não o como**.

w) Alterar o código das classes

```
7 namespace Calculos
8 {
9     /// <summary>
10    /// Implementa operações aritméticas
11    /// </summary>
12    2 referências
13    internal static class Aritmetica
14    {
15        /// <summary>
16        /// Operação soma
17        /// </summary>
18        /// <returns>Retorna a soma de dois números</returns>
19        1 referência
20        public static int Somar(int x, int y)
21        {
22            return x + y;
23        }
24        /// <summary>
25        /// Operação subtração
26        /// </summary>
27        /// <returns>Retorna o resultado da subtração de dois números</returns>
28        1 referência
29        public static int Subtrair(int x, int y)
30        {
31            return x - y;
32        }
33        /// <summary>
34        /// Operação multiplicação
35        /// </summary>
36        /// <returns>Retorna o resultado da multiplicação de dois números</returns>
37        0 referências
38        public static int Multiplicar(int x, int y)
39        {
40            return x * y;
41        }
42    }
```

```

namespace Calculos
{
    /// <summary>
    /// Implementa operações de conversão
    /// </summary>
    internal static class Conversoes
    {
        /// <summary>
        /// Tipo de conversão de temperatura a executar
        /// </summary>
        public enum ConversaoTemperatura
        {
            Nulo = 0,
            CelsiusFahrenheit,
            FahrenheitCelsius
        }

        /// <summary>
        /// Tipo de conversão de distâncias
        /// </summary>
        public enum ConversaoDistancia
        {
            Nulo = 0,
            MetrosMilhas,
            MilhasMetros
        }

        /// <summary>
        /// Conversão de temperaturas
        /// </summary>
        /// <param name="conversao">a conversão a efetuar</param>
        /// <param name="temperatura">a temperatura a converter</param>
        /// <returns>Retorna o resultado da conversão da temperatura</returns>

        public static double ConverterTemperatura(ConversaoTemperatura conversao, double
temperatura)
        {
            if (conversao == ConversaoTemperatura.CelsiusFahrenheit)
            {
                return (temperatura * 1.8000 + 32);
            }
            else if (conversao == ConversaoTemperatura.FahrenheitCelsius)
            {
                return ((temperatura - 32) / 1.8000);
            }

            return -1;
        }

        /// <summary>
        /// Conversão de Distâncias
        /// </summary>
        /// <param name="conversao">a conversão a efetuar</param>
        /// <param name="temperatura">a distância a converter</param>
        /// <returns>Retorna o resultado da conversão da distância</returns>

        public static double ConverterDistancias(ConversaoDistancia conversao, double
distancia)
        {
            if (conversao == ConversaoDistancia.MetrosMilhas)
            {
                return distancia * 0.0006213712;
            }
            else if (conversao == ConversaoDistancia.MilhasMetros)
            {
                return distancia * 1609.344;
            }
        }
    }
}

```

```

    }
    return -1;
}
}

```

x) No **git desktop** fazer um *commit* seletivo para isso

- seleccionar apenas o ficheiro **Aritmetica.cs** e escrever o commit “**Adicionar o método Multiplicar()**”; clicar em **commit to main**
- seleccionar apenas o ficheiro **Conversoes.cs** e escrever o commit “**Adicionar o método ConverterDistancias()**”; clicar em **commit to main**

6. Editar o ficheiro README.md

Linguagem Markdown – desenvolvida por: John Gruber e Aaron Swartz

| opção | descrição |
|---------------|-------------------|
| # texto | headers (títulos) |
| **texto** | bold |
| *texto* | itálico |
| > texto | citação |
| [texto](url) | hiperligação |
| ![texto](url) | imagem |

```

1  # Cálculos :1234 📄 :
2
3  Aplicação **C#** para efetuar diversos tipos de cálculos.
4
5  ![Aplicação Cálculos](aplicacao-calculos.png)
6
7  Desenvolvida no âmbito da ação de formação **Introdução
  ao * __Git__ e * __GitHub__.
8
9  ## Operações suportadas :arrow_down_small 📄 :
10
11  Neste momento esta aplicação implementa as seguintes
  operações:
12  - soma
13  - subtração
14  - conversão de temperaturas
15  |   - celsius ➡ fahrenheit
16  |   - fahrenheit ➡ celsius
17  - conversão de distâncias
18  |   - metros ➡ milhas
19  |   - milhas ➡ metros
20
21  ## Tecnologias utilizadas neste projeto 📄
22
23  - Visual Studio
24  - C#
25  - Git
26  - GitHub Desktop
27  - Plataforma GitHub
28
29  ## Site Oficial
30
31  Faça download desta aplicação no site oficial: [em
  construção](#)

```

7. Publicar repositório

