

Git (Windows): guia para armazenamento e gestão de código fonte.

Jonathan Vinícius Suter

1 O que é o *Git* e para que serve?

É uma ferramenta de cliente que permite armazenar, centralizar e gerir arquivos, sejam eles códigos de aplicações ou não.

2 Como utilizar?

2.1 Baixar o *Git Client*:

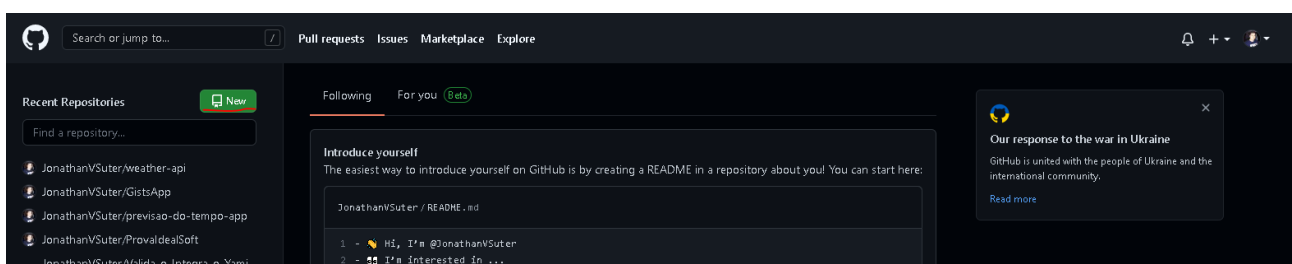
Acessar o link abaixo e baixar o arquivo ([clique aqui](#)) na opção “*Standalone Installer*” (baixar de acordo com a versão do SO). Basta seguir os passos indo até o fim da instalação, sem quaisquer personalizações.

2.2 Criar um repositório (GitHub)

O repositório será criado no GitHub, porém, poderia estar no Azure Devops ou outro serviço de armazenamento de código.

Deverá ser criada uma conta, caso não possua nenhuma. Com a conta criada, a criação do repositório é simples:

- Na tela inicial, clicar em *New*:




- Dar um nome ao repositório:

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner * Repository name *

 JonathanVSuter ▾ /

Great repository names are short and memorable. Need inspiration? How about [sturdy-octo-journey?](#)


Description (optional)

 ☐ Public

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner * Repository name *

 JonathanVSuter ▾ / ✓

Great repository names are short teste is available. Need inspiration? How about [sturdy-octo-journey?](#)

Description (optional)

E então, *Create Repository*:

The screenshot shows the 'Create repository' form on GitHub. It has a dark theme. At the top, there are two radio buttons: 'Public' (selected) and 'Private'. Below them, there's a section 'Initialize this repository with:' with a note to skip if importing an existing repository. There are three checkboxes: 'Add a README file' (checked), 'Add .gitignore' (unchecked), and 'Choose a license' (unchecked). Each checkbox has a description and a 'Learn more' link. The '.gitignore' dropdown is set to 'None'. The 'License' dropdown is also set to 'None'. A blue information box states: 'You are creating a public repository in your personal account.' At the bottom, there is a large green 'Create repository' button.

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

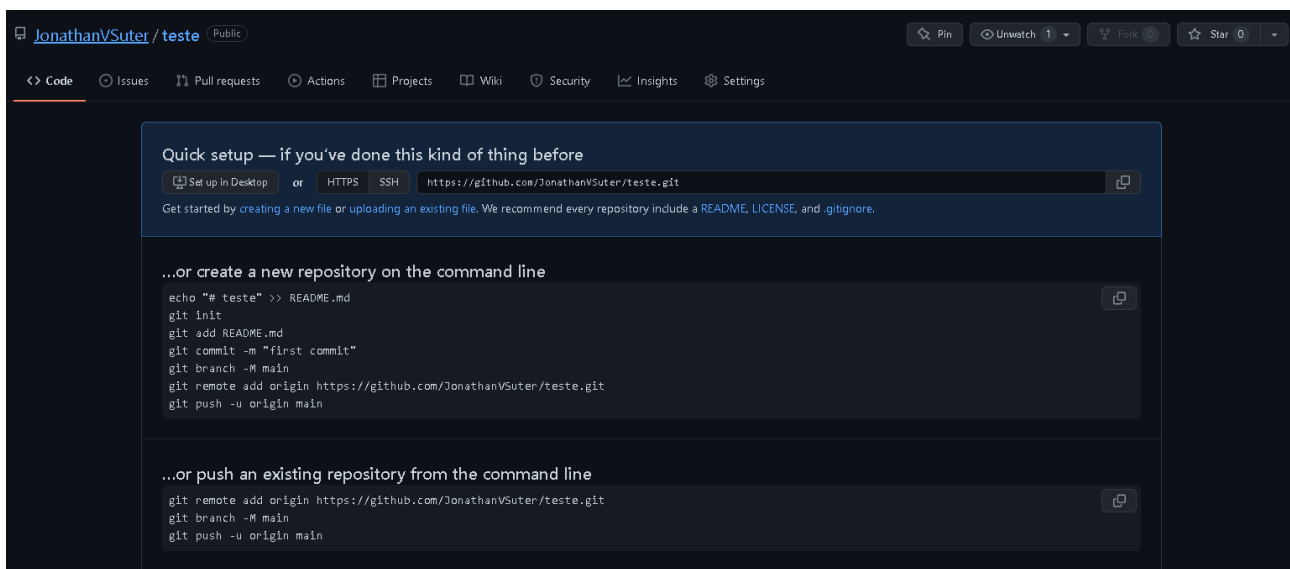
☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

You are creating a public repository in your personal account.

Create repository

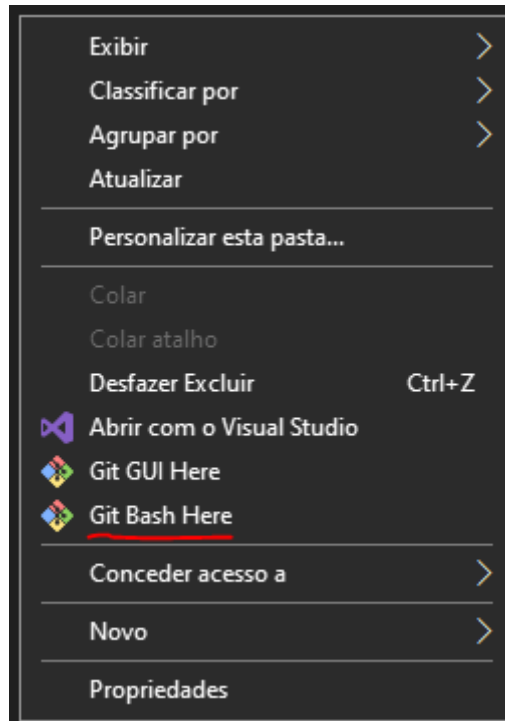
Repositório criado:



Após isso, será possível enviar fontes ao GitHub.

2.3 Adicionando o repositório local ao repositório remoto

Para subir fontes ao Git, deverá abrir a pasta onde está o diretório raiz do código, clicar com o botão direito do mouse e em *Git Bash Here*:



No console que abrir, digitar *git init*, ele inicializará um repositório a partir do diretório atual.

```
MINGW64:/c:/Users/jonat/source/repos/TesteGithub
jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (main)
$ git init
Initialized empty Git repository in C:/Users/jonat/source/repos/TesteGithub/.git
/
jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ |
```

OBS: esse comando só precisa ser executado quando se inicializará o repositório. Uma vez executado no diretório, não precisa ser feito novamente.

Depois, digitar `touch .gitignore`, para adicionar o arquivo `gitignore` (ele estará vazio), que guarda uma série de regras sobre os arquivos que devem ser ignorados.

Após criar o arquivo, abrir o mesmo com algum editor de textos e copiar o conteúdo [deste](#) arquivo. Assim, os arquivos gerados pelo Visual Studio 2019 não entram nas modificações e nem irão para o repositório.

Arquivo criado:

Nome	Data de modificação	tipo
.git	17/04/2022 15:16	Pasta
TesteGithub	17/04/2022 14:53	Pasta
.gitignore	17/04/2022 15:17	Docu
TesteGithub.sln	17/04/2022 14:53	Visua

Arquivo preenchido:

```
.gitignore - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
# Fody - auto-generated XML schema
FodyWeavers.xsd

# VS Code files for those working on multiple tools
.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json
*.code-workspace

# Local History for Visual Studio Code
.history/

# Windows Installer files from build outputs
*.cab
*.msi
*.msix
*.msm
*.msp

# JetBrains Rider
*.sln.iml

Ln 388, Col 10 100% Windows (CRLF) UTF-8
```

Novamente no console, digitar o comando *git add .* (com o ponto no final) para adicionar todos os arquivos do diretório à lista de arquivos para o *commit*.

```
MINGW64:/c/Users/jonat/source/repos/TesteGithub
$ touch .gitignore

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ AC

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git commit -m "commit inicial"
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        TesteGithub.sln
        TesteGithub/

nothing added to commit but untracked files present (use "git add" to track)

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git add .

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ |
```

OBS: se executar o comando “*git commit*” antes de adicionar os arquivos, o console apontará que há arquivos ainda não rastreados, como na mensagem acima.

Após, basta executar o comando *git commit -m “Commit Inicial”* para adicionar o primeiro *commit*. O *git* listará os arquivos adicionados.

```
MINGW64:/c/Users/jonat/source/repos/TesteGithub
Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        TesteGithub.sln
        TesteGithub/

nothing added to commit but untracked files present (use "git add" to track)

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git add .

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git commit -m "commit inicial"
[master (root-commit) a8b1487] commit inicial
4 files changed, 443 insertions(+)
create mode 100644 .gitignore
create mode 100644 TesteGithub.sln
create mode 100644 TesteGithub/Program.cs
create mode 100644 TesteGithub/TesteGithub.csproj

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ |
```

Inserir um novo comando `git branch -M master`, onde “master” é a *branch*.

Então, o comando `git remote add origin` (endereço do repositório) indicará onde os arquivos serão inseridos e em seguida, `git push -u origin master`, em que a master é a branch que aparecerá no GitHub.

```
MINGW64:/c:/Users/jonat/source/repos/TesteGithub
create mode 100644 TesteGithub.sln
create mode 100644 TesteGithub/Program.cs
create mode 100644 TesteGithub/TesteGithub.csproj

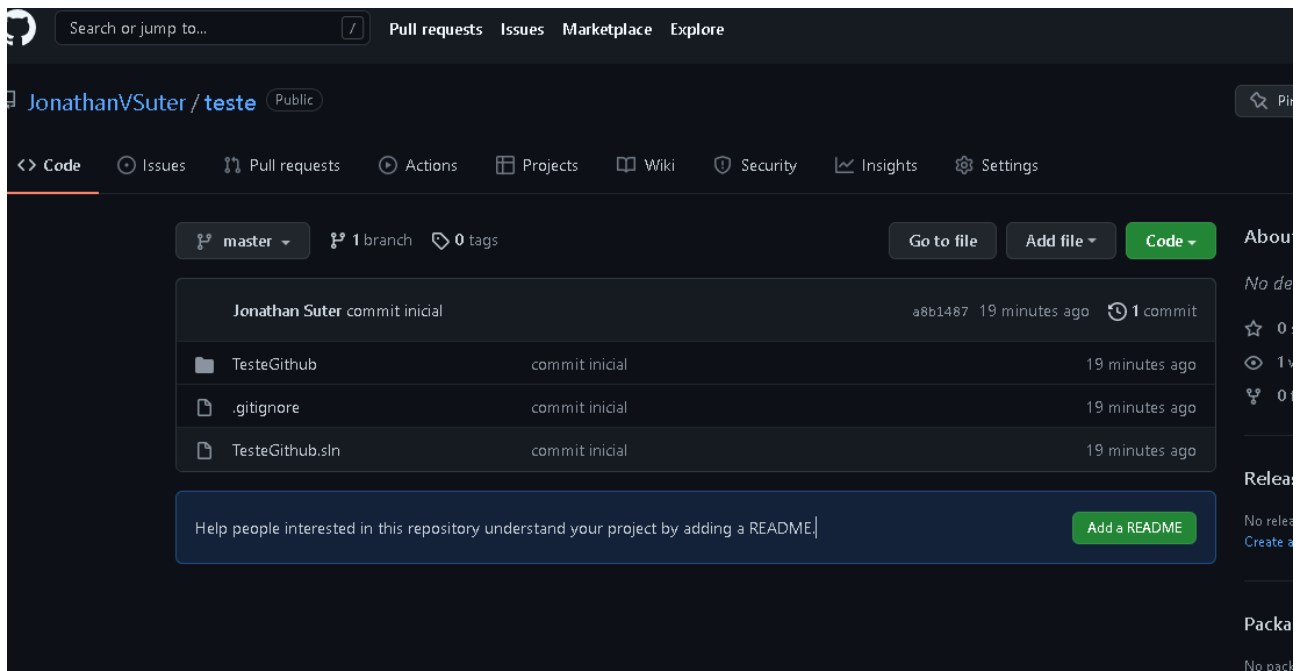
jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git branch -M master

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git remote add origin https://github.com/JonathanVSuter/teste.git

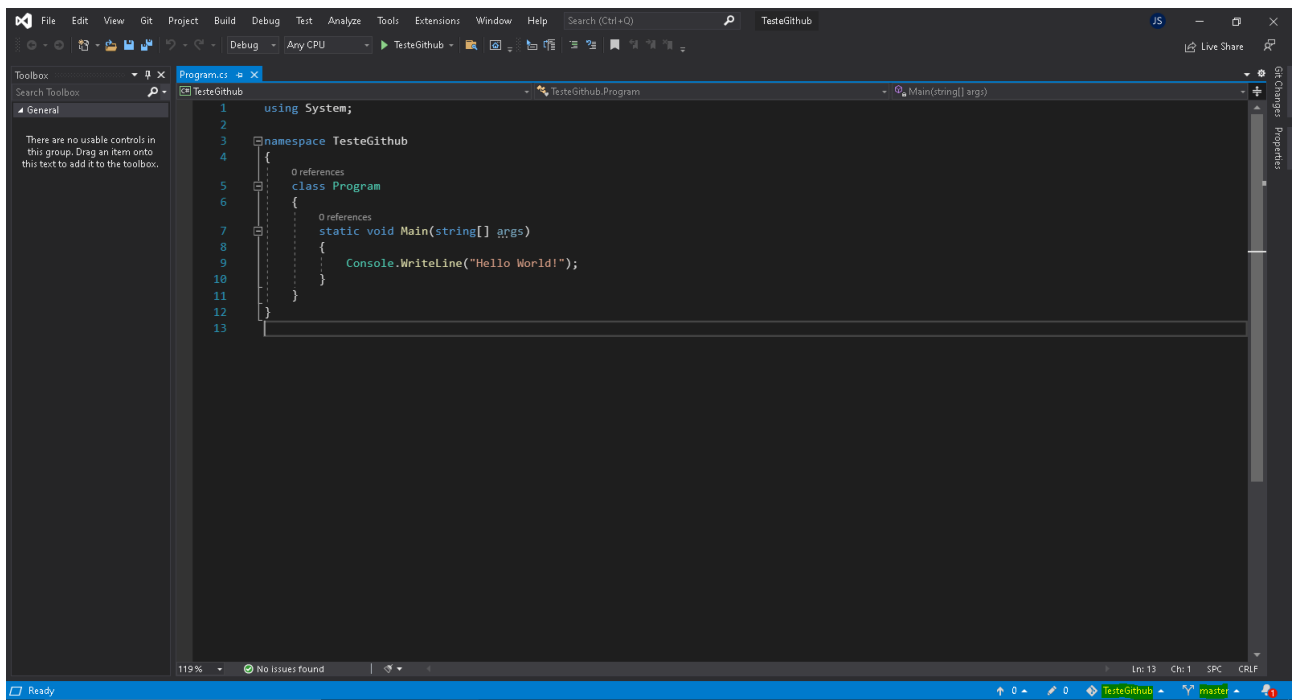
jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 3.95 KiB | 1.98 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JonathanVSuter/teste.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (master)
$ |
```

Desta forma, o repositório aparecerá no GitHub, conforme abaixo:



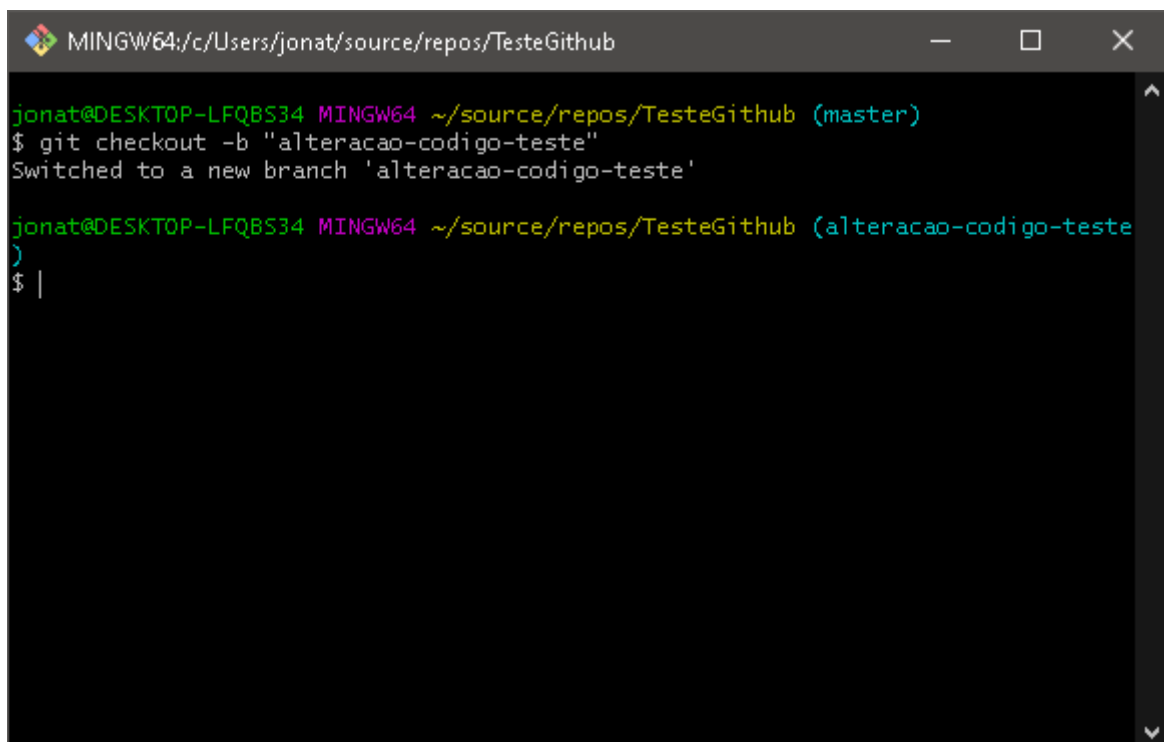
Ao abrir a solução no Visual Studio 2019, no canto inferior direito aparecerá a descrição da *branch* e o nome do repositório:

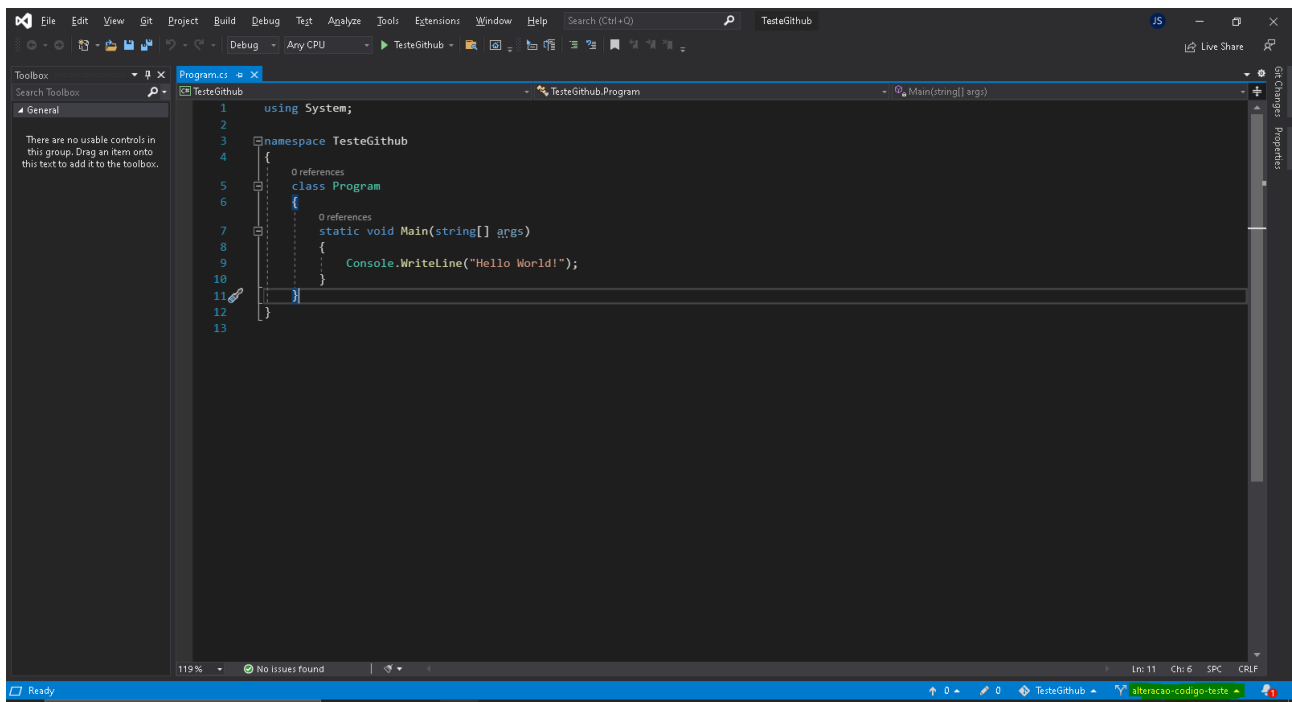


A partir daqui é possível usar apenas o Visual Studio para criar *branch*'s. Mas caso necessário, é possível criar *branch*'s e fazer *commit*'s usando o *Git Client*.

Para criar uma branch, o *Client* deve estar aberto no diretório do repositório. Inserir o comando `git checkout -b nome-da-branch-sem-caracteres-especiais-e-ou-espacos`.

Então o *Client* alternará para esta *branch*, assim como o próprio Visual Studio:

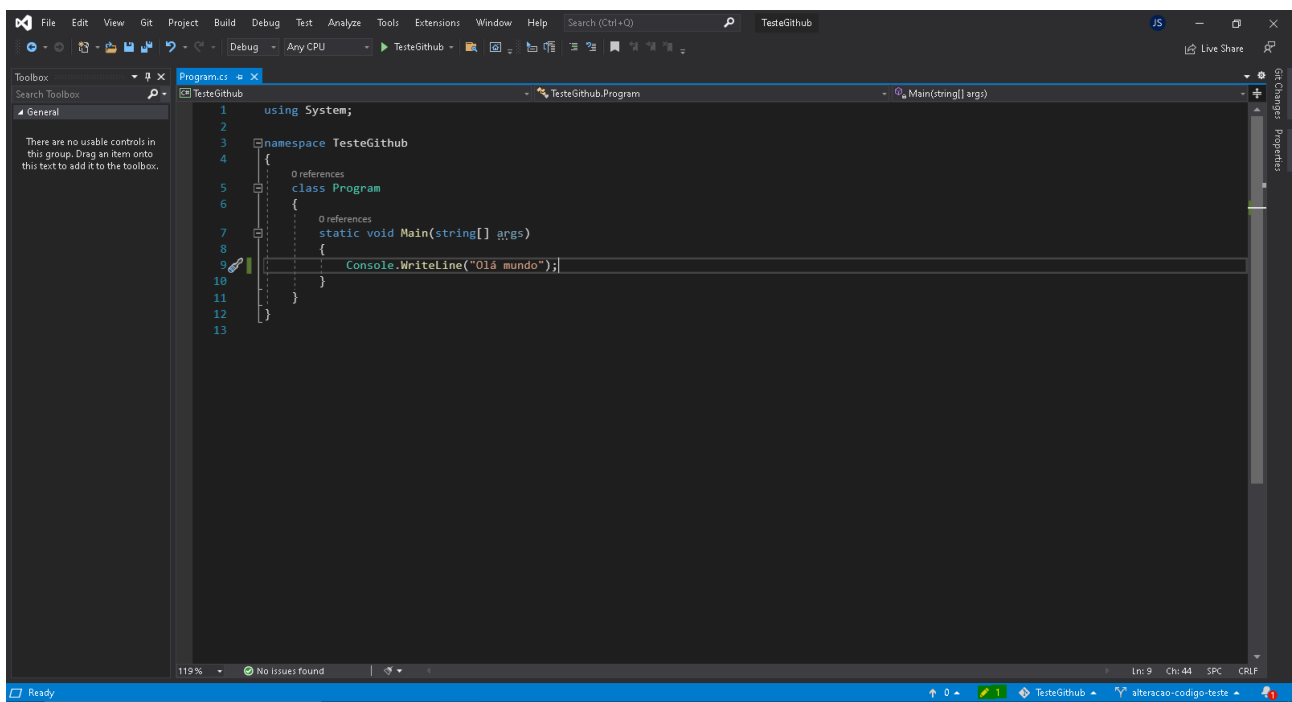


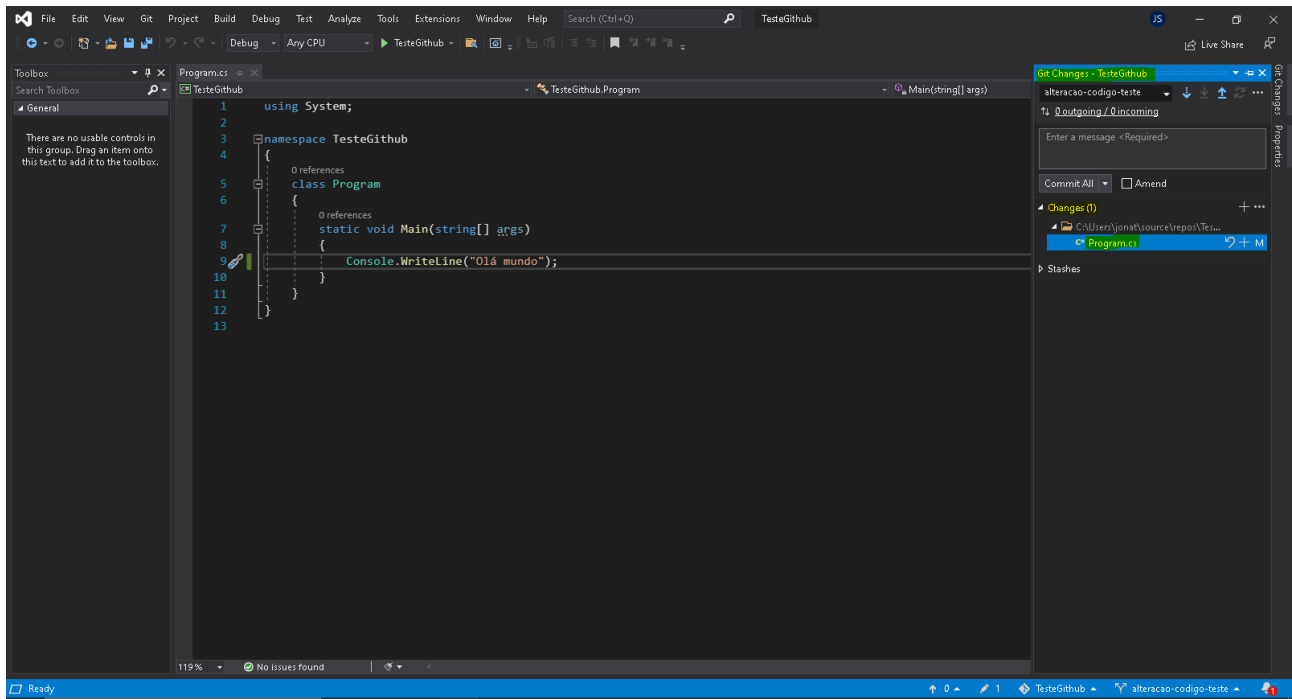


OBS: caso haja necessidade ou vontade de utilizar uma branch diferente como base, antes de executar o comando *Checkout*, executar o comando *git switch nome-da-branch*.

2.4 Enviando alterações no código e criando *Pull Requests*

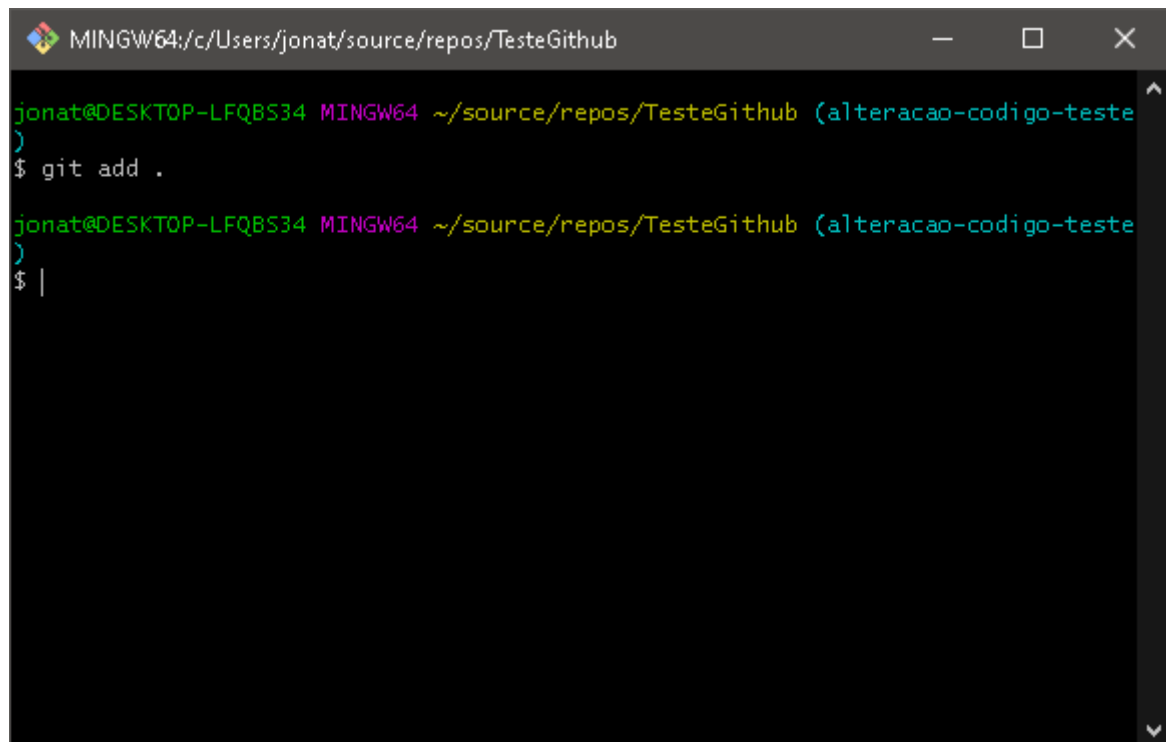
Com o código no Visual Studio, fazendo quaisquer modificações, o próprio mostrará as modificações feitas e onde foram feitas:



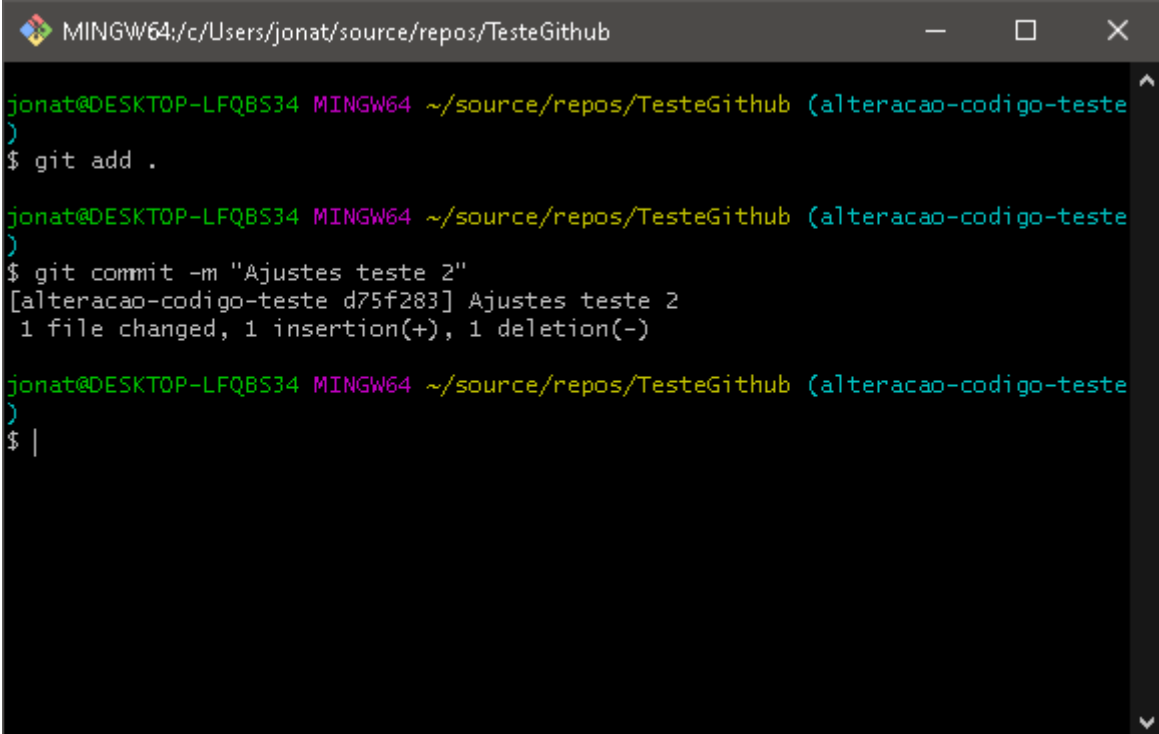


Será feito o envio ao repositório da seguinte maneira:

Após a modificação no código, deve-se acessar o *Git Client* e com a branch que possui a modificação selecionada (nesse caso, *alteracao-codigo-teste*), inserir o comando “*git add .*” para adicionar todos os arquivos modificados ao *commit*:



Então será possível fazer o *commit*, com o comando *git commit -m "Descrição do commit"* :

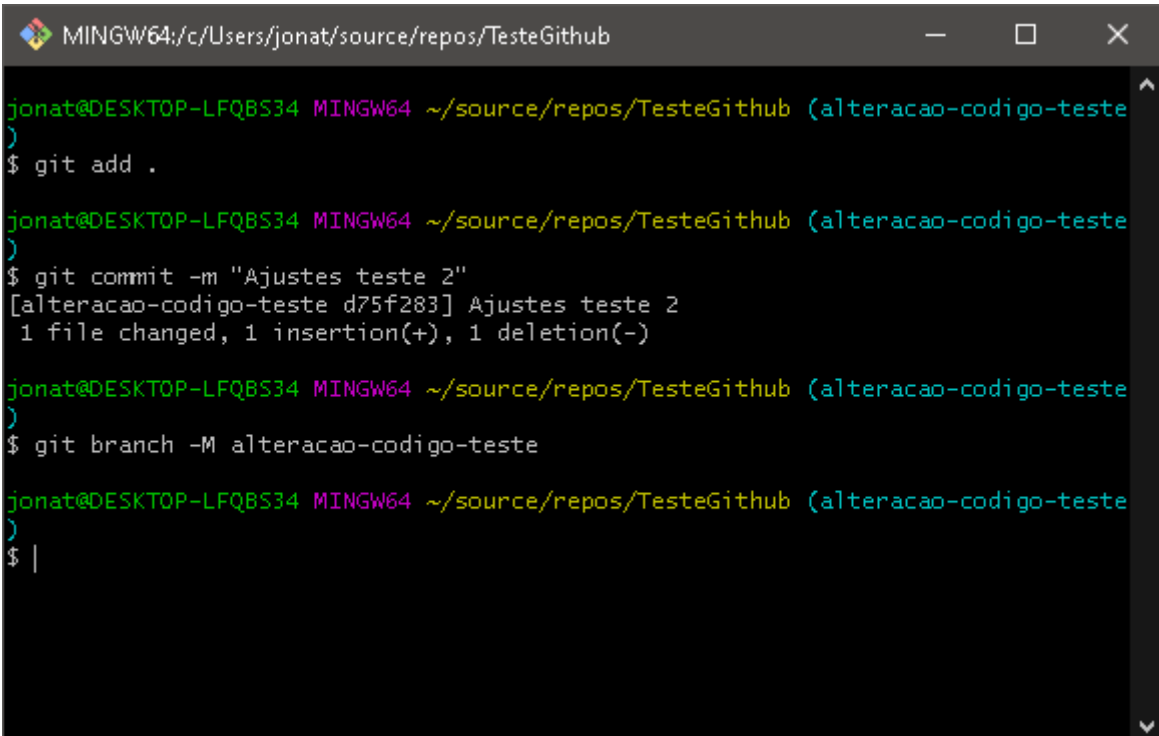
A terminal window titled 'MINGW64:/c/Users/jonat/source/repos/TesteGithub' with standard window controls. The prompt is 'jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)'. The user enters '\$ git add .' followed by '\$ git commit -m "Ajustes teste 2"'. The output shows the commit hash '[alteracao-codigo-teste d75f283]', the commit message 'Ajustes teste 2', and a summary '1 file changed, 1 insertion(+), 1 deletion(-)'. The prompt returns to '\$ |' with a cursor.

```
jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git add .

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git commit -m "Ajustes teste 2"
[alteracao-codigo-teste d75f283] Ajustes teste 2
1 file changed, 1 insertion(+), 1 deletion(-)

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ |
```

Para subir a *branch* para o *origin*, deve-se usar o comando *git branch -M nome-da-branch*.

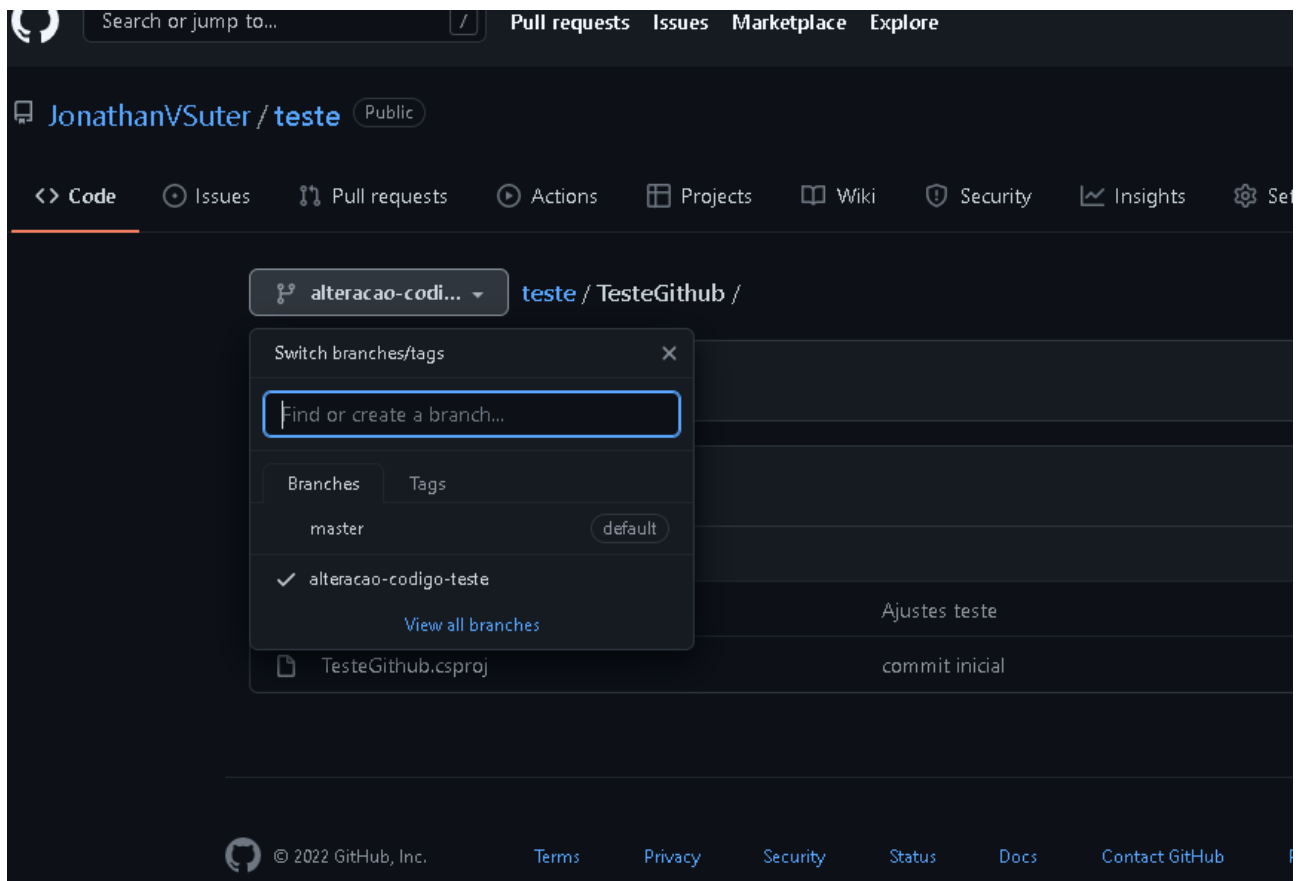
A terminal window titled 'MINGW64:/c/Users/jonat/source/repos/TesteGithub' with standard window controls. The prompt is 'jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)'. The user enters '\$ git add .' followed by '\$ git commit -m "Ajustes teste 2"'. The output shows the commit hash '[alteracao-codigo-teste d75f283]', the commit message 'Ajustes teste 2', and a summary '1 file changed, 1 insertion(+), 1 deletion(-)'. Then the user enters '\$ git branch -M alteracao-codigo-teste'. The prompt returns to 'jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)' with a cursor.

```
jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git add .

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git commit -m "Ajustes teste 2"
[alteracao-codigo-teste d75f283] Ajustes teste 2
1 file changed, 1 insertion(+), 1 deletion(-)

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git branch -M alteracao-codigo-teste

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ |
```



Assim, a nova *branch* ficará disponível no repositório remoto.

Para subir as modificações, usar o comando `git push -u origin nome-da-branch`:

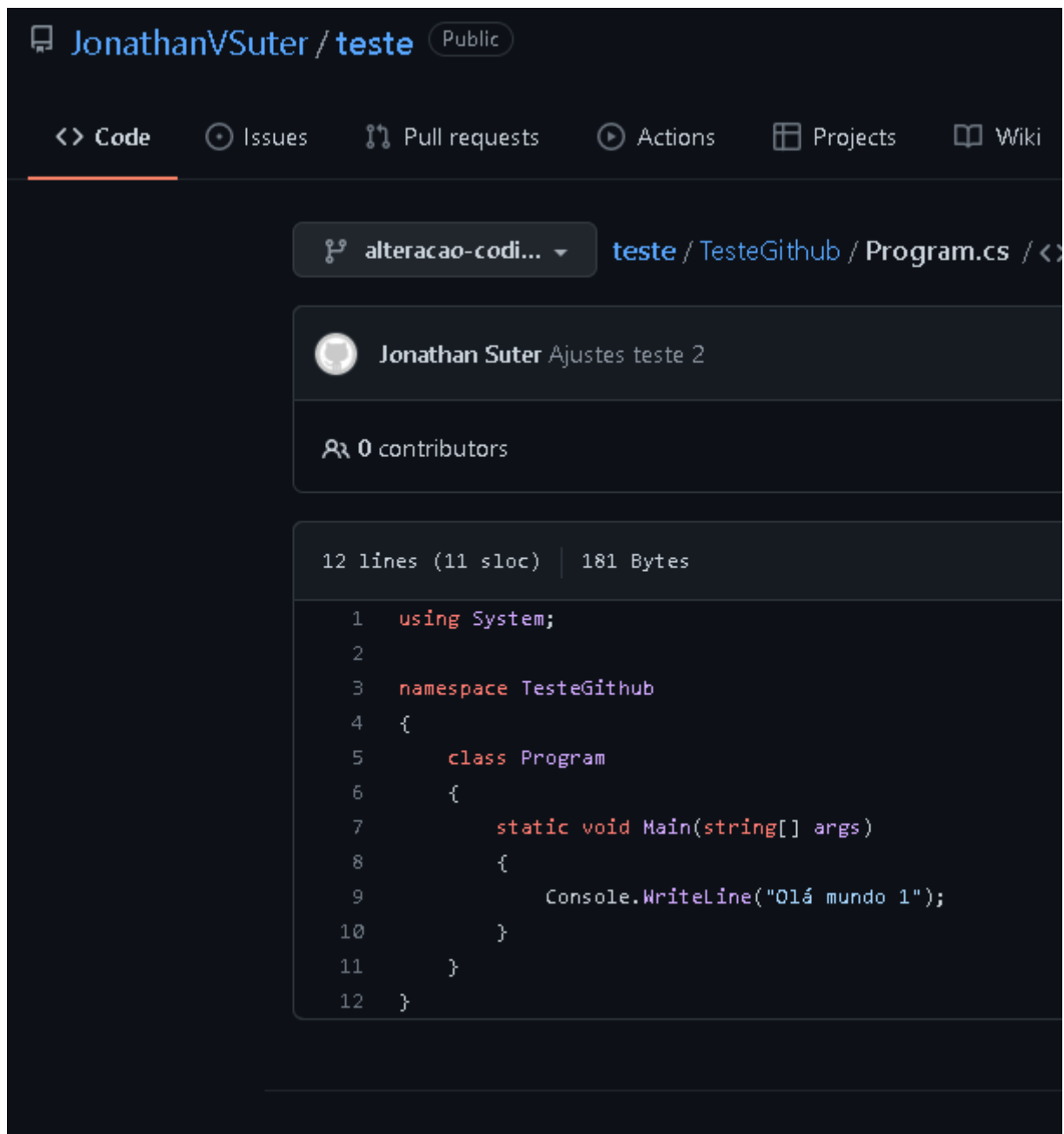
```
MINGW64/c/Users/jonat/source/repos/TesteGithub
[alteracao-codigo-teste d75f283] Ajustes teste 2
1 file changed, 1 insertion(+), 1 deletion(-)

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git branch -M alteracao-codigo-teste

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ git push -u origin alteracao-codigo-teste
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 395 bytes | 395.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/JonathanVSuter/teste.git
   c4e991d..d75f283  alteracao-codigo-teste -> alteracao-codigo-teste
branch 'alteracao-codigo-teste' set up to track 'origin/alteracao-codigo-teste'.

jonat@DESKTOP-LFQBS34 MINGW64 ~/source/repos/TesteGithub (alteracao-codigo-teste)
$ |
```

Caso haja sucesso na operação, o código ficará disponível no GitHub, selecionando a *branch* alterada:



JonathanVSuter / teste Public

<> Code Issues Pull requests Actions Projects Wiki

alteracao-codi... teste / TesteGithub / Program.cs / <>

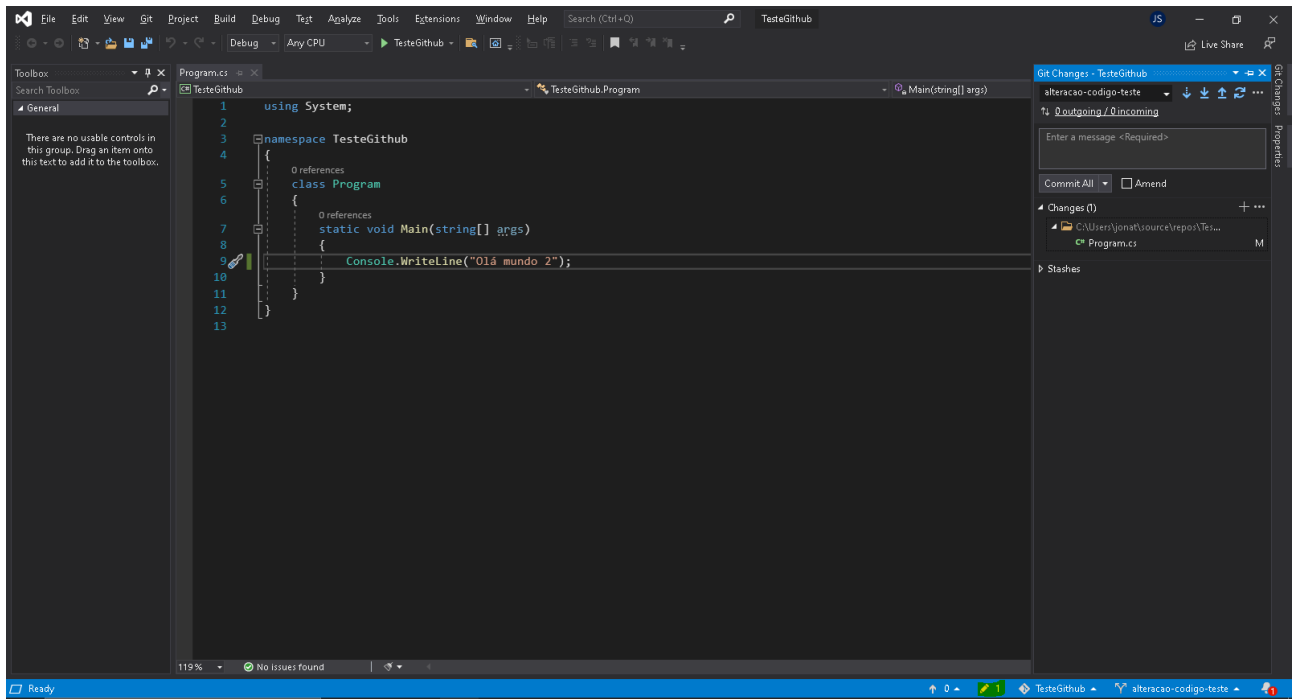
Jonathan Suter Ajustes teste 2

0 contributors

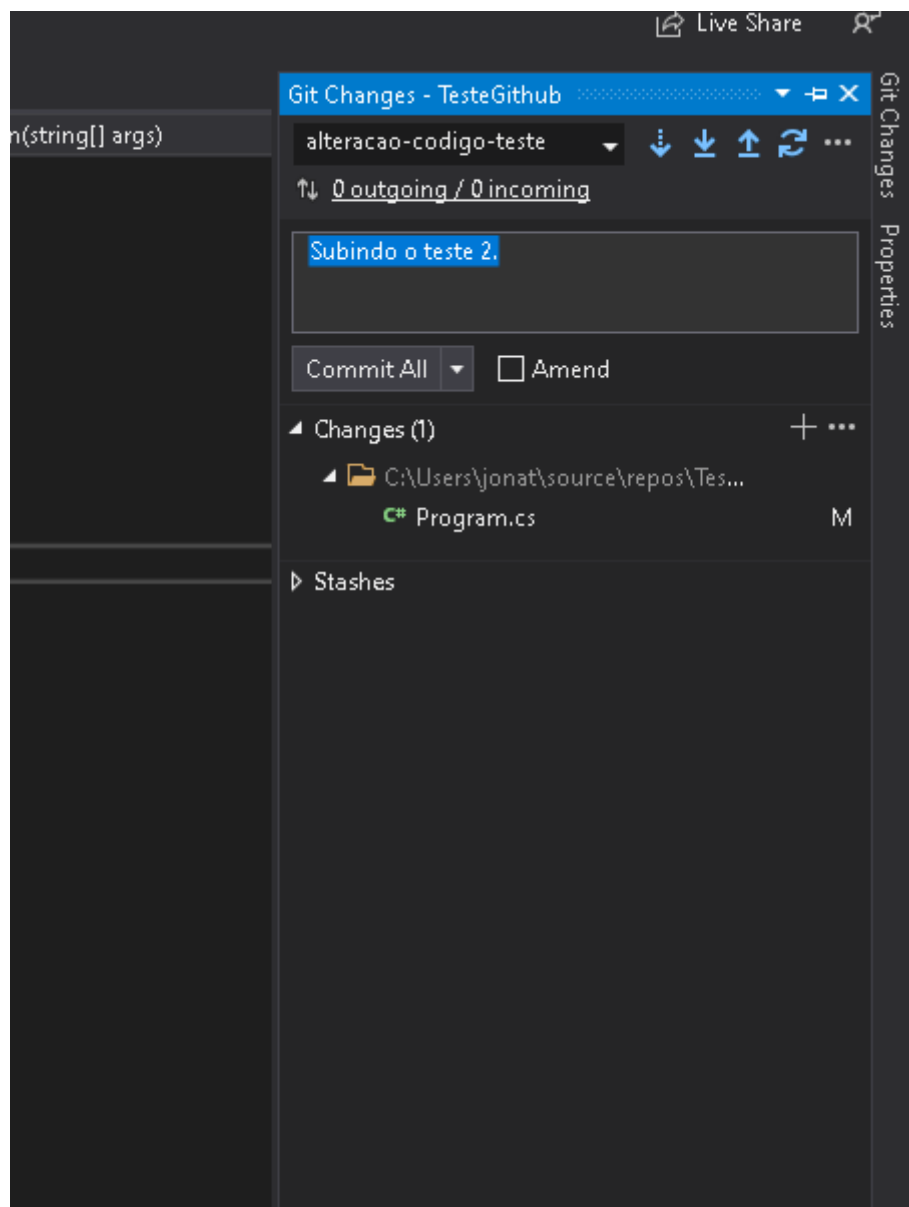
12 lines (11 sloc) | 181 Bytes

```
1 using System;
2
3 namespace TesteGithub
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             Console.WriteLine("Olá mundo 1");
10        }
11    }
12 }
```

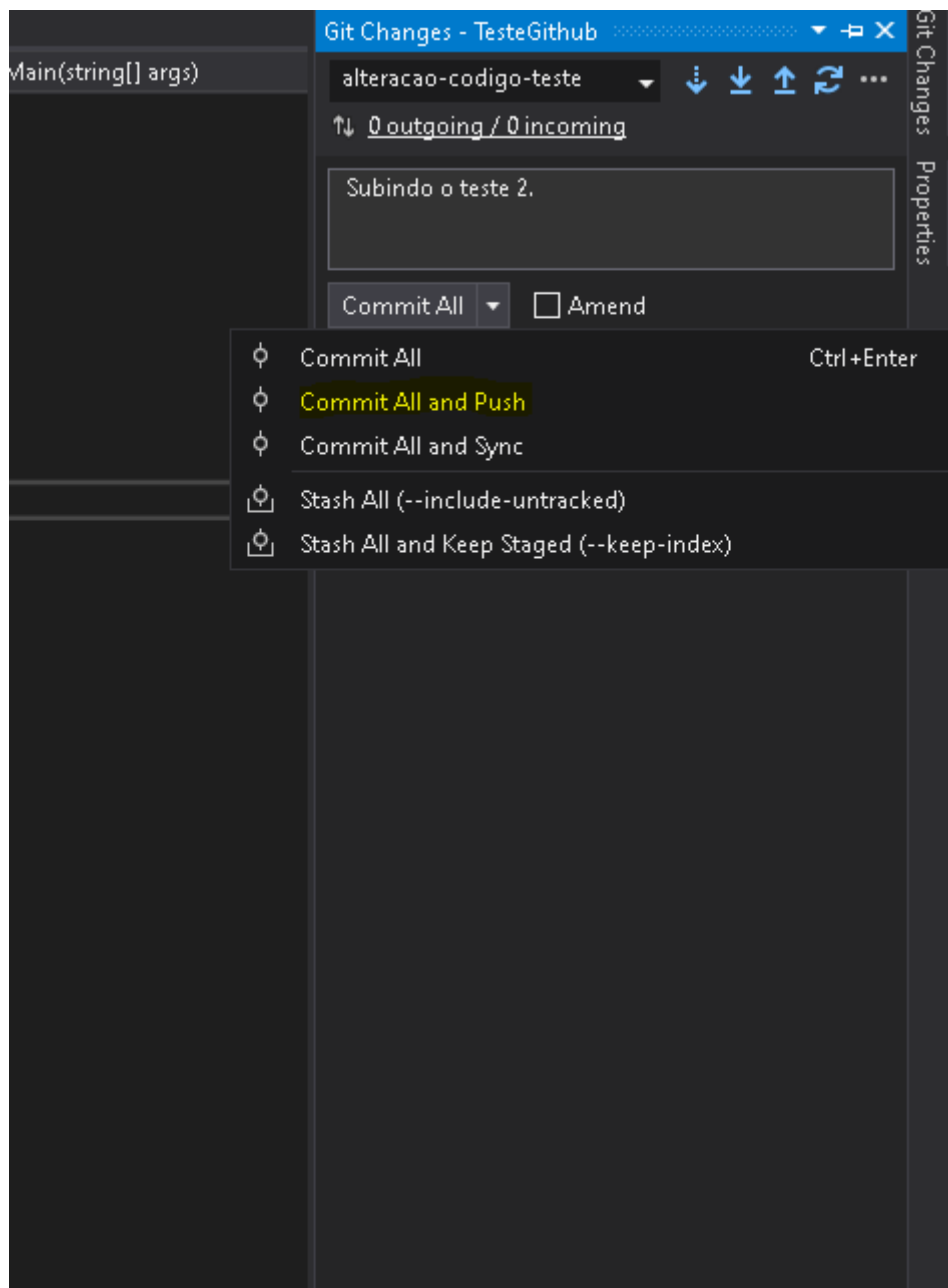
Para subir o código via Visual Studio, basta acessar o menu de alterações:



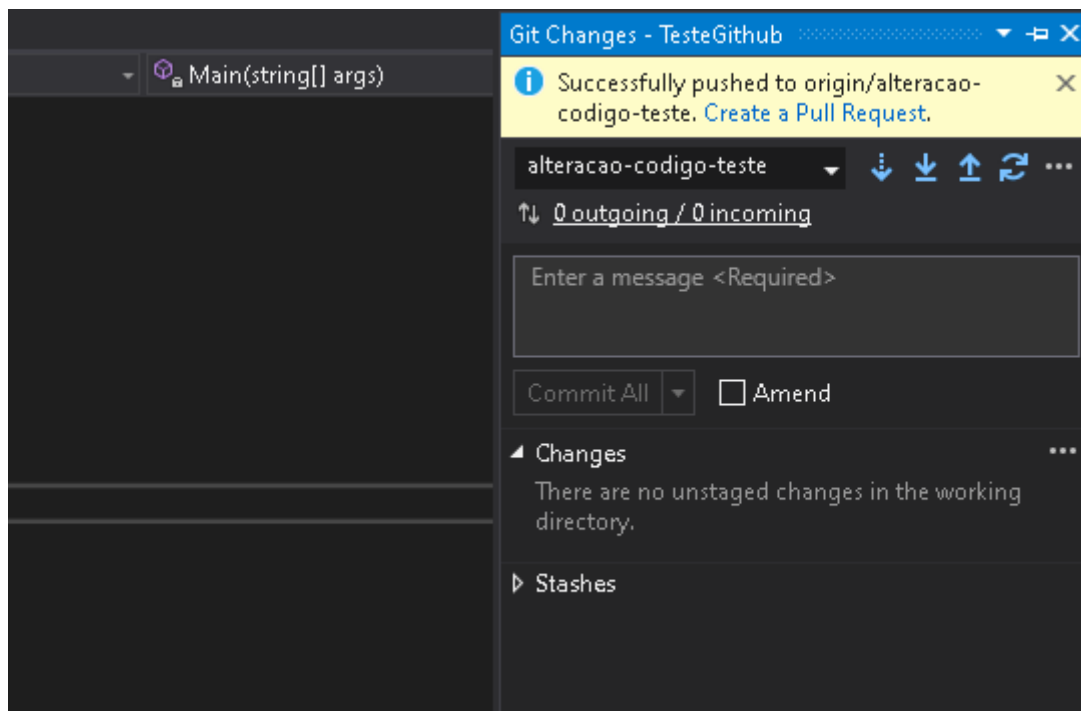
Insira a mensagem do *commit*:



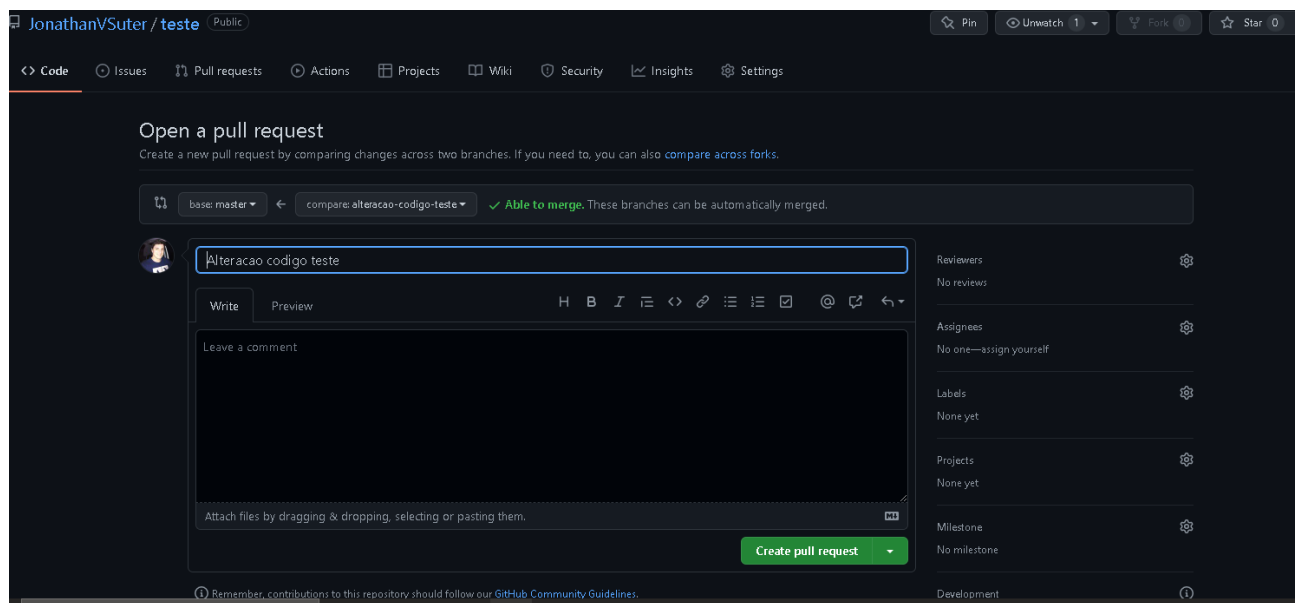
Selecionar a opção *Commit All and Push* e então o seu código será enviado, caso não hajam conflitos.



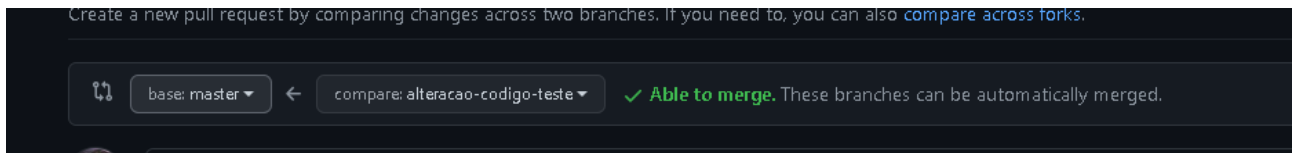
Quando o código for enviado, o Visual Studio sugerirá criar uma solicitação de *Pull Request* (*Create a Pull Request*).



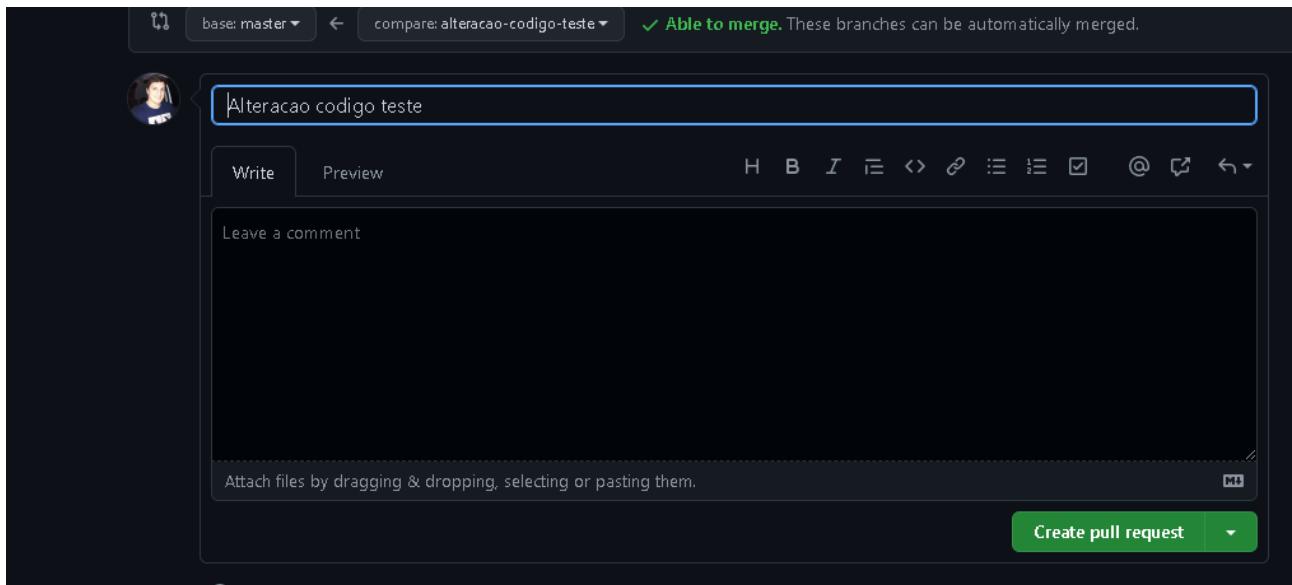
Acessando o *link* para criação do *Pull Request*, abrirá a seguinte página:



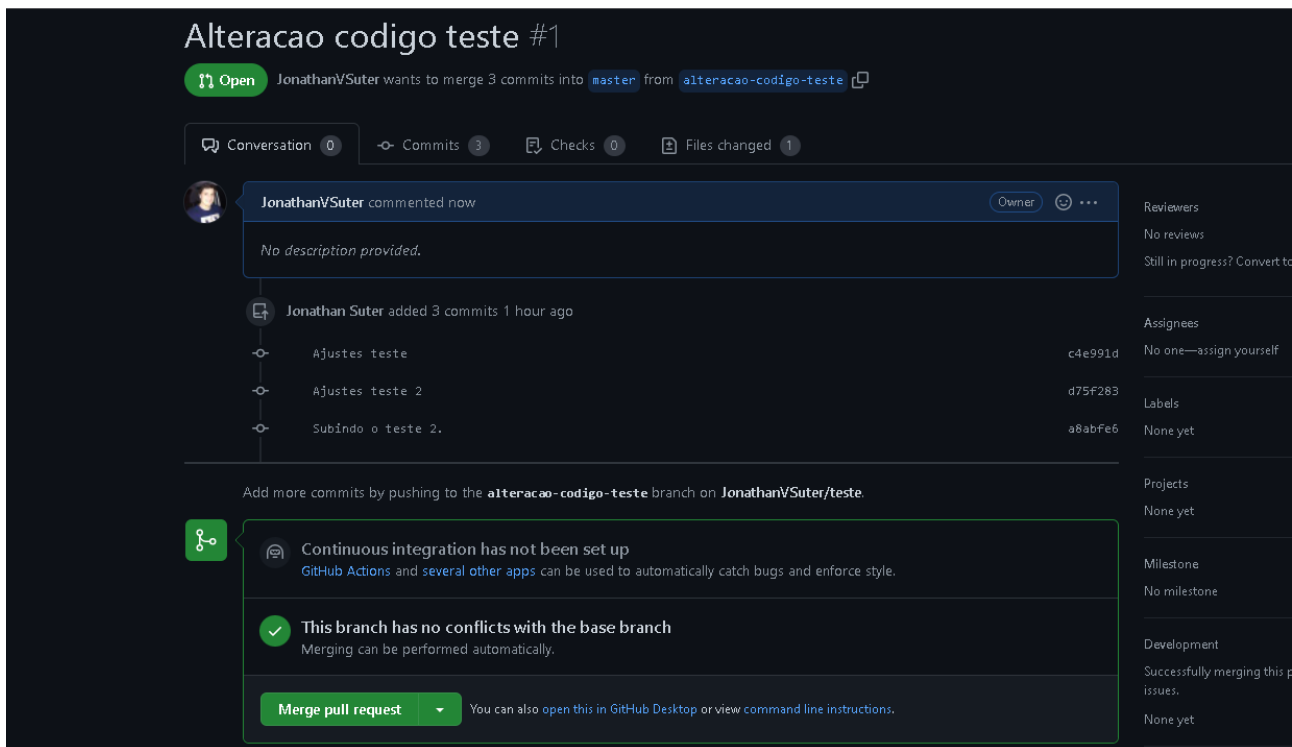
Nesta página, será exibida a mensagem inserida no *commit* e o *GitHub* mostrará se é possível mesclar as *branch's* selecionadas.



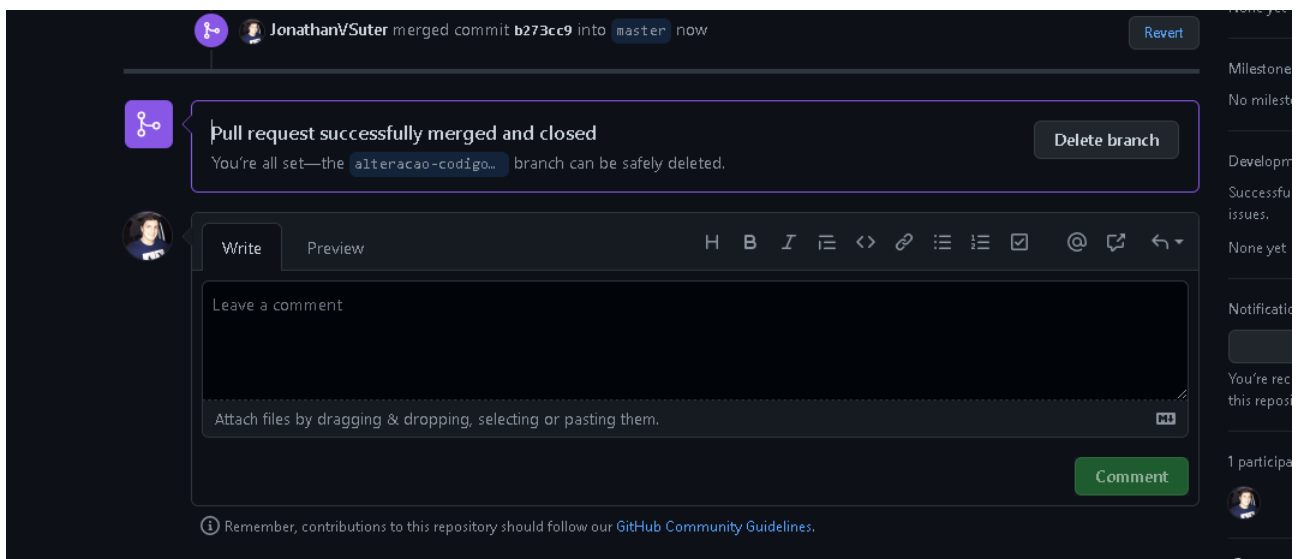
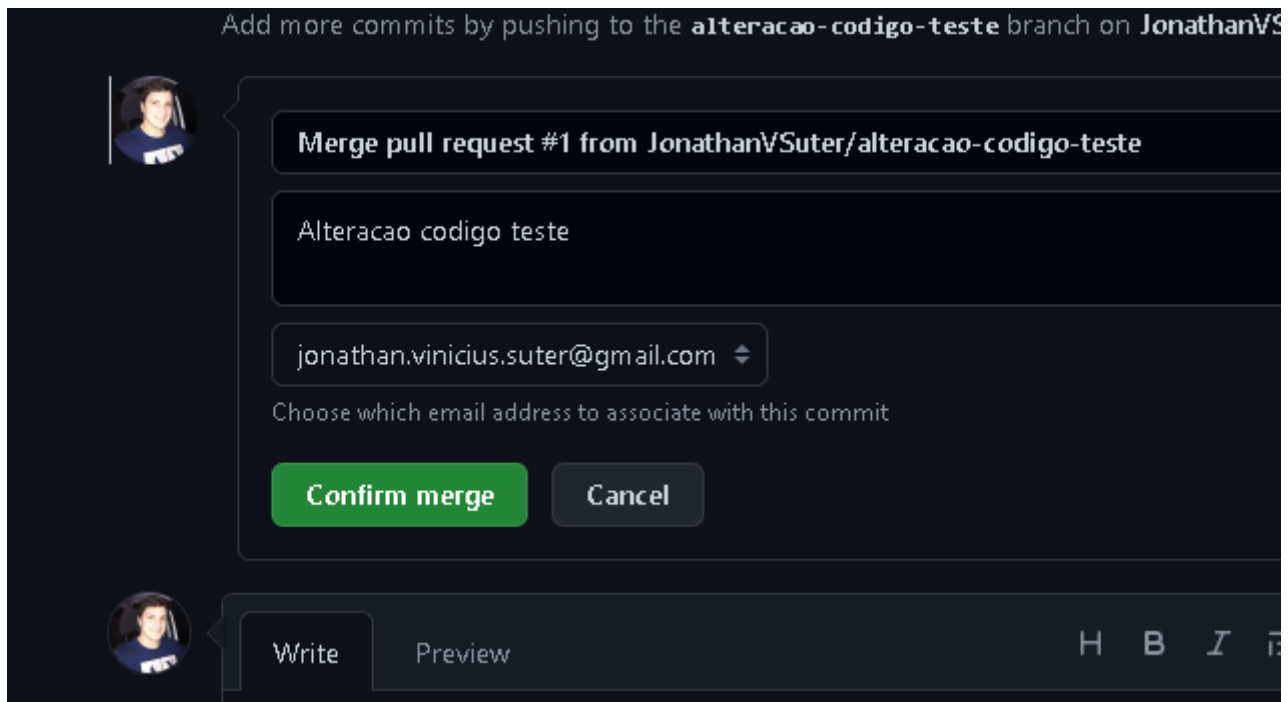
Esta linha tem o seguinte significado: “Mesclando a *branch* alteracao-codigo-teste na *branch* master”. Ou seja, as modificações feitas na *branch* alteracao-codigo-teste serão colocadas na master.



Clicando em *Create pull request*, abrirá a seguinte tela com a confirmação da mesclagem, mostrando os *commit's* feitos nesta *branch*.

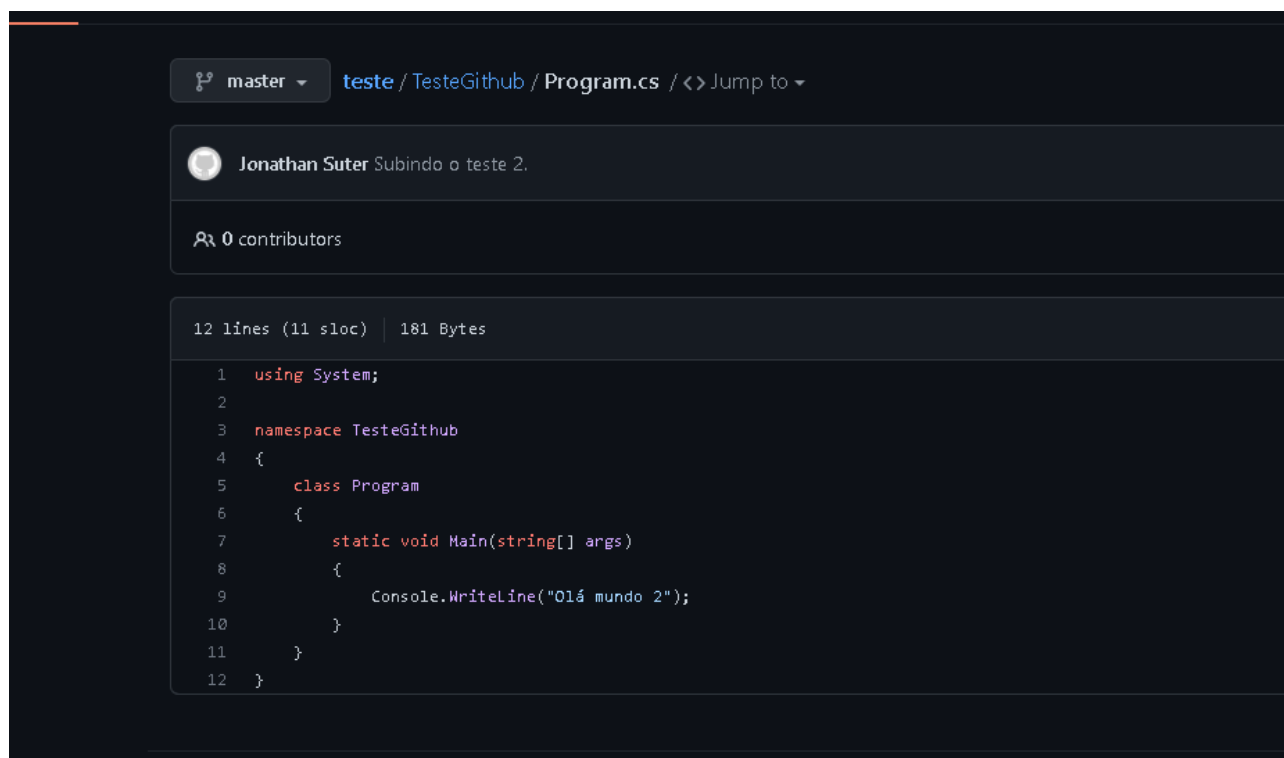


Caso tudo esteja certo, clicando em *Merge pull request*, solicitará a confirmação *Confirm merge*.



Assim, as modificações feitas serão empurradas para a *branch* selecionada.

Analisando a *branch* master, será possível ver que as modificações foram propagadas:



The screenshot shows a GitHub commit interface. At the top, the branch is set to 'master' and the file path is 'teste / TesteGithub / Program.cs'. The commit is by 'Jonathan Suter' with the message 'Subindo o teste 2.'. Below the commit message, it shows '0 contributors'. The code file 'Program.cs' is displayed with 12 lines (11 sloc) and 181 Bytes. The code is as follows:

```
1  using System;
2
3  namespace TesteGithub
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Olá mundo 2");
10         }
11     }
12 }
```