# FICHA DE TRABALHO 5

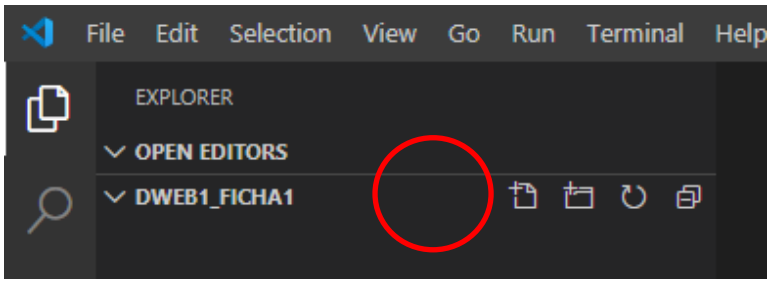
**Objetivos:**
- Utilização do Git e GitHub.

*O git é um sistema de controlo e versionamento de projetos, permitindo a divisão de um projeto pelas pessoas que trabalham no mesmo e ainda controlar versões de desenvolvimento.*

Nesta ficha de trabalho vamos criar um repositório git, manipular o repositório e ligar ao repositório online GitHub. Realçar que a explicação teórica e prática está disponível em vídeo em: https://youtu.be/5fM4hxxAuWI

## Parte I – criação do repositório local

1. Crie uma pasta e atribua-lhe o nome "DWEB1_Ficha1".
2. No Visual Studio Code abra o menu "File" e selecione "Open Folder". Escolha a pasta e clique no botão "Selecionar Pasta". Conseguirá perceber que é apresentada do lado esquerda uma zona com o nome da pasta.
3. Crie um novo ficheiro com o nome "README.md" dentro da pasta criada. Para isso deve clicar no botão "New File" junto ao nome da pasta, atribuir o nome ao ficheiro e carregar na tecla "Enter":




*O ficheiro README.md é usado para apresentar informação sobre o repositório em que está inserido. Há normas específicas para formatação de um ficheiro deste tipo mas é algo que não vamos focar, ainda assim se quiser saber mais pode consultar o link: https://medium.com/@raullesteves/github-como-fazer-um-readme-md-bonit%C3%A3o-c85c8f154f8*

4. Acrescente o texto: "# Comandos de Git" ao ficheiro e grave.

*Para aumentar a eficiência do trabalho pode selecionar a opção "Auto Save" no menu "File", assim sempre que clicar fora de um ficheiro este será gravado de forma automática.*

5. Vamos agora criar o repositório. No menu "Terminal" selecione a opção "New Terminal" e verifique a abertura de uma janela na parte de baixo do editor.
6. Para verificar a correta instalação do git insira o comando "git --version" e verifique se é apresentada a informação sobre a versão instalada. Caso por alguma razão não seja apresentada a informação deve verificar se o git foi corretamente instalado na máquina de trabalho.
7. Para finalmente iniciar o repositório insira o comando "git init". Verifique se na mensagem de feedback há informação de criação do repositório. Repare que com a execução deste comando será apresentado ao lado do ficheiro "README.md" a letra "U", o que indica que o ficheiro não está a ser rastreado ("untracked").

Vamos neste ponto fazer uma configuração necessária na primeira utilização do git, que será a identificação do programador, assim no terminal execute os seguintes comandos trocando o nome e email do exemplo pelos seus dados.

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git config --global user.name "Bruno"
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git config --global user.email "brunofrs7@gmail.com"
```

NOTA: este procedimento será necessário realizar apenas uma vez.

8. Crie agora um novo ficheiro com o nome "index.html" e insira como conteúdo "Ficheiro de HTML".
9. Vamos agora verificar o estado do repositório. No terminal insira o comando "git status". Verifique a mensagem:

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        index.html

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

É possível verificar que os dois ficheiros que criamos não estão a ser rastreados.

10. Adicione o ficheiro "index.html" ao controlo de versões utilizando o comando "git add index.html". Execute novamente o "git status" e verifique que o ficheiro já está a ser rastreado.
11. Altere o conteúdo do ficheiro "README.md" acrescentando:

```
## Configuração de utilizador
git config --global user.name "NOME"
git config --global user.email "EMAIL"

## Comandos
git init --> inicia o repositório
git status --> verifica o estado do repositório
git add NOMEFICHEIRO --> adiciona um ficheiro específico ao repositório
```

Neste ponto o repositório local contém dois ficheiros, sendo que apenas o ficheiro HTML está a ser rastreado, assim o controlo de versões será realizado apenas nesse ficheiro.

## Parte II – controlo de versões no repositório

1. Vamos agora criar o primeiro ponto de controlo do nosso projeto, para isso vamos utilizar o comando "git commit". Sempre que o comando é executado é necessário adicionar uma mensagem que servirá para controlar o que foi feito desde a última versão. Assim execute o comando abaixo e verifique a mensagem de confirmação:

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git commit -m "Adicionado ficheiro index.html"
[master (root-commit) a5decb8] Adicionado ficheiro index.html
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

2. Executando o comando "git status" é possível verificar que o não há informação sobre o ficheiro "index.html" pois não foi alterado desde o último commit.
3. Altere o conteúdo do ficheiro "index.html" para "Ficheiro de HTML alterado".
4. Execute novamente o comando "git status" e verifique que é apresentada a informação de que o ficheiro "index.html"
5. Adicione o ficheiro "index.html" novamente para o controlo de versões e efetue novo commit com a mensagem "Ficheiro index.html alterado".

NOTA: pode utilizar o comando "git commit -am MENSAGEM" para adicionar os ficheiros alterados ao controlo de versões e commitar tudo no mesmo comando (evita a necessidade de fazer git add NOMEFICHEIRO).

6. Utilize o comando "git log" para verificar o histórico do repositório.

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git log
commit 03a5d3e9277cf7a6c302170ce554f47bf87d50a9 (HEAD -> master)
Author: Bruno <brunofrs7@gmail.com>
Date:   Sat May 30 19:00:36 2020 +0100

    Ficheiro index.html alterado

commit a5decb8d1cd4c6a4b86b07623283f8d1e0d9bc34
Author: Bruno <brunofrs7@gmail.com>
Date:   Sat May 30 18:53:37 2020 +0100

    Adicionado ficheiro index.html
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

É possível verificar o código do commit (código a amarelo), autor, data e hora e mensagem informativa de cada commit, o que permite em cada momento verificar o que foi feito e por quem. De notar que neste caso como estamos a usar um repositório local só nosso o único autor apresentado será o utilizador atual, mas num repositório partilhado é essencial para perceber o trabalho desenvolvido por cada elemento.

7. Para verificar o que foi atualizado entre versões é possível utilizar o comando "git show". Copie o código do commit mais recente e execute o comando: "git show CODIGOCOMMIT".

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git show 03a5d3e9277cf7a6c302170ce554f47bf87d50a9
commit 03a5d3e9277cf7a6c302170ce554f47bf87d50a9 (HEAD -> master)
Author: Bruno <brunofrs7@gmail.com>
Date:   Sat May 30 19:00:36 2020 +0100

    Ficheiro index.html alterado

diff --git a/index.html b/index.html
index ed66369..89601a7 100644
--- a/index.html
+++ b/index.html
@@ -1 +1 @@
-Ficheiro de HTML
\ No newline at end of file
+Ficheiro de HTML alterado
\ No newline at end of file
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

Como o git funciona com alteração de ficheiros por linha conseguimos verificar que no ficheiro index.html foi eliminada a linha que continha "Ficheiro de HTML" e adicionada a linha "Ficheiro de HTML alterado". Efetivamente o que foi feito não foi a eliminação da linha e criação de uma nova, mas o funcionamento interno do git faz o controlo dessa forma.

NOTA: se executar apenas o comando "git show" são apresentadas as alterações do último commit.

8. Acrescente ao ficheiro "README.md" o seguinte conteúdo:

| |
|---|
| git commit -m "MENSAGEM" --> cria ponto no controlo de versões do repositório com os ficheiros adicionados<br>git commit -am "MENSAGEM" --> cria um commit e adiciona os ficheiros modificados já adicionados a um commit anterior do repositório<br>git log --> mostra as informações de todos os commit feitos<br>git show CODIGOCOMMIT--> apresenta as alterações efetuadas no commit do código fornecido<br>git show --> apresenta as alterações efetuadas no último commit |

## Parte III – fluxos de trabalho com branch

Mais uma das funcionalidades muito interessantes e úteis é a possibilidade de criar fluxos de trabalho. Imaginando que no repositório está um projeto já terminado e em produção num cliente, mas que é necessário acrescentar uma nova funcionalidade. Caso estejamos a trabalhar no projeto e façamos uma alteração numa funcionalidade já existente pode acontecer de o programa deixar de funcionar, assim deve ser criado um fluxo de trabalho separado para não influenciar a versão atual do programa e quando tudo estiver implementado e testado passaremos para a versão em produção.

1. No terminal execute o comando "git branch novaFuncionalidade"
2. Execute agora o comando "git branch". São apresentadas as duas linhas do tempo do repositório, neste caso a "master" que é a linha principal de qualquer repositório e a "novaFuncionalidade" que foi a que acabamos de acrescentar. Conseguimos ainda verificar que existe um * antes de "master" uma vez que é a branch que onde estamos.
3. Para mudar para a nova branch execute o comando "git checkout novaFuncionalidade" e verifique a mensagem de confirmação. Execute também um "git status" e verifique que somos informados de que estamos na branch "novaFuncionalidade".

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git branch
* master
  novaFuncionalidade
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git checkout novaFuncionalidade
Switched to branch 'novaFuncionalidade'
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git status
On branch novaFuncionalidade
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

4. Acrescente um novo ficheiro com o nome "ficheiro.txt" e conteúdo "Nova Funcionalidade do projeto".
5. Adicione o ficheiro e realize o commit do mesmo com a mensagem "Nova Funcionalidade implementada".
6. Execute o comando "git log" e verifique que o último commit está relacionado com a branch "novaFuncionalidade" e que existe a branch "master" com o commit anterior.

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git log
commit 81dfa9c4b189b3528bbe51a5d98b8f24151e3a18 (HEAD -> novaFuncionalidade)
Author: Bruno <brunofrs7@gmail.com>
Date:   Sat May 30 19:39:57 2020 +0100

    Nova Funcionalidade implementada

commit 03a5d3e9277cf7a6c302170ce554f47bf87d50a9 (master)
Author: Bruno <brunofrs7@gmail.com>
Date:   Sat May 30 19:00:36 2020 +0100

    Ficheiro index.html alterado

commit a5decb8d1cd4c6a4b86b07623283f8d1e0d9bc34
Author: Bruno <brunofrs7@gmail.com>
Date:   Sat May 30 18:53:37 2020 +0100

    Adicionado ficheiro index.html
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

7. Execute o comando "git checkout master" para voltar à branch principal e verifique o que acontece. A primeira alteração é que o ficheiro "ficheiro.txt" desapareceu, e realizando um "git log" não é possível verificar a existência do commit "Nova Funcionalidade implementada". Isto deve-se ao facto de não estar presente nesta linha do tempo o que permite que não haja interferência de uma branch noutra.

NOTA: neste caso estamos a trabalhar com alteração de ficheiros diferentes, mas as alterações dentro de um ficheiro já existente também são consideradas.

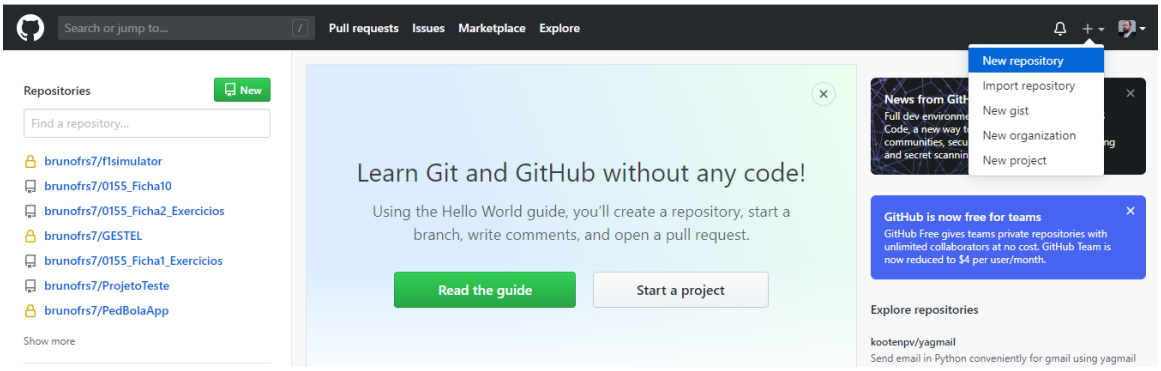
8. Considerando que a nova funcionalidade está concluída e testado e pode ser passada para a branch "master" devemos fazer o "merge" das duas, aplicando o comando "git merge novaFuncionalidade". É possível verificar que o "ficheiro.txt" é transportado para a branch "master" bem como qualquer outra alteração que pudesse ter sido feito na branch "novaFuncionalidade".
9. Finalmente e como não vamos fazer mais alterações na branch "novaFuncionalidade" devemos alterar a mesma, assim é possível executar o comando "git branch -D novaFuncionalidade". Execute o comando "git branch" para confirmar que só existe a branch "master".
10. No ficheiro "README.md" acrescente o seguinte:

```
git branch NOME --> cria uma nova branch
git branch --> mostra as branch existentes
git checkout NOME --> troca para a branch NOME
git merge NOME --> estando numa branch acrescenta as funcionalidades da branch NOME à atual
git branch -D NOME --> elimina a branch NOME
```

## Parte IV – Utilização do GitHub

O GitHub é um dos repositórios online mais utilizados pela comunidade de desenvolvimento de software servindo não só para armazenamento e partilha de repositórios, mas também como rede social entre os desenvolvedores.

1. Registe-se no site https://github.com/
2. Após registo e autenticação será apresentada a sua área de trabalho. Clique no botão verde à esquerda com a mensagem "New" ou no botão + em cima à direita e de seguida "New Repository" para criar um novo repositório online.

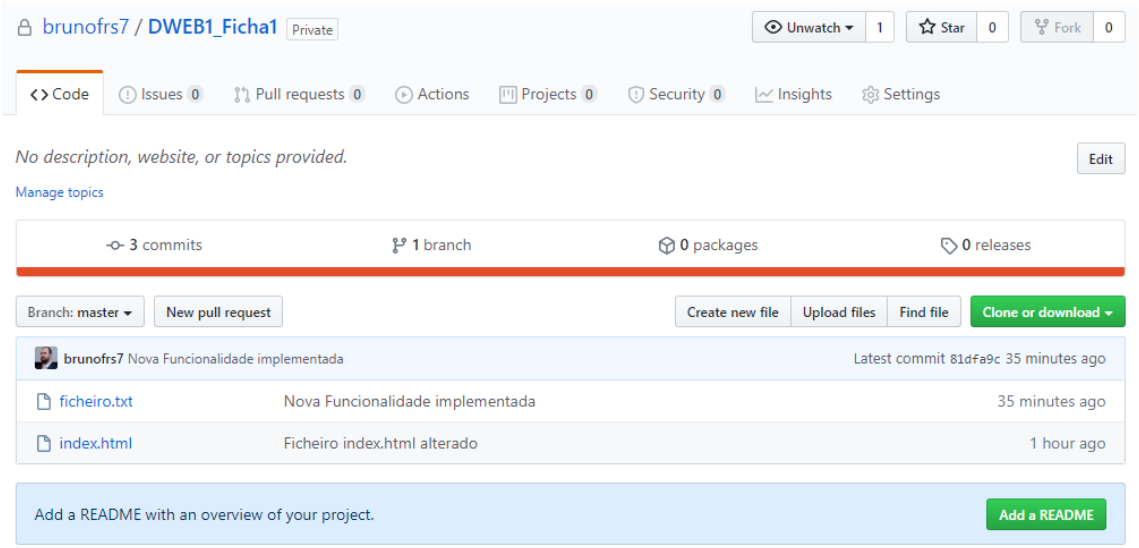



3. No campo "Repository name" coloque: "DWEB1_Ficha1" e coloque-o como privado. Não marque a opção de criar um ficheiro README pois já temos esse ficheiro no nosso repositório privado e clique em "Create repository".
4. Na nova janela são-nos apresentados dois blocos de código, sendo o primeiro "…or create a new repository on the command line" os comandos que devemos utilizar no terminal caso ainda não tenhamos um repositório local. No nosso caso vamos utilizar o bloco seguinte "…or push an existing repository from the command line". Copie as duas linhas abaixo e execute-as no terminal.
   NOTA: deverá, na primeira execução, ser apresentada uma mensagem ou no terminal ou numa nova janela para o utilizador inserir as credenciais de acesso ao GitHub; cada endereço será diferente pois está ligado à conta pessoal do GitHub de cada um.

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git remote add origin https://github.com/brunofrs7/DWEB1_Ficha1.git
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git push -u origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (9/9), 777 bytes | 259.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/brunofrs7/DWEB1_Ficha1.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

O comando "git remote" vai permitir criar uma ligação entre o repositório atual local e o passado no link do repositório online.
O comando "git push" permite enviar o repositório atual local para o GitHub. De notar que na primeira execução é necessário especificar a branch master para que a mesma seja criada no repositório online. Posteriormente bastará utilizar o comando "git push" para atualizar o projeto no GitHub.
NOTA: caso de cada vez que executa um "git push" seja obrigado a inserir as suas credenciais do GitHub pode utilizar o comando "git config credential.helper store" para armazenar os dados de acesso na máquina em que está a trabalhar.

5. Neste ponto temos o repositório local ligado ao repositório online. No browser clicando na aba "Code" é possível verificar que já estão online os ficheiros do último commit.



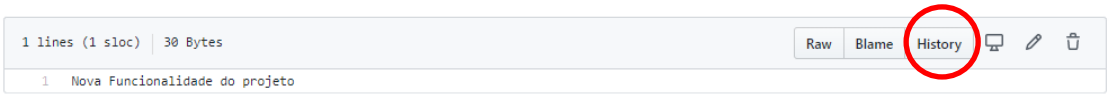

6. No ficheiro "README.md" acrescente o seguinte:

| |
|---|
| git remote add origin LINK --> liga o repositório atual local a um repositório online<br>git push -u origin master --> envia a branch master para o repositório online (necessário apenas no primeiro envio)<br>git push --> enviar o repositório atual local para o repositório online |

## Parte V – Resolução de conflitos

Se o repositório em utilização for partilhado por vários contribuidores ou a mesma pessoa utilize duas máquinas diferentes para realizar operações, é possível que em algum momento haja alterações conflituosas. Por exemplo, o utilizador A eliminou o "ficheiro.txt" e o utilizador B alterou o conteúdo do "ficheiro.txt". Neste caso tem de haver um ponto de entendimento entre ambas as partes.

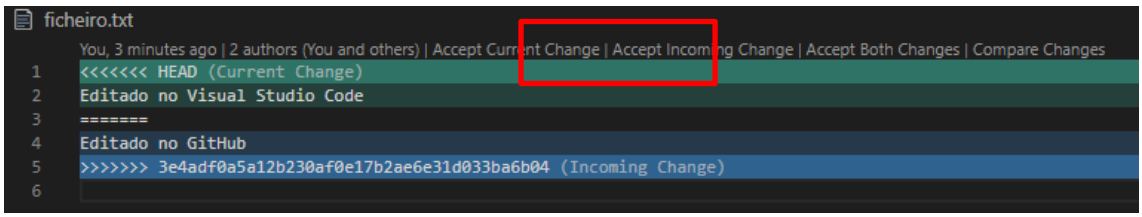
1. Clique no "ficheiro.txt" e verifique o seu conteúdo. Clique agora o símbolo de edição à direita e edite o ficheiro para "Editado no GitHub".




2. Após edição clique no botão verde "Commit changes" no fundo da página.
3. No Visual Studio Code edite o mesmo ficheiro colocando "Editado no Visual Studio Code".
4. Abra o terminal e grave as alterações e acrescente o ficheiro README.md ao controlo de versões (notar que para adicionar todos os ficheiros do repositório pode no comando "git add" substituir o nome do ficheiro pelo símbolo ponto final). Utilize como mensagem de commit "ficheiro.txt alterado; adicionado ficheiro README.md".
5. Até este ponto não há qualquer erro, mas o conteúdo no GitHub e no repositório local estão diferentes. Vamos tentar enviar o repositório local para o GitHub com o comando "git push".

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git push
To https://github.com/brunofrs7/DWEB1_Ficha1.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/brunofrs7/DWEB1_Ficha1.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

6. Sempre que estamos a trabalhar num repositório partilhado devemos descarregar o repositório online antes de enviar as nossas alterações, assim fazemos primeiro "git pull".
7. É-nos apresentada uma mensagem de conflito e apresentado o mesmo. Com o Visual Studio Code temos a vantagem de poder automaticamente selecionar qual a alteração que queremos que esteja válida: a local, a do repositório ambas (caso não haja sobreposição) ou comparar. Neste caso vamos aceitar as alterações do repositório.




8. Após aceitar as alterações devemos fazer nova adição de ficheiros e commit.
9. Para finalizar voltamos a fazer um pull e push de seguida para garantir que as alterações serão gravadas numa versão única.

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git add .
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git commit -m "alterações do repositório aceites"
[master f8a3a38] alterações do repositório aceites
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git pull
Already up to date.
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.23 KiB | 627.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/brunofrs7/DWEB1_Ficha1.git
   3e4adf0..f8a3a38  master -> master
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

10. No ficheiro "README.md" acrescente o seguinte:

| git pull --> descarregar alterações que existam no repositório online |
|---|

## Parte VI – Descarregar repositório online

O GitHub possui um grande número de repositórios públicos, repositórios esses que estão totalmente disponíveis para utilização por parte de qualquer pessoa.

1. No site do GitHub pesquise por um repositório qualquer ou utilize este como exemplo: https://github.com/brunofrs7/ProjetoTeste
2. No Visual Studio Code, abra o terminal e utilize o comando "git clone LINK" substituindo o LINK pelo link do repositório que pretende descarregar.

```
PS C:\Users\bruno\Desktop\DWEB1_Ficha1> git clone https://github.com/brunofrs7/ProjetoTeste
Cloning into 'ProjetoTeste'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
PS C:\Users\bruno\Desktop\DWEB1_Ficha1>
```

3. Verifique que na pasta do projeto foi criada uma nova pasta com o conteúdo do repositório descarregado.


Estas são algumas funcionalidades disponibilizadas pelo Git e GitHub, uma ferramenta utilizada em larga escala pelas empresas e que permite aumentar a produtividade.

**Bom trabalho!** ☺