

Git/GitHub

Sumário

Aula 01 – O Que É, Pra Que Serve, Como Funciona?.....	2
Aula 02 – Instalando o Git.....	2
Aula 03 – Configurando o Git.....	2
Aula 04 – Iniciando um Repositório.....	3
Aula 05 – Branch, README e Commit.....	4
Aula 06 – Revertendo Modificações.....	7
Aula 07 – Trabalhando com Diferentes Branchs.....	8
Aula 08 – Diferenças entre Arquivos (Diff).....	10
Aula 09 – Criando um Repositório no Github.....	11
Aula 10 – Conectando Repositório Local ao Remoto.....	13
Aula 11 – Fazendo Alterações no Repositório Remoto.....	17
Aula 12 – Ignorando Os Arquivos do Repositório.....	17
Aula 13 – Reverter Sem Perder o Código (revert).....	19
Aula 14 – Deletando Branches Locais e Remotos.....	19
Aula 15 – Puxando Alterações de Outras Pessoas (pull).....	22
Aula 16 – Clonando Repositórios Remotos.....	23
Aula 17 – Contribuindo Com Outros Repositórios (fork / pull request).....	23

Aula 01 – O Que É, Pra Que Serve, Como Funciona?

Aula 02 – Instalando o Git

Como verificar se o Git está instalado no computador.

- Abrimos o cmd(Prompt de Comando)
- No prompt de comando digite o seguinte comando: `git --version`

Site para baixar o Git:

<https://git-scm.com/>

Aula 03 – Configurando o Git

1º Passo, precisamos adicionar (configurar) o nosso nome de usuário no Git. Para isto utilizamos o seguinte comando:

```
git config --global user.name "Daniel Quadros"
```

2º Após definirmos o nome de usuário, precisamos configurar o nosso e-mail. Para isto utilizamos o seguinte comando:

```
git config --global user.email "danieltquadros@hotmail.com"
```

OBS.: Nenhum destes comandos retornará qualquer tipo de mensagem. Eles apenas abrirão uma nova linha no Prompt de Comando

É possível também configurar qual editor de códigos está sendo utilizado. Para isto, utilizamos o seguinte comando:

```
git config --global core.editor vscode
```

Aqui, utilizei o vscode no final porque é o editor de códigos que eu utilizo. O Bonieky utilizou o sub, para o Sublime.

Podemos utilizar os seguintes comandos para visualizar o que foi configurado no nosso Git:

git config user.name → Para visualizar o nome de usuário.

git config user.email → Para visualizar o e-mail.

```
C:\Users\danie>git config user.name
Daniel Quadros

C:\Users\danie>git config user.email
danieltquadros@hotmail.com
```

git config --list → Este comando exibe todas as configurações existentes.

Aula 04 – Iniciando um Repositório

1º → Por questão de organização, é interessante criar uma pasta onde estarão todos os projetos que irão para o GitHub.

2º → Utilizando o Prompt de Comando, navegue até a pasta criada para este propósito. No meu caso:

D:\Informática\Programação\2021\Daniel\Projetos GitHub onde criei pastas para cada curso e projeto.

Para este curso utilizarei a pasta:

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git

3º → No Prompt de Comando, nesta pasta utilizamos o seguinte comando:

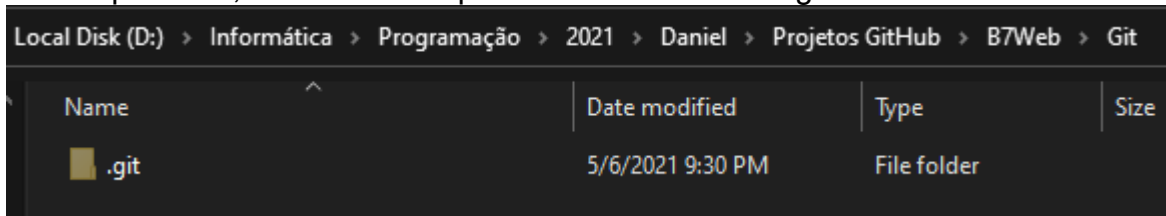
git init → Este comando inicia um novo repositório.

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git init  
Initialized empty Git repository in D:/Informática/Programação/2021/Daniel/Projetos GitHub/B7Web/Git/.git/
```

Aqui, o sistema informou que foi inicializado um repositório vazio (empty) do Git.

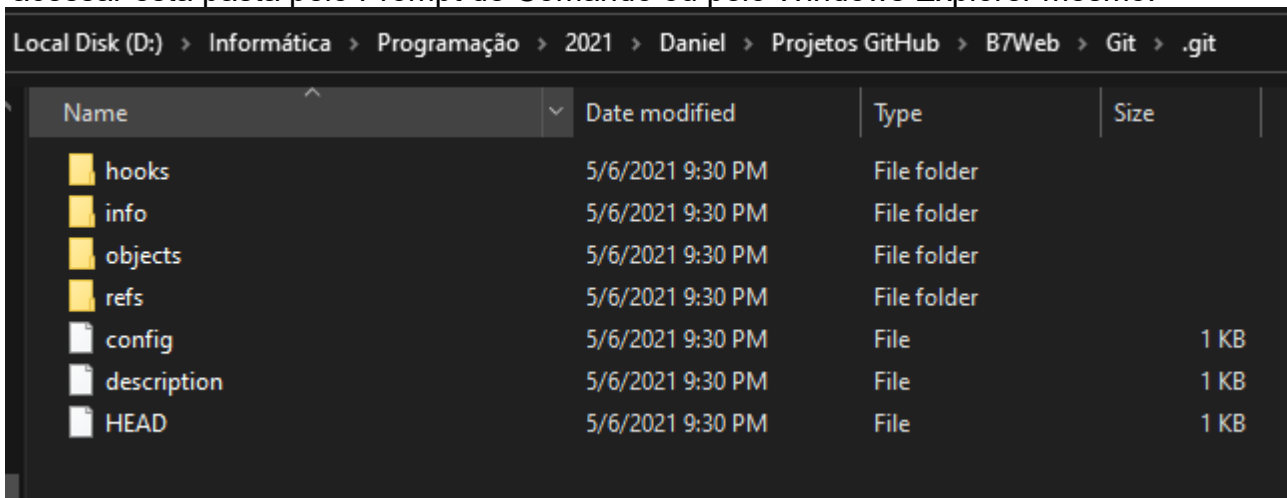
Se por acaso já existirem arquivos na pasta, este comando criará o repositório levando em consideração todos os arquivos que estão lá.

Quando é criado o repositório, o Git cria uma pasta oculta chamada '.git'



Local Disk (D:) > Informática > Programação > 2021 > Daniel > Projetos GitHub > B7Web > Git			
Name	Date modified	Type	Size
.git	5/6/2021 9:30 PM	File folder	

Podemos acessar esta pasta pelo Prompt de Comando ou pelo Windows Explorer mesmo.



Local Disk (D:) > Informática > Programação > 2021 > Daniel > Projetos GitHub > B7Web > Git > .git			
Name	Date modified	Type	Size
hooks	5/6/2021 9:30 PM	File folder	
info	5/6/2021 9:30 PM	File folder	
objects	5/6/2021 9:30 PM	File folder	
refs	5/6/2021 9:30 PM	File folder	
config	5/6/2021 9:30 PM	File	1 KB
description	5/6/2021 9:30 PM	File	1 KB
HEAD	5/6/2021 9:30 PM	File	1 KB

Obs.:

Podemos sempre que quisermos criar repositórios tanto em pastas vazias ou em pastas que nosso projeto já esteja em funcionamento sem problema nenhum.

Apenas esse passo, não é suficiente para colocar nosso projeto lá no GitHub, mas é um passo essencial para que isto aconteça.

Aula 05 – Branch, README e Commit

Branch → São versões diferentes do seu sistema. A branch principal (versão principal) se chama **master** ou **main**.

Obs.: Quando criamos o nosso repositório, ou seja, quando realizamos o nosso primeiro ‘commit’ de um projeto ele cria a versão chamada **master** ou **main**.

Commit → Comando que faz com que os arquivos fiquem sincronizados (novos ou atualizados) com o Git.

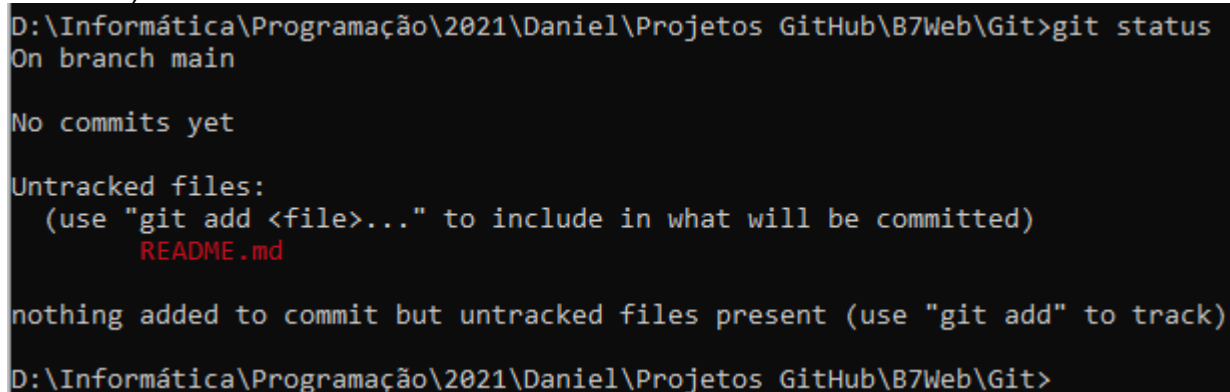
Obs.: No caso de uma atualização, ele sincroniza apenas o que foi modificado atrelando esta modificação ao usuário que realizou esta modificação. Algo muito útil em um projeto realizado em equipe.

O Commit vem (ou pelo menos, deve vir) acompanhado de um comentário. Geralmente adiciona-se um comentário sobre que alterações foram feitas. É muito importante ser claro nestas mensagens para uma melhor compreensão.

README → Deve conter instruções, comentários... Podemos criar o arquivo README.txt ou README.MD. O mais usual é o README.MD.

Após criarmos o nosso README.MD, podemos acessar e editar este arquivo através do editor de código (vscode).

git status → Este comando realiza uma varredura a pasta e verifica tudo o que foi modificado (adicionado, alterado ou excluído)

A screenshot of a Windows command prompt window with a black background and white text. The text shows the execution of the 'git status' command in a directory path: 'D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git'. The output indicates that there are no commits yet and that the 'README.md' file is untracked. It provides instructions on how to use 'git add' to track the file. The prompt ends with a greater-than sign '>'.

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git status
On branch main

No commits yet

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)
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Neste exemplo, foi informado que no Branch **main** não foi realizado nenhum commit e que foi verificada a existência de um novo arquivo, o README.md e informando que este arquivo não está sendo monitorado (Untracked). É necessário adicionar este arquivo ao monitoramento do GIT.

Há duas formas para adicionarmos arquivos ao monitoramento do GIT:

Adicionando arquivo por arquivo: **git add** → git add README.md

Ou adicionando todos os arquivos que foram encontrados e ainda não estão sendo monitorados de uma só vez: **git add -A** → A = All

Se executarmos o git status mais uma vez, após adicionarmos um novo arquivo, veremos o seguinte resultado:

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Realizando o 1º Commit:

git commit -m “Primeiro Commit” → Comando para realizar Commit. Entre as aspas, adicionamos a mensagem referente ao motivo do commit ou qualquer informação importante ou necessária que seja registrada para este Commit.

O Prompt retornará uma mensagem com as informações do Commit realizado. No nosso exercício tivemos o seguinte retorno:

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git commit -m "Primeiro Commit"
[main (root-commit) 64d5ebe] Primeiro Commit
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

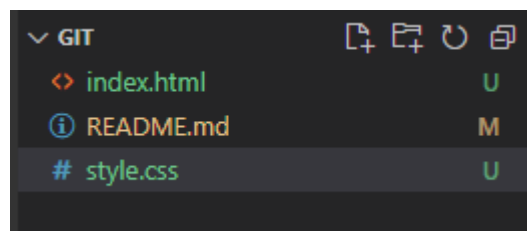
git log → Este comando retorna a lista de todos os comites que foram realizados no Branch específico, com informações essenciais como o nome do usuário (Author) e o e-mail da pessoa que realizou aquele commit assim também como a data e a hora em que foi realizado e o texto da informação que adicionamos ao realizarmos este Commit, neste exemplo “Primeiro Commit.:

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git log
commit 64d5ebe9e3bc60121b03fc0edaf171309df25c65 (HEAD -> main)
Author: Daniel Quadros <danielquadros@hotmail.com>
Date:   Thu May 6 22:28:42 2021 -0300

    Primeiro Commit

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

INFORMAÇÃO EXTRA À AULA:



Status dentro do VS Code.: Neste Print acima, o meu README.md já havia sido commitado, porém foi realizada alteração após o commit (ficou com a cor amarela e com a letra ‘M’, provavelmente indicando Modificação) os arquivos index.html e style.css foram criados após o último commit (neste caso, ficaram verdes e receberam a letra ‘U’ provavelmente de Uncommitted)

Por questões de aprendizagem, realizamos as seguintes operações via VS Code:

- Alterado o conteúdo do arquivo README.md.
- Criado dois novos arquivos. O index.html e o style.css

Retornado ao Prompt de comando e novamente executado o comando git status. Obtivemos o seguinte retorno:

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

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

no changes added to commit (use "git add" and/or "git commit -a")
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

O sistema verificou a mudança ocorrida no README.md e verificou também a presença dos nossos dois novos arquivos.

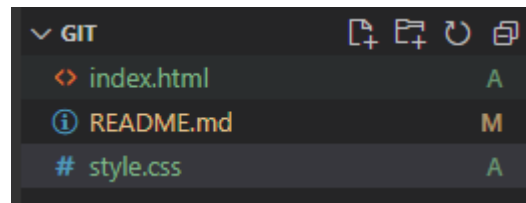
- Executamos o **git add -A** para que estas modificações sejam rastreadas pelo GIT

O git status retornou a seguinte informação:

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
        new file:   index.html
        new file:   style.css

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

E o VS Code ficou assim:

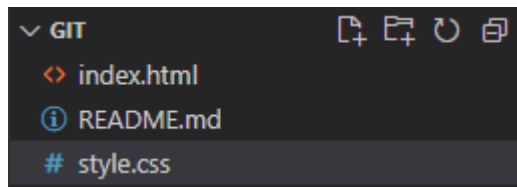


Obs.: Enquanto não executarmos o Commit, essas alterações, ficarão apenas no nosso computador. Para torná-las disponíveis no repositório, é necessário a execução do Commit.

Aqui no exercício executamos o **git commit -m** "Criando os arquivos principais, index.html e style.css"

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git commit -m "Criando os arquivos principais, index.html e style.css bem como modificando o README"
[main f776499] Criando os arquivos principais, index.html e style.css bem como modificando o README
3 files changed, 2 insertions(+), 1 deletion(-)
create mode 100644 index.html
create mode 100644 style.css
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>_
```

Após este Commit o VS Code ficou assim:



Aula 06 – Revertendo Modificações

Iniciamos a aula aqui executando o **git log** no cmd.

É importante observar que cada commit realizado gerou um número de identificação, porém o importante deste número são os 7 primeiros caracteres:

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git log
commit f776499030fee956a06480d865ad2ce423122a27 (HEAD -> main)
Author: Daniel Quadros <danielquadros@hotmail.com>
Date: Thu May 6 22:59:44 2021 -0300

    Criando os arquivos principais, index.html e style.css bem como modificando o README

commit 64d5ebe9e3bc60121b03fc0edaf171309df25c65
Author: Daniel Quadros <danielquadros@hotmail.com>
Date: Thu May 6 22:28:42 2021 -0300

    Primeiro Commit

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Além disto, podemos verificar o HEAD que estamos, que neste caso é o **'main'**

Para verificarmos o HEAD, podemos também utilizar o comando **git branch**.

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git branch
* main
```

Este comando lista todos os branches que houverem no projeto, e o que estiver com o **'*'** é o nosso, ou seja o branch que estamos naquele momento.

Aqui, para o exercício, faremos modificações no nosso README.md

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git commit -am "Nova modificação para testar o -am no Commit"
[main 6e95fec] Nova modificação para testar o -am no Commit
1 file changed, 3 insertions(+), 1 deletion(-)

D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

git commit -am → Se já tiver feito modificação e não quiser fazer o add novamente, podemos utilizar essa variação com o **-am**.

Se após um commit realizado, por algum motivo, quisermos retornar a uma versão anterior.

git reset → Existem três formas:

--soft → Volta para a versão anterior, porém com alterações realizadas, mas não commitadas (Volta ao estado antes do commit → ignora apenas o commit).

--mixed → Faz a mesma coisa que o **--soft**, porém, não fica preparado para commitar. Retorna para a versão anterior sem a realização dos comandos **'add'** e **'commit'**.

--hard → Retorna totalmente para a versão válida anterior, excluindo todas as alterações realizadas antes do último commit especificado.

Para isto, precisaremos do número do commit, que queremos reverter, ou seja, para que a nossa aplicação retorne ao ponto que era antes do commit que iremos especificar.

Obs. Podemos copiar o número todo, ou apenas os 7 primeiros dígitos

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>git reset --hard 64d5ebe9e3bc60121b03fc0edaf171309df25c65
HEAD is now at 64d5ebe Primeiro Commit
```

```
D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Aqui utilizamos o comando: **git reset --hard 64d5ebe9e3bc60121b03fc0edaf171309df25c65**

Aula 07 – Trabalhando com Diferentes Branchs

Até aqui, estamos trabalhando com o **branch main**. Agora, vamos criar novos branches.

Para visualizarmos em qual branch estamos, utilizamos o comando **git branch**

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch
* main
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Para criar um novo branch, utilizamos o comando **git branch nomedobbranch**

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

O comando não retorna nenhuma informação, mas se dermos o comando **git branch**, podemos visualizar:

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch
* main
  teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Para trocarmos de branch, utilizamos o comando **git checkout nomedobbranch**

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git checkout teste
Switched to branch 'teste'
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch
  main
* teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Para o exercício, fizemos alteração no README.md e criamos o arquivo index.html com algum conteúdo dentro. Com o comando **git status**, podemos ver as alterações realizadas.

Realizamos um novo commit e verificamos que permanecemos no branch teste.

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git commit -am "Testando o novo branch"
[teste 0a3fcdf] Testando o novo branch
 1 file changed, 3 insertions(+), 1 deletion(-)
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch
  main
* teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```


Obs.: Quando criamos um novo branch, ele importa o que tinha no branch atual e começa um histórico a partir daí. Demos um git log e podemos visualizar o histórico de commits

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git log
commit 0a3fcdfcbb8469986a187d16973db2fff5800a2e (HEAD -> teste)
Author: Daniel Quadros <danieltquadros@hotmail.com>
Date: Wed May 19 07:02:05 2021 -0300

    Testando o novo branch

commit 8278743453ab27e4224573c1c1f2ec87ad38e8d5 (main)
Author: Daniel Quadros <danieltquadros@hotmail.com>
Date: Wed May 19 06:49:04 2021 -0300

    Criando o style

commit 08248a6738d965b7287fc2cc08314efb85e8abe4
Author: Daniel Quadros <danieltquadros@hotmail.com>
Date: Tue May 18 23:02:24 2021 -0300

    Criando o arquivo css
:...skipping...
commit 0a3fcdfcbb8469986a187d16973db2fff5800a2e (HEAD -> teste)
Author: Daniel Quadros <danieltquadros@hotmail.com>
Date: Wed May 19 07:02:05 2021 -0300

    Testando o novo branch

commit 8278743453ab27e4224573c1c1f2ec87ad38e8d5 (main)
Author: Daniel Quadros <danieltquadros@hotmail.com>
Date: Wed May 19 06:49:04 2021 -0300

    Criando o style

commit 08248a6738d965b7287fc2cc08314efb85e8abe4
Author: Daniel Quadros <danieltquadros@hotmail.com>
```

Agora que estamos com o branch teste e fizemos as alterações, vamos voltar com o branch main (versão principal). Para isto utilizaremos o comando **git checkout main**

Obs.: Aqui, no branch **main**, se executarmos o comando `git status`, percebemos que o main não está reconhecendo o index.html como trackeado, pois não foi feito o commit aqui no main. Assim, também, se executarmos o `git log`, não veremos aqui, o commit que foi feito no branch teste.

Outro ponto interessante é que, voltando pro main, não temos nem a alteração que fizemos no arquivo README.md quando estávamos utilizando o branch teste. Porém o arquivo index.html que criamos com o branch teste, aparece aqui no main. Como se cada branch tivesse o seu próprio README.md

Obs.: Temos versões diferentes do sistema rodando em diferentes branches

Obs.: Sempre que criamos um novo branch, é como se dessemos um CTRL+C e CTRL+V no main e a partir daí, seguisse o seu próprio histórico.

Aula 08 – Diferenças entre Arquivos (Diff)

git diff → Este comando mostra o conteúdo que foi modificado.

Para testar o comando, alterei a palavra 'primeiro' pela palavra 'segundo' na linha 2 do README e adicionei a linha 5.

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git diff
diff --git a/README.md b/README.md
index 7b290df..14eeae5 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,5 @@
 Hello Word!!!
 :...skipping...
diff --git a/README.md b/README.md
index 7b290df..14eeae5 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,5 @@
 Hello Word!!!
 -Meu primeiro sistema orientado (B7Web - Bonieky Lacerda) no GIT.
 -Efetuado a criação do style.css
 \ No newline at end of file
+Meu segundo sistema orientado (B7Web - Bonieky Lacerda) no GIT.
+Efetuação a criação do style.css
+
+alguma nova linha
 \ No newline at end of file
~
~
~
~
~
~
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Além disso, na aula anterior, eu já havia criado o index.html e não havia dado o commit ainda.

Se modificarmos diversos arquivos, poderemos ter dificuldades em encontrar alguma modificação específica. Para otimizar a pesquisa, podemos utilizar o seguinte comando:

git diff --name-only → Este comando retorna apenas o nome do(s) arquivo(s) que foi(foram) alterado(s)

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git diff --name-only
README.md
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Ao descobrir diversos arquivos modificados e eu quiser saber o conteúdo que foi modificado em apenas um arquivo específico, eu posso utilizar o comando:

git diff nomedoarquivo → Este comando lista as alterações que foram realizadas em um arquivo específico.

```

PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git diff README.md
diff --git a/README.md b/README.md
index 7b290df..14eeae5 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,5 @@
Hello Word!!!
-Meu primeiro sistema orientado (B7Web - Bonieky Lacerda) no GIT.
-Efetuada a criação do style.css
\ No newline at end of file
+Meu segundo sistema orientado (B7Web - Bonieky Lacerda) no GIT.
+Efetuada a criação do style.css
+
+alguma nova linha
\ No newline at end of file
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>

```

Adicionamos uma linha de comentário no style.css apenas para teste

git checkout HEAD -- nomedoarquivo → Este comando reverte as modificações realizadas no arquivo com o branch atual. O **HEAD** é utilizado para chamar o branch atual, poderíamos ter utilizado no lugar do HEAD o nome do branch atual.

```

PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git checkout HEAD -- style.css
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>

```

OU

```

PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git checkout main -- style.css
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>

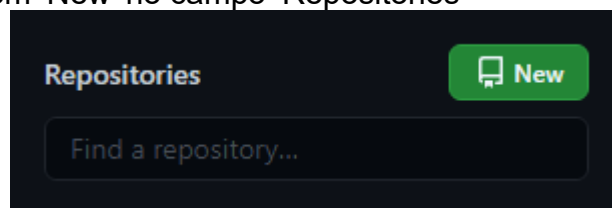
```

Após executar qualquer um dos dois comandos, o arquivo style.css voltou para o estado de antes da modificação realizada.

Obs.: Após retornarmos um arquivo para um estado anterior a qualquer modificação que não tenha sido commitado, utilizando o checkout, se executarmos o diff, o arquivo não será listado dentre os arquivos modificados.


Aula 09 – Criando um Repositório no Github

- Crie uma conta no GitHub
- Dentro do seu perfil, clicamos em 'New' no campo 'Repositories'




Após clicar em 'New' aparecerá a tela de definições do repositório:


Owner ^{*} Repository name ^{*}

 danieltquadros ▾ /

Great repository names are short and memorable. Need inspiration? How about **fuzzy-funicular**?

Description (optional)

☒  **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.](#)

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

[Create repository](#)

O **Owner** já vem preenchido. Indicando o proprietário do repositório. Ao lado damos um **nome para este repositório**.

Obs.: O nome do repositório deve ter letras minúsculas, sem espaço, sem caracteres especiais. Podemos utilizar nomes já existentes em outros usuário, pois o GitHub leva em consideração o nome do proprietário para diferenciar um repositório de outro. Não podemos utilizar nome repetido que nós mesmo já estivermos utilizando apenas.

Abaixo, podemos adicionar uma descrição. O campo é opcional, porém é muito recomendável a sua utilização.


Mais abaixo, precisamos definir se o nosso repositório será **público**. Para que qualquer pessoa possa ver, ou **privado**, para o meu acesso exclusivo.

Nos campos abaixo, é oferecido a opção de criarmos o nosso repositório e adicionarmos um README, e o gitignore e se for o caso, escolhermos uma licença.

Aqui, na aula deixamos tudo desmarcado, já que o README já criamos no nosso projeto local e o gitignore não foi explicado ainda.

Após clicar em Create Repository, será exibida a tela abaixo, onde o endereço web do campo Quick setup é o endereço web do repositório recém criado.

Quick setup — if you've done this kind of thing before

 Set up in Desktop

 or

HTTPS

SSH

https://github.com/danieltquadros/cursogithubb7web.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# cursogithubb7web" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/danieltquadros/cursogithubb7web.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/danieltquadros/cursogithubb7web.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Aula 10 – Conectando Repositório Local ao Remoto


Para o Windows é necessário fazer o download do Git Bash



No terminal do Git Bash executamos o seguinte comando:
\$ ssh-keygen -t rsa -b 4096 -C "danieltquadros@hotmail.com"

Obs.: O e-mail utilizado aqui deve ser o mesmo e-mail utilizado na conta do GitHub

```
danie@EDGE-14 MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "danieltquadros@hotmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/danie/.ssh/id_rsa): .....
```

Após executarmos este comando, apenas pressionamos 'Enter' mais uma vez. O terminal pedirá para criarmos uma passphrase (frase senha), porém, aqui na aula, não criaremos. Apenas pressionaremos 'Enter' para confirmar e 'Enter' novamente



Name	Date modified	Type	Size
 id_rsa	5/26/2021 6:22 PM	File	4 KB
 id_rsa.pub	5/26/2021 6:22 PM	Microsoft Publish...	1 KB

```
C: > Users > danie > .ssh > id_rsa.pub  
1 ssh-rsa [redacted]  
[redacted] danielquadros@hotmail.com
```

SSH and GPG keys

New SSH key

Em 'New SSH key, daremos um título para a chave e colamos aquela chave que copiamos do arquivo 'id_rsa.pub'

SSH keys / Add new

Title

Daniel Edge-14

Key

ssh-rsa


Add SSH key

A partir deste momento, eu tenho a minha chave pública cadastrada no GitHub e a minha chave privada está no meu computador.

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



SSH

Daniel Edge-14

SHA256: [redacted]

Added on May 26, 2021

Never used — Read/write

Delete

Obs. Esta chave é feita uma para cada computador que eu utilizo, pois ele irá verificar a chave pública no GitHub com a chave privada no meu computador.

Feito esta etapa, voltamos para o repositório.

No repositório, vemos as duas opções abaixo:

...or create a new repository on the command line

```
echo "# cursogithubb7web" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/danielquadros/cursogithubb7web.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/danielquadros/cursogithubb7web.git
git branch -M main
git push -u origin main
```

Onde a **primeira opção** é para quando ainda não tivermos criado o repositório no nosso computador. A **segunda opção** é para quando o repositório já houver sido criado. Neste caso, utilizaremos a segunda opção.

Copiaremos essa segunda opção:

```
git remote add origin https://github.com/danielquadros/cursogithubb7web.git
git branch -M main
git push -u origin main
```

Este comando fará a ligação entre o repositório do meu computador (local) ao repositório remoto (GitHub)

Agora precisamos colar este comando no terminal, dentro da pasta do projeto:

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git remote add origin https://github.com/danielquadros/cursogithubb7web.git
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch -M main
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git push -u origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.18 KiB | 120.00 KiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/danielquadros/cursogithubb7web.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> █
```

git remote → Este comando mostra que temos um 'origin' adicionado. (Verificação)

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git remote
origin
```

git remote -v → Mostra com mais detalhes:

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git remote -v
origin https://github.com/danielquadros/cursogithubb7web.git (fetch)
origin https://github.com/danielquadros/cursogithubb7web.git (push)
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> █
```

Mostra o **fetch** e o **push**

fetch → É a capacidade que eu tenho de puxar o conteúdo remoto para o local.

push → é o contrário do **fetch**, nos dá a possibilidade de levamos o nosso conteúdo local para o remoto.

Aqui, neste caso, já executamos o **push** enviando para o GitHub o nosso projeto local.

Aula 11 – Fazendo Alterações no Repositório Remoto

Para Iniciarmos os exercícios desta aula, realizamos uma alteração no README e criamos o arquivo index.html.

Após estas modificações que realizamos no projeto, executamos um commit, porém as alterações realizadas e commitadas ainda não foram para o GitHub. Para isto, iremos utilizar o comando **push**

git push origin main → Utilizamos aqui, este comando para atualizar as informações que foram commitadas no GitHub.

Ao verificarmos no GitHub, podemos perceber que o README foi atualizado, porém o arquivo novo que criamos (index.html), não foi adicionado.

Para isto, devemos utilizar o **git add index.html** ou ainda, podemos executar o comando **git add -A** para garantir que, no caso, se houvesse mais arquivos novos, ele adicionasse todos de uma só vez:

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git commit -am "add index.html"
[main d54a17f] add index.html
1 file changed, 12 insertions(+)
create mode 100644 index.html
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 497 bytes | 124.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/danielqtquadros/cursogithubb7web.git
b0504c8..d54a17f main -> main
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Após isto, repetimos o comando **git push origin main** para adicionar o index.html no GitHub também.

Criamos a pasta 'js' e dentro dela criamos o arquivo 'script.js'. Além disso, realizamos uma alteração no index.html, apenas para praticar a sequencia de comandos:

git status → Para verificar as modificações que não foram commitadas.

git commit -am → Para Commitar todas as modificações e adicionar comentário no commit.

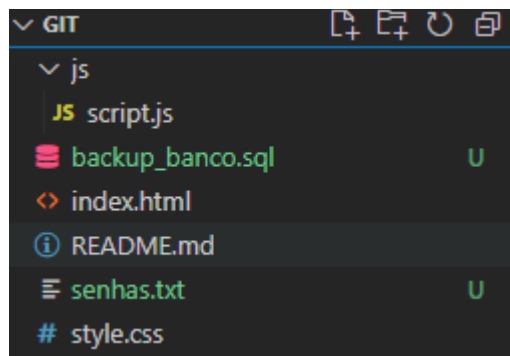
git status → Novamente, apenas para verificar se todas as alterações foram commitadas

git push origin main → Para adicionar todas as modificações no GitHub com o branch main.

Aula 12 – Ignorando Os Arquivos do Repositório

Vamos aprender nesta aula fazer com que o Git ignore um arquivo para que ele não seja colocado no Github, no caso em que quisermos que um determinado arquivo do nosso projeto não seja publicado no GitHub.

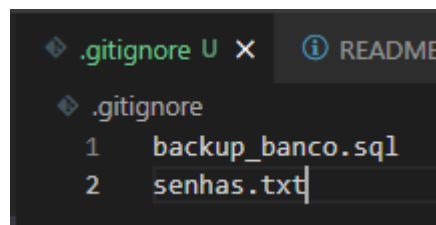
Para iniciarmos a aula, estamos criando na raiz do projeto dois arquivos chamados 'backup_banco.sql' e senhas.txt.



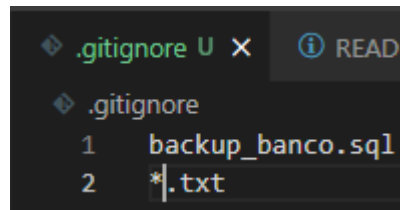
Se executarmos o `git status`, verificamos que estes dois arquivos aparecem como pendentes de commit.

Criaremos então um terceiro arquivo, o **.gitignore**

.gitignore → Dentro do gitignore colocamos tudo o que quisermos que seja ignorado apenas digitando nele o nome do arquivo desejado:



Podemos também ignorar os arquivos por extensão:



Neste caso, o ***.txt** manda ignorar qualquer arquivo que tenha a extensão **.txt**

A partir desta configuração, os arquivos adicionados no gitignore, não aparecem mais como pendentes de commit.

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> █
```

Realizado estas verificações, vamos adicionar o gitignore no projeto:

git add -A

git commit -am "Adicionado o gitignore"

git push origin main

Obs.: Se quisermos, podemos ignorar pastas inteiras adicionando o seu nome no gitignore, por exemplo a pasta **js** com o comando **js/** ou ignorar todos os arquivos que estão dentro da pasta com o comando **js/***

Aula 13 – Reverter Sem Perder o Código (revert)

git revert → Volta commitando

Qual a **diferença entre o git revert e o git reset**

git reset → Volta as alterações anteriores. Retorna o sistema a versão anterior

git revert → Volta as alterações anteriores e me dá acesso ao commit que fiz errado. Retorna o sistema para a versão anterior, porém, me devolve tudo o que eu fiz neste último commit que precisei reverter, permitindo assim que eu tenha acesso as últimas alterações que foram feitas entre o penúltimo e o último commit.

git revert é conhecido como o salvador das sextas-feiras

Vamos ao testes:

Adicionamos um texto qualquer no nosso README e no index.html e vamos fazer um commit

git add -A

git commit -am “Nova Funcionalidade X”

Aqui, imaginando que este último commit trouxe uma instabilidade ao nosso sistema, vamos utilizar o **git revert**.

Para utilizarmos esta função, precisamos executar o **git log** e copiarmos a chave do commit específico que precisamos fazer o revert.

O **git revert** tem uma funcionalidade chamada **--no-edit** ficando assim **git revert --no-edit**.

git revert --no-edit → Este comando retorna para o estado anterior retornando cada alteração realizada. Com o número do commit, executamos este comando assim

git revert --no-edit numerodocommit

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git revert --no-edit 48af94e1d113aa1833b773b7ed010bfcad932141
[main 11fcadb] Revert "Nova Funcionalidade X"
Date: Sun Jun 6 19:43:13 2021 -0300
2 files changed, 2 insertions(+), 4 deletions(-)
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Realizado este comando, eu posso acessar os arquivos que geraram a instabilidade e retomar a edição do projeto.

Aula 14 – Deletando Branches Locais e Remotos

No início deste aula, executamos o comando **git branch** apenas para visualizar o s branches que criamos ao decorrer do curso:

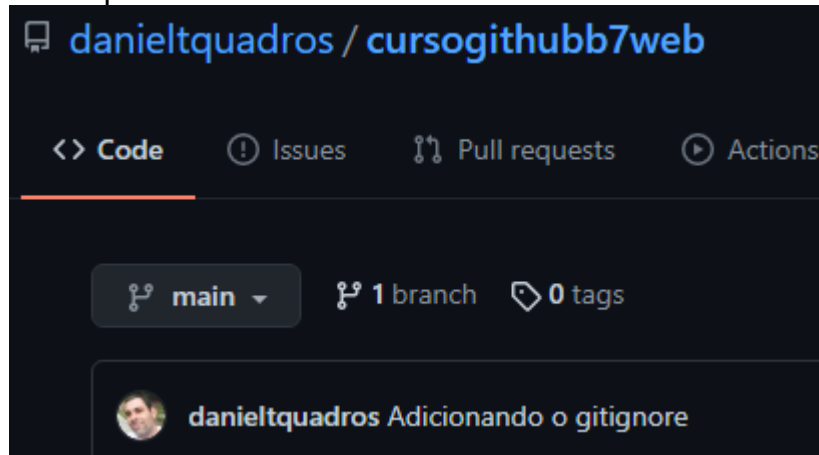
```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch
* main
teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

Percebemos aqui, que além do branch main, possuímos o branch teste

Lembrando também, que se executarmos o comando `git checkout teste`, por exemplo, passamos a utilizar o branch teste.

Obs.: É importante verificarmos que para que este branch teste esteja disponível no GitHub, eu preciso executar o push no branch teste.

Perceba que no GitHub consta apenas 1 branch:

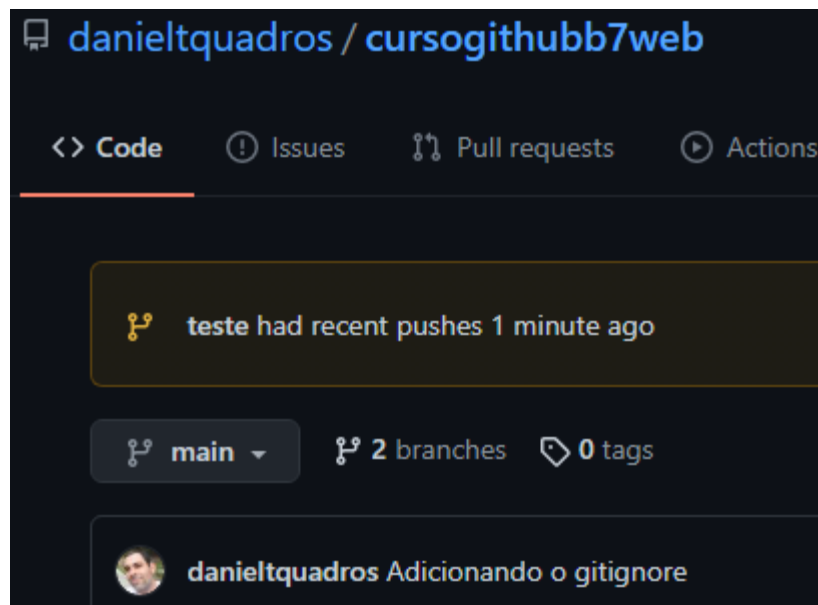


Para isto, acessaremos o branch teste e executaremos o push

git checkout teste
git push origin teste

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch
* main
  teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git checkout teste
Switched to branch 'teste'
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git push origin teste
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 113.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'teste' on GitHub by visiting:
remote:   https://github.com/danielquadros/cursogithubb7web/pull/new/teste
remote:
To https://github.com/danielquadros/cursogithubb7web.git
* [new branch]      teste -> teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> 
```

Agora, se olharmos o GitHub, podemos verificar que possuímos 2 branches, inclusive, apareceu a mensagem do push do branch teste:

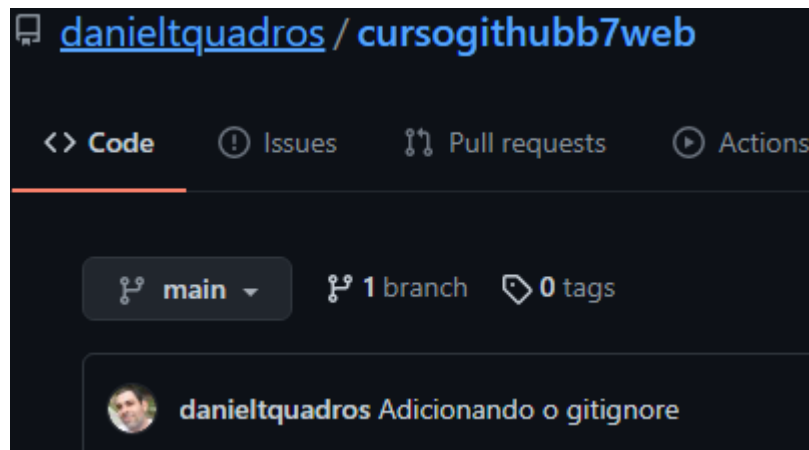


O próximo passo será aprendermos como removemos este branch remoto que acabamos de criar.

git push origin :teste → Adicionando os dois pontos(:) antes do nome do branch, removemos ele do GitHub:

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git push origin :teste
To https://github.com/danieltquadros/cursogithubb7web.git
- [deleted]      teste
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```

E no GitHub voltamos a ter apenas o branch main:



Obs.: Utilizando os (:) conseguimos deletar do nosso repositório remoto outros objetos também, como commits, por exemplo.

Prosseguiremos voltando a utilizar o nosso branch main:

git checkout main

Para deletarmos um branch do repositório local, utilizamos o (-D) e o nome do branch que queremos deletar, desta forma:

git branch -D teste → Este comando utiliza o D maiúsculo.

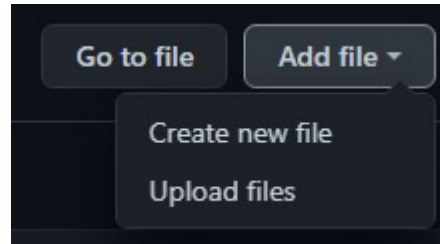
```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git branch -D teste
Deleted branch teste (was 0a3fcdf).
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```


Aula 15 – Puxando Alterações de Outras Pessoas (pull)

git pull → Este comando tem a função inversa do push, ou seja, ele traz os arquivos que estão no GitHub para dentro do projeto local (na minha máquina). Normalmente será utilizado quando for criado ou alterado, por mim ou por outro programador, um arquivo do projeto diretamente no GitHub ou em outra máquina que tenha realizado um push para o GitHub.

Para testarmos o comando pull, acessamos o nosso projeto no GitHub e criamos um arquivo chamado de teste.js:

Este comando



	danielquadros Criando o teste.js	e70c2d8 3 minutes ago	🕒 14 commits
js	Commit Aula 12 para adição do arquivo script.js	2 days ago	
.gitignore	Adicionando o gitignore	yesterday	
README.md	Revert "Nova Funcionalidade X"	yesterday	
index.html	Revert "Nova Funcionalidade X"	yesterday	
style.css	Segundo commit da aula 12 para adicionar o style.css	2 days ago	
teste.js	Criando o teste.js	3 minutes ago	

A sintaxe de sua execução é a seguintes

git pull origin main → **git pull** é o comando para trazer as alterações do remoto para o repositório local, **origin** é de quem estou trazendo e o **main** para qual branch estou trazendo.

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git> git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 681 bytes | 40.00 KiB/s, done.
From https://github.com/danielquadros/cursogithubb7web
 * branch                main          -> FETCH_HEAD
    11fcadb..e70c2d8    main          -> origin/main
Updating 11fcadb..e70c2d8
Fast-forward
 teste.js | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 teste.js
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Git>
```


Recomendação importante para quando estiver trabalhando em equipe. **Antes de executar um push, executar um pull.**

Aula 16 – Clonando Repositórios Remotos

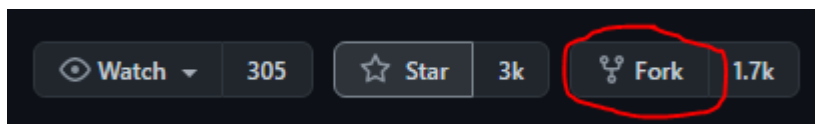
Clonando um projeto para o meu repositório local:

- 1º – Devemos copiar o link do repositório que desejamos clonar.
 - 2º – Abrimos o nosso terminal e navegamos até a pasta local onde queremos armazenar o clone deste projeto.
 - 3º – **git clone URLdoProjeto** → Comando que clona todos os arquivos do repositório remoto para o meu repositório local.
-

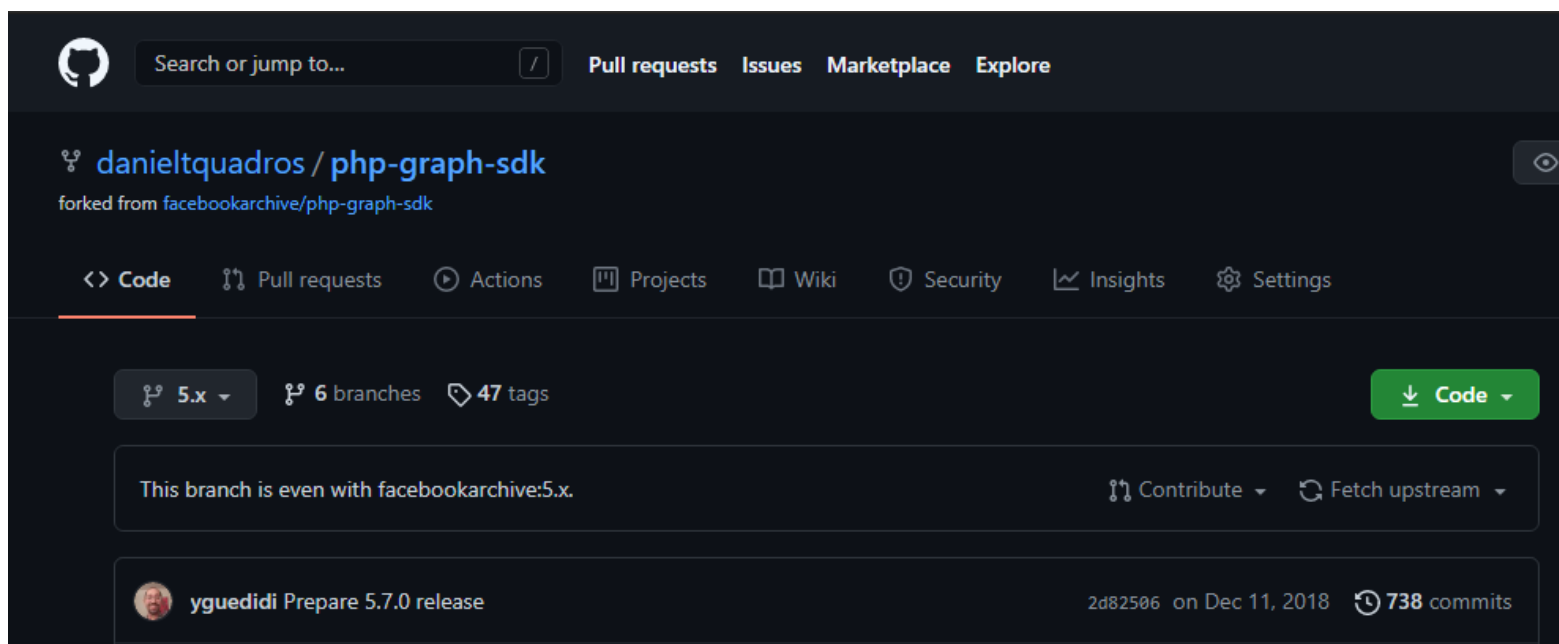
Aula 17 – Contribuindo Com Outros Repositórios (fork / pull request)

Quando queremos contribuir para um projeto que não nos pertence, precisamos executar um **fork**

Para executarmos um fork precisamos acessar o repositório que desejamos contribuir e clicarmos no botão 'fork'



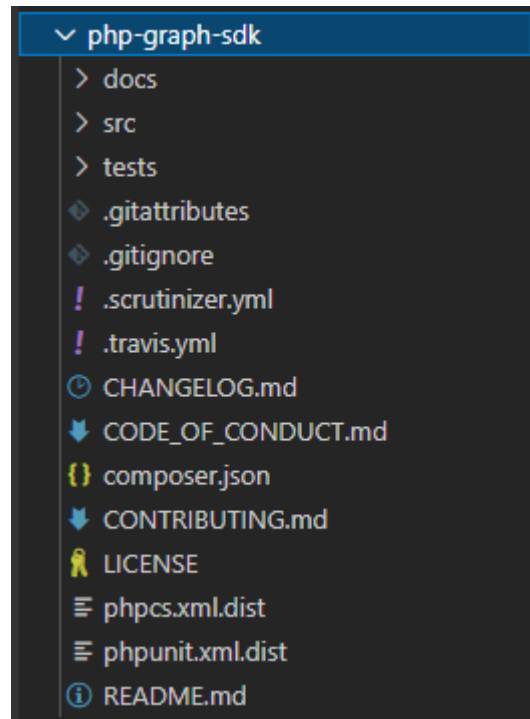
Este comando fará uma cópia do projeto para o meu repositório:



Podemos agora, copiar o link deste repositório que forkamos e clonarmos para o nosso repositório local:

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17> git clone https://github.com/danielquadros/php-graph-sdk
Cloning into 'php-graph-sdk'...
remote: Enumerating objects: 6996, done.
Receiving objects: 100% (6996/6996), 1.90 MiB | 2.07 MiB/s, done.96

Resolving deltas: 100% (4845/4845), done.
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17> █
```



No terminal, acessei a pasta do projeto clonado.

Apenas para teste, criei na raiz do projeto, um arquivo chamado de 'teste.html' e adicionei um comentário nele e executei o git status para visualizar.

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17> cd php-graph-sdk
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git status
On branch 5.x
Your branch is up to date with 'origin/5.x'.

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

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> █
```

git add -A
git status

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git add -A
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git status
On branch 5.x
Your branch is up to date with 'origin/5.x'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   teste.html

PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> █
```

git commit -m "Adicionado o arquivo teste.html"

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git commit -m "Adicionado o arquivo teste.html"
[5.x cecd3af] Adicionado o arquivo teste.html
1 file changed, 1 insertion(+)
create mode 100644 teste.html
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk>
```

Executamos o git branch para verificarmos que branch estamos. Verificamos que estamos num branch chamado de 5.x

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git branch
* 5.x
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk>
```

Para executarmos o push, precisamos saber qual o nome do servidor remoto do projeto, para isto, executamos o **git remote -v**:

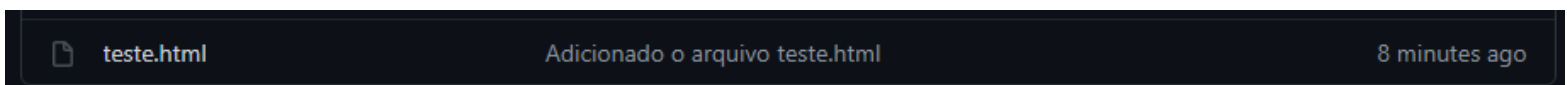
```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git remote -v
origin https://github.com/danielqtquadros/php-graph-sdk (fetch)
origin https://github.com/danielqtquadros/php-graph-sdk (push)
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk>
```

Verificamos que o nome do servidor é **origin**.

git push origin 5.x

```
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk> git push origin 5.x
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 326 bytes | 108.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/danielqtquadros/php-graph-sdk
2d82506..cecd3af 5.x -> 5.x
PS D:\Informática\Programação\2021\Daniel\Projetos GitHub\B7Web\Aula17\php-graph-sdk>
```

Verificamos no GitHub:



e podemos ver que o nosso arquivo teste.html foi para o repositório que nós forkamos.

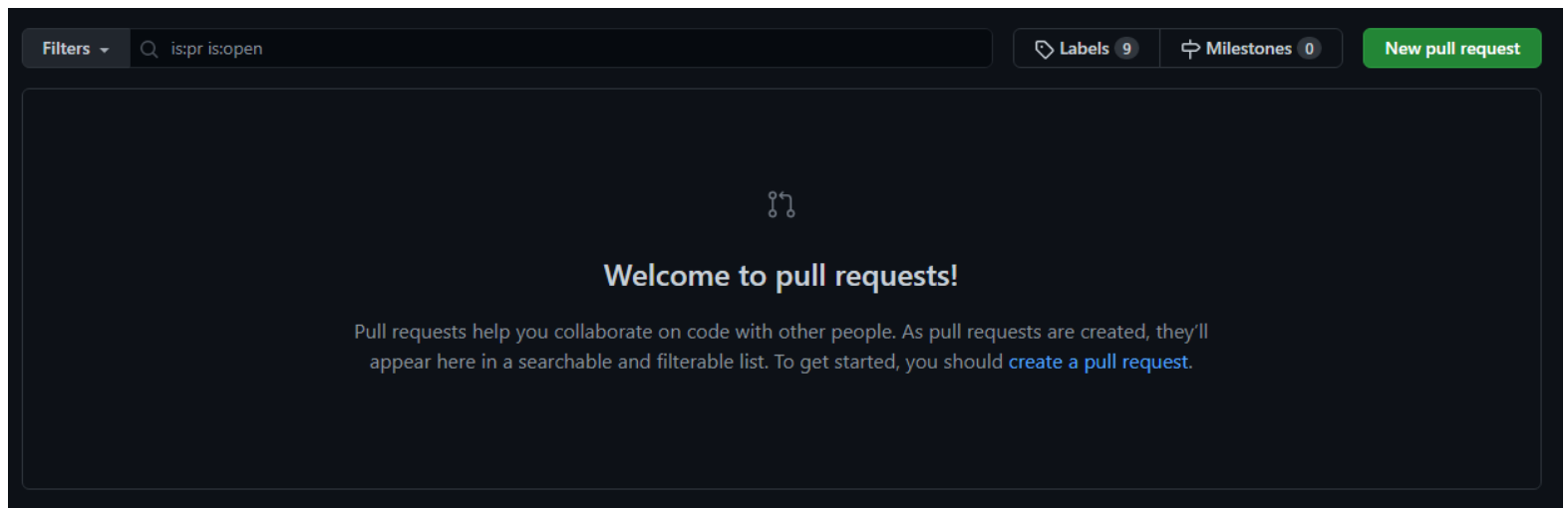
A partir daqui, como fazemos para que as alterações que eu realizei no projeto, possam fazer parte do projeto original.

Obs.: É importante entender que para que estas alterações sejam recebidas no projeto original, o proprietário do projeto original terá que aprovar a aceitação destas alterações.

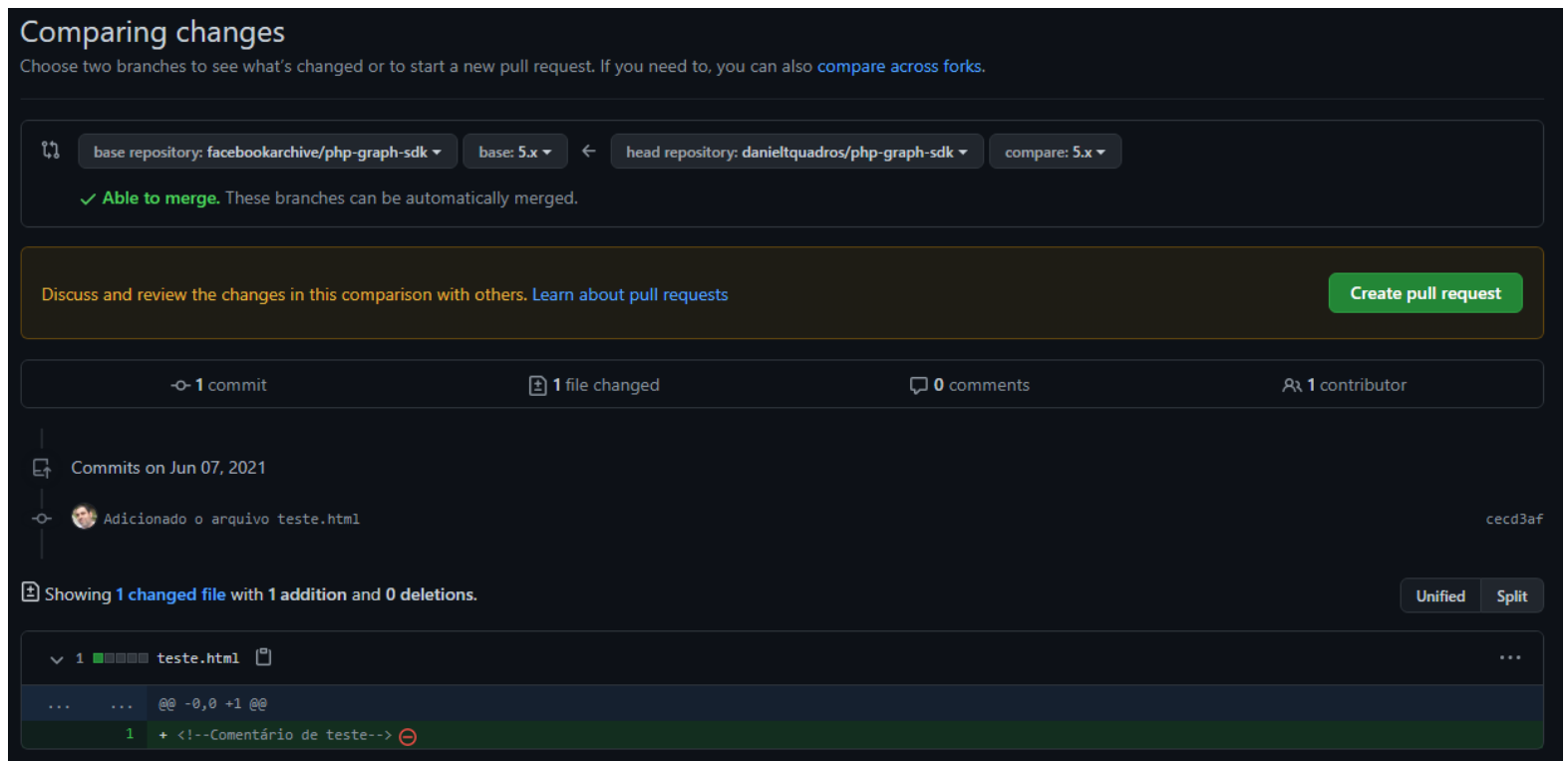
Para isto, acessamos o nosso projeto dentro do nosso GitHub e clicamos no botão Pull Request

<> Code ? Pull requests

Então veremos a seguinte tela:



Apenas clicando em ‘**New pull request**’, temos a seguinte tela contendo as informações do que foi alterado por mim.:



Clicando em **Create pull request**, abrirá uma outra página para que possamos explicar (argumentar) ao proprietário do projeto quais as alterações que foram realizadas, para que ele possa analisá-las, e se for do interesse dele, aprová-las.