



Pontifícia Universidade Católica de Minas Gerais

Trabalho Interdisciplinar I

Git & GitHub

Introdução

Rommel Vieira Carneiro / Wesley Dias Maciel

Prática 01

Git.

Sistema de controle de versão (Version Control System, VCS). Criado pelo Linus Torvalds.



“O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou em um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas. Mesmo que os exemplos desse livro mostrem arquivos de código fonte sob controle de versão, você pode usá-lo com praticamente qualquer tipo de arquivo em um computador”.

(Scott Chacon e Ben Strub, 2 Edição, 2014)

1) Página: <https://git-scm.com/>.



The screenshot shows the official Git website. At the top, the Git logo is followed by the tagline "--distributed-even-if-your-workflow-isnt". A search bar is on the right. The main content area describes Git as a free and open source distributed version control system, highlighting its speed, efficiency, ease of learning, and tiny footprint. It mentions that Git outclasses other SCM tools like Subversion, CVS, Perforce, and ClearCase. A diagram on the right illustrates the distributed nature of Git with multiple stacks of code blocks connected by lines. Below this, there's a section for learning Git in the browser for free with Try Git. The bottom section features links to About, Documentation, Downloads, and Community. A monitor icon displays the latest source release (2.14.2) and release notes. Further down, there are links for Windows GUIs, Tarballs, Mac Build, and Source Code. The footer lists various companies and projects using Git, including Google, Facebook, Microsoft, Twitter, LinkedIn, Netflix, PostgreSQL, Android, Rails, Qt, GNOME, Eclipse, and Xfce. It also mentions that the site is hosted on GitHub and that Git is a member of the Software Freedom Conservancy.

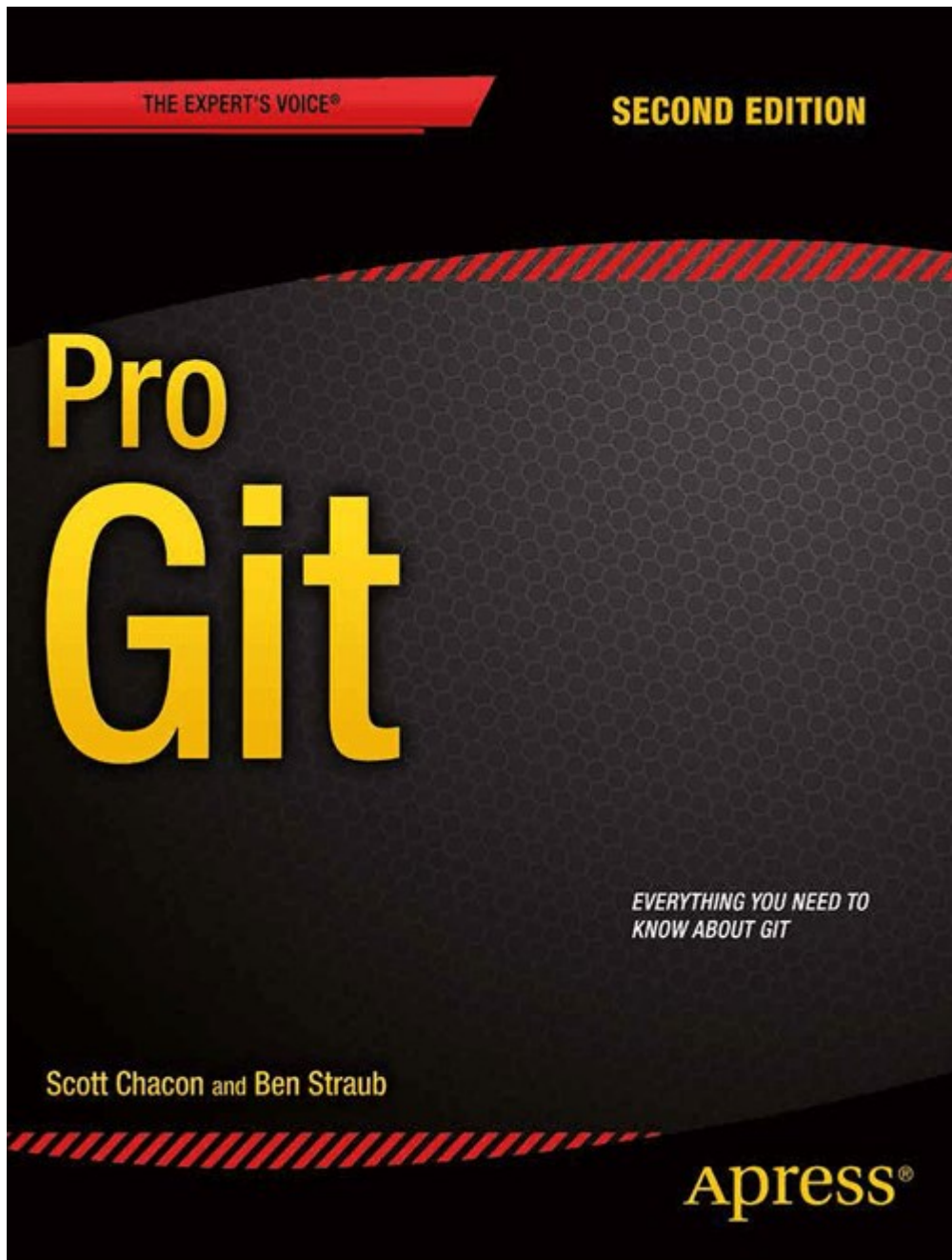
2) Download para Windows: <https://git-scm.com/download/win>.

The screenshot shows the 'Downloading Git' page on the official Git website. The page has a light gray background with a white content area. At the top left is the Git logo and tagline '--fast-version-control'. A search bar is at the top right. A left sidebar contains links: About, Documentation, Blog, Downloads (with sublinks for GUI Clients and Logos), and Community. Below these is a box about the 'Pro Git' book. The main content area is titled 'Downloading Git' and features a large downward arrow icon. Text indicates the latest download is the 64-bit version of Git for Windows, released 6 days ago. It provides a link to download manually if the automatic download fails. Below this, it lists other download options: Git for Windows Setup (32-bit and 64-bit), Git for Windows Portable (32-bit and 64-bit), and the source code. A 'Now What?' section follows, suggesting users read the book, download a GUI, or get involved in the community, each with an icon and brief description. At the bottom, it notes the site is open-sourced on GitHub and Git is a member of the Software Freedom Conservancy.

3) Instalar o Git, seguindo os passos do instalador.

4) Livro:

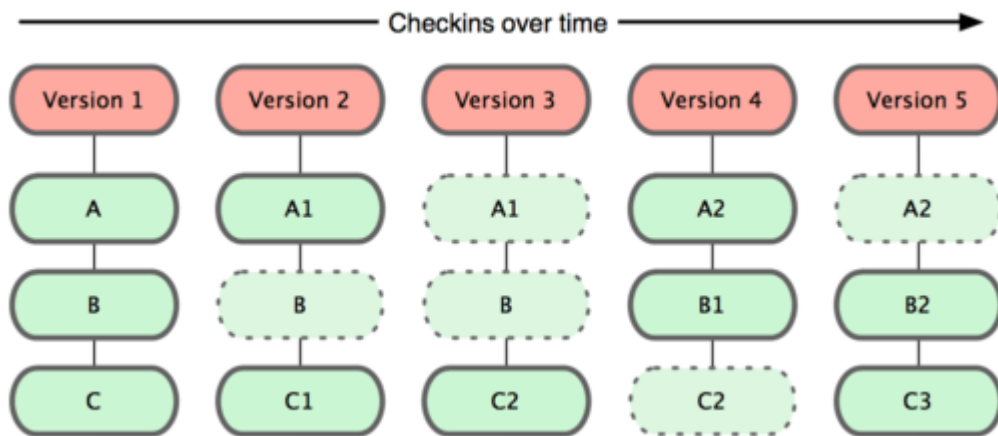
The screenshot shows the 'Pro Git' book page. The header features the Git logo and tagline '--local-branching-on-the-cheap', along with a search bar. The left sidebar lists navigation links: About, Documentation (with sublinks for Reference, Book, Videos, and External Links), Blog, Downloads, and Community. Below these is a box stating the book is available in English and other languages, with partial translations in Arabic, Spanish, Indonesian, Italian, and Swahili. The main content area is titled 'Book' and describes the 'Pro Git' book by Scott Chacon and Ben Straub, published by Apress. It mentions the Creative Commons license and provides a link to Amazon for print versions. A table of contents is listed under '1. Getting Started', including sections from 'About Version Control' to 'Summary'. To the right is a book cover image for 'Pro Git' 2nd Edition (2014), with a note to switch to the 1st Edition.



5) Link para o livro em português: <https://git-scm.com/book/pt-br/v1/Primeiros-passos>.

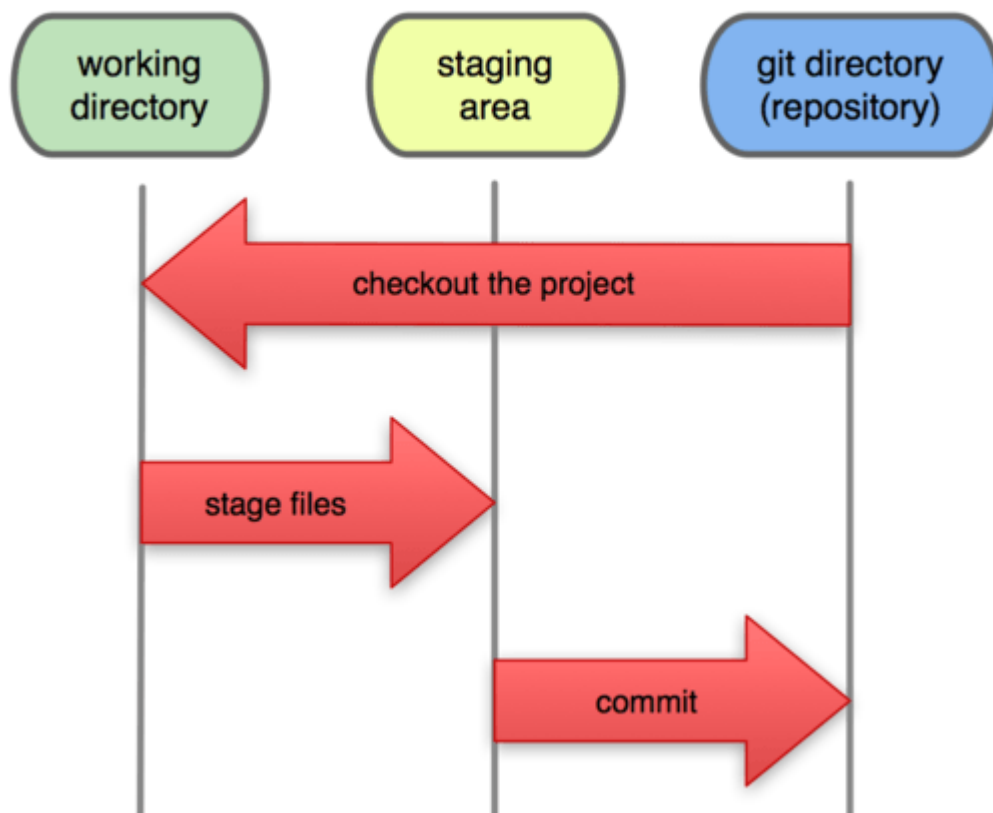


6) Git armazena dados como “snapshots” do projeto ao longo do tempo.



7) Os três estados: diretório de trabalho, área de preparação (“staging area”) e o diretório do Git.

Local Operations



Diretório do Git:

- 1) Local onde o Git armazena os metadados e o banco de objetos de seu projeto.
- 2) Essa é a parte mais importante do Git.
- 3) É a parte copiada quando você clona um repositório de outro computador.

Diretório de trabalho:

- 1) Uma versão do projeto que está sendo trabalhada.
- 2) Esses arquivos são obtidos a partir da base de dados comprimida no diretório do Git e colocados em disco para que você possa utilizar ou modificar.

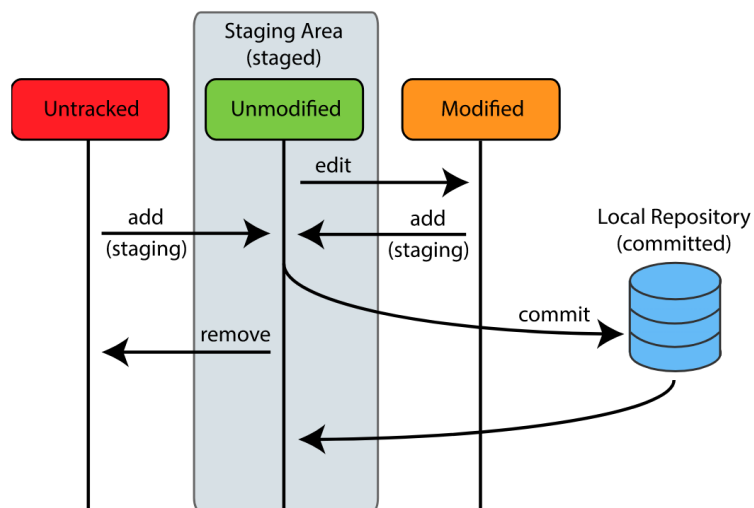
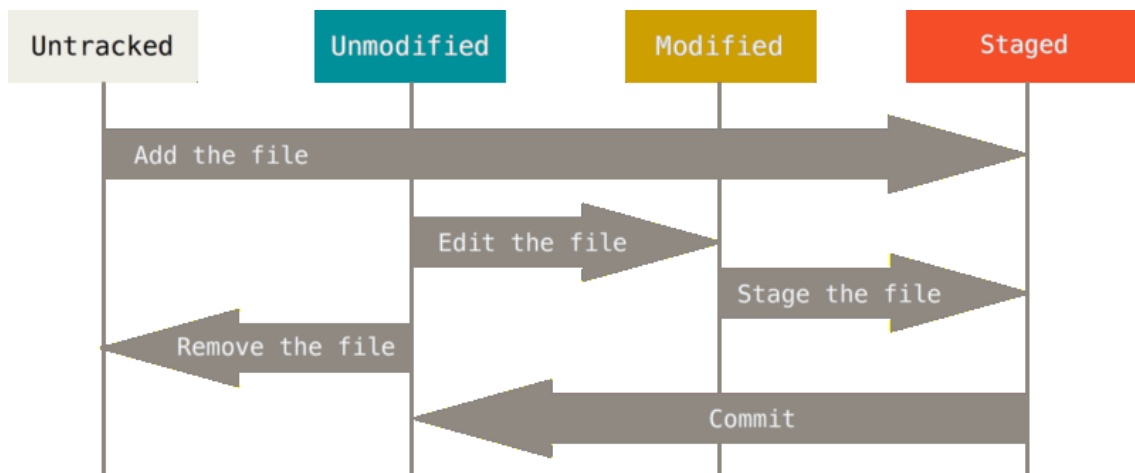
“Stage área”:

- 1) É um arquivo, geralmente armazenado no diretório do Git.
- 2) Armazena informação sobre o que entrará no próximo “commit”.
- 3) Algumas vezes, é chamado de índice, “index”, “stage” ou “staging area”.

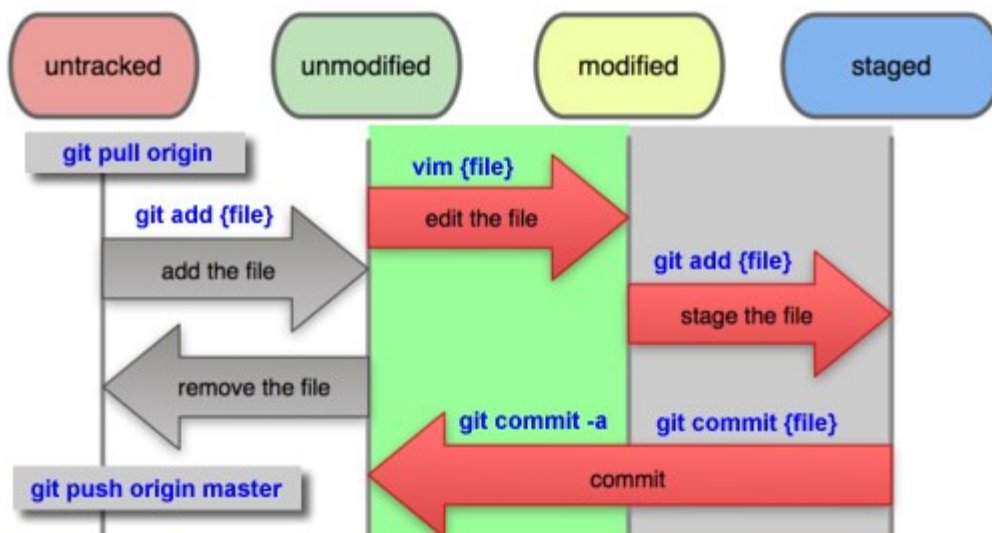
Fluxo básico do Git:

- 1) Você modifica arquivos no seu diretório de trabalho.
 - 2) Você seleciona os arquivos, adicionando snapshots deles para a “staging area”.
 - 3) Você faz um commit, que leva os arquivos como eles estão na “staging area” e os armazena permanentemente no seu diretório do Git.
-
- 1) Se uma versão de um arquivo está no diretório do Git, ela é considerada “committed”.
 - 2) Se uma versão de um arquivo foi adicionada à “staging area”, ela é considerada “staged”.
 - 3) Se uma versão de um arquivo foi alterada, mas não foi adicionada à “staging area”, ela é considerada modificada.

8) Ciclo de vida dos arquivos.



File Status Lifecycle



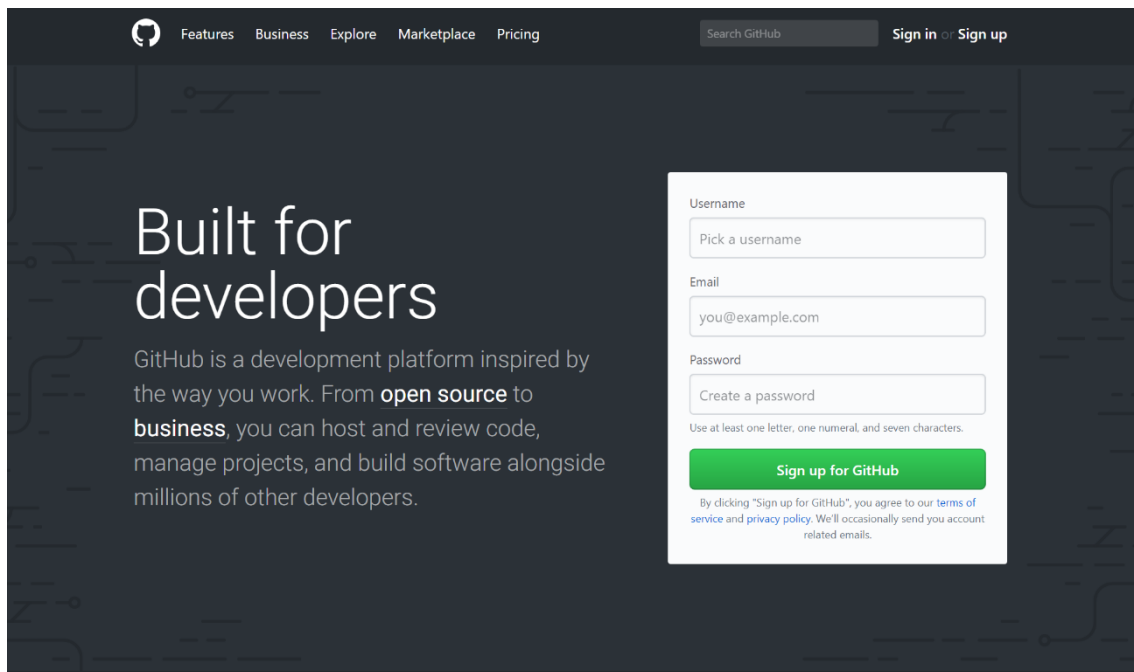
- 1) **untracked**: arquivo adicionado ao projeto, mas ainda não rastreado pelo Git. O arquivo ainda não é conhecido pelo Git, não faz parte de repositório. O arquivo não monitorado pelo Git.
- 2) **unmodified**: arquivo adicionado ao Git e ainda não modificado. O arquivo faz parte do repositório e não foi alterado.
- 3) **modified**: arquivo do repositório que foi modificado.
- 4) **staged**: arquivos que farão parte de uma nova versão, quando o usuário executar o próximo comando “commit”. Esses arquivos são considerados “unmodified”.
- 5) **committed**: arquivos que fazem parte de uma versão.

Prática 02

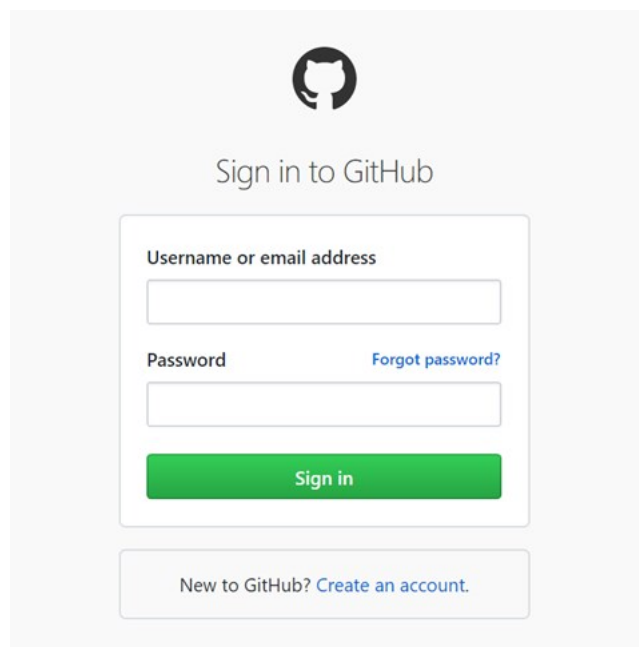
GitHub.

Repositório remoto, público e privado (pago). Plataforma de hospedagem (de código-fonte) que usa o Git como sistema de controle de versão. Local na Web para armazenamento de projetos. Também usado como rede social.

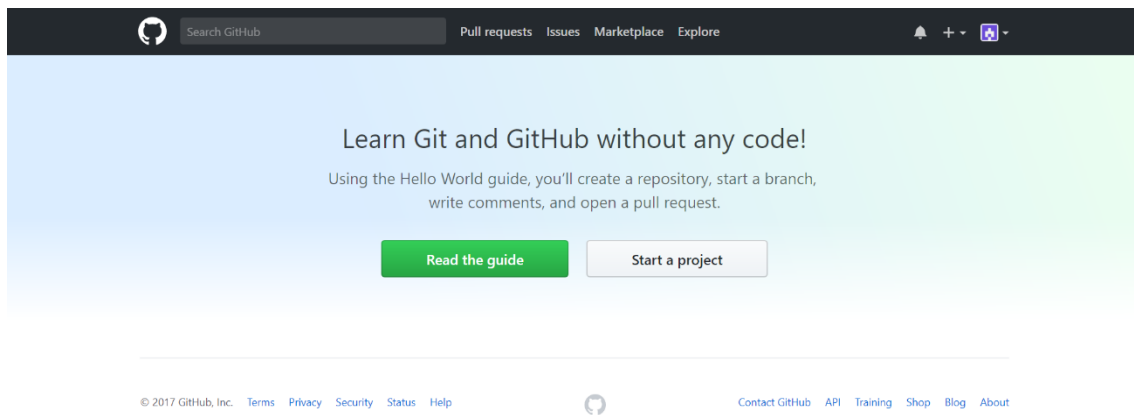
1) Página: <https://github.com/>.



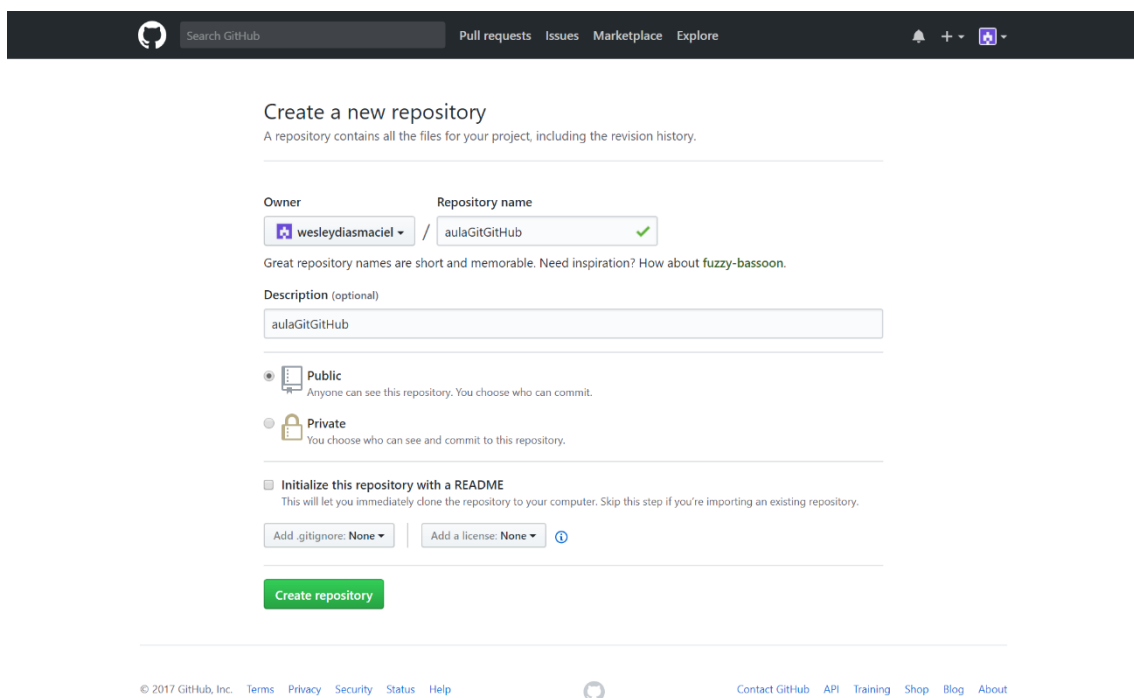
2) Login: <https://github.com/login>.



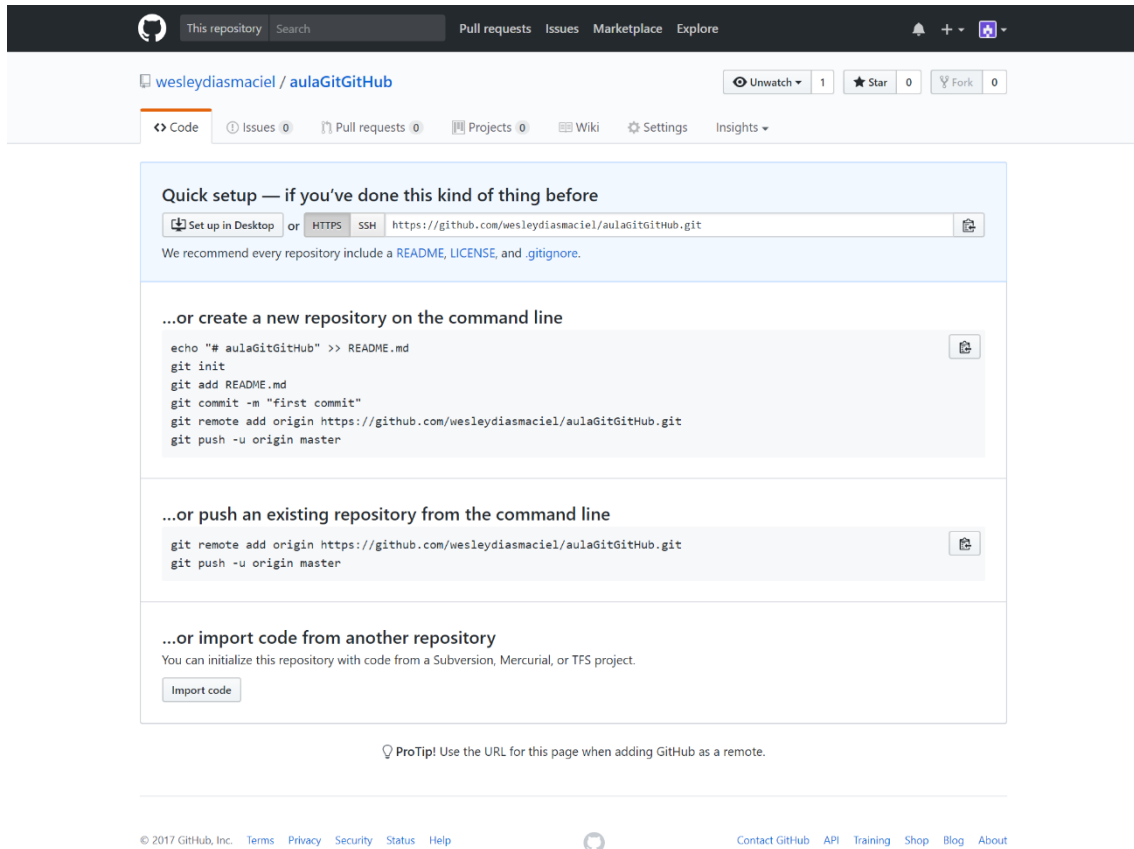
3) Página inicial:



4) Página de criação de novo projeto: <https://github.com/new>.



5) Página inicial do repositório: <https://github.com/wesleydiasmaciell/aulaGitGitHub>.




This repository Pull requests Issues Marketplace Explore

wesleydiasmaciell / aulaGitGitHub Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

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

 Set up in Desktop or

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

```
echo "# aulaGitGitHub" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/wesleydiasmaciell/aulaGitGitHub.git
git push -u origin master
```


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


```
git remote add origin https://github.com/wesleydiasmaciell/aulaGitGitHub.git
git push -u origin master
```

...or import code from another repository

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

[Import code](#)

 **ProTip!** Use the URL for this page when adding GitHub as a remote.

© 2017 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Help](#)  [Contact GitHub](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)

Prática 03

Git: configurações iniciais.

- 1) Verificar a instalação local, reportando a versão do Git instalada.

```
$ git --version
```

```
git version 2.14.1.windows.1
```

git config

Configurações para todo o sistema, todos os usuários: informar o parâmetro “--system”.

Configurações para o usuário corrente: informar o parâmetro “--global”.

Configurações locais do projeto: não informar parâmetro, deixar em branco.

- 2) Definir nome, e-mail e editor de texto padrão do usuário.

```
$ git config --global user.name "Wesley Dias Maciel"
```

```
$ git config --global user.email "wesleydiasmaciел@gmail.com"
```

```
$ git config --global core.editor notepad
```

- 3) Exibir as configurações fornecidas.

```
$ git config user.name
```

```
Wesley Dias Maciel
```

```
$ git config user.email
```

```
wesleydiasmaciел@gmail.com
```

```
$ git config core.editor
```

```
notepad
```

```
$ git config --list
```

```
core.symlinks=false
```

```
core.autocrlf=true
```

```
core.fscache=true
```

```
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.required=true
filter.lfs.process=git-lfs filter-process
credential.helper=manager
user.name=Wesley Dias Maciel
user.email=wesleydiasmaciel@gmail.com
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
core.editor=notepad
```

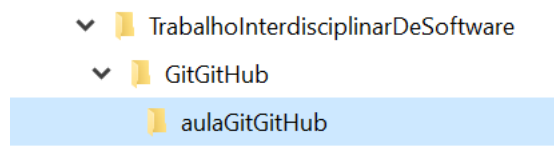
Prática 04

Git: iniciar um repositório local.

git init

Responsável por iniciar o repositório e por observar as alterações realizadas no projeto. Gera o diretório .git, que é o diretório do Git.

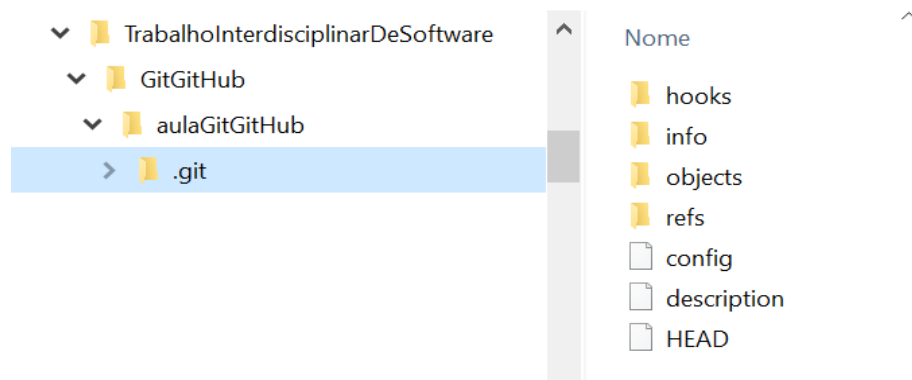
- 1) Criar um diretório para o projeto e ir para esse diretório, no caso “aulaGitGitHub”.



- 2) Iniciar o repositório.

```
$ git init
```

Initialized empty Git repository in \$
TrabalhoInterdisciplinarDeSoftware/GitGitHub/aulaGitGitHub/.git/



git status

- 3) Reportar o estado do repositório.

```
$ git status
```

On branch master

No commits yet

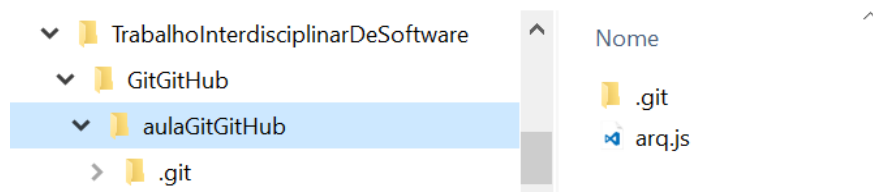
nothing to commit (create/copy files and use "git add" to track)

Prática 05

Git: adicionar arquivo a um repositório local.

- 1) Criar um arquivo com o conteúdo abaixo e salvá-lo com o nome `arq.js` no diretório do projeto.

```
function soma () {  
    var n1, n2;  
  
    n1 = document.getElementById ("primeiroNumero").value;  
    n2 = document.getElementById ("segundoNumero").value;  
}
```



- 2) Reportar o estado do repositório.

```
$ git status
```

On branch master

No commits yet

Untracked files:

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

arq.js

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

git add

- 3) Adicionar o arquivo na “staging area”.

```
$ git add arq.js
```


4) Reportar o estado do repositório.

```
$ git status
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: arq.js

5) Modificar o arquivo arq.js como apresentado abaixo.

```
function soma () {  
    var n1, n2;  
  
    n1 = document.getElementById ("primeiroNumero").value;  
    n2 = document.getElementById ("segundoNumero").value;  
  
    n1 = parseInt (n1);  
    n2 = parseInt (n2);  
}
```

6) Reportar o estado do repositório.

```
$ git status
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: arq.js

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: arq.js

7) Enviar o arquivo modificado para a “staging area”:

```
$ git add arq.js
```

8) Reportar o estado do repositório.

```
$ git status
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: **arq.js**

git commit

Responsável por criar uma imagem, snapshot. Cria uma versão.

9) Criar o primeiro “commit” do projeto.

```
$ git commit -m "Primeiro commit do projeto: ler dois inteiros informados pelo usuário."
```

[master (root-commit) 1a2a164] Primeiro commit do projeto: ler dois inteiros informados pelo usuário.

1 file changed, 9 insertions(+)

create mode 100644 arq.js

10) Reportar o estado do repositório.

```
$ git status
```

On branch master

nothing to commit, working directory clean

11) Modificar o arquivo arq.js como apresentado abaixo.

```
function soma () {  
    var n1, n2, soma;  
  
    n1 = document.getElementById ("primeiroNumero").value;  
    n2 = document.getElementById ("segundoNumero").value;  
  
    n1 = parseInt (n1);  
    n2 = parseInt (n2);  
  
    soma = n1 + n2;  
}
```

12) Reportar o estado do repositório.

```
$ git status
```

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: arq.js

no changes added to commit (use "git add" and/or "git commit -a")

Simulação de erro

13) Simular um erro: executar um “commit” sem executar um “add” antes, a “staging area” está vazia.

```
$ git commit -m "Simulação de um erro: executar um 'commit' sem executar um 'add' antes, a 'staging' area está vazia."
```

On branch master

Changes not staged for commit:

modified: arq.js

no changes added to commit

14) Enviar o arquivo modificado para a “staging area”:

```
$ git add arq.js
```

15) Reportar o estado do repositório.

```
$ git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: arq.js

16) Criar o segundo “commit” do projeto.

```
$ git commit -m "Segundo commit do projeto: somar os dois inteiros informados pelo usuário."  
[master 6e87bf3] Segundo commit do projeto: somar os dois inteiros informados pelo usuário.  
1 file changed, 3 insertions(+), 1 deletion(-)
```

17) Reportar o estado do repositório.

```
$ git status  
On branch master  
nothing to commit, working directory clean
```

Prática 06

Git: histórico dos “commits” do repositório.

git log

Responsável por exibir o histórico dos “commits” do repositório.

- 1) Exibir os “commits” do repositório.

```
$ git log
```

```
commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992 (HEAD -> master)
```

```
Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>
```

```
Date: Wed Oct 4 15:28:37 2017 -0300
```

Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

```
commit 1a2a164e12db88059c19a041ea70531068935430
```

```
Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>
```

```
Date: Wed Oct 4 14:48:48 2017 -0300
```

Primeiro commit do projeto: ler dois inteiros informados pelo usuário.

Filtros

O comando git log aceita vários filtros que facilitam pesquisar informação no histórico.

- 2) Realizar pesquisa por autor no histórico.

```
$ git log --author="Wesley"
```

```
commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992 (HEAD -> master)
```

```
Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>
```

```
Date: Wed Oct 4 15:28:37 2017 -0300
```

Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

```
commit 1a2a164e12db88059c19a041ea70531068935430
```

```
Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>
```

```
Date: Wed Oct 4 14:48:48 2017 -0300
```

Primeiro commit do projeto: ler dois inteiros informados pelo usuário.

- 3) Exibir o grafo do histórico.

```
$ git log --graph
```

```
* commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992 (HEAD -> master)
```

```
| Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>
```

```
| Date: Wed Oct 4 15:28:37 2017 -0300
```

Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

* **commit 1a2a164e12db88059c19a041ea70531068935430**

Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>

Date: Wed Oct 4 14:48:48 2017 -0300

Primeiro commit do projeto: ler dois inteiros informados pelo usuário.

```
* commit 9c8e9fd335381fe6a97708f7b3cd1d5acf670d2d
| Merge: 8aba87e... 6041ddd...
| Author: [redacted] <[redacted]>
| Date: Sun Jan 25 13:22:03 2009 -0500
|
| Fixing conflict!
|
| * commit 6041dddac354fff0feec911e75a575082d8addb8
| Author: [redacted] <[redacted]>
| Date: Sun Jan 25 13:10:23 2009 -0500
|
| Changing cutoff default
|
| * commit 8aba87e2e24744b7d1941e104b35033b9e2dbab5
| Author: [redacted] <[redacted]>
| Date: Sun Jan 25 13:16:04 2009 -0500
|
| Causing a merge on purpose
|
| * commit 670e3538533554d0643ca128428997c98eb5d54e
| Author: [redacted] <[redacted]>
| Date: Sun Jan 25 13:04:30 2009 -0500
|
| Adding cutoff method to string
```

git shortlog

Exibe um histórico resumido. Lista os “commits” agrupados por autores. A lista é ordenada por autor.

4) Exibir os “commits” agrupados por autor, resumidamente.

```
$ git shortlog
```

Wesley Dias Maciel (2):

Primeiro commit do projeto: ler dois inteiros informados pelo usuário.

Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

5) Exibir o número de “commits” por autor.

```
$ git shortlog -s
```

2 Wesley Dias Maciel

6) Exibir o número de “commits” por autor, ordenando pelo número de “commits”.

```
$ git shortlog -sn
```

2 Wesley Dias Maciel

```
$ git shortlog -sn
80 Harry
34 Samantha
3 Tom
```

git show

Exibe detalhes sobre um “commit”. É preciso informar o identificador do “commit”.

7) Apresentar os detalhes do “commit”: **6e87bf3c8cb4d59b1527a8971fed8c3218152992**.

```
$ git show 6e87bf3c8cb4d59b1527a8971fed8c3218152992
```

commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992 (HEAD -> master)

Author: Wesley Dias Maciel <wesleydiasmaciell@gmail.com>

Date: Wed Oct 4 15:28:37 2017 -0300

Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

```
diff --git a/arq.js b/arq.js
```

```
index 264e979..8a23e95 100644
```

```
--- a/arq.js
```

```
+++ b/arq.js
```

```
@@ -1,9 +1,11 @@
```

```
function soma () {
```

```
-   var n1, n2;
```

```
+   var n1, n2, soma;
```

```
    n1 = document.getElementById ("primeiroNumero").value;
```

```
    n2 = document.getElementById ("segundoNumero").value;
```

```
    n1 = parseInt (n1);
```

```
    n2 = parseInt (n2);
```

```
+
```

```
+   soma = n1 + n2;
```

```
}
```

```
PS C:\Users\wesle\Documents\Aulas\TrabalhoInterdisciplinarDeSoftware\Git\GitHub\aulaGit\GitHub> git show 6e87bf3c8cb4d59b1527a8971fed8c3218152992
commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992 (HEAD -> master)
Author: Wesley Dias Maciel <wesleydiasmaciell@gmail.com>
Date: Wed Oct 4 15:28:37 2017 -0300

    Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

diff --git a/arq.js b/arq.js
index 264e979..8a23e95 100644
--- a/arq.js
+++ b/arq.js
@@ -1,9 +1,11 @@
function soma () {
-   var n1, n2;
+   var n1, n2, soma;

    n1 = document.getElementById ("primeiroNumero").value;
    n2 = document.getElementById ("segundoNumero").value;

    n1 = parseInt (n1);
    n2 = parseInt (n2);
+
+   soma = n1 + n2;
}
PS C:\Users\wesle\Documents\Aulas\TrabalhoInterdisciplinarDeSoftware\Git\GitHub\aulaGit\GitHub>
```


Prática 07

git diff

Exibe as alterações, diferenças, do arquivo antes que um “commit” seja executado.

- 1) Reportar o estado do repositório.

```
$ git status
```

On branch master

nothing to commit, working directory clean

- 2) Modificar o arquivo arq.js como apresentado abaixo.

```
function soma () {  
    var n1, n2, soma;  
  
    n1 = document.getElementById ("primeiroNumero").value;  
    n2 = document.getElementById ("segundoNumero").value;  
  
    n1 = parseInt (n1);  
    n2 = parseInt (n2);  
  
    soma = n1 + n2;  
  
    document.getElementById ("resposta").innerHTML = soma;  
}
```

- 3) Reportar o estado do repositório novamente.

```
$ git status
```

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: arq.js

no changes added to commit (use "git add" and/or "git commit -a")

- 4) Exibir as diferenças, modificações do arquivo, antes de realizar o “commit”.

```
$ git diff
```

diff --git a/arq.js b/arq.js

index 8a23e95..dae2f5f 100644

--- a/arq.js

+++ b/arq.js

@@ -8,4 +8,6 @@ function soma () {

n2 = parseInt (n2);

soma = n1 + n2;

+

```
+ document.getElementById("resposta").innerHTML = soma;
}
```

5) Exibir apenas o nome dos arquivos modificados.

```
$ git diff --name-only
arq.js
```

6) Enviar o arquivo modificado para a “staging area”:

```
$ git add arq.js
```

7) Reportar o estado do repositório.

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   arq.js
```

8) Executar o comando “commit”.

```
$ git commit -m "Terceiro commit do projeto: apresentar o resultado da soma dos dois
números."
```

8) Exibir os “commits” do repositório.

```
$ git log
commit cc808fc5d859014762827b7908fba258c04971a4 (HEAD -> master)
Author: Wesley Dias Maciel <wesleydiasmaciел@gmail.com>
Date: Fri Oct 6 14:22:05 2017 -0300

    Terceiro commit do projeto: apresentar o resultado da soma dos dois números.

commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992
Author: Wesley Dias Maciel <wesleydiasmaciел@gmail.com>
Date: Wed Oct 4 15:28:37 2017 -0300

    Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

commit 1a2a164e12db88059c19a041ea70531068935430
Author: Wesley Dias Maciel <wesleydiasmaciел@gmail.com>
Date: Wed Oct 4 14:48:48 2017 -0300

    Primeiro commit do projeto: ler dois inteiros informados pelo usuário.
```

9) Apresentar os detalhes do terceiro “commit”:
cc808fc5d859014762827b7908fba258c04971a4.

```
$ git show cc808fc5d859014762827b7908fba258c04971a4

commit cc808fc5d859014762827b7908fba258c04971a4 (HEAD -> master)
```

Author: Wesley Dias Maciel <wesleydiasmaci@gmail.com>

Date: Fri Oct 6 14:22:05 2017 -0300

Terceiro commit do projeto: apresentar o resultado da soma dos dois números.

```
diff --git a/arq.js b/arq.js
index 8a23e95..dae2f5f 100644
--- a/arq.js
+++ b/arq.js
@@ -8,4 +8,6 @@ function soma () {
    n2 = parseInt (n2);

    soma = n1 + n2;
+
+ document.getElementById ("resposta").innerHTML = soma;
}
```

- 9) Executar novamente o comando “git diff”. Nenhuma alteração, diferença, é apresentada, pois o comando “commit” foi executado.

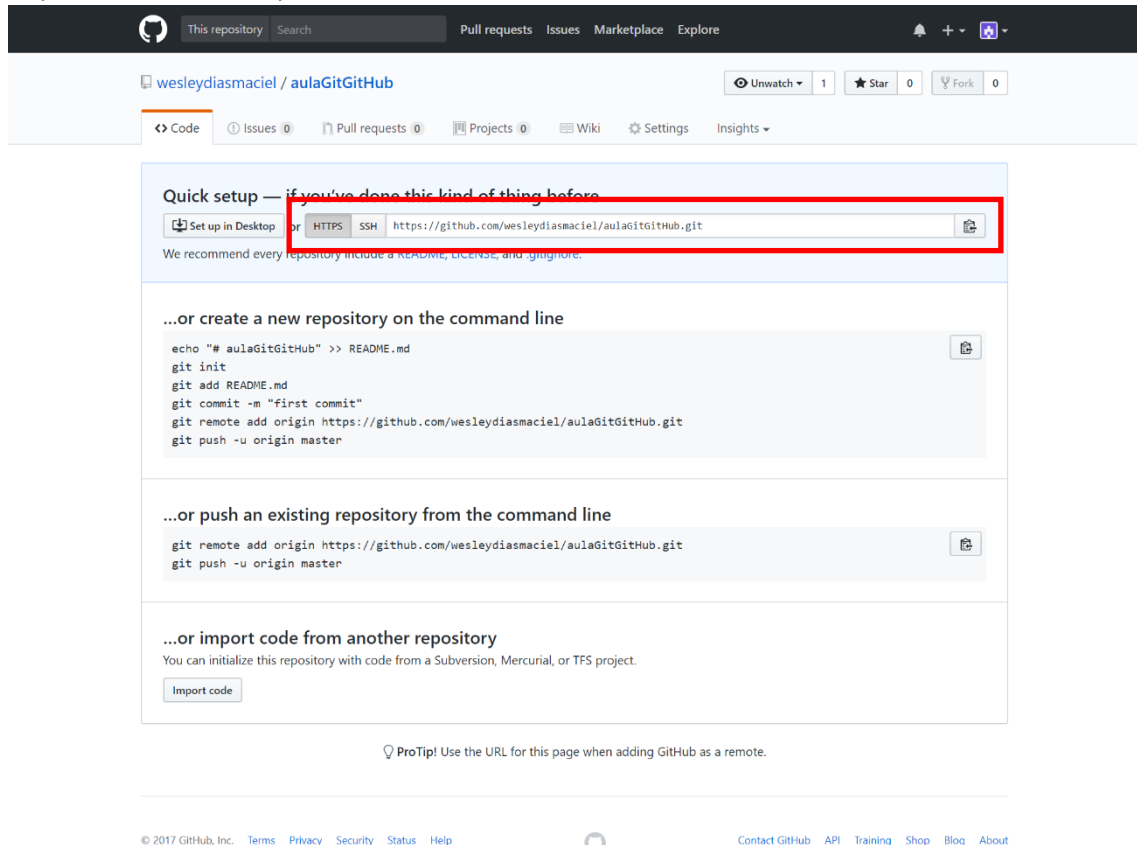
```
$ git diff
```

Prática 08

git remote

Associar um repositório local a um repositório remoto.

- 1) Copiar a URL do seu repositório remoto.



- 2) Associar o repositório local ao repositório remoto.

```
$ git remote add origin https://github.com/wesleydiasmaciell/aulaGitGitHub.git
```

A palavra “origin” pode ser substituída por qualquer outra palavra. A palavra escolhida, no caso “origin”, é um sinônimo para a URL do projeto remoto.

- 3) Listar a associação criada.

```
$ git remote  
origin
```

- 4) Listar detalhes sobre a associação criada.

```
$ git remote -v  
origin https://github.com/wesleydiasmaciell/aulaGitGitHub.git (fetch)  
origin https://github.com/wesleydiasmaciell/aulaGitGitHub.git (push)
```

Prática 09

git push

Enviar do repositório local para o repositório remoto.

- 1) Realizar envio do repositório local para o repositório remoto.

```
$ git push -u origin master
```

```
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 974 bytes | 243.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/wesleydiasmaciel/aulaGitGitHub.git
 * [new branch]    master -> master
Branch master set up to track remote branch master from origin.
```

-u: para não ter que informar “origin” e “mater” nas próximas vezes que o repositório local tiver que ser enviado para o repositório remoto.

origin: repositório remoto.

master: para enviar do “branch” “master” local, para o “branch” “master” remoto. Se o “branch” “master” remoto não existir, ele é criado. Equivale a:

```
$ git push -u origin master:master
```

Que significa:

```
$ git push -u origin <branch local>:<branch remoto>
```

```
git push origin master
```

Find a ref that matches `master` in the source repository (most likely, it would find `refs/heads/master`), and update the same ref (e.g. `refs/heads/master`) in `origin` repository with it. If `master` did not exist remotely, it would be created.

(Scott Chacon e Ben Strub, 2 Edição, 2014: <https://git-scm.com/docs/git-push>)

OBS:

“Este comando funciona apenas se você clonou de um servidor em que você tem permissão para escrita e se mais ninguém enviou dados nesse meio tempo. Se você e mais alguém clonarem o repositório ao mesmo tempo e você enviar suas modificações após a pessoa ter enviado as dela, o seu **push** será **rejeitado**. Antes, você terá que fazer um **pull** das modificações deste outro alguém e incorporá-las às suas, para que você tenha permissão para enviá-las”.

(Scott Chacon e Ben Strub, 2 Edição, 2014: <https://git-scm.com/docs/git-push>)

2) Atualizar a página do repositório remoto no navegador e observar as alterações.

This screenshot shows the main page of the GitHub repository `wesleydiasmacieli / aulaGitGitHub`. The repository has 3 commits, 1 branch, 0 releases, and 1 contributor. The `Code` tab is selected, showing the commit history. The latest commit is by `wesleydiasmacieli` with the message "Terceiro commit do projeto: apresentar o resultado da soma dos dois n..." and commit hash `cc88fc`, made 2 hours ago. Below the commit history, there is a button to "Add a README".

This screenshot shows the file view of the `arq.js` file in the `aulaGitGitHub` repository. The file is 291 bytes and contains 14 lines of JavaScript code. The code defines a `soma` function that takes two arguments, `n1` and `n2`, and returns their sum. The function is called with values from the `primeironumero` and `segundonumero` elements, and the result is displayed in the `resposta` element.

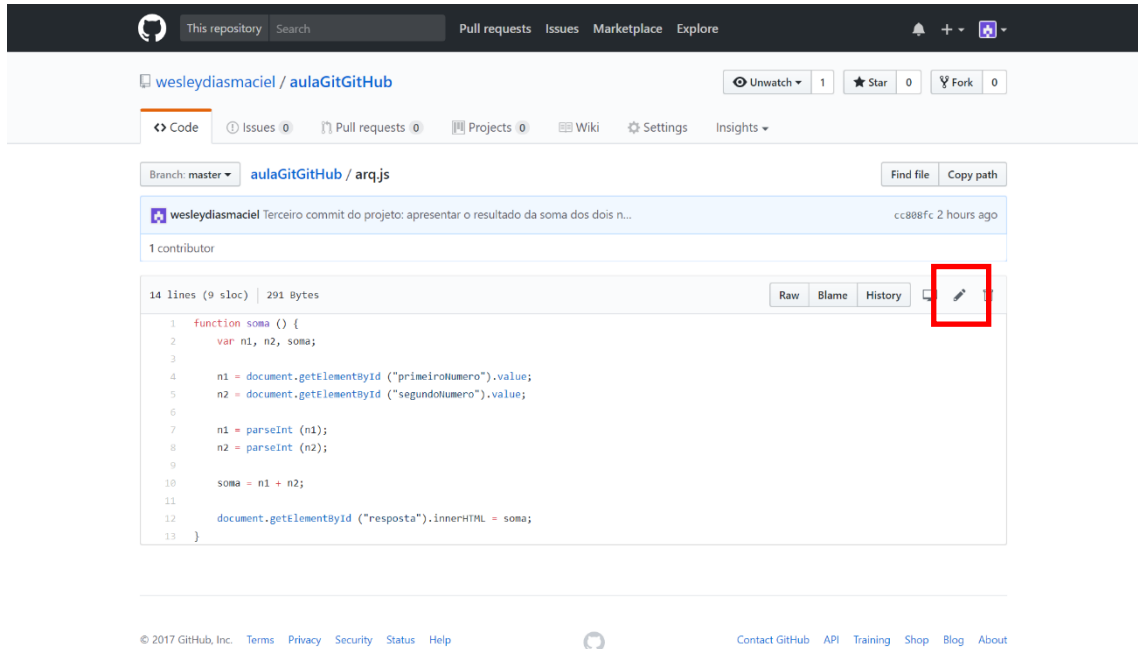
```
1 function soma () {  
2   var n1, n2, soma;  
3  
4   n1 = document.getElementById ("primeironumero").value;  
5   n2 = document.getElementById ("segundonumero").value;  
6  
7   n1 = parseInt (n1);  
8   n2 = parseInt (n2);  
9  
10  soma = n1 + n2;  
11  
12  document.getElementById ("resposta").innerHTML = soma;  
13 }
```

Prática 10

git fetch

Trazer o repositório remoto para o repositório local, sem realizar o “merge” dos “branches”.

- 1) Na página do repositório remoto, modifique o conteúdo do arquivo “arq.js”, conforme apresentado nas figuras abaixo:



aulaGitGitHub / arq.js or cancel

Edit file Preview changes Spaces 4 No wrap

```
1 function soma () {
2   var n1, n2, soma;
3
4   n1 = parseInt (document.getElementById ("primeironumero").value);
5   n2 = parseInt (document.getElementById ("segundotumero").value);
6
7   soma = n1 + n2;
8
9   document.getElementById ("resposta").innerHTML = soma;
10 }
11
```

Commit changes

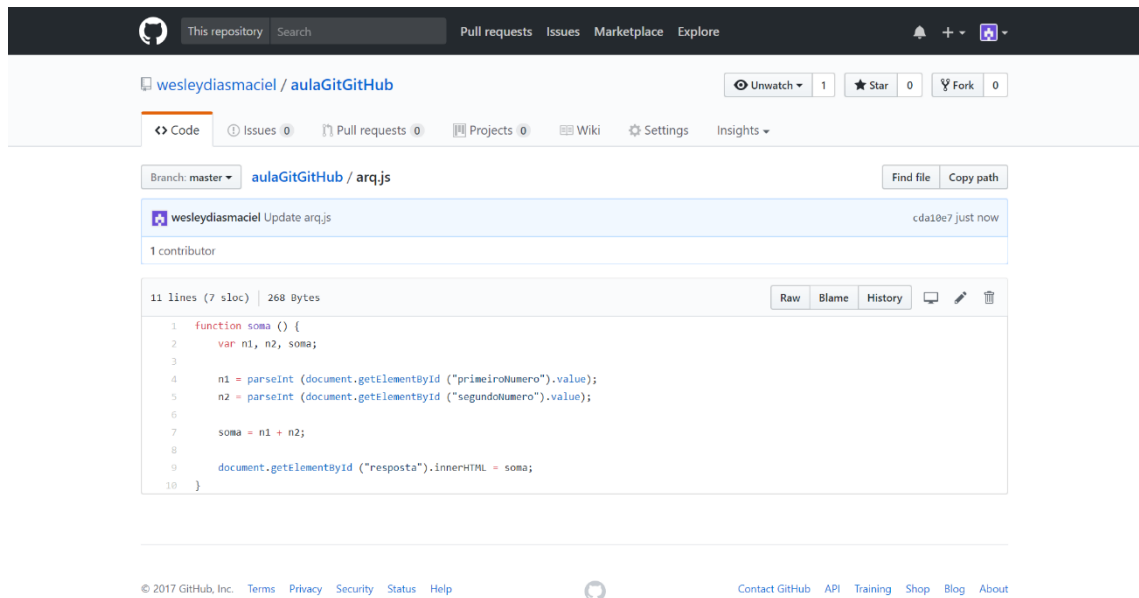
Update arq.js

Add an optional extended description...

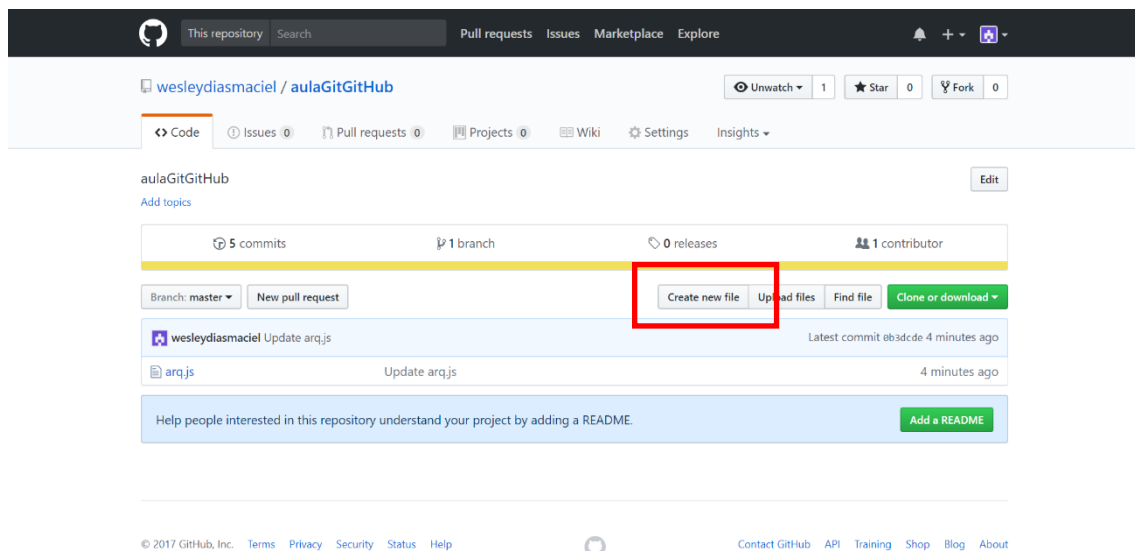
☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel




- 2) Ainda na página do repositório remoto, crie o arquivo “arq.css”, conforme apresentado nas figuras abaixo:



aulaGitGitHub / arq.css or cancel

Edit new file Preview Spaces 2 No wrap

```
1  html {
2    font-size: 16px;
3  }
```

 Commit new file

Create arq.css

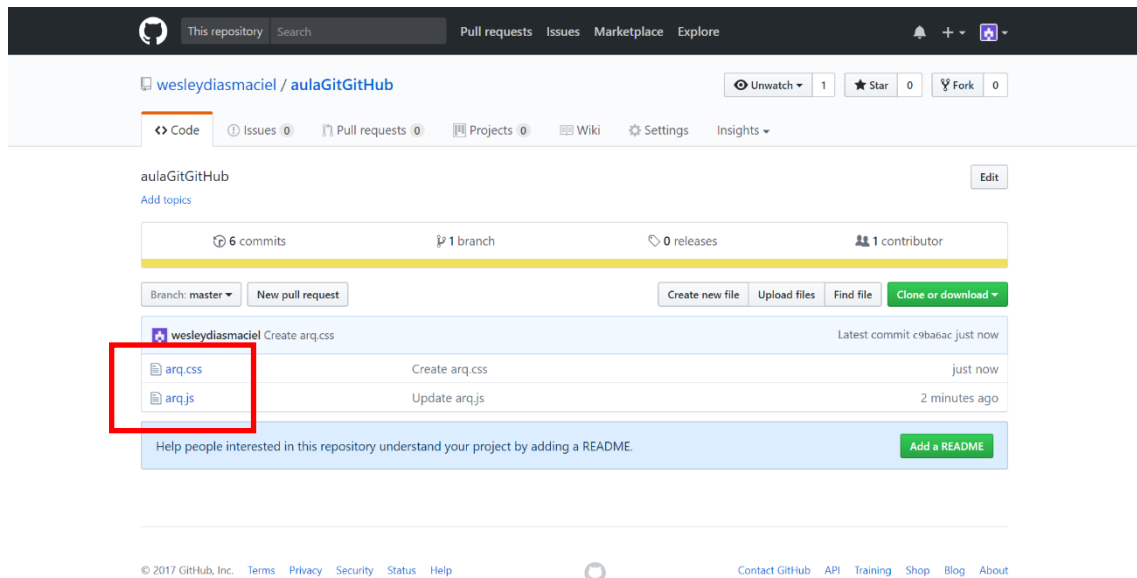
Add an optional extended description...

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file Cancel





3) Buscar o repositório remoto para o repositório local.

```
$ git fetch origin
```

```
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/wesleydiasmaciel/aulaGitGitHub
 0b3dcde..c9ba6ac master -> origin/master
```

4) Reportar o estado do repositório local.

```
$ git status
```

```
On branch master
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)

nothing to commit, working tree clean
```

OBS:

“(...) `git fetch origin` busca qualquer novo trabalho que foi enviado para esse servidor desde que você o clonou (ou fez a última busca). É importante notar que o comando `fetch` traz os dados para o seu repositório local — **ele não faz o merge automaticamente com os seus dados e não modifica o que você está trabalhando atualmente**. Você terá que fazer o merge manualmente no seu trabalho quando estiver pronto”.

(Scott Chacon e Ben Strub, 2 Edição, 2014: <https://git-scm.com/docs/git-push>)

Prática 12

git pull

Trazer o repositório remoto para o repositório local, realizando o “merge” dos “branches”.

- 1) Buscar o repositório remoto para o repositório local, realizando o “merge” dos “branches”.

```
$ git pull origin
```

```
Updating cc808fc..c9ba6ac
Fast-forward
 3 files changed, 4 insertions(+)
 create mode 100644 arquivo.css
```

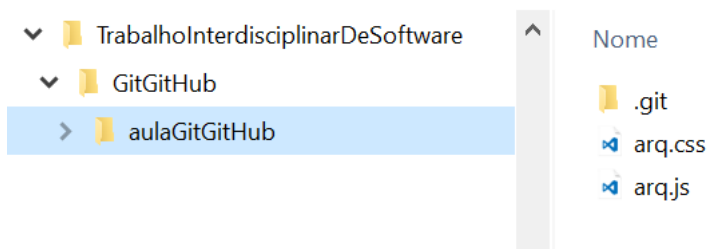
- 2) Reportar o estado do repositório local.

```
$ git status
```

```
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean
```

- 3) Alterações realizadas no diretório de trabalho:



- 4) Exibir histórico.

```
$ git log
```

```
commit c9ba6ac523db54840425257f01ad55b75d927aa1 (HEAD -> master, origin/master)
Author: wesleydiasmaciel <wesleydiasmaciel@gmail.com>
Date: Fri Oct 6 17:00:52 2017 -0300
```

```
Create arquivo.css
```

```
commit 0b3dcde029495afbc0f2f351584a6ed305547db8
Author: wesleydiasmaciel <wesleydiasmaciel@gmail.com>
Date: Fri Oct 6 16:58:52 2017 -0300
```

```
Update arquivo.js
```

commit cc808fc5d859014762827b7908fba258c04971a4

Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>

Date: Fri Oct 6 14:22:05 2017 -0300

Terceiro commit do projeto: apresentar o resultado da soma dos dois números.

commit 6e87bf3c8cb4d59b1527a8971fed8c3218152992

Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>

Date: Wed Oct 4 15:28:37 2017 -0300

Segundo commit do projeto: somar os dois inteiros informados pelo usuário.

commit 1a2a164e12db88059c19a041ea70531068935430

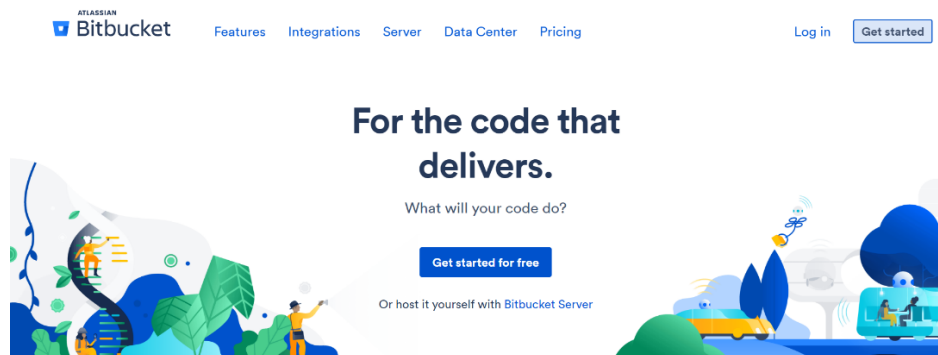
Author: Wesley Dias Maciel <wesleydiasmaciel@gmail.com>

Date: Wed Oct 4 14:48:48 2017 -0300

Primeiro commit do projeto: ler dois inteiros informados pelo usuário.

Prática 13

- 1) Pesquisar sobre o arquivo “.gitignore” do Git.
- 2) Pesquisar e testar os seguintes comandos do Git:
 - a) git-rm.
 - b) git checkout .
 - c) git reset --soft.
 - d) git reset --mixed.
 - e) git reset --hard.
 - f) git remote rename.
 - g) git remote remove.
 - h) git clone.
 - i) git commit –amend.
 - j) git branch.
 - k) git merge.
 - l) git rebase.
- 3) Pesquisar e testar os seguintes comandos do GitHub:
 - a) clone.
 - b) fork.
- 4) Pesquisar sobre o Bitbucket, <https://bitbucket.org/product>.



- 5) Pesquisar sobre o GitLab, <https://about.gitlab.com/>.

