

Módulo 2

Versionamento de código usando o Git

Gian Souza



Conteúdo do módulo

- Stage
- Commits
- Logs
- Histórico
- Branches
- Merge
- Github



Módulo 2 / Aula 1

Stage

Gian Souza



Conteúdo a aula



- **Comandos básicos para usar no terminal**
- **Configuração inicial do Git**
- **Criando um repositório**
- **Adicionando e removendo arquivos**

Comandos básicos para usar no terminal



Descrição do comando	Powershell	Prompt de Comando
Listar o conteúdo da pasta atual	ls	dir
Limpar o conteúdo do terminal	Ctrl + L ou cls	cls
Navegar entre pastas	cd ./pasta_destino	cd ./pasta_destino
Criar uma nova pasta	mkdir nome_pasta	mkdir nome_pasta
Criar um novo arquivo	echo "conteúdo" > destino.txt	echo "conteúdo" > destino.txt
Remover pasta	rm -r ./nome_pasta	rmdir ./nome_pasta
Remover arquivo	rm -r ./arquivo.txt	del arquivo.txt
Ver o conteúdo de um arquivo	cat ./arquivo.txt	more ./arquivo.txt

Configuração inicial do Git



No terminal, executar os comandos

```
git config --global user.name "Seu nome"  
git config --global user.email "Seu e-mail"
```

Criando um repositório Git



No terminal, dentro da pasta do projeto,
executar o comando

`git init .`

Adicionando e removendo arquivos



Ao se trabalhar com o Git, não pasta incluir ou deletar um conteúdo na pasta do projeto, é necessário adicionar essa mudança ao stage.

Podemos ver os arquivos que estão sendo considerados pelo Git, que estão no stage, através do comando:

```
git status
```

Adicionando e removendo arquivos



Para adicionar um arquivo ao stage do Git, dizer ao Git que ele deve considerar o arquivo, execute o comando:

`git add nome_do_arquivo`

E para remover o arquivo do stage:

`git rm --cached nome_do_arquivo`

Para adicionar todos os arquivos da pasta no stage:

`git add .`

Módulo 2 / Aula 2

Commits

Gian Souza



Conteúdo a aula

- Salvando alterações
- Padrões para nomear commits
- Changelog



Salvando alterações



Para salvar as alterações que foram adicionadas ao stage, usamos o comando:

git commit – irá abrir um editor de textos, dentro do terminal

Salvando alterações



```
GNU nano 6.2                               C:/git/.git/COMMIT_EDITMSG

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   deleted:    arquivo.txt
#   deleted:    arquivo2.txt
#



[ Read 9 lines ]
^G Help      ^O Write Out    ^W Where Is    ^K Cut        ^T Execute    ^C Location    M-U Undo    M-A Set Mark
^X Exit      ^R Read File    ^\ Replace     ^U Paste      ^J Justify    ^/ Go To Line  M-E Redo    M-6 Copy
```

Salvando alterações



Nesse editor vamos descrever a alteração que será salva, ex: remoção de arquivos: arquivo.txt e arquivo2.txt

Para salvar o texto usamos: Ctrl + O

Para sair do editor: Ctrl + X

Salvando alterações



Podemos salvar as alteração, criar um commit, sem ter que acessar esse editor, fazemos isso através do comando:

git commit –m “minha mensagem”

Salvando alterações



**Existe um caminho ainda mais curto,
onde podemos adicionar o arquivo ao
stage e já salvar a alteração, para isso
executamos o comando:**

`git commit –am “minha mensagem”`

Padrões para nomear commits



Existe um padrão que podemos seguir, ao escrever as mensagens para os commits.

Chore: usado para pequenas tarefas

`git commit -m "chore: removendo arquivo.txt"`

Fix: correções

`git commit -m "fix: correção no cálculo de médias"`

Feat: inclusão de funcionalidade

`git commit -m "feat: inclusão de função para calcular mediana"`

Docs: atualização de documentação

`git commit -m "atualizando o changelog.md"`

Changelog



Changelog é um arquivo onde registramos todas as mudanças que aconteceram em um projeto ao decorrer do tempo.

Módulo 2 / Aula 3

Logs

Gian Souza



Conteúdo a aula



- Visualizando o registro de commits realizados
- Variações na visualização de commits

Git log



Para visualizar o histórico de commits,
usamos o comando:

`git log`

```
PS C:\git> git log
commit 484d17034a7c68dcf5d2a77af631c15b4f3f9bf8 (HEAD -> master)
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:32:29 2022 -0300

    inclusão de changelog.md

commit b50c4485148c909f50ef97b3df07844ee19577c2
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:12:34 2022 -0300

    colocando acentuação no arquivo2.txt
```

Git log



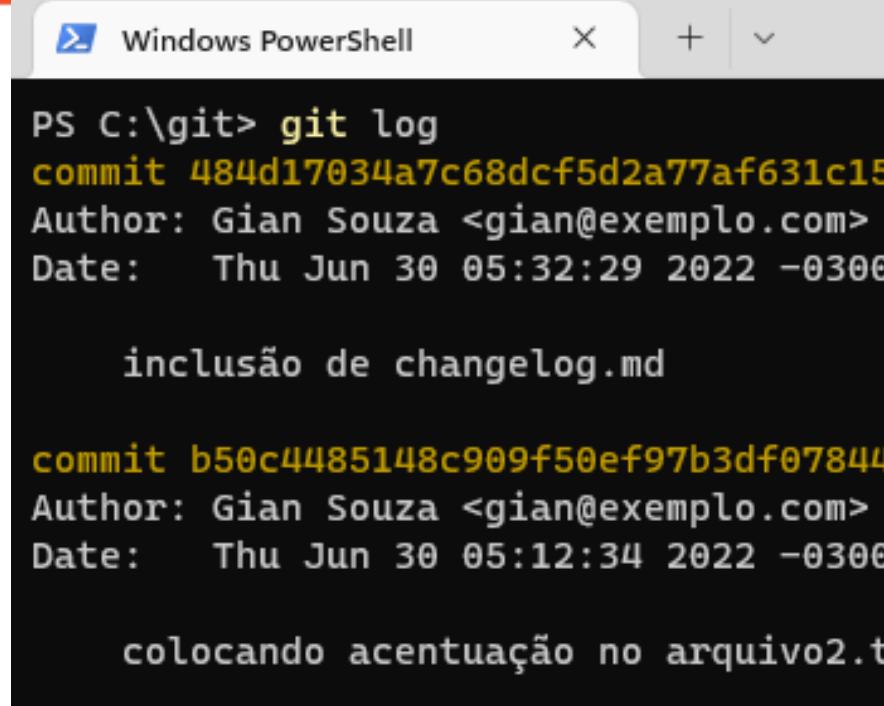
O registro de um commits é composto por:

Commit: identificador único do commit

Author: autor do commit

Date: data em que o commit foi feito

Mensagem informado no git commit



```
Windows PowerShell

PS C:\git> git log
commit 484d17034a7c68dcf5d2a77af631c15
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:32:29 2022 -0300

    inclusão de changelog.md

commit b50c4485148c909f50ef97b3df07844
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:12:34 2022 -0300

    colocando acentuação no arquivo2.t
```

Git log – variações



Para visualizar os últimos dois commits:

git log -n 2* (ou outro número)

```
PS C:\git> git log -n 2
commit 484d17034a7c68dcf5d2a77af631c15b4f3f9bf8 (HEAD -> master)
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:32:29 2022 -0300

    inclusão de changelog.md

commit b50c4485148c909f50ef97b3df07844ee19577c2
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:12:34 2022 -0300

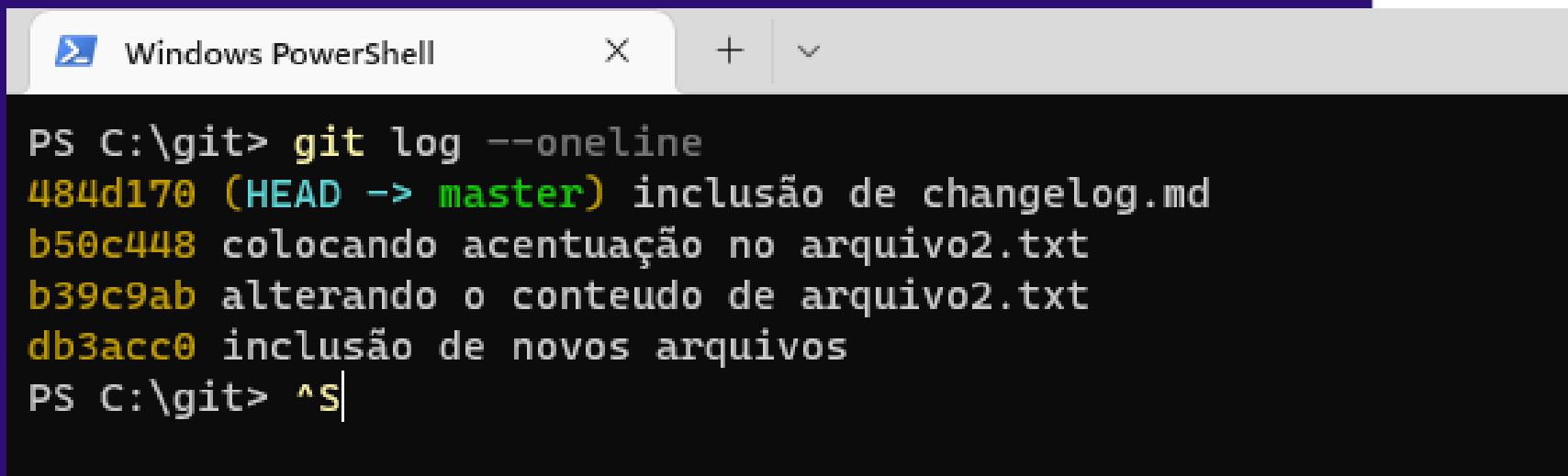
    colocando acentuação no arquivo2.txt
PS C:\git> |
```

Git log – variações



Para visualizar uma versão resumida do registro de commits:

`git log --oneline`



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command `git log --oneline` is run, displaying a summary of recent commits:

```
PS C:\git> git log --oneline
484d170 (HEAD -> master) inclusão de changelog.md
b50c448 colocando acentuação no arquivo2.txt
b39c9ab alterando o conteudo de arquivo2.txt
db3acc0 inclusão de novos arquivos
PS C:\git> ^S|
```

Git log – variações



Podemos visualizar o registro de uma forma mais detalhada solicitando o conteúdo da alteração, através do comando:

`git log --stat`

```
Windows PowerShell

PS C:\git> git log --stat
commit 484d17034a7c68dcf5d2a77af631c15b4f3f9bf8 (HEAD -> master)
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:32:29 2022 -0300

    inclusão de changelog.md

    changelog.md | 9 ++++++++
    1 file changed, 9 insertions(+)

commit b50c4485148c909f50ef97b3df07844ee19577c2
Author: Gian Souza <gian@exemplo.com>
Date:   Thu Jun 30 05:12:34 2022 -0300

    colocando acentuação no arquivo2.txt

    arquivo2.txt | Bin 34 -> 34 bytes
    1 file changed, 0 insertions(+), 0 deletions(-)
```

Git log – variações



É possível combinar as variações, solicitamos o conteúdo de dois registros com apenas o identificador do commit:

`git log --stat -n 2 --oneline`

```
Windows PowerShell
PS C:\git> git log --stat -n 2 --oneline
484d170 (HEAD -> master) inclusão de changelog.md
  changelog.md | 9 ++++++++
  1 file changed, 9 insertions(+)
b50c448 colocando acentuação no arquivo2.txt
  arquivo2.txt | Bin 34 -> 34 bytes
  1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\git> |
```

Módulo 2 / Aula 4

Histórico

Gian Souza



Conteúdo a aula

- Desfazendo alterações
- Navegando entre commits
- Revertendo commits



Desfazendo alterações



Podemos desfazer alterações em arquivos que ainda não foram adicionados ao stage, usando o comando:

```
git checkout meu_arquivo
```

Navegando entre commits



Com o Git podemos viajar no tempo, para o passado, através do comando git checkout, podemos voltar o conteúdo de um repositório para o conteúdo de um determinado commit

`git checkout COMMIT_ID`

Revertendo commits



O Git nos dá a possibilidade de reverter os commits, uma reversão significa que outro commit será criado desfazendo a alteração, se incluímos um arquivo no commit que queremos reverter, outro commit será criado removendo este arquivo.

Fazemos isso através do comando:

```
git revert HEAD COMMIT_ID
```

Módulo 2 / Aula 5

Branches

Gian Souza



Conteúdo a aula

- Visão geral
- Listando as branches
- Criando uma branch
- Convenções para nomear uma branch
- Navegando entre branches
- Deletando uma branch



Branches – Visão geral



Com o Git podemos criar versões paralelas do nosso projeto, fazemos isso no Git através da criação de branches, que pode ser traduzido como galhos, ramificação, então pensamos em nosso repositório como uma árvore que pode possuir diversos galhos.

Com essa funcionalidade podemos fazer experimentos e trabalhar em equipe de uma forma organizada.

Listando as branches do repositório



Para visualizar todas as branches que fazem parte do repositório, execute o comando:

`git branch`

Criando uma branch



Para criar uma branch, executamos o comando:

```
git branch nova_branch
```

Com esse comando será criada uma nova ramificação do projeto a partir do estado da branch atual, com uma variação do comando podemos indicar uma branch de referência para a sua criação:

```
git branch nova_branch branch_referencia
```

Convenções para nomear branches



No dia a dia seguimos um padrão para a nomenclatura de branches, o que nos facilita a identificação de seu conteúdo

Convenções para nomear branches



hotfix/nome_do_bug: usado para correções rápidas, exemplo: o botão de adicionar ao carrinho não funciona, a correção desse bug será feita numa branch hotfix.

bugfix/nome_do_bug: correção de um bug que não é um impedimento ao usuário, não compromete um fluxo do site.

feature/nome_da_funcionalidade: a adição de uma funcionalidade será contida nessa nova branch, até que ela esteja desenvolvida e validada, exemplo: feature/add_to_favorites.

Navegando entre branches



Para trocar de branch usamos o comando:

`git checkout nome_branch`

Podemos combinar a criação com a troca de branch com o comando:

`git checkout -b nome_branch`

A branch nova será criada e nosso repositório estará baseado nela.

Deletando uma branch



Para deletar uma branch, executamos o comando:

```
git branch -D nome_branch
```

Módulo 2 / Aula 6

Merge

Gian Souza



Conteúdo a aula

- Mesclando branches
- Conflitos entre branches



Mesclando branches



Para mesclar duas branches, trazendo as atualizações feitas em uma para a outra, executamos o comando:

```
git merge nome_branch
```

Conflitos entre branches



Apesar do Git ser uma ótima ferramenta para o trabalho em equipe, ele não pode impedir que duas pessoas alterem o mesmo arquivo.

Imagine que o arquivo teste.html foi alterado na branch X e na branch Y, no momento em que o merge for realizado o Git criará um conflito, pois ele não sabe qual versão considerar, esse conflito deverá ser resolvido de forma manual.

Módulo 2 / Aula 7

Github

Gian Souza



Conteúdo a aula



- **Sobre o Github**
- **Criando um repositório – projeto já existente**
- **Criando um repositório – novo projeto**
- **Enviando atualizações**
- **Recebendo atualizações**

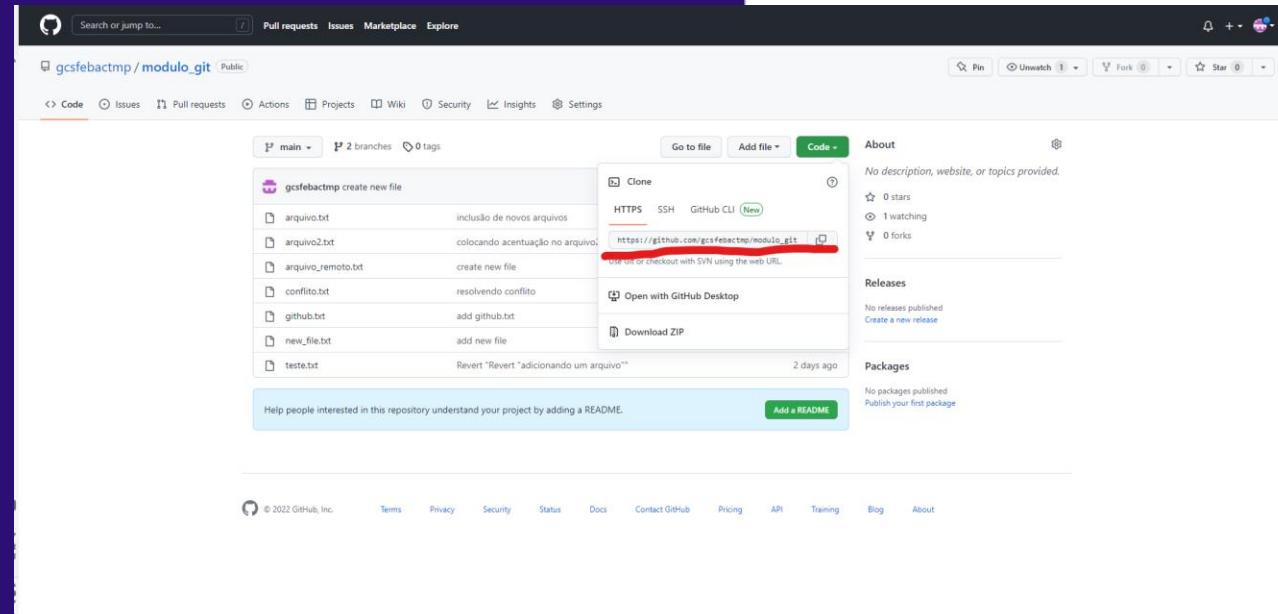
Github



O Github é um site onde podemos armazenar nossos repositórios Git, além do armazenamento ele proporciona algumas funcionalidades de comunidade, como favoritar seus repositórios, seguir usuários e reportar bugs.

Clonando um projeto

Para baixarmos um projeto do Github, o termo correto é clonar, estando na página do repositório, clique em code e escolha a opção HTTPS, copiando o endereço:



Clonando um projeto



Com o endereço do repositório copiado,
execute o comando:

git clone LINK –l pasta_destino*

* O parâmetro –l é opcional, sem ele será criada uma pasta com o nome do repositório

Enviando atualizações



Após fazer o commit de nossas alterações, precisamos envia-las ao repositório, que está armazenado no Github, fazemos isso através do comando:

git push

Recebendo atualizações



Para manter nosso projeto atualizado com o conteúdo remoto, o conteúdo presente no Github, executamos o comando:

`git pull`

Enviando uma determinada branch

Para enviarmos o conteúdo de outra branch, que não é a principal, primeiro mudamos para ela, usando o git checkout e depois executamos o git push com um parâmetro:

`git push origin nome_da_branch`