CRIAR RAMIFICAÇÕES

Criar ramificações permite que testes e mudanças no projeto sejam feitos sem interferir no projeto original. Também é possível realizar a fusão das ramificações. A imagem a seguir ilustra uma ideia geral desse conceito, onde cada círculo corresponde a um *commit* realizado e, a partir dos conceitos de rastreio de mudanças vistos no PDF “2_Repositorio_Local”, é possível criar as ramificações.

O ramo original do Projeto é chamado de **ramo *master***. Para criar uma nova ramificação, utiliza-se o comando

```
git checkout -b <nome_ramo>
```

e para verificar a posição atual na ramificação, utiliza-se `git log --oneline` ou então o comando

```
git branch
```

Todos os *commits* realizados a partir da criação de novos ramos serão, então, atribuidos ao ramo atual. Além disso, todos os ramos criados irão herdar os *commits* do ramo *master* e, no momento da criação de um novo ramo, o *HEAD* apontará tanto ao novo ramo, quanto ao ramo *master*, pois não há ainda um novo *commit* para o novo ramo criado.

Após realizar os *commits* no ramo criado, para retornar ao ramo *master* utiliza-se o comando

```
git checkout master
```

De forma análoga, para voltar para o novo ramo criado, utiliza-se

```
git checkout <nome_ramo>
```

Ao alternar entre os ramos, é possível verificar que os arquivos e pastas do diretório do projeto também são alterados de acordo com a versão do *master* ou dos ramos. Isso significa que se um mesmo arquivo for alterado de formas diferentes em cada ramo, ao realizar a união dos ramos, haverá um conflito.

Para realizar a união de um ramo secundário com o ramo *master*, utiliza-se, primeiramente, o comando a seguir, inserindo o nome do ramo que deseja-se unir ao *master*

```
git merge <nome_ramo>
```

Caso não haja conflitos, utiliza-se o comando `git status` para verificar as mudanças e, se for necessário cancelar a união, utiliza-se o comando

```
git merge --abort
```

Para confirmar a união, utiliza-se os comandos `git add` e `git commit`, como de costume.

Caso haja conflitos, o Exemplo 1 mostra como prosseguir.

EXEMPLO 1

1. Criou-se uma nova ramificação chamada “teste” com o comando

```
git checkout -b teste
```

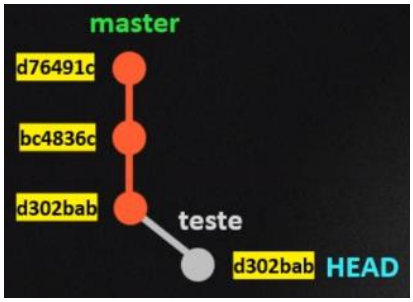
```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git checkout -b teste
Switched to a new branch 'teste'
```

2. Como ainda não realizou-se novos *commits*, *HEAD* apontará tanto para o ramo *master* quanto para o ramo “teste”. É possível verificar que o ramo atual é “teste” e os novos *commits* serão realizados nele

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git log --oneline
d98adda (HEAD -> teste, master) Criado o arquivo d.txt
02e31c5 modificados a.txt e b.txt. Criado c.txt
78e3e5c Criado os arquivos a.txt e b.txt

adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git branch
  master
* teste
```

A figura a seguir ilustra a ideia (é apenas ilustrativa, os *hashes* não são os mesmos do exemplo)



3. Após escrever em “a.txt” e em “b.txt” o texto

“Esta linha foi modificada no ramo Teste”

utilizou-se o comando `git diff` para ver as mudanças e, em seguida, realizou-se o *commit* diretamente com

```
git commit -am “Modificado os arquivos a.txt b.txt”
```

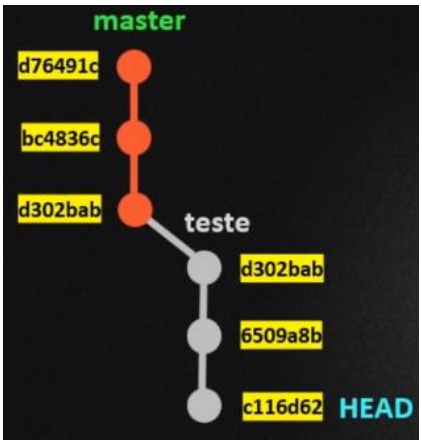
```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git diff
diff --git a/a.txt b/a.txt
index e0b7ce8..b8fdd53 100644
--- a/a.txt
+++ b/a.txt
@@ -1 +1 @@
-Teste
\ No newline at end of file
+Esta linha foi modificada no ramo Teste
\ No newline at end of file
diff --git a/b.txt b/b.txt
index 82624ec..b8fdd53 100644
--- a/b.txt
+++ b/b.txt
@@ -1 +1 @@
-teste2
+Esta linha foi modificada no ramo Teste
\ No newline at end of file

adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git commit -am "Modificado os arquivos a.txt b.txt"
[teste 6691e8c] Modificado os arquivos a.txt b.txt
 2 files changed, 2 insertions(+), 2 deletions(-)
```

4. Ao utilizar o comando `git log` é possível verificar que o *HEAD* e o *commit* pertencem apenas ao ramo “teste”

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git log --oneline
6691e8c (HEAD -> teste) Modificado os arquivos a.txt b.txt
d98adda (master) Criado o arquivo d.txt
02e31c5 modificados a.txt e b.txt. Criado c.txt
78e3e5c Criado os arquivos a.txt e b.txt
```

A figura a seguir ilustra a ideia do que está ocorrendo



5. Para retornar ao ramo *master*, utiliza-se `git checkout master`

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git checkout master
Switched to branch 'master'

adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git branch
* master
  teste
```

6. Ao utilizar o comando `git log` é possível verificar que em *master* a última modificação nunca ocorreu

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git log --oneline
d98adda (HEAD -> master) Criado o arquivo d.txt
02e31c5 modificados a.txt e b.txt. Criado c.txt
78e3e5c Criado os arquivos a.txt e b.txt
```

Mas caso seja feito o `git checkout teste`, é possível ver que a modificação permanecerá no ramo

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git checkout teste
Switched to branch 'teste'

adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (teste)
$ git log --oneline
6691e8c (HEAD -> teste) Modificado os arquivos a.txt b.txt
d98adda Criado o arquivo d.txt
02e31c5 modificados a.txt e b.txt. Criado c.txt
78e3e5c Criado os arquivos a.txt e b.txt
```

7. Retornando mais uma vez ao ramo *master*, escreveu-se em “a.txt” e “b.txt” o seguinte texto

“Esta linha foi modificada no ramo Master”

e realizou-se o *commit* em *master* diretamente com

`git commit -am “Modificado os arquivos a.txt b.txt”`

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git commit -am "Modificado os arquivos a.txt b.txt"
[master 06cd1c1] Modificado os arquivos a.txt b.txt
 2 files changed, 2 insertions(+), 2 deletions(-)
```

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git log --oneline
06cd1c1 (HEAD -> master) Modificado os arquivos a.txt b.txt
d98adda Criado o arquivo d.txt
02e31c5 modificados a.txt e b.txt. Criado c.txt
78e3e5c Criado os arquivos a.txt e b.txt
```

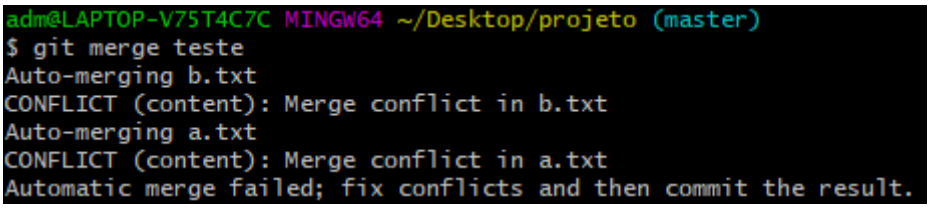
8. Nesse instante, em cada um dos ramos (*master* e “teste”), os arquivos “a.txt” e “b.txt” possuem um texto diferente. É possível ver todas as ramificações utilizando o comando

`git log --oneline --graph --all`

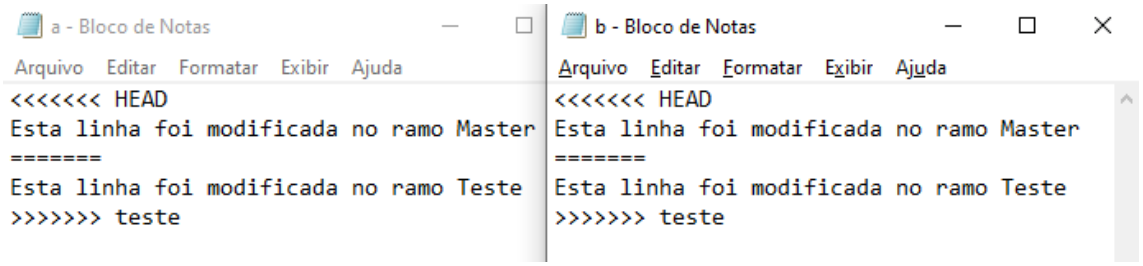
```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git log --oneline --graph --all
* 06cd1c1 (HEAD -> master) Modificado os arquivos a.txt b.txt
| * 6691e8c (teste) Modificado os arquivos a.txt b.txt
|/
* d98adda Criado o arquivo d.txt
* 02e31c5 modificados a.txt e b.txt. Criado c.txt
* 78e3e5c Criado os arquivos a.txt e b.txt
```

9. Ao utilizar o comando para unir os ramos, **ocorre um conflito** por conta dos textos serem diferentes em cada ramo

```
git merge teste
```



O Git também marca dentro dos arquivos a parte em que ocorre o conflito

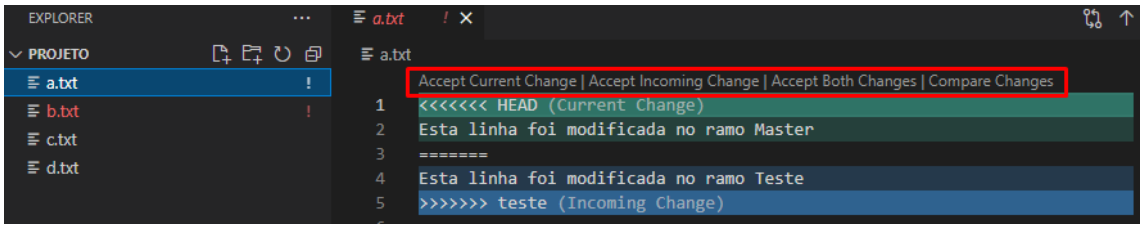


10. Para corrigir, é necessário acessar os arquivos pelo editor configurado ao instalar o Git Bash. Nesse caso, o Visual Studio Code. Abriu-se o Visual Studio Code no diretório do projeto utilizando o comando

```
code .
```

A configuração desse comando foi realizada no PDF “1_Configurações_Iniciais”.

11. As opções de solução do problema aparecem na parte superior da tela, ao abrir o arquivo que causa o conflito



12. Nesse caso, escolheu-se a opção de manter as duas mudanças e salvou-se as alterações feitas com Ctrl+S

13. De volta ao Git Bash, utilizou-se os comandos

```
git status
```

```
git commit -am “Fusão dos ramos e resolução dos conflitos”
```

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master|MERGING)
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   a.txt
        both modified:   b.txt

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

adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master|MERGING)
$ git commit -am "Fusão dos ramos e resolução dos conflitos"
[master 94b0be6] Fusão dos ramos e resolução dos conflitos
```

14. Resultado final

```
adm@LAPTOP-V75T4C7C MINGW64 ~/Desktop/projeto (master)
$ git log --oneline --graph --all
*   94b0be6 (HEAD -> master) Fusão dos ramos e resolução dos conflitos
|\
| * 6691e8c (teste) Modificado os arquivos a.txt b.txt
* | 06cd1c1 Modificado os arquivos a.txt b.txt
|/
* d98adda Criado o arquivo d.txt
* 02e31c5 modificados a.txt e b.txt. Criado c.txt
* 78e3e5c Criado os arquivos a.txt e b.txt
```

A figura a seguir ilustra a ideia do que ocorreu