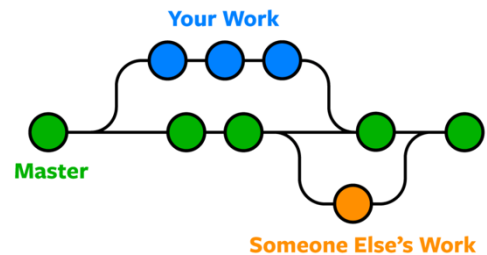


Ficha 3 – Git II: *branches* & resolução de conflitos

Um *branch* no Git é simplesmente um ponteiro móvel e leve para um *commit*. O *branch* por omissão no Git chama-se **master**. À medida que são feitos os *commits*, o *branch master* aponta para o último *commit* efetuado. Sempre que é feito um *commit*, o ponteiro do *branch master* avança automaticamente.



NOTA: Reproduza os passos descritos nesta ficha, tendo o cuidado de **adaptar ao seu caso nomes, emails, URLs, etc.** São mostrados os resultados nas figuras em ambiente MAC OS (esquerda) e Windows (direita), no entanto, os resultados são iguais. Pode haver diferenças a nível dos nomes de ficheiros e pastas pelo que é aconselhado ver o ambiente MAC OS que terá os nomes corretos.

1. Ficheiro *gitignore*

O Git vê todos os ficheiros da árvore de trabalho de uma das seguintes 3 maneiras:

- **tracked** – um ficheiro que foi previamente adicionado à *staging area* ou foi *committed*;
- **untracked** – um ficheiro que não foi adicionado à *staging area* ou *committed*;
- **ignored** – um ficheiro que foi especificamente dito ao Git para o ignorar.

O ficheiro *gitignore* especifica intencionalmente os ficheiros ou tipos de ficheiros não rastreados que o Git deve ignorar.

Imagine que o código que está a desenvolver na diretoria local gera automaticamente vários ficheiros com extensão *.bak* e *.log* e pretende que, ao fazer *commit*, estes sejam ignorados, i.e., que não sejam inseridos no repositório local para posterior envio para o repositório remoto no GitHub.

Posicione-se da diretoria do repositório local IAPSI/ficha1_gitproj. Para criar o ficheiro *gitignore*, executar o comando:

```
$touch .gitignore
```

que comece por um ponto “.”

→ O ficheiro *.gitignore* é um *hidden file*, tal como qualquer ficheiro

```
$touch file1.log
```

→ O comando `$touch <file name>` cria um ficheiro vazio

Para visualizar os ficheiros escondidos (*hidden files*) além dos visíveis, contidos numa diretoria, executar o comando:

```
$ls -la
```

→ Mostra todos os ficheiros e pastas visíveis e escondidos

```
$ls -Rla
```

→ Mostra recursivamente todos os ficheiros e pastas visíveis e

escondidos a partir da diretoria atual

Na Figura 1 é possível a visualização de diversos ficheiros .log e .bak (visíveis) que foram adicionados e o .gitignore que na listagem normal não aparece, mas com o comando `$ls -la` já aparece na listagem de ficheiros.

The image shows two terminal windows side-by-side. The left window is a macOS terminal with the title 'ficha1_gitproj -- bash -- 94x41'. It shows the output of 'ls -la' in a directory named 'ficha1_gitproj'. The listing includes files like '.DS_Store', '.git', 'file1.bak', 'file1.log', 'file1.txt', 'file2.bak', 'file2.log', 'file3.bak', 'file3.log', 'file3.txt', 'file4.txt', and 'file5.txt'. The right window is a Windows terminal with the title 'MINGW64/d/Outros/git/meu_projeto_git'. It shows the output of 'ls -la' in a directory named 'meu_projeto_git'. The listing includes files like './', './.', './git/', 'ficheiro1.bak', 'ficheiro1.log', 'ficheiro1.txt', 'ficheiro2.bak', 'ficheiro2.txt', 'ficheiro3.bak', 'ficheiro3.txt', 'ficheiro4.bak', 'ficheiro4.log', and 'ficheiro4.txt'. Below the first listing, the command 'touch .gitignore' is executed in the macOS terminal. Below the second listing, the command 'touch .gitignore' is also shown.

Figura 1 – Criação do ficheiro .gitignore e visualização dos ficheiros

TODO: Exercícios

1. Crie vários ficheiros ".log" e ".bak" e de outros tipos na sua diretoria IAPSI/ficha1_gitproj.
2. Na raiz do repositório local, crie um ficheiro chamado ".gitignore" (*hidden file*) (ver Figura 1).
3. Editar o ficheiro .gitignore e inserir os ficheiros ou tipos de ficheiros que pretende não colocar no controlo de versões. Adicionar em cada linha um ficheiro ou tipo de ficheiros de extensão "*.log" e "*.bak" (ver Figura 2).
4. Verificar que os ficheiros ou tipo de ficheiros que foram adicionados ao gitignore não serão committed (ver Figura 3).

The image shows a vim editor window with the title 'ficha1_gitproj -- vim .gitignore -- 44x13'. The editor displays the contents of the .gitignore file, which include the following lines: '*.bak', 'file1.log', 'file2.log', 'file4.txt', followed by several tilde '~' characters, and finally ':wq' at the bottom.

Figura 2 – Ficheiros que não serão submetidos em controlo de versões: todos os ficheiros do tipo .bak, e os ficheiros específicos file1.log, file2.log e file4.txt

```
MacBook-Pro-2:figura1_gitproj dianasantos$ ls
file1.bak  file2.bak  file3.bak  file4.txt
file1.log  file2.log  file3.log  file5.txt
file1.txt  file2.txt  file3.txt
MacBook-Pro-2:figura1_gitproj dianasantos$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        file3.log

nothing added to commit but untracked files present (use "git add" to track)
MacBook-Pro-2:figura1_gitproj dianasantos$

MINGW64:/d/Outros/git/meu_projeto_git
rsma1@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        ficheiro1.bak
        ficheiro1.log
        ficheiro2.bak
        ficheiro3.bak
        ficheiro4.bak
        ficheiro4.log

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

Figura 3 – Ficheiros candidatos (modificados/adicionados) a serem adicionados à staging area e que não se encontram no gitignore

2. Criar um *branch*

Por omissão, ao criar um repositório, este tem um único *branch* (chamado **master**).

Imagine o seguinte cenário: pretende-se acrescentar código de uma nova *feature* a um conjunto de ficheiros que é partilhado por vários programadores. Ainda se encontram em fase de testes, portanto não é pretendido que ninguém o utilize como se fosse uma versão final.

A solução do Git através de *branches* resolve este problema: trabalhar numa cópia dos ficheiros e quando a versão final estiver concluída, submetê-la como versão final. Esta cópia corresponde a criar um novo *branch* e posteriormente fazer um *merge* do novo *branch* com o *branch* original (o master, caso tenha vindo do master).

Seguem-se alguns comandos que serão úteis para a resolução dos próximos exercícios:

Comando	Significado
<code>\$git branch <branch></code>	Cria um novo <i>branch</i> com o nome dado
<code>\$git checkout <branch></code>	Muda para o <i>branch</i> referido
<code>\$git merge <branch></code>	<i>Merge</i> do <i>branch</i> atual e do <i>branch</i> referido

TODO: Exercícios

1. No repositório atual, IAPSI/figura1_gitproj, em que o *branch* por omissão é o **master**, criar um novo *branch* "NovaFeature" (ver Figura 4).
2. **Mudar** para o novo *branch* "NovaFeature" (ver Figura 4).
3. **Criar um novo ficheiro vazio "newfile.txt"** e fazer **alterações no ficheiro "file1.txt"** acrescentando a linha "Alterações efetuadas no branch NovaFeature" (ver Figura 5).
4. Fazer o **add** e o **commit** das alterações (ver Figura 6).
5. **Mudar** para o *branch* original **master**. Verifique que as alterações efetuadas não são visíveis no *branch* master (o newfile.txt) (ver Figura 6).
6. Fazer o **merge** das alterações contidas no *branch* NovaFeature para o *branch* master. **O merge deve ser efetuado a partir do branch destino** (master) (ver Figura 7).
7. Verifique as alterações efetuadas no *branch* master (ver Figura 7).

NOTA: Deve tentar fazer estes exercícios de forma autónoma aplicando os conhecimentos da ficha corrente e da Ficha 1. Caso tenha dúvidas, consulte a resolução nas imagens abaixo.

```

MINGW64/d:/Outros/git/meu_projeto_git

msmal@DESKTOP-IPF7R13 MINGW64 /d:/Outros/git/meu_projeto_git (master)
$ ls
ficheiro1.bak  ficheiro1.txt  ficheiro2.txt  ficheiro3.txt  ficheiro4.log
ficheiro1.log  ficheiro2.bak  ficheiro3.bak  ficheiro4.bak  ficheiro4.txt

msmal@DESKTOP-IPF7R13 MINGW64 /d:/Outros/git/meu_projeto_git (master)
$ git branch NovaFeature

msmal@DESKTOP-IPF7R13 MINGW64 /d:/Outros/git/meu_projeto_git (master)
$ git checkout NovaFeature
Switched to branch 'NovaFeature'

msmal@DESKTOP-IPF7R13 MINGW64 /d:/Outros/git/meu_projeto_git (NovaFeature)
$ touch novoficheiro.txt

msmal@DESKTOP-IPF7R13 MINGW64 /d:/Outros/git/meu_projeto_git (NovaFeature)
$

```

Figura 4 – Exercícios 1 e 2: criação de novo branch NovaFeature, mudança para esse branch e criação de um novo ficheiro

A Figura 5 mostra que o repositório remoto (GitHub) foi atualizado com o projeto que existia localmente aquando do término da Ficha 1 – Git:

[illegible]

Figura 5 – Exercício 3: alterações efetuadas no file1.txt do branch NovaFeature

```
MacBook-Pro-2:~$ cd /Users/luiz/OneDrive/Desenvolvimento/Projetos/MeuProjeto
MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git init
MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git add .
MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git status
On branch NovaFeature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        modified:   file1.txt
        new file:   file3.log
        new file:   newfile.txt

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git commit -m "Novas alterações no branch NovaFeature"
[NovaFeature 892433b] Novas alterações no branch NovaFeature
 4 files changed, 7 insertions(+), 3 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 file3.log
 create mode 100644 newfile.txt

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ ls
file1.bak  file1.txt  file2.log  file3.bak  file3.txt  file5.txt
file1.log  file2.bak  file2.txt  file3.log  file4.txt  newfile.txt

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ ls
file1.bak  file1.txt  file2.log  file3.bak  file4.txt
file1.log  file2.bak  file2.txt  file3.log  file5.txt

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git checkout NovaFeature
Switched to branch 'NovaFeature'
Your branch is ahead of 'origin/NovaFeature' by 4 commits.
  (use "git push" to publish your local commits)

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git add .
MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git commit -m "Novos updates no branch NovaFeature"
[NovaFeature 4bdc0b] Novos updates no branch NovaFeature
 3 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 .gitignore
 create mode 100644 novoficheiro.txt

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

MacBook-Pro-2:~/OneDrive/Desenvolvimento/Projetos/MeuProjeto$ git push
Enumerating objects: 12, done.
Compressing objects: 100% (10/12), 1.1 KiB, 0.1 MiB, 0.000 MiB/s, 0.000 MiB/s, 0.000 MiB/s
Writing objects: 100% (12/12), 1.1 KiB, 0.1 MiB, 0.000 MiB/s, 0.000 MiB/s, 0.000 MiB/s
Total 12 (delta 10), reused 0, pack-reused 0
remote: Compressing objects: 100% (10/10), 1.1 KiB, 0.1 MiB, 0.000 MiB/s, 0.000 MiB/s, 0.000 MiB/s
remote: Total 12 (delta 10), reused 0, pack-reused 0
Uncompressing objects: 100% (12/12), 1.1 KiB, 0.1 MiB, 0.000 MiB/s, 0.000 MiB/s, 0.000 MiB/s
remote: Creating branch master for feature: master
To: https://github.com:luizalmeida/MeuProjeto.git
 * [new branch] master -> master
Pushed 12 new commits to this repository.

```

Figura 6 – Exercícios 4 e 5: a vermelho são listados os ficheiros que se encontram no branch NovaFeature; a verde os que estão no branch master

```
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
MacBook-Pro-2:ficha1_gitproj dianasantos$ git merge NovaFeature
Merge made by the 'recursive' strategy.
 .gitignore | 4 ++++
 file1.txt | 5 +++-
 file3.log | 1 +
 newfile.txt | 0
 4 files changed, 7 insertions(+), 3 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 file3.log
 create mode 100644 newfile.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.bak  file1.txt  file2.log  file3.bak  file3.txt  file5.txt
file1.log  file2.bak  file2.txt  file3.log  file4.txt  newfile.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$

MINGW64:/d/Outros/git/meu_projeto_git
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
$ git merge NovaFeature
Merge made by the 'recursive' strategy.
 ficheiro1.txt | 2 +-
 novoficheiro.txt | 0
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 novoficheiro.txt
$
```

Figura 7 – Exercícios 6 e 7: merge com o branch NovaFeature (o master ficou com as alterações que estavam no branch NovaFeature)

3. Resolução de conflitos

A resolução de conflitos é extremamente importante no controlo de versões quando se faz *merge* de *branches*. Os próximos exercícios têm por objetivo provocar conflitos e os alunos aprenderem a resolvê-los. O próximo comando pode ser usado nas situações em que os ficheiros já se encontram no repositório e que estão apenas a sofrer alterações:

```
$git commit -a -m "<msg>"
```

Posicionando-se no *branch master*, editar o file3.txt para ficar com o texto da imagem da esquerda da Figura 8:

```
$vim file3.txt
```

<pre>Este é o exemplo de um ficheiro de texto file3 <h2>Bebidas</h2> Coffee Tea Milk ~</pre>	<pre><h1>Listas de coisas</h1> <h2>Comidas</h2> Bolos Tostas Sandes ~</pre>
---	--

Figura 8 – Texto a inserir no file3.txt (esquerda: branch master; direita: branch NovaFeature)

Fazer *commit* das alterações para o repositório local:

```
$git commit -a -m "Fiz uma alteração no file3.txt"
```

Mudar para o *branch NovaFeature*:

```
$git checkout NovaFeature
```

Posicionando-se no *branch NovaFeature*, editar o file3.txt para ficar com o texto da imagem da direita da Figura 8 e efetuar o *commit* (ver Figura 9):

```
$vim file3.txt
$git commit -a -m "Fiz uma alteração no file3.txt"
```

```
ficha1_gitproj — bash — 114x34
MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file3.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -a -m "Fiz uma alteração no file3.txt"
[master 194bec8] Fiz uma alteração no file3.txt
1 file changed, 7 insertions(+), 6 deletions(-)
MacBook-Pro-2:ficha1_gitproj dianasantos$ git checkout NovaFeature
Switched to branch 'NovaFeature'
MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file3.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -a -m "Fiz uma alteração no file3.txt"
[NovaFeature c166a8e] Fiz uma alteração no file3.txt
1 file changed, 7 insertions(+), 4 deletions(-)
MacBook-Pro-2:ficha1_gitproj dianasantos$
```

Figura 9 – Comandos executados para alterações no file3.txt do branch master e NovaFeature

Mudar novamente para o **branch master** e efectuar o merge com o **branch NovaFeature**:

```
$git checkout master
$git merge NovaFeature
```

Pode verificar que ocorreu um conflito no file3.txt:

```
MacBook-Pro-2:ficha1_gitproj dianasantos$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)
MacBook-Pro-2:ficha1_gitproj dianasantos$ git merge NovaFeature
Auto-merging file3.txt
CONFLICT (content): Merge conflict in file3.txt
Automatic merge failed; fix conflicts and then commit the result.
MacBook-Pro-2:ficha1_gitproj dianasantos$
```

Figura 10 – Conflito no file3.txt após o merge

Através do vim, pode resolver o conflito. A Figura 11 mostra o que automaticamente é feito ao file3.txt:

- A **zona azul** mostra o conteúdo do ficheiro que veio do HEAD (neste caso, como está no master, o HEAD aponta para o master);
- A **zona amarela** mostra o conteúdo do ficheiro que veio do **branch NovaFeature**;
- Os **retângulos vermelhos** são linhas criadas automaticamente para que o utilizador possa saber onde começa o conteúdo de cada **branch** e a divisória. Estas são linhas criadas automaticamente, mas que não são eliminadas automaticamente, pelo que é necessário eliminá-las manualmente após a seleção do conteúdo pretendido (caso contrário, ficarão gravadas no ficheiro):
 - <<<<<< HEAD
 - =====
 - >>>>>> NovaFeature

```
<<<<<< HEAD
Este é o exemplo de um ficheiro de texto file3

<h2>Bebidas</h2>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

=====
<h1>Listas de coisas</h1>

<h2>Comidas</h2>

<ul>
  <li>Bolos</li>
  <li>Tostas</li>
  <li>Sandes</li>
</ul>
>>>>>> NovaFeature
```

Figura 11 – Conteúdo vindo do branch master (azul) e NovaFeature (amarelo)

Também pode usar o comando:

```
$git mergetool file3.txt
```

Na Figura 12, o mergetool não está configurado, no entanto sugere o uso do vimdiff como mergetool.

```
MacBook-Pro-2:ficha1_gitproj dianasantos$ git mergetool file3.txt

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff
Merging:
file3.txt

Normal merge conflict for 'file3.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
4 files to edit
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
  modified:   file3.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  file3.txt.orig

MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -a -m "Resolvi conflito no file3.txt"
[master afc3c0d] Resolvi conflito no file3.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master
Your branch is ahead of 'origin/master' by 5 commits.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  file3.txt.orig

nothing added to commit but untracked files present (use "git add" to track)
MacBook-Pro-2:ficha1_gitproj dianasantos$
```

Figura 12 – Outras formas de resolver conflitos

Desta forma, na Figura 13 é mostrado pela, da esquerda para a direita:

1. O file3.txt do **branch master** (./file3_LOCAL_3248.txt);
2. O file3.txt **original**, na versão anterior antes das alterações (./file3_BASE_3248.txt);
3. O file3.txt do **branch NovaFeature** (./file3_REMOTE_3248.txt);
4. Em baixo, a junção do conteúdo de ambos os *branches* conforme a Figura 11.

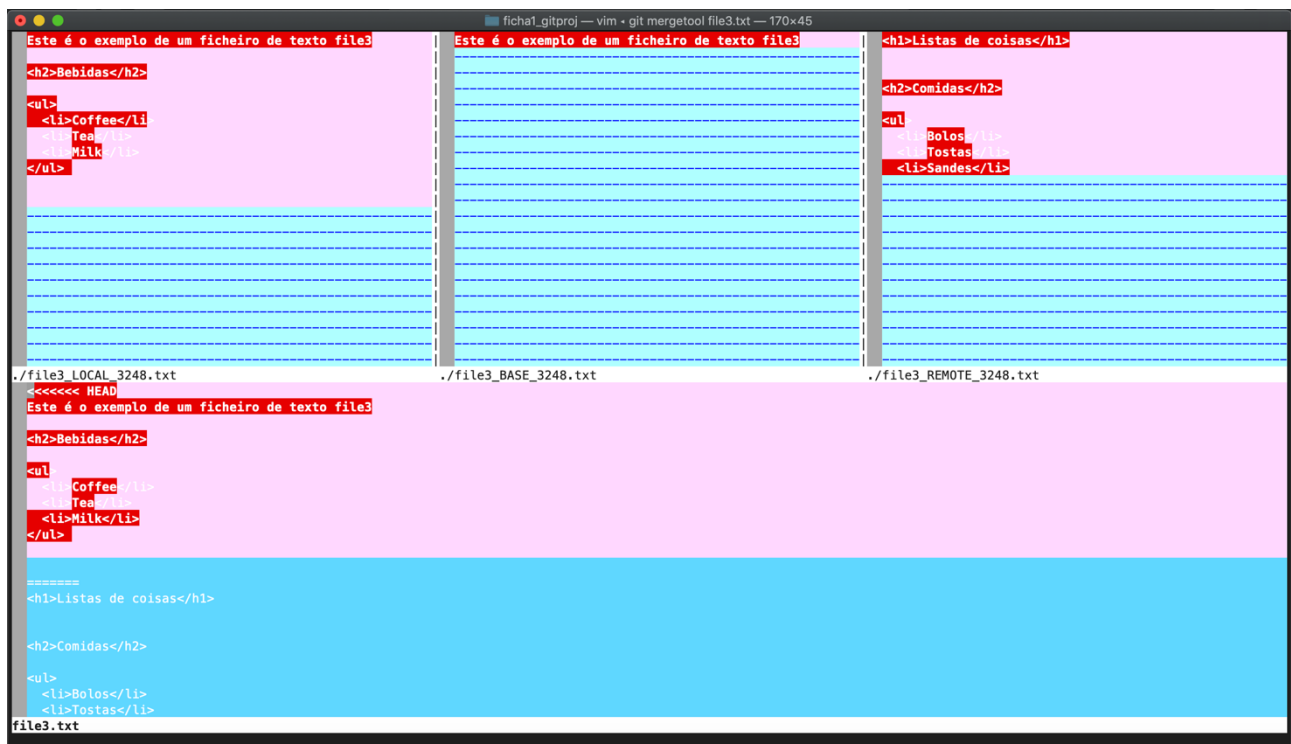


Figura 13 – Git mergetool file3.txt

Ao fazer as alterações pretendidas em baixo, removeram-se as 3 linhas criadas automaticamente e trocou-se a ordem do conteúdo, bem como a linha “Este é o exemplo...” que foi eliminada (ver Figura 14).

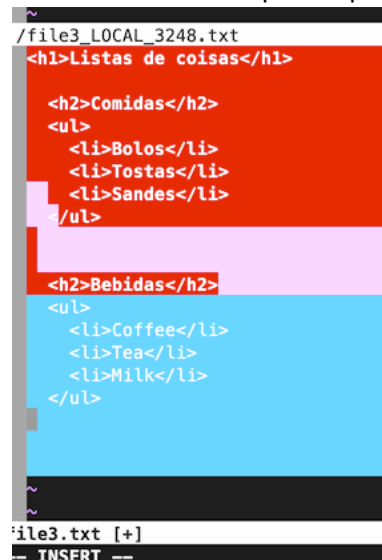


Figura 14 – Aspeto final do file3.txt

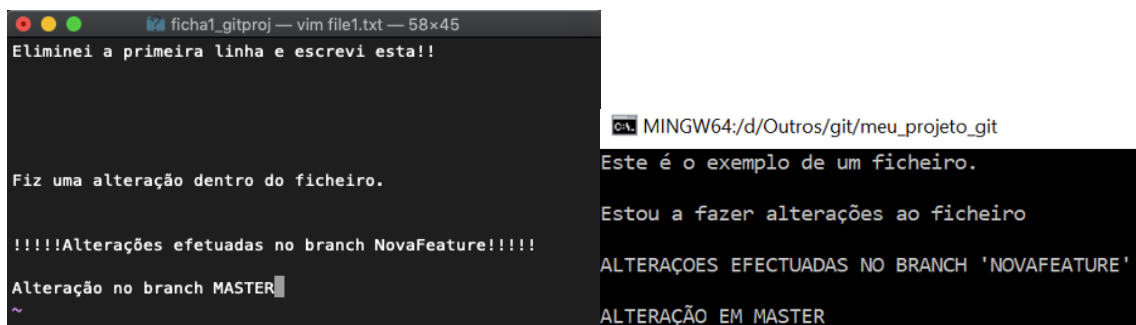
Deve ser feito um novo *commit* no final para aceitar as alterações e eliminar ficheiros que possam ter surgido (ver Figura 12).

Apesar de o vim, servir perfeitamente para a resolução de conflitos, em conflitos mais complexos pode ser usado o `$git mergetool`. E inclusive, utilizar as ferramentas *tortoisemerge* ou *winmerge*.

TODO: Exercícios

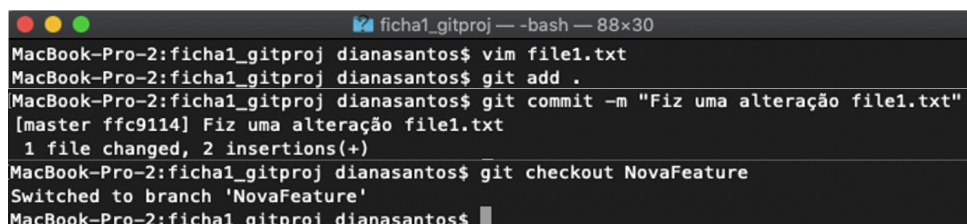
1. No **branch master**, adicione a linha "Alteração no branch MASTER" ao ficheiro "file1.txt" (ver Figura 15).
2. Faça o **add e commit** dessa alteração (ver Figura 16).
3. **Mude para o branch NovaFeature** (ver Figura 16).
4. **Abra para edição o ficheiro file1.txt** e adicione a linha "Alteração no branch NOVAFEATURE" (ver Figura 17).
5. Faça o **add e commit** dessa alteração (ver Figura 18).
6. Efectue o **merge**, desta vez **do branch master para o branch NovaFeature** (i.e., fique no *branch* NovaFeature). Verifique pela Figura 19 que existe um conflito no ficheiro file1.txt.
7. **Edite o ficheiro file1.txt**. Através da Figura 20 pode perceber as razões do conflito.
8. Imaginando que se pretende manter ambas as linhas de cada *branch*, o conflito é resolvido diretamente no ficheiro editado apagando aquilo que não interessa de forma a ficar como na Figura 21. É necessário apagar obrigatoriamente as linhas criadas automaticamente para indicar o nome dos *branches* e a separação, conforme dito anteriormente.
9. Faça o **add e commit** do ficheiro no *branch* atual NovaFeature

NOTA: Deve tentar fazer estes exercícios de forma autónoma aplicando os conhecimentos da ficha corrente e da Ficha 1. Caso tenha dúvidas, consulte a resolução nas imagens abaixo.



The image shows two terminal windows. The left window is a vim editor titled 'ficha1_gitproj — vim file1.txt — 58x45'. It contains the following text: 'Eliminei a primeira linha e escrevi esta!!', 'Fiz uma alteração dentro do ficheiro.', '!!!!Alterações efetuadas no branch NovaFeature!!!!', and 'Alteração no branch MASTER'. The right window is a command prompt titled 'MINGW64:/d/Outros/git/meu_projeto_git'. It contains the following text: 'Este é o exemplo de um ficheiro.', 'Estou a fazer alterações ao ficheiro', 'ALTERAÇÕES EFECTUADAS NO BRANCH 'NOVAFEATURE'', and 'ALTERAÇÃO EM MASTER'.

Figura 15 – Exercício 1: alteração no file1.txt



The image shows a terminal window titled 'ficha1_gitproj — -bash — 88x30'. It contains the following commands and output: 'MacBook-Pro-2:ficha1_gitproj dianasantos\$ vim file1.txt', 'MacBook-Pro-2:ficha1_gitproj dianasantos\$ git add .', 'MacBook-Pro-2:ficha1_gitproj dianasantos\$ git commit -m "Fiz uma alteração file1.txt"', '[master ffc9114] Fiz uma alteração file1.txt', '1 file changed, 2 insertions(+)', 'MacBook-Pro-2:ficha1_gitproj dianasantos\$ git checkout NovaFeature', 'Switched to branch 'NovaFeature'', and 'MacBook-Pro-2:ficha1_gitproj dianasantos\$'.

Figura 16 – Exercício 1, 2 e 3

The left terminal window shows a user editing a file in a vim editor. The text includes: "Eliminei a primeira linha e escrevi esta!!", "Fiz uma alteração dentro do ficheiro.", "!!!!Alterações efetuadas no branch NovaFeature!!!!", and "Alteração no branch NOVAFEATURE". The right terminal window shows a user in a MINGW64 environment editing a file. The text includes: "Este é o exemplo de um ficheiro.", "Estou a fazer alterações ao ficheiro", "ALTERAÇÕES EFECTUADAS NO BRANCH 'NOVAFEATURE'", and "ALTERAÇÕES EM NOVAFEATURE".

Figura 17 – Exercício 4

The terminal window shows a user in a MacBook-Pro-2 environment. The commands and output are: "MacBook-Pro-2:ficha1_gitproj dianasantos\$ vim file1.txt", "MacBook-Pro-2:ficha1_gitproj dianasantos\$ git add .", "MacBook-Pro-2:ficha1_gitproj dianasantos\$ git commit -m 'Fiz uma alteração em file1.txt'", "[NovaFeature fac5daa] Fiz uma alteração em file1.txt", "1 file changed, 2 insertions(+)", and "MacBook-Pro-2:ficha1_gitproj dianasantos\$".

Figura 18 – Exercício 5

The left terminal window shows a user in a MacBook-Pro-2 environment. The commands and output are: "MacBook-Pro-2:ficha1_gitproj dianasantos\$ git merge master", "Auto-merging file1.txt", "CONFLICT (content): Merge conflict in file1.txt", "Automatic merge failed; fix conflicts and then commit the result.", "MacBook-Pro-2:ficha1_gitproj dianasantos\$ git status", "On branch NovaFeature", "You have unmerged paths.", "(fix conflicts and run 'git commit')", "(use 'git merge --abort' to abort the merge)", "Changes to be committed:", "new file: file1.bak", "new file: file1.log", "new file: file2.bak", "new file: file2.log", "new file: file3.bak", "new file: file4.txt", "Unmerged paths:", "(use 'git add <file>...' to mark resolution)", "both modified: file1.txt", and "MacBook-Pro-2:ficha1_gitproj dianasantos\$". The right terminal window shows a user in a MINGW64 environment. The commands and output are: "MINGW64/d/Outros/git/meu_projeto_git", "Este é o exemplo de um ficheiro.", "Estou a fazer alterações ao ficheiro", "ALTERAÇÕES EFECTUADAS NO BRANCH 'NOVAFEATURE'", "ALTERAÇÕES EM NOVAFEATURE", "MINGW64/d/Outros/git/meu_projeto_git", "MINGW64/d/Outros/git/meu_projeto_git (NovaFeature)", "\$ git merge master", "Auto-merging ficheiro1.txt", "CONFLICT (content): Merge conflict in ficheiro1.txt", "Automatic merge failed; fix conflicts and then commit the result.", "MINGW64/d/Outros/git/meu_projeto_git (NovaFeature|MERGING)", "\$ git status", "On branch NovaFeature", "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: ficheiro1.txt", "no changes added to commit (use 'git add' and/or 'git commit -a')", "MINGW64/d/Outros/git/meu_projeto_git (NovaFeature|MERGING)", and "\$".

Figura 19 – Exercício 6: merge e conflitos

The left terminal window shows a user editing a file in a vim editor. The text includes: "Eliminei a primeira linha e escrevi esta!!", "Fiz uma alteração dentro do ficheiro.", "!!!!Alterações efetuadas no branch NovaFeature!!!!", and "Alteração no branch NOVAFEATURE". The right terminal window shows a user in a MINGW64 environment editing a file. The text includes: "Este é o exemplo de um ficheiro.", "Estou a fazer alterações ao ficheiro", "ALTERAÇÕES EFECTUADAS NO BRANCH 'NOVAFEATURE'", "ALTERAÇÕES EM NOVAFEATURE", "ALTERAÇÃO EM MASTER", and "master".

Figura 20 – Exercício 7: edição do ficheiro que contém conflitos. A verde é mostrada a alteração efetuada no branch atual (indicado por HEAD); a vermelho é mostrada a alteração que foi feita e veio do branch master.

```

ficha1_gitproj — vim file1.txt
Eliminei a primeira linha e escrevi esta!!

Fiz uma alteração dentro do ficheiro.

!!!!Alterações efetuadas no branch NovaFeature!!!!

Alteração no branch NOVAFEATURE

Alteração no branch MASTER

MINGW64:/d/Outros/git/meu_projeto_git
Este é o exemplo de um ficheiro.

Estou a fazer alterações ao ficheiro

ALTERAÇÕES EFECTUADAS NO BRANCH 'NOVAFEATURE'

ALTERAÇÕES EM NOVAFEATURE

ALTERAÇÃO EM MASTER

```

Figura 21 – Exercício 8: resolução do conflito, através da aceitação de ambas as linhas (uma vinda do branch master e outra do NovaFeature)

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file1.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git add .
MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -m "Aceitei as alterações vindas do
branch master e do NovaFeature no file1.txt"
[NovaFeature f397a4a] Aceitei as alterações vindas do branch master e do NovaFeature no
file1.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$

```

Figura 22 – Exercício 9: add e commit

4. Adicionar um *branch* criado no repositório local para o repositório remoto

Na Ficha 2 – GitHub, o repositório local *ficha1_gitproj* foi transferido para o repositório remoto no GitHub. O *upload* de alterações efetuadas do repositório local para o remoto é feito através do comando:

```
$git push
```

Aquando da criação do repositório remoto, foi executado o comando `$git push -u origin master`. Quer dizer que no *origin* existe um *branch* master, pelo que é possível atualizar novamente o *branch* master com as alterações que entretanto efetuámos localmente no *branch* master.

Certifique-se que neste momento se encontra no *branch* master, executando os seguintes comandos:

```
$git status
```

```
$git checkout master → Executar caso esteja no branch NovaFeature
```

Para visualizar todos os *branches* existentes executar:

```
$git branch -a → Inclui todos os branches incluindo os que existem no repositório remoto.
```

Na Figura 23, verifica-se que existem 2 *branches* locais: NovaFeature e master (onde está posicionado através do *). Também existe um *branch* remoto master (assinalado a vermelho). Neste momento as últimas alterações que foram feitas localmente nesta ficha, não se encontram atualizadas no repositório remoto.

Para fazer o *push* das alterações efetuadas no *branch* master local para o repositório remoto, executar o comando (ver Figura 23):

```
$git remote → Ver os repositórios associados (imagem da direita da Figura 23)
```

```
$git push → Como só existe o origin, pode usar-se este comando
```

```
$git push origin master → Também se pode especificar o repositório e o branch
```

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git branch -a
NovaFeature
* master
remotes/origin/master
MacBook-Pro-2:ficha1_gitproj dianasantos$ git push
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (13/13), 1.26 KiB | 1.26 MiB/s, done.
Total 13 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/dianassantos/ficha2_myproj.git
c8254ce..ffc9114 master -> master

MINGW64/d/Outros/git/meu_projeto_git
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git remote
origin
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git push origin master
Counting objects: 16, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (13/13), done.
Writing objects: 100% (16/16), 1.70 KiB | 868.00 KiB/s, done.
Total 16 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/rsmal08/meu_projeto_git.git
6c22a63..ae601c4 master -> master
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$

```

Figura 23 – Visualização dos branches existentes localmente e remotamente. Push do master para o repositório remoto

Para atualizar também o repositório remoto com as alterações efetuadas no *branch* NovaFeature, executar o comando (ver Figura 24):

\$git push origin NovaFeature

\$git branch -a

→ Verificar que remotamente já existem ambos os *branches*

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git push origin NovaFeature
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 648 bytes | 648.00 KiB/s, done.
Total 6 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
remote:
remote: Create a pull request for 'NovaFeature' on GitHub by visiting:
remote:   https://github.com/dianassantos/ficha2_myproj/pull/new/NovaFeature
remote:
To https://github.com/dianassantos/ficha2_myproj.git
* [new branch]      NovaFeature -> NovaFeature
MacBook-Pro-2:ficha1_gitproj dianasantos$ git branch -a
NovaFeature
* master
remotes/origin/NovaFeature
remotes/origin/master
MacBook-Pro-2:ficha1_gitproj dianasantos$

```

Figura 24 – Push do master para o repositório remoto. Visualização dos branches existentes localmente e remotamente.

Ir até ao repositório no GitHub e verificar que foram refletidas as alterações como na Figura 25:

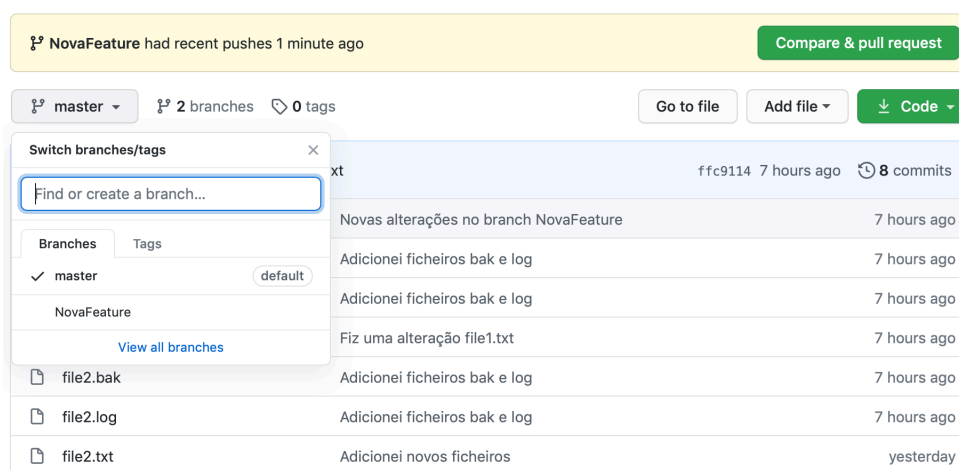


Figura 25 – Verificação dos 2 branches no repositório ficha2_myproj no GitHub