

Ficha 1 – GIT I: repositório local

O Git é um sistema gratuito, *open source*, rápido e distribuído para controlo de versões de software. Este sistema foi inicialmente desenvolvido por Linus Torvalds que assim disponha de um mecanismo automático para disponibilizar o código referente ao desenvolvimento do Kernel Linux.



Além do Kernel Linux, existem atualmente muitos projetos mundiais que usam este sistema de controlo de versões, dos quais se destacam o projecto Perl, Eclipse, Gnome, KDE, QT, Ruby on Rails, Android, PostgreSQL, DEbian, X.org, entre outros.

O Git pode ser instalado em **Mac OS X, Linux/Unix, Windows** e **Solaris** do site oficial (<https://git-scm.com/>).

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. Instalação do GIT

Aceder a <https://git-scm.com/downloads>, fazer o download do git para o seu sistema operativo (e.g., Windows) e efetuar a instalação. Os computadores do laboratório já têm o Git instalado.

O núcleo do Git é um conjunto de programas utilitários de linha de comandos que foram desenhados para o executar num ambiente de linha de comandos do estilo Unix. Sugere-se a aplicação do **Git Bash** para Windows, ou do **Git CMD**. A instalação disponibiliza ainda uma *Graphical User Interface* (GUI) para aceder ao Git, *Git GUI*, em modo gráfico. Neste tutorial, irá ser usada a opção Git Bash. As imagens mostram sempre os resultados em Mac OS X (primeiro) e em Windows (segundo).

Após executar o Git Bash, na imagem da direita (Windows) da Figura 1, o diretório atual é `c:/users/rsmal`, ou seja, a pasta raíz do utilizador `rsmal`. A pasta raiz/`home` é representada pelo carácter: `~` (incluindo na imagem da esquerda com o utilizador `dianasantos`).

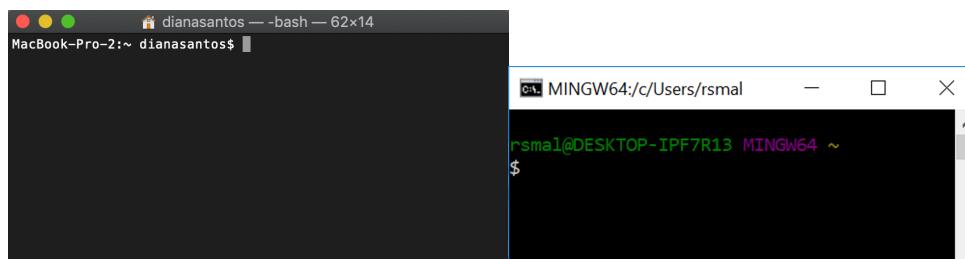


Figura 1 – Terminais abertos (esquerda: Mac OS; direita: Windows)

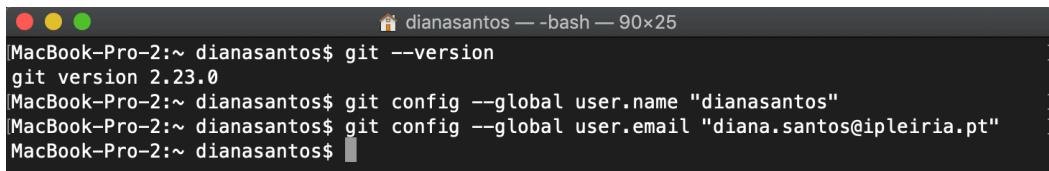
2. Configuração do Git

Ver a versão atual do Git (ver imagem de baixo da Figura 2):

```
$git --version
```

Configurar o Git com o nome e email do aluno (ver Figura 2):

```
$git config --global user.name "[o_meu_nome]"  
$git config --global user.email "[o_meu_email]"
```



The screenshot shows a Mac OS X terminal window titled 'dianasantos — bash — 90x25'. The command history is as follows:

```
[MacBook-Pro-2:~ dianasantos$ git --version  
git version 2.23.0  
[MacBook-Pro-2:~ dianasantos$ git config --global user.name "dianasantos"  
[MacBook-Pro-2:~ dianasantos$ git config --global user.email "diana.santos@ipleiria.pt"  
MacBook-Pro-2:~ dianasantos$ ]
```

Figura 2 – Configuração do Git

3. Criar um repositório local para o projeto

Para criar um repositório Git para um projeto, **escolher/criar uma diretoria adequada** onde achar melhor no seu PC pessoal (ex., "*d:/IAPSI/ficha1_gitproj*"). Nos computadores do laboratório convém criar no próprio Desktop do computador.

Sugere-se criar a seguinte estrutura: **Desktop/IAPSI/ficha1_gitproj**. Existem duas opções para a criação do endereço pretendido: pode criar através da GUI do explorador do Windows ou com comandos na própria linha de comandos que se segue na próxima tabela.

1. Criando através da GUI, copie o caminho absoluto criado e basta escrever no terminal:
`$cd "Desktop/IAPSI/ficha1_gitproj"`
2. Para a criação de pastas e navegação pelo terminal, seguem-se os comandos básicos:

Comando	Significado
<code>\$ls</code>	Visualizar o conteúdo da diretoria (pasta) atual
<code>\$cd /c</code>	Mudar para a drive "c:/"
<code>\$cd IAPSI</code>	Entrar num nível na árvore de diretórias (ex., passando de "d://" para "d:/IAPSI")
<code>\$mkdir ficha1_gitproj</code>	Criar uma pasta da diretoria atual chamada "ficha1_gitproj"
<code>\$cd ..</code>	Subir um nível na árvore de diretórias (ex., se em "d:/IAPSI", executar o comando "cd ..", passaria para "d://"
<code>\$pwd</code>	Mostra qual a diretoria atual (caminho)

ATENÇÃO: Garantir que o terminal se encontra dentro da pasta "*ficha1_gitproj*". Para tal, após a criação do sistema de pastas pretendido, copiar o endereço completo do caminho das pastas (barra superior do explorador do Windows) e, no Git Bash, executar o seguinte comando:

```
$cd "c:/Users/Userxxxxxx/Desktop/IAPSI/ficha1_gitproj"
```

NOTA: o caminho indicado a vermelho será o copiado pelo aluno.

A Figura 3 mostra, nos dois ambientes, a criação do repositório pretendido "*IAPSI/ficha1_gitproj*" (em qualquer diretoria):

The image shows two terminal windows side-by-side. The left terminal window is on a Mac OS X system (MacBook-Pro-2) and shows the creation of a directory named 'IAPSI' in the user's home directory. The right terminal window is on a Windows system (MINGW64) and shows the creation of a directory named 'meu_projeto_git' within the 'Outros' folder.

```

MacBook-Pro-2:~ dianasantos$ ls
Applications      Movies          bin
Calibre Library   Music           pref-silh
Desktop          Pictures         skype-export
Documents         Public          temp
Downloads        Sites           tmp
Dropbox           Virtual Machines
Library           backup-silh
MacBook-Pro-2:~ dianasantos$ cd Desktop/
MacBook-Pro-2:Desktop dianasantos$ mkdir IAPSI

[MacBook-Pro-2:Desktop dianasantos$ cd IAPSI/
[MacBook-Pro-2:IAPSI dianasantos$ mkdir ficha1_gitproj
[MacBook-Pro-2:IAPSI dianasantos$ pwd
/Users/dianasantos/Desktop/IAPSI
MacBook-Pro-2:IAPSI dianasantos$ ]
```

```

MINGW64:/d/Outros/git/meu_projeto_git

rsmal@DESKTOP-IPF7R13 MINGW64 ~
$ cd /d

rsmal@DESKTOP-IPF7R13 MINGW64 /d
$ ls
$RECYCLE.BIN/  Dropbox-windows/  Found.000/  Multimedia/  Outros/  Software/
Dropbox/        Drop-win/       KP_RM.kdbx  MV/          'PD_Docs e Livros'/  'System Volume Information'/

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros
$ cd Outros/
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros
$ ls
Outras-Variado/  Outros-Ler/

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros
$ mkdir git

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros
$ ls
git/  Outras-Variado/  Outros-Ler/

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros
$ cd git
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git
$ ls
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git
$ mkdir meu_projeto_git
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git
$ ls
meu_projeto_git/
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git
$ cd meu_projeto_git/
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git

```

Figura 3 – Criação da diretoria para os exercícios da corrente ficha

Para indicar que a pasta *ficha1_gitproj* é de facto um repositório Git, sujeito a versionamento, é necessário **estar dentro da diretoria *ficha1_gitproj*** e executar o comando (ver Figura 4):

\$git init

The image shows a terminal window on a Mac OS X system (MacBook-Pro-2) where the command '\$git init' is being executed within the 'ficha1_gitproj' directory. The output shows that an empty Git repository has been initialized in the specified directory.

```

ficha1_gitproj — bash — 91x25

MacBook-Pro-2:IAPSI dianasantos$ ls
ficha1_gitproj
MacBook-Pro-2:IAPSI dianasantos$ cd ficha1_gitproj/
MacBook-Pro-2:ficha1_gitproj dianasantos$ git init
Initialized empty Git repository in /Users/dianasantos/Desktop/IAPSI/ficha1_gitproj/.git/
MacBook-Pro-2:ficha1_gitproj dianasantos$ ]
```

```

MINGW64:/d/Outros/git/meu_projeto_git
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git
$ git init
Initialized empty Git repository in D:/Outros/git/meu_projeto_git/.git/

```

Figura 4 – Inicializar a pasta como uma repositório Git

Verificar que a mensagem de confirmação “Initialized empty Git repository in ...” foi mostrada e que o caminho está correto.

4. Adicionar ficheiros ao repositório

Pretende-se agora criar 3 ficheiros de texto dentro no nosso repositório (ficha1_gitproj):

- **file1.txt** - com o texto "Este é o exemplo de um ficheiro de texto file1";
- **file2.txt** - com o texto "Este é o exemplo de um ficheiro de texto file2";
- **file3.txt** - com o texto "Este é o exemplo de um ficheiro de texto file3".

Estes ficheiros podem ser criados da forma tradicional, através de ambientes gráficos no explorer (Windows)/finder (Mac OS) ou em editores de texto (ex., wordpad). No entanto, é pretendido que haja um entendimento maior sobre a linha de comandos, pelo que se irá usar um editor associado ao git batch, como por ex., o "Vim" (ver Figura 5). Sempre dentro da diretoria do projeto, executar o comando:

\$ vim file1.txt → Escrever o texto “Este é o exemplo de um ficheiro de texto file1”

NOTA: No Vim para editar texto é necessário carregar na tecla “I” (Insert) para começar a escrever. No final, para sair do editor e guardar as alterações, efetuar (Write Quit): **ESC “:wq” ENTER**

```

ficha1_gitproj — bash — 91x25
MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file1.txt

ficha1_gitproj — vim file1.txt — 58x9
Este é o exemplo de um ficheiro de texto file1
~
~
~
~
~
~
-- INSERT --
ficha1_gitproj — vim file1.txt — 57x8
Este é o exemplo de um ficheiro de texto file1
~
~
~
~
~
:wq

MINGW64:/d/Outros/git/meu_projeto_git
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ vim file1.txt

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

```

Figura 5 – Criação de um ficheiro no repositório local do projeto Git

Repetir o mesmo processo para os file2.txt e file3.txt (ver Figura 6).

```
ficha1_gitproj — -bash — 83x19
[MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.txt
[MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file2.txt
[MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file3.txt
[MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.txt      file2.txt      file3.txt
[MacBook-Pro-2:ficha1_gitproj dianasantos$ ]
```

Figura 6 – Resultado da criação dos ficheiros

5. Verificar o estado do repositório

Os 3 ficheiros de texto ficaram criados dentro da diretoria local conforme a Figura 6. Para ver o estado no repositório executar o comando (ver Figura 7):

```
$git status
```

```
ficha1_gitproj — -bash — 83x19
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file1.txt
    file2.txt
    file3.txt

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



```
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto.git (master)
$ ls
ficheiro1.txt  ficheiro2.txt  ficheiro3.txt

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto.git (master)
$ git status
On branch master

No commits yet

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

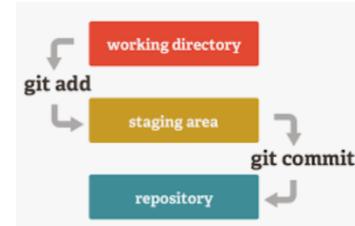
    ficheiro1.txt
    ficheiro2.txt
    ficheiro3.txt

nothing added to commit but untracked files present (use "git add" to track)
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto.git (master)
$ 
```

Figura 7 – Estado do repositório

6. Adicionar os ficheiros à *Staging Area*

A *staging area* permite marcar determinados ficheiros da árvore de trabalho ao executar o comando `$git add`, para que posteriormente possam ser integrados no repositório local (controlo de versões) após o `$git commit`.



Existem 3 ficheiros na pasta atual do projeto. Imaginando que se pretende fazer o controlo de versões apenas do ficheiro `file1.txt`, executar o seguinte comando (ver Figura 8):

```
$git add file1.txt          → Adiciona o ficheiro file1.txt à staging area
$git status                  → Verificar o estado dos ficheiros existentes na pasta atual no git
```

```
ficha1_gitproj — -bash — 70x19
MacBook-Pro-2:ficha1_gitproj dianasantos$ git add file1.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file2.txt
    file3.txt

MacBook-Pro-2:ficha1_gitproj dianasantos$ 
```



```
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto.git (master)
$ git add ficheiro1.txt

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto.git (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  ficheiro1.txt

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

    ficheiro2.txt
    ficheiro3.txt

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto.git (master)
$ 
```

Figura 8 – Staging Area com um ficheiro

Para remover o ficheiro da *staging area* (não apagar, mas sim, desmarcá-lo) executar o comando (ver Figura 9):

```
$git rm --cached file1.txt → Retira o ficheiro file1.txt da staging area
```

Para adicionar mais ficheiros ao mesmo tempo, ou todos os ficheiros que se encontrem na pasta do projeto à *staging area*, executar um dos seguintes comandos (ver Figura 9):

```
$git add . → Adiciona todos os ficheiros ao mesmo tempo à staging area
```

area

```
MacBook-Pro-2:ficha1_gitproj dianasantos$ git rm --cached file1.txt
rm 'file1.txt'
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master

No commits yet

Untracked files:
| (use "git add <file>..." to include in what will be committed)
|   file1.txt
|   file2.txt
|   file3.txt
nothing added to commit but untracked files present (use "git add" to track)
MacBook-Pro-2:ficha1_gitproj dianasantos$ git add .
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master

No commits yet

Changes to be committed:
 (use "git rm --cached <file>..." to unstage)
   new file:  file1.txt
   new file:  file2.txt
   new file:  file3.txt

MacBook-Pro-2:ficha1_gitproj dianasantos$
```

```
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git rm --cached ficheiro1.txt
rm 'ficheiro1.txt'

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git status
On branch master

No commits yet

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

   ficheiro1.txt
   ficheiro2.txt
   ficheiro3.txt

nothing added to commit but untracked files present (use "git add" to track)
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$
```

Figura 9 – Remover ficheiros adicionados à *staging area* e adicionar todos ao mesmo tempo

7. Adicionar os ficheiros da *staging area* (*add*) para o repositório local (*commit*)

Para efetivar o controlo de versões dos ficheiros existentes na *staging area*, executar os seguintes comandos (ver Figura 10):

```
$git rm --cached file2.txt file3.txt → Retira o ficheiro file2.txt e file3.txt da staging area.
```

```
$git status → Verificar que apenas o file1.txt se encontra na staging area.
```

```
$git commit -m "Primeiro versionamento" → Efetuar o commit para o repositório Git e verificar que o file1.txt foi corretamente adicionado ao repositório Git.
```

```
MacBook-Pro-2:ficha1_gitproj dianasantos$ git rm --cached file2.txt file3.txt
rm 'file2.txt'
rm 'file3.txt'
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master

No commits yet

Changes to be committed:
 (use "git rm --cached <file>..." to unstage)
   new file:  file1.txt

Untracked files:
 (use "git add <file>..." to include in what will be committed)
   file2.txt
   file3.txt

MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -m "Primeiro versionamento"
[master (root-commit) 6e6d03d] Primeiro versionamento
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

MacBook-Pro-2:ficha1_gitproj dianasantos$
```

```
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git commit -m "Meu primeiro commit..."
[master (root-commit) 7203aaf] Meu primeiro commit...
 1 file changed, 1 insertion(+)
 create mode 100644 ficheiro1.txt

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git status
On branch master

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

   ficheiro1.txt

nothing added to commit but untracked files present (use "git add" to track)
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$
```

Figura 10 – Primeiro *commit* para o repositório Git

Repetir o mesmo para os ficheiros file2.txt e file3.txt (adicionar à *staging area* e seguidamente ao repositório Git).

8. Fazer atualização de ficheiros no repositório

Os 3 ficheiros neste momento estão no controlo de versões do repositório Git. Para simular uma alteração no ficheiro file1.txt, abrir no editor de texto o correspondente ficheiro e escrever algo “Fiz uma alteração dentro do ficheiro” e guardar as alterações (ver Figura 11).

The screenshot shows two terminal windows side-by-side. The left window, titled 'ficha1_gitproj — vim file1.txt — 64x26', contains the text: 'Eliminei a primeira linha e escrevi esta!' followed by several blank lines and the message 'Fiz uma alteração dentro do ficheiro.' The right window, titled 'MINGW64:/d/Outros/git/meu_projeto_git', shows the same text: 'Este é o exemplo de um ficheiro.' and 'Estou a fazer alterações ao ficheiro.' Both windows have their status bars visible at the bottom.

Figura 11 – Modificação no ficheiro file1.txt

Executar o comando (ver Figura 12):

\$git status → Verificar que o file1.txt se encontra modificado e que essa modificação não está atualmente no controlo de versões do Git.

The screenshot shows two terminal windows. The left window, titled 'ficha1_gitproj — bash — 84x22', shows the command '\$ git status' and its output: 'Changes not staged for commit:' followed by '(use "git add <file>..." to update what will be committed)' and '(use "git restore <file>..." to discard changes in working directory) modified: file1.txt'. The right window, titled 'MINGW64:/d/Outros/git/meu_projeto_git (master)', also shows the command '\$ git status' and its output: 'On branch master', 'Changes not staged for commit:', '(use "git add <file>..." to update what will be committed)' and '(use "git checkout -- <file>..." to discard changes in working directory)', and 'modified: arquivo1.txt'. Both windows have their status bars visible at the bottom.

Figura 12 – Verificar que o ficheiro file1.txt foi modificado e que não está sobre controlo de versões

Para verificar as alterações efetuadas desde o último *commit*, executar o comando (ver Figura 13):

\$git diff → Analisar o conteúdo alterado/adicionado/eliminado no file1.txt em relação ao último *commit*

The screenshot shows two terminal windows. The left window, titled 'ficha1_gitproj — bash — 64x26', shows the command '\$ git diff' and its output: 'diff --git a/file1.txt b/file1.txt' and 'index 9f6a456..fd4713e 100644'. The right window, titled 'MINGW64:/d/Outros/git/meu_projeto_git (master)', also shows the command '\$ git diff' and its output: 'diff --git a/arquivo1.txt b/arquivo1.txt' and 'index fe6e9e7..2ea51fd 100644'. Both windows have their status bars visible at the bottom.

Figura 13 – Verificação das alterações no ficheiro file1.txt modificado

Explicação do output da Figura 13:

- A **vermelho** encontra-se aquilo que foi eliminado
- A **verde** encontra-se tudo aquilo que foi adicionado/alterado (incluindo linhas novas)

NOTA: Se existir mais do que um ficheiro modificado e pretender visualizar as diferenças de apenas um dos ficheiros, utilizar o comando:

```
$git diff file1.txt
```

Para atualizar o ficheiro modificado no repositório, é necessário adicioná-lo novamente à *staging area* e só depois efetuar o *commit* (ver Figura 14):

```
$git status  
$git add -u      → -u adiciona à staging area apenas os ficheiros modificados (e não tudo)  
$git status  
$git commit -m "Fiz alterações no file1.txt"
```

```
ficha1_gitproj dianasantos$ git status  
On branch master  
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:   file1.txt  
  
no changes added to commit (use "git add" and/or "git commit -a")  
MacBook-Pro-2:ficha1_gitproj dianasantos$ git add -u  
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status  
On branch master  
Changes to be committed:  
  (use "git restore --staged <file>..." to unstage)  
    modified:   file1.txt  
  
MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -m "Fiz alterações no file1.txt"  
[master 0a447d3] Fiz alterações no file1.txt  
 1 file changed, 11 insertions(+), 1 deletion(-)  
MacBook-Pro-2:ficha1_gitproj dianasantos$  
  
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)  
$ 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:   file1.txt  
  
no changes added to commit (use "git add" and/or "git commit -a")  
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)  
$ git add -u  
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)  
$ git status  
On branch master  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
    modified:   file1.txt  
  
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)  
$ git commit -m "Acrescentei uma linha no file1.txt"  
[master 893ce28] Acrescentei uma linha no file1.txt  
  1 file changed, 6 insertions(+), 1 deletion(-)  
rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)  
$
```

Figura 14 – Adicionar um ficheiro modificado à *staging area* e efetuar o *commit*

TODO: Exercícios

1. Acrescente algumas linhas ao *file3.txt*.
2. Verifique as alterações entre os 2 ficheiros.
3. Faça o *commit* do ficheiro alterado, sendo que na mensagem a enviar associada ao *commit* deve ter as duas linhas de texto seguintes:

"Este commit tem o objetivo de atualizar o file3.txt"

"Este exercício está englobado no âmbito da ficha 1 de Git"

NOTA: Se pretender inserir mais do que uma linha de texto, são disponibilizadas 3 maneiras:

- `$git commit` → Em seguida será solicitado o texto a inserir na mensagem;
- `$git commit -m "mensagem 1" -m "mensagem 2"` → repetir com o `-m "..."`;
- `$git commit -m 'íncio da msg ENTER 2ª linha da msgENTER 3ª linha da msg.'` (ver imagem da direita da Figura 19).

9. Visualizar o histórico do projeto (*snapshots*)

Para visualizar o histórico do projeto pode ser usado o seguinte comando (ver Figura 15):

```
$git log
```

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git log
commit 6e6d03d82e3e21560f95538d009df79023f0e53e
Author: dianasantos <diana.santos@ipleiria.pt>
Date: Mon Oct 12 12:31:36 2020 +0100

    Primeiro versionamento

diff --git a/file1.txt b/file1.txt
new file mode 100644
index 000000..9f6a456
--- /dev/null
+++ b/file1.txt
@@ -0,0 +1 @@
+Este é o exemplo de um ficheiro de texto file1
MacBook-Pro-2:ficha1_gitproj dianasantos$ 

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git log
commit 893ce288dbcde177af0ce9e85b02fa18a38a7b8c (HEAD -> master)
Author: rsmal <[REDACTED].com>
Date: Wed Feb 21 15:51:28 2018 +0000

    Acrescentei uma linha no ficheiro1

commit b4305a431b5fabdff317b0e2c15d3374d201daae
Author: rsmal <[REDACTED].com>
Date: Wed Feb 21 15:28:15 2018 +0000

    Primeiro commit dos outros 2 ficheiros

commit 7203afc08566258dda2bf010cac7e0712cd8b400
Author: rsmal <[REDACTED].com>
Date: Wed Feb 21 15:22:59 2018 +0000

    Meu primeiro commit...

```

Figura 15 – Visualizar o histórico de commits

Para saber mais informações sobre cada um dos *commits* efetuados, utilizar o comando:

`$git show <hash>` → Onde "hash" é o número hexadecimal associado a cada *commit*. Por exemplo o "hash" do primeiro *commit* da imagem da esquerda da Figura 15 é "7203afc".

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git show 6e6d03
commit 6e6d03d82e3e21560f95538d009df79023f0e53e
Author: dianasantos <diana.santos@ipleiria.pt>
Date: Mon Oct 12 12:31:36 2020 +0100

    Primeiro versionamento

diff --git a/file1.txt b/file1.txt
new file mode 100644
index 000000..9f6a456
--- /dev/null
+++ b/file1.txt
@@ -0,0 +1 @@
+Este é o exemplo de um ficheiro de texto file1
MacBook-Pro-2:ficha1_gitproj dianasantos$ 

rsmal@DESKTOP-IPF7R13 MINGW64 /d/Outros/git/meu_projeto_git (master)
$ git show 7203afc
commit 7203afc08566258dda2bf010cac7e0712cd8b400
Author: rsmal <rsmal08@gmail.com>
Date: Wed Feb 21 15:22:59 2018 +0000

    Meu primeiro commit...

diff --git a/ficheiro1.txt b/ficheiro1.txt
new file mode 100644
index 000000..fe6e9e7
--- /dev/null
+++ b/ficheiro1.txt
@@ -0,0 +1 @@
+Este é o exemplo de um ficheiro de texto...

```

Figura 16 – Detalhes do primeiro commit efetuado

10. Voltar a uma versão anterior

Na Figura 15 verifica-se que foram efetuados 3 *commits* ao projeto. Supondo que ocorreu um erro num determinado ficheiro que entretanto foi modificado, e que se quer voltar à 1^a versão funcional do projeto (no ex., cujo *hash* é "7203afc"), deve executar o seguinte comando (ver Figura 17):

`$git checkout <hash>` → Onde "hash" é o número hexadecimal associado ao 1º commit

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git checkout 6e6d03
Note: switching to '6e6d03'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at 6e6d03d Primeiro versionamento
MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ 

rsmal@DESKTOP-IPF7R13 MINGW64 ~/meu_projeto_git (master)
$ git checkout 7203afc
Note: checking out '7203afc'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 7203afc Meu primeiro commit...

```

Figura 17 – Projeto na 1^a versão

Pode-se verificar que na Figura 17 o repositório voltou a ficar com um único ficheiro, que estava no 1º *commit*. Esta funcionalidade tem como vantagem ir buscar determinados ficheiros com informação anterior que poderia estar mais correta. Coloca o “HEAD” (um género de ponteiro a apontar um determinado *commit*) no 1º *commit*, mas não permite alterações a esta versão. Só com criação de um novo *branch* é que seria possível trabalhar nesta versão.

Para voltar novamente ao último *commit* (3º) efetuado, executar o comando (ver Figura 18):

\$git checkout master → O “ponteiro” HEAD irá novamente para o último *commit* e é possível continuar a fazer alterações nesta versão

```

ficha1_gitproj — bash — 91x26
MacBook-Pro-2:ficha1_gitproj dianasantos$ git checkout master
Previous HEAD position was 6e6d03d Primeiro versionamento
Switched to branch 'master'
MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.txt file2.txt file3.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ 

rsmal@DESKTOP-IPF7R13 MINGW64 ~/meu_projeto_git ((7203afc...))
$ git checkout master
Previous HEAD position was 7203afc Meu primeiro commit...
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
rsmal@DESKTOP-IPF7R13 MINGW64 ~/meu_projeto_git (master)
$ ls
ficheiro1.txt ficheiro2.txt ficheiro3.txt ficheiro4.txt

```

Figura 18 – Projeto na última versão novamente

Adicionar um novo file4.txt, fazer o add à *staging area* e efetuar o commit conforme a Figura 19:

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git commit -m 'esta
|> é a
|> primeira'
|> [master bde9c1a] esta é a primeira
|>   1 file changed, 1 insertion(+)
|>   create mode 100644 file4.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git log
commit bde9c1a1acb6fa2c1a6fbf3bddf7e1f9cdd83da5 (HEAD -> master)
Author: dianasantos <diana.santos@ipleiria.pt>
Date: Mon Oct 12 16:50:12 2020 +0100

esta
é a
primeira

commit 0a447d37cdc6c81ca7a2c69a5c4ebabb892acabb
Author: dianasantos <diana.santos@ipleiria.pt>
Date: Mon Oct 12 13:08:29 2020 +0100

    Fiz alterações no file1.txt

commit 539377bbafc39cf7debaef109a0efe1481eb42ad0
Author: dianasantos <diana.santos@ipleiria.pt>
Date: Mon Oct 12 12:40:19 2020 +0100

        Adicionei novos ficheiros

commit 6e6d03d82e3e21560f95538d009df79023f0e53e
Author: dianasantos <diana.santos@ipleiria.pt>
Date: Mon Oct 12 12:31:36 2020 +0100

        Primeiro versionamento

MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.txt file2.txt file3.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ vim file4.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git add .
MacBook-Pro-2:ficha1_gitproj dianasantos$ ls
file1.txt file2.txt file3.txt file4.txt
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file: file4.txt

```

Figura 19 – Novo ficheiro em controlo de versões

Se for pretendido apenas anular o último *commit* e fazer o(s) ficheiro(s) voltar(em) para a *Staging Area*, executar o comando (ver Figura 20):

\$git reset HEAD~ → O *commit* está a ser anulado, não se está a voltar a uma versão anterior

```

MacBook-Pro-2:ficha1_gitproj dianasantos$ git reset HEAD~
MacBook-Pro-2:ficha1_gitproj dianasantos$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file4.txt

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

```

Figura 20 – Anular o último *commit* efetuado

TODO: Exercícios

1. Verifique a listagem de *commits* efectuados.
2. Anule o último *commit* efectuado.
3. Verifique que as alterações efectuadas nesse *commit* estão agora na *staging area*.
4. Volte a fazer o *commit* dessas alterações presentes na *Staging Area*.