



Desenvolvimento Colaborativo de Software

Git & GitHub

Rui Sousa
Instituto Politécnico do Cávado e do Ave
2023/2024

Rui Sousa | ✉ rsousa@ipca.pt

Introdução

Desenvolvimento Colaborativo de Software

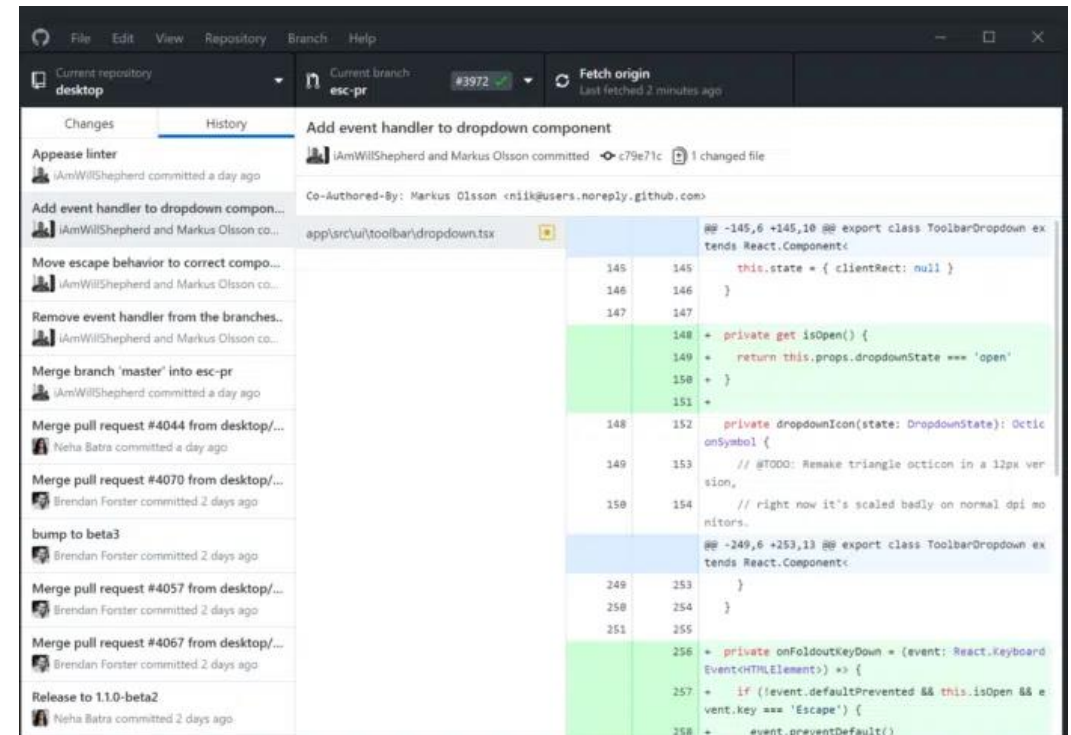
Git & GitHub

O que vamos aprender:

- Git – Conceitos base
- Entender o que é o GitHub
- Ferramentas recomendadas
- Setup do ambiente de desenvolvimento colaborativo
- Comandos básicos de SO (command line)
- Trabalhar com repositórios locais
- Alterar e realizar resets de alterações
- Exercícios práticos
- Trabalhar com repositórios partilhados
- Criação de Branches

Pré-requisitos:

- Visual Studio Code (VS Code)
- Git
- GitHub



Conceitos sobre Git e GitHub

Conceitos sobre Git e GitHub

Enquadramento e desafios para equipas de desenvolvimento

- Controlo de versões de ficheiros / programas

- Cenário:
 - Equipa de 2 ou mais programadores estão a trabalhar no mesmo projeto;
 - A página da turma de DWM 2º Ano;
 - Com funcionalidades de consulta, mas também fórum de discussão sobre as unidades curriculares e outras atividades.

- Pergunta: como trabalhar no mesmo projeto, ao mesmo tempo e resolver os conflitos existentes?

Conceitos sobre Git e GitHub

Solução comum (pequenos projetos)

- Organizar uma estrutura de pastas;
- Dividir o projeto em pequenos módulos:
 - Funções core: 1 programador;
 - Módulo A: outro programador;
 - Módulo B: outro programador;
 - ...
- No final o responsável do projeto junta os códigos todos e resolve os problemas de integração.
- Este processo resulta sobretudo na 1ª versão da solução e em pequenas aplicações, com equipas pequenas e muito organizadas.

Conceitos sobre Git e GitHub

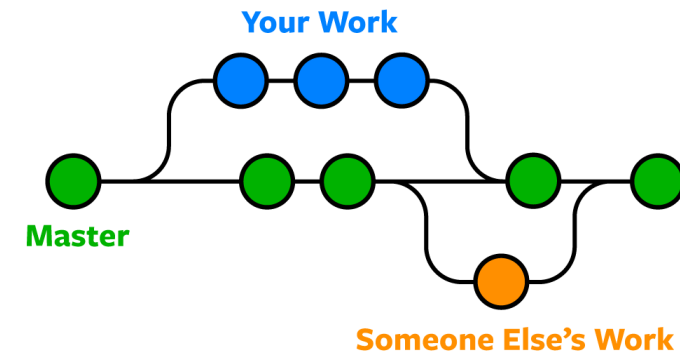
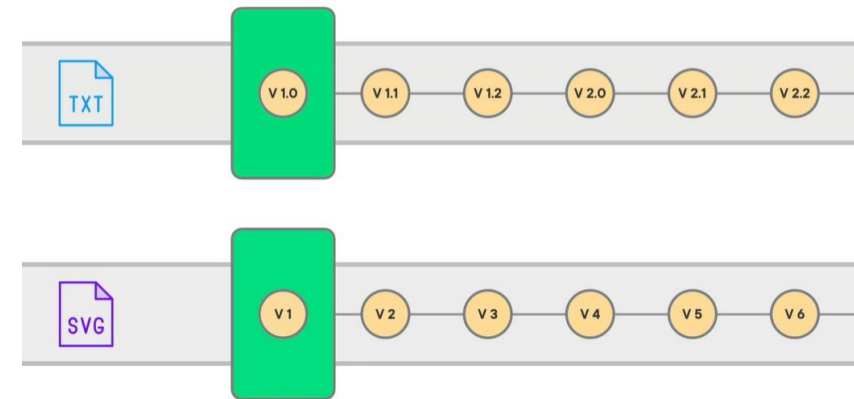
Crescimento da solução

- Problemas:
 - Agora é necessário adicionar mais funcionalidades;
 - Para o mesmo módulo é necessário ter 2 ou mais programadores;
 - É necessário fazer pequenos ajustes nas funções “core” e estas são feitas por programadores dos módulos;
 - É necessário ir buscar versões antigas das funções;
 - Perderam as cópias de segurança da aplicação;
 - ...
- Os sistemas de “versionamento” e trabalho colaborativo ajudam a resolver estes problemas.
- O **Git e o GitHub** são exemplo disso.

Conceitos sobre Git e GitHub

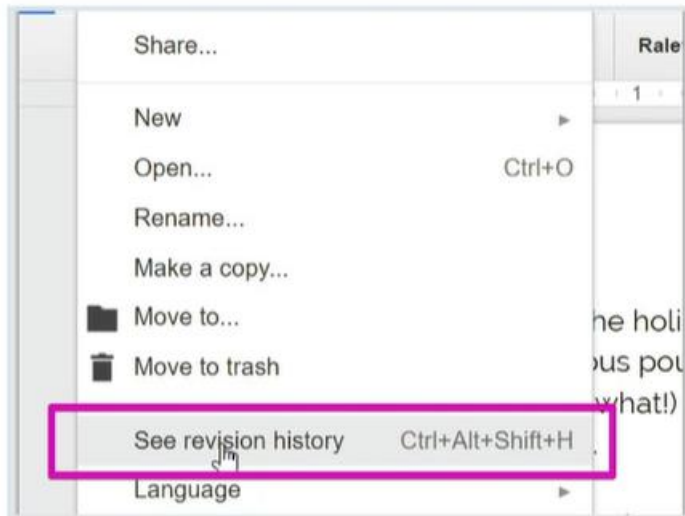
O que é o GIT?

- É um sistema popular de controlo de versões (Decentralized Version Source Control - DVCS);
- É um sistema distribuído e descentralizado de controlo de versões;
- O Git permite manter o histórico de todos os ficheiros (de qualquer tipo de ficheiros), alterações e projetos da nossa conta;
- Permite fazer “deploy” de várias versões do projeto, a partir da mesma conta;
- Limita as alterações entre utilizadores, ou seja, quando um utilizador altera a sua versão, não afeta a versão de outro;
- Permite que o projeto consiga voltar a uma versão estável;
- Gratuito e open-source.

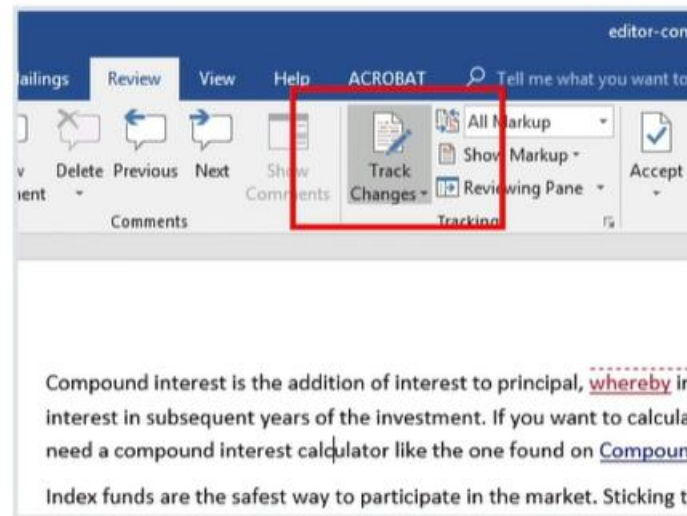


Conceitos sobre Git e GitHub

Tipos de versões de controlo



Google Docs
Controlo de Versões



Office 365
Controlo de Versões



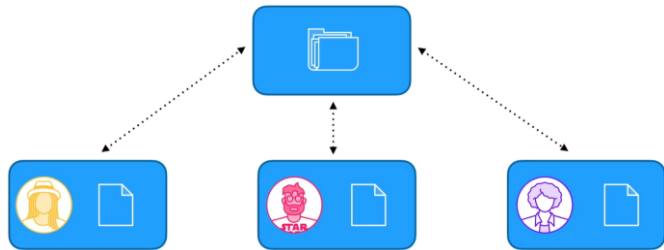
Estrutura de Pastas
Controlo de Versões

Conceitos sobre Git e GitHub

Sistema Centralizado de Controlo de Versões

Centralizado (CVCS)

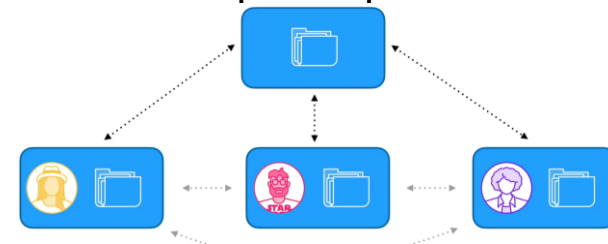
- 1 repositório central;
- Todas as alterações são aplicadas sobre um repositório (repo) central;
- Necessário ‘sincronizar’ com o repositório central para adicionar as atualizações.
- Requer disponibilidade do repositório central.



Ex: CVS, Subversion (SVN)

Descentralizado (DVCS)

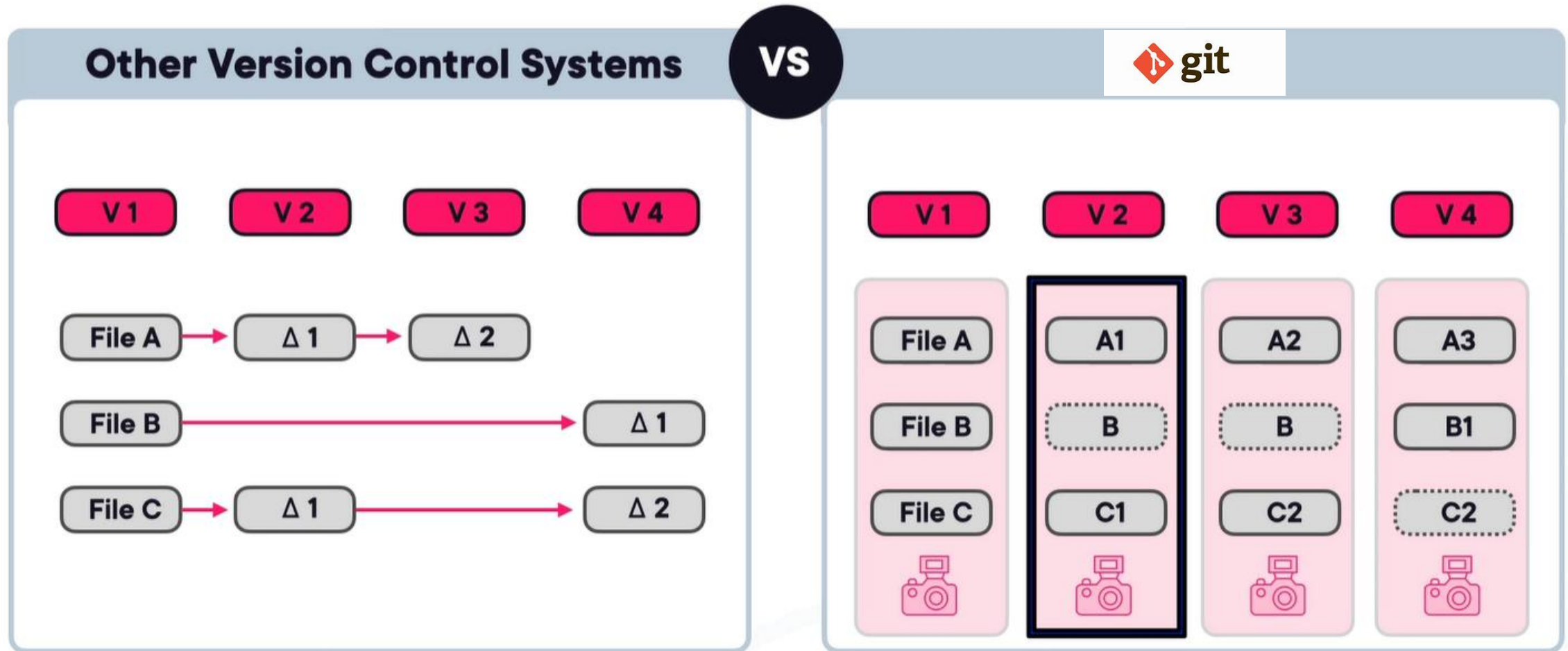
- Múltiplas cópias do repositório;
- Podem ter sempre um repositório (repo) local;
- Podemos opcionalmente ter um repositório central.
 - ‘Push’ para o repositório remoto
 - ‘Pull’ do repositório remote
- Podemos ter múltiplos repositórios remotos



Ex: Git, Mercurial.

Conceitos sobre Git e GitHub

Caraterísticas do Git



Conceitos sobre Git e GitHub

Porque devemos utilizar o GIT?

Rápido

Mais rápido e escalável do que outros sistemas de controlo de versões.
Sistema distribuído que trabalha com repositório local.

Desligado / Desconectado

Podemos trabalhar localmente. Cada elemento da equipa tem uma cópia total localmente no ser posto de trabalho.

Fácil de utilizar

Sistema de controlo de versões de fácil adopção.

Branching ("Ramos")

Sistema baseado em ramificações. O ramo permite trabalhar numa nova funcionalidade ou bugfix sem afectar a versão 'main'.

Pull Requests

Permite a colaboração de Código utilizando uma solicitação pull. Review e Merge de alterações executadas.



Conceitos sobre Git e GitHub

Coisas que devem parar de fazer!

1. Partilhar Código por email
2. Partilhar Código em ficheiros comprimidos .tar, .zip, .tgz
3. Partilhar password
4. Controlo de versões baseados em ficheiros
Index.html.old ou _oldIndex.html
5. Usar OneDrive, Dropbox ou outras plataformas para realizar controlo de versões

Razões para aprender Controlo de Versões

1. Standard estabelecido no mercado
2. Colaboração e partilha de Código com outras pessoas
3. Colaboração em projetos
4. Auditoria de Código
5. Proteção legal?

Conceitos sobre Git e GitHub

O que [não] colocar no sistemas de controlo de versões (Boas Práticas)

Colocar

- Ficheiros baseados em texto
- Fontes de Código
- Scripts

Não colocar

- Ficheiros grandes que são alterados regularmente
 - Images, audio, Video
- Ficheiros que são construídos ('Build') automaticamente: ficheiros executáveis, e ficheiros de objectos
- Ficheiros temporários
- Dados sensíveis: password, private keys
- Estes ficheiros podem ser ignorados utilizando o ficheiro '.gitignore'.

Conceitos sobre Git e GitHub

Diferença entre Git e GitHub

Git
(faz o controlo de
versões dos
arquivos)



GitHub
(sistema remoto de
alojamento de
projetos Git)



Conceitos sobre Git e GitHub

Problema dos projectos

Git
(vamos usar o Git
para controlar as
versões do nosso
projeto)



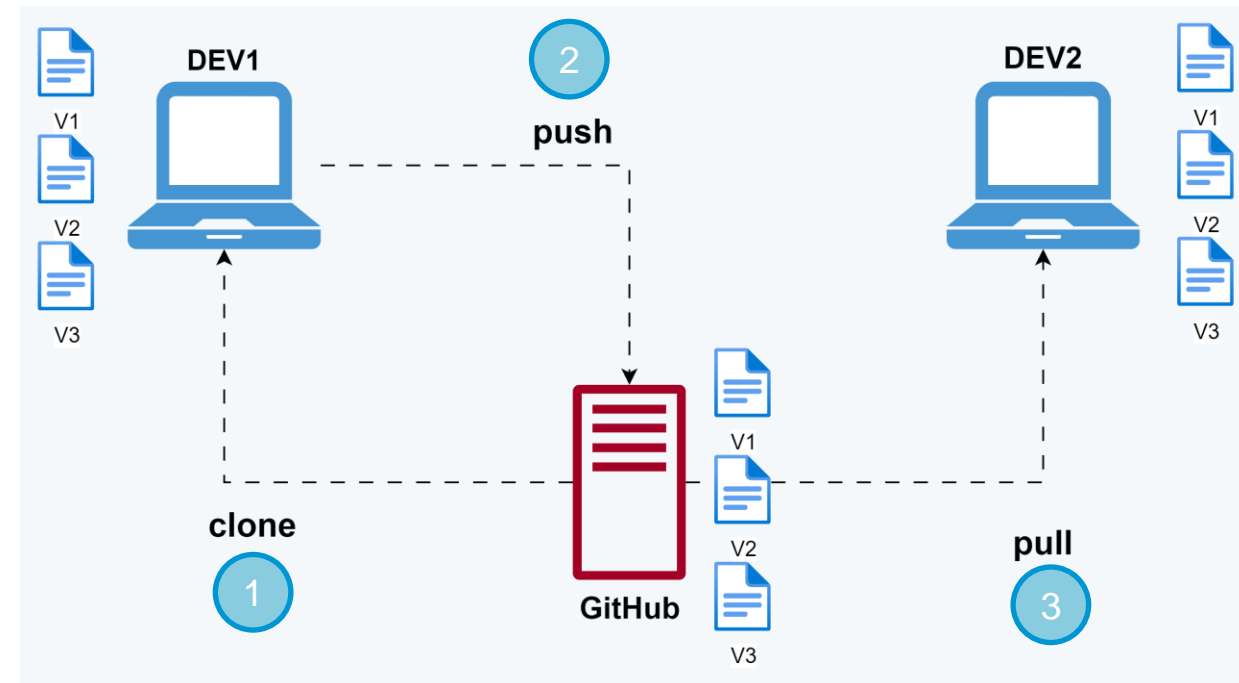
GitHub
(vamos usar o
GitHub para partilhar
e trabalhar em
colaboração)



Conceitos sobre Git e GitHub

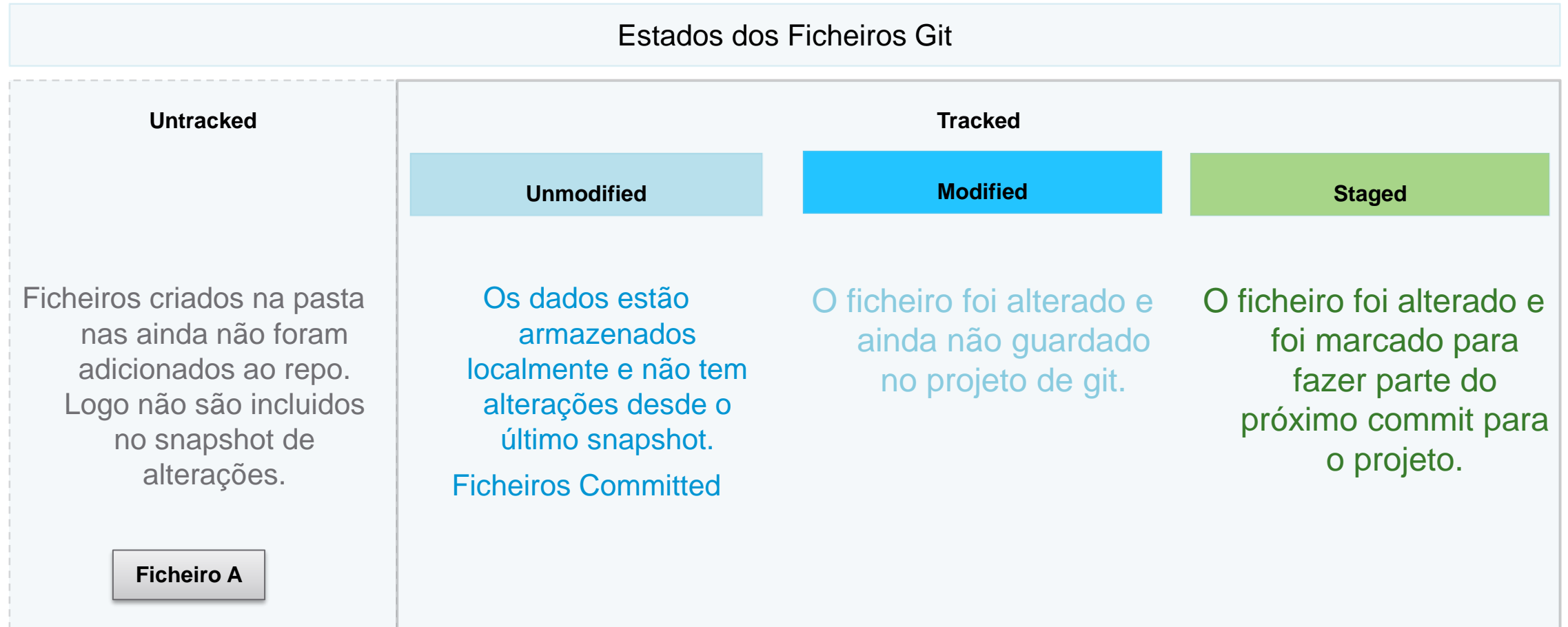
Sistema distribuído de control de versões

- Dev1 irá **clonar** todo o repositório.
- Pode trabalhar offline.
- Pode criar ramificações locais do repositório
- Quando tiver tudo pronto para partilhar com outras pessoas ou equipas, efetua o envio para o repositório central, onde pode ser distribuído novamente.
- O processo de envio de informação de volta ao repositório é designado por 'push'.
- Quando as alterações estiverem submetidas e o novo Código estiver disponível, outros programadores podem puxar (pull) essas alterações para as suas máquinas.



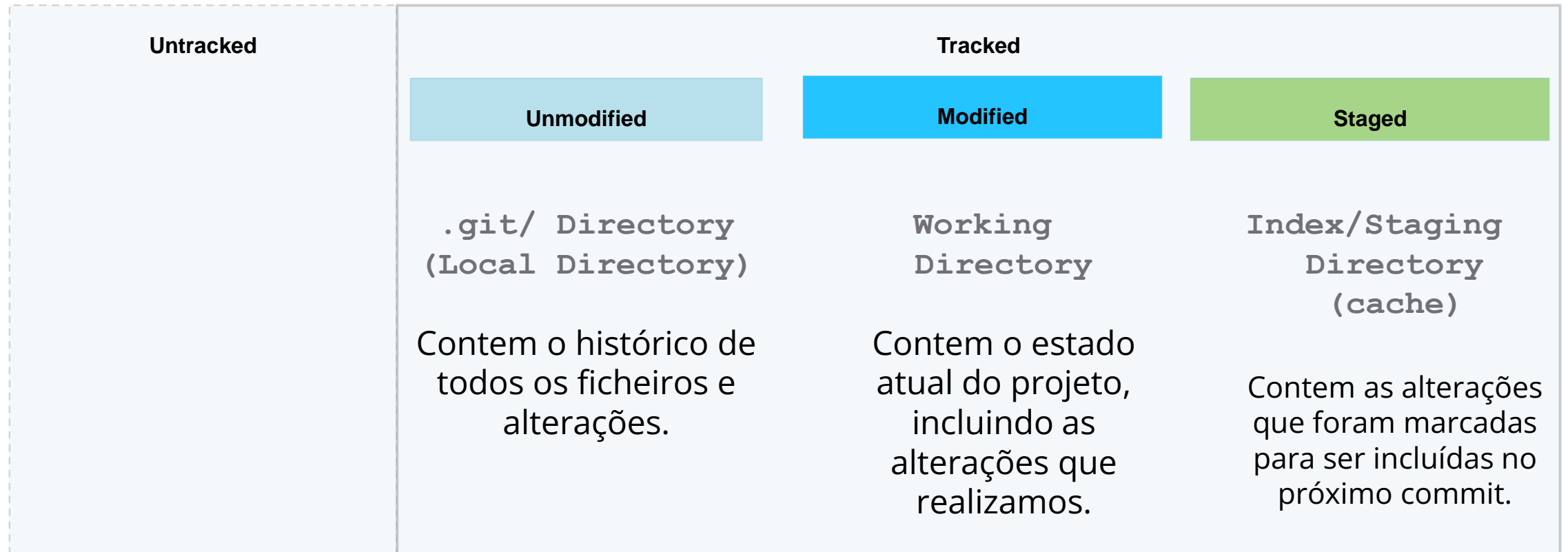
Conceitos sobre Git e GitHub

Os 3 estados dos ficheiros no Git



Conceitos sobre Git e GitHub

As 3 áreas/directorias do Git



Conceitos sobre Git e GitHub

Questão

Qual é o objetivo de um sistema de controlo de versões?

Um sistema de controle de versões permite rastrear quando e quem fez alterações nos ficheiros de um projeto ou repositório. Estes ficheiros podem ser código, configurações, imagens ou qualquer outra coisa que precisemos rastrear.

Setup do ambiente de desenvolvimento colaborativo

Instalação e configuração das ferramentas

Ferramentas recomendadas

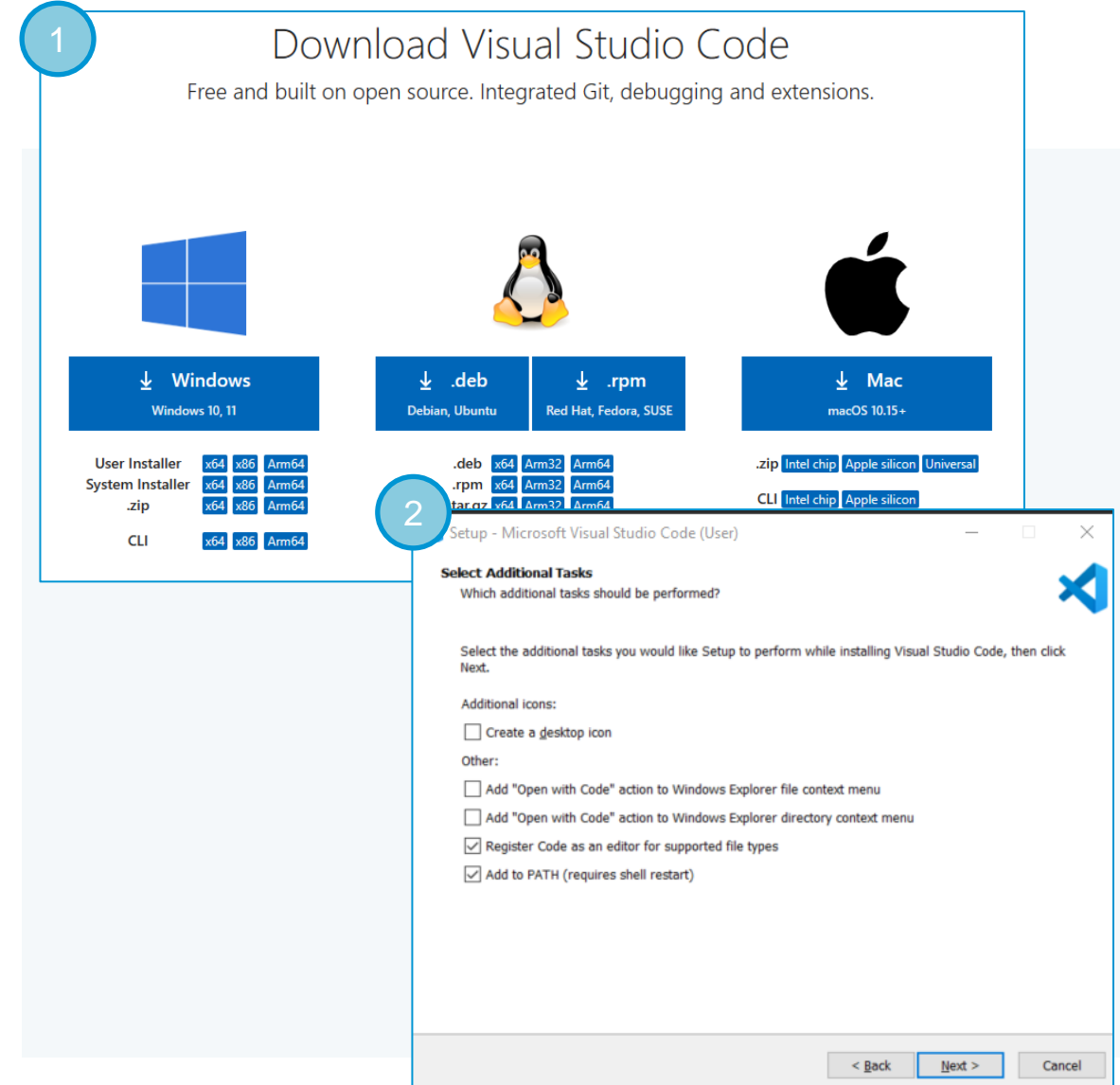
As seguintes ferramentas estão disponíveis para S.O. Windows, Mac e Linux.

Nome da Ferramenta	Tipo	Description	Download
Visual Studio Code (vscode)	Editor de Código fonte	Editor de Código open-source da Microsoft	https://code.visualstudio.com/
Git	Controlo de versões distribuido	Sistema distribuido open-source de controlo de versão	https://git-scm.com/downloads
GitHub (conta)	Alojamento	Serviço de alojamento online para repositórios git.	https://github.com

Ferramentas recomendadas

Instalação do Visual Studio Code (VS Code)

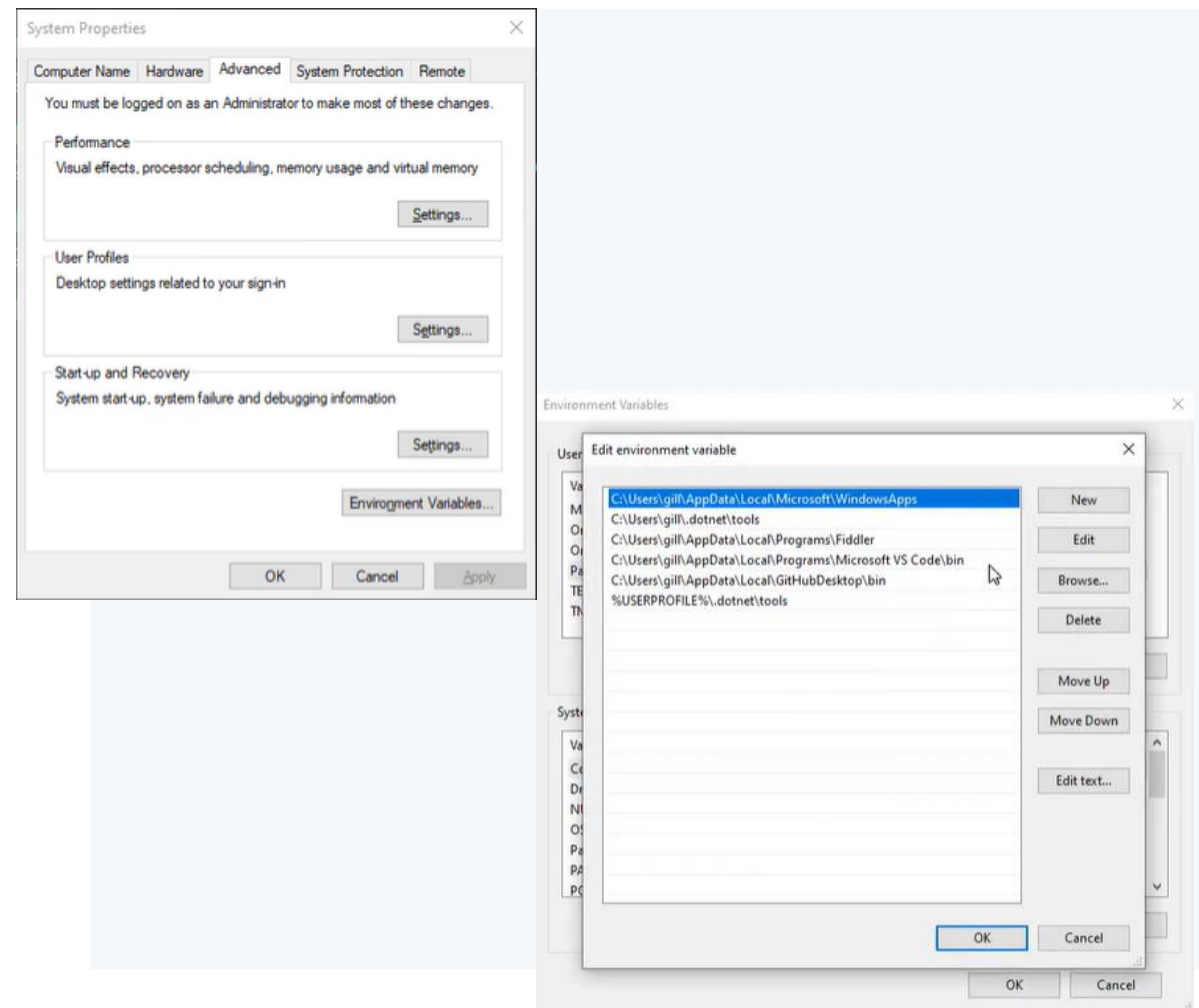
1. Download Visual Studio Code
(<https://code.visualstudio.com/download>)
2. On the 'Additional Tasks' screen select:
Register Code as an editor for supported files types
Add to PATH (requires shall restart)
3. Finish installing Visual Studio Code.



Ferramentas recomendadas

Instalação do Visual Studio Code (VS Code)

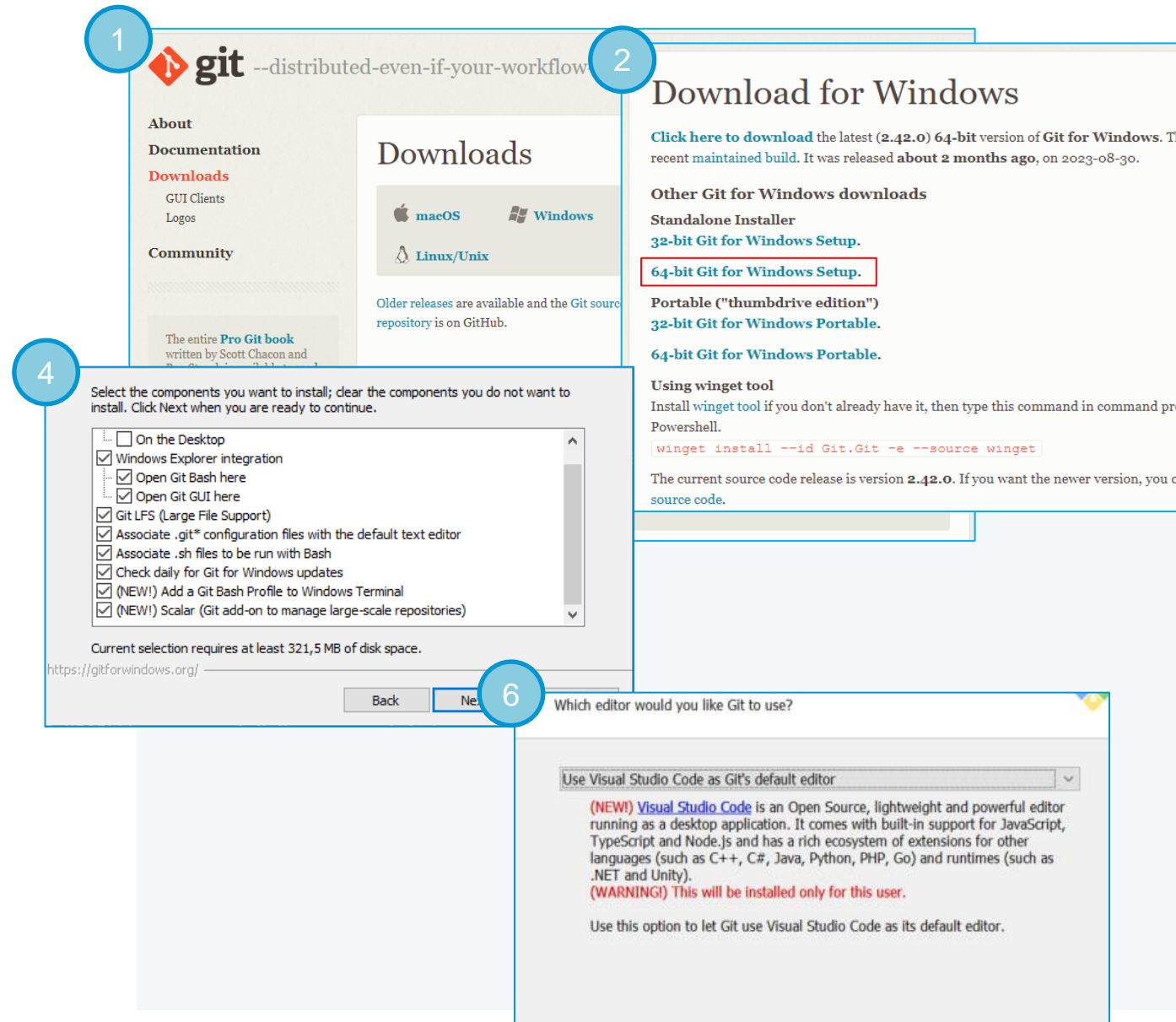
Verificar que o VS Code está definido nas variáveis de ambiente do Windows.



Ferramentas recomendadas

Instalação step-by-step do Git no Windows

1. Aceder ao site <https://git-scm.com/downloads> e clique na opção **Windows**.
2. Descarregue a versão '**64-bit Git for Windows Setup**' na secção '**Standalone Installer**'.
3. Executar o ficheiro de instalação do Git. Clicar em '**Next**'. Aceitar a pasta de destino sugerida e clicar em '**Next**'.
4. No ecrã 'Select Components' seleccionar as seguintes opções, e clicar em '**Next**'.
5. No ecrã "Select Start Menu Folder" clicar em '**Next**'.
6. Seleccionar '**Use Visual Studio Code as Git's default editor**' clicar em '**Next**'.



Installing Git on Windows (Continued)

Instalação step-by-step do Git no Windows (continuação)

7. No ecrã 'Adjust the name of the initial branch in new repositories' aceite a opção **Let Git decide** e clique em Next.
8. Adjusting PATH environment: manter como **'Git from the command line and also 3rd-party software'**.
9. Choosing SSH executable: **Use bundled OpenSSH**.
10. HTTPS Transport backend: **Use the OpenSSL Library**.
11. Configuring the line ending conversations: **Checkout Windows-Style**.

7 Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?

☒ **Let Git decide**
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

☐ **Override the default branch name for new repositories**
NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

Back Next Cancel

8 Adjusting PATH environment

☐ **Use Git from Git Bash only**
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☒ **Git from the command line and also from 3rd-party software**
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software located in your PATH.

☐ **Use Git and optional Unix tools from the Command Prompt (if available)**
Both Git and the optional Unix tools will be added to your PATH. **Warning: This will override Windows tools like "find" and "dir". Use this option if you understand the implications.**

<https://gitforwindows.org/> Back

9 Choosing SSH executable

☒ **Use bundled OpenSSH**
This uses ssh.exe that comes with Git.

☐ **Use external OpenSSH**
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.

Back

10 HTTPS Transport backend

☒ **Use the OpenSSL library**
Server certificates will be validated using the ca-bundle.crt file.

☐ **Use the native Windows Secure Channel library**
Server certificates will be validated using Windows Certificate Store. This option also allows you to use your company's internal Root CA distributed e.g. via Active Directory Domain Services.

Back

11 Configuring the line ending conversations

☒ **Checkout Windows-style, commit Unix-style line endings**
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

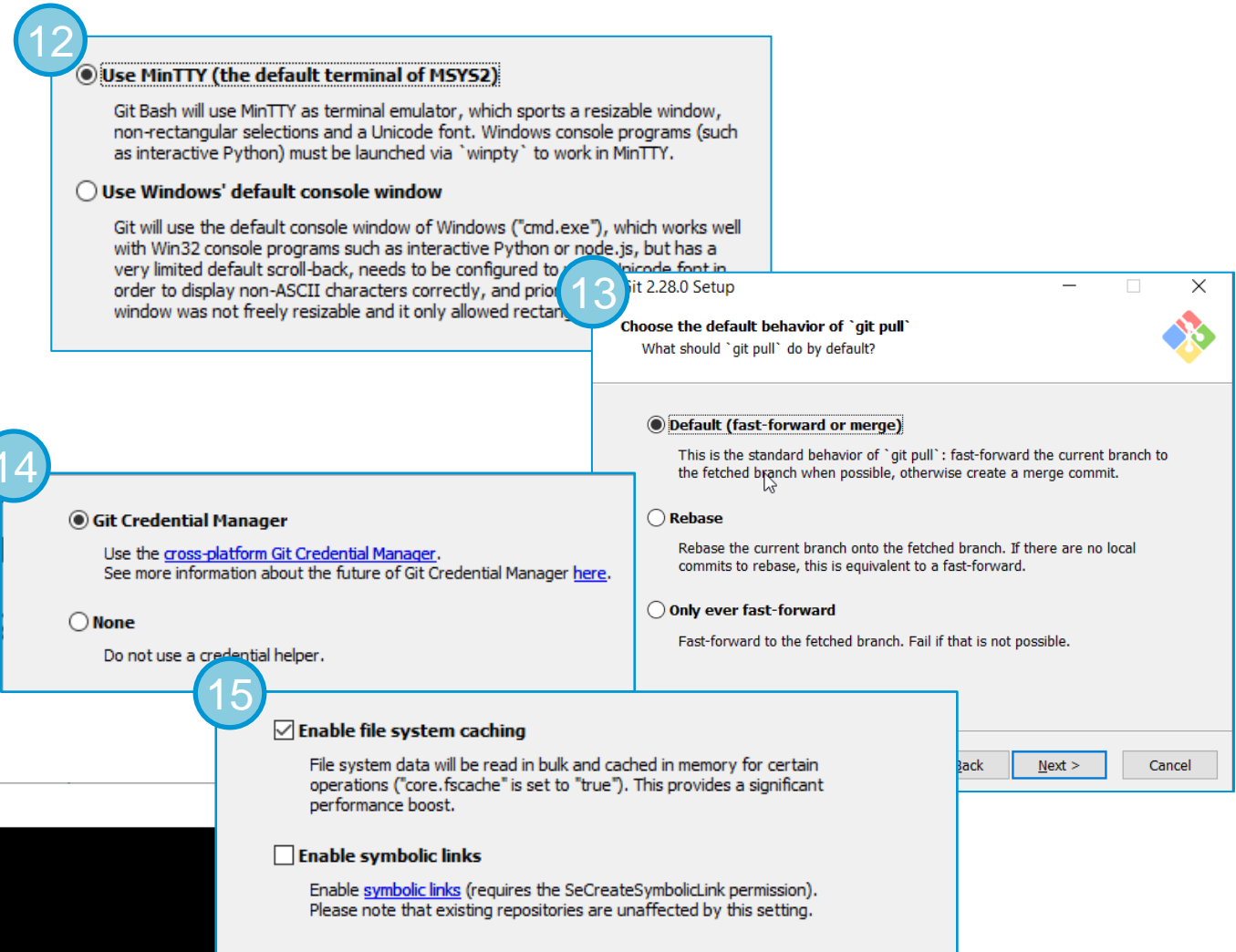
☐ **Checkout as-is, commit Unix-style line endings**
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

☐ **Checkout as-is, commit as-is**
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

Installing Git on Windows (Continued)

Instalação step-by-step do Git no Windows (continuação)

12. Terminal emulator: **Use MinTTY**
13. Default behavior of 'git pull': Choose **Default**.
14. Credential Helper: **Git Credential Manager**
15. Extra options: **Enable File system caching**
16. **Experimental options:** don't select any option. Click **Install**.
17. Abrir o 'Git Bash'.



Git

Criação do primeiro repositório

Git Bash

Comandos essenciais Shell

#	Comando Shell	Descrição
1	pwd	Lista a diretoria corrente/atual
2	mkdir	Cria uma diretoria
3	ls	Lista o conteúdo da diretoria atual
4	dir	Semelhante ao ls.
5	cd	Permite a alteração de diretoria
6	touch	Permite a criação de um novo ficheiro vazio na diretoria atual
7	clear	Limpa o ecrã e trás a linha de comandos para o topo.
8		
9		
10		

Linha de comandos Git Bash

```
MINGW64/c/Users/rmbso
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  diff       Show changes between commits, commit and working tree, etc
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  commit     Record changes to the repository
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  reset      Reset current HEAD to the specified state
  switch     Switch branches
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

rmbso@DESKTOP-NKESK4T MINGW64 ~
$
```

Configure Git for the first time

Configurar o Git

Depois de instalação do Git, é necessário realizar a configuração de um perfil de Git:

- A partir da linha de comandos do Git Bash, configurar os dados do utilizador:

```
git config --global user.name "Lastname, Firstname"
```

```
git config --global user.email alias@domain.com
```

- Exercício de configuração do meu sistema:**

```
git config --global user.name "Rui Sousa"
```

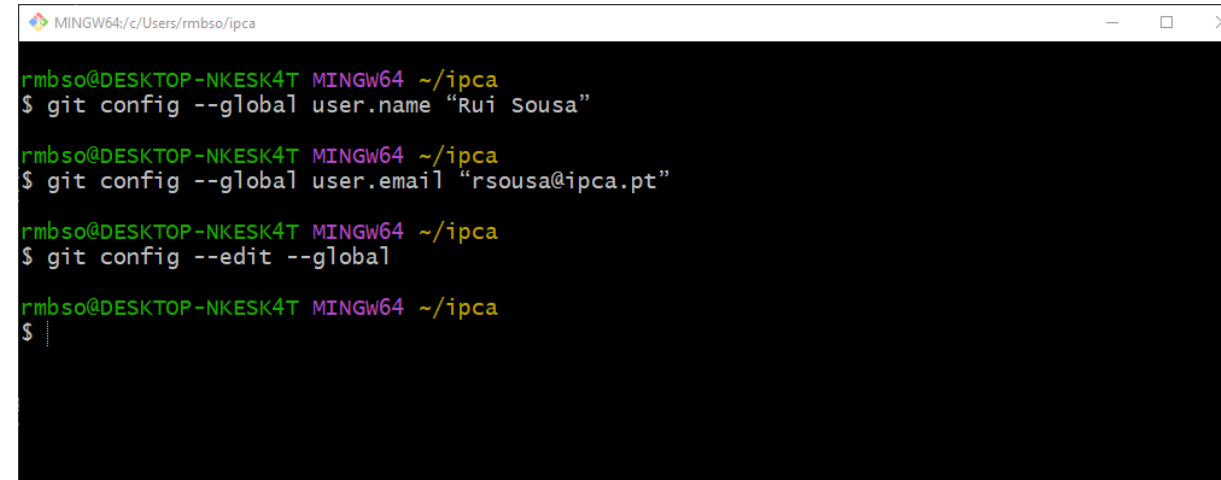
```
git config --global user.email "rsousa@ipca.pt"
```

```
git config -global core.editor "code --wait --new-window"
```

```
git config --edit --global
```

```
Git config --global -list
```

```
Git config --list
```

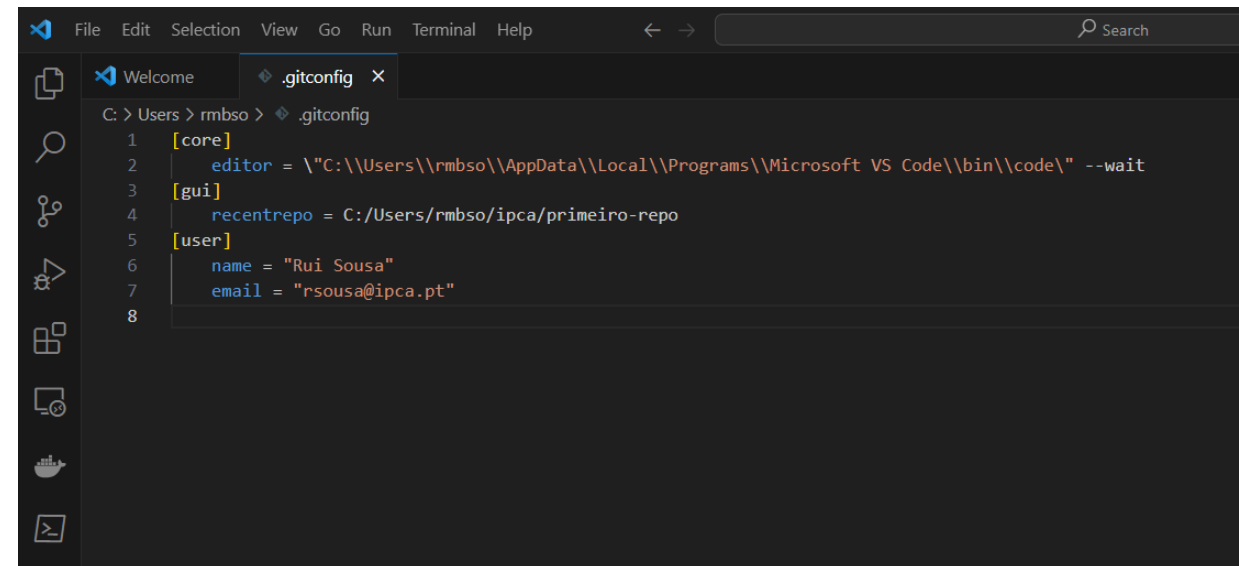


```
MINGW64/c/Users/rmbso/ipca
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --global user.name "Rui Sousa"

rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --global user.email "rsousa@ipca.pt"

rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --edit --global

rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$
```



```
File Edit Selection View Go Run Terminal Help
Welcome .gitconfig x
C:\Users> rmbso> .gitconfig
1 [core]
2   editor = "C:\\Users\\rmbso\\AppData\\Local\\Programs\\Microsoft VS Code\\bin\\code\" --wait
3 [gui]
4   recentrepo = C:/Users/rmbso/ipca/primeiro-repo
5 [user]
6   name = "Rui Sousa"
7   email = "rsousa@ipca.pt"
8
```


Configure Git for the first time

Configurar o Git

- Alterar o ficheiro de configuração do git “c:\users\<username>\.gitconfig”:

```
[filter "lfs"]
```

```
clean = git-lfs clean -- %f
```

```
smudge = git-lfs smudge -- %f
```

```
process = git-lfs filter-process
```

```
required = true
```

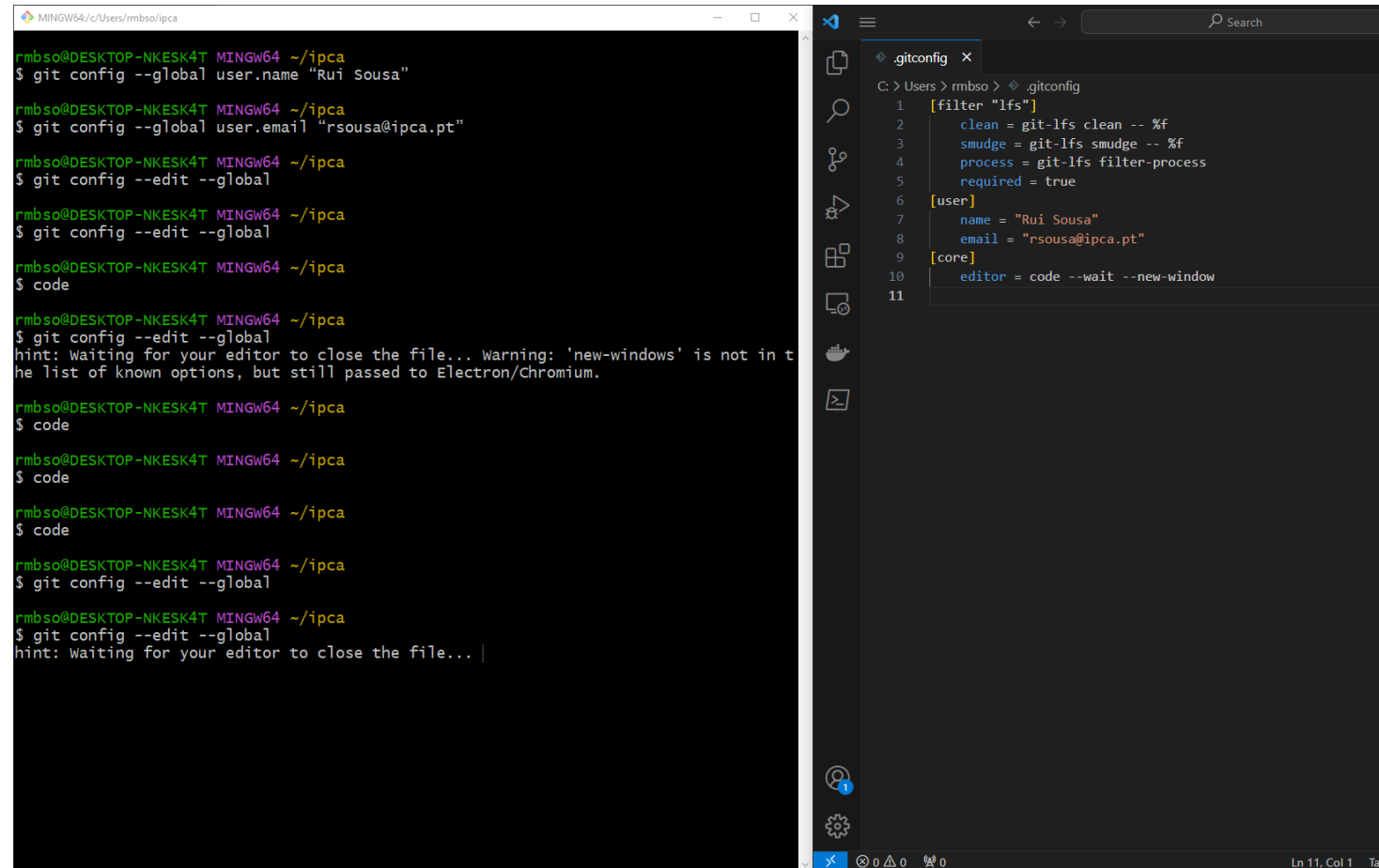
```
[user]
```

```
name = "Rui Sousa"
```

```
email = "rsousa@ipca.pt"
```

```
[core]
```

```
editor = code --wait --new-window
```



```
MINGW64/c:/Users/rmbso/ipca
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --global user.name "Rui Sousa"
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --global user.email "rsousa@ipca.pt"
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --edit --global
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --edit --global
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ code
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --edit --global
hint: Waiting for your editor to close the file... Warning: 'new-windows' is not in the list of known options, but still passed to Electron/Chromium.
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ code
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ code
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --edit --global
rmbso@DESKTOP-NKESK4T MINGW64 ~/ipca
$ git config --edit --global
hint: Waiting for your editor to close the file... |
```

```
.gitconfig
1  [filter "lfs"]
2    clean = git-lfs clean -- %f
3    smudge = git-lfs smudge -- %f
4    process = git-lfs filter-process
5    required = true
6  [user]
7    name = "Rui Sousa"
8    email = "rsousa@ipca.pt"
9  [core]
10  editor = code --wait --new-window
11
```

Trabalhar com repositórios locais

Criar um repositório local

Exercício:

- Passos para criar um repositório local:

1. Criar uma pasta para associada ao repositório

Exemplo: c:\code\ipca

- mkdir "c:\code"
- mkdir "c:\code\ipca"

2. Navegar para a pasta

cd "c:\code\ipca"

3. Criar/Inicializar o repositório

git init

4. Verificar o status do repositório

git status

5. Alterar o nome do branch de master para main

git branch -m main

```
MINGW64/c:/code/ipca

rmbso@DESKTOP-NKESK4T MINGW64 /
$ mkdir "c:\code"

rmbso@DESKTOP-NKESK4T MINGW64 /
$ mkdir "c:\code\ipca"

rmbso@DESKTOP-NKESK4T MINGW64 /
$ cd "c:\code\ipca"

3 rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca
$ git init
Initialized empty Git repository in C:/code/ipca/.git/

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ ls -la
total 4
drwxr-xr-x 1 rmbso 197609 0 oct 28 18:49 ./
drwxr-xr-x 1 rmbso 197609 0 oct 28 18:47 ../
drwxr-xr-x 1 rmbso 197609 0 oct 28 18:49 .git/

4 rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master

No commits yet

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

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ |
```

Trabalhar com repositórios locais

Criar um repositório local

Exercício (continuação):

- Criar um ficheiro na pasta com o nome olamundo.txt

code olamundo.txt

- Executar o comando **git status**

Vermelho:

- Ficheiros que nunca foram gravados no Git
- Ficheiros que foram alterados mas ainda não estão gravados

Verde:

- Os ficheiros que estão sincronizados com o Git

```
MINGW64/c/code/ipca
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ code olamundo.txt

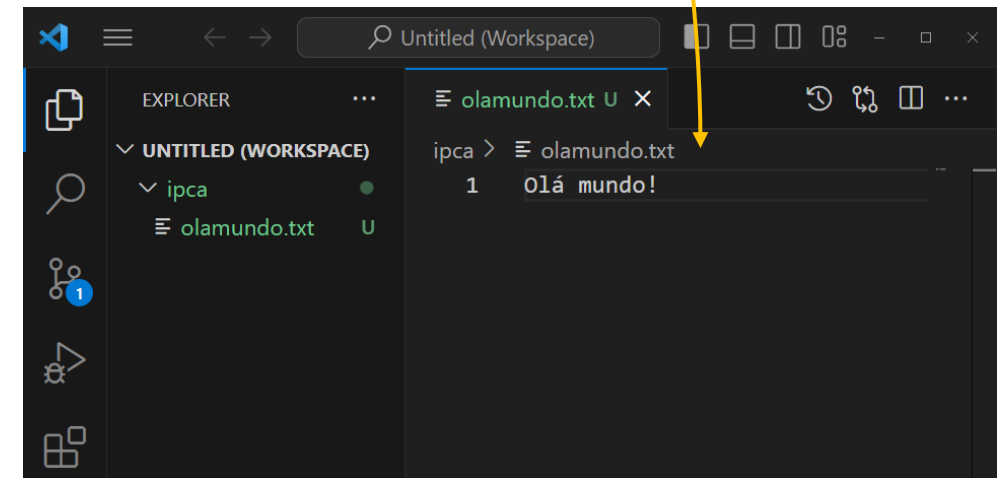
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master

No commits yet

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

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

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$
```



Trabalhar com repositórios locais

Criar um repositório local

Exercício (continuação):

1. Adicionar o ficheiro olamundo.txt ao repositório.
Comando: **git add olamundo.txt**
2. Verificar o status
Comando: **git status**
3. Realizar o primeiro commit de dados
Comando: **git commit -m "Primeira versão"**

```
MINGW64/c/code/ipca
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master

No commits yet

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

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

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git add olamundo.txt

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git commit -m "Primeira versão"
[master (root-commit) 2c7f077] Primeira versão
1 file changed, 1 insertion(+)
create mode 100644 olamundo.txt

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$
```

Trabalhar com repositórios locais

Criar um repositório local

Exercício (continuação):

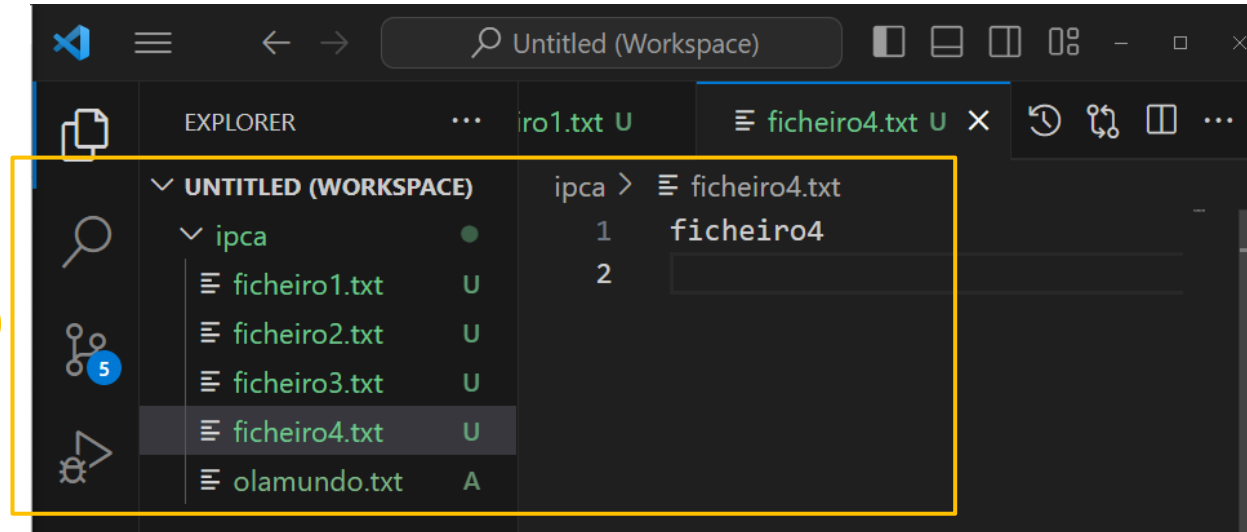
1. Criar vários ficheiros:
 - Exemplo: ficheiro1.txt, ficheiro2.txt, ficheiro3.txt e ficheiro4.txt
2. Gravar todos os documentos
3. Comando: **git status**
4. Adicionar todos os ficheiros ao repositório
Comando: **git add .**

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git add .

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master

No commits yet

changes to be committed:
(use "git rm --cached <file>..." to unstage)
    new file:   ficheiro1.txt
    new file:   ficheiro2.txt
    new file:   ficheiro3.txt
    new file:   ficheiro4.txt
    new file:   olamundo.txt
```



```
MINGW64/c/code/ipca

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ cat ficheiro1.txt
ficheiro 1

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ cat ficheiro1.txt
ficheiro 1

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ficheiro1.txt
        ficheiro2.txt
        ficheiro3.txt
        ficheiro4.txt

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

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$
```

Trabalhar com repositórios locais

Criar um repositório local

Exercício (continuação):

1. Realizar o commit de dados
Comando: **git commit -m "Segunda versão"**
2. Criar a versão do nosso projeto
3. Só são enviados os ficheiros que:
Estão na lista, usando o comando git add
4. Verificar se todos os ficheiros foram committed
Comando: **git status**
5. Consultar o histórico de logs das versões submetidas.
Comando: **git log**
6. Depois correr o comando: gitk
7. Depois correr o comando: git log -p filename

1

```
MINGW64: /c/code/ipca
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git commit -m "Segunda versão"
[master ab01222] Segunda versão
4 files changed, 4 insertions(+)
create mode 100644 ficheiro1.txt
create mode 100644 ficheiro2.txt
create mode 100644 ficheiro3.txt
create mode 100644 ficheiro4.txt
```

4

```
MINGW64: /c/code/ipca
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master
nothing to commit, working tree clean
```

5

```
MINGW64: /c/code/ipca
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git log
commit ab0122245d915bc4ca7df7df8364e260ee1aaa33 (HEAD -> master)
Author: Rui Sousa <rsousa@ipca.pt>
Date: Sat Oct 28 20:15:13 2023 +0100

    Segunda versão

commit 2c7f077dc0bfd932cffd88b53c79d57df566cecc
Author: Rui Sousa <rsousa@ipca.pt>
Date: Sat Oct 28 20:08:51 2023 +0100

    Primeira versão
```

```
MINGW64: /c/code/ipca
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$
```

Git

Comandos essenciais

#	Comando Git	Descrição
1	git	Lista os comandos comuns do Git
2	git config	Lista e define as opções de configuração do Git
3	git config --get user.name	Consulta o nome do utilizador
4	git config user.name "<Desired Username>"	Alterar o nome do utilizador
5	git config --get user.email	Consulta o endereço de email configurado no Git
6	git config user.name "<Desired Email>"	Alterar o endereço de email do utilizador
7	git init	Cria um novo repositório de Git local. Pode ser usado para transformar o diretório atual num repositório Git. Também pode ser usado para converter um projeto existente sem versão num repositório Git ou para inicializar um repositório novo e vazio.
8	git clone	Efetua o download de um projeto de um repositório remote
9	git add	Prepara um ficheiro para a área de Staging preparando para um commit
10	git add . Ou git add *	Prepara vários ficheiros para a área de Staging para um commit

Git

Comandos essenciais

#	Comando Git	Descrição
10	git status	List the files you've changed and those you still need to add or commit
11	git commit	Cria um snapshot e efetua o Commit de um ficheiro ou vários ficheiros para o repositório
12	git commit -m " <commit message> "	
13	git commit -a -m " <commit message> "	Allows you to move changes from the working directory to the staging area and takes a staged snapshot to commit it to the project history.
14	git log	Mostra o histórico de alterações submetidas para o repositório.
15	git diff	Mostra as alterações entre a 'Working directory' e a área de staging.

Existe um total de 120 comandos no git com opções para estender as suas capacidades. Cerca de 20 ou 30 comandos são considerados a fundação ou comandos essenciais do Git.

Para consultar a ajuda de um commando podem sem executar o seguinte comando:

```
git <comando> --help  
exemplo: git add --help
```


Comandos essenciais

git add --help

git-add(1) Manual Page

NAME

git-add - Add file contents to the index

SYNOPSIS

```
git add [--verbose | -v] [--dry-run | -n] [--force | -f] [--interactive | -i] [--patch | -p]
        [--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]] [--sparse]
        [--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing] [--renormalize]
        [--chmod=(+|-)x] [--pathspec-from-file=<file>] [--pathspec-file-nul]
        [--] [<pathspec>...]
```

DESCRIPTION

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working tree, and before running the commit command, you must use the add command to add any new or modified files to the index.

This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at the time the add command is run; if you want subsequent changes included in the next commit, then you must run git add again to add the new content to the index.

The git status command can be used to obtain a summary of which files have changes that are staged for the next commit.

The git add command will not add ignored files by default. If any ignored files were explicitly specified on the command line, git add will fail with a list of ignored files. Ignored files reached by directory recursion or filename globbing performed by Git (quote your globs before the shell) will be silently ignored. The git add command can be used to add ignored files with the -f (force) option.

Please see git-commit(1) for alternative ways to add content to a commit.

Conceitos sobre Git e GitHub

Questões

Q1: Quando fazemos o commit de novos ficheiros ou alterações, é pedido ao utilizador para introduzir uma mensagem de commit. O que acontece se for introduzida uma mensagem vazia de commit?

R1: O commit é abortado. Não podemos fazer um commit com uma mensagem vazia.

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)  
$ git commit -m ""  
Aborting commit due to empty commit message.
```

Q2: Que conteúdo devemos incluir na mensagem do commit?

R2: A mensagem de commit deve ter uma pequena descrição da alteração (até 50 caracteres), seguido de 1 ou mais paragrafos com mais detalhe sobre a alteração (caso seja necessário).

Q3: Antes que alterações em novos ficheiros possam ser adicionadas/realizadas no diretório Git, qual comando que dirá ao Git para rastrear os novos ficheiros na lista de alterações a serem confirmadas?

R3: git add

Q4: Qual comando usaríamos para rever o histórico de commits do nosso projeto?

R4: git log

Conceitos sobre Git e GitHub

Questões

Q5: Qual comando usaríamos para fazer o Git rastrear o nosso ficheiro?

R5: `git add`

Q6: Qual o comando que devemos usar para rever a nossa configuração?

R6: `git config -l`

Q7: Qual comando que usaríamos para visualizar as alterações pendentes?

R7: `git status`

Q8: O que acontece se criamos um novo ficheiro dentro de um repositório Git?

R8: O ficheiro é marcado como untracked. É necessário adicionar ao ficheiro ao repo usando o comando `git add`, ou em alternativa fazer um commit com os parametros `-am`

Trabalhar com repositórios locais

Introduzir alterações no ficheiro

Exercício (continuação):

1. Fazer alterações ao ficheiro `ficheiro1.txt`
2. Salvar e sincronizar o repositório
3. Verificar o resultado
Comando: **git status**
4. Enviar as alterações efetuadas
Comando:
git add .
git commit -m "Alteração do ficheiro1"

The image displays three sequential screenshots of the Visual Studio Code interface, illustrating the steps to make a commit:

- Step 1:** The Explorer sidebar shows a workspace named 'ipca' containing several files. The file `ficheiro1.txt` is selected and opened in the editor. The editor shows the following content:

```
ipca > ficheiro1.txt
1 ficheiro 1
2 alteração
```
- Step 2:** The terminal window shows the output of the `git status` command, indicating that `ficheiro1.txt` is modified and ready to be committed.

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   ficheiro1.txt
```
- Step 3:** The terminal window shows the output of the `git add .` and `git commit -m "alteração do ficheiro1"` commands, confirming that the changes have been staged and committed.

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git add .

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git commit -m "alteração do ficheiro1"
[master 917fe1b] alteração do ficheiro1
1 file changed, 1 insertion(+)

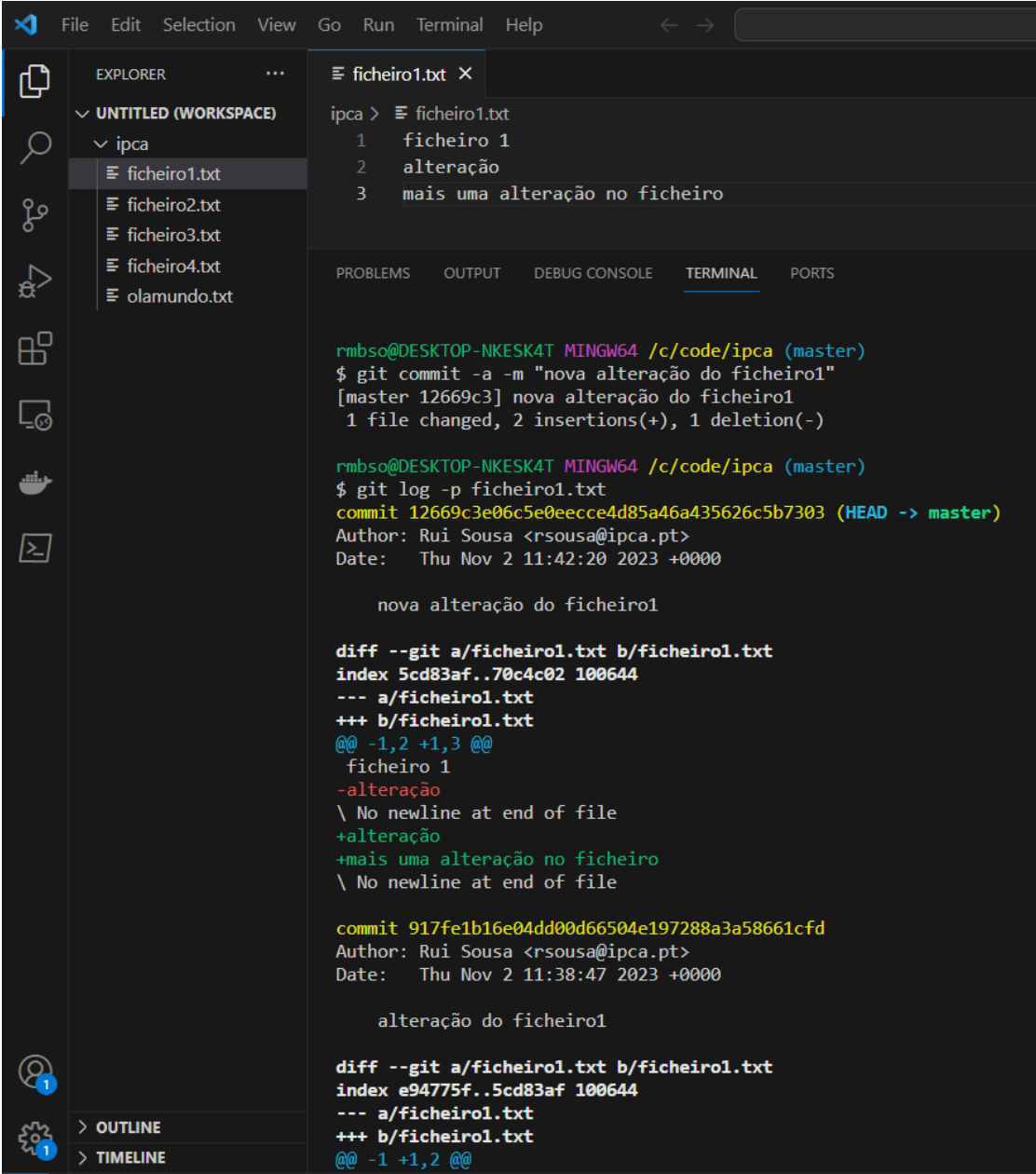
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$
```

Trabalhar com repositórios locais

Introduzir novas alterações no ficheiro

Exercício (continuação):

1. Fazer uma nova alteração no ficheiro `ficheiro1.txt`
2. Salvar e sincronizar o repositório
3. Verificar o resultado
4. Depois correr o comando: `git log -p filename`
5. Verificar o resultado



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  UNTITLED (WORKSPACE)
    ipca
      ficheiro1.txt
      ficheiro2.txt
      ficheiro3.txt
      ficheiro4.txt
      olamundo.txt
ficheiro1.txt X
ipca > ficheiro1.txt
1  ficheiro 1
2  alteração
3  mais uma alteração no ficheiro
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git commit -a -m "nova alteração do ficheiro1"
[master 12669c3] nova alteração do ficheiro1
1 file changed, 2 insertions(+), 1 deletion(-)
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git log -p ficheiro1.txt
commit 12669c3e06c5e0eccc4d85a46a435626c5b7303 (HEAD -> master)
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:42:20 2023 +0000

    nova alteração do ficheiro1

diff --git a/ficheiro1.txt b/ficheiro1.txt
index 5cd83af..70c4c02 100644
--- a/ficheiro1.txt
+++ b/ficheiro1.txt
@@ -1,2 +1,3 @@
 ficheiro 1
-alteração
\ No newline at end of file
+alteração
+mais uma alteração no ficheiro
\ No newline at end of file

commit 917fe1b16e04dd00d66504e197288a3a58661cfd
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:38:47 2023 +0000

    alteração do ficheiro1

diff --git a/ficheiro1.txt b/ficheiro1.txt
index e94775f..5cd83af 100644
--- a/ficheiro1.txt
+++ b/ficheiro1.txt
@@ -1 +1,2 @@
```

Trabalhar com repositórios locais

Analisar o log de alterações – git log

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git log
commit 12669c3e06c5e0eacce4d85a46a435626c5b7303 (HEAD -> master)
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:42:20 2023 +0000

    nova alteração do ficheiro1

commit 917fe1b16e04dd00d66504e197288a3a58661cfd
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:38:47 2023 +0000

    alteração do ficheiro1

commit ab0122245d915bc4ca7df7df8364e260ee1aaa33
Author: Rui Sousa <rsousa@ipca.pt>
Date: Sat Oct 28 20:15:13 2023 +0100

    Segunda versão

commit 2c7f077dc0bfd932cffd88b53c79d57df566cecc
Author: Rui Sousa <rsousa@ipca.pt>
Date: Sat Oct 28 20:08:51 2023 +0100

    Primeira versão
```

Trabalhar com repositórios locais

Analisar o log de alterações de um ficheiro – git log -p

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git log -p
commit 12669c3e06c5e0eecce4d85a46a435626c5b7303 (HEAD -> master)
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:42:20 2023 +0000

    nova alteração do ficheiro1

diff --git a/ficheiro1.txt b/ficheiro1.txt
index 5cd83af..70c4c02 100644
--- a/ficheiro1.txt
+++ b/ficheiro1.txt
@@ -1,2 +1,3 @@
 ficheiro 1
-alteração
\ No newline at end of file
+alteração
+mais uma alteração no ficheiro
\ No newline at end of file
```

```
commit 917fe1b16e04dd00d66504e197288a3a58661cfd
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:38:47 2023 +0000

    alteração do ficheiro1

diff --git a/ficheiro1.txt b/ficheiro1.txt
index e94775f..5cd83af 100644
--- a/ficheiro1.txt
+++ b/ficheiro1.txt
@@ -1 +1,2 @@
 ficheiro 1
+alteração
\ No newline at end of file
```

Trabalhar com repositórios locais

Analisar o log de alterações de um commit – outra opção

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git show 12669c3e06c5e0eecce4d85a46a435626c5b7303
commit 12669c3e06c5e0eecce4d85a46a435626c5b7303 (HEAD -> master)
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:42:20 2023 +0000

    nova alteração do ficheiro1

diff --git a/ficheiro1.txt b/ficheiro1.txt
index 5cd83af..70c4c02 100644
--- a/ficheiro1.txt
+++ b/ficheiro1.txt
@@ -1,2 +1,3 @@
     ficheiro 1
-    alteração
\ No newline at end of file
+    alteração
+    mais uma alteração no ficheiro
\ No newline at end of file
```


Trabalhar com repositórios locais

Analisar o log de alterações de um commit – obter estatísticas de alterações

```
$ git log --stat
commit 12669c3e06c5e0eecce4d85a46a435626c5b7303 (HEAD -> master)
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:42:20 2023 +0000

    nova alteração do ficheiro1

    ficheiro1.txt | 3 ++-
    1 file changed, 2 insertions(+), 1 deletion(-)

commit 917fe1b16e04dd00d66504e197288a3a58661cfd
Author: Rui Sousa <rsousa@ipca.pt>
Date: Thu Nov 2 11:38:47 2023 +0000

    alteração do ficheiro1

    ficheiro1.txt | 1 +
    1 file changed, 1 insertion(+)

commit ab0122245d915bc4ca7df7df8364e260ee1aaa33
Author: Rui Sousa <rsousa@ipca.pt>
Date: Sat Oct 28 20:15:13 2023 +0100

    Segunda versão

    ficheiro1.txt | 1 +
    ficheiro2.txt | 1 +
    ficheiro3.txt | 1 +
    ficheiro4.txt | 1 +
    4 files changed, 4 insertions(+)

commit 2c7f077dc0bfd932cffd88b53c79d57df566cecc
Author: Rui Sousa <rsousa@ipca.pt>
Date: Sat Oct 28 20:08:51 2023 +0100

    Primeira versão

    olamundo.txt | 1 +
    1 file changed, 1 insertion(+)
```


Trabalhar com repositórios locais

Criar um novo repositório local e realizar alterações

- Criar uma pasta com o nome “Segundo projeto”
- Criar o ficheiro index.html

```
<h1>Hello World</h1>
<p>O meu primeiro projeto com GIT</p>
```
- Fazer o commit do projeto com a mensagem: segundo projeto
- Alterar o conteúdo para:

```
<p>O meu segundo projeto com GIT</p>
```
- Fazer um commit com a mensagem “corrigir o erro no ficheiro”
- Adicionar um novo ficheiro à pasta “Readme.txt” com alguma informação do projeto
- Fazer um commit com a mensagem “adicionar o ficheiro Readme.txt”
- Alterar o conteúdo para:

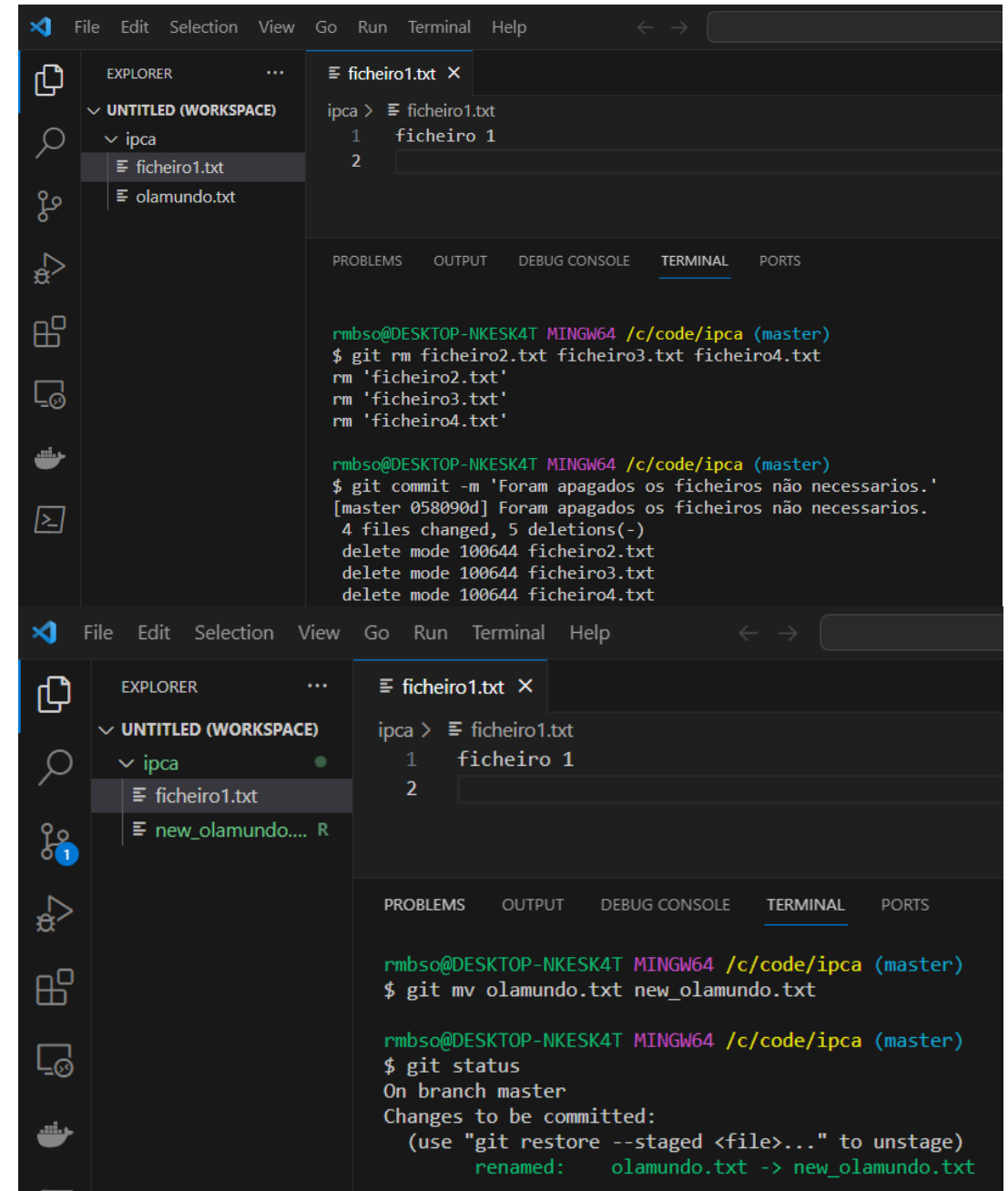
```
<p>O meu terceiro projeto com GIT</p>
```
- Fazer um commit com a mensagem “correcao de novo erro”
- Agora vamos recuperar a versão 2 do ficheiro

Trabalhar com repositórios locais

Apagar e renomear ficheiros

Exercício: Remover e renomear ficheiros

1. Remover ficheiros do repositório
comando: **git rm ficheiro2.txt**
2. Verificar os status
3. Commit das alterações
comando: **git commit -m "Apagado ficheiro2 não necessário"**
4. Verificar os logs de alterações
5. Renomear um ficheiro
comando: **git mv olamundo.txt new_olamundo.txt**
6. Commit das alterações
comando: **git commit -m "Alterado o nome do ficheiro olamundo.txt para new_olamundo.txt"**



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  UNTITLED (WORKSPACE)
    ipca
      ficheiro1.txt
      olamundo.txt
ficheiro1.txt X
ipca > ficheiro1.txt
1 ficheiro 1
2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git rm ficheiro2.txt ficheiro3.txt ficheiro4.txt
rm 'ficheiro2.txt'
rm 'ficheiro3.txt'
rm 'ficheiro4.txt'

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git commit -m 'Foram apagados os ficheiros não necessarios.'
[master 058090d] Foram apagados os ficheiros não necessarios.
4 files changed, 5 deletions(-)
delete mode 100644 ficheiro2.txt
delete mode 100644 ficheiro3.txt
delete mode 100644 ficheiro4.txt

File Edit Selection View Go Run Terminal Help
EXPLORER
  UNTITLED (WORKSPACE)
    ipca
      ficheiro1.txt
      new_olamundo.... R
ficheiro1.txt X
ipca > ficheiro1.txt
1 ficheiro 1
2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git mv olamundo.txt new_olamundo.txt

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    olamundo.txt -> new_olamundo.txt
```

Git

Comandos essenciais

#	Comando Git	Descrição
16	<code>git log -p</code>	Apresenta alterações e informação de patch dos commits.
17	<code>git show</code>	Apresenta várias alterações de um commit
18	<code>git diff</code>	Apresenta diferenças introduzidas nos commits
19	<code>git mv</code>	Renomear um ficheiro do repo
20	<code>git rm</code>	Remover um ficheiro do repo
21	<code>git checkout</code>	Restaura o ficheiro para o ultimo snapshot armazenado, revertendo todas as alterações antes da fase de staging.
22	<code>git reset</code>	Restaura o ficheiro para a partir de um commit.
23	<code>git commit --amend</code>	Permite modificar e adicionar alterações ao commit mais recente. -> Usar apenas em repositório locais!
24	<code>git revert</code>	Permite realizar um novo commit para reverter as alterações. Este comando cancela as ultimas alterações submetendo um novo commit.
25	<code>git branch -m master master</code>	Rename do branch

Conceitos sobre Git e GitHub

Questões

Q1: Qual comando que devo usar para verificar as alterações efetuadas num ficheiro usando o Git?

R1: `git diff <ficheiro>`

Q2: Se estivermos a fazer pequena alteração e quisermos pular a etapa de staging, quais são os dois parametros (flags) que precisamos adicionar ao comando `git commit`?

R2:
-m – permite adicionar a mensagem de commit no comando.
-a – permite adicionar e fazer o commit no mesmo passo.

Q3: Se quiser analisar um commit especifico qual é o comando que devo user identificado o commit id?

R3: `git show <commit id>`

Q4: Que tipo de ficheiros o comando `git commit -a` não submete?

R4: Novos ficheiros

Conceitos sobre Git e GitHub

Commit ID

```
commit ab0122245d915bc4ca7df7df8364e260ee1aaa33
Author: Rui Sousa <rsousa@ipca.pt>
Date:   Sat Oct 28 20:15:13 2023 +0100
```

- O Commit ID são hashes únicos (SHA-1) que são criados quando é registado num novo commit.
- Representa um ponto fixo na história do repositório.
- O commit ID Providencia consistência que é critico para um sistema de controlo de versões
- Os commit ID são gerados a utilizando a mensagem de commit, data, autor e o snapshot tirado à working directory.
- São compostos de 40 caracteres.

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ echo -n 'ab0122245d915bc4ca7df7df8364e260ee1aaa33' | wc -c
40
```

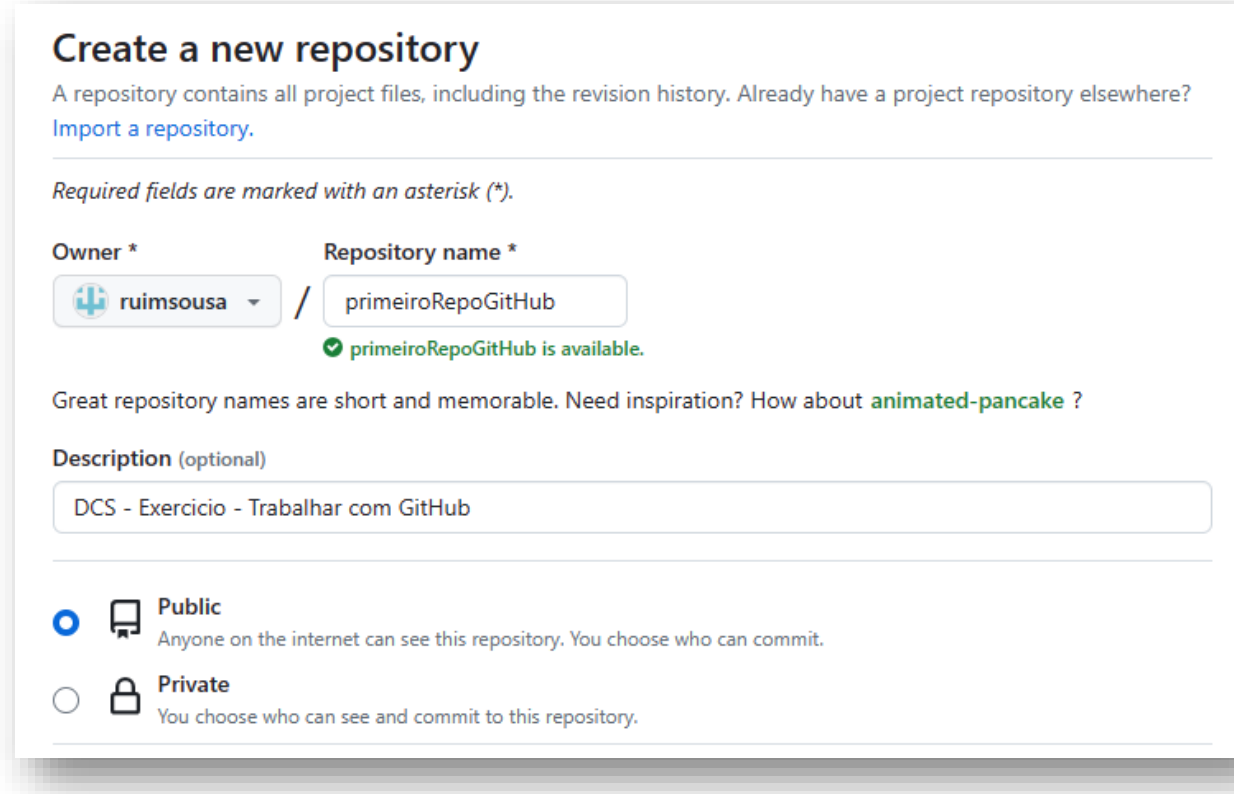
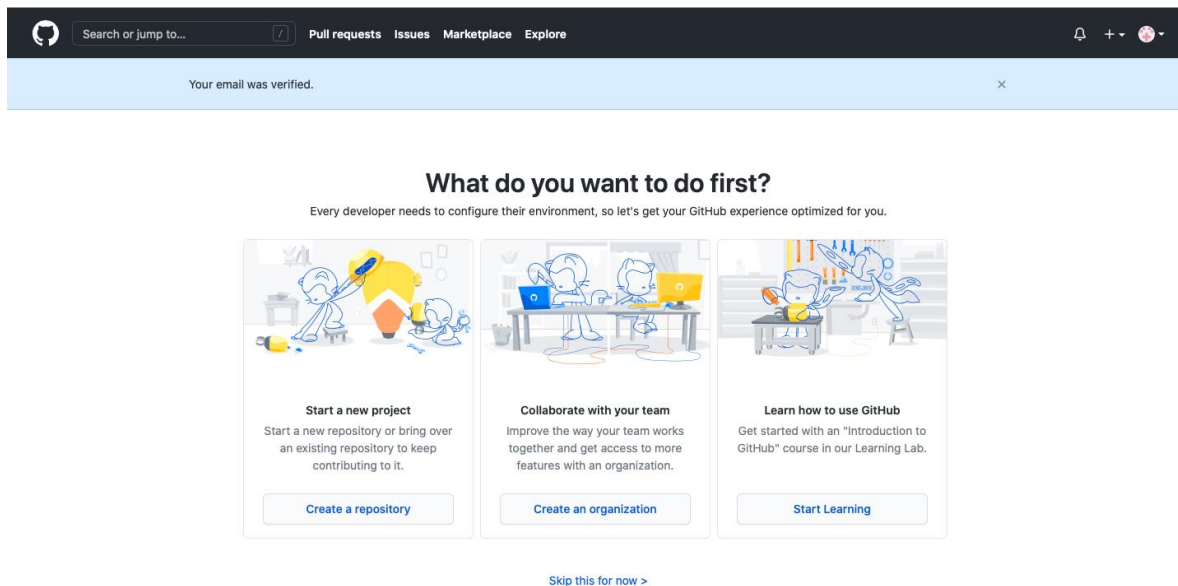
GitHub

Trabalhar com repositórios remotos

Trabalhar com o GitHub

Criar uma conta no GitHub

- Criar uma conta no GitHub (pode ser como estudante)
- Procurar a opção novo repositório
- Criar o novo repositório



Trabalhar com o GitHub

Ligar Git ao Github e fazer push

- **Guardar as credenciais do GitHub**
Comando: `git config --global credential.helper store`
- **Adicionar as credenciais**
Comando: `git config --global credential.username "username"`
Comando: `git config --global credential.password "password"`

alternativa

Comando: `git config --local credential.username "username"`
Comando: `git config --local credential.password "password"`

- **Alterar o repositório**
Comando: `git remote add origin "url"`
- **Ligar o repositório local ao repositório remote**
Comando: `git remote add origin "url_do_respositorio"`
Ou: `git remote set-url origin "url_do_respositorio"`
- **Enviar os ficheiros para o GitHub**
Comando: `git push origin master`
Ou: `git push -f -u origin <name of branch>`

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git remote add origin https://github.com/ruimsousa/primeiroRepoGitHub.git

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (master)
$ git push origin master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (12/12), 862 bytes | 431.00 KiB/s, done.
Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/ruimsousa/primeiroRepoGitHub/pull/new/master
remote:
To https://github.com/ruimsousa/primeiroRepoGitHub.git
 * [new branch]      master -> master
```

The screenshot shows the GitHub web interface for a repository named 'primeiroRepoGitHub' (Public). At the top, there are buttons for 'Pin', 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below this, the repository is shown with 'main' branch selected, '1 branch', and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. The 'About' section on the right indicates the repository is for 'DCS - Exercício - Trabalhar com GitHub', has '0 stars', '1 watching', and '0 forks'. The 'Releases' section shows 'No releases published' with a link to 'Create a new release'. The 'Packages' section shows 'No packages published' with a link to 'Publish your first package'. The main content area displays a commit history table:

Commit Message	Author	Time	Commits
Rui Sousa push github repo	00b0b22	19 minutes ago	4 commits
ficheiro1.txt	alteração do ficheiro1	4 hours ago	
ficheiro2.txt	Segunda versão	5 days ago	
ficheiro3.txt	Segunda versão	5 days ago	
ficheiro4.txt	Segunda versão	5 days ago	
new_olamundo.txt	push github repo	19 minutes ago	
newfile2.txt	push github repo	19 minutes ago	
olamundo.txt	Primeira versão	5 days ago	

At the bottom, there is a light blue box with the text 'Help people interested in this repository understand your project by adding a README.' and a green button labeled 'Add a README'.

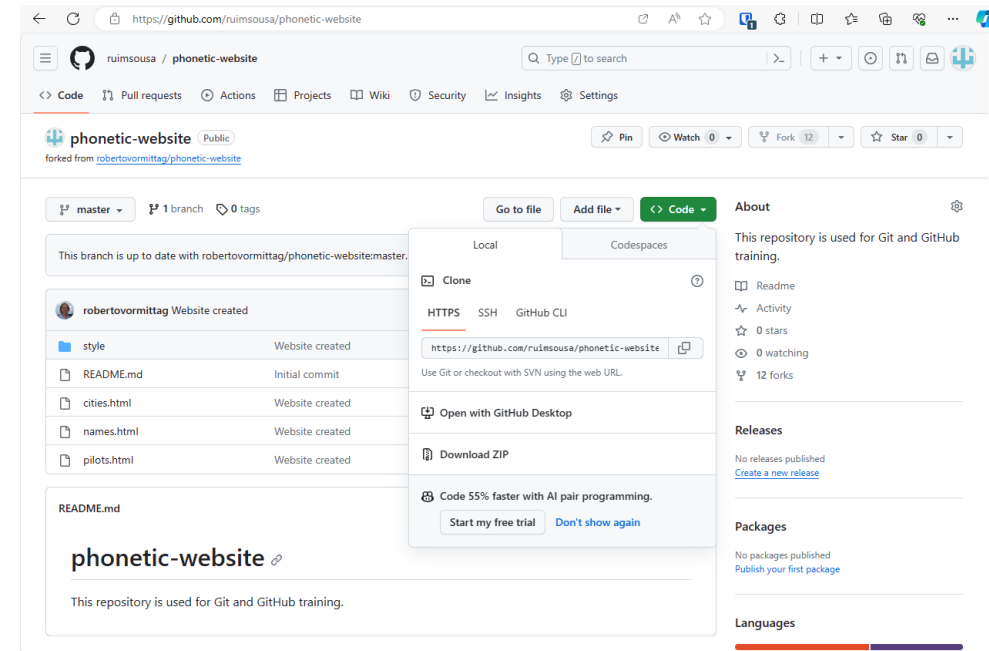
Trabalhar com o GitHub

Clone de um projeto alojado no GitHub

- Fazer download de um projeto no interface web é simples, seguindo a imagem.
- Na linha de comando podes usar:
cd /c/code
mkdir phonetic-website
Comando: `git clone 'url'`

git clone https://github.com/ruimsousa/phonetic-website.git

- Abrir o repositório no VS code
code .



```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code
$ mkdir phonetic-website

rmbso@DESKTOP-NKESK4T MINGW64 /c/code
$ git clone https://github.com/ruimsousa/phonetic-website.git
Cloning into 'phonetic-website'...
remote: Enumerating objects: 13, done.
remote: Total 13 (delta 0), reused 0 (delta 0), pack-reused 13
Receiving objects: 100% (13/13), done.
Resolving deltas: 100% (1/1), done.

rmbso@DESKTOP-NKESK4T MINGW64 /c/code
$ ls
coffee-shop-recipes/  ipca/  phonetic-website/

rmbso@DESKTOP-NKESK4T MINGW64 /c/code
$ cd phonetic-website/

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/phonetic-website (master)
$ ls -lias
total 12
10977524091825620 4 drwxr-xr-x 1 rmbso 197609  0 Nov  2 16:18 ./
19140298416595862 0 drwxr-xr-x 1 rmbso 197609  0 Nov  2 16:18 ../
3659174697306618 4 drwxr-xr-x 1 rmbso 197609  0 Nov  2 16:18 .git/
11821949021975708 1 -rw-r--r-- 1 rmbso 197609 571 Nov  2 16:18 cities.html
17169973579497424 1 -rw-r--r-- 1 rmbso 197609 569 Nov  2 16:18 names.html
5066549581059592 1 -rw-r--r-- 1 rmbso 197609 568 Nov  2 16:18 pilots.html
2814749767231582 1 -rw-r--r-- 1 rmbso 197609  74 Nov  2 16:18 README.md
29273397578217314 0 drwxr-xr-x 1 rmbso 197609  0 Nov  2 16:18 style/
```

Trabalhar com o GitHub

Clone de um projeto alojado no GitHub

- Criar um novo repositório no GitHub para o exercício 1
- Ligar o repositório local ao repositório remoto (GitHub)
- Alterar o ficheiro “Readme.txt”: adicionar a informação do URL do GitHub
- Fazer novo commit
- Abrir o repositório no VS code **code .**

Branches

Branches

Trabalhar com “branches”

Esta é uma funcionalidade interessante dos sistemas Git, ou seja, Ramos ou “Branches”.

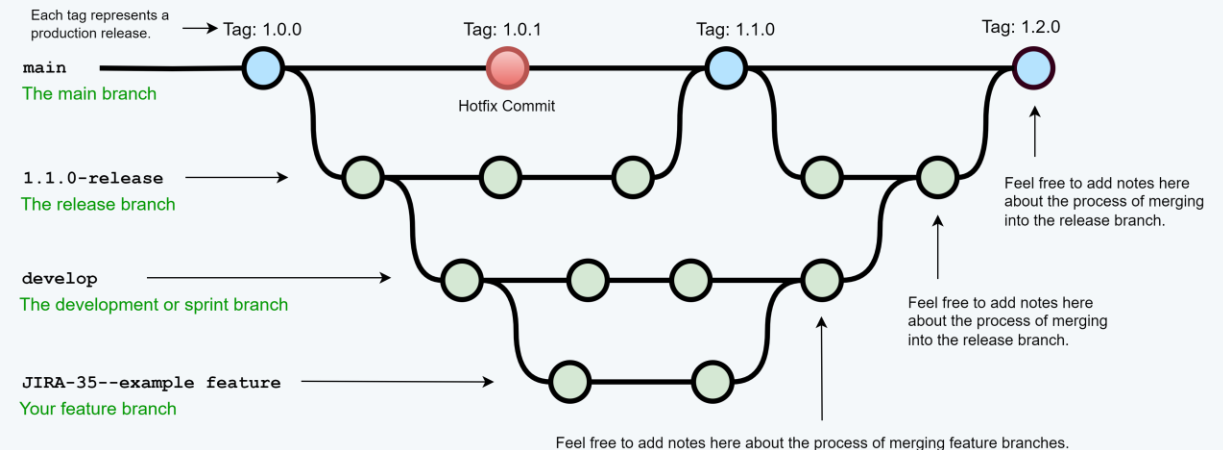
A melhor forma de definir uma “branch” é: um ambiente diferente para o mesmo projeto

Por padrão o Git define o “Branch” com o nome “main” ou “master”.

Este é o “branch” principal e neste devemos ter sempre a última versão estável do projeto

Example diagram for a workflow similar to "Git-flow" :

See: <https://nvie.com/posts/a-successful-git-branching-model/>



Branches

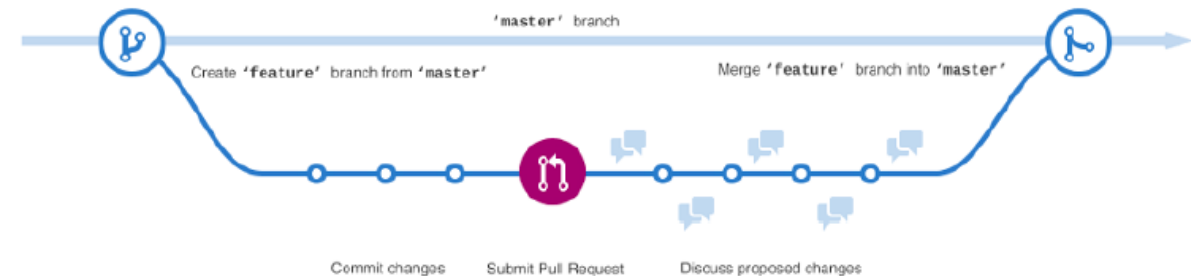
Trabalhar com “branches”

As boas práticas dizem que sempre que pretendemos fazer alterações, devemos criar uma nova “branch”.

Todas as alterações e “commit” devem ser feitas neste “branch”.

No final realizar um pedido para incorporar estas alterações no “branch master”.

Os processos de revisão de projeto são realizados neste momento de integração de versões na “branch master”.

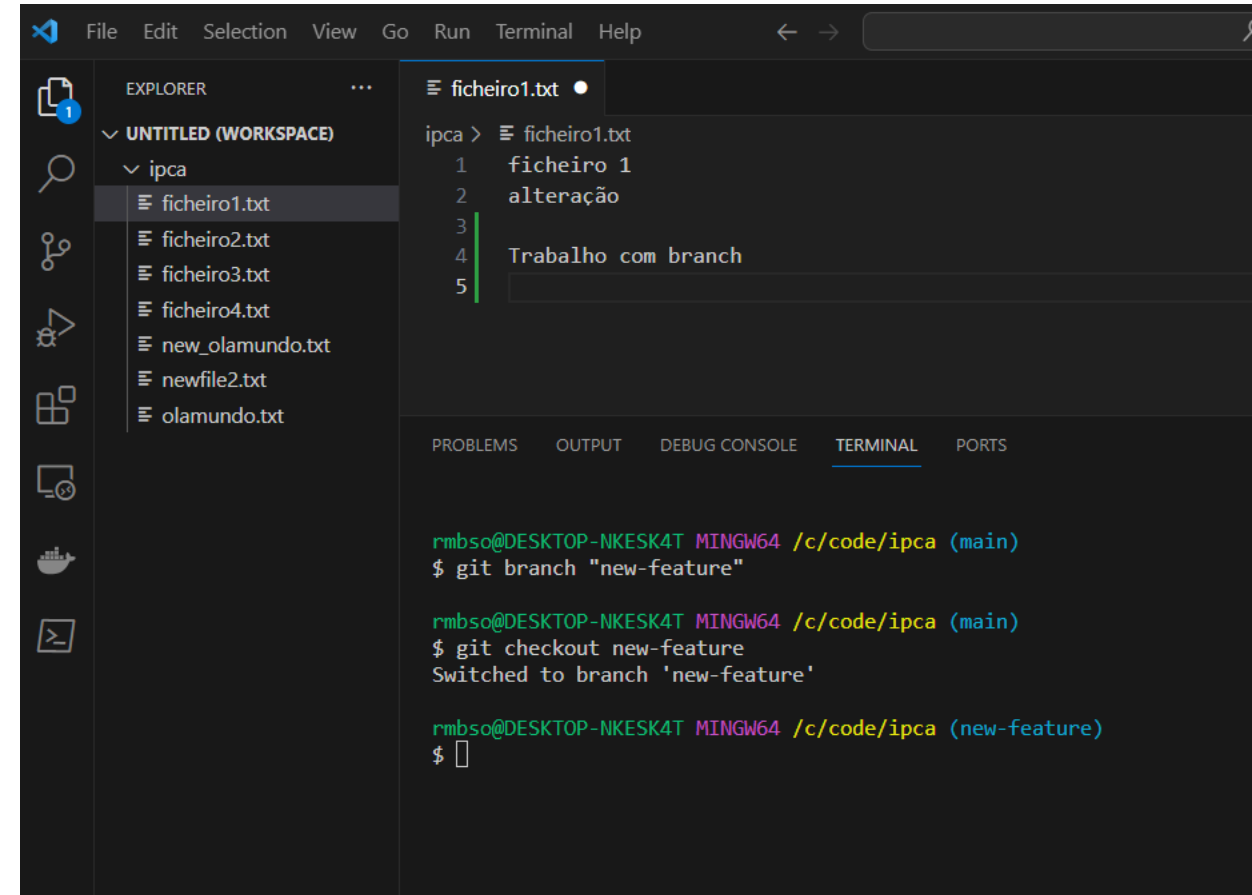


Branches

Criar um “branch” ou ramo

- No Ficheiro1.txt vamos adicionar uma linha com a seguinte descrição: “trabalho com branch”
- Não vamos fazer commit, pois este seria na “branch master”
- Vamos criar um “branch” novo
Comando: `git branch “nome_do_branch”`
- Mudar de “branch”:
Comando: `git checkout “nome_do_branch`

alternativa:
- Criar o “branch” e fazer “checkout”
Comando: `git checkout -b “nome_do_branch”`



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a workspace named 'ipca' containing several text files: 'ficheiro1.txt', 'ficheiro2.txt', 'ficheiro3.txt', 'ficheiro4.txt', 'new_olamundo.txt', 'newfile2.txt', and 'olamundo.txt'. The file 'ficheiro1.txt' is open in the editor, showing the following content:

```
ipca > ficheiro1.txt
1  ficheiro 1
2  alteração
3
4  Trabalho com branch
5
```

The Terminal panel at the bottom shows the following commands and output:

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (main)
$ git branch "new-feature"

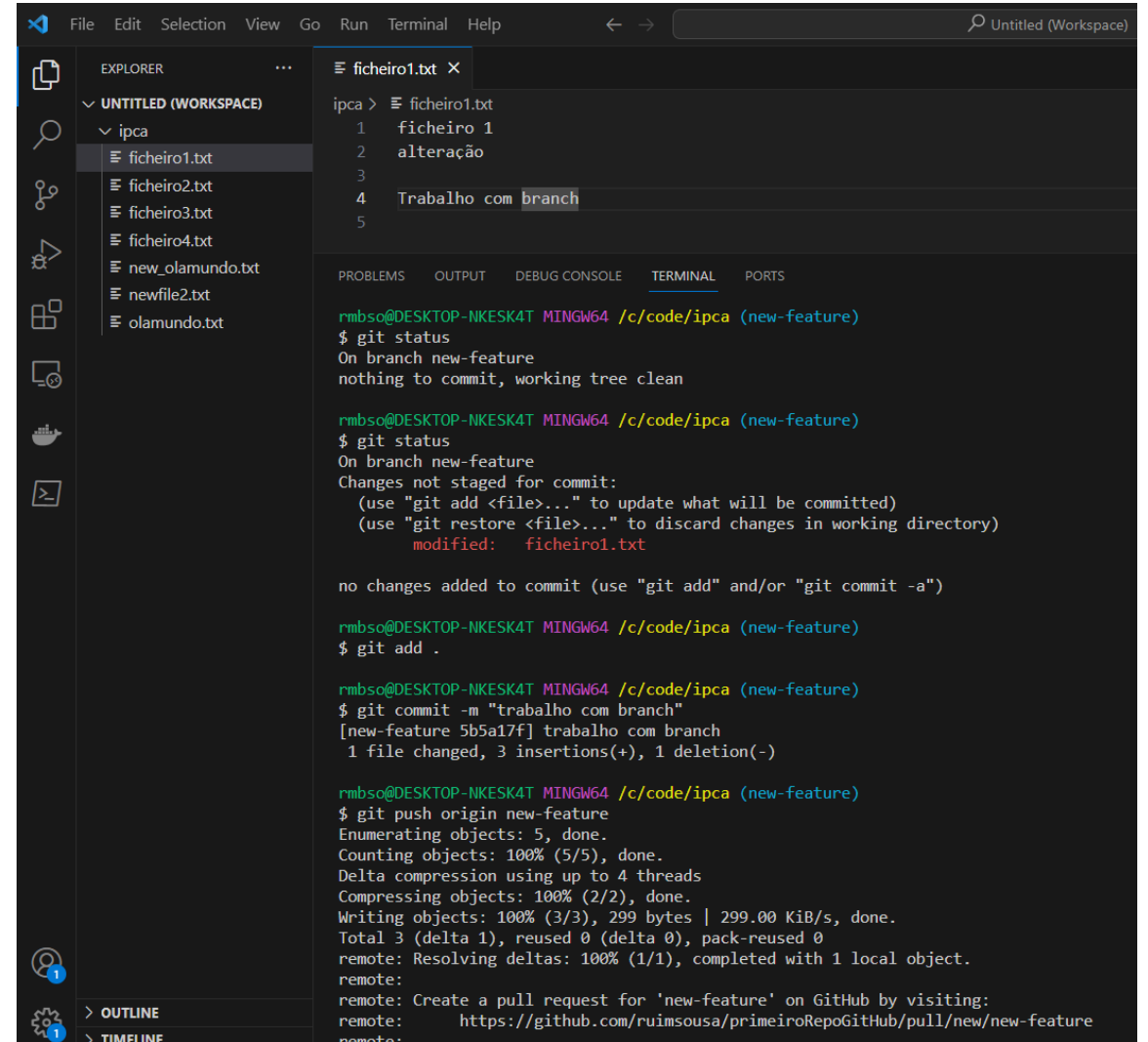
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (main)
$ git checkout new-feature
Switched to branch 'new-feature'

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (new-feature)
$
```


Branches

Criar um “branch” ou ramo

- Fazer o “commit” do ficheiro
Comandos:
- `git status`
- `git add .`
- `git commit -m “trabalho com branch”`
- `git push origin “nome_do_branch”`



The screenshot shows the Visual Studio Code interface. The Explorer view on the left shows a workspace named 'UNTITLED (WORKSPACE)' with a folder 'ipca' containing files: 'ficheiro1.txt', 'ficheiro2.txt', 'ficheiro3.txt', 'ficheiro4.txt', 'new_olamundo.txt', 'newfile2.txt', and 'olamundo.txt'. The file 'ficheiro1.txt' is open in the editor, showing the following content:

```
ipca > # ficheiro1.txt
1  ficheiro 1
2  alteração
3
4  Trabalho com branch
5
```

The TERMINAL view at the bottom shows the following commands and output:

```
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (new-feature)
$ git status
On branch new-feature
nothing to commit, working tree clean

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (new-feature)
$ git status
On branch new-feature
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:   ficheiro1.txt

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

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (new-feature)
$ git add .

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (new-feature)
$ git commit -m "trabalho com branch"
[new-feature 5b5a17f] trabalho com branch
1 file changed, 3 insertions(+), 1 deletion(-)

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (new-feature)
$ git push origin new-feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 299.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'new-feature' on GitHub by visiting:
remote:   https://github.com/ruimsousa/primeiroRepo/pull/new/new-feature
remote:
```

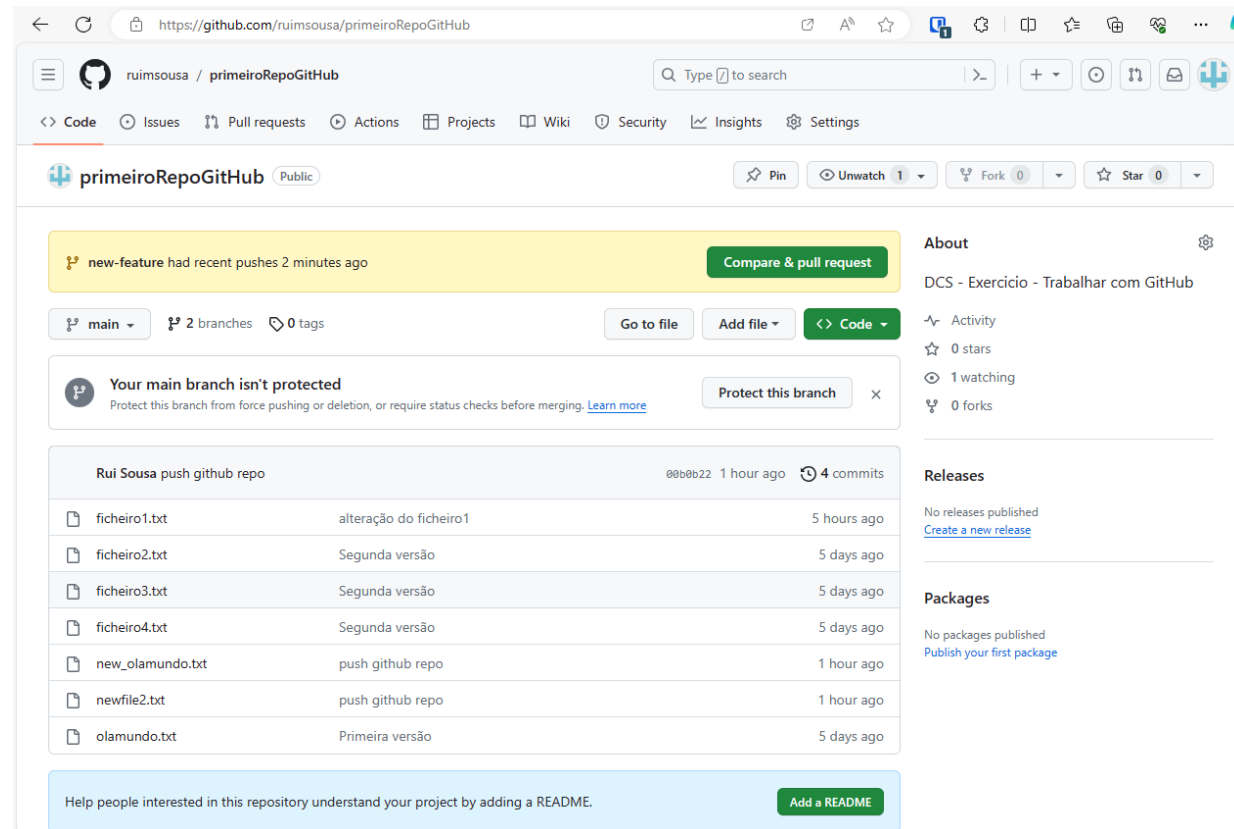
Branches

Criar um “branch” ou ramo

Neste momento temos 2 “branches” na nossa conta de GitHub

Podemos fazer um pedido de “pull-request”

Pull-request é o processo de unificar o projeto entre os vários desenvolvimentos.



Branches

Criar um “branch” ou ramo

O GitHub irá verificar as duas “branch” e verificar se existe algum conflito.

Caso não exista conflito será possível unificar o projeto.

Em desenvolvimento, inicia-se a fase de revisão de código.

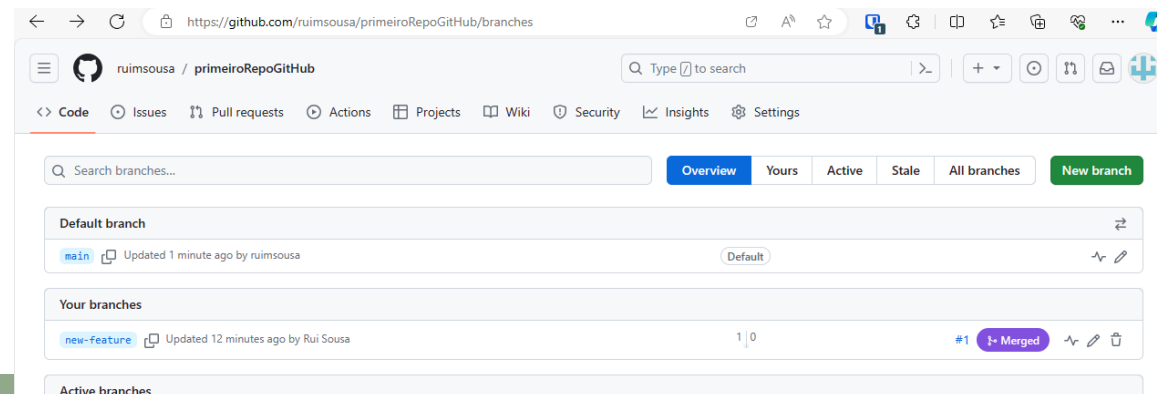
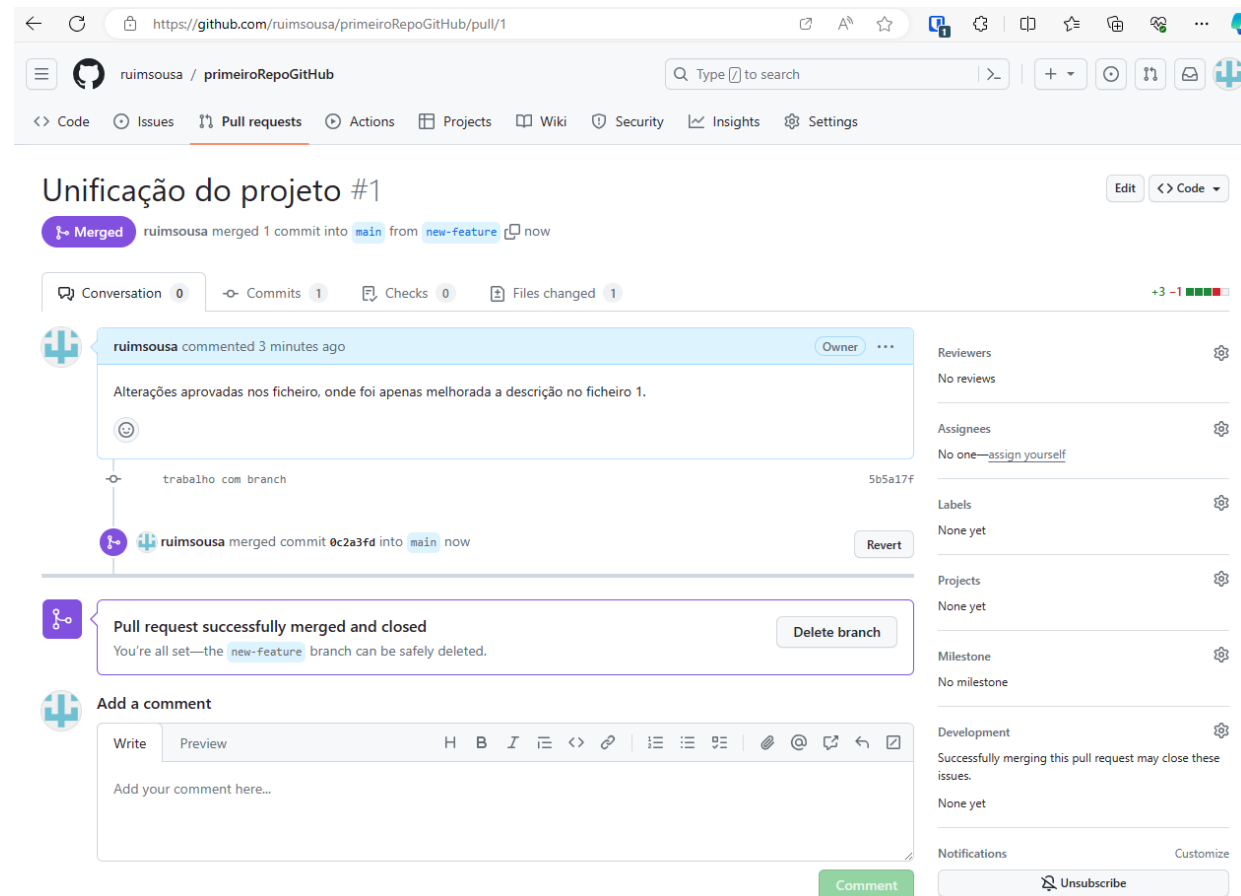
The screenshot shows the GitHub 'Open a pull request' page for a repository named 'primeiroRepoGitHub'. The page is titled 'Open a pull request' and includes a subtitle: 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).' Below the title, there is a comparison bar showing 'base: main' and 'compare: new-feature', with a green checkmark indicating 'Able to merge. These branches can be automatically merged.' The main form has two sections: 'Add a title' with a text input field containing 'trabalho com branch', and 'Add a description' with a text area containing 'Add your description here...'. To the right of the form, there are sections for 'Reviewers' (No reviews), 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), and 'Milestone' (No milestone). At the bottom right, there is a 'Development' section with a note about using closing keywords. A green 'Create pull request' button is located at the bottom right of the form. Below the form, there is a summary bar showing '1 commit', '1 file changed', and '1 contributor'. The 'Commits on Nov 2, 2023' section shows a commit titled 'trabalho com branch' by Rui Sousa, committed 6 minutes ago. The 'Showing 1 changed file with 3 additions and 1 deletion' section shows a diff for 'ficheiro1.txt' with a table of changes:

File	Line	Change
ficheiro1.txt	1	ficheiro 1
ficheiro1.txt	2	+ alteração
ficheiro1.txt	3	+
ficheiro1.txt	4	+ Trabalho com branch

Branches

Criar um “branch” ou ramo

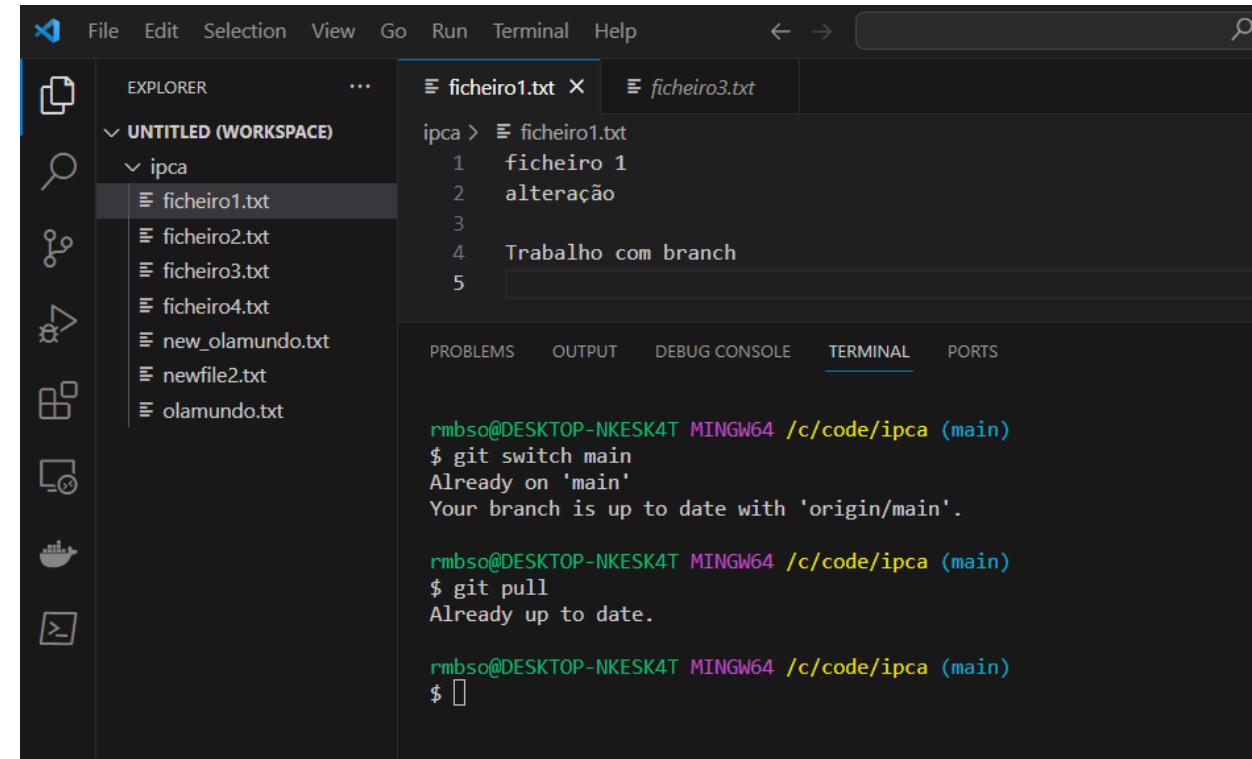
Se a unificação for efetuada com sucesso é possível remover a “branch”.



Branches

Criar um “branch” ou ramo

- Alterar para o branch main
comando: `git switch main`
- Sincronizar as alterações para o Git local
Comando: `git pull`
- Recomenda-se ter todos os “branch” atualizados



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  UNTITLED (WORKSPACE)
    ipca
      ficheiro1.txt
      ficheiro2.txt
      ficheiro3.txt
      ficheiro4.txt
      new_olamundo.txt
      newfile2.txt
      olamundo.txt
  ficheiro1.txt x  ficheiro3.txt
ipca > ficheiro1.txt
1  ficheiro 1
2  alteração
3
4  Trabalho com branch
5
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (main)
$ git switch main
Already on 'main'
Your branch is up to date with 'origin/main'.

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (main)
$ git pull
Already up to date.

rmbso@DESKTOP-NKESK4T MINGW64 /c/code/ipca (main)
$
```

Branches

Criar um “branch” ou ramo

- Unificar “branches” diferentes
- Comando: `git merge A B`

Branches

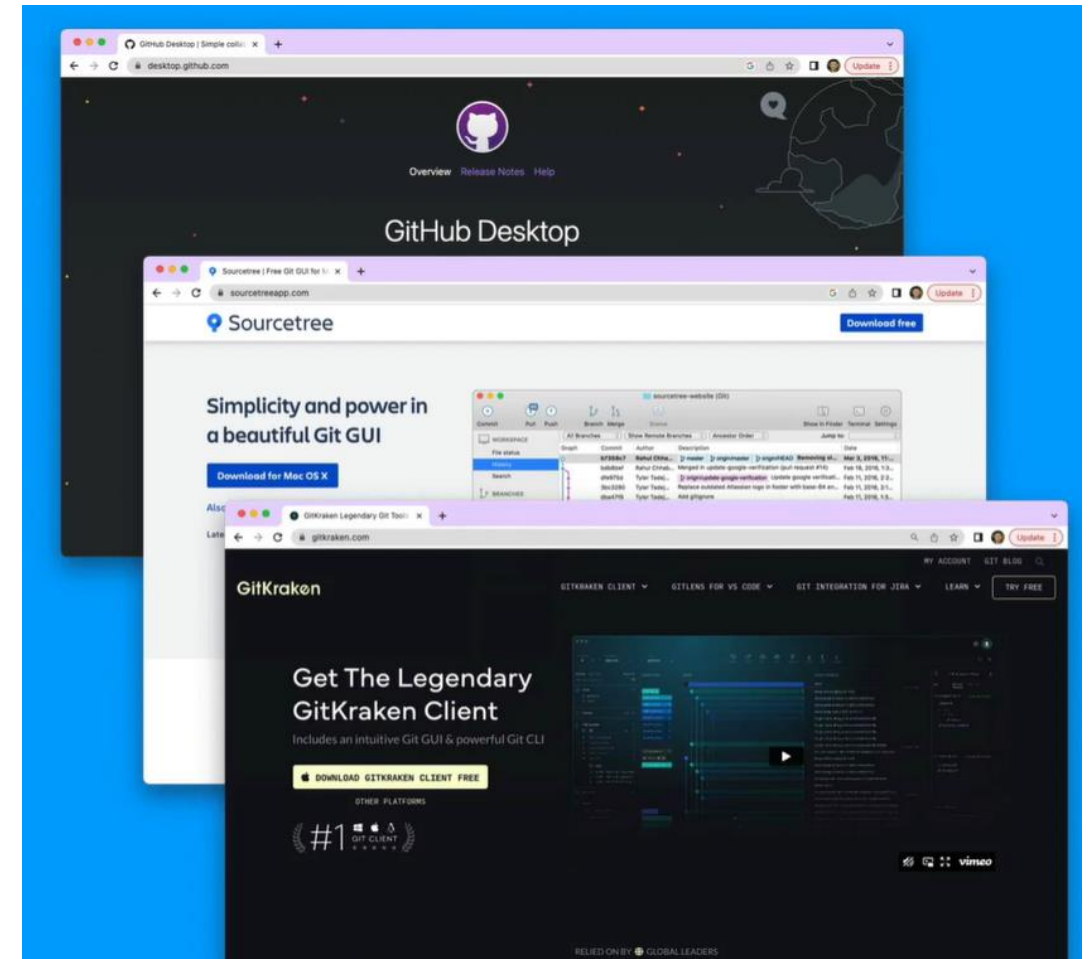
Exercício prático

- Criar um novo branch com o nome “page1” e adicionar um novo ficheiro page1.html, adicionando conteúdo
- Fazer commit deste novo ficheiro
- Alterar o ficheiro index.html e adicionar um link para o page1.html
- Fazer commit da alteração ao ficheiro index.html
- Criar um novo branch com o nome “page2” e adicionar um novo ficheiro page2.html
- Fazer commit deste novo ficheiro
- Fazer merge do page2 para o page1
- Abrir o PR e fazer o merge do branch page1 para o master
- Fazer pull do master

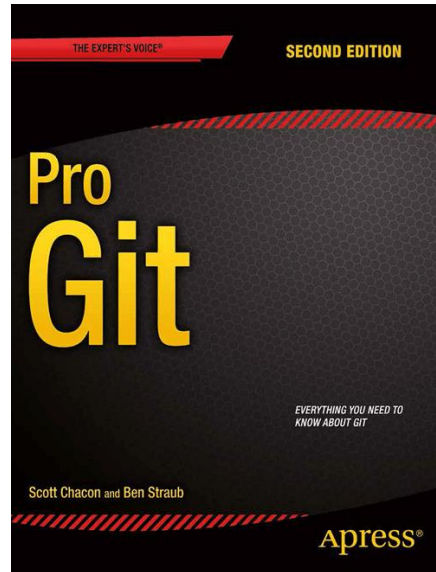
Git em Ambiente Gráfico

Ferramentas Git

Programa	Windows	MacOSx	Linux
GitKraken	X	X	X
GitHub Desktop	X	X	
SourceTree	X	X	
Fork	X	X	



Bibliografia



Pro Git book, written by Scott Chacon and Ben Straub and published by Apress.

- <https://git-scm.com/book/en/v2>
- <https://github.com/progit/progit2/releases/download/2.1.411/progit.pdf>

Outras informações

- Lista de comandos Git: <https://gist.github.com/leocomelli/2545add34e4fec21ec16>
- Aprender comandos de Git: <https://learngitbranching.js.org>