

Tutorial sobre Git e GitHub

Índice de Figuras	3
1. Git.....	4
Instalação.....	4
Configuração	4
Comandos Básicos do GIT	5
git config.....	5
git init	5
git add.....	5
git clone.....	5
git commit	5
git status.....	6
git push	6
git checkout.....	6
git remote.....	6
git branch	6
git pull.....	7
git merge	7
git diff	7
git tag.....	7
git log.....	8
git reset	8
git rm	8
git show	8
git fetch	8
git ls-tree	8
git cat-file.....	9
git grep	9
gitk.....	9
git instaweb	9
git gc	9
git archive	9
git prune	10

git fsck	10
git rebase.....	10
2. GitHub	11
Registar.....	11
Repositório	11
Adicionar	11
Para criar	13
Para fazer download	13

Índice de Figuras

Figura 1 - Instalação GIT	4
Figura 2 - Registo no GitHub	11
Figura 3 - GitHub - Área do Utilizador	12
Figura 4 - GitHub - Criar Repositório	12

1. Git

O **Git** é um sistema de controlo de versões gratuito e de código aberto. Um projeto Web com suporte Git é composto por três fases, nas quais o programador usa um conjunto de comandos Git para:

- Modificar os ficheiros no diretório de trabalho;
- Adicionar *snapshots* dos ficheiros (comando `add`);
- Registar esses *snapshots* no repositório local Git (comando `commit`), com a particularidade de adicionar uma mensagem que descreve a alteração efetuadas, informações do autor e a data.

Também podemos usar um repositório remoto de forma a garantir um backup mais sólido.

Instalação

Para instalar é necessário aceder a <https://git-scm.com/downloads> e escolher a versão a fazer download tendo em conta o seu sistema operativo (ver a Figura 1 - Instalação GIT).



Figura 1 - Instalação GIT

Configuração

Após a instalação, podemos definir o utilizador a ser usado através dos comandos seguintes a serem executados na CLI (linha de comandos) [*preferencialmente em modo de administrador*]:

```
git config --global user.name "username"  
git config --global user.email "aluno@email.pt"
```

Comandos Básicos do GIT

git config

Um dos comandos mais usados que pode ser usado para definir valores de configuração específicos do utilizador como e-mail, nome do utilizador, formato de ficheiro, etc. Por exemplo, o seguinte comando pode ser usado para definir o email:

```
git config --global <email>@xpto.pt
```

git init

Este comando é usado para iniciar um repositório novo. Por exemplo:

```
git init
```

git add

Este comando pode ser usado para adicionar ficheiros ao índice. Por exemplo, o seguinte comando irá adicionar um ficheiro chamado 'temp.txt' no diretório local para o índice de sincronização:

```
git add temp.txt
```

git clone

Este comando é usado para verificar um repositório. Se o repositório estiver num servidor remoto, usamos:

```
git clone <user>@<ip_do_servidor>:/path/to/repository
```

Por outro lado, se uma cópia de trabalho de um repositório local for criada, usamos:

```
git clone /path/to/repository
```

git commit

Este comando é usado para confirmar (sincronizar) as alterações no *HEAD* do repositório. Preste atenção que quaisquer alterações efetuadas não irão automaticamente para o repositório remoto. Por exemplo:

```
git commit -m "coloque sua mensagem aqui"
```

git status

Este comando exibe a lista de ficheiros alterados juntamente com os ficheiros que ainda não foram adicionados ou confirmados. Por exemplo:

```
git status
```

git push

É um dos comandos mais usados. Envia as alterações feitas para o ramo *MAIN* do repositório remoto associado ao diretório de trabalho (repositório local). Por exemplo:

```
git push origin main
```

git checkout

Este comando pode ser usado para criar ramos (*branch*) ou alternar entre eles. Por exemplo, para criar um ramo e automaticamente mudar para ele, executamos o comando:

```
command git checkout -b <branch-name>
```

Para simplesmente mudar de um ramo para outro, usamos:

```
git checkout <branch-name>
```

git remote

Este comando permite que um utilizador se conecte a um repositório remoto. O comando a seguir lista os repositórios remotos atualmente configurados:

```
git remote -v
```

Para conectar a um servidor remoto:

```
git remote add origin <000.000.0.00>
```

git branch

Este comando pode ser usado para listar, criar ou excluir ramos (*branch*). Para listar todos os ramos presentes no repositório, usamos:

```
git branch
```

Para excluir um ramo:

```
git branch -d <branch-name>
```

git pull

Para sincronizar todas as alterações do repositório remoto para o diretório de trabalho local. Por exemplo:

```
git pull
```

Nota: em alguns casos, após um *git pull* pode ser necessário resolver conflitos, executando um *git merge* ou *git rebase*.

git merge

Este comando é usado para “unificar” (*merge*) um ramo (*branch*) no repositório ativo. Por exemplo:

```
git merge <branch-name>
```

git diff

Este comando é usado para listar “conflitos”. Para visualizar os conflitos, tendo por base um ficheiro, usamos:

```
git diff --base <filename>
```

Para exibir os conflitos entre dois ramos (*branch*):

```
git diff <source-branch> <target-branch>
```

Para listar todos os conflitos atuais, usamos:

```
git diff
```

git tag

Este comando é usado para a definição de uma “categoria” (*tag*) específica no repositório. Por exemplo:

```
git tag 1.1.0 <insert-commitID-here>
```

git log

Exibe uma lista (log) de execuções de uma árvore (*tree*) do repositório, juntamente com os detalhes pertinentes. Um exemplo de saída pode ser:

```
commit 2342347823gb48723b4238b4237848723b47y23784y
Author: Xpto <xpto@xpto.pt>
Date:   Tue Jan 28 11:34:32 2019 -0600
```

git reset

Permite redefinir o índice e o repositório para o último *commit*. Por exemplo:

```
git reset --hard main
```

git rm

Este comando é usado para remover ficheiro(s) do repositório. Por exemplo:

```
git rm filename.txt
```

git show

Permite visualizar informações de um qualquer “objeto”. Por exemplo:

```
git show
```

git fetch

Permite obter todos os “objetos” do repositório remoto que atualmente não se encontram armazenados no diretório do repositório local. Por exemplo:

```
git fetch origin
```

git ls-tree

Este comando exibe os “objetos” de uma árvore (*tree*) do repositório juntamente com o nome, item e o valor SHA-1 do blob. Por exemplo:

```
git ls-tree master
```


git cat-file

Com recurso a um valor encriptado com SHA-1, exhibe o tipo de um “objeto”. Por exemplo:

```
git cat-file -p d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

git grep

Permite que um utilizador procure um conteúdo (frase e/ou palavra(s)) no repositório. Por exemplo, para pesquisar ‘xpto’ em todos os ficheiros usa-se:

```
git grep "xpto"
```

gitk

A interface gráfica de um repositório local pode ser invocado digitando e executando o comando:

```
gitk
```

git instaweb

Com este comando um servidor web pode ser executado em ligação com o repositório local. Um navegador (*browser*) é automaticamente direcionado para ele. Por exemplo:

```
git instaweb --httpd=apache2
```

git gc

Otimiza o repositório através da “recolha de lixo”, que irá limpar os ficheiros desnecessários e otimizá-los. Por exemplo:

```
git gc
```

git archive

Permite ao utilizador criar um ficheiro ‘zip’ ou ‘tar’ contendo os componentes de uma única árvore do repositório. Por exemplo:

```
git archive --format=tar master
```

git prune

Com a execução deste comando, os “objetos” que não têm referências de entrada são excluídos. Por exemplo:

```
git prune
```

git fsck

Executa uma verificação de integridade do sistema de ficheiros *git*. Todos os “objetos” corrompidos são identificados:

```
git fsck
```

git rebase

É usado para reaplicar os compromissos num outro ramo (*branch*). Por exemplo:

```
git rebase main
```

2. GitHub

Existem vários sites que proveem hospedagem gratuita de código-fonte para repositórios **Git**, o **GitHub** é um deles. O GitHub é uma plataforma que permite aos programadores alojarem os seus projetos de forma que seguidores (outros utilizadores que acompanham o projeto) possam acompanhar e participar (contribuição) nos projetos.

Registar

Para criar conta é necessário aceder a <https://github.com/> e efetuar o “**Sign up**” (ver a Figura 2 - Registo no GitHub).

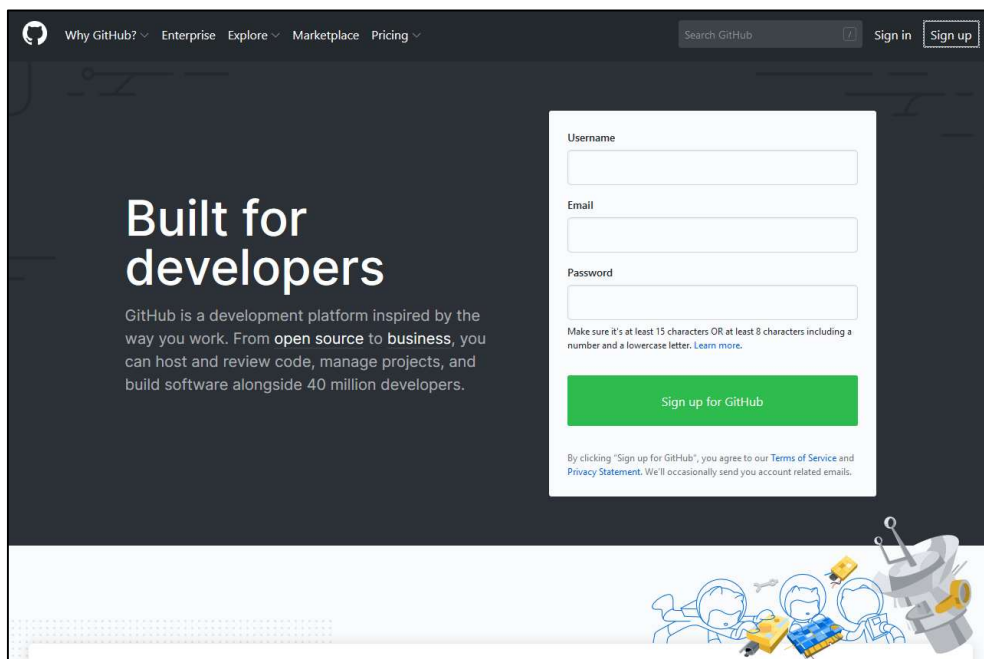


Figura 2 - Registo no GitHub

Repositório

Um repositório é o local onde se aloja o projeto. Os repositórios podem ser **públicos** ou **privados**.

Adicionar

Uma das formas de criar é clicar em “New” ou em “Start a project” (ver a Figura 3 - GitHub - Área do Utilizador).

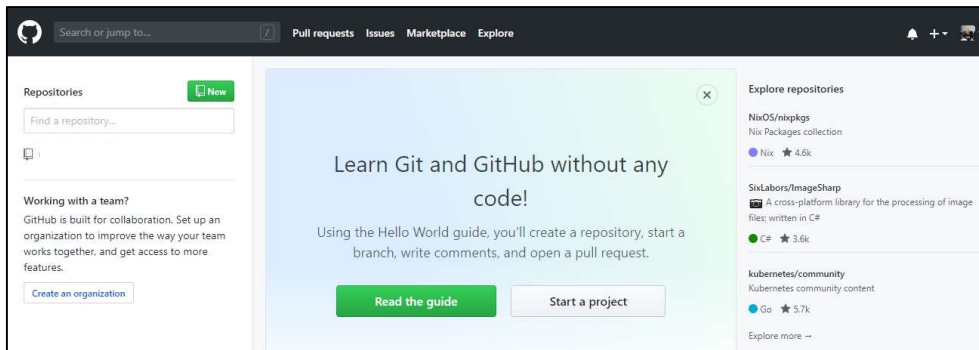


Figura 3 - GitHub - Área do Utilizador

Ao registar um repositório é necessário atribuir um nome, uma breve descrição (opcional) e seleccionar se é Público ou Privado (ver a Figura 4 - GitHub - Criar Repositório).

Figura 4 - GitHub - Criar Repositório

Neste momento também podemos inicializar o repositório ao seleccionar a *checkbox* em “**Initialize this repository with a README**”, ficando logo disponível para efetuar o clone.

Se não seleccionarmos, o repositório fica à espera que seja inicializado apresentando duas formas de o fazer.

Para criar ...

```
echo "# <nome_repositorio>" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin
https://github.com/<user>/<nome_repositorio>.git
git push -u origin main
```

Para fazer download ...

```
git remote add origin
https://github.com/<user>/<nome_repositorio>.git
git push -u origin main
```

Nota: em alternativa, no VSC é possível carregar em CTRL + SHIFT + P e escolher a opção ">Git: Clone" > "Clone from GitHub" e escolher o repositório que queremos fazer download para o nosso *workspace* local. É necessário que anteriormente já se tenha executado os comandos do Git que permitem configurar o *username* e o email da conta do GitHub.