



git

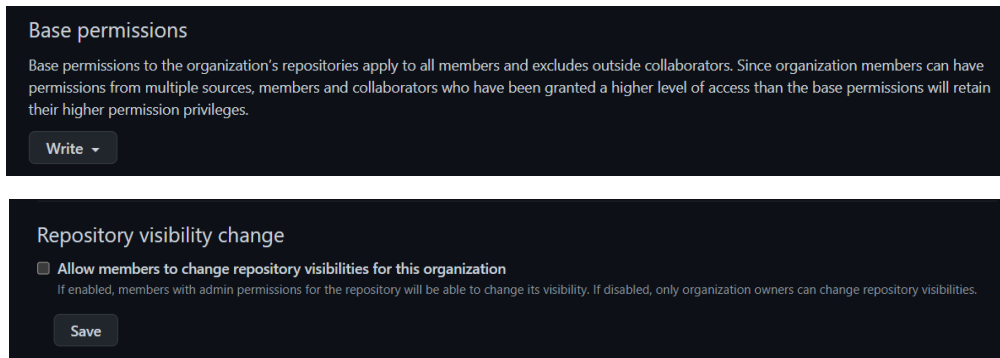
PROJETO FINAL DAS
Criação de um repositório com GitFlow

Pedro Monteiro
Pedro.rocha.2022014@my.istec.pt

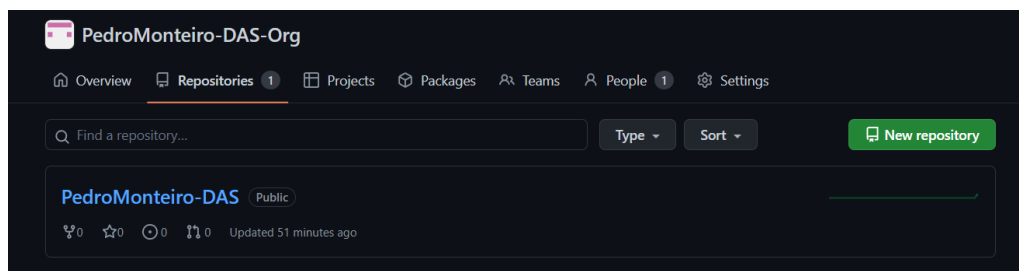
Criação do repositório (e organização)

O primeiro passo para o desenvolvimento do trabalho foi a criação de uma organização no GitHub e de seguida procedemos à criação do repositório dentro da organização.

A organização permite a criação de diversos níveis de acesso. Para criar uma organização apenas temos de aceder ao nosso perfil, de seguida abrir a aba das organizações e aí podemos criar uma nova organização. De modo a cumprir com os requisitos do projeto a desenvolver, nas configurações da organização, e na aba “*Member privileges*”, foram utilizadas as seguintes opções:



Com a organização criada podemos criar um repositório como normal.



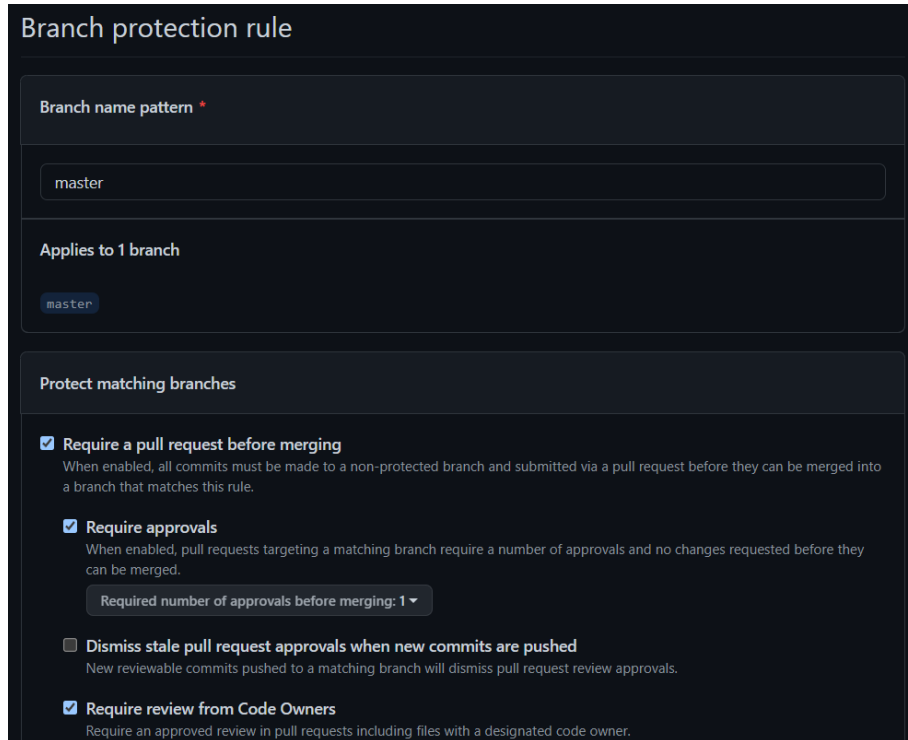
Com o repositório criado apenas temos de abrir o *Git Bash* e utilizar o seguinte comando para fazer a copia do repositório remoto para um repositório local:

```
clone [link]
```

Nota: Temos de ter em conta que o comando acima referido faz copia do repositório remoto para o diretório onde estamos, logo devemos utilizar o comando dentro de um local fácil acesso, como por exemplo ambiente de trabalho.

Depois disto podemos fazer o nosso primeiro *commit*, seguindo as instruções do *Git Hub*.

Finalmente podemos criar as *branch protection* rules. Para isso devemos aceder ao nosso repositório, de seguida as configurações e na aba “*Branches*” adicionamos uma nova regra.



The screenshot shows the 'Branch protection rule' configuration page in GitHub. It includes a 'Branch name pattern' field set to 'master', a list of branches it applies to (also 'master'), and a 'Protect matching branches' section. In this section, four options are checked: 'Require a pull request before merging', 'Require approvals' (with a dropdown set to 1), 'Dismiss stale pull request approvals when new commits are pushed', and 'Require review from Code Owners'.

Git Flow – Iniciação

O *Git Flow* é um modelo de trabalho muito utilizado por equipas de desenvolvimento de software. Este método destaca-se por ser extremamente simples de ser utilizado e compreendido e de ser bastante flexível, adaptando-se a equipas de todas as dimensões.

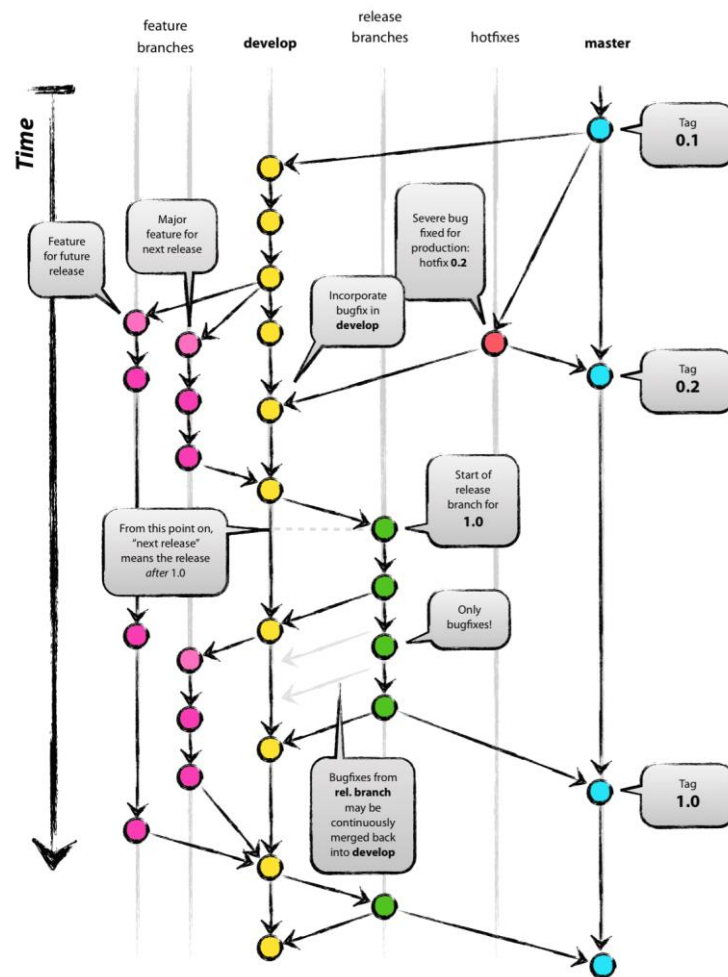
Para iniciar a utilização do *Git Flow* no nosso repositório devemos seguir as seguintes

Dentro do repositório criado, no *Git Bash*, apenas temos de executar o seguinte comando para inicializar o *Git Flow*:

```
git flow init
```

Depois de executar este comando irão aparecer diversas opções, mas devemos deixar os valores *default*.

Estrutura do *Git Flow*



Configuração do *.gitignore*

O *.gitignore* é um ficheiro que tem como função ignorar ficheiros que poderiam ser incluídos num *commit*. Neste caso o objetivo do *.gitignore* é ignorar ficheiros com o formato *.doc* e *.docx*.

Para criar e configurar o *.gitignore* devemos seguir os seguintes passos:

```
git checkout master  
nano .gitignore
```

Ao fazer o comando acima indicado irá abrir um editor de texto. Dentro do ficheiro devemos escrever:

```
*.doc  
*.docx
```

De seguida fazemos *Ctrl + O* para guardar o ficheiro e *Ctrl + X* para fechar o editor de texto.

Depois do ficheiro criado e devidamente configurado podemos executar os seguintes comandos para enviar o nosso progresso para o repositório e para o *branch develop*:

```
git status  
git add .  
git commit -m "Configuração do .gitignore"  
git push origin master  
git checkout develop  
git pull origin master
```

Para testar se o *.gitignore* está a funcionar corretamente podemos criar um ficheiro do formato *.doc* ou *.docx* e quando o comando `git status` for executado não irá aparecer nenhuma alteração.

Carregamento inicial

Para o carregamento inicial foi criado um PDF vazio. De seguida foram feitas as seguintes operações:

```
git checkout develop  
git status  
git add .  
git commit -m "Carregamento inicial"  
git push origin develop
```

Utilizar *features*

De forma bastante simplificada, quando queremos fazer novos desenvolvimentos dentro do repositório devemos organiza-los em forma de *features*. Depois do *Git Flow* inicializado vão ser criadas secções para *branches*. As *features* devem ser sempre criadas com base no *master*. Para iniciar uma nova *feature* corretamente devemos utilizar os seguintes comandos:

```
git checkout master  
git flow feature start [feature_name]
```

Depois disso irá ser criada uma nova *branch* com o nome que inserimos e de seguida podemos fazer as alterações necessárias dentro dessa *branch*.

Depois de todas as alterações feitas devemos acabar a *feature*. Para isso devemos executar a seguinte sequencia de comandos:

```
git status  
git add .  
git commit -m "[descrição]"  
git flow feature finish [feature_name]  
git push origin develop
```

Se tudo correr bem a *branch feature* irá ser apagada, iremos voltar ao *branch master* e um novo *commit* irá aparecer no *Git Hub* para ser aprovado.

Utilizar *releases*

De todas as *branches* no modelo *Git Flow*, a *branch release* é a que tem o tempo de vida mais curto. Esta *branch* é normalmente utilizada quando os testes de todas as *features* estão completos e se pretende fazer o *merge* para a *branch master*.

Para utilizar releases devemos seguir os seguintes passos:

```
git checkout develop  
git flow release start [release_name]  
git flow release finish [release_name]
```

Ao executar os comando acima citados irão ser apresentados alguns “erros” dizendo se não queremos adicionar alguma *tag*, mostrando um editor de texto. Neste editor de texto podemos escrever a *tag* que desejamos que seja adicionada ao *branch master*, caso não exista nenhuma *tag* podemos apenas escrever :qa! e pressionar *enter*.

Utilizar hotfix

Um *hotfix* é utilizado quando queremos fazer alguma alteração num ambiente de produção. O *hotfix* deve ser sempre feito a partir do *branch master* ou do *branch develop*. Normalmente estes *branches* são utilizados para corrigir pequenos erros e problemas existentes no repositório.

Para utilizar um *hotfix* devemos seguir os seguintes passos:

```
git flow hotfix start [hotfix_name] [base_branch]
git flow hotfix [hotfix_name] -m [tag]
```

Depois de executar os comandos acima mencionados irá ser feito um *merge* da *branch hotfix* para a *branch* base utilizado. Caso seja adicionada alguma *tag* podemos verificar as *tags* do *branch* atual fazendo o comando `git tag -l`. Caso não seja adicionada nenhuma *tag* irá abrir um novo editor de texto onde devemos escrever `:qa!` para sair.

Comandos extra

`ls -a` → Mostra ficheiros ocultos dentro de um repositório;

`nano [file_name]` → Cria/edita um ficheiro;

`cat [file_name]` → Mostra o conteúdo de um ficheiro;

`touch [file_name]` → Cria um ficheiro vazio;

`[command] --help` → Mostra todas as funções de um comando;

`git add .` → Adiciona todos os ficheiros alterados desde o último *commit*;

`clear` → Limpa o ecrã;

`Ctrl + C` → Trava a execução de um comando;