



# Relatório Trabalho Final DAS

Leonardo Andrade DS 2022097

## Objetivo

Neste projeto pretende-se que crie um repositório no GitHub, usando em primeira instância o command line do Git.

O repositório terá de ter as seguintes regras:

1. Deve ter criados todos os branches necessários para a utilização do modelo GitFlow no que toca a gestão de branches;
2. Ter pelo menos dois níveis de acesso ao GitHub, permitindo a developers a submissão de código, mas nunca a alteração de visibilidade do repositório (esta permissão deve ser dada apenas a proprietários do repositório);
3. Tornar obrigatória a revisão de código antes de um pull request ser aprovado;
4. Adicionar ficheiro .gitignore de forma a ignorar ficheiros, .docx e .doc;

5. Controlar as versões do relatório a entregar usando o repositório, sendo que é

necessário que existam pelo menos:

- a. 1 carregamento inicial do ficheiro no branch develop;
- b. 5 alterações no próprio branch develop com os devidos comentários;
- c. 1 release que contribua para o branch master;
- d. 1 hotfix efetuado utilizando a gestão de branches GitFlow;

6. A versão do relatório final deve estar presente no repositório, no branch master.

-----

1. Deve ter criados todos os branches necessários para a utilização do modelo GitFlow no que toca a gestão de branches;

```
efma7@Emanuel MINGW64 ~ (master)
$ cd TrabalhoFinalDAS

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (main)
$ Git flow init

No branches exist yet. Base branches must be created now.
Branch name for production releases: [master] Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/efma7/TrabalhoFinalDAS/.git/hooks]

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (develop)
$
```

2- Ter pelo menos dois níveis de acesso ao GitHub, permitindo a developers a submissão de código, mas nunca a alteração de visibilidade do repositório

## Member privileges

---

### Base permissions

Base permissions to the organization's repositories apply to all members and excludes outside collaborators. Since organization members can have permissions from multiple sources, members and collaborators who have been granted a higher level of access than the base permissions will retain their higher permission privileges.

Write -

### Repository creation

Members will be able to create only selected repository types. Outside collaborators can never create repositories.

☐ Public

Members will be able to create public repositories, visible to anyone. [Why is this option disabled?](#)

☒ Private

Members will be able to create private repositories, visible to organization members with permission.

Save

### Repository forking

☐ Allow forking of private repositories

If enabled, forking is allowed on private and public repositories. If disabled, forking is only allowed on public repositories. This setting is also configurable per-repository.

Save

### Pages creation

Members will be able to publish sites with only the selected access controls.

☒ Public

Members will be able to create public sites, visible to anyone.

☐ Private

Members will be able to create private sites, visible to anyone with permission. [Why is this option disabled?](#)

Save

### Integration access requests

☒ Allow integration requests from outside collaborators

Outside collaborators will be able to request access for GitHub or OAuth apps to access this organization and its resources.

Save

## Admin repository permissions

---

### Repository visibility change

☐ Allow members to change repository visibilities for this organization

If enabled, members with admin permissions for the repository will be able to change its visibility. If disabled, only organization owners can change repository visibilities.

Save

### 3- Tornar obrigatória a revisão de código antes de um pull request ser aprovado

**Protect matching branches**

☒ **Require a pull request before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.

☒ **Require approvals**  
When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.  

Required number of approvals before merging: 1 ▼

☐ **Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

☒ **Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

☐ **Require approval of the most recent reviewable push**  
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

### 4. Adicionar ficheiro .gitignore de forma a ignorar ficheiros, .docx e .doc;

```
efma7@Emanuel MINGW64 ~ (master)
$ cd TrabalhoFinalDAS

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ nano .gitignore

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ ls -a
./ ../ .git/ .gitignore .

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ cat .gitignore
*.doc
*.docx

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ |
```

```
efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ git commit -m "configuração do gitignore"
[master 95d01e7] configuração do gitignore
1 file changed, 2 insertions(+)
create mode 100644 .gitignore

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (master)
$ git push origin master
```

5. Controlar as versões do relatório a entregar usando o repositório, sendo que é

Necessário que existam pelo menos:

- a. 1 carregamento inicial do ficheiro no branch develop;
- b. 5 alterações no próprio branch develop com os devidos comentários

```
efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (develop)
$ git flow feature start feature_relatorio1
Switched to a new branch 'feature/feature_relatorio1'

Summary of actions:
- A new branch 'feature/feature_relatorio1' was created, based on 'develop'
- You are now on branch 'feature/feature_relatorio1'

Now, start committing on your feature. When done, use:

    git flow feature finish feature_relatorio1

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (feature/feature_relatorio1)
$ git add .

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (feature/feature_relatorio1)
$ git commit -m "relatorio1"
[feature/feature_relatorio1 3e69d77] relatorio1
1 file changed, 0 insertions(+), 0 deletions(-)

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (feature/feature_relatorio1)
$ git flow feature finish feature_relatorio1
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
Updating 5a2b20d..3e69d77
Fast-forward
 Doc1.pdf | Bin 140950 -> 407443 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
Deleted branch feature/feature_relatorio1 (was 3e69d77).

Summary of actions:
- The feature branch 'feature/feature_relatorio1' was merged into 'develop'
- Feature branch 'feature/feature_relatorio1' has been locally deleted
- You are now on branch 'develop'

efma7@Emanuel MINGW64 ~/TrabalhoFinalDAS (develop)
$ git push origin develop
```