

Instituto Superior de Tecnologias Avançadas do Porto
CTESP DS - Curso Técnico Superior Profissional de
Desenvolvimento de Software

Ano letivo 2022/2023
DAS - Desenvolvimento Ágil de Software

Trabalho Prático Individual

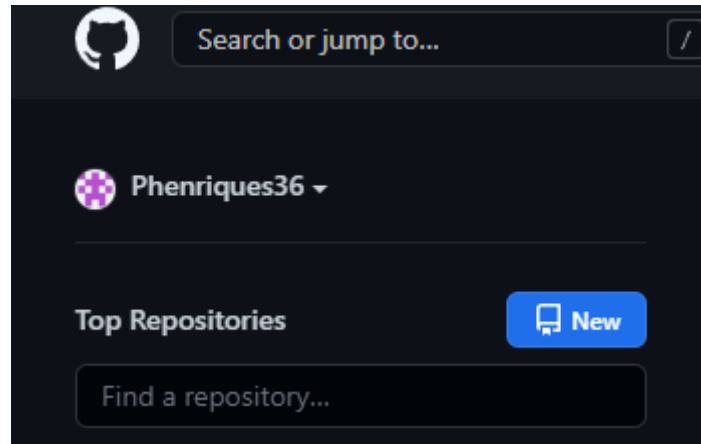
Final - DAS



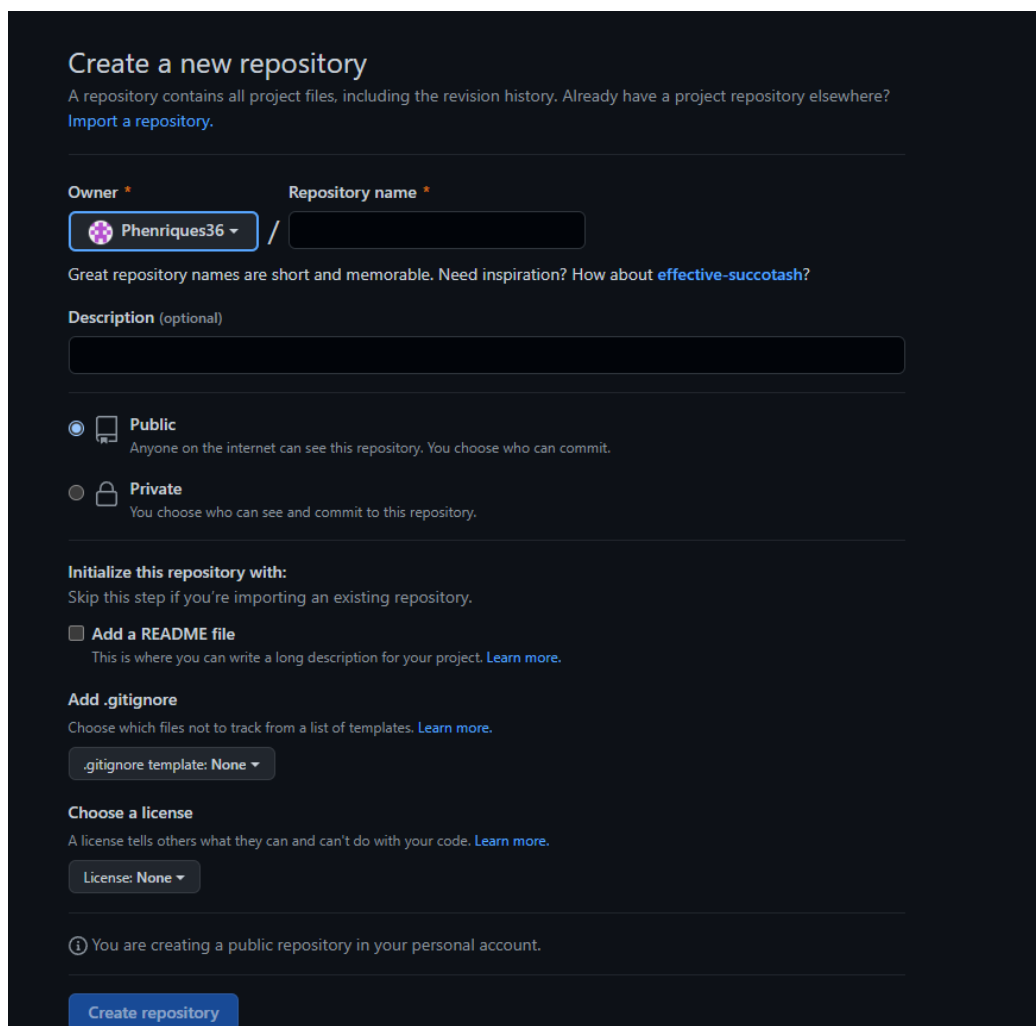
Paulo Henriques

Ponto 1

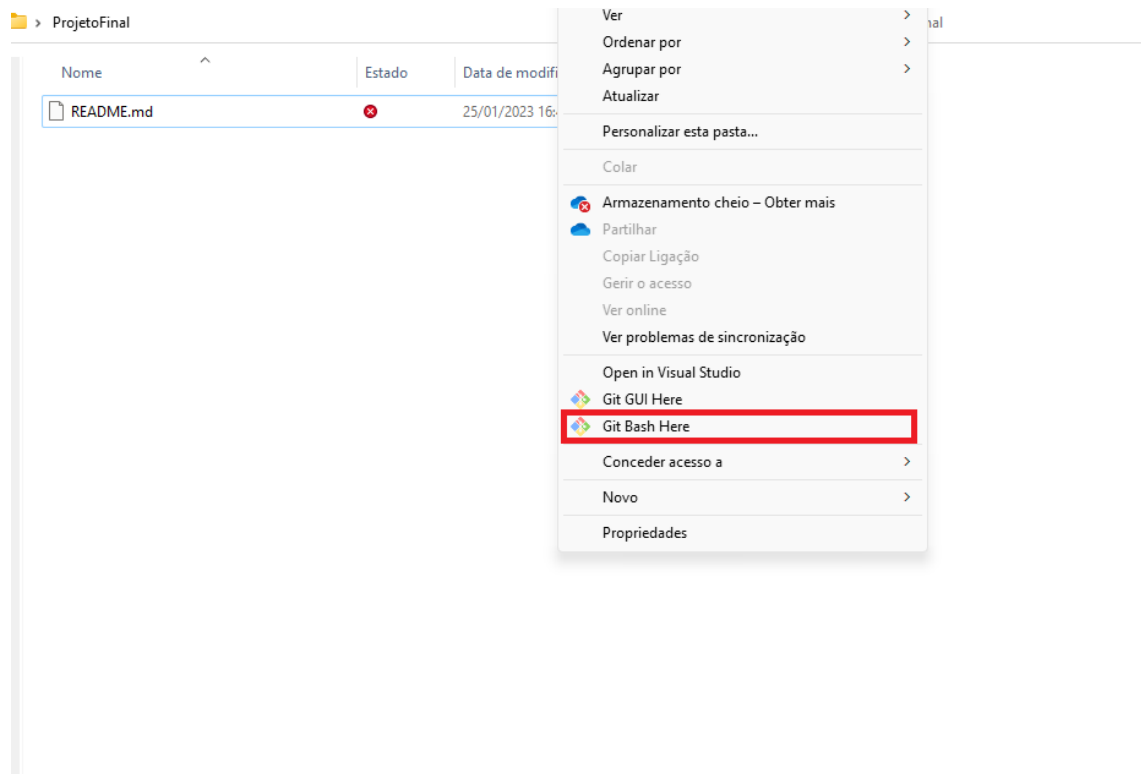
Primeiramente temos de criar um repositório no GitHub para isso depois de entrarmos com a nossa conta no GitHub devemos seleccionar a opção New



Depois vai aparecer este ecrã onde devemos preencher com as informações que queremos

A screenshot of the 'Create a new repository' form on GitHub. The form is titled 'Create a new repository' and includes a subtitle: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' section has a dropdown menu showing 'Phenriques36'. The 'Repository name' section has an empty text input field. Below these, there is a note: 'Great repository names are short and memorable. Need inspiration? How about [effective-succotash](#)?'. The 'Description' section has an optional text input field. The 'Visibility' section has two radio buttons: 'Public' (selected) and 'Private'. The 'Initialize this repository with:' section has a checkbox for 'Add a README file'. The 'Add .gitignore' section has a dropdown menu for '.gitignore template: None'. The 'Choose a license' section has a dropdown menu for 'License: None'. At the bottom, there is a blue button labeled 'Create repository' and a note: 'You are creating a public repository in your personal account.'

De seguida vamos abrir o git bash na pasta que queremos usar como repositório local da seguinte forma, neste caso vou utilizar uma pasta com o nome ProjetoFinal e seleccionar a opção marcada a vermelho



Depois do GIT Bash aberto devemos usar o seguinte código para ligar o repositório local ao GitHub

```
echo "# ProjetoFinal" >> README.md
git init
git add README.md
git commit -m "Add README"
git branch -M main
git remote add origin "https://github.com/Phenriques36/TrabalhoFinalGITDAS.git"(o
link aqui é diferente para todos, este é apenas o link do meu repositório)
git push -u origin main
git flow init
```

Ponto 2

Neste ponto é suposto definir níveis de acesso de uma organização, para isso devemos criar uma organização e definir os seguintes passos

The screenshot shows the GitHub organization overview page for 'TrabalhoFinalDAS'. The top navigation bar includes links for Overview, Repositories, Projects, Packages, Teams, People, and Settings. The main content area features a large heading 'We think you're gonna like it here.' followed by a subheading 'We've suggested some tasks here in your organization's overview to help you get started.' Below this, there are four task cards: 'Invite your first member', 'Customize members' permissions', 'Create a pull request', and 'Create a branch protection rule'. To the right, there are several informational boxes: 'You can now follow organization', 'You can create a README file visible to anyone', 'You can hide the tasks we've suggested on this page and bring them back later.', 'Discussions', 'Repositories', and 'People'.

The screenshot shows the 'Member privileges' settings page in GitHub. The page is divided into four sections: 'Base permissions', 'Repository creation', 'Repository forking', and 'Pages creation'. Each section contains a description of the privilege and a 'Save' button. The 'Base permissions' section shows a 'Write' dropdown menu. The 'Repository creation' section has checkboxes for 'Public' and 'Private'. The 'Repository forking' section has a checkbox for 'Allow forking of private repositories'. The 'Pages creation' section has checkboxes for 'Public' and 'Private'.

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

Repository deletion and transfer

☒ **Allow members to delete or transfer repositories for this organization**

If enabled, members with admin permissions for the repository will be able to delete or transfer public and private repositories. If disabled, only organization owners can delete or transfer repositories.

Save

Issue deletion

☐ **Allow repository administrators to delete issues for this organization**

If enabled, members with admin permissions for the repository will be able to delete issues. If disabled, only organization owners can delete issues. [Learn more.](#)

Save

Member team permissions

Team creation rules

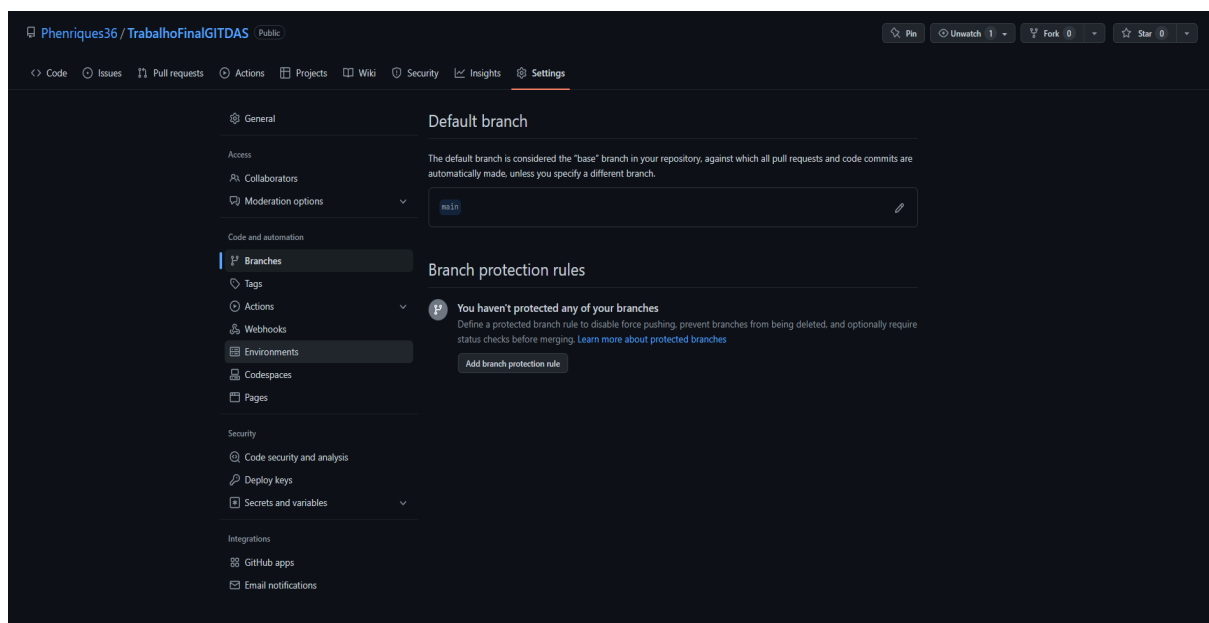
☒ **Allow members to create teams**

If enabled, any member of the organization will be able to create new teams. If disabled, only organization owners can create new teams.

Save

Depois disto feito as permissões estarão bem definidas permitindo a developers a submissão de código, mas nunca a alteração de visibilidade do repositório.

Ponto 3



Branch protection rule



Protect your most important branches

[Branch protection rules](#) define whether collaborators can delete or force push to the branch and set requirements for any pushes to the branch, such as passing status checks or a linear commit history.

Your [GitHub Free plan](#) can only enforce rules on its public repositories, like this one.

Branch name pattern *

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 status checks to pass before merging

Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.



Require conversation resolution before merging

When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more.](#)



Require signed commits

Commits pushed to matching branches must have verified signatures.



Protect your most important branches

[Branch protection rules](#) define whether collaborators can delete or force push to the branch and set requirements for any pushes to the branch, such as passing status checks or a linear commit history.

Your [GitHub Free plan](#) can only enforce rules on its public repositories, like this one.

Branch name pattern *

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.



Require status checks to pass before merging

Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.



Require conversation resolution before merging

When enabled, all conversations on code must be resolved before a pull request can be merged into a branch that matches this rule. [Learn more.](#)

Ponto 4

Para adicionar um ficheiro .gitignore basta usar o seguinte código:

```
git checkout develop
nano .gitignore
Inserir isto dentro do documento (*.docx \n *.doc) depois Ctrl O + Enter + Ctrl X
cat .gitignore
git add .
git commit -m "Adicionar .gitignore"
git push -u origin main
git checkout develop
git pull origin main
```

Ponto 5

O ponto 5 é dividido em 4 partes cada uma diferente:

- A) git flow feature start relatorioinicial
git add .
git commit -m "exemplo"
git flow feature finish relatorioinicial
git push -u origin
- B) git flow feature start mudancasrelatorio
git add .
git commit -m "exemplo"
git flow feature finish mudancasrelatorio
git push -u origin
Repetir 4x
- C) Continuacao