

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

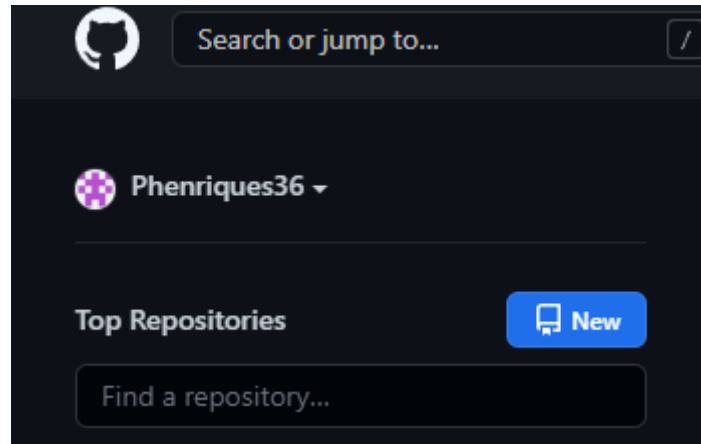
Final - DAS



Paulo Henriques

Ponto 1

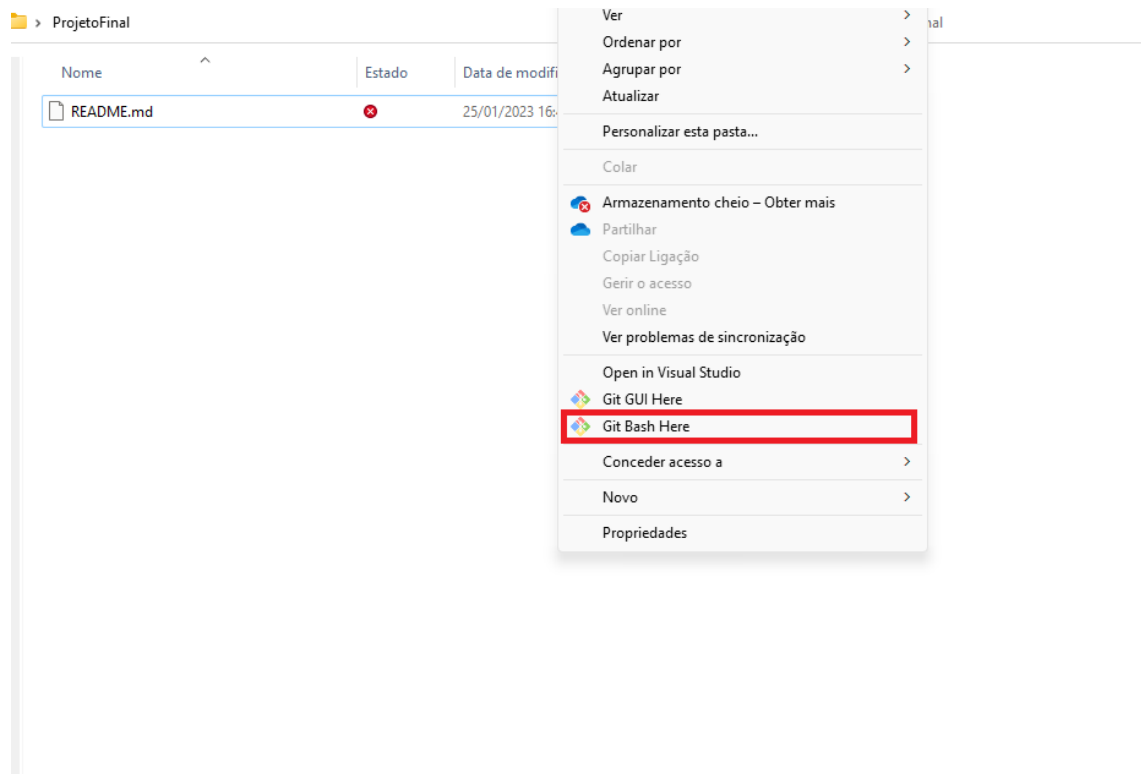
Primeiramente temos de criar um repositório no GitHub para isso depois de entrarmos com a nossa conta no GitHub devemos seleccionar a opção New



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 has a dark background. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two input fields: 'Owner' (with a dropdown menu showing 'Phenriques36') and 'Repository name'. A note suggests using short and memorable names. There's a 'Description' field with a placeholder '(optional)'. Below that, there are two radio buttons for 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.' Underneath, there's a section 'Initialize this repository with:' with a note to skip if importing an existing repository. There's a checkbox for 'Add a README file' with a note that it's a long description for the project. Below that, there's a section 'Add .gitignore' with a note to choose which files not to track from a list of templates. There's a dropdown menu for '.gitignore template' currently set to 'None'. Then, there's a section 'Choose a license' with a note that a license tells others what they can and can't do with the code. There's a dropdown menu for 'License' currently set to 'None'. At the bottom, there's an information icon and a note: 'You are creating a public repository in your personal account.' Finally, there's a blue button labeled 'Create repository'.

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 is titled 'We think you're gonna like it here.' and lists several suggested tasks: 'Invite your people' (with sub-tasks 'Invite your first member' and 'Customize members' permissions'), 'Collaborative coding' (with sub-tasks 'Create a pull request' and 'Create a branch protection rule'), and 'Automation and CI/CD'. A right-hand sidebar contains a notification 'You can now follow organization', a 'Discussions' section, and a 'Repositories' section with buttons for 'Create new repository' and 'Import'. The 'People' section at the bottom of the sidebar shows a placeholder for a member's profile.

The screenshot shows the 'Member privileges' settings page for a GitHub organization. It is divided into four sections: 'Base permissions' (with a 'Write' dropdown), 'Repository creation' (with checkboxes for 'Public' and 'Private'), 'Repository forking' (with a checkbox for 'Allow forking of private repositories'), and 'Pages creation' (with checkboxes for 'Public' and 'Private'). Each section includes a brief description of the permissions and a 'Save' button at the bottom.

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

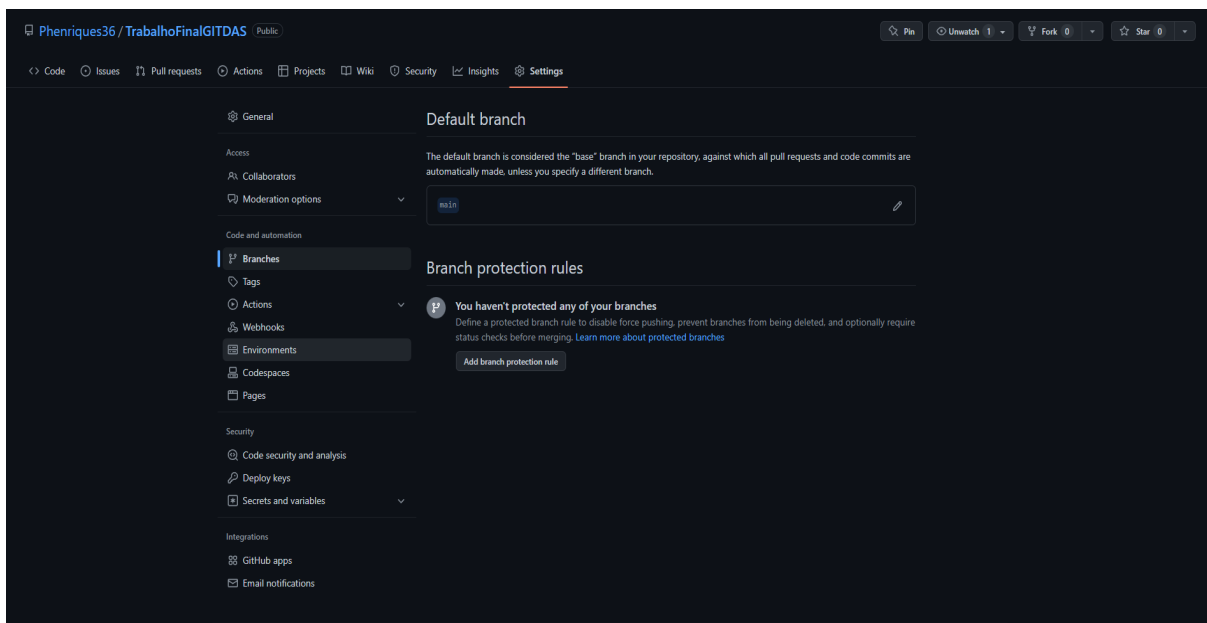
☒ **Allow members to create teams**

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