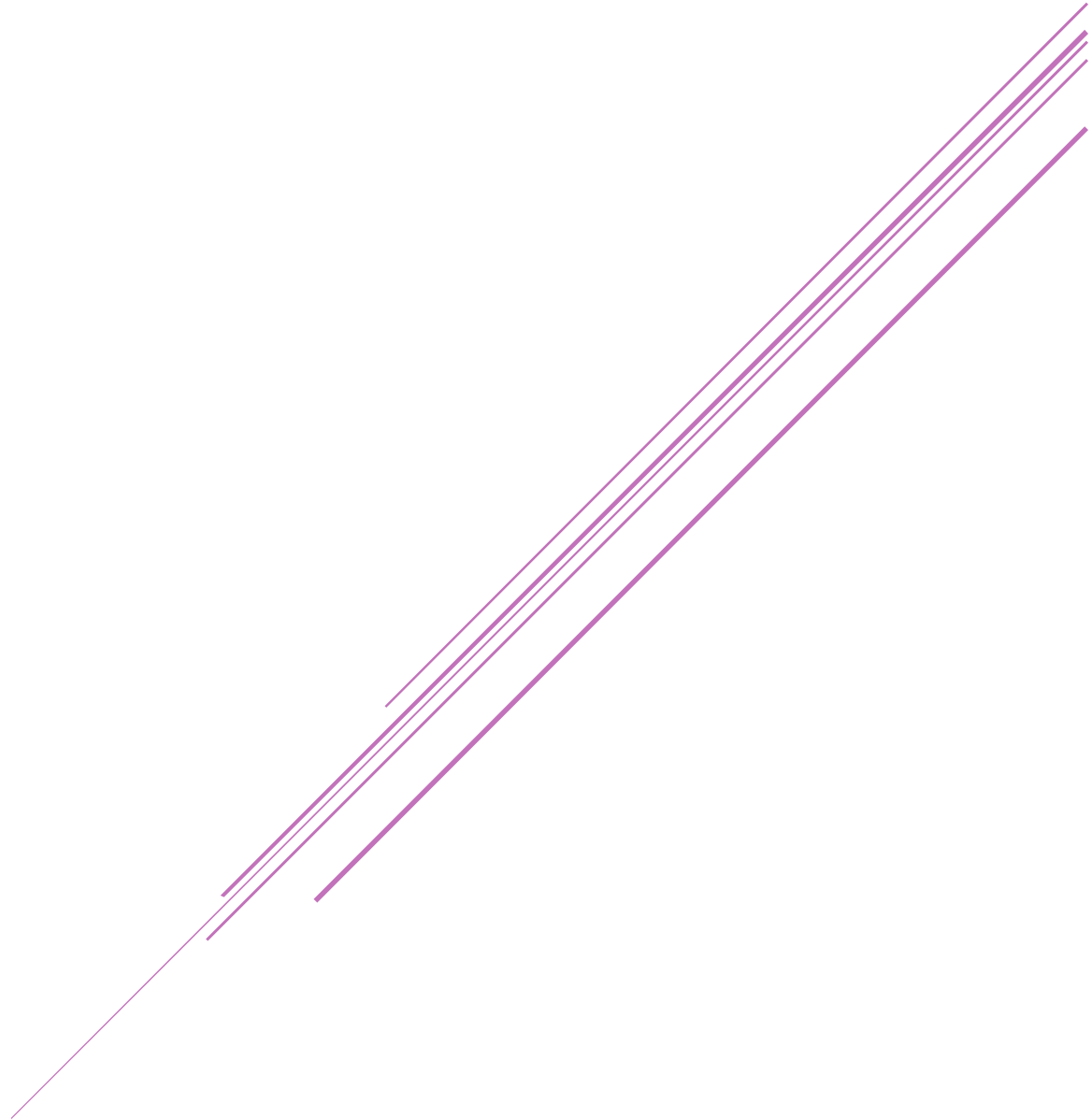


GitHub and Digital Repository Management

SPRINT 2

IN-MEXICO PROGRAM BACKEND

DEVELOPER CERTIFICATION





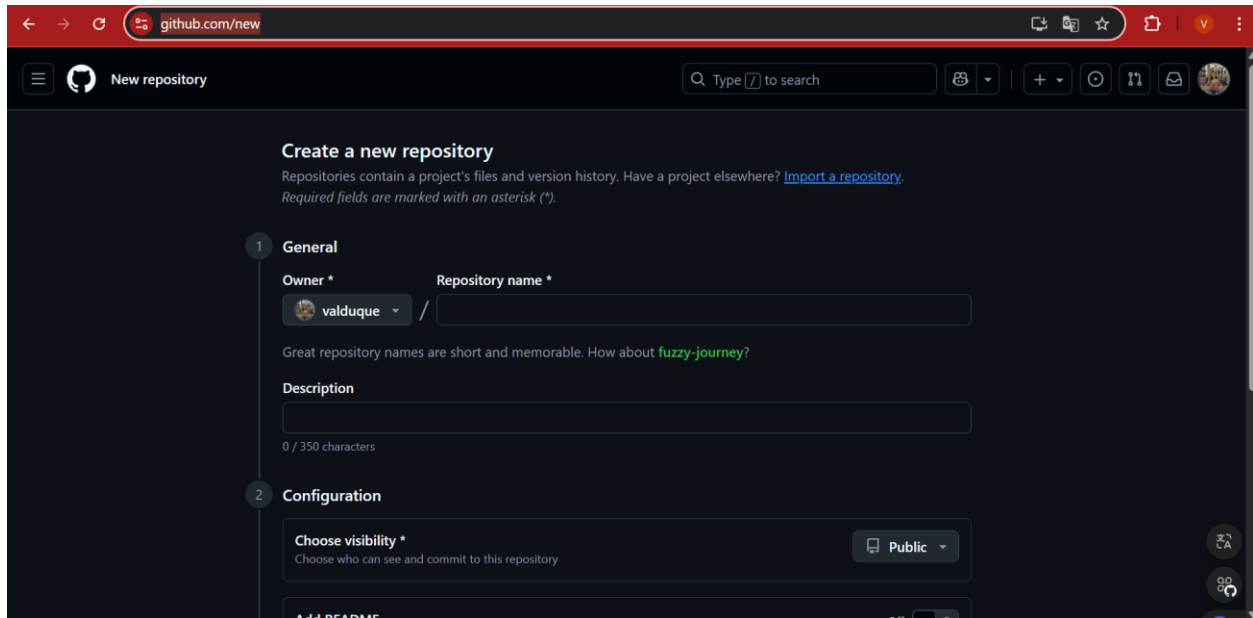
Contents:

| | |
|---|---|
| REPOSITORY CREATION IN GITHUB AND PROJECT UPLOAD: | 3 |
| 1. REPOSITORY CREATION: | 3 |
| 2. REPOSITORY CONNECTION WITH LOCAL REPOSITORY: | 4 |
| 3. PROJECT UPLOAD: | 4 |
| NEW BRANCH CREATION: | 6 |
| 4. UPDATE TO THE REQUIREMENTS.TXT FILE: | 6 |
| CREATED PR: | 7 |
| 5. PR APPROVAL IN GITHUB: | 9 |

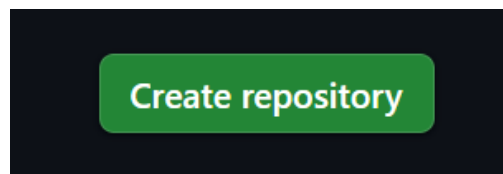
REPOSITORY CREATION IN GITHUB AND PROJECT UPLOAD:

1. REPOSITORY CREATION:

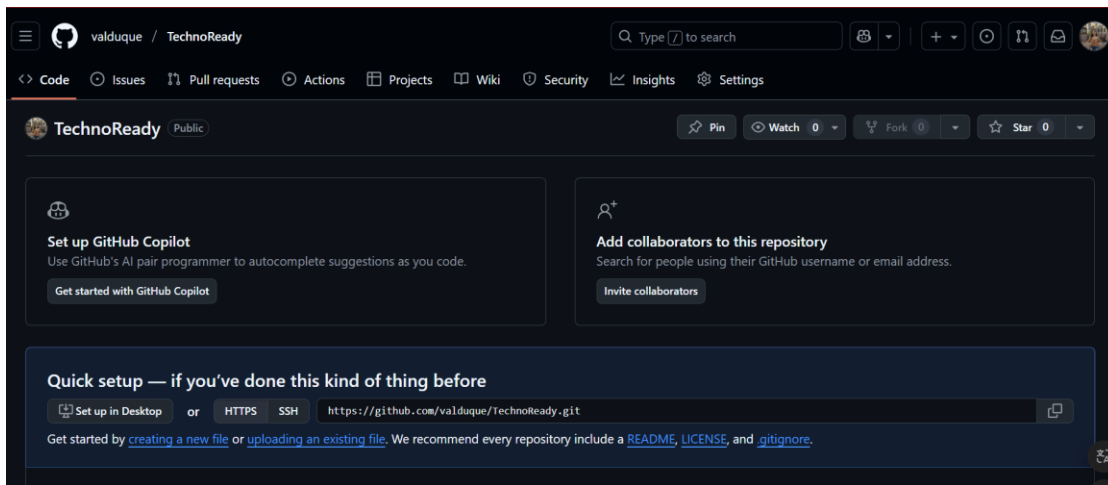
Inside the GitHub account, we must go to <https://github.com/new>. So, we can customize our repository:

A screenshot of the GitHub 'Create a new repository' page. The page has a dark theme. At the top, there's a navigation bar with the GitHub logo, 'New repository', a search bar, and several icons. The main content area is titled 'Create a new repository' and includes a sub-header 'Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository.](#)' and a note 'Required fields are marked with an asterisk (*)'. Below this, there are two sections: '1 General' and '2 Configuration'. In the 'General' section, there are fields for 'Owner *' (a dropdown menu showing 'valduque') and 'Repository name *' (a text input field). Below these is a suggestion: 'Great repository names are short and memorable. How about fuzzy-journey?'. There is also a 'Description' text area with a character count '0 / 350 characters'. The 'Configuration' section is partially visible, showing a 'Choose visibility *' dropdown menu set to 'Public'.

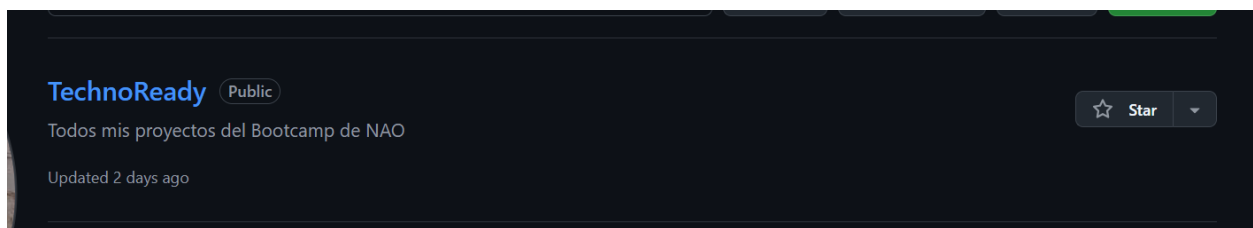
After that, just click the “Create repository” button:



And now you can see your repository's page:



Also, it'll appear on your repositories' list:



2. REPOSITORY CONNECTION WITH LOCAL REPOSITORY:

From the local repository, GitBash must be open and initialize with the “git init” command:

```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git init
Initialized empty Git repository in C:/Users/valer/Documents/TechnoReady/.git/
```

Then add the command “git init add origin” followed by the URL of you github repository:

```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git remote add origin ^[[200~https://github.com/valduque/TechnoReady~
```

3. PROJECT UPLOAD:

Inside our repository, and once our git is initiated, git execute the git add command, this will send all our files to the staging area:

```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git add .
```

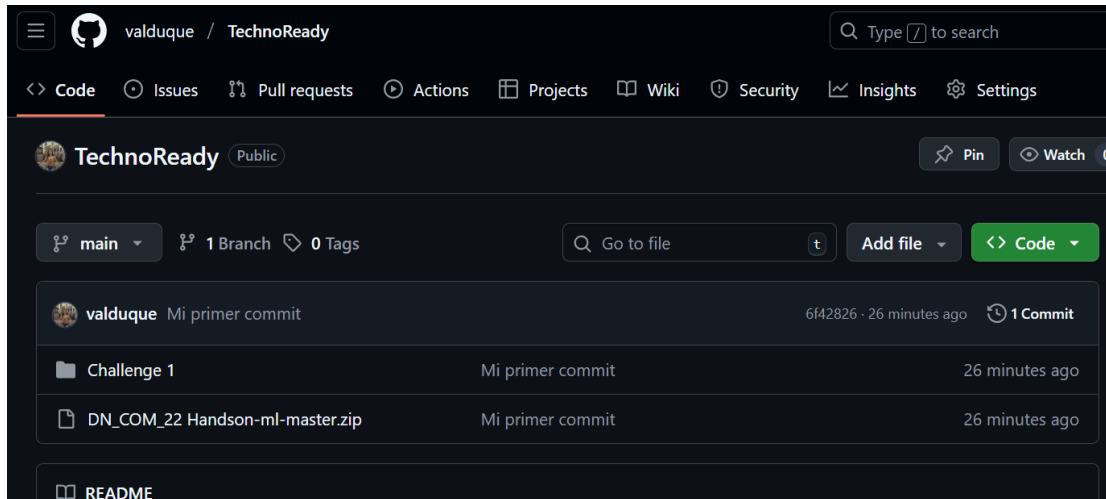
Then, we create a commit with the git commit -m "message" and press enter:

```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git commit -m "Mi primer commit"
[main (root-commit) 6f42826] Mi primer commit
10 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Challenge 1/Develop.docx
create mode 100644 Challenge 1/Plan.docx
create mode 100644 Challenge 1/SPRINT 1/Develop.pdf
create mode 100644 Challenge 1/SPRINT 1/Github link.pdf
create mode 100644 Challenge 1/SPRINT 1/Plan.pdf
create mode 100644 Challenge 1/SPRINT 1/ROADMAP.pdf
create mode 100644 Challenge 1/SPRINT 2/Develop.docx
create mode 100644 Challenge 1/SPRINT 2/~$evelop.docx
create mode 100644 Challenge 1/SPRINT 2/~WRL0003.tmp
create mode 100644 DN_COM_22 Handson-ml-master.zip
```

Then just write the command git push origin branch.name:

```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git push -u origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 18.28 MiB | 3.33 MiB/s, done.
Total 14 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/valduque/TechnoReady
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

And now the project appears inside the repository on GitHub:



NEW BRANCH CREATION:

Inside your project, write the command `git branch name.brach:`

```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git branch develop
```

To confirm the branch creation, just write the command `git branch:`

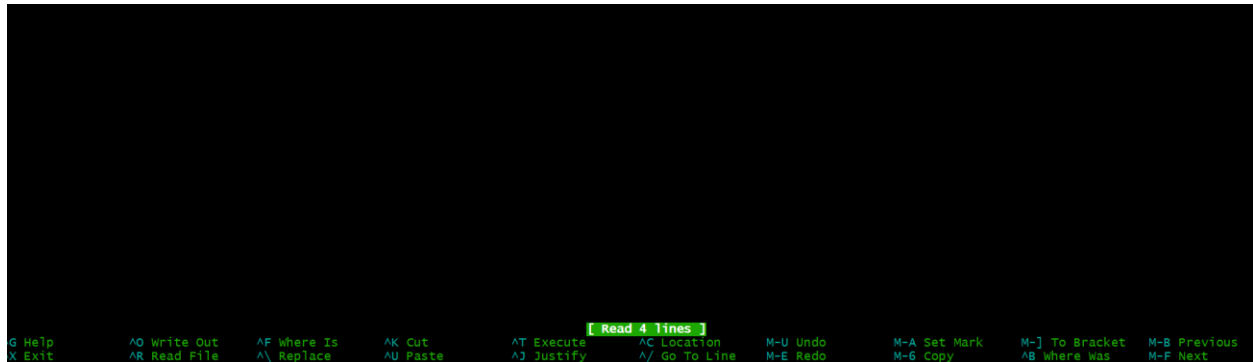
```
valer@LAPTOP-SI5J5VLM MINGW64 ~/Documents/TechnoReady (main)
$ git branch
* main
  develop
```

4. UPDATE TO THE REQUIREMENTS.TXT FILE:

If there is ever a desire to update something, there are some rules that must be followed. First, make the desired changes in your file:

```
MINGW64:/c/Users/valer/Documents/TechnoReady
GNU nano 8.5 requirements.txt
Requirements list
User stories
needs
priorities
```

Then, save the changes with the combinations in the console below:

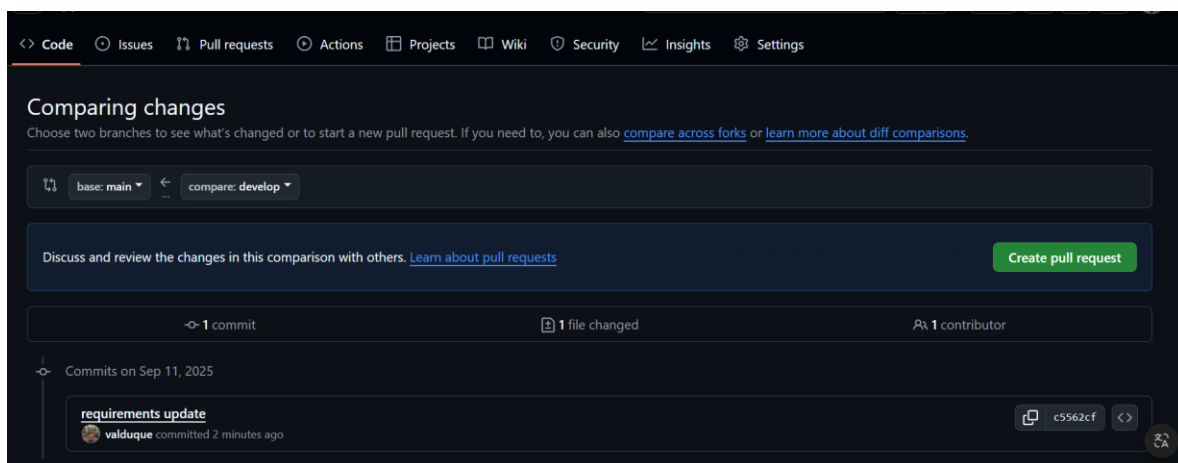


Once the changes have been done, you can upload them with a push to the desired branch.

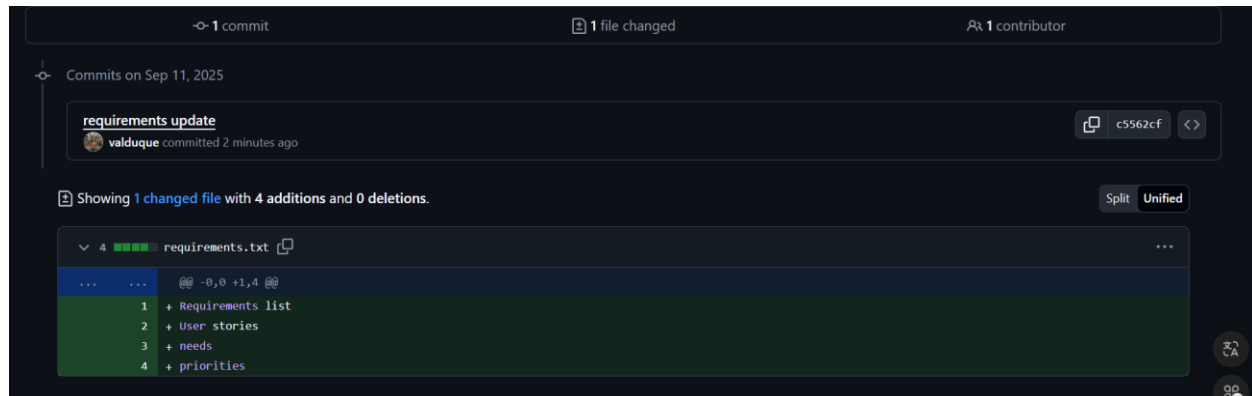
CREATED PR:

If there is content uploaded that must be reviewed by other collaborators, it's possible to create a PR to make the whole team to be on the same page or at least, on the same code.

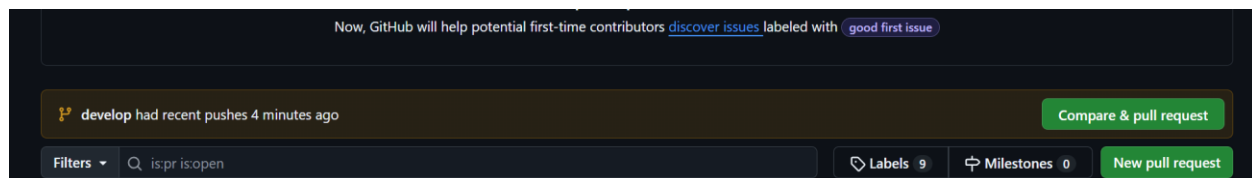
To do this, it must be already committed and pushed to the branch. Once it's done, in the project's menu should appear an option to create a pull request, it shall be clicked on the "create pull request" button:



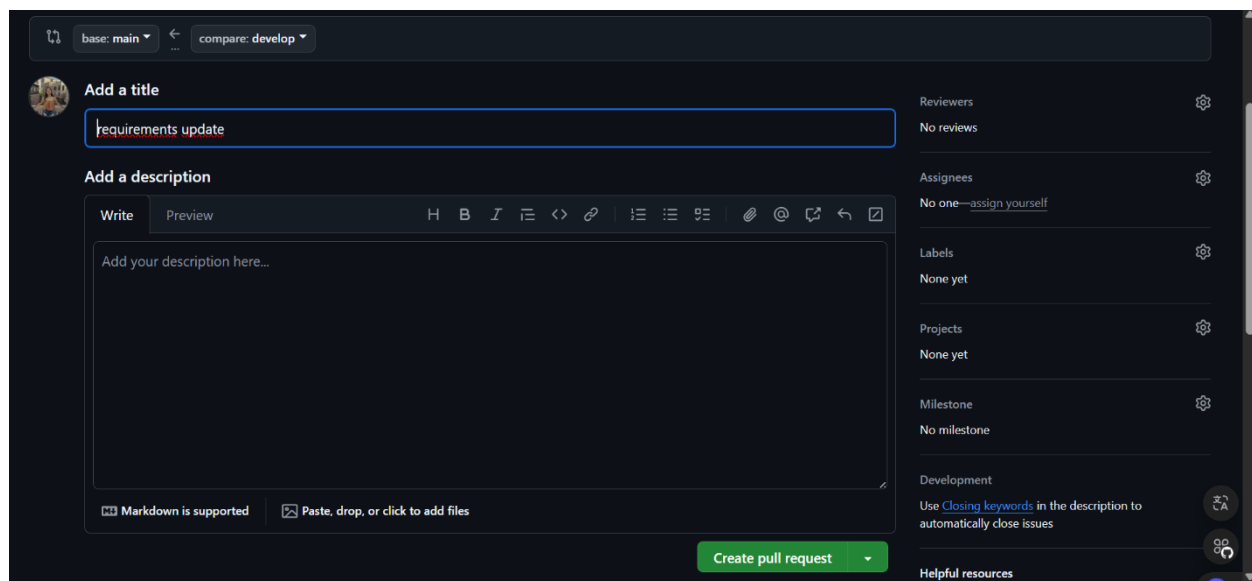
It will be able to show the changes:



And now, it's possible to compare and do the pull request:

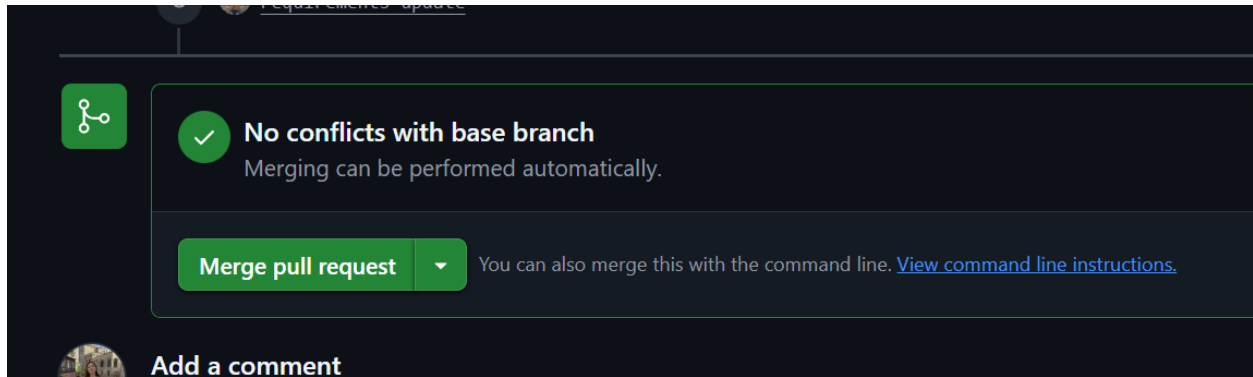


It can be added descriptions and comments:

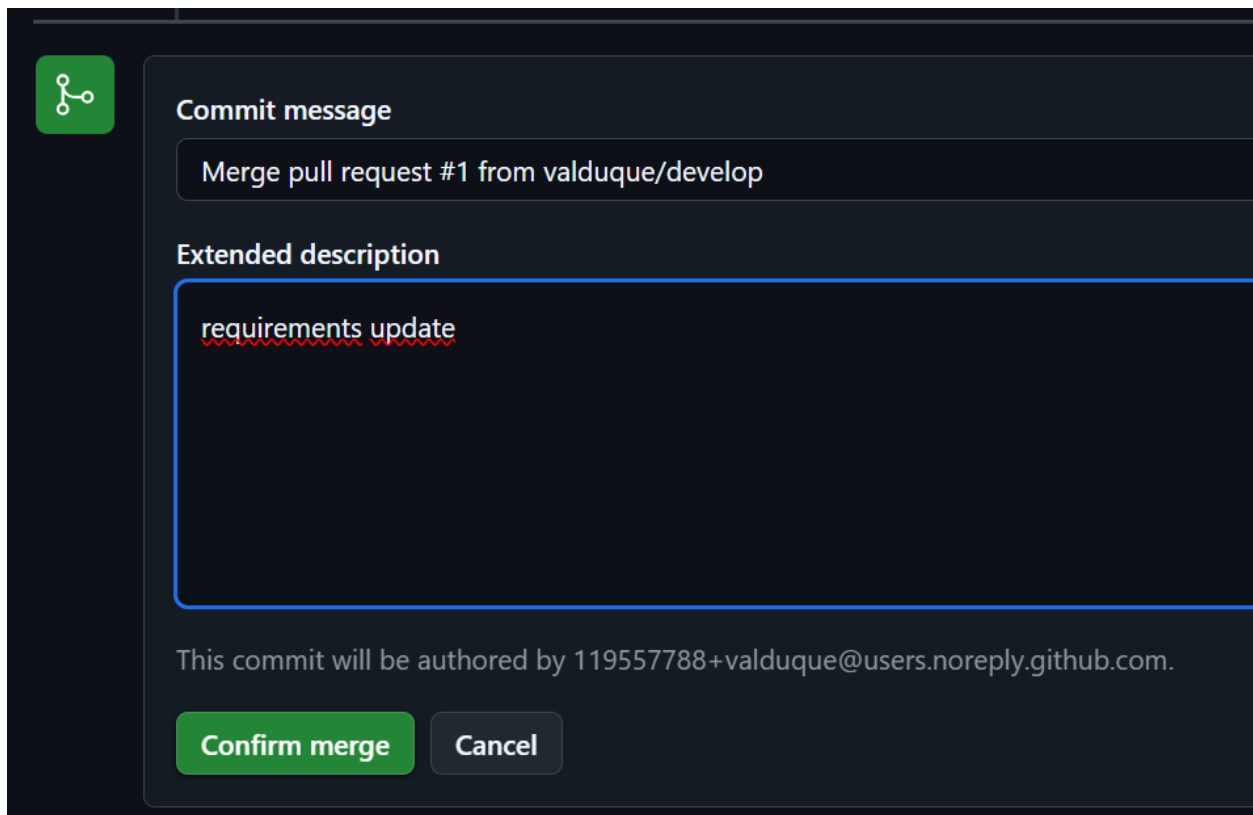


5. PR APPROVAL IN GITHUB:

If a pull request is done on the project, a collaborator can approve another collaborator's work to be added to the project. The GitHub system will show if there is any conflict with the branch, or if it's not possible to merge with the new work. In this case, it's possible:



It must be clicked the "merge pull request" button and add any commentary if needed:



Lastly, the system will confirm that the action is finished successfully:

