1º passo Criar e Conectar repositórios

* Repositório remoto: É o repositório online, no github;
* Repositório local: É o que criamos em nossas maquinas;
* Branch: como se fosse gavetas que colocamos as alterações do código para decisões e testagens. e só colocar na gaveta principal (main) quando as alterações forem aprovadas.

- Nosso objtivo é criar um repositório no github e depois criarmos um repositório local para assim conectar ambos. Para isso, no nosso terminal, após criar o repositório no github, digitamos:

"git imit " - Para iniciarmos um repositório local.
"git branch -M main" - Assim criamos a "gaveta principal".
"git remote add origim url.git " - Com isso, conectamos os repositórios

2º Passo Criar Commits

* Arquivos com a letra u: Significa que não está no repositorio ou não está atualizado;

- Aqui precisamos verificar qual arquivo foi enviado ou não, adicionar arquivos e enviá-los (commitar) com uma mensagem definindo o que muda com esse envio / atalização. Para isso, digitamos no terminal:

"git status" - Para ver quais arquivos não estão atualizados no repo (são os em vermelho).
"git add ." - Esse comando encaminha os arquivos em vermelho no repositório local.
"git commit -m "mensagem"" - Aqui resumimos qual foi o update.
"git log" - Por ultimo, aqui vemos os registros de commits.
"git push origim main" - Envia tudo acima do repositório local para o remoto.
"git pull origim main" - Baixar no seu repositório local as mudanças feitas no local.

Branch / Ramificações

* Merge: Integrar uma branch qualquer em outra (mesclagem do código).

"git branch" - Mostra as branchs que há no repositório remoto.

"git checkout -b nova branch" - Assim, o checkout altera para a nova branch, e o -b cria.

"git checkout branch" - Muda para a branch escolhida.

"git merge branch" - Para integrar as branchs.

↳ Tenho que estar na branch que será mesclada e escrever a que quero mesclar.

"git branch -d branch" - Exclui a branch do repositório local.

"git push origin branch --delete" - Exclui a branch do repositório remoto.

Padrões de branchs

→ Gitflow

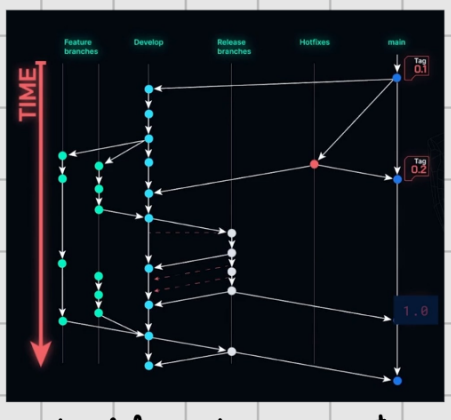
* Branchs principais

- Main: Código estável, passível de uso.
- Develop: Código base p/ criação de outras branchs.

* Branchs de feature: Para implementar funcionalidades.

* Branchs de release: Para fazer testes no código que estava na develop, para depois mesclar com a main. Se caso os testes derem errado, voltamos para a Develop, corrigimos e retornamos à realise até não ter mais erros e poder implementar o código na main. → Pode usar tags p/ cada versão.

* Branchs de hotfixes: Caso haja um problema crítico que precisa ser arrumado rápido, criamos essa branch, resolvemos o erro e integramos na MAIN e na DEVELOP. ↳ (os que impedem de usar)



(cada bolinha é um commit).

→ tags

"git tag -a versão x -m "versão x" - Dessa forma, com o -a definimos a versão e com -m, a mensagem que terá na tag.

→ Trunk Based

* Uma única branch, só cria uma outra se necessário. Usa para projetos menores, com menos pessoas.

Conflitos

* Conflito é quando duas pessoas fazem alteração na mesma parte do arquivo, e na hora de commitar, o repositório local e o remoto não estão sincronizados.

* Também acontece se tiver duas partes diferentes na mesma parte do arquivo, em branchs diferentes, e tenta mesclar uma delas na main.

→ Para resolver o conflito:

"git pull origin main" - Assim atualizamos nosso repo local com o que estava no repo local, e, na parte em conflito aparecerá dessa forma:

```
<<<<< HEAD
    Alterações feitas no repo local
= = = = = = = = = = = = = = = = = = = = = = = =
    Alterações feitas no repo remoto
```

Aqui há três opções:
- Manter a local e excluir a remota.
- Manter a remota e excluir a atual.
- Manter ambas alterações.

"git commit" - Para, após a correção poder commitar.

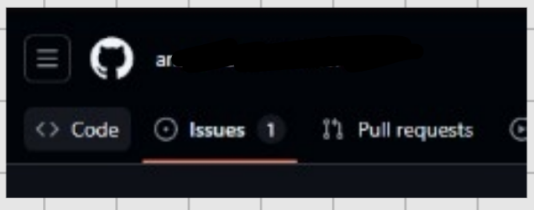
"git add ." - Logo em seguida adicionar as alterações.

"git commit" - Commitar sem mensagem, pois, nesse caso, o github coloca automático.

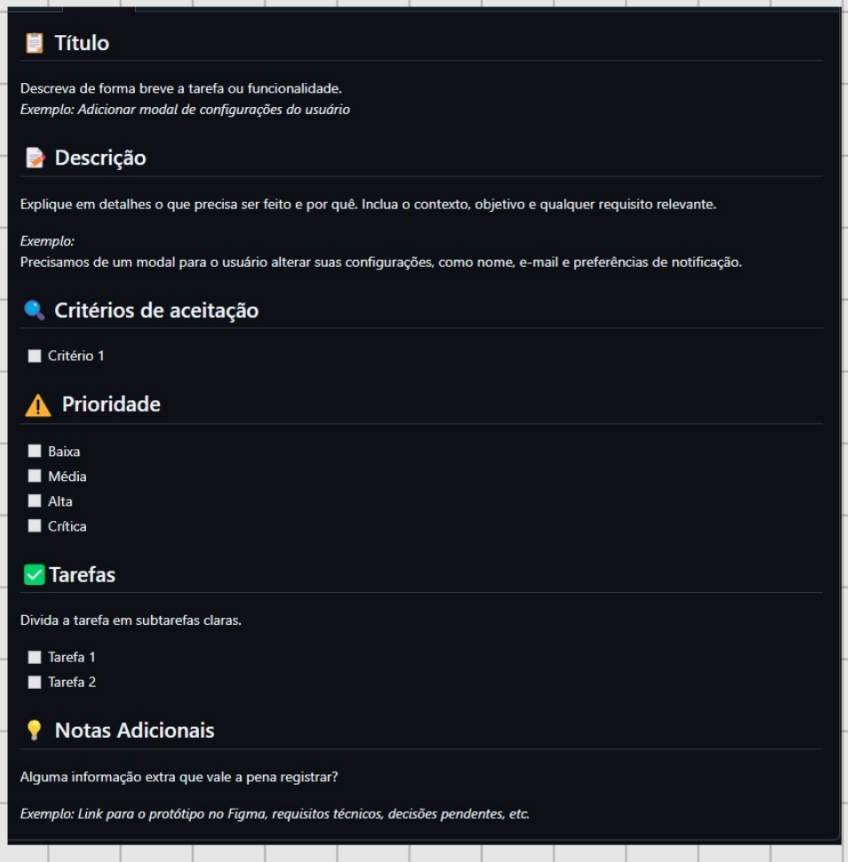
":qa" - Caso abra o vim, assim o fecha e salva as alterações.

Issues

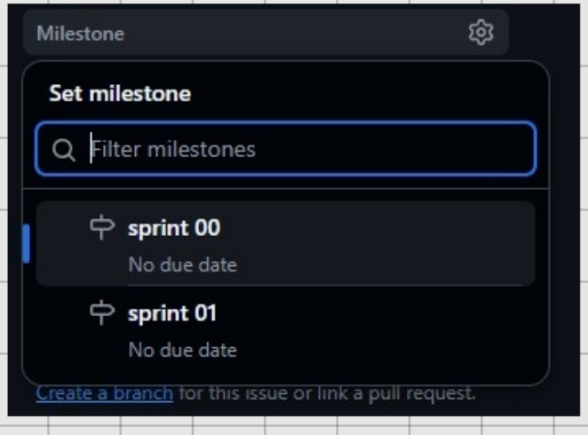
* Issues é uma forma de organizar o projeto com checklists, determinando quando precisa fazer documentações, resoluções de bugs, adicionar funcionalidades, etc...



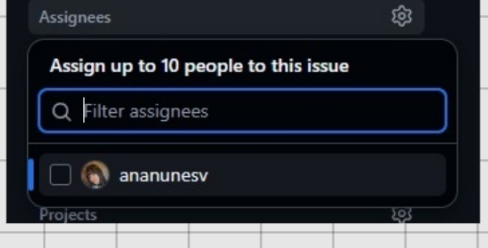
→ A issues fica nessa parte do github.



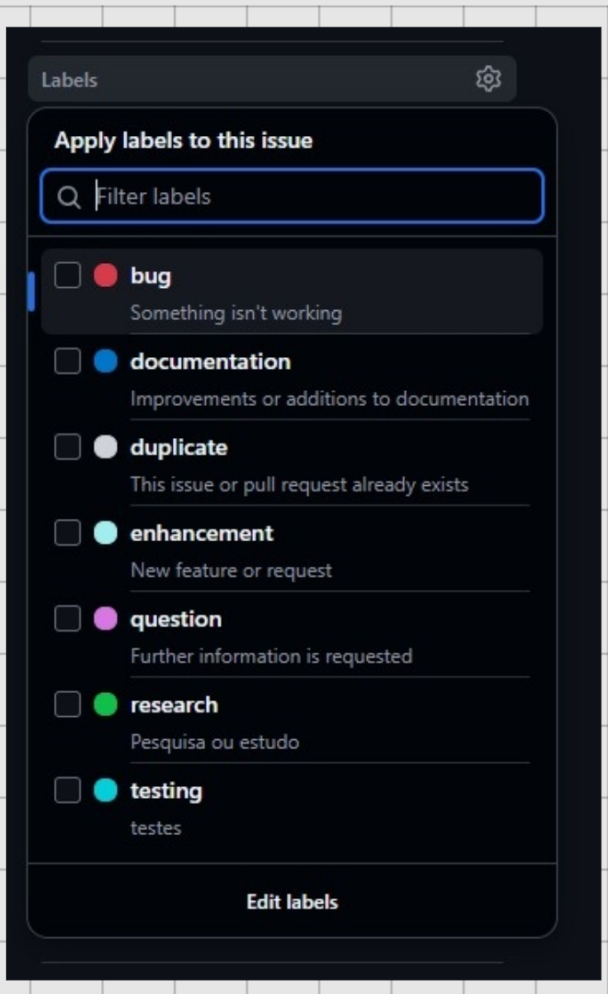
→ Esse será o template das nossas issues.



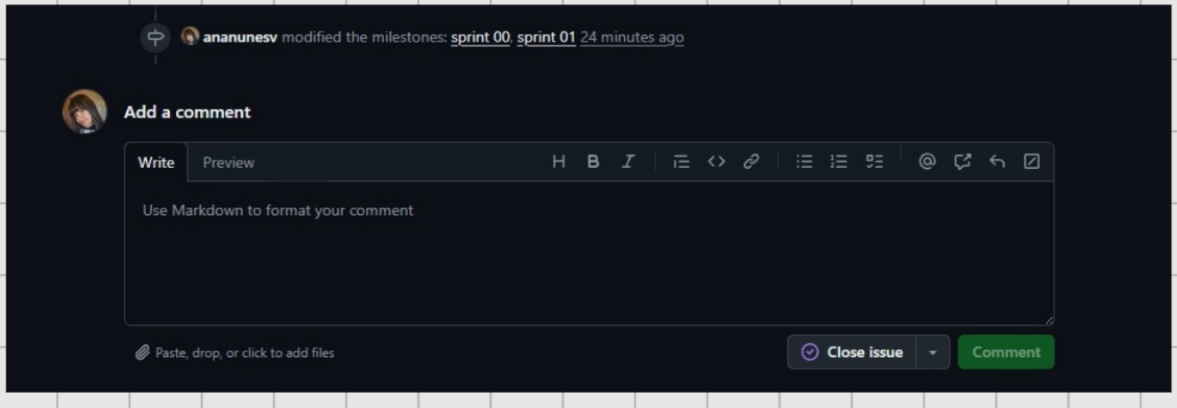
→ Aqui selecionamos em qual "sessão" de issues ficará.



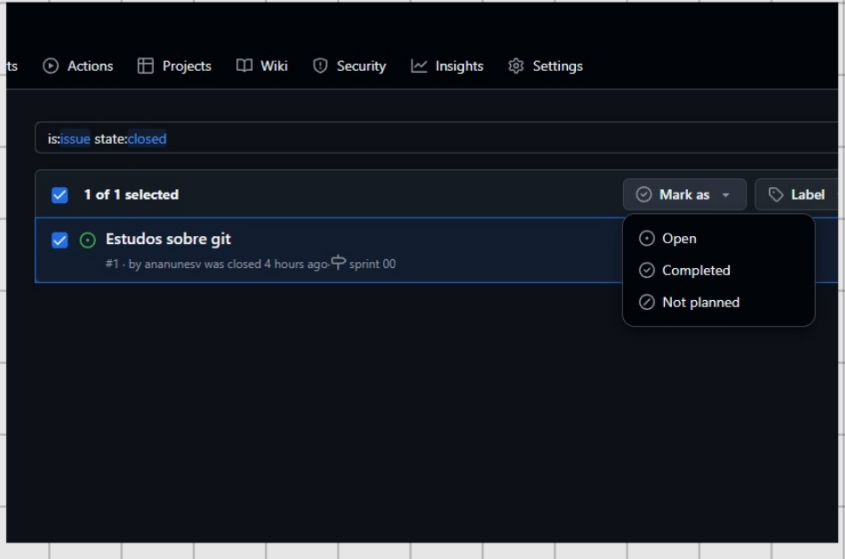
→ Aqui definimos os responsáveis.



→ Aqui ficam as tags para organizar cada issue.



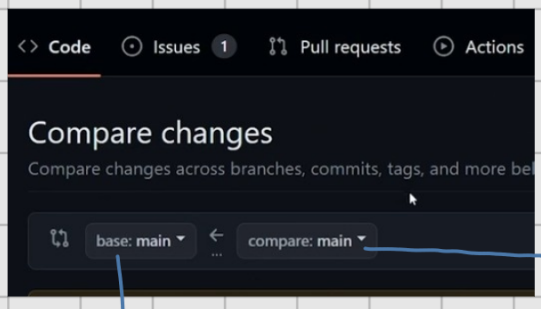
→ Nos comentários, atualizamos sobre a issue (Anotações, etc).



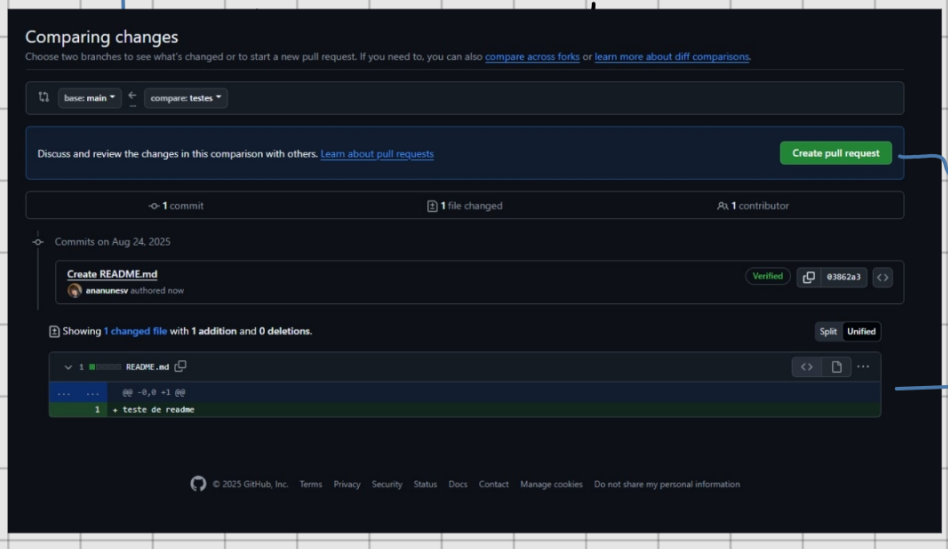
→ Na opção "Mark as" marcamos a issue como concluída.

Pull Request

* São registros de possíveis merges que podemos fazer, assim, analisamos e testamos antes de integrar as branchs (uma boa prática).

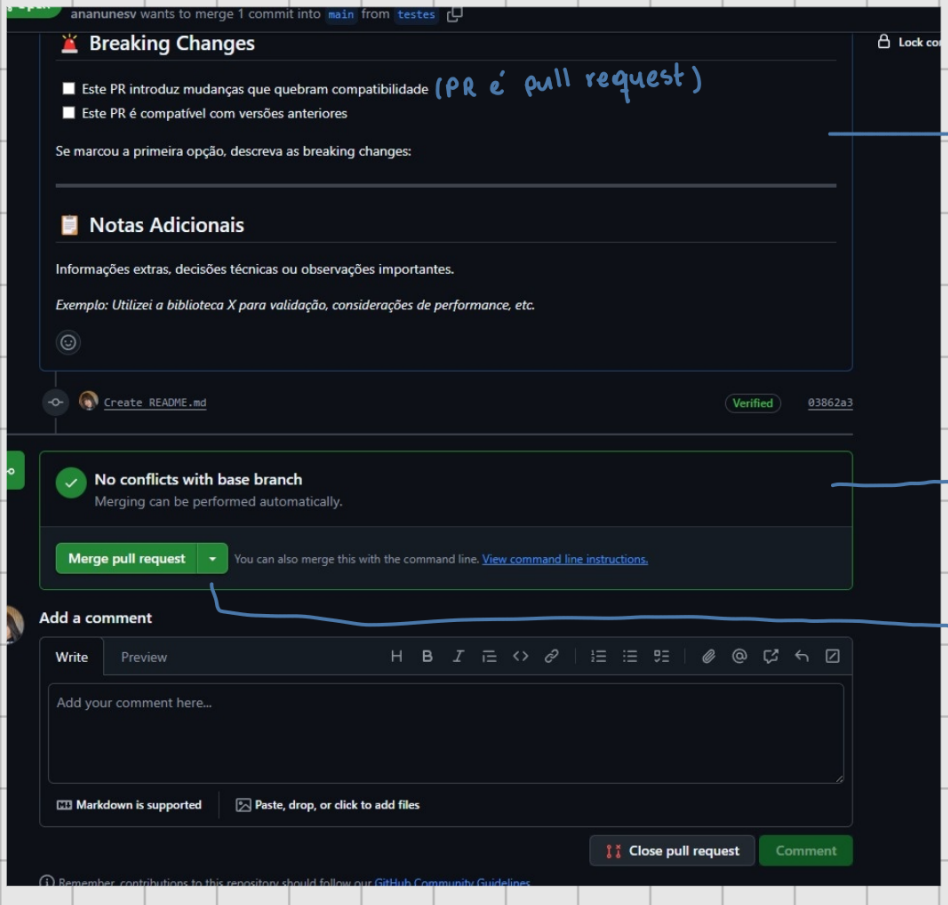


Branch que contém as mudanças.



Aqui criamos a pull request.

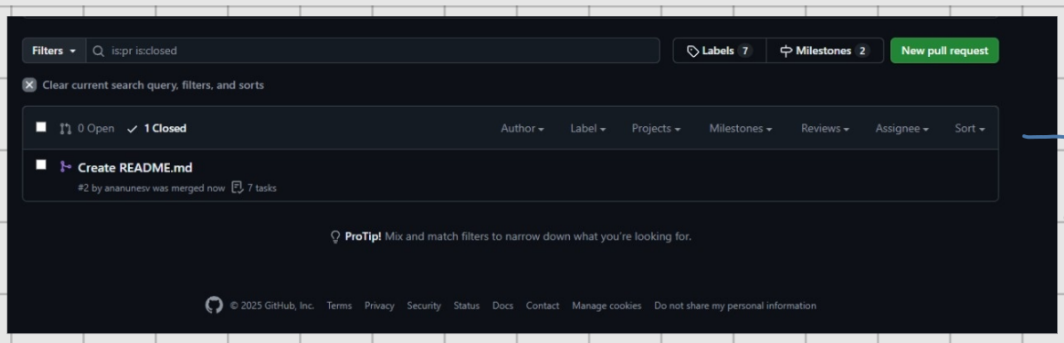
Aqui são apontadas as alterações.



Template para definir e registrar a pull request.

A pull request fica registrada até decidir fazer a merge.

Por último, aqui fará a merge.



→ Ao final, cada pull request ficará registrada.