

1º passo Criar e Conectar repositórios

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

- Nosso objetivo 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 init" - Para iniciarmos um repositório local.

"git branch -M main" - Assim criarmos a "gaveta principal".

"git remote add origem url.git" - Com isso, conectarmos os repositórios.

2º Passo Criar Commits

- * Arquivos com a letra u: Significa que não está no repositório ou não está atualizado;
- Aqui precisaremos verificar qual arquivo foi enviado ou não, adicionar arquivos e enviá-los (commitar) com uma mensagem definindo o que muda com esse envio / atualização. Para isso, digitamos no terminal:

"git status" - Para ver quais arquivos não estão atualizados no repositório (não ou 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 último, aqui vemos os registros de commits.

"git push origem main" - Envia tudo acima do repositório local para o remoto.

"git pull origem main" - Baixa no seu repositório local as mudanças feitas no local.

Branch / Ramificações

- * Merge: Integrar uma branch qualquer em outra (mergulhagem de código).

"git branch" - Mostra as branches 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 branches.

↳ Tenho que estar na branch que será mergulhada e escrever a que quero merger.

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

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

Padrões de branches

→ Griffflow

* Branches principais

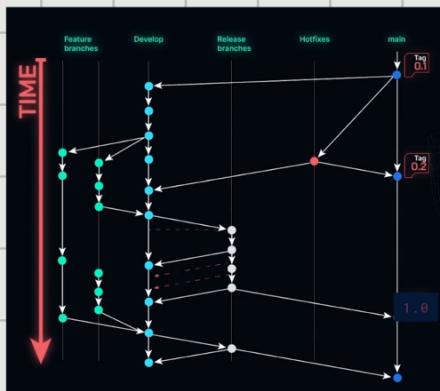
- Main: Código estável, passível de uso.

- Develop: Código base p/ criação de outras branches.

* Branches de feature: Para implementar funcionalidades.

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

* Branches de hotfixes: Caso haja um problema crítico que precisa ser corrigido rápido, criar essa branch, resolver o erro e integrar na MAIN e na DEVELOP. → (os que impedem de usar)



→ tags

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

(cada bolinha é um commit).

→ Trunk Based

- * Uma única branch, só cria uma outra se necessário. Usa para projetos menores, com menor peso.

Conflicts

- * Conflito é quando duas pessoas fazem alterações na mesma parte do arquivo, e na hora de commitar, o repositório local e o remoto não estão sincronizados.
 - * Também ocorre se tiver duas partes diferentes na mesma parte do arquivo, em branches diferentes, e

→ Para resolver o conflito:

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

<<<< HEAD

Alterações feitas no repositorios local

Aqui há três opções:

Mantener local e excluir a remota.

Mantener la remota y excluir la actual.

Mantener ambar alteraçõez.

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

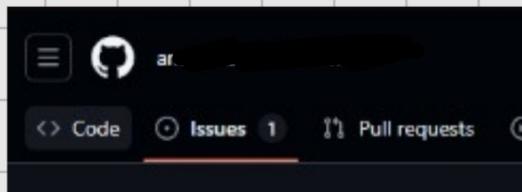
"git add . " - logo em seguida adicionam as alterações.

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

"**:ga**" - Cosa abra o vin, 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.

Título
Descreva de forma breve a tarefa ou funcionalidade.
Exemplo: Adicionar modal de configurações do usuário

Descrição
Explique em detalhes o que precisa ser feito e por quê. Inclua o contexto, objetivo e qualquer requisito relevante.
Exemplo:
Precisamos de um modal para o usuário alterar suas configurações, como nome, e-mail e preferências de notificação.

Critérios de aceitação
Critério 1

Prioridade
Baixa
Média
Alta
Crítica

Tarefas
Divida a tarefa em subtarefas claras.
Tarefa 1
Tarefa 2

Notas Adicionais
Alguma informação extra que vale a pena registrar?
Exemplo: Link para o protótipo no Figma, requisitos técnicos, decisões pendentes, etc.

→ Esse será o template das nossas issues.

Milestone

Set milestone

Filter milestones

sprint 00
No due date

sprint 01
No due date

Create a branch for this issue or link a pull request.

→ Aqui selecionaremos em qual "sprint" de issues ficará.

Assignees

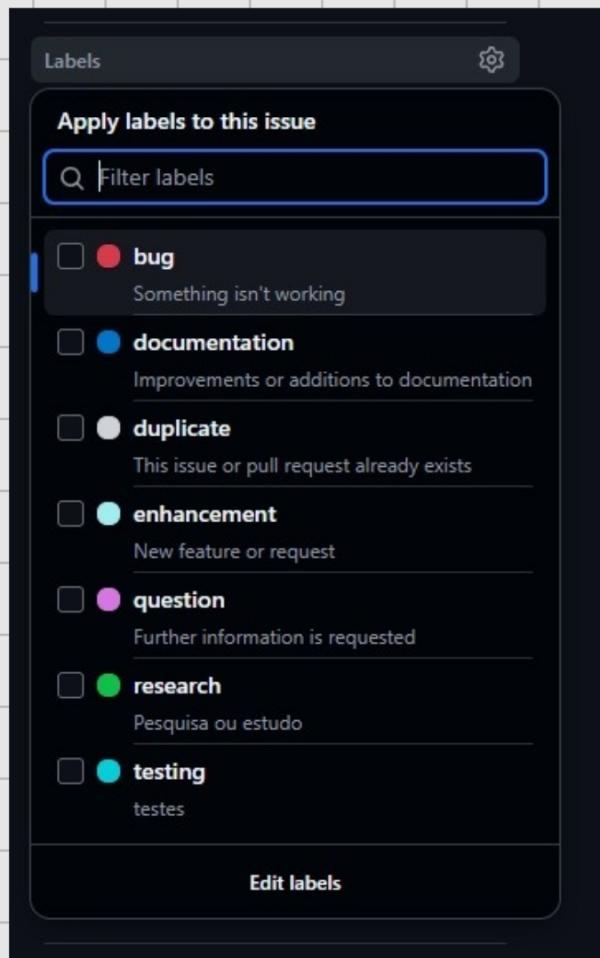
Assign up to 10 people to this issue

Filter assignees

ananunesv

Projects

→ Aqui definimos os responsáveis.



→ Aqui ficam as tags para organizar cada issue.

A screenshot of a comment input field. At the top, it shows a notification: "ananunesv modified the milestones: sprint 00, sprint 01 24 minutes ago". Below is a "Add a comment" section with "Write" and "Preview" tabs. The "Write" tab has a rich text editor toolbar with icons for H, B, I, etc. Below the toolbar is a text area with placeholder text "Use Markdown to format your comment". At the bottom right are "Close issue" and "Comment" buttons.

→ Novos comentários, atualizarmos sobre a issues (Anotações, etc).

A screenshot of a list view. At the top, there are navigation links: Actions, Projects, Wiki, Security, Insights, Settings. Below is a search bar with the query "is:issue state:closed". A list item is selected: "Estudos sobre git" (checkbox checked). To the right of the list item is a "Mark as" dropdown menu with three options: "Open", "Completed", and "Not planned".

→ Na opção "Mark as" marcar a issues como concluida.

Pull Request

- * São requisitos de possíveis merges que podemos fazer, assim, analisarmos e testarmos antes de integrar as branches (uma boa prática).

The screenshot shows the GitHub interface for comparing changes between two branches. At the top, there are dropdown menus for 'base: main' and 'compare: main'. A blue arrow points from this area to the text 'Branch que contém as mudanças' (Branch that contains the changes). Below the dropdowns, there's a summary: '1 commit', '1 file changed', and '1 contributor'. A green button labeled 'Create pull request' is highlighted with a blue arrow and labeled 'Aqui criamos a pull request.' (Here we create the pull request.) The main content area shows a diff view of the README.md file, with one addition: '+ teste de readme'. Another blue arrow points to this diff area with the text 'Aqui não apontadas as alterações.' (Here the changes are not pointed out.)

This screenshot shows a modal dialog for creating a pull request. It includes sections for 'Breaking Changes' (with a note that 'Este PR introduz mudanças que quebram compatibilidade' and 'PR é pull request'), 'Notas Adicionais' (with a note about extra information), and a summary section stating 'No conflicts with base branch' and 'Merging can be performed automatically'. A large green button labeled 'Merge pull request' is prominent. A blue arrow points to this button with the text 'Template para definir e registrar a pull request.' (Template to define and register the pull request.) Below the button, there's a comment input field with the placeholder 'Add a comment' and a rich text editor. A blue arrow points to this field with the text 'Por último, aqui fará a merge.' (Finally, here it will perform the merge.)

Filters **Labels** 7 **Milestones** 2 **New pull request**

0 Open 1 Closed

Create README.md

#2 by ananunesv was merged now 7 tasks

ProTip! Mix and match filters to narrow down what you're looking for.

© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

→ Ao final, cada pull request ficará registrada.