📘 Parte Teórica — O que é uma *Branch* no Git?

Uma **branch** (ramo) em Git é como uma linha do tempo paralela do seu projeto. Imagine que o repositório principal (geralmente chamado main ou master) é uma linha reta. Sempre que alguém cria uma branch, está basicamente "duplicando" o estado atual do projeto para começar a trabalhar em algo novo, sem interferir no que já está funcionando no projeto principal.

Por que usar *branches*?

- **Isolamento de tarefas**: cada pessoa pode trabalhar em uma nova funcionalidade, correção ou melhoria, separadamente.
- **Evita conflitos no código principal**: como as alterações estão separadas da branch principal, evita-se problemas em funcionalidades que já estão em produção.
- **Facilita a colaboração**: diferentes pessoas podem trabalhar em diferentes branches e depois unir (*merge*) tudo de forma controlada.
- **Histórico limpo**: ao usar branches, é mais fácil organizar o histórico de mudanças.

💡 Cenário Prático: Entrando em um Projeto com Git e Trabalhando com Branches

Vamos simular a seguinte situação:

Você chegou hoje para trabalhar em um projeto que já está em andamento. Seu supervisor te passou uma tarefa para implementar em uma das páginas. O projeto está hospedado no GitHub e você precisa trabalhar de forma organizada, sem atrapalhar os outros colaboradores.

📌 Etapas detalhadas (sem comandos, só explicação):

🔹 1. Clonar o projeto do repositório principal

Você acessa o repositório oficial do projeto (geralmente está no GitHub) e **clona ele para a sua máquina**. A partir desse momento, você tem uma cópia completa do projeto localmente.

🔹 2. Criar uma nova branch para sua tarefa

Antes de começar a programar qualquer coisa, você **cria uma nova branch** com um nome descritivo para a tarefa que você vai realizar. Exemplo: ajuste-formulario-contato.

Essa branch serve para isolar suas alterações. Você vai trabalhar nela sem modificar diretamente a branch principal (main), garantindo que o projeto continue estável para os outros.

🔹 **3. Implementar a funcionalidade**

Agora sim, dentro da sua branch, você realiza todas as alterações necessárias: edita arquivos, cria novos, testa localmente, etc.

Tudo que for feito aqui está sendo registrado apenas na sua branch. O repositório principal continua intacto até que você peça para unir suas mudanças com ele.

🔹 **4. Salvar suas alterações localmente**

Depois de terminar sua implementação, você **salva e registra** essas alterações (através de commits). Ainda assim, tudo continua apenas na sua máquina ou na sua branch específica.

🔹 **5. Enviar a sua branch para o repositório remoto**

Você **envia sua branch para o GitHub**. Isso permite que outras pessoas vejam o que você fez, revisem, e testem se necessário.

🔹 **6. Criar um *Pull Request* (PR)**

Agora que sua branch está no GitHub, você **abre um pull request**. Isso é basicamente um pedido para que as suas alterações na sua branch sejam **mescladas (mergeadas)** com a branch principal do projeto.

O PR pode ser revisado por um responsável técnico ou pelo supervisor. Eles podem comentar, pedir ajustes, ou aprovar diretamente.

🔹 **7. Revisão e aprovação**

O responsável pela revisão analisa seu pull request. Se estiver tudo certo, ele **aprova e faz o merge** das suas alterações com a branch principal. A partir deste ponto, sua contribuição passa a fazer parte do projeto oficial.

🔹 **8. Fim do ciclo: branch pode ser excluída**

Depois do merge, sua branch **pode ser excluída** tanto localmente quanto no GitHub, porque a tarefa já foi concluída e integrada ao projeto.

✅ **Resumo da Sequência:**

1. Você clona o repositório principal.
2. Cria uma nova branch com base na main.
3. Faz as alterações referentes à sua tarefa.
4. Salva essas alterações (commits).
5. Envia sua branch para o repositório remoto.
6. Cria um *pull request* pedindo para juntar com a main.
7. O responsável revisa e aprova.
8. A branch é unida à principal (*merge*), e pode ser deletada.

🛠️ **Parte Prática — Trabalhando com *branches* em um projeto Git**

👣 **1. Clonar o repositório principal**

Primeiro, você precisa **clonar** o repositório oficial do projeto para a sua máquina:

git clone https://github.com/usuario/repo-projeto.gitEntre na pasta do projeto:

cd repo-projeto

👣 **2. Criar uma nova *branch* com base na main**

Antes de criar a branch, é importante garantir que sua cópia da branch principal esteja atualizada:

git checkout main

git pull origin main

Agora sim, crie uma nova branch com um nome que represente a tarefa:

git checkout -b ajuste-formulario-contato

Esse comando cria e já troca para a nova branch.

👣 3. Fazer as alterações no código

Agora é o momento de implementar a funcionalidade no código. Quando terminar, salve todos os arquivos modificados.

👣 4. Adicionar e commitar as alterações

Depois de implementar a funcionalidade, adicione os arquivos modificados:

git add .

E crie um commit explicando o que foi feito:

git commit -m "Implementa melhorias no formulário de contato"

👣 5. Enviar a branch para o repositório remoto

Agora que tudo está pronto localmente, envie sua branch para o GitHub:

git push origin ajuste-formulario-contato

👣 6. Criar um *Pull Request* no GitHub

Depois de fazer o *push*, acesse o repositório no GitHub. O site vai sugerir automaticamente que você crie um **Pull Request** da sua branch (ajuste-formulario-contato) para a main.

Clique em **"Compare & pull request"** e depois em **"Create pull request"**.

👣 7. Revisão e *merge* pelo responsável

O responsável do projeto vai revisar suas mudanças. Se estiver tudo certo, ele pode fazer o *merge* diretamente pela interface do GitHub clicando em:

- **Merge pull request**
- **Confirm merge**

👣 8. Excluir a branch (opcional)

Após o *merge*, o GitHub oferece a opção de deletar a branch no botão:

Delete branch

Você também pode deletá-la da sua máquina local:

git branch -d ajuste-formulario-contato

E do repositório remoto, se desejar:

git push origin --delete ajuste-formulario-contato

✅ Fluxo completo em resumo com comandos:

1. Clonar o repositório

git clone https://github.com/usuario/repo-projeto.git

cd repo-projeto

2. Atualizar a main e criar uma branch nova

git checkout main

git pull origin main

git checkout -b nome-da-sua-branch

3. Fazer alterações no código

4. Adicionar e commitar

git add .

git commit -m "Descrição clara do que foi feito"

5. Enviar a branch

git push origin nome-da-sua-branch

6. Criar o Pull Request no GitHub

7. Aguardar revisão e merge

8. (Opcional) Apagar a branch

git branch -d nome-da-sua-branch

git push origin --delete nome-da-sua-branch