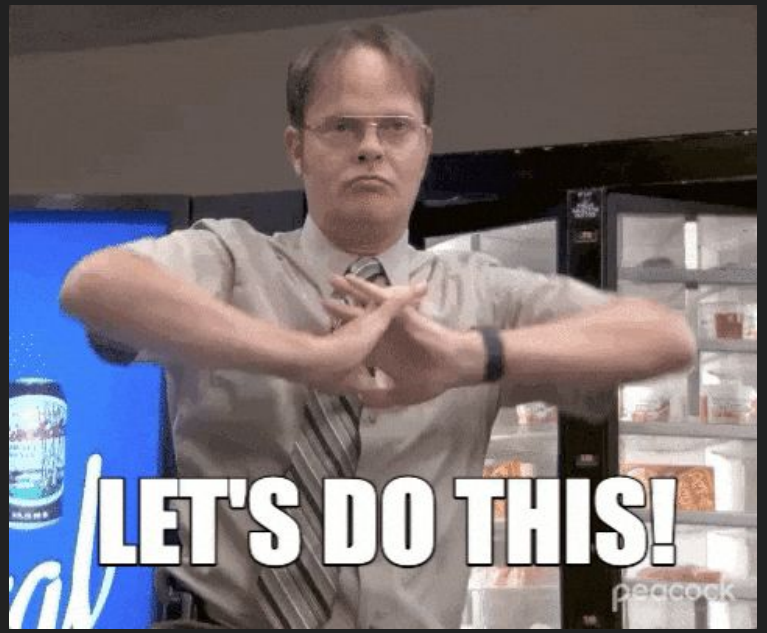
Olá, GitHub!

Entendendo o fluxo do Gitflow

O que veremos hoje?

- Conceitos
- Repositórios
- Issues
- Branches
- Commits
- Pull Requests
- Merge
- Gitflow



O que é o GitHub?

O GitHub é uma plataforma online que permite armazenar, gerenciar e colaborar em projetos de software usando o sistema de controle de versões Git.

Git != GitHub

GIT

- Ferramenta de controle de versões
- Funciona offline
- É apenas o mecanismo
- É open source

GITHUB

- Plataforma baseada na web
- Precisa de internet
- Adiciona recursos extras
- É uma empresa

Git é a ferramenta que controla versões de código localmente, enquanto GitHub é uma plataforma online que hospeda repositórios Git e facilita a colaboração.

Ferramentas que fazem uso do git:

“Talk is cheap, show me the code.”
- Linus Torvalds

Criando um repositório

- Um repositório é um local onde os arquivos de um projeto e o histórico de alterações ficam armazenados.

- O GitFica contará com 3 repositórios:
  - Documentação
  - Backend
  - Frontend

Vamos criar um repositório?

- Crie um repositório no GitHub chamado hello-world.
- Esse repositório deve conter um arquivo de README.
- O repositório pode ser público ou privado.
- Um repositório pode possuir uma licença.

Clonando o repositório

- Em um terminal em seu computador, execute o comando:

  ```
  git clone URL-REPOSITORY
  ```

- Assim teremos um **repositório local** em nosso computador, e um **repositório remoto** no GitHub.

Trabalhando com issues

- Uma issue é uma forma de registrar tarefas, problemas ou sugestões de melhorias dentro de um repositório.

- No GitFica temos templates para criação de issues:
  - Issues genéricas
  - Bugs
  - Features

- Vamos criar uma Issue?

Zenhub

- O ZenHub é uma ferramenta de gerenciamento de projetos integrada ao GitHub, que permite organizar e acompanhar tarefas, sprints e fluxos de trabalho diretamente no repositório.

- Recomendamos fortemente que vocês instalem a extensão do Zenhub, para terem acesso diretamente pelo GitHub.

Trabalhando com Branches

- Uma branch é uma linha independente de desenvolvimento onde você pode fazer mudanças sem afetar o projeto principal.

- A branch main é a principal linha de desenvolvimento que representa a versão estável e oficial do projeto.

- Criamos branches adicionais com base na main para desenvolver novas funcionalidades ou corrigir erros sem comprometer a versão principal do projeto.

Trabalhando com branches

- Para visualizar a branch atual, rode o comando:

  ```
  git branch
  ```

- Para criar uma nova branch a partir da main, rode o comando:

  ```
  git checkout -b nome-nova-branch
  ```

Criando uma nova branch

- Crie uma nova branch, chamada atualizando-readme:

  ```
  git checkout -b atualizando-readme
  ```

- Seguimos algumas boas práticas na criação de branches. Verifique os detalhes em nosso guia de contribuição.

Realizando modificações no projeto

- Na IDE de sua preferência, modifique o conteúdo do arquivo de README.

Subindo as modificações para o GitHub

- Para sincronizar as alterações do seu repositório local, para o repositório do GitHub, siga o seguinte passo a passo:

```
git pull origin nome-branch
```

  - Verifique as modificações:

```
git status
```

  - Adicione quais alterações vão para o GitHub:

```
git add nome-do-arquivo
```

  - Crie um commit:

```
git commit -m “Mensagem breve descrevendo o que foi feito”
```

  - Dê push no commit:

```
git push origin nome-branch
```

Voilá! As mudanças estão no GitHub.

Boas práticas para commits

- **Mensagens claras e concisas:** Escreva mensagens que descrevam claramente o que foi alterado e por que.

- **Commits pequenos e frequentes:** Faça commits menores, que abordem uma única alteração ou tarefa, para facilitar a revisão e o entendimento.

- **Use o tempo verbal correto**: Use o imperativo no presente (ex: "Adiciona nova funcionalidade de login").

- **Evite commits de "última hora":** Não faça commits de mudanças não relacionadas em um único commit.

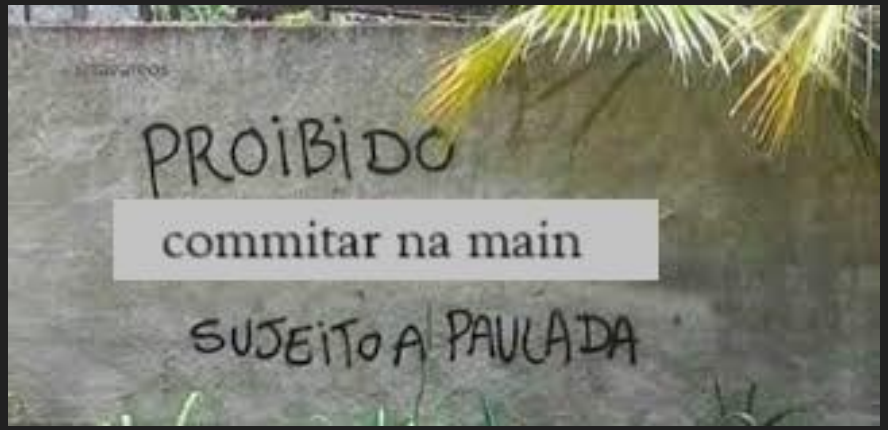
Commits colaborativos

- Para criar um commit colaborativo com co-autores no Git, você pode usar o seguinte padrão na mensagem de commit:

```
git commit -m "Mensagem do commit


Co-authored-by: Nome do Coautor <email@exemplo.com>"
```

Por que usar diferentes branches?



Como trazer modificações do remoto para o local

- Modifique o arquivo de README diretamente pelo GitHub, na branch nova.
- Faça o commit das alterações.
- No terminal em seu computador rode o comando:

```
git pull origin atualizando-readme
```

Abrindo um pull request

- Em seu repositório do GitHub, abra um novo pull request, para solicitar o merge das alterações da nova branch criada na branch main.

Entendendo Pull Requests

- **Central para colaboração:** Pull requests são a principal forma de colaboração no GitHub.
- **Proposta de alterações:** Ao abrir um pull request, você propõe suas alterações para revisão e integração.
- **Análise e aprovação**: Outros colaboradores analisam e fazem o merge das alterações no seu branch.
- **Exibição de diferenças (diffs):** O pull request mostra as diferenças entre os branches, com alterações destacadas em cores diferentes.
- **Facilidade de revisão:** As adições e remoções são facilmente visualizadas com a ajuda das cores, facilitando a revisão do código.

Aceitando o Pull Request

- Aceite o pull request.
- Realize o merge.
- O merge é o processo de combinar as alterações de dois branches diferentes em um único branch.

Conflitos de merge

- Conflitos de merge ocorrem quando duas ou mais alterações incompatíveis são feitas nas mesmas linhas de código em branches diferentes.

- Os desenvolvedores precisam revisar e resolver as diferenças entre as versões conflitantes antes de concluir o merge.

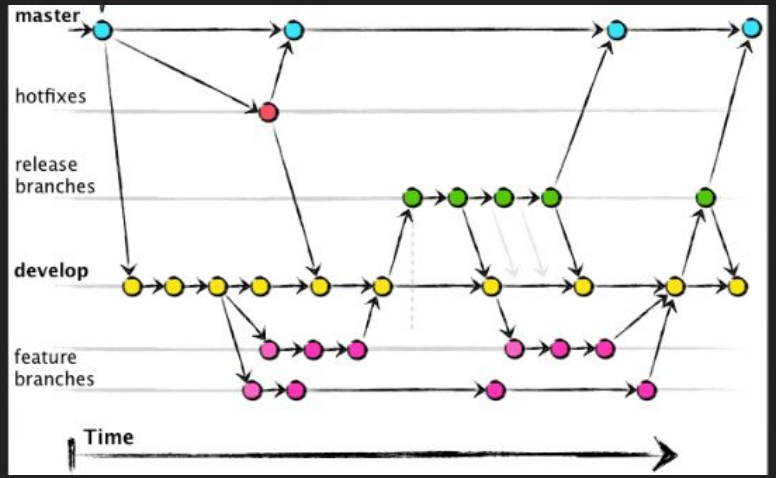
Navegando entre branches

Para voltar para a branch principal use o comando:

```
git checkout main
```

Gitflow

- O GitFlow é uma estratégia de ramificação (branching) para organizar o desenvolvimento de software com Git. Ele define um fluxo de trabalho com diferentes tipos de branches

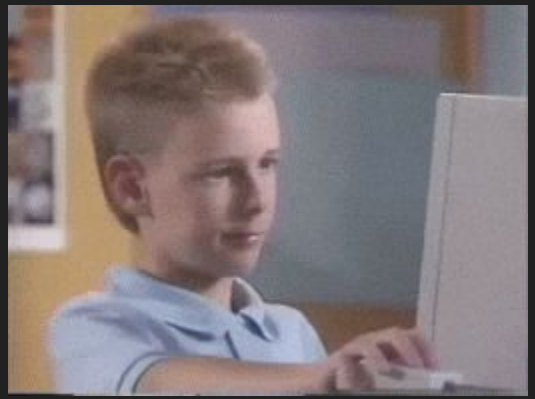


Resumo das principais branches

- **Main:** Contém a versão estável e pronta para produção.
- **Develop:** Onde o código de desenvolvimento é consolidado antes de ser lançado.
- **Feature:** Criado a partir de develop para adicionar novas funcionalidades. Cada nova feature tem seu próprio branch.
- **Release:** Quando o desenvolvimento está quase pronto para produção, cria-se um branch release a partir de develop para ajustes finais.
- **Hotfix:** Usado para corrigir problemas urgentes diretamente na main, sem esperar pelo próximo ciclo de release.

Resumo do fluxo

- O fluxo é: criar novas funcionalidades em feature, preparar uma versão em release, corrigir bugs em hotfix e sempre integrar tudo de volta em main e develop.



Dúvidas?