@fpftech.educacional

FPFtech
Escola Tecnológica

FPFtech
Escola Tecnológica

Versionamento GIT

Controle de versão é uma forma de gerenciar mudanças em arquivos de um projeto (não só código, pode ser qualquer coisa).

Serve para saber o histórico, voltar versões anteriores, trabalhar em equipe sem bagunça.

É como um salvamento inteligente que registra o que mudou e por que mudou.

Projeto_final_v3_finalMesmo_esseSim_finalmente.docx

Versionamento GIT

Controle de versão resolve problemas como:

- *Perda de histórico de alterações.*
- *Dificuldade de trabalhar em equipe.*
- *Erros difíceis de desfazer.*

Controle de Versão = Histórico + Colaboração + Segurança

"é o processo de gerenciar mudanças em arquivos e projetos ao longo do tempo."

Como funciona?

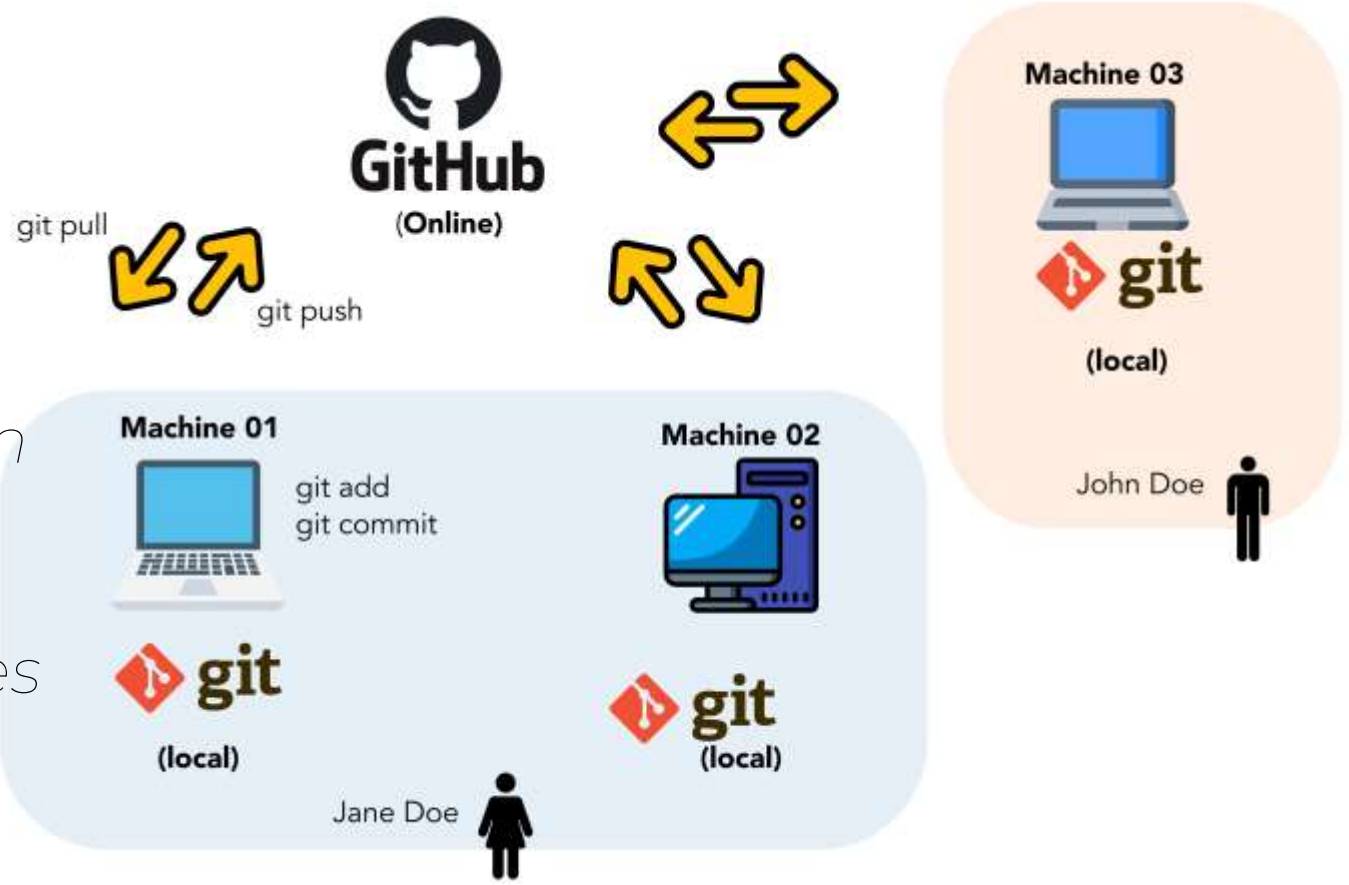
Git trabalha com snapshots, ou seja, ele guarda a foto do projeto a cada commit, não apenas a diferença (diff) como sistemas antigos.

Repositório local: tudo acontece primeiro no seu computador.

Repositório remoto: depois você pode sincronizar com servidores como GitHub, GitLab, Bitbucket.

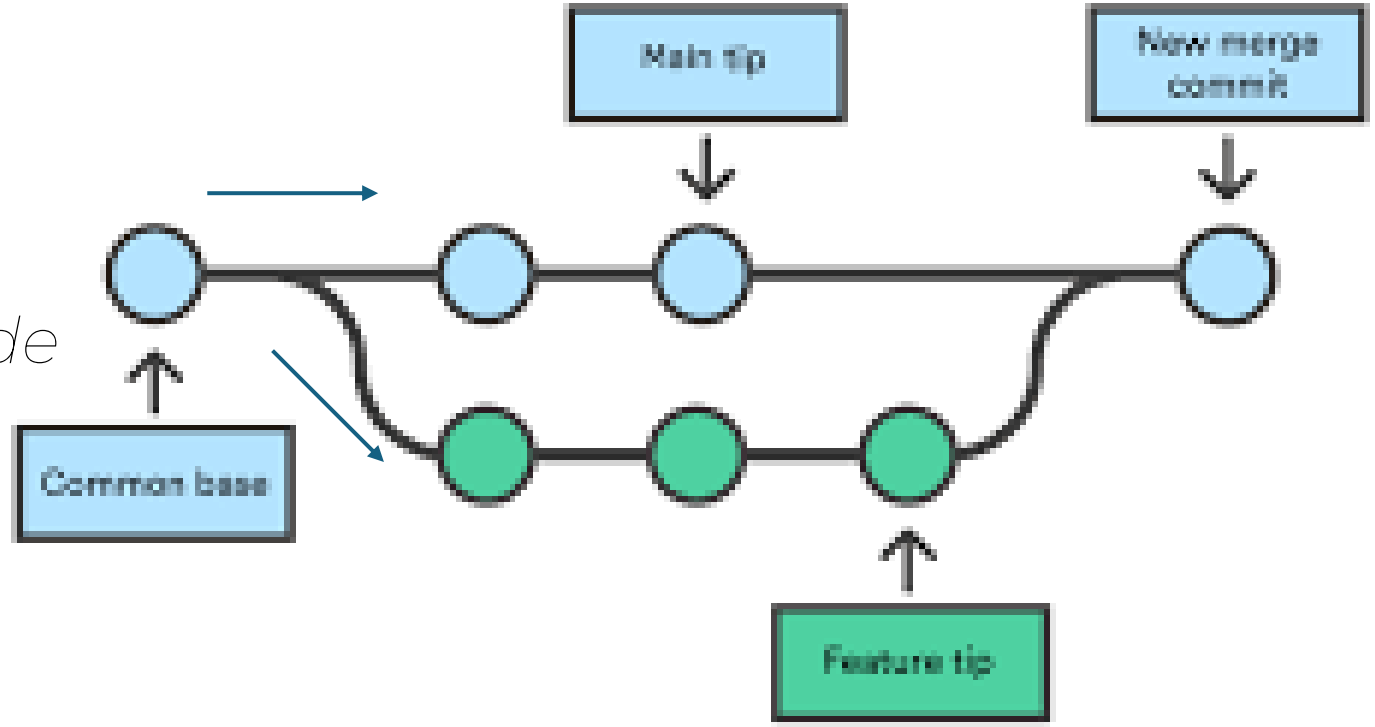
Porque usar GIT?

- *Evita perder trabalho.*
- *Facilita colaboração (várias pessoas trabalhando no mesmo projeto).*
- *Histórico de decisões: saber quem mudou o quê e por quê.*
- *Permite testes de ideias (branches para testar coisas novas sem quebrar o projeto principal).*



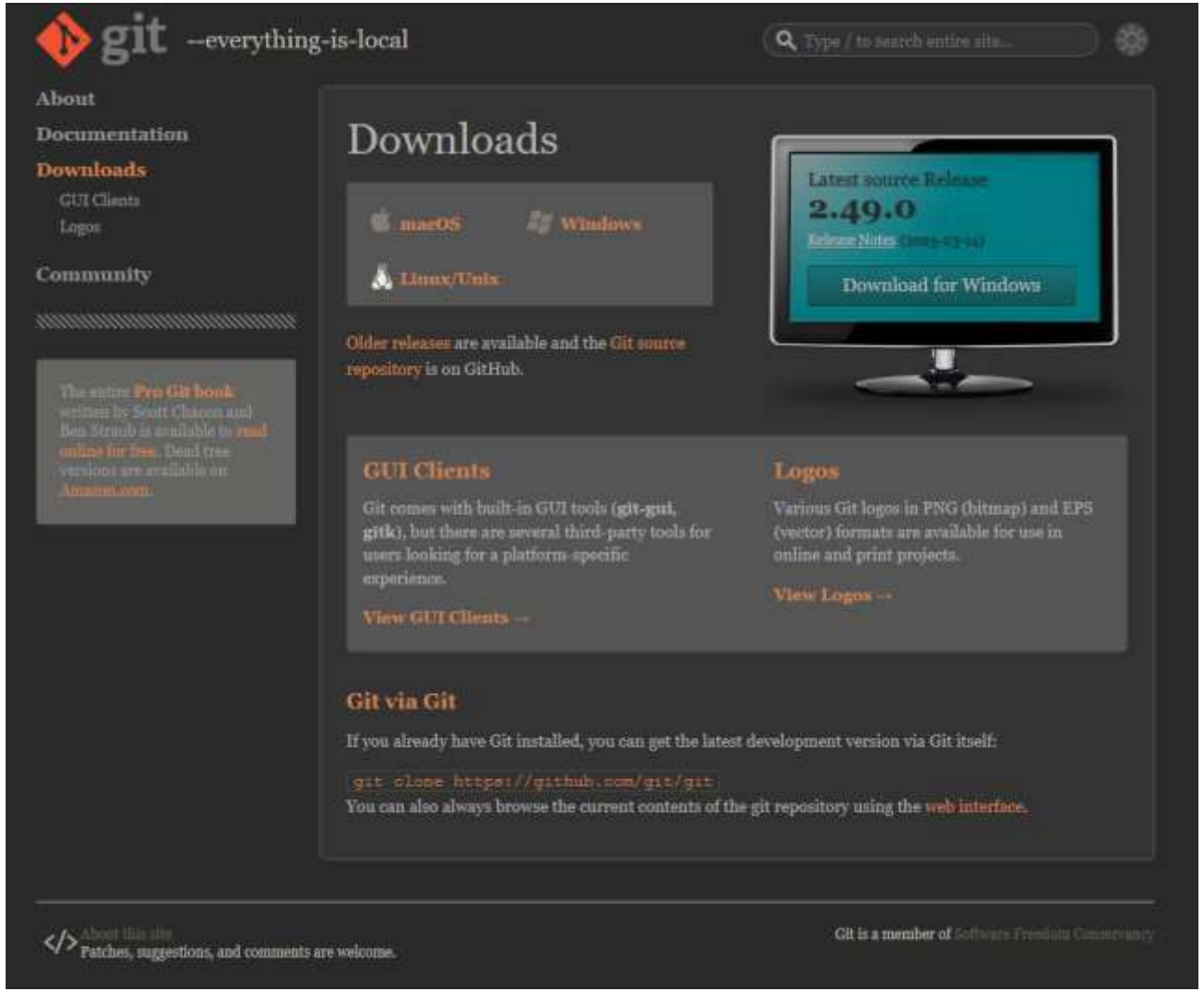
Porque usar GIT?

- *Commit = Registrar uma mudança.*
- *Branch = Linha paralela de desenvolvimento.*
- *Merge = Unir as linhas de desenvolvimento.*



Instalação

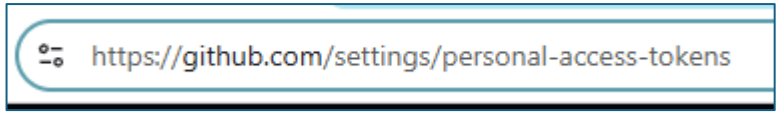
- *Windows: instalar o Git Bash.*
- *Linux: sudo apt install git*
- *Mac: brew install git*



Instalação

1 – Criar conta github

2 – Acessar url "personal-acess-tokens"



3 – Criar novo token, clicar pra copiar.

4 – No pycharm > terminal > "git init"

(criar pasta .git no seu projeto)

```bash
git config --global user.name "Seu Nome"
git config --global user.email "seuemail@example.com"
```

Instalação

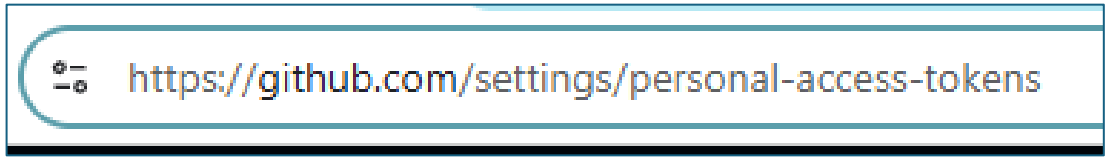
1 – Criar conta github

2 – Acessar url "personal-acess-tokens"

3 – Criar novo token, clicar pra copiar.

4 – Criar projeto novo vazio (nomear).

5 – No terminal do computador, verificar se suas credenciais estão ok




```bash
git config --global user.name "Seu Nome"
git config --global user.email "seuemail@example.com"
```

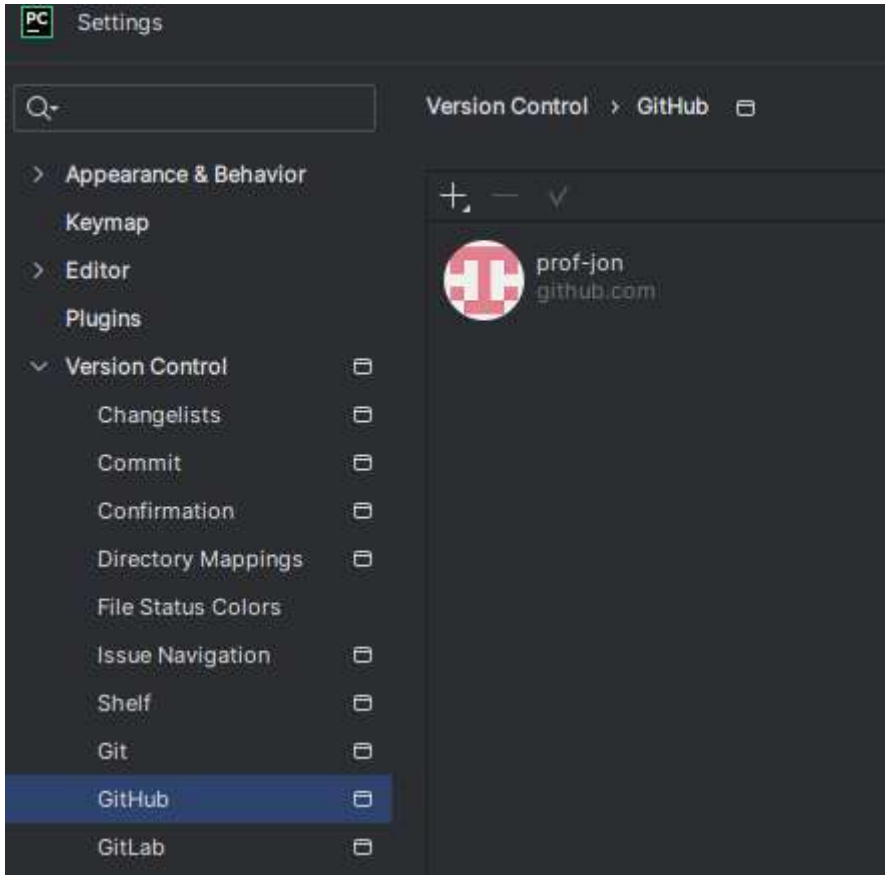
```
git config user.name "Your Name Here"
git config user.email your@email.example
```

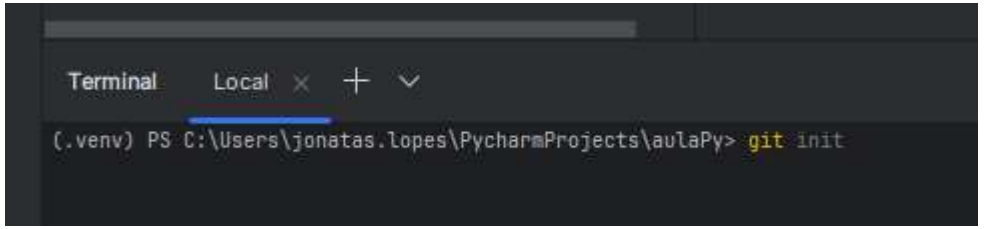
Instalação

6 – No pycharm, adicionar conta github (colocar token do passo 3)

7 – No terminal (pycharm) > digitar “git init”

(isso vai criar a nova pasta local do .git no projeto)

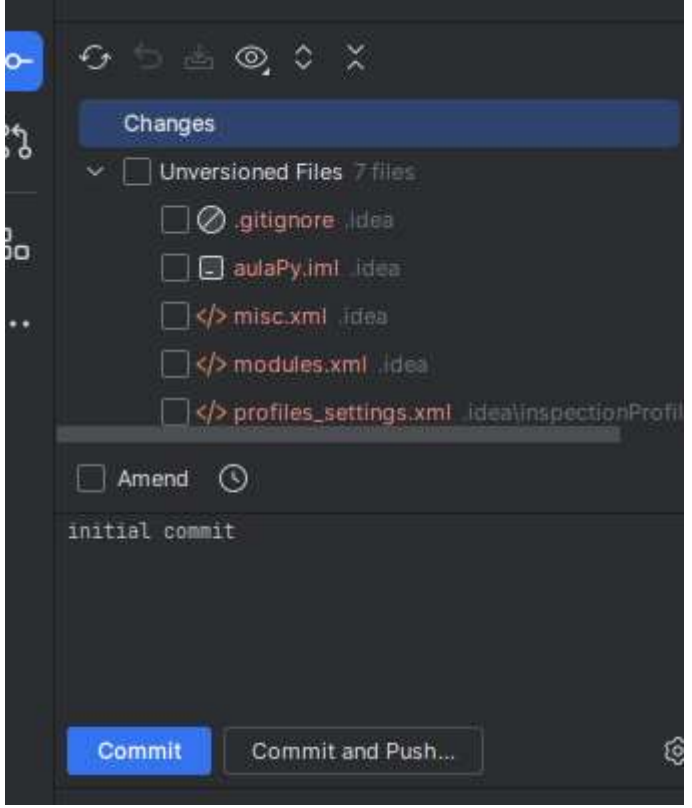




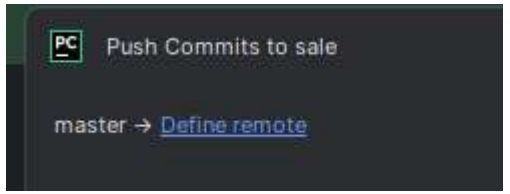
Instalação e uso

8 – fazer alterações no projeto, escrever mensagem de commit e commitar.

**agora está no git local*



9 – configurar o remoto colocando a url do projeto vazio github



10 – fazer push para o remoto. Pronto!

FPFtech
Escola Tecnológica

INSTALE O GIT

GitHub fornece clientes desktop que incluem uma interface gráfica para as ações mais comuns em um repositório e atualiza automaticamente para a linha de comando do Git para cenários avançados.

GitHub para Windows
https://windows.github.com

GitHub para Mac
https://mac.github.com

Distribuições do Git para Linux e sistemas POSIX são disponíveis no site oficial do Git SCM.

Git para todas plataformas
http://git-scm.com

FPFtech
Escola Tecnológica

CONFIGURE A FERRAMENTA

Configure informações de usuário para todos os repositórios locais

```
$ git config --global user.name "[nome]"
```

Configura o nome que você quer ligado as suas transações de commit

```
$ git config --global user.email "[endereco-de-email]"
```

Configura o email que você quer ligado as suas transações de commit

```
$ git config --global color.ui auto
```

Configura o email que você quer ligado as suas transações de commit

FPFtech
Escola Tecnológica

CRIE REPOSITÓRIOS

Inicie um novo repositório ou obtenha de uma URL existente

```
$ git init [nome-do-projeto]
```

Cria um novo repositório local com um nome específico

```
$ git clone [url]
```

Baixa um projeto e seu histórico de versão inteiro

FAÇA MUDANÇAS

Revise edições e crie uma transação de commit

`$ git status`
Lista todos os arquivos novos ou modificados para serem commitados
`$ git diff`
Mostra diferenças no arquivo que não foram realizadas
`$ git add [arquivo]`
Faz o snapshot de um arquivo na preparação para versionamento
`$ git diff --staged`
Mostra a diferença entre arquivos selecionados e a suas últimas versões
`$ git reset [arquivo]`
Deseleciona o arquivo, mas preserva seu conteúdo
`$ git commit -m "[mensagem descritiva]"`
Grava o snapshot permanentemente do arquivo no histórico de versão

MUDANÇAS EM GRUPO

Nomeie uma série de commits e combine os esforços completos

$ git branch
Lista todos os branches locais no repositório atual
$ git branch [nome-do-branch]
Cria um novo branch
$ git checkout [nome-do-branch]
Muda para o branch específico e atualiza o diretório de trabalho
$ git merge [branch]
Combina o histórico do branch específico com o branch atual
$ git branch -d [nome-do-branch]
Exclui o branch específico

FPFtech
Escola Tecnológica

REFATORE NOMES DOS ARQUIVOS

Mude e remova os arquivos versionados

`$ git rm [arquivo]`
Remove o arquivo do diretório de trabalho e o seleciona para remoção
`$ git rm --cached [arquivo]`
Remove o arquivo do controle de versão mas preserva o arquivo localmente
`$ git mv [arquivo-original] [arquivo-renomeado]`
Muda o nome do arquivo e o seleciona para o commit

FPFtech
Escola Tecnológica

SUPRIMA O RASTREAMENTO

Exclua arquivos e diretórios temporários

```
*.log
build/
temp-*
```

Um arquivo de texto chamado `.gitignore` suprime o versionamento acidental de arquivos e diretórios correspondentes aos padrões especificados

```
$ git ls-files --other --ignored --exclude-standard
```

Lista todos os arquivos ignorados neste projeto

FPFtech
Escola Tecnológica

SALVE FRAGMENTOS

Arquive e restaure mudanças incompletas

$ git stash
Armazena temporariamente todos os arquivos rastreados modificados
$ git stash pop
Restaura os arquivos recentes em stash
$ git stash list
Lista todos os conjuntos de alterações em stash
$ git stash drop
Descarta os conjuntos de alterações mais recentes em stash

FPFtech
Escola Tecnológica

REVISE HISTÓRICO

Navegue e inspecione a evolução dos arquivos do projeto

`$ git log`
Lista o histórico de versões para o branch atual
`$ git log --follow [arquivo]`
Lista o histórico de versões para um arquivo, incluindo mudanças de nome
`$ git diff [primerio-branch]...[segundo-branch]`
Mostra a diferença de conteúdo entre dois branches
`$ git show [commit]`
Retorna mudanças de metadata e conteúdo para o commit especificado

FPFtech
Escola Tecnológica

DESFAÇA COMMITS

Apague enganos e crie um histórico substituto

```
$ git reset [commit]
```

Desfaz todos os commits depois de `[commit]`, preservando mudanças locais

```
$ git reset --hard [commit]
```

Descarta todo histórico e mudanças para o commit especificado

FPFtech
Escola Tecnológica

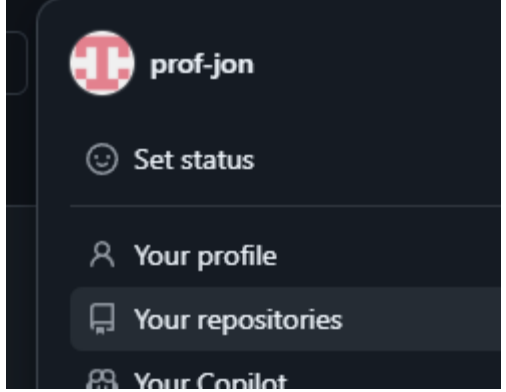
SINCRONIZE MUDANÇAS

Registre um marcador de repositório e troque o histórico de versão

$ git fetch [marcador]
Baixe todo o histórico de um marcador de repositório
$ git merge [marcador]/[branch]
Combina o marcador do branch no branch local
$ git push [alias] [branch]
Envia todos os commits do branch local para o GitHub
$ git pull
Baixa o histórico e incorpora as mudanças

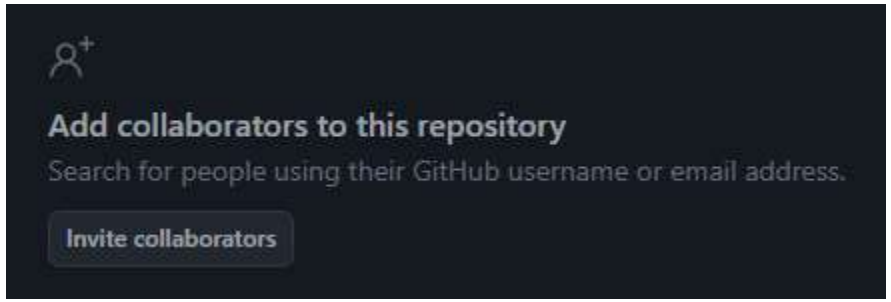
Em pares

1 – escolham 1 projeto de vcs



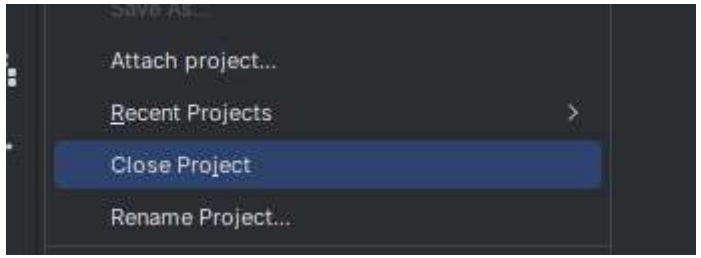
2 – O dono vai na aba "repositórios"

acessa o projeto a compartilhar.



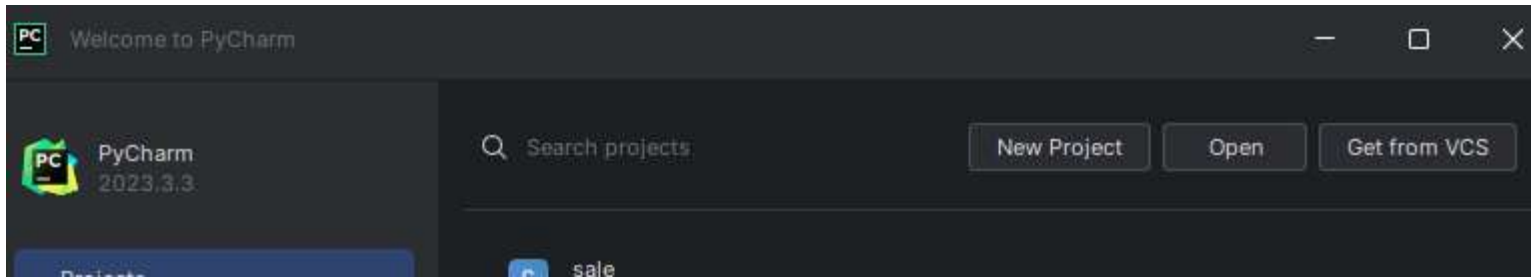
3 – O outro pega a url do git. Vai em

"fechar projeto" do pycharm.



4 –Painel novo projeto. "Get from VCS"

Coloca a url lá. Teste Branch.



FPFtech
Escola Tecnológica

Obrigado!

FPFtech
Escola Tecnológica

@fpftech.educacional