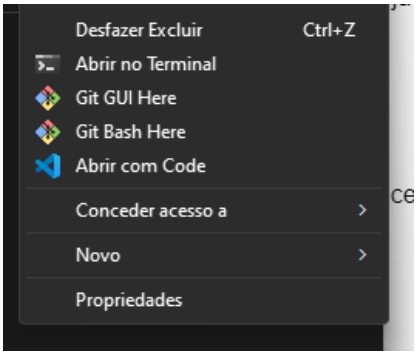
Introdução ao git

O git é uma ferramenta de versionamento de código. O git trabalha com repositórios, eles são lugares de trabalho onde você irá criar versões do seu código.

Para instalar, use o link: https://git-scm.com/



Para rodar os comandos, sugiro usar o git bash. Caso esteja no windows, para acessá-lo dê clique direito no lugar que deseja trabalhar e depois clique em “Git Bash Here”.

Principais comandos

git --version
Vê a versão instalada do git. (Consegue ter certeza que instalou o git corretamente)

git init
Cria um repositório de código

git add pasta_dos_arquivos_ou_arquivos
Este comando adiciona arquivos para serem 'commitados'. Commits são uma forma de subir uma nova versão.

git commit -m "texto falando o que você alterou"
Esse comando cria um novo commit

git log
Mostra os commits realizados e seus ids

git reset --hard id_do_commit
Volta para versão informada no id_do_commit (cuidado, deve ser usado com cautela, ações feitas aqui não podem retroceder)

Todos esses comandos acima realizam versões na própria máquina, todavia, como alguém tem acesso ao meu código? Por isso existe o github. Gosto de pensar que ele funciona como uma rede social de programadores, onde interagem e trocam código. (OBS: também é muito usado como portifólio). Agora segue alguns comandos para interagir com o github. (não se esqueça de criar uma conta em https://github.com/).

git config --global user.name "Fulano de Tal"
Coloque o seu nickname do github

git config --global user.email fulanodetal@exemplo.b
coloque o seu email do github

git clone site_do_repositorio
Clona repositórios

git pull origin nome_da_branch (mas o que raios é uma branch? Calma, já chegamos lá)
Pega mudanças realizadas numa branch

git push origin nome_da_branch
Envia mudanças para uma branch (sempre uma boa prática usar o comando git pull antes de um git push)

Isso faz resolver parte do nosso problema, repare que o git pull ou git push pode entrar em conflito. Nesses casos os conflitos devem ser resolvidos para que o comando possa funcionar. Um conflito ocorre quando o git entende que duas pessoas mexeram numa mesma parte de um código, portanto alguém deve analisar para entender qual parte deve seguir para frente. Mas aí segue um problema, e se um desenvolvedor quiser criar um botão numa página e um outro desenvolvedor quiser mudar a cor do fundo. Ambos podem se atrapalhar no "git flow" nesse caso, então como funciona para que ambos os desenvolvedores tenham uma versão estável de código sem que um atrapalhe o outro? Com branchs, branchs criam outra linha de versão (normalmente copiando a linha de versão da branch principal (da main/master) ou da branch de desenvolvimento (a develop)). Agora, os principais comandos para trabalhar com branchs.

git checkout -b nome_branch_nova
Cria uma branch nova com base na que você está agora.

git branch -a
Vê branchs já criadas

git checkout nome_branch
Muda de branch

git merge branch_a_se_juntar
Junta uma branch com a que você está agora