**Módulo 3: Metodologias de Desenvolvimento – conceitos essenciais + passo a passo dos vídeos**

**3.1) Introdução ao Git: O Git é um sistema de versão e o GitHub, um sistema em nuvem que fornece esse serviço.**

**O que é GIT?**

**Git** é um sistema de controle de versão que registra as mudanças feitas em um arquivo ou conjunto de arquivos ao longo do tempo, possibilitando recuperar ou acessar versões específicas quando desejarmos. É ideal para desenvolver projetos onde diversas pessoas podem contribuir simultaneamente, buscando evitar que suas alterações sejam sobrescritas.

**O que é GitHub?**

O **GitHub** é uma plataforma criada para hospedar e controlar versões de projetos de programação na nuvem de forma gratuita, ou seja, você pode usá-lo para guardar seus projetos pessoais ou open source (projetos abertos a todos), possibilitando uma colaboração maior entre equipes de uma forma eficiente.

**3.2) Instalação do Git:** já fizemos.

Quando falarmos em terminal, vamos trabalhar no GitBash

Os comandos começam com “**git**” na frente deles.

**git –version (git menos menos version)**: vai te mostrar a versão do gitBash instalada no computador.

**3.3) O que é GitHub (escrevi lá em cima)**

Uma vez criada a conta no GitHub, você pode começar a criar repositórios (lugar em que arquivos de projetos serão armazenados e é através do qual você poderá rastreá-los). Um projeto corresponde a um repositório e, no GitHub, você pode ter quantos projetos/repositórios quiser. Um repositório no GitHub é chamado de **repositório remoto porque estão na nuvem**. Em paralelo, cada pessoa terá uma cópia desse repositório no seu computador, que chamamos de **repositórios locais**. Para que o git funcione bem, é necessário criar um link entre o local e o remoto, pra que você possa manter os backups dos seus arquivos locais conectados aos arquivos da nuvem.

=>mostrou como criar um repositório no GitHub: clicar no sinal de + no canto superior direito -> criar novo repositório. Configurar como quiser. Nomear repositório com o nome do projeto. Não pode ter mais de um repositório com o mesmo nome.

**3.4)** **Criando seu primeiro repositório (passo a passo): Entrar no Git Bash.**

**3.4.1) git cd pastaDesejada.** Dando um **pwd,** o terminal mostra em que pasta você tá**.**

**3.4.2) git init** – vai mostrarnoconsole a mensagem de que o repositório vazio foi inicializado. Esse comando criou um repositório local na pasta onde estão os arquivos. Agora você precisa mostrar quais arquivos deseja adicionar.

OBS: um repositório, seja local ou remoto, é um armazém de arquivos, como uma prateleira onde seus arquivos serão armazenados em pequenos pacotes que chamamos de commits. Esses pacotes permitem que você acompanhe as mudanças que você está fazendo no arquivo, já que cada um deles tem uma data de criação e o autor. Os commits serão o nosso histórico de mudanças que foram feitas no projeto. É por isso que, antes de mais nada, você precisa dizer ao repositório quem você é (nome da conta no GitHub), porque é com essa informação que o git pode acompanhar as mudanças feitas, para que vc possa adicionar a sua identidade ao repositório. O que preciso fazer?

**3.4.3)** **git config – global user.name “meuNomeDeUsuárioAqui” (e Enter)**: não vai mostrar nenhum texto adicional no terminal. Pra ver se meu nome foi add corretamente, digitar **git config user.name (e Enter)** e vc deve ver no terminal o texto que você digitou.

**3.4.4) git config –global user.email “meuemail@email.com” + Enter:** não vai mostrar nenhum texto adicional no terminal. Pra ver se meu nome foi add corretamente, digitar **git config user.email + Enter** (vc deve ver no terminal o email que você digitou antes). Agora seu repositório sabe quem você é.

**3.4.5)** Criar repositório no GitHub, caso não o tenha feito ainda.

**3.4.6)** **git remote add origin** seguido pela url do seu repositório que copiamos no site do GitHub. Pra verificar se deu tudo certo, digitar git remote -v. o gitbash vai responder “origin (seguida pela) url” (HTTP).

**ATALHO:** pra colar algo no gitBash: shift insert

**3.4.7)** **git remote -v**: pra ver se o git remote add origin deu certo. O gitbash vai trazer duas informações: uma com o final fetch (de onde eu vou trazer as informações) e outra como final push (pra onde eu vou enviar as informações).

**3.5) Confirmando arquivos:** Quando trabalhamos com arquivos de forma contínua, é comum termos o costume de salvá-los automaticamente ou ctrl+s. Quando usamos o git, o ato de confirmar as modificações de um arquivo é de maior importância, já que é isso que nos permite estabelecer um ponto de controle. Agora chegou a hora de confirmarmos oficialmente que os arquivos adicionados ao repositório serão um ponto na linha do tempo para o qual podemos retornar se precisar. Vamos começar a fazer commits. O commit é uma confirmação de que informamos o repositório que os arquivos que estamos adicionando sejam um pequeno pacote de adições ou modificações, que terão uma marca definitiva em um determinado período de tempo e serão assinados por um autor. Os commits geram um ponto cronológico na linha do tempo do projeto, o que nos permite identificar o status do projeto até aquele momento específico, além de poder voltar ao mesmo se quiser voltar em algum momento.

Passo a passo:

**3.5.1)** Depois de fazer as alterações necessárias no seu projeto, você precisar dar um **git status** pra verificar a situação. Se aparecer algum arquivo em vermelho, digitar **git add . (pra adicionar ou atualizar todos os arquivos) ou git add nomeDoArquivo.extensão.** Dar um **git status** novamente e o arquivo em vermelho passa a aparecer na cor verde. Refazer esse processo sempre que alterar ou criar algum arquivo.

**3.5.2)** Quando, ao dar git status, aparecerem todos os arquivos em verde, está na hora de fazermos um commit: **git commit -m** **“mensagem que quero adicionar”.**

**3.5.3)** **git status** novamente e vai aparecer uma mensagem com “on branch master nothing to commit, working tree clean” (então está tudo certo).

**3.5.4)** Se modificar ou adicionar algum arquivo, dar **git status** => **git add nomeArquivo**(ou **git add . (ponto)** pra adicionar todos os arquivos) Depois, criar um commit => **git commit -m** **“mensagem que quero adicionar”.**

**3.6) Adicionando arquivos: mesmo passo a passo explicado acima.**

**3.7) Subindo arquivos:**

OBS: para fazer upload dos arquivos, eles têm que estarem commitados!

**3.7.1)** **git push origin master**: dar esse comando e observar a resposta do git bash.

**3.7.2)** Se você alterou algum arquivo, vc deve dar **git status** e ver a situação. **git add nomeArquivo.extensão** pra adicionar esse arquivo modificado ou **git add .** pra adicionar todos os arquivos. Depois, fazer um **git commit -m “mensagem aqui”**. git status novamente pra verificar a situação.

**3.7.3)** **git push origin master** para enviar ao GitHub.

**3.8) Baixando arquivos**

**3.8.1)** No GitHub, clicar no botão “**clone or download**” (botão verde) e **copiar a URL em HTTP**.

**3.8.2)** estar em uma pasta de fácil acesso, como uma pasta dentro do desktop. (entrar com o comando **cd nomeDaPasta** pelo gitbash.

**3.8.3) git clone URLaqui** => **ls** pra ver o que tem dentro da pasta => **cd nomeDaPastaCriadaAgora.**

O QUE FAZER QUANDO HOUVEREM ATUALIZAÇÕES NOS ARQUIVOS/COMO MANTER OS ARQUIVOS SINCRONIZADOS: **git pull origin master** para baixar só as atualizações do projeto.