**Git**

Primeiramente devemos saber que Git e GitHub e diferente



O que é Git?

Basicamente é um software de controle de versão (VCS)

*O que é GitHub?*

*Uma rede social para programadores (Próxima aula)*

É extremamente necessário que todo programador profissional tenha um repositório no GitHub e um perfil para ser seguido.

Como já sabemos o Git é um software de controle de versão (VCS), logo podemos dizer que o Git é uma ferramenta de versionamento de software.

A versão de um sistema é extremamente importante para um programador, pois ele deve fazer alterações no sistema e salvar seu projeto até aquele momento, para que quando estiver programando melhorias, se algo der errado aquela versão que está salva não está com aqueles erros. Também serve para que quando houver uma nova melhoria ela seja tratada como uma nova versão do sistema.

Uma prática bem interessante é salvar seus sistemas em .zip, para cada atualização do sistema, ou cada versão um novo .zip deve ser criado.

Exemplo:



Nomes bem engraçados podem surgir durante o processo de desenvolvimento do sistema, principalmente quando o cliente pede pra mudar tudo em determinado momento do desenvolvimento do sistema. Imagina se nessa situação o cliente pede par voltar com uma imagem que antes estava.

É muito importante que você crie esses zips do seu sistema, mas que você também faça backup desses arquivos. Porém, imagine que você trabalha com várias pessoas em um mesmo projeto e há um local onde todos salvam seus arquivos, como por exemplo o google drive ou OneDrive, tem grande chance de que em algum momento isso dê algum problema. E para que isso seja melhor controlado ou organizado você precisa de um VCS (Software de Controle de Versão).

Um VCS vai controlar tudo pra você, ele salvará as mudanças, mas não perderá o formato antigo do arquivo ou mesmo algo arquivo que antes existia em seu sistema e já não existe mais.

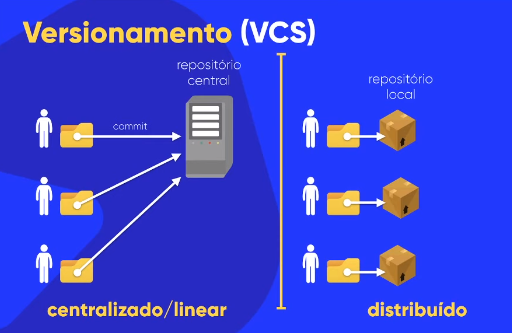
O versionamento funciona como se fosse uma “**máquina do tempo**”.

O Git é um VCS e é o que vamos usar em nosso curso.

Quando nos pegamos uma pasta de arquivos ou mudamos e algo em nosso projeto e mandamos pro repositório central, nós estamos fazendo um commit. Nesse processo várias pessoas envolvidas no projeto também podem mandar arquivos para o repositório central ou fazer modificações nele. Esse repositório controla as versões de acordo com as mudanças que são feitas pelos desenvolvedores desse sistema. Esse tipo de Versionamento e chamado de “**centralizado ou linear**”.

Através da evolução do versionamento centralizado ou linear surgiu o “**versionamento distribuído”.**

Nesse caso quando o commit é feito ele não vai diretamente para o repositório central, mas ele vai pra um repositório dentro do computador do desenvolvedor. Esse pode ser chamado também de Versionamento Local.



Sempre que for criar um novo projeto crie um versionamento para ele, crie a sua máquina do tempo, para que se algo der errado em alguma parte do projeto você tenha as versões anteriores salvas em algum lugar.

Mas aí vem a seguinte pergunta:

Se o repositório está no meu computador local e preciso desenvolver com minha equipe, como fazer pra que eles vejam o que eu programo e como posso ver o que eles programam?

Para isso vamos trabalhar com **repositório remoto**.

Para trabalhar com esse tipo de repositório além do commit também temos que dar um ‘**push**’, que basicamente você pegar o Versionamento que você tem na sua máquina e jogar para o repositório remoto.

Os VCS distribuídos são os mais evoluídos de todos e é o que vamos ver nesse curso.

Exemplos de VCS centralizado ou Linear

* CA Software Change Manager (CCC)
* Source Code Control System (SCCS)
* Clear Case

Exemplos de VCS centralizado ou Linear

* Mercurial
* Bazaar
* Code Co-op
* GNU arch
* Monotone
* Fossil
* **BitKeeper (Funciona atualmente)**
* **Git (Funciona atualmente)**

*Obs. Todos os marcados sem negrito não funcionam mais.*

**Vantagens de se utilizar VCS**

* **Controle de Histórico**

1. Voltar há uma versão anterior
2. Ver quem mudou seu código
3. Ver quando algo foi alterado

* **Trabalho em Equipe**
* **Ramificações do projeto**

1. Separar por equipe ou colaborador cada uma das funções do sistema, uma pessoa fica por conta do designer, outra por conta do front-end, outra por conta do back-end, e depois de tudo feito conseguir juntar tudo isso em um projeto final.

* Segurança
* Organização