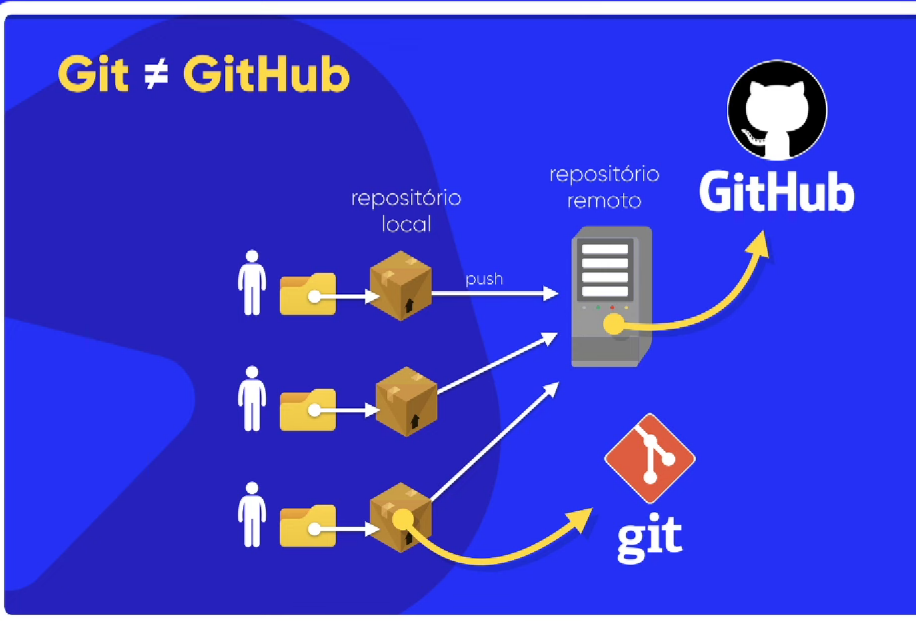
**Aula 01**

**Git e GitHub são coisas diferentes.**

Git é um software de controle de versão. E o GitHub é uma plataforma de redes sociais para programadores.

**Explicando Git:**

Git é um VSC “software de controle de versão”, resumindo em uma palavra, versionamento/versões, e programador sempre lida com diferentes versões de algo, não só quem programa mais também quem utiliza softwares.

**Versionamento**

Existem 2 tipos de versionamento: o centralizado/linear, que é forma de guardar arquivos em servidor, de forma centralizada, onde você divide seus arquivos diretamente com as outras pessoas relacionadas a um projeto. Existe também o distribuído, onde você tem sua partição de arquivos ao mesmo tempo que consegue dividir eles com os demais.

A imagem mostra um exemplo de versionamento distribuído.

**Principais Vantagens de usar esse tipo de software:**

* Controle de histórico
* Trabalho em equipe
* Ramificação de projeto
* Segurança
* Organização

**Aula 02**

GitHub, já sabemos que é uma rede social de programados, o uso maioral dela é guardar códigos, mostrar códigos prontos para aprimorar chances de emprego, funcionar diretamente como repositório remoto para empresas que usam GitHub.

**Vantagens:**

* Repositório ilimitado
* Hospedagem de código-fonte
* Características de rede social
* GitHub Pages integrado
* Colaboração
* Forks

Só existe o GitHub? Além dele temos:

* GitHub
* GitLab
* Bitbucket
* PHABRICATOR
* Gogs
* Kallithea

**Aula 03**

Falando um pouco da história...

**CVS: SVN:**  **BitKeeper**

1985 2000 2000

Centralizado Centralizado Distribuído

Open Source Open Source Proprietário

Mais Popular Ativo até hoje Versão comunidade

Alguns problemas CVS-Like CVS-free

**BitKeeper BitKeeper** **Git**

2004 2005 Distribuido

Source Puller Nova licença Open Source

Engenharia Reversa Acesso a metadados feito em 10 dias

Recursos destravados Só na versão comercial Perfomance

O quer dizer **Git**?

Três letras do alfabeto

Teimoso, cabeça-dura, pensa que sempre está certo

Global information tracker

Goddamn idiotic truckload of sh#t

Aqui claramente são brincadeiras que o criador usava para se referir.

O **Git** tem esse nome por que era o que tinha.

E o GitHub?

**GitHub GitHub**

2008 2018

Proprietário Adquirido pela Microsoft

Hospedagem de código US$ 7.5 BI 🫥

Baseado em Git Operação independente

Em 2011: Ultrapassou **SourceForge** 2020: GitHub compra a npm

2018: Maior ataque dos **DDoS** da história

**Aula 04**

O que preciso instalar?

Navegador -> já utilizo o Chrome, então vou me manter nele.

Editor de códigos -> Já tenho o Visual Code, não tenho conhecimento mas irei me manter nele.

**Git** e **GitHub Desktop** -> hehe

Bom, aqui foi só isso kkkkk, de aproveitamento dessa aula foi apenas algumas alterações no Visual Code que eu só foi instalado algumas extensões recomendadas até o momento.

**Aula 05**

Aula totalmente pratica, até o momento consegui criar meu primeiro repositório 🫡, me pareceu passos simples, primeiro a pasta é criada de forma local “Git” e depois tenho a opção de jogar essa pasta para o GitHub de forma publica ou de forma privada.

Pelo Visual Code, é possível gerenciar o README, esse arquivo vai possibilizar personalizar o seu repositório do GitHub, Adicionar um comentário, fazer uma anotação ou deixar explicado como o seu repositório funciona e o que tem nele.

Anotações de como estou entendendo:

O **commit** serve para aplicar alterações no Git.

**Push** serve para mandar ou puxar essas alterações para o GitHub.

**Pull** pega o que está no GitHub “que foi alterado de forma Remota” e trazer para o Git, local.

Isso tudo vai interagir com meu repositório Local, com o GitHub Desktop, GitHub remoto “acesso pelo site” e pelo visual Code 🤯.

Alguns Atalhos:

CRTL + Enter= commitar depois de digitar o titulo

CTRL + P= Push logo após commitar

Aula Top hehe 🧠

**Aula 06**

Instalando o GitHub no Linux (não uso, mas não é por isso que irei deixar de assistir a aula.

Basicamente o mesmo processo usado para o Windows, vi a aula completa por comprometimento ao conteúdo, não pretendo usar Linux, caso precise já tenho uma Base, e já sei em que aula devo voltar caso precise de ajuda.

Pra ser sincero, revisão da aula 05 no Linux, pra quem tem Linux é uma aula fundamental, pra quem já viu o processo no Windows pode até pular.

**Aula 07**

Clonando repositório:

Clonagem de repositório é algo bem simples de ser feito e muito útil, consigo ir lá no perfil publico de uma pessoa, como o Gustavo Guanabara, e clonar os arquivos públicos que essa pessoa disponibiliza, quando um repositório é clonado todos os arquivos que estão lá vem pro dispositivo de forma local, é como fazer downloads de arquivos na internet ou fazer um **pull** dos seus arquivos que estão na nuvem.

Inclusive já clonei o repositório de HTML e CSS do guanabara por que vai ser minha próxima fonte de estudos.

**Aula 08**

Versionando Projetos

Tudo que foi feito durante a aula eu já tinha feito no intervalo da aula anterior para essa, peguei exercícios e anotações que foram criados durante o estudo de algoritmos e consegui commitar e fazer o push.

É falado sobre Gitignore, posso selecionar um arquivo para não commitar, ele vai fazer a leitura se tem mais arquivos iguais e jogar todos esses arquivos para pasta gitignore, tudo que estiver nesta pasta não vai versionar nem adicionar no GitHub.

Até o momento está bem claro como é feito o funcionamento entre Git e GitHub, versionamento e também sobre o nome dos comandos usados.

Ele reexplica sobre a parte de clonar, puxar esses arquivos que está de na Nuvem do GitHub para o Dispositivo Local, deixando claro que clonar não é jogar o repositório de outra pessoa, e exatamente como foi explicado, ele baixa de forma local.

Se quero puxar o projeto pro perfil existe o Fork, ele menciona mais não entrou a fundo ainda.

**Aula 09**

Aula sobre Issues

Issues é um problema que o publico do GitHub tem, não conseguiu identificar e resolver, e assim publica no GitHub para procurar pessoas que pode ajuda a resolver aquele problema, aquele Issues.

Essa lacuna só aparece se o problema ainda não foi resolvido, então se alguém publica um Issues, esse é resolvido, ele deixa de ser um.

Na aba de Issues temos a aba Open” abertos”, e Closed” fechados”, os abertos ainda não foram resolvidos e os fechados foram abertos e resolvidos deixando de ser Issues, nas verdades tem inúmeros motivos para fechar kkk.

Dicas do Guanabara: Sempre que tiver um problema, não crie diretamente uma Issue, procure no GitHub, dentro do projeto, se alguém teve a mesma questão, depois de procurar e confirmar que não vai criar uma “issue duplicada”, aí sim pode criar a sua e procurar por ajuda.

É ensinado a criar e reponder Issues, fechar Issues, criar e responder com imagens e adicionar alguns efeitos de formatação.

Coisa mais simples que podem ser feitas usando você mesmo a rede social, e prestando atenção a cada detalhe.

É possível criar assinatura padrão, criar uma mensagem que vai ser aplicada com poucos clicks ao invés de sempre ficar digitando a mesma coisa.