7.1 Entendendo o GIT

00:00:05 - 00:00:28
Bom, a nossa aula extra vai ser sobre Git e Git Hub, que é algo que eu sei que muitos de vocês em pedindo ou tem interesse em aprender, até porque é muito importante a gente ter esse conhecimento porque é algo que a gente vai usar muito no nosso dia a dia, independente se você trabalha com Front ou se trabalha com Back, é algo que vai ajudar muito vocês a terem esse controle de versão do código em que vocês estão trabalhando, do projeto em que vocês estão trabalhando.

00:00:28 - 00:00:50
Vamos passar um pouco do que era antes,o que a gente chamava de controle de versão.Gente, em comparação com antes, está muito melhor. E para a gente começar eu trouxe essa tirinha, é em Inglês, mas dá para vocês entenderem mais ou menos.

00:00:50 - 00:01:18
O rapaz ali está escrevendo: "Final.doc", um arquivo final ponto doc. E ele passa para o chefe. E volta para fazer o ajuste "Final_rev2.doc"."Revisão 2 ponto doc". E manda de novo para a pessoa aprovar. E depois coloca mais alguns comentários. E ele volta a escrever, colocando mais comentários e vai nesse vai e volta...daqui a pouco ele está com um arquivo desse tamanho. Se eu falar para vocês que era assim que a gente trabalhava.

00:01:18 - 00:01:43
Não estou brincando. Quando a gente utilizava o nosso controle de versão era um controle de versão muito manual, principalmente para Front-end, era o que a gente chamava de fft, que é o protocolo de transferência de arquivos. Esse é um print que eu peguei do FileZilla, que era o programa que a gente utilizava para poder fazer esse gerenciamento de dados, esse controle de versão.E como é que a gente fazia esse controle de versão?

00:01:43 - 00:02:14
Do lado esquerdo, como vocês podem ver aqui, é o lado onde estão os arquivos do nosso computador e o outro lado é o arquivo que tem as informações contidas no servidor. A gente se conectava através do FTP para subir os arquivos da nossa máquina para o repositório do projeto. E quando tínhamos que fazer backup de alguma informação para ter o controle da versão... "Ah, eu vou atualizar index!"

00:02:14 - 00:02:42
A gente escrevia index e botava BKP que é "backup", underline e a data em que estávamos fazendo alteração. E coloca a data de hoje, tal, comenta, depois sobe a index para atualizar. Gente, era assim. Era uma das formas,haviam outras, mas a mais comum era utilizar o FTP e fazer essa substituição de arquivos. Ou então a gente subia, para o site não ficar muito tempo fora do ar, a gente transferia, na verdade, os arquivos para a nossa máquina.

00:02:42 - 00:03:09
Fazia a alteração para BKP, subia tudo com index, tudo certinho, porque já sobrescrevia, subia junto com BKP e a gente fazia as atualizações dessa forma. Olha, não era fácil, porque muitas vezes a gente substituía por erro, a gente não tinha nenhum controle, porque era tudo manual, era um controle totalmente manual. A gente não tinha uma estrutura. Às vezes até conseguia estruturar por pastas e tal, mas olha o trabalho que dá fazer isso tudo, controlar dessa forma.

00:03:09 - 00:03:41
E depois,com o tempo, vieram outras ferramentas que a gente chama de sistema de controle de versão. Tem alguns que tem o nome de SVN, existe um chamado Tortoise e tal. Só que aí veio uma ferramenta, um sistema, que ao mesmo tempo que era uma sistema de controle de versão era também uma forma de gerenciar essas versões. Que é o nosso Git. O Git é um sistema de controle de versão que permite gerenciar diferentes versões de um código fonte.

00:03:41 - 00:04:09
Voltando aqui para esse slide, o Git veio para facilitar a vida da pessoa desenvolvedora porque, como eu falei para vocês, na época a gente controlava a versão dessa forma,ou com outras ferramentas de controle de versão e o Git veio para facilitar esse processo. Porque você imagina fazer tudo de uma forma manual ou, às vezes, usando outras ferramentas que também não eram tão usuais, eu achava até a experiência confusa, na minha opinião.

00:04:09 - 00:04:32
Na minha experiência pessoal trabalhando com outras ferramentas e o Git veio para facilitar o nosso trabalho e garantir que a gente tenha as versões dos nossos códigos de uma forma um pouco mais segura, no sentido de garantir que a versão vai estar lá e quando acontecer alguma coisa a gente consegue pegar aquela versão,ou quando a gente sem querer deleta. Localmente, na nossa máquina, a gente consegue puxar onde está hospedado.

00:04:32 - 00:04:57
E uma facilidade absurda. Como é que se instala o Git? Tem esse link, não vamos fazer o processo de salvação aqui porque depende de máquina para máquina, mas eu coloquei alguns prints aqui para facilitar para vocês no momento que forem fazer o processo de instalação. Não é complexo. É bem de boas, a documentação do Git é bem completa.

00:04:57 - 00:05:25
Por padrão, quando vocês entram nessa URL aqui, vocês vão ver que de acordo com o sistema operacional de vocês ele vai detectar. O site detecta qual é o sistema operacional que vocês estão utilizando. E por exemplo,como eu uso o Mac ele vai aparecer o Mac, mas se você usa o Windows vai aparecer Windows, se você usa Linux vai aparecer Linux, mas se não aparecer nada você vem aqui no guia de downloads e você vai ver que estão os três principais sistema operacionais aqui.

00:05:25 - 00:05:57
"Ah, Aline não apareceu isso aqui lá na página inicial do Git." Você pode clicar na versão que você utilizar e você usa da melhor forma.Eu vou mostrar aqui para vocês, vou voltar aqui os slides só para pegar a URL certinha, para vocês verem como que é. Você vem aqui no download, coloquei Windows, porque a maioria utiliza.E aqui vocês têm todo o passo a passo,
aqui você baixa a última versão do Windows.

00:05:57 - 00:06:25
Você baixa aqui e faz todo o processo de instalação se você tem Linux é a mesma coisa, você tem a instalação por terminal. E se você tem Mac, você tem todo um processo aqui de instalação pela linha de comando. Quando vocês instalarem o Git... Vou até fazer um exemplo para vocês verem, depois de instalarem o Git. Só deixa eu atualizar aqui...

00:06:25 - 00:06:45
Está atualizando...Atualizou.
Se vocês colocarem aqui...Vou dar um "clear" aqui. "Clear" é quando a gente precisa limpar todo o nosso terminal se você der um "git version" você já sabe a versão do Git que está instalado na sua máquina.

00:06:45 - 00:07:06
Quando você fizer todo o processo de instalação do Git é só dar este comando:"git version" "git --version" E vocês vão saber se está instalado ou não. Ok?! Nesse ponto aqui não tem muito mistério. É só você seguirem o passo a passo. Continuando sobre o Git, vamos falar um pouquinho mais dessa coisa de estrutura de repositório, diretório de trabalho, espaço de armazenamento.

00:07:06 - 00:07:34
Porque esse é o momento que pode deixar vocês um pouco confusos, mas com decorrer da aula vocês vão entender um pouco melhor o que eu estou querendo dizer. É igual o Javascript, gente, eu sei que foi confuso lá, mas aqui também eu sei que é muita informação. Respeitem o processo de aprendizado de vocês. No início, quando eu comecei a estudar sobre Git, entender essa estrutura, repositório, isso e aquilo, também foi muito confuso, mas depois eu fui pegando porque é a prática, gente, não tem jeito.

00:07:34 - 00:07:50
Programar é prática. Quanto mais vocês praticarem mais vocês vão entender tudo que tá sendo dito aqui. Ok!?
Então vamos lá. Nós temos o nosso diretório de trabalho, temos o nosso HTML, CSS, Javascript.

00:07:50 - 00:08:22
Imagine que temos o nosso projeto final, aquele dicionário que fizemos no nosso projetinho lá do curso. E essas estrelinhas são as modificações, a gente modificou HTML e o CSS.
O diretório de trabalho é o nosso ambiente local, ok!? É onde a gente armazenou localmente o nosso projeto, lá no arquivo de documentos, na parte de download, onde vocês salvaram esse projeto. E quando a gente faz uma alteração local, é preciso subir para o nosso repositório.

00:08:22 - 00:08:48
Que é um espaço onde a gente armazena o nosso código. Certo!?
Nesse meio termo a gente tem um espaço de armazenamento temporário em que a gente prepara o nosso arquivo e sobe, e beleza, deixa aqui temporariamente depois sobe para o repositório, mas como é que a gente faz isso?
O que eu estou explicando aqui é mais ou menos um fluxo de trabalho como o Git vai, mais ou menos, trabalhar.

00:08:48 - 00:09:12
Vocês devem estar se perguntando: Como é que eu faço isso? Como é que eu subo? Como é que isso aqui vai para o espaço temporário de armazenamento? Como é que depois isso vai para o repositório?
É aí que vem os famosos comandos do Git. Através desses comandos, que vocês utilizam no terminal, é que vocês vão conseguir gerenciar esses arquivos.

00:09:12 - 00:09:42
Para que consigam conectar o que vocês estão fazendo dentro do diretório local, da máquina de vocês, até o repositório até onde está hospedado o código onde vocês conseguem puxar ou subir essas alterações e também ter esse espaço de armazenamento temporário, que eu vou falar um pouquinho mais à frente. Eu estou mostrando isso aqui antes para vocês porque lá na frente vamos trazer de novo essa estrutura de uma forma mais técnica para vocês começarem a juntar as peças sobre como funciona o Git.

00:09:42 - 00:10:06
Passando agora pelos comandos, que é super importante, gente, não precisa decorar tudo. Com o tempo vocês vão pegando prática mesmo, porque é uma coisa de repetição, mas não precisa decorar tudo. Inclusive, vamos deixar alguns materiais de apoio alguns links para vocês estudarem e servir como material de base. Fiquem tranquiles quanto a isso. Vamos lá!

00:10:06 - 00:10:30
Para criar um novo repositório da nossa máquina podemos dar um "git inite". Vai aparecer uma tela de configuração e vamos criar esse novo repositório. Obter um novo repositório.
Para você puxar um novo repositório no Git a gente usa o "git clone". Basta dar um "git clone" e a URL do repositório. Fiquem tranquiles, vamos fazer isso tudo na aula prática.

00:10:30, - 00:10:55
Quando chegarmos no tópico de github a gente vai falar isso com exemplos, porque se eu falar eu sei que vai entrar em um ouvido e sair pelo outro, vocês vão ter dificuldade de entender algumas coisas, fiquem tranquiles em relação a isso que a gente vai fazer um exemplo prático, vamos usar algum dos comandos, não vamos usar todos esses aqui, mas vamos usar alguns para vocês entenderem como é que funciona o fluxo de trabalho utilizando o Git. Ok!?

00:10:55 - 00:11:16
O "git clone" é usado quando queremos obter um repositório, ou na linguagem mais comum, no meio das pessoas desenvolvedoras, clonar um repositório. "Ah, Eu Vou Clonar um repositório." Quando você ouvir esse termo é porque a pessoa vai pegar o link do repositório e vai clonar na máquina dela. Vai dar um "git clone", vai botar o URL do repositório, geralmente é "ponto git" (.git).

00:11:16 - 00:11:40
Eu vou mostrar para vocês depois, geralmente é o ".git", e ela vai conseguir clonar o repositório na máquina dela. Ok!?
Adicionar um repositório remoto."git remote add origin" e o nome do servidor. O "remote" nada mais é que um repositório remoto que está hospedado em algum outro lugar. Por exemplo, o github, que está hospedado no servidor online.

00:11:40 - 00:12:04
Ele vai adicionar um repositório remoto na máquina dele, ou dela. Ele vai dar o "git remote add origin servidor". Provavelmente vamos usar isso quando subirmos nosso projeto para o github. E reforçando, eu vou explicar isso de forma prática. Adicionar mudanças no repositório local, a gente dá um "git add."

00:12:04 - 00:12:31
Estamos adicionando as mudanças. Lembra daquela imagemzinha do HTML, CSS, JavaScript, estrelinhas. E lá no meio termo, naquele espaço temporário, ficou só HTML e CSS, foi porque demos um "git add." Adicionamos. Olha, esses dois arquivos eu quero que você suba, para isso ele vai dar um "git add." para adicionar aqueles arquivos. E depois temos que confirmar essas mudanças, que é o "git commit".

00:12:31 - 00:12:55
Quando alguém falar: "Eu vou comitar esse arquivo". "Vou dar um comite".
A pessoa está dizendo que vai confirmar essas mudanças, geralmente é "git commit -m".E o comentário da alteração vai para aquele espaço de teste o "stage index" que é o pré para subir para o repositório.

00:12:55 - 00:13:22
Que é um espaço temporário onde, depois que você adiciona e comita, você confirma essas mudanças, ele fica naquele espaço temporário aguardando justamente você fazer depois o "push" que é enviar alterações para o repositório. Quando você envia você informa que está subindo os arquivos, que estavam anteriormente na sua máquina e que vão continuar na sua máquina, porém, eles vão para o repositório onde você está hospedando esse código.

00:13:22 - 00:13:49
Geralmente você dá um "git push origin" mais o nome do ramo. O ramo a gente chama de "branch". Ok!? Toda vez que vocês ouvirem: "Ah, é porque na "branch main", na "branch test", na "branch stage, na "branch production", na "branch dev", estamos falando de ramo. Nós temos um grande repositório no PrograMaria, certo!? E dentro do repositório PrograMaria temos uma "branch" chamada Dicionário.

00:13:49 - 00:14:15
Ou seja, é uma ramificação daquele repositório em que temos o dicionário. Só que aí, temos uma outra "branch", um outro ramo chamado Ada, que é o site da Ada. Ou seja, é uma outra ramificação, dentro do repositório PrograMaria, que temos, e assim como muitos outros, outros projetos e tal, que podem estar dentro daquele repositório. Eu acho que um bom exemplo poderia ser uma árvore.

00:14:15 - 00:14:41
Imagina aquela arvorezinha que vai criando vários ramos. Poderíamos associar isso a uma árvore. Uma árvore de informações armazenadas que vai criando novas ramificações. Ok!?
Para atualizarmos um repositório local damos um "git pull". Por exemplo, duas pessoas estão trabalhando no projeto, e uma pessoa mexeu no ramo, na "branch", do dicionário. E você precisa puxar essas atualizações que essa outra pessoa fez.

00:14:41 - 00:15:07
Você vai lá no terminal, dentro do repositório do seu projeto, você vai dar um "git pull", ou seja, você vai puxar essas atualizações para sua máquina.
E assim você vai conseguir visualizar as modificações que aquela pessoa fez. E se a gente precisar mesclar as alterações de um ramo para o outro a gente dá o "git merge" e o nome da "branch". Isso é muito comum quando a gente precisa subir coisas para a "main".

00:15:07 - 00:15:32
Costuma ser a "branch" principal. Quando for criar o processo da conta do github vocês vão ver que, automaticamente, o Git tem um repositório com nome "main". Que é o nosso repositório principal. Quando você está trabalhando dentro do repositório PrograMaria na "branch" dicionário e você precisa subir para "main", por exemplo, você vai dar um "git merge". Ou seja, você vai mergear, você vai mesclar.

00:15:32 - 00:15:57
Inclusive "mergear" é um termo que acabamos usando no dia a dia. "Ah, vou mergear o arquivo tal". Você já sabe que a pessoa vai mesclar alterações de um ramo para outro.
Ou seja, ele vai mesclar alterações da "branch" PrograMaria para a "main". Ok!?
Lembra que eu falei lá no início que depois eu ia mostrar um outro slide mostrando, tecnicamente, toda aquela estrutura que provavelmente algumas de vocês já devem ter ficado um pouco confusas?

00:15:57 - 00:16:24
Aqui já fica mais claro, mas eu precisava primeiro passar pelos comandos para vocês entenderem de uma forma muito mais clara como é que é a estrutura, como que é o fluxo de trabalho do Git, agora com os comandos certos. Lembra que a gente tem o nosso diretório de trabalho, que é o "working directory"? Temos aqui HTML CSS JavaScript, aquele nosso arquivo de projeto.

00:16:24 - 00:16:56
E quando temos as alterações, por exemplo, a gente alterou HTML CSS, damos um "git add" para adicionar os arquivos no "Staging Area", que é aquele armazenamento temporário. Lembra? E depois a gente dá o "git commit", porque ele armazena no repositório local, que a gente chama de "red" que a ramificação atual do nosso trabalho. Ainda não é o repositório final porque a gente ainda não deu o "push".

00:16:56 - 00:17:20
Só demos o "commit" e deixamos armazenado temporariamente porque depois que a gente der o comando para subir, aí sim, nós vamos para o repositório remoto, que é o "github", gitlab" e muitos outros que existem. Tem um comando aqui que eu não comentei que é o "git reset". O "git reset" é quando a gente precisa voltar algum arquivo que a gente colocou nessa área de "Staging", depois que a gente adicionou, para o diretório local. Ok!? "Ai, meu Deus, eu fiz uma alteração sem querer!"

00:17:20 - 00:17:54
Preciso voltar o que estava antes. Você dá um "git reset" e cancela essa adição dos arquivos e você volta para o diretório de trabalho e não vai confirmar as modificações que alterou.
Mas voltando aqui, depois você dá o "git push", vai para o repositório remoto e por fim você dá o "git pull", caso tenha alguma atualização, para atualizar o seu repositório local. O repositório local nada mais é do que aquela pasta onde você tem todos os seus arquivos. Ok!?

00:17:54 - 00:18:10
Diretório local é onde está local na sua máquina. O repositório local é onde você sobe... é como se fosse uma pré-produção, vamos dizer assim, fica ali só aguardando o comando para justamente subir para o repositório final. Ok!?