**Git Curso de Git (OtterWise)**

O Git é uma ferramenta que irá nos ajudar a criar um versionamento dos nossos arquivos. O versionamento irá realizar uns checkpoints, ou seja, iremos criar várias versões do nosso arquivo.

Assim iremos conseguir comparar versões, ou voltar para versões onde a nossa aplicação funcionava antes de subirmos o nosso deploy.

**Conceitos básicos do Git**

* Repositório: Pasta onde os arquivos e subpastas estão sob controle de versão
* Branch: Linha do tempo que guarda os pontos de salvamento dos nossos arquivos. Podemos ter múltiplas branchs em nosso projeto
* Commit: Ponto de salvamento do estado dos arquivos no projeto

*Repositório*: É uma pasta que teremos os arquivos e subpastas protegidos pelo versionamento, podemos entender ele como pasta raiz do nosso projeto.

*Branch*: Seria a linha do tempo que guarda os pontos de salvamento. Aqui será onde nós iremos realizar o salvamento dos nossos arquivos e essa branch cria uma linha temporal, contendo o histórico de todos os salvamentos que nós realizamos.

*Commit*: É o ponto de salvamento em sí.

Pense no repositório como o projeto total como um todo, e a branch é a linha total de onde você estará trabalhando.

O git pode ser usado somente de forma local, e para termos uma segurança maior, iremos utilizar o armazenamento em nuvem disponibilizado pelo GitHub.

**Comandos básicos do Git**

* Init
  + Inicia localmente um repositório git.
* Status
  + Mostra o Status da branch que estamos e as modificações realizadas
* Add
  + Adiciona os arquivos indicados em um estado preparado para o commit
* Commit
  + Salva todos os arquivos preparados em uma nova versão
* Push
  + Publica nossa branch local atualizada na remota
* Branch
  + Cria uma nova branch
* Checkout
  + Troca de branch

Nós podemos indicar um commit somente dos arquivos que estão prontos através do add

**GitFlow**

A definição de um fluxo pelo qual nossos commits devem passar para termos um ambiente organizado.

O git flow começa com uma linha principal indicada pelo main, essa linha é criada logo após o nosso commit inicial. A partir daqui teremos todo um fluxo a ser seguido, esse fluxo varia de empresa para empresa ou de projeto por projeto.

Quando queremos realizar uma tarefa nova, nós criamos uma nova branch

Feita as realizações da nossa branch nós realizamos um commit e feito isso realizamos um pull request para que as modificações da nossa branch criada para a branch principal.

Em um fluxo de trabalho todo mundo consome tudo da branch principal, vai criar a sua própria branch paralela e depois irá apontar de volta para a branch principal, para que ela seja avaliada e finalmente aprovada para a branch principal.

Veja a nossa imagem abaixo como exemplo:

Diagrama

Descrição gerada automaticamente

Funcionalidade nova chamamos de feat/jsbasico-ex01 é para informar que é de js do exercício 01.

Podemos começar com bug quando queremos realizar a correção de um bug, podemos utilizar ref, quando estamos refatorando algo e por aí vai.

Ao adicionarmos o git add nós precisamos passar um parâmetro para ele, nesse caso iremos utilizar como parâmetro apenas o arquivo que acabamos de criar, chamado de arquivo.txt, e ficará da seguinte forma:

Git add .\arquivo.txt

Caso você tenha enviado um arquivo de forma acidental, é possível reverter essa ação utilizando o seguinte comando.

Git rm –cached <file> ...

Veja aqui um exemplo prático:

Git rm –cached arquivo.txt

Nós podemos realizar um seguinte argumento quando queremos subir por exemplo 12 arquivos de uma vez, nós podemos utilizar o “.” Ele quer dizer que nós iremos adicionar todos os arquivos.

Git add .

O git log irá nos dar o histórico dos commit na branch que estamos

Vamos agora commitar o nosso arquivo utilizando a seguinte instrução:

Git commit -m “add arquivo.txt file”

O menos m significa uma mensagem

Criamos agora uma nova branch utilizando o git branch

Git branch nova\_branch

Podemos agora verificar quais branchs existem em nosso projeto, utilizamos assim o git branch

Para trocarmos de branch iremos utilizar o git checkout nova\_branch

**Prática repositório remoto**

Começamos criando um arquivo para ser um readig

Echo “# curso-front-end” >> README.md

O echo é um comando do sistema e não está relacionado ao git, ele basicamente cria arquivos.

Vamos renomear a nossa branch

Vamos mudar o nome da nossa branch para main da seguinte forma: git branch -M main

Vamos adicionar agora o nosso repositório remoto

Git remote add origin “link”

Criado o repositório remoto, podemos agora subir o nosso arquivo criado para o repositório remoto, da seguinte forma: git push -u origin main

O push commita do repositório local, para o nosso repositório remoto.

O -u é o set upstream

O -u irá subir para o repositório remoto e crie uma branch chamada main.

O push vai sempre subir para o repositório que você está e o -u vai manter essas branches sempre vinculadas

Vamos criar uma nova branch no nosso projeto e como ela será uma funcionalidade iremos chama-la de feat da seguinte forma: git branch feat/git-pratica-branch

O fix estaremos corrigindo algo

O doc add um documento

O style eraremos criando um arquivo de estilo etc

Também é uma boa prática termos essa nomenclatura dentro dos nossos commit para o repositório remoto, exemplo: git commit -m “feat: changes contents of README.md and arquivo.txt”

O # no github cria um título para o seu comentário

Quando já unificamos as nossas branchs podemos excluir elas .

O git fetch –all irá nos informar todas as mudanças

O git pull realiza a atualização das branchs

**Gitclone**

* Comando que cria uma cópia de um repositório
* Todos os arquivos, branchs e commits serão copiados para o novo repositório
* Utilizado para “baixar” um repositório remoto para um repositório local na máquina
* Com o clone o vínculo entre repositório remoto e local é mantido, por isso, se realizado um git push as alterações subirão para o repositório remoto, sem a necessidade de maiores configurações.

**Fork**

* Criar uma bifurcação do repositório remoto
* Utilizado para testar projetos, propor modificações e realizar correções

Exercícios

1. Criar uma conta no Github e criar um repositório remoto privado para armazenar os exercícios e desafios realizados ao longo do curso
2. Criar um repositório local e sincronizar com o repositório remoto criado
3. Convidar o(a) instrutor(a) da Otterwise como colaborador de seu primeiro repositório remoto (Settings > manage access > invite collaborators

O comando git remote -v irá nos informar a nossa origem linkado no nosso repositório la no git.