**GIT**

**Git Introdução**

**Controle de versão:** - Gerenciamento de código-fonte

- Registrar todas as modificações

- Alteração facilitada entre as versões

- Possibilita cada membro trabalhar em um *branche* diferente

- Utiliza ferramentas como git e SVN

**Branches(Ramificações):** É a forma que o git separa as diferentes versões do projetos.

**O que é git:** - Sistema de controle de versão mais utilizado no mundo

- Baseado em repositórios

- Operações otimizadas para ter alto desempenho

- São protegidos com criptografia

- Projeto de código aberto

**Git Fundamental**

**Repositório:** - Onde o código é armazenado

- Cada projeto tem um repositório na maioria das vezes

- Alguns servidores que gerenciam repos: GitHub e Bitbucket

**GitHub:** - Serviço de gerenciamento de repositórios

- Compartilhamento de projetos

- Gratuitos para projetos públicos e privados

**Enviando repo para o GH:**  - Precisa criar o projeto no GitHub e inicializar o mesmo no git em nossa máquina, depois sincronizar com o GH e enviar.

**.gitignore:** Ignora arquivos que estiverem listados nele.

**COMANDOS IMPORTANTES**

**git init:** Cria um novo repo dentro do local selecionado.

**git status:** Serve para verificar o status do seu repo, mapear mudanças.

**git add nome\_do\_arquivo:** Adiciona o arquivo no git que será compartilhado.

**git add . :** Adiciona todos os arquivos de uma vez.

**git commit -m(adiciona uma msg) “mensagem”:** Cria uma commit no repositório.

**git commit -a(todos os arquivos)**

**git branch -M nome\_do\_branch:** Cria um branch no repositório.

**git remote add origin https://github.com/Morvanmc/sendFirst.git:** Sincroniza com o repositório no GitHub.

**git remote rm origin:** Remove o caminho do GitHub.

**git push:** Adiciona os arquivos adicionados nos servidores do GitHub.

**git push -u origin nome\_do\_branch:** Adiciona um arquivo específico adicionado nos servidores do GitHub.

**git pull:** Atualiza o repo na máquina com atualizações e arquivos novos criados.

**git clone https://github.com/Morvanmc/sendFirst.git:** Clona um repo. Geralmente usado quando entramos em um novo projeto.

**git rm nome\_do\_arquivo:** Deleta um arquivo.

**git log:** Traz os commits realizados no projeto em ordem do mais atual para o mais antigo. Para sair do terminal de log aperta ‘q’ ou ‘ctrl+c’.

**git shortlog:** Mostra um log resumido do projeto, com commits e autores.

**git reflog:**  Mostra todos os passos que realizamos dentro no projeto.

**git mv nomeAntigo nomeNovo:** Serve para renomear um arquivo ou até mesmo muda-lo de diretório.

**git checkout nome\_do\_arquivo:** Volta o arquivo para o estado original, que está no GitHub.

**git reset –hard origin/master:** Exclui todas as alterações commitadas e também as pendentes.

**git branch:** Lista os branches criados.

**git branch nome\_do\_branch:** Cria um novo branch.

**git branch -d ou –delete:** Deleta um branch.

**git checkout nome\_do\_branch:** Muda de branch.

**git checkout -b nome\_do\_branch:** Cria um novo branch e muda de branch.

**git merge nome\_do\_branch:** Une o código de duas branches.

**git stash:** Envia o arquivo pro “lixo” pra iniciar uma nova abordagem, mas dá a possibilidade de recupera-lo.

**git stash list:** Mostra a lista de stash criadas.

**git stash apply numero\_da\_stash:** Volta pra branch a stach selecionada.

**git stash show -p numero\_da\_stash:** Mostra os arquivos na stach.

**git stash clear:** Apaga todas as stashs.

**git stash drop numero\_da\_stash:** Apaga uma stash específica.

**git tag -a nome -m “mensagem”:** Cria uma tag. Tag são espécies de checkpoints.

**git tag:** Exibe as tags criadas.

**git show:** Mostra as informações do branch atual e seus commits.

**git show nome\_da\_tag:** Mostra as tags criadas e suas informações.

**git checkout nome\_da\_tag:** Muda para a tag selecionada.

**git push origin nome\_da\_tag:** Envia e compartilha uma tag específica.

**git push origin --tags:** Envia todas as tags.

**git fetch -a:** É utilizado para mapear os branches que ainda não foram mapeados pela sua máquina, normalmente branches criados por outros devs.

**git submodule add <link> <nome>:** Adiciona um submódulo dentro do projeto atual.

**git submodule:** Mostra os submódulos dentro do projeto atual.

**git push --recurse-submodules=on-demand :** Atualiza o submódulo.

**git diff:** Exibe a diferença entre os branch atual e o branch remoto.

**git diff <arquivo><arquivo\_b>:** Exibe a diferença entre 2 arquivos.

**git clean:** Verifica e limpa todos os arquivos que não foram trackeados, ou seja, aqueles que não foram add. A flag -f força a limpeza.

**git gc:** Garbage Collector é utilizados para limpar arquivos desnecessários e que não são mais utilizados no projeto, otimizando-o.

**git fsck:** File System Check, verifica se há arquivos corrompidos. Comando de rotina no repo.

**git archive –format zip –output master\_zip master:** Compacta o repositório em um arquivo zip.

**GitHub**

**Criando Repositório**

As informações essenciais quando criamos um novo repositório no GitHub são: Nome do repo, descrição e licença.

**Aba Code**

Na aba Code temos as principais informações do seu projeto, como o código em si e também as informações adicionais como README, tipo de licença, branches, tags, etc.

**Aba Issues**

A aba Issues funciona como uma espécie de sistema de OS, onde podemos abrir requisições de tarefas e resoluções de bugs, cadastrar as issues de forma direcionadas para a evolução do projeto.

**Aba Pull Request**

Nessa aba os colaboradores do projeto criam pull request para enviar códigos que resolvam as issues e adicionem novas funcionalidades no projeto. Normalmente são analisadas pelo team lead antes de serem mergeadas no branch master.

**Aba Actions**

Onde se cria automatizações de deploy com integração em outros serviços, incluindo CI/CD(Continuous Integration/ Continuous Development).

**Aba Projects**

Podemos criar um projeto e utilizar um quadro de tarefas, em um processo conhecido como Kanban. É uma espécie de Trello dentro do GitHub, que ajuda a identificar em qual parte do desenvolvimento se encontra cada tarefa.

**Aba Wiki**

O propósito da aba wiki é criar uma documentação completa do seu projeto, onde podemos esmiuçar sobre bugs conhecidos e não solucionados, funcionalidades e tudo o que venha a ajudar no entendimento do projeto.

**Aba Insights**

Nessa aba temos acesso a diversas informações detalhadas sobre o projeto.

**Aba Settings**

Temos acesso a diversas configurações do projeto, como: Adicionar colaboradores, remover repo, etc.

**CRIANDO GIST**

Gist são blocos de códigos que ajudam a resolver algum problema específico. No GitHub é só ir no botão de + e ir em New Gist.

**MARKDOWN**

É uma forma de adicionar estilo a textos na web, dando melhor experiência para o usuário nas suas documentações. O arquivo é de extensão .md.

**Títulos**

**#** => h1

**##** => h2

**###** => h3 ...

**Ênfase nos textos**

**\*\*texto\*\* ou \_\_texto\_\_ :** Negrito

**\*texto\* ou \_texto\_ :** Italico

**Listas com Markdown**

\*item: UL

1.item: OL

**Inserindo imagens**

![Texto Alt](link imagem)

**Links**

[Texto do link](link)

**Inserindo Código Fonte**

``` código ```

**Lista de tarefas do projeto**

Concluídas: - [x] texto

Não conluídas: - [ ] texto

**GitHub Pages**

Uma forma de criar uma página estática nos servidores do GitHub. Ótima alternativa para fazer o seu próprio portifólio.