1º O que é e como funciona o Git e o GitHub?

O que é Git?

Criado pelo engenheiro de software Linus Torvalds, conhecido por ter desenvolvido, também, o núcleo Linux, o GIT é um Sistema de Controle de Versões Distribuído — ou DVCS.

Estes sistemas de controle possuem a função de registrar quaisquer alterações feitas em cima de um código, armazenando essas informações e permitindo que, caso seja necessário, um(a) programador(a) possa regredir a versões anteriores de uma aplicação de modo simples e rápido.

Este tipo de sistema também simplifica muito o processo de compartilhamento de um projeto com um time, por exemplo, ou com outros(as) programadores(as).

Esta compreensão prévia quebrará alguns obstáculos característicos dos primeiros momentos em que manipulamos uma plataforma nova.

Abaixo, listamos e explicamos o que significam algumas nomenclaturas básicas muito comuns na manipulação de códigos-fonte no GIT e no GitHub.

Repositório

Os repositórios Branch

Branch é o nome dado a uma versão (ramificação) do projeto.

Isso é útil porque possibilita gerenciar múltiplas alterações acontecendo simultaneamente. Por exemplo, podemos fazer com que cada equipe de desenvolvimento

Merge

Para unir as modificações feitas em um branch ao código original, utilizamos o comando merge.

Com esta funcionalidade, todas as alterações feitas em cópias manipuláveis são inseridas, após aprovadas, no código-fonte original sem complicações.

Fork

Quando um profissional desenvolvedor precisa começar a trabalhar em um projeto, seu primeiro passo é copiar este repositório para a sua máquina.

Este processo é realizado pelo comando fork.

O fork também é uma funcionalidade útil quando um membro da equipe precisa pegar um código público para manuseá-lo em um editor de código local ou interno.

são os ambientes criados para armazenar seus códigos.

Você pode possuir um ou mais repositórios, públicos ou privados, locais ou remotos, e eles podem armazenar não somente os próprios códigos a serem modificados, mas também imagens, áudios, arquivos e outros elementos relacionados ao seu projeto.

É através dos seus repositórios públicos que outros programadores poderão ter acesso aos seus códigos no GitHub, podendo, inclusive, cloná-los para adicionar melhorias.

Principais comandos Git

- Init: este comando dá origem a um repositório novo, local ou remoto, ou reinicializa um repositório já existente;
- Clone: este comando clona o código de um repositório para sua manipulação em outro ambiente;
- Commit: este comando move os arquivos da state area para um repositório local;
- Add: este comando adiciona um arquivo alterado a uma staging area, ou seja, o prepara para ser vinculado a um commit;
- Push: este comando envia arquivos de um repositório local para um repositório remoto. No GitHub, por exemplo;
- Pull: ao contrário do push, este comando traz um arquivo do repositório remoto para o repositório local.
- Merge: este comando serve para unir arquivos alterados ao arquivo original de um projeto. Em outras palavras, é ele quem une os branchs as commits.
- Log: este comando permite a visualização do histórico de commits de um arquivo ou usuário, ou o acesso de uma versão específica.

Como Git funciona na prática?

Como o Git opera de forma majoritariamente local, é possível trabalhar mesmo off-line. Talvez depois você precise fazer uns ajustes, se o time fizer alterações conflitantes com as suas edições no código, mas essa mesclagem de versões e branches também é simples de fazer no Git (ouvi relatos do tamanho de deixar qualquer desenvolvedor de cabelo em pé sobre a dificuldade que era era resolver esse tipo de conflito pré-Git). O legal de tudo ser bem local também é a liberdade de errar: trabalhou no código, mas deu ruim, criou um bug absurdo e não conseguiu resolver? Tudo bem, é só não enviar isso para o resto do time.

O Git estimula o desenvolvedor a ter boas práticas, como abrir branches para trabalhar. Primeiro de tudo: criar uma branch é muito rápido. Uma linha de comando e pronto, você já está no seu próprio braço do projeto e tem liberdade para criar e alterar qualquer coisa, com a tranquilidade de saber que só irá para o código master se você

quiser. Você também pode ir e voltar, é fácil e rápido sair da branch que você está e ir a outra, uma antiga, por exemplo, para atualizar ou resolver algo.

A real é que o Git faz muita coisa e facilita demais a vida do programador, mas vou falar sobre o que eu acho mais mágico é que te permite viajar no tempo. É fantástica a simplicidade do Git para se navegar pelas diferentes versões do seu projeto, reverter para alguma instância específica, ou até, para projetos longos, ver como estavam as coisas há alguns meses. Dá para fazer pesquisas no git log para, por exemplo, ver commits feitos em uma determinada janela de tempo ou por pessoas específicas. Para quem curte uma ferramenta gráfica, o GitGraph, extensão para o VS Code, é muito legal nesse aspecto, ilustra bem toda a timeline e as diferentes branches e mesclagens com a master:

O que é o GitHub?

O GitHub, tão famoso entre a comunidade de programadores de todo o mundo, é uma espécie de rede social voltada a profissionais de TI cuja tecnologia que o sustenta é o GIT.

Em outras palavras, GitHub é uma plataforma totalmente online onde você pode criar repositórios e hospedar neles seus projetos, colaborar com softwares open source, seguir outros(as) programadores(as) e interagir com códigos de terceiros.

O GitHub armazena todos estes dados em uma nuvem e você pode acessá-los de onde estiver: basta logar-se no site em qualquer navegador.

2º Mandar imprimir de 1 a 50 usando for:

```
import java.util.Scanner;
class HelloWorld {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        System.out.println("Informe um numero:   ");
        int num = sc.nextInt();
        for (int i = 1; i <= 50; i++)   {
            int valor = i + num;
```

```
            System.out.println(valor);
            sc.close();
        }
    }
}
```