O QUE É GIT: CONCEITOS, PRINCIPAIS COMANDOS E QUAIS AS VANTAGENS?

Se você é uma pessoa programadora ou de alguma forma está relacionado ao mercado de tecnologia, com certeza já ouviu falar em **GIT**.

Mas, afinal, o que é GIT, o que é GitHub, o que estes dois nomes têm em comum e como a tecnologia por trás destas ferramentas pode auxiliar programadores(as) em todo o mundo?

No artigo de hoje, responderemos a cada uma destas dúvidas e traremos, para te ajudar a compreender melhor a ferramenta, um panorama básico sobre os principais comandos GIT.

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).

Os conceitos do Git

Para começar a utilizar o GIT, é importante que você reconheça e compreenda alguns dos principais conceitos utilizados pela ferramenta.

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** 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.

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.

Principais comandos Git

Se você estiver familiarizado com alguns dos principais comandos do GIT, seus primeiros passos na ferramenta podem se tornar mais descomplicados.

Para isso, trouxemos abaixo uma lista dos comandos mais utilizados pelos programadores(as) e o que eles significam.

- **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.

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.

Resumindo a diferença entre Git e Github

O GitHub serve, fundamentalmente, para facilitar o controle de versões de um **software** ou aplicação.

As diferenças entre ele e o GIT estão nas interações proporcionadas pelo GitHub: funcionando de modo semelhante a uma rede social, o GitHub é hoje um dos maiores pontos de encontro virtuais entre programadores de todo o mundo.

Ele é, também, o maior repositório de softwares de código aberto de toda a internet, tendo, inclusive, como um de seus maiores colaboradores quando o assunto é *open source* a gigante Microsoft.

Vantagens do Git e Github

Não é à toa que essas ferramentas são muito utilizadas entre os desenvolvedores. Suas funcionalidades são diversas, gerando uma gama de oportunidades para os profissionais de tecnologia, além de diversas vantagens que auxiliam o desenvolvimento dos projetos em equipe. Abaixo listaremos algumas dessas vantagens:

Git

Imagine que, há alguns meses, você desenvolveu um site que possuía determinada função.

Com o passar do tempo, essa função perdeu sua utilidade e você foi instruído a retirá-la do código fonte deste site.

Após seis meses, por uma questão de estratégia de negócio, essa função deve ser novamente implementada.

Para programadores(as) que utilizam o GIT, reincluir a funcionalidade é uma tarefa simples: basta buscar, em seu repositório, a versão que a contempla e retomá-la.

Para aqueles(as) que não utilizam, talvez seja necessário reescrever o código desta função, consumindo novamente o tempo gasto para desenvolvê-la da primeira vez.

Diante desta situação, podemos concluir que uma das maiores vantagens de usar o GIT é a economia de tempo e recursos, uma vez que a consulta de diferentes versões de uma mesma aplicação é muito recorrente no trabalho do(a) programador(a).

Outro grande benefício do GIT é justamente o fato de ele ser um sistema **distribuído**.

Isso significa que, diferentemente de outros sistemas de controle de versionamento populares na época em que foi lançado, o GIT possui repositórios, e não somente um único local com o histórico de seu trabalho.

Para auxiliar o trabalho em equipe, recursos como o fluxo de desenvolvimento do **Gitflow** também trazem muitos benefícios, inclusive porque pode ser acessado por qualquer membro do time em qualquer lugar.

Ou seja, por ser o mais rápido controle de versionamentos existente hoje no mercado, o GIT vale a pena porque proporciona a seus usuários e estudantes de programação grande otimização de tempo e recursos.

GitHub

Para os profissionais da área de tecnologia, as vantagens de utilizar o GitHub são inúmeras.

A maior delas, talvez, seja a oportunidade de aprender com programadores e programadoras mais experientes que você, especializado nas mesmas ou em diferentes áreas.

Estas conexões são riquíssimas para os estudantes.

Outros benefícios são:

- Possibilidade de acompanhar e colaborar com projetos de diferentes equipes;

- Aprender programação na prática ao observar o avanço do desenvolvimento de aplicações de terceiros;
- Participar de discussões a respeito de novas tecnologias;
- Obter auxílio de outros programadores para resolver problemas relacionados a seus projetos;
- Controlar as diferentes versões de um código com armazenamento em nuvem;
- Registrar ações e projetos desenvolvidos por você em uma espécie de portfólio online, etc.

Aprenda a programar

Se você tinha dúvidas sobre começar ou não começar a usar o GIT, saiba que este é um recurso que tem muito a agregar à sua carreira e conhecimento, então vá fundo!

Ao cursar o programa de **programação full stack** da Kenzie Academy Brasil, além de apresentar a desenvolver nas principais linguagens do mercado, front e back-end, você também aprenderá o GIT.

É através dele que nossos alunos gerenciam seus projetos, além de participarem ativamente do GitHub.Se você gostou desse conteúdo e gostaria de entrar na área de programação, **conheça mais sobre a formação da Kenzie Academy** e mergulhe de vez nesse mundo da tecnologia. Estude Desenvolvimento Full Stack e se torne referência no mercado.

2º

Scanner sc = new

Scanner (System.in);

For(int=0;i<=50;i++){ System.out.println(i):}