**Conheça o Git, o sistema de versionamento que protege os projetos**

*O Git é um Sistema de Controle de Versão Distribuído que evita que alterações realizadas em um projeto modifiquem o código-fonte. Neste conteúdo você vai entender como funciona!*

(Atualizado 1 de setembro 2021)

O Git é uma solução que está diretamente ligada à área de desenvolvimento e que foi **criada para facilitar a vida do profissional**.

Você já teve que editar um arquivo e subí-lo via FTP e, quando terminou, descobriu que pouco tempo depois uma outra pessoa tinha subido esse mesmo arquivo, sobrescrevendo as suas alterações?

Pois é para evitar esse desperdício de tempo e energia  que pode colocar a perder todo o trabalho, que serve o Git.

Mas afinal, **o que é Git, porque ele foi criado e quais as suas principais funcionalidades?**

Continue a leitura e descubra!

## O que é Git?

Por definição, o Git é um Sistema de Controle de Versão Distribuído. Ele foi criado por Linus Torvalds, que é **mais conhecido por outra criação, o Linux**.

Linus desenvolveu o Git após ter ficado insatisfeito com uma mudança de licença de utilização do [BitKeeper](http://www.bitkeeper.org/" \t "_blank), o sistema de controle de versão que era utilizado por ele e pela equipe que desenvolvia o Linux.

Como na época não existia nada similar, ele resolveu criar a sua própria versão de um sistema para controlar as alterações do projeto [Linux](https://www.hostgator.com.br/blog/o-que-e-linux/?itm_source=blog&itm_medium=o-que-e-git&itm_campaign=linux). E foi aí que surgiu o Git.

Como um bom sistema de controle de versão, o Git **registra as mudanças que ocorrem no código-fonte de um projeto**. Logo, permite que os arquivos sejam alterados de forma simultânea, por inúmeras pessoas, sem a preocupação que essas alterações sejam sobrescritas umas pelas outras.

Devido a essa funcionalidade, caso haja algum problema em uma alteração do código-fonte, **é possível restaurar à versão anterior de forma fácil e rápida**.

Uma tecnologia como essa é essencial para equipes que estejam trabalhando em um mesmo projeto. Afinal, ele simplifica o controle de operações realizadas por inúmeras pessoas.

## Qual o diferencial do Git?

**A sua arquitetura descentralizada** é o que faz com que o Git seja diferenciado em relação às outras opções de sistemas similares.

O [Subversion](https://subversion.apache.org/" \t "_blank) e CSV, por exemplo, possuem uma arquitetura centralizada, ou seja, eles **contam com um repositório central**.

Uma das grandes desvantagens desse tipo de arquitetura é a baixa performance gerada caso tenha muitas equipes e projetos em um mesmo repositório. Além disso, é preciso estar sempre on-line para realizar as operações, como criar as ramificações.

Já na arquitetura descentralizada adotada pelo Git, não existe um repositório oficial, embora isso possa ser feito. Sendo assim, em cada repositório, incluindo o da máquina do contribuidor, vai existir uma cópia completa e funcional, **permitindo a utilização das operações mesmo offline**.

Vale citar que existem outros sistemas semelhantes do Git, como o [Mercurial](https://www.mercurial-scm.org/) e o [Bazaar](https://bazaar.canonical.com/en/). Porém, eles não são muito populares.

## Qual a importância do Git?

O Git é essencial em quase todos os **projetos realizados em equipes**que geralmente trabalham em paralelo. Isso porque ele evita que haja conflitos entre as alterações realizadas.

Como os projetos costumam estar em constante evolução, é comum ocorrer alterações que possam causar problemas no funcionamento deles. Nesse caso, o Git permite que tais alterações sejam revertidas de maneira simples e rápida, voltando a versão antiga do projeto.

Atualmente, o Git é utilizado de forma tão ampla que **ter conhecimento sobre ele é praticamente obrigatório** para qualquer oportunidade de emprego envolvendo a área de desenvolvimento.

## Como o Git funciona, na prática?

De forma simplificada, o Git trabalha em uma arquitetura em Branch (ou ramificações). Cada novo commit, ou seja, alteração do código, **cria um novo ponto na ramificação atual** (uma branch).

Veja uma exemplificação de um fluxo de contribuição em um projeto utilizando o Git:

### 1. Clone do projeto

O primeiro passo para obter o código-fonte do projeto é cloná-lo em nossas máquinas, para que seus arquivos fiquem disponíveis localmente.

### 2. Criação da Branch

Ao criar uma Branch, estamos criando uma nova ramificação, totalmente independente, para podermos alterar os arquivos do projeto sem interferir nos originais.

Esse processo é considerado uma boa prática quando se está trabalhando em nova funcionalidade.

### 3. Commits

Conforme vão sendo criados e alterados os arquivos, elas vão sendo divididas em commits. É importante que a descrição de cada commit seja objetiva, pois ela vai ficar salva no histórico das alterações.

### 4. Hora do Push

Uma vez que a funcionalidade está totalmente finalizada, devemos enviar nossa Branch, com todas as alterações, de volta ao repositório remoto. Assim, ela ficará disponível para os demais contribuidores do projeto poderem ver e alterar.

### 5 – Merge para juntar tudo

Para mesclar as modificações de sua Branch com os arquivos originais do projeto da Branch principal ou máster, você pode utilizar o comando Merge. Após isso, é necessário dar um commit e um push, para enviar a ramificação máster mesclada ao repositório remoto e deixar tudo disponível para os demais contribuidores.

Existe também o Pull Request, que geralmente tem relação com a contribuição em projetos open source. Basicamente, ele ocorre quando se pede para o dono do repositório que suas modificações sejam incluídas nele.

### Quais os comandos mais utilizados no Git?

Existem alguns comandos que costumam ser mais utilizados por quem usa diariamente esse sistema de versionamento. Conheça alguns deles:

* **Initialize**: Serve para inicializar um repositório vazio ou reinicializar um já existente;
* **Clone**: Utilizado para clonar o repositório;
* **Add**: Serve para adicionar um arquivo para o State Area, para que seja vinculado a um próximo commit;
* **Commit**: Comando que serve para mover os arquivos do State Area para o repositório local;
* **Pull**: Utilizado para buscar e trazer mudanças do repositório remoto para o repositório local, ou seja, unir os conteúdos dos arquivos alterados. Em alguns casos, pode ser necessária uma intervenção humana para realizar essa união;
* **Push**: Utilizado para enviar o arquivo do repositório local para o remoto;
* **Murge**: Serve para mesclar commits e Branchs na Branch atual;
* **Log**: Permite ver o histórico de commits ou um específico.

## Git é a mesma coisa que Github?

Muitas pessoas confundem Git com Github, mas tratam-se de termos distintos. Como explicamos, o Git é o sistema de versionamento, enquanto o Github é uma plataforma para criação de repositórios Git.

Em outras palavras, o Github permite que você crie um repositório Git remoto, possibilitando que outras pessoas possam clonar esse repositório localmente, realizar alterações e enviar essas alterações de volta para o repositório remoto.

Existem também outros sistemas que permitem hospedar**repositórios Git**, incluindo o Gitlab e o Bitbucket.

É importante destacar que utilizar um sistema de **hospedagem de código** é uma forma mais rápida e simples de criar um repositório Git e começar a desenvolver projetos incríveis em equipe.