

# **Guia Prático de Git e Github**

Matheus Larentis Mallmann

PORTO ALEGRE  
2021

# Guia Git Básico

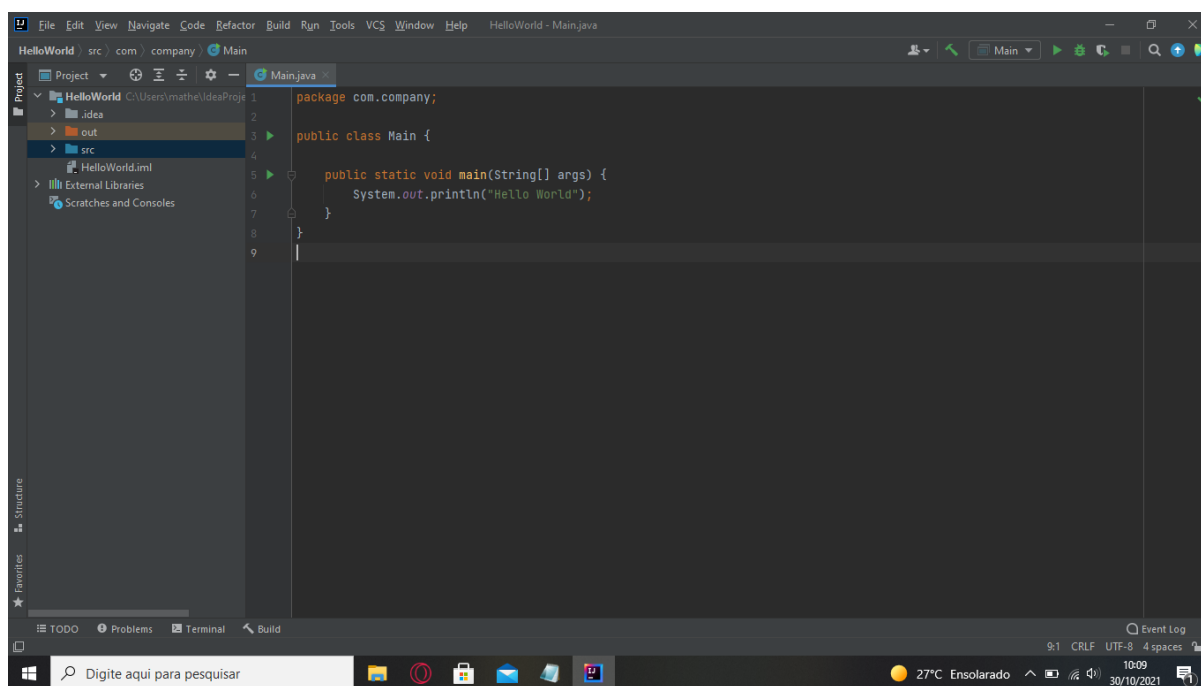
## 1. O que é Git?

Git é um sistema de controle de versão, que além de armazenar o código, é capaz de manter o histórico de alterações do código. Com isso, somos capazes de voltar atrás e recuperar uma versão do sistema anterior ao que temos hoje. Mas o Git não trabalha sozinho. O Github é uma plataforma que gerencia o código e cria um ambiente de compartilhamento entre todos os desenvolvedores que estão trabalhando naquele código.

## 2. Como usar Git?

### 1. Criação do repositório local

O pontapé inicial para a criação do versionamento do código. Neste guia utilizarei o IntelliJ para criação do projeto. Então vamos criar um projeto simples de Hello World.



Projeto criado. Agora vamos dar início ao Git. Utilizando o Git Bash, vamos até o diretório do projeto, no meu caso o projeto está localizado `C:/Users/mathe/Ideas Projects/HelloWorld/.git/`.

Estando no diretório correto, iniciamos o gerenciamento do projeto com o comando:

`git init`

Após executar o comando, será iniciado um repositório vazio neste diretório.

```
$ git init
Initialized empty Git repository in C:/Users/mathe/IdeaProjects/HelloWorld/.git/
```

## 2. Adicionando arquivos a serem salvos

Com o comando `git status`, conseguimos ver quais arquivos não estão sendo trackeados pelo repositório. Isso pode acontecer quando adicionamos um arquivo no projeto e não adicionamos ele ainda ao repositório, ou alguma alteração foi feita em algum arquivo e está diferente da versão do repositório.

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/
        HelloWorld.iml
        out/
        src/

nothing added to commit but untracked files present (use "git add" to track)
```

Em vermelho estão os arquivos não trackeados ainda. Podemos adicionar um arquivo por vez, utilizando o comando `git add <nome_arquivo>` ou podemos adicionar todos de uma vez, que será o nosso caso.

`git add .`

### 3. Salvando nova versão do projeto

Depois de adicionar os arquivos, se dermos um git status, veremos que aqueles arquivos que antes estavam em vermelho, agora estarão verde.

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .idea/.gitignore
    new file:   .idea/description.html
    new file:   .idea/encodings.xml
    new file:   .idea/misc.xml
    new file:   .idea/modules.xml
    new file:   .idea/project-template.xml
    new file:   .idea/runConfigurations.xml
    new file:   HelloWorld.iml
    new file:   out/production/HelloWorld/com/company/Main.class
    new file:   src/com/company/Main.java
```

Porém, nossos arquivos só foram adicionados ao repositório. Utilizamos o comando git commit -m "<comentário\_simples>" para salvar o estado atual do nosso projeto, nossa primeira versão.

```
$ git commit -m "Nosso primeiro commit"
[master (root-commit) 8a3f33b] Nosso primeiro commit
10 files changed, 62 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/description.html
create mode 100644 .idea/encodings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/project-template.xml
create mode 100644 .idea/runConfigurations.xml
create mode 100644 HelloWorld.iml
create mode 100644 out/production/HelloWorld/com/company/Main.class
create mode 100644 src/com/company/Main.java
```

Para visualizarmos todos os commit realizados até o momento no repositório, utilizamos o comando git log. Este comando irá retornar:

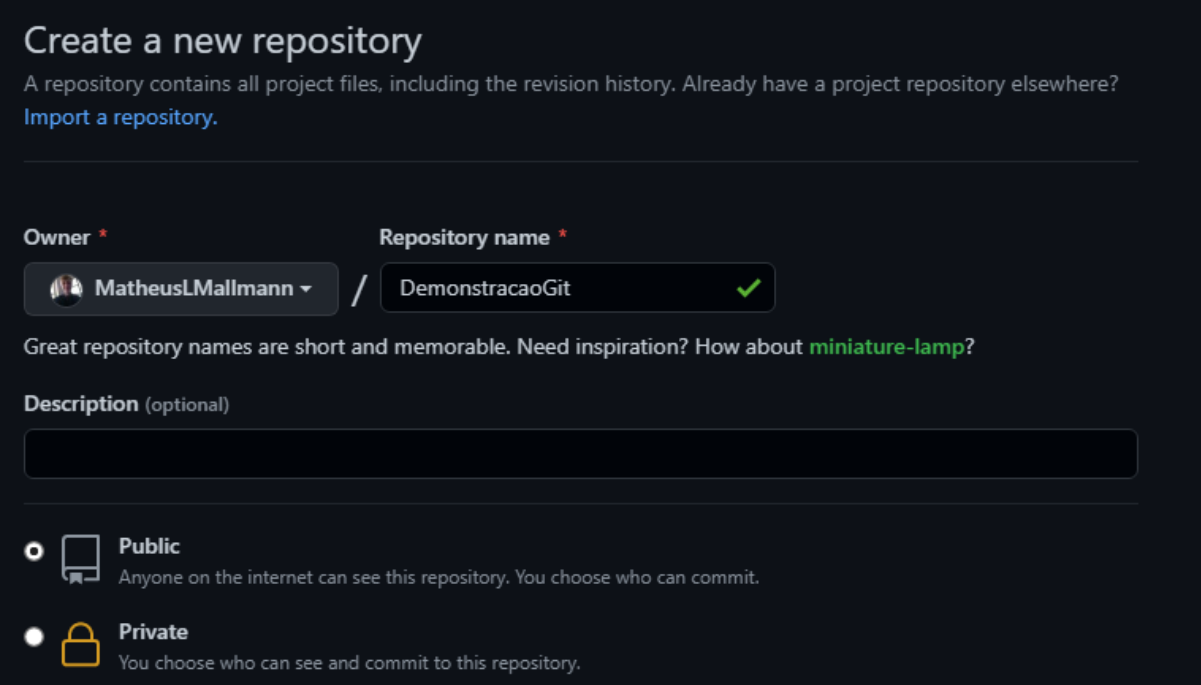
- A ID do commit
- O autor do commit
- A data do commit
- O comentário do commit

```
$ git log
commit 8a3f33baf143346cb6b82c873e56542dcb1f125a (HEAD -> master)
Author: MatheusLMallmann <matheusl.mallmann@gmail.com>
Date:   Sat Oct 30 10:26:27 2021 -0300

    Nosso primeiro commit
```

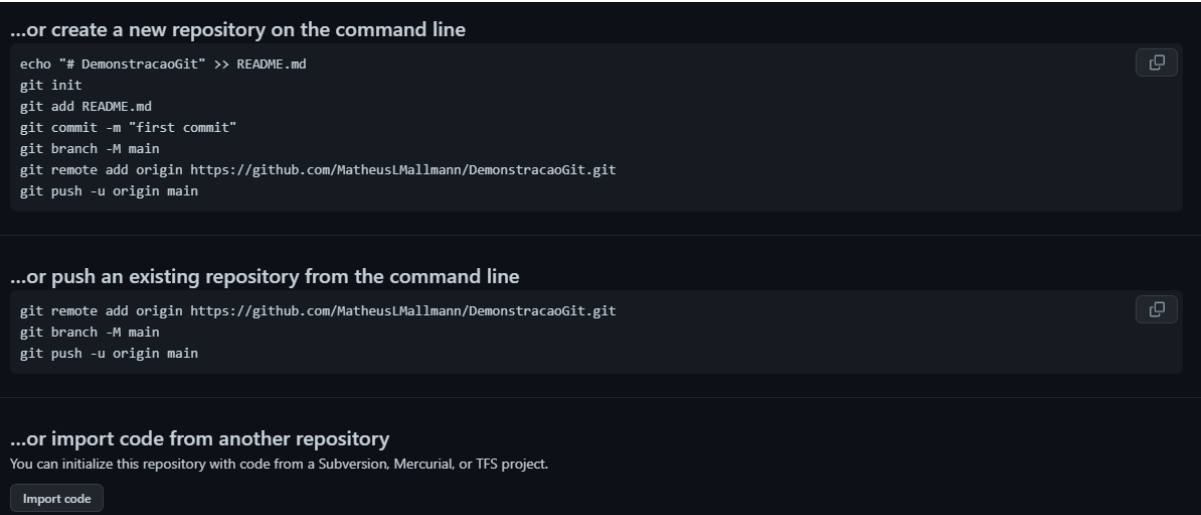
## 4. Adicionando ao repositório remoto

Primeiro de tudo, deve ser criado um repositório no GitHub com o nome que você preferir e de forma pública.



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' with a dropdown menu showing 'MatheusLMallmann' and 'Repository name' with a text input showing 'DemonstracaoGit' and a green checkmark. A hint text says 'Great repository names are short and memorable. Need inspiration? How about [miniature-lamp?](#)'. There is a 'Description (optional)' text area. At the bottom, there are two radio button options: 'Public' (selected) with the description 'Anyone on the internet can see this repository. You choose who can commit.' and 'Private' with the description 'You choose who can see and commit to this repository.'

Ao criar o repositório, o próprio GitHub mostra instruções de como adicionar o seu repositório local ao remoto.



The screenshot shows the GitHub instructions for adding a repository to the command line. It is divided into three sections: 1. '...or create a new repository on the command line' with a copy icon, containing the commands: `echo "# DemonstracaoGit" >> README.md`, `git init`, `git add README.md`, `git commit -m "first commit"`, `git branch -M main`, `git remote add origin https://github.com/MatheusLMallmann/DemonstracaoGit.git`, and `git push -u origin main`. 2. '...or push an existing repository from the command line' with a copy icon, containing the commands: `git remote add origin https://github.com/MatheusLMallmann/DemonstracaoGit.git`, `git branch -M main`, and `git push -u origin main`. 3. '...or import code from another repository' with a description 'You can initialize this repository with code from a Subversion, Mercurial, or TFS project.' and an 'Import code' button.

Enviando ao repositório remoto.

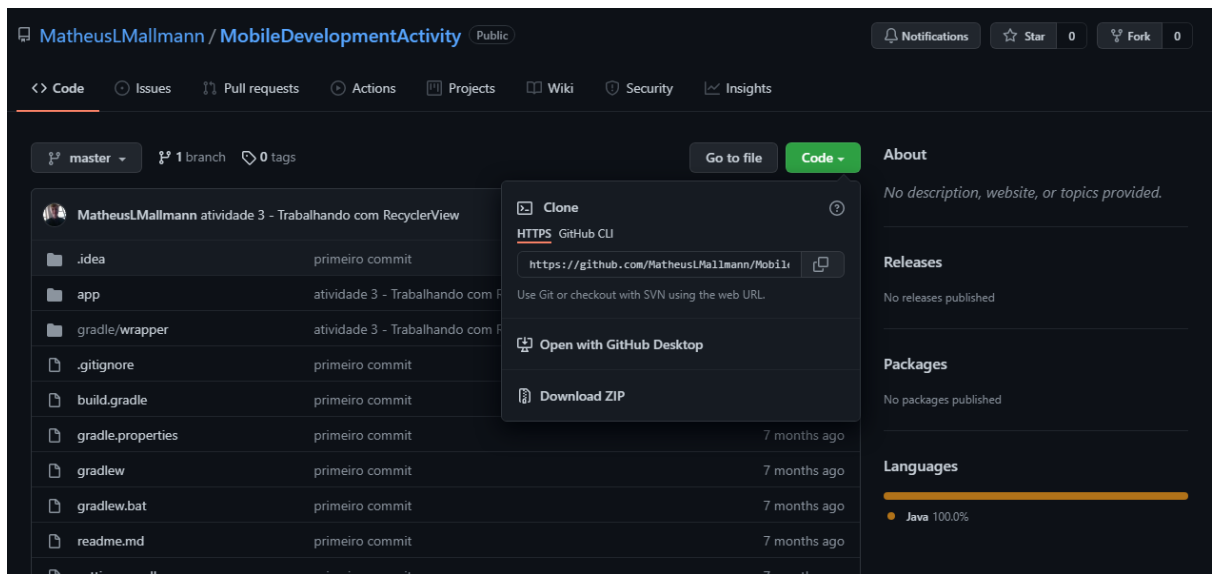
```
$ git remote add origin https://github.com/MatheusLMallmann/DemonstracaoGit.git
mathe@DESKTOP-3L6PSQI MINGW64 ~/IdeaProjects/HelloWorld (master)
$ git branch -M main
mathe@DESKTOP-3L6PSQI MINGW64 ~/IdeaProjects/HelloWorld (main)
$ git push -u origin main
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (25/25), 3.12 KiB | 798.00 KiB/s, done.
Total 25 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/MatheusLMallmann/DemonstracaoGit.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

## 5. Clonando repositório remoto

Quando se deseja copiar um repositório Git existente em um novo diretório local, utilizamos o comando `git clone <link_diretório>`. Clonar um repositório permite que você faça mudanças locais para o repositório antes de commitar e enviar ao repositório remoto.

Clonando um repositório remoto:

- No GitHub, encontre um repositório que você queira clonar.
- Clique no botão verde *Code* e copie o endereço do repositório.



Utilizando o Git Bash, use o comando visto acima junto com o endereço do repositório.

```
$ git clone https://github.com/MatheusLMallmann/MobileDevelopmentActivity.git
Cloning into 'MobileDevelopmentActivity'...
remote: Enumerating objects: 237, done.
remote: Counting objects: 100% (237/237), done.
remote: Compressing objects: 100% (112/112), done.
Receivremote: Total 237 (delta 95), reused 215 (delta 73), pack-reused 0
Receiving objects: 100% (237/237), 166.38 KiB | 3.54 MiB/s, done.
Resolving deltas: 100% (95/95), done.
```

## 6. Branches

Em ambientes corporativos, é comum vários developers compartilharem e trabalharem no mesmo código. Para isso, uma branch permite que cada dev possa ter uma cópia do código base original e isolado, facilitando assim que não altere ou atrapalhe o trabalho dos outros colegas.

Para criar uma nova branch basta utilizar o comando

git branch “nome-nova-branch”

```
mathe@DESKTOP-3L6PSQI MINGW64 ~/OneDrive/Area de Trabalho/PraticasEngenhariaSoftware (master)
$ git branch MatheusMallmann
```

Usando o comando git checkout nome\_branch, é possível ficar alternando entre as branches criadas.

```
mathe@DESKTOP-3L6PSQI MINGW64 ~/IdeaProjects/HelloWorld (NovaBranch)
$ git checkout master
Switched to branch 'master'

mathe@DESKTOP-3L6PSQI MINGW64 ~/IdeaProjects/HelloWorld (master)
$ git checkout NovaBranch
Switched to branch 'NovaBranch'

mathe@DESKTOP-3L6PSQI MINGW64 ~/IdeaProjects/HelloWorld (NovaBranch)
$ |
```

Para alterar o código principal com os arquivos adicionados na nova branch, basta utilizar o comando `git merge nome_branch`, estando na branch principal

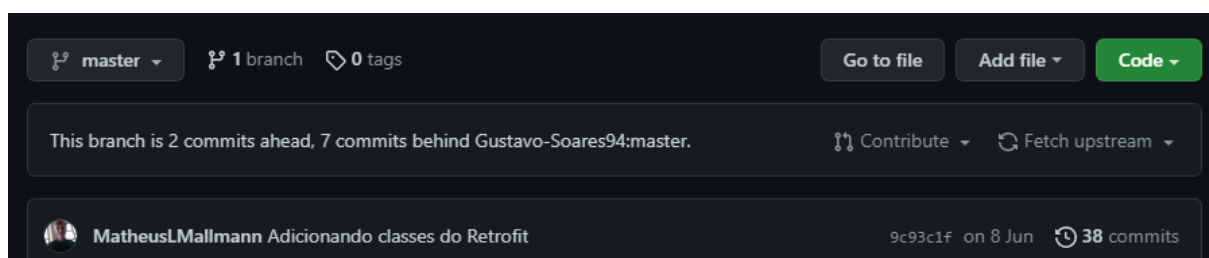
```
git checkout master
git merge NovaMaster
```

## 7. Pull requests

O pull request é fazer uma solicitação ao repositório original para incluir as atualizações feitas na cópia desse repositório. Utilizado para que haja uma análise do código antes que seja implementado no código principal.

Para realizar uma pull request, você deve dar um fork no projeto principal, que nada mais é que uma cópia original do repositório.

Faça as alterações no repositório copiado, vá até o repositório remoto no Github, o próprio Github irá lhe informar caso seu código esteja diferente do original com base na quantidade de commits.



Após isso, é só solicitar a Pull Request.