



Git é um sistema de versionamento de código muito utilizado por desenvolvedores para manter registro do histórico de alterações de um projeto e também para facilitar o trabalho em conjunto com outros devs.

1. Criação de Repositórios

Para que cada projeto possa ser salvo e versionado no Git é necessário a criação de um repositório. Existem diversas plataformas que armazenam versionamento através do Git, os exemplos serão feitos no Github que é a plataforma mais utilizada. Primeiro é necessário criar um repositório local com o seguinte comando:

```
git init
```

Após a criação do repositório local, é necessário adicionar os arquivos ao repositório para que sejam rastreados:

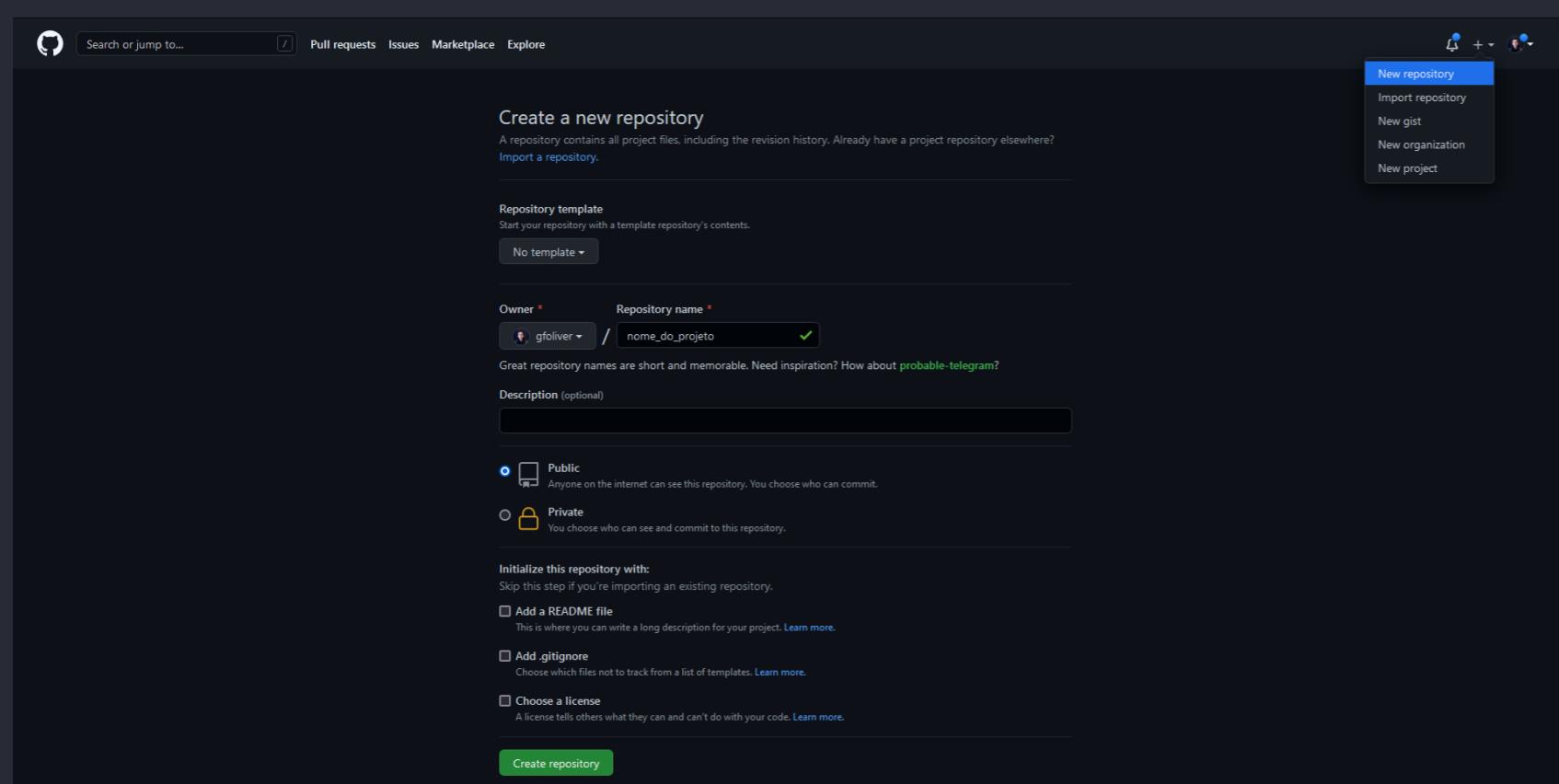
```
git add .
```

Agora que os arquivos do projeto já estão sendo rastreados pelo repositório, é necessário realizar um **commit**, commits são versões do código que serão enviadas ao repositório.

```
git commit -m "Mensagem descrevendo as alterações feitas nesta versão"
```

Após realizar o commit, esta versão do código estará salva no repositório local e pronta para ser enviada para alguma plataforma, para isso, criaremos um repositório remoto na plataforma de escolha (Neste caso utilizaremos o Github, mas existem diversas plataformas, como BitBucket ou GitLab).

No GitHub o botão de criar um repositório fica no canto superior direito da página, ao entrar na tela de criação existem alguns campos obrigatórios e opcionais, é possível iniciar a partir de um template, incluir um arquivo readme, uma licença e inicializar com um arquivo **.gitignore**



.gitignore

O arquivo **.gitignore** é um arquivo de configuração do Git, que informa quais arquivos e/ou diretórios devem ser ignorados pelo repositório. Os arquivos e diretórios presentes neste arquivo não serão rastreados quando realizarmos o **git add**, isto é geralmente utilizado para ignorar arquivos de bibliotecas de terceiros como os **node_modules** do npm, e também para ignorar arquivos de configuração da IDE.

Uma vez que temos o repositório local e também o remoto (Github), podemos enviar os dados locais para a plataforma, para isso é necessário primeiro adicionar o url do repositório remoto no local:

```
git remote add origin https://github.com/gfoliver/nome_do_projeto.git
```

E depois realizar o **push** para atualizar o remoto com os commits locais:

```
git push origin master
```

Guia de Git

Guilherme Fleck Oliveira



2. Clonagem

Quando queremos baixar um repositório de uma plataforma nós utilizamos o comando **clone**



```
git clone https://github.com/gfoliver/nome_do_projeto.git
```



3. Branches

O versionamento do Git é dividido em **branches**, as branches são diferentes linhas do tempo, cada uma com seu histórico de alterações do código, quando uma nova branch é criada ela herda o estado atual do projeto, mas ela não se atualiza automaticamente com nenhuma outra branch.

- Para criar uma branch existem dois comandos possíveis:



```
git branch nome_da_branch  
git checkout -b "nome_da_branch"
```

- Para listar os branches existentes no projeto se utiliza o comando **git branch**
- Para trocar de branch se utiliza o comando **checkout**:



```
git checkout nome_da_branch
```

- Deletar um branch:



```
git branch -D nome_da_branch
```



4. Push

O push é o comando que envia as alterações locais que foram adicionadas e **commitadas** para o repositório remoto.

Sintaxe do comando:



```
git push nome_do_remoto nome_da_branch
```



5. Unindo o código de branches diferentes

Após trabalharmos e enviarmos atualizações em branches separadas, o código precisa ser unido na branch principal, existem duas maneiras principais de fazer esta união, o **merge** e o **pull request**.

5.1 Merge

Para atualizar o branch atual, com o estado de outro branch, se utiliza o comando **git merge**, primeiro é necessário entrar no branch que deve ser atualizado através do comando **git checkout**.

```
git checkout master
```

Estando no branch desejado, se utiliza o merge:

```
git merge nome_da_branch
```

5.2 Pull Request

O Pull Request é uma solicitação para que a branch principal do projeto seja atualizada com as alterações de outra branch, essa solicitação é feita na plataforma remota, ao navegar por um branch que possua alterações comparado ao principal, aparecerá um aviso indicando a possibilidade de comparar as alterações e fazer esta solicitação.

Para criar um Pull Request é necessário dar um nome, também é possível fornecer uma descrição e até mesmo anexar arquivos para explicar as alterações feitas nesta branch.

Após o Pull Request ser criado, ele deve ser analisado e avaliado pelos contribuidores do projeto, é possível adicionar comentários na discussão, labels para categorizar esta solicitação, atribuir contribuidores para analisarem, entre diversas funcionalidades. Uma vez que o Pull Request seja aprovado, o botão de merge irá unir estas alterações a branch principal, caso ele seja rejeitado ele será fechado.