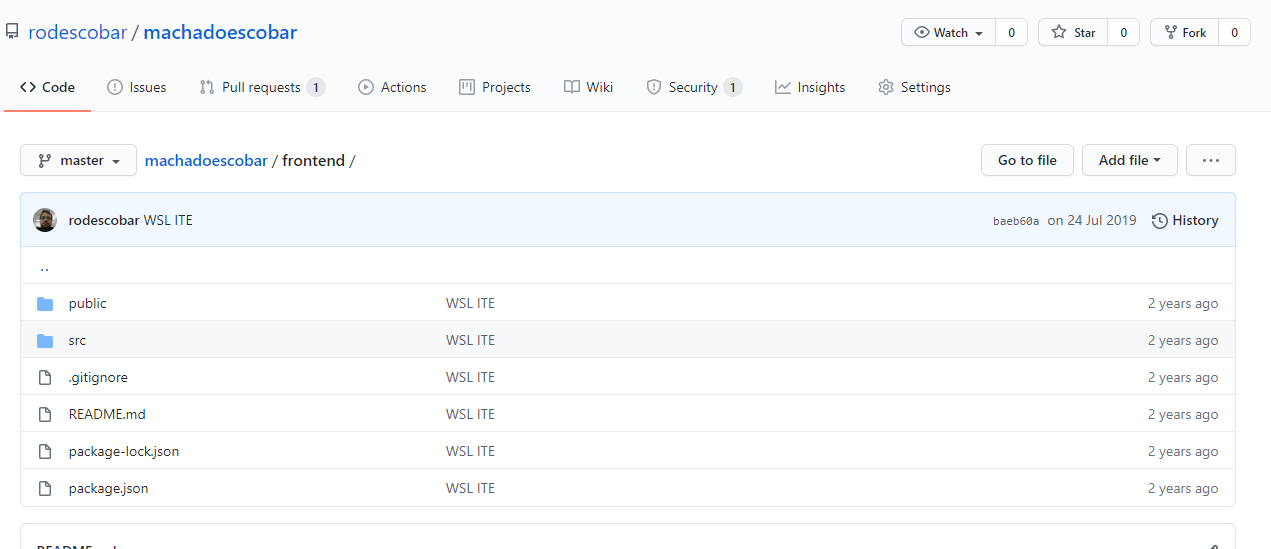
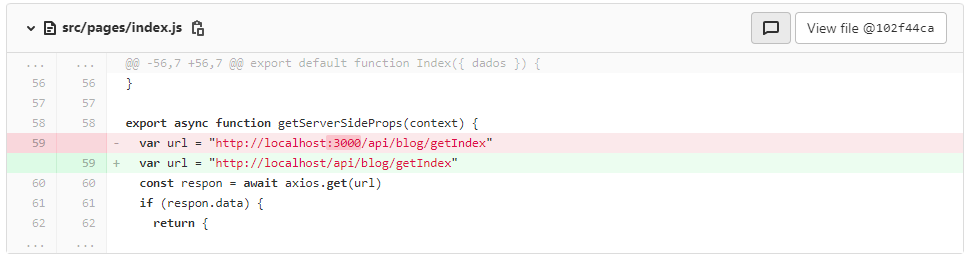
Git

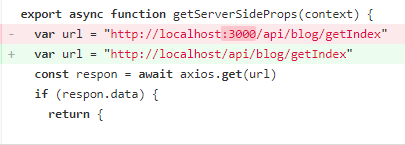
É um sistema de controle de versão com projeto de código aberto, criado em 2005 por Linus Trovalds.

Possui uma arquitetura distribuída, o GIT é um exemplo de sistema de controle de versão distribuído. Isso possibilita que o histórico de versões de uma aplicação não fique armazenado apenas em um local, assim como o Subversion.



Diferenças geradas após o último ***commit***.





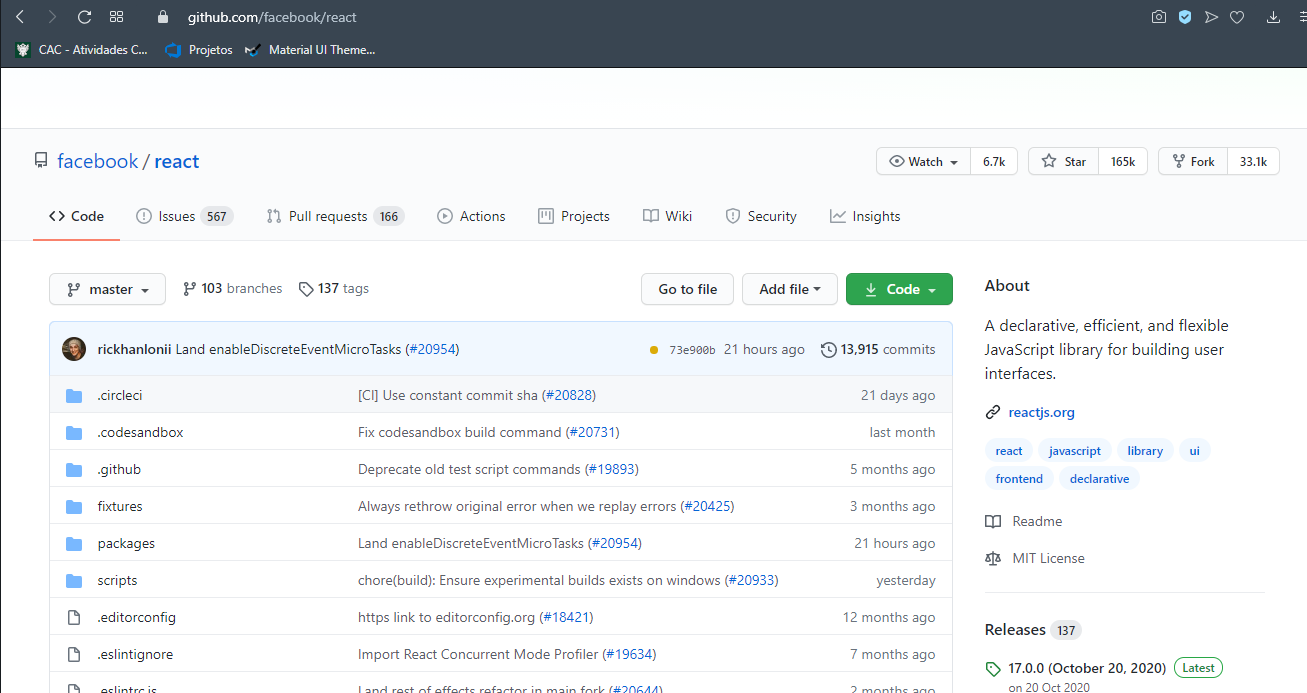
# Iniciando com o git.

<https://github.com>

# Repositório

É o local onde ficarão os arquivos de sua aplicação. Ele pode ser público ou privado ($$$), geralmente pode ser acessado através da URI <https://github.com/>**<usuário>**/**<projeto>**

[**https://github.com/facebook/react**](https://github.com/facebook/react)

****

Repositório do React

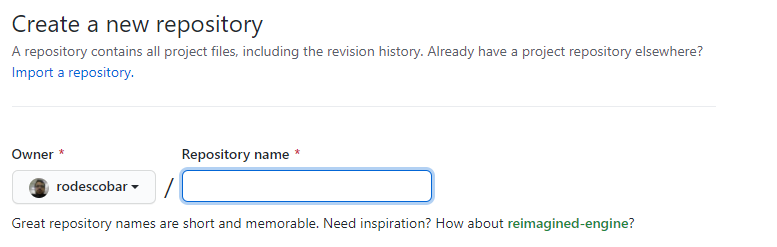
# Criando um repositório:

Os repositórios podem ser criados tanto no micro do desenvolvedor quanto no próprio site do git.

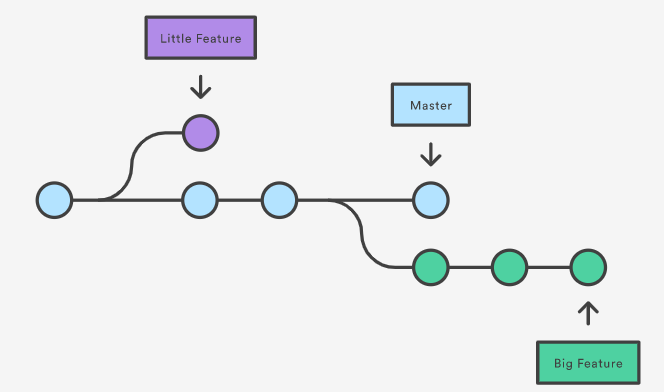
Através da linha de comando

|  |
| --- |
| echo "Aqui serão adicionados todos os conteúdos da aula de ANEV" >> README.md  git init  git add README.md  git commit -m "first commit"  git branch -M main  git remote add origin https://github.com/rodescobar/anev2021.git  git push -u origin main |

Através do site do github.com



# Branch



Uma *branch* no git é um ponteiro para alterações feitas no arquivo do projeto. Em resumo podemos pensar na seguinte situação:

Um grupo de 3 desenvolvedores estão juntos criando um site. O analista do projeto separou cada desenvolvedor em uma área específica, porém, estas áreas se complementam, ou seja, a alteração em uma área poderá espelhar mudanças em outra.

Através deste cenário descrito o analista criou 5 branchs, sendo elas:

1 - A branch **principal**

2 - A branch **teste**

3 - A branch do **desenvolvedor1**

4 - A branch do **desenvolvedor2**

5 - A branch do **desenvolvedor3**

As alterações que o ***desenvolvedor1*** fizer em sua branch no arquivo **main.js**, não causará qualquer mudança ou interferência para os outros desenvolvedores.

Mas o ***desenvolvedor2*** criou uma nova função no mesmo arquivo **main.js** e esta deverá ser replicada para os outros desenvolvedores.

Neste caso, os desenvolvedores 1 e 3, podem atualizar sua *branch* atual a partir das alterações feitas pelo ***desenvolvedor2.***

Desta forma o git analisará os códigos e apenas incluirá no arquivo **main.js** do **desenvolvedor1** a nova função criada pelo **desenvolvedor2.**

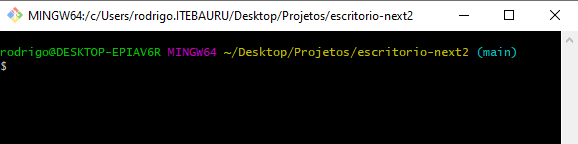
Então as *branchs* separam as atualizações de cada projeto e sempre atualizam após o “*update”* deixando o código mais novo.

E por fim, o analista chefe, pode “juntar” todos os *updates* na *branch desenvolvimento*, fazer os testes e correções se necessário e após isso, adicionar a nova versão em ***produção.***

# Comandos

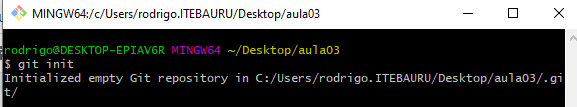
O ideal é que você utilize o Git Bash para a execução de comandos, ele lhe possibilitará analisar em qual ***branch*** seu projeto está de forma mais simplificada.

Você também pode utilizar o próprio ***VSCODE***ou outra *ide.*



|  |
| --- |
| ***git init*** |

Comando responsável por iniciar o versionamento no diretório desejado.



|  |
| --- |
| **git add *<nome do arquivo>*** |

Após as alterações feitas dentro do projeto, você deve informar ao git, quais arquivos devem ser versionados.

Caso você queira versionar todas as atualizações, o comando deve ser seguido por um . (ponto)

***git add .***

Porém se você desejar apenas versionar um único arquivo, você deverá informar o nome e a extensão.

***git add README.md***

Importante lembrar que: O versionamento ainda não está feito, ou seja, o git (local) apenas sabe qual arquivo ele deverá versionar.

|  |
| --- |
| **git commit -m “Texto sobre as alterações”** |

O versionamento das alterações se dará após o *commit*, este comando é o responsável a informar ao git que as alterações acabaram e que o projeto já pode ser versionado.

**Lembre-se:** Sempre verifique se a *branch* atual é a correta. Caso contrário você fará atualizações na *branch* de outro desenvolvedor.

|  |
| --- |
| **git push origin master** |

Após o *commit* ser executado e todas as alterações prontas para serem importadas para o repositório, o comando push enviará as alterações.

**master** é o nome da branch.

|  |
| --- |
| **git remote add origin https://github.com/rodescobar/anev2021.git** |

O comando *git remote add origin* cria um link entre a pasta atual com o seu repositório GIT.

|  |
| --- |
| **git branch -b “teste”** |

O comando *git branch -b* cria uma nova *branch* em seu repositório.

|  |
| --- |
| **git fetch** |

Executado localmente atualiza todos os *branches* existentes no projeto. Exemplo: busca as atualizações mais novas de outro desenvolvedor, porém, não faz alterações em sua *branch.*

|  |
| --- |
| **git checkout branchTeste** |

Altera a *branch* atual para a *branch* branchTeste

|  |
| --- |
| **git pull** |

Faz o download das últimas atualizações existentes no repositório do git

|  |
| --- |
| **git merge branchTeste** |

Sobrepõem atualizações da *branch* branchTeste na branch atual

|  |
| --- |
| **git clone https://github.com/facebook/react.git** |

Faz o clone do repositório git para seu computador.