

Oficina Online de Git



Agência Experimental de Engenharia de Software



Gabriel Fanto Stundner



Estudante de Engenharia de Software



5 anos mexendo com Git



Github: <https://github.com/F4NT0>



Website: <https://f4nt0.github.io/PR0GR4M1NG/>



Youtube: Gabriel Fanto

O que é GIT?



Linus Torvalds

- Criador do Kernel Linux
- Criador do Sistema GIT
- Engenheiro de Software da Finlândia



Prazer, sou o Tux



Bom Humor:
Global Information Tracker

Mau Humor:
Goddamn Idiotic Truckload of Sh*t

Gíria Britânica:
Git = Cabeça Dura, Pessoa que sempre acha que tem razão

Principal a se saber...

GIT ou **Git** é um **Sistema de Controle de Versões** de Arquivos muito utilizado no mundo do Desenvolvimento De Software.

Primeiro Lançamento do Sistema foi em 7 de Abril de 2005

Github, Gitlab e Bitbucket são Sites de **Hospedagem de Códigos Fontes** Pelo Sistema GIT

O que é um Sistema de Controle de Versões?

- ➔ Trabalhamos com Arquivos Textos
- ➔ Cada modificação feita em um arquivo e salvo, é uma versão diferente
- ➔ Várias pessoas mexendo no mesmo arquivo
- ➔ Cada pessoa tem uma versão diferente em seu computador



Arquivo Original

Versão Inicial



Versão Inicial



Pessoa A



Versão Editada por A



Versão Inicial



Pessoa B



Versão Editada por B

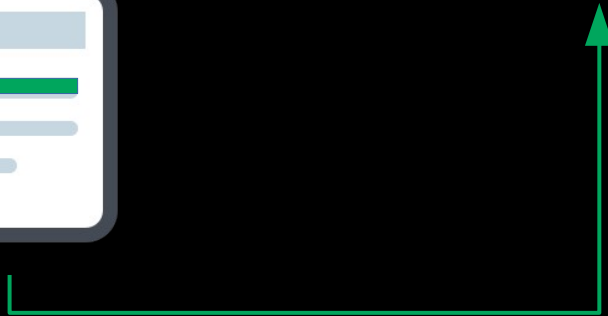
Versão de A



Versão Inicial Externa



Versão de B



Versão de A



Versão Inicial Externa



Versão de B



Conflito de Versões



“Um Sistema de Gerenciamento de Versões vai auxiliar o Desenvolvedor a adquirir, criar, editar, lidar com conflitos e enviar versões de um arquivo para o Meio Remoto.”

O GIT é um Software que nos ajuda a Gerenciar Versões de Arquivos utilizando comandos específicos Via Terminal ou com Ferramentas Gráficas!

Agora, algumas informações importantes.....

Nomenclaturas Importantes!

Repositório ou Repository



Diretório ou Pasta

Projeto



.git

Repositório

Nomenclaturas Importantes!

Branch

- “Ramificação” do Projeto inicial
- A principal Branch do Projeto é `Master` ou `Main`
- Toda vez que formos modificar alguma coisa grande no Arquivo, devemos criar uma Branch nova para não modificar o Arquivo original e perder a Versão Principal!
- Podemos criar quantas Branchs quisermos em um Repositório

Nomenclaturas Importantes!

Commit

- Salvamento das Modificações feitas
- Auxilia para sabermos o que foi modificado
- Podemos voltar se algo der errado

```
commit 254d54f5b11e813911887c1d7fba09af51b9c5e8 ← Id do Commit
Author: Gabriel Fanto Stundner <gabriel.stundner@acad.pucrs.br> ← Autor
Date:   Sun Jul 19 14:41:40 2020 -0300 ← Data do Commit

Initial commit ← Mensagem explicando a modificação
```

Nomenclaturas Importantes!

log

→ Lista de todos os Commits feitos na Branch

head

→ Último Commit feito, o Principal Commit atual

origin

→ nome do Repositório Remoto, fora do seu sistema

Nomenclaturas Importantes!

merge

- Quando juntamos duas Versões em uma única versão
- Durante o Merge pode dar um **Conflito**, ocorre quando Queremos unir duas versões com a mesma linha modificada Onde se deve escolher uma das versões para ser a oficial
- No final do desenvolvimento, unimos todas as Informações Das Branchs na Branch Principal **master/main** para entregar uma Versão Oficial ao Cliente

VAMOS AGORA AO MAIS IMPORTANTE!

Utilizando o Sistema GIT



Criando um Repositório!

Existem duas Maneiras de Criar um Repositório,
Irei apresentar Primeiro a forma mais Simples.

Vamos acessar o Site Github!



<https://github.com/>

Baixando um Repositório

Iremos usar o nosso Primeiro Comando GIT:

`git clone url`

- Chamada dos Comandos GIT, sempre presente
- Comando desejado, sendo a ação que queremos fazer
- Texto que será modificado, somente para sabermos o que devemos por no lugar

Criando nova Branch

→ Forma de Criação Rápida de Branch:

```
git checkout -b nome-branch
```

-b : tag para criar nova Branch e acessar ela direto

Exemplo: `git checkout -b fanto_tsk01_sprint1`

Criando nova Branch

→ Forma para Criar Várias Branchs de uma Vez:

```
git branch nome-branch
```

```
git checkout nome-branch
```

Com o comando Branch crie quantas Branchs quiser

Somente precisa fazer checkout naquela Branch desejada

Criando nova Branch

Atenção!!!!

Quando criamos uma Branch, estamos fazendo uma Cópia de outra Branch, normalmente da Branch **Master/Main**

Não é recomendado criar uma Branch de outra Branch que não seja a mais atualizada, podendo ocorrer inúmeros conflitos e dores de cabeça, por isso é sempre bom saber qual é a Branch Oficial!

Verificando as Branchs

→ Verificando as Branchs existentes Localmente

```
git branch
```

→ Verificando as Branchs existentes Remotamente

```
git branch -r
```


Verificando Status das Modificações



Após fazer modificações em Arquivos, podemos ver o que foi mudado e o que foi criado de novo

`git status`

Arquivos não rastreados irão aparecer em Vermelho

Arquivos rastreados irão aparecer em Verde

Adicionando Arquivos

- ➔ Adicionar arquivos significa pegar todas as modificações E fazer com que eles fiquem prontos para salvar em um Commit
- ➔ Podemos Dizer quais arquivos desejamos adicionar

git add file

git add . = adiciona todos os Arquivos do Diretório

git add *.txt = adiciona todos os Arquivos com extensão .txt

git add exemplo.txt = adiciona somente o Arquivo exemplo.txt

Criando um Commit

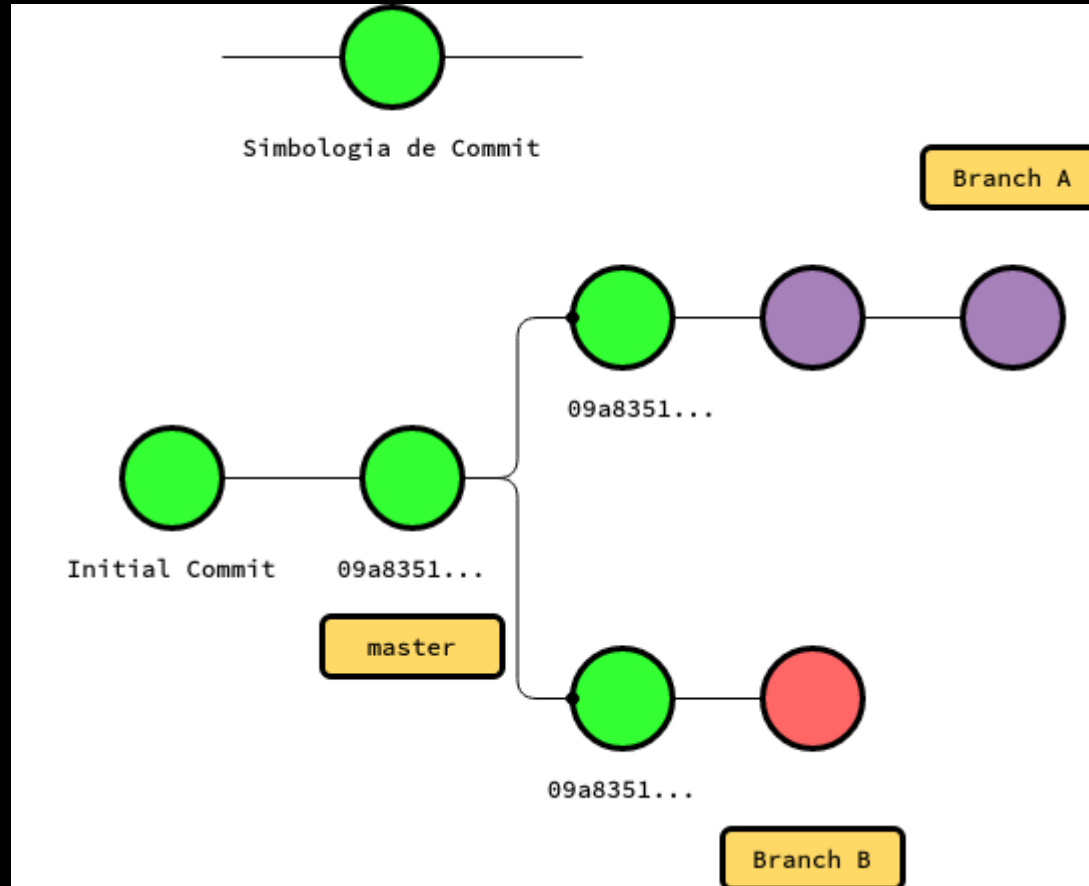
- ➡ Agora iremos salvar todas as Modificações
- ➡ Das 3 formas de Criar um Commit esta é a Principal Forma:

```
git commit -m "message"
```

Forma 2: `git commit` = vai ser aberto um editor de texto para colocar a mensagem

Forma 3: `git commit -am "message"` = se não foi criado arquivos novos Podemos usar esse comando, onde ele Rastreia os Arquivos existentes

Criando um Commit



Enviando Remotamente!

- ➔ Agora que temos salvo as modificações, devemos enviar Para o Repositório Remoto para, por exemplo, os outros Desenvolvedores saibam o que foi feito
- ➔ Sempre tome cuidado para saber qual Branch foi trabalhada, Para evitar que foi enviado modificações no trabalho de outra Pessoa.
- ➔ Cada Repositório Local possui um ou mais Repositórios Remotos, conectado a uma URL

Enviando Remotamente

`git push origin branch`

`origin`: branch remota original, definida na criação dela

`branch`: nome da Branch que iremos enviar as modificações feitas

`git push` : Versão 2.x: branch atual , Versão 1.x: todas as Branchs

`git push --set-upstream origin master`: quando tiver recém utilizado o comando `REMOTE`

Atualizando as Branchs

- Se tiver sido feito modificações na Branch em outro local, devemos atualizar as nossas Branchs locais, para não ocorrer conflitos

`git pull`

- Esse comando vai atualizar todas as Branchs, mas vai mostrar As modificações na Branch atual que você estiver

Verificando os Commits

→ Podemos ver quais Commits já foram feitos na nossa Branch

`git log`

→ As informações apresentadas são como mostrados abaixo:

```
commit 254d54f5b11e813911887c1d7fba09af51b9c5e8 ← Id do Commit
Author: Gabriel Fanto Stundner <gabriel.stundner@acad.pucrs.br> ← Autor
Date:   Sun Jul 19 14:41:40 2020 -0300 ← Data do Commit

    Initial commit ← Mensagem explicando a modificação
```


Revertendo um Commit

- ➔ Caso Necessite retornar um Commit para um anterior
- ➔ Muito usado quando necessita retornar para uma versão antiga

`git revert id-commit`

- ➔ O `id-commit` pode ser somente os primeiros 6 dígitos do id
- ➔ Esse comando assim, vai abrir um editor de texto para adicionar a mensagem que deseja para o novo commit que vai ser criado
- ➔ A tag `--no-edit` faz com que não seja necessário escrever uma Mensagem no editor, deixando a mensagem base

Revertendo um Commit

- Caso deseje saber qual commit voltar, pode usar o comando `checkout` para saber qual commit voltar.

`git checkout id-commit`

- Esse retorno a um commit anterior é somente temporário, para se poder ver quais modificações foram feitas e se esse commit é o qual deseja voltar

Revertendo Vários Commits

- ➔ Agora, um caso que necessite voltar vários commits de uma vez
- ➔ Usamos a palavra `HEAD` para dizermos que estamos pegando o Commit atual

`git revert HEAD~n`

- ➔ Por exemplo, queremos voltar dois Commits, então Pegamos o Commit `HEAD` mais 1 commit: `HEAD~1`
- ➔ Utilize um Editor de Texto ou IDE para lidar com futuros Conflitos
- ➔ `add` e depois faça um novo `commit`

Entendendo o Merge!

Merge é um ato onde queremos pegar as modificações de uma Branch Criada por um Desenvolvedor e adicionar essas Modificações na nossa Branch Oficial **master/main** ou adicionar em outra Branch desejada

Existem duas formas de fazermos um Merge, a segunda forma irei Explicar quando formos falar mais sobre o **GITHUB**

Para mexer com Merge, tenha em mãos um Editor de Texto ou IDE para poder lidar com os **Conflitos** que podem ocorrer

Esses **Conflitos** ocorrem quando nas duas Branchs acontecem de ter Modificado a mesma Linha de um Arquivo, onde o Desenvolvedor Deve saber qual dessas Linhas vai ser a Oficial do Merge

Meu Editor de Texto!



Visual Studio Code

Link para Download: <https://code.visualstudio.com/>

Merge por comando

- Verifique primeiro qual a Branch que você está Trabalhando
- Verifique o nome da Branch que deseja pegar as modificações
- Exemplo: trabalhei na branch `teste-1` e desejo enviar as Modificações para a branch `master`
- Utilizamos o comando `checkout` para a Branch `master` da branch `teste-1`

```
git checkout master
```

Merge por comando

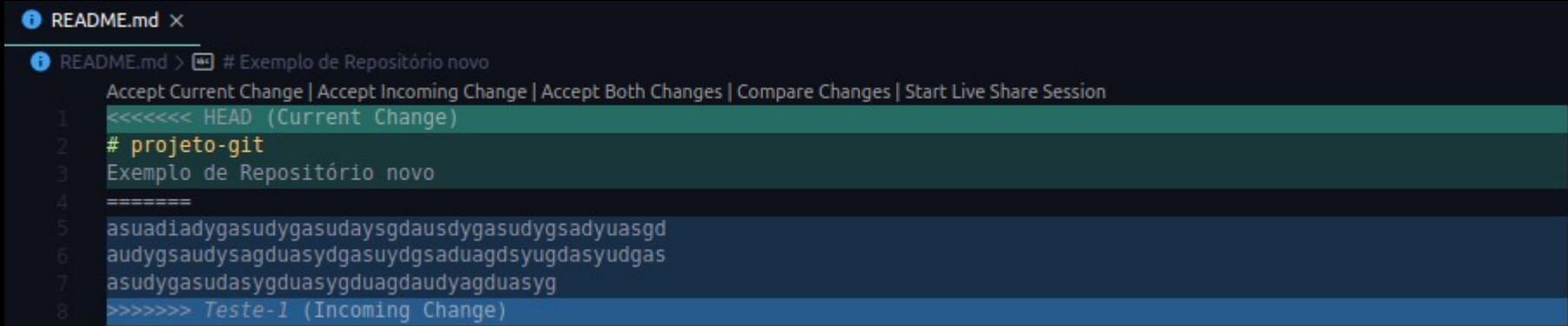
→ Agora que estamos na Branch Oficial, podemos usar o comando:

`git merge branch`

- Esse comando vai pegar todos os commits e modificações da `branch` e colocar essas modificações na branch que estamos atualmente
- No nosso caso de exemplo, será `git merge teste-1`
- Se tiver um arquivo onde nas duas Branchs tiverem a mesma linha Modificada irá ocorrer um conflito, onde no VSCODE ele te mostra Quais Conflitos foram e como resolver eles.

Resolvendo Conflitos

➡ Pelo VSCODE:



```
README.md x
README.md > # Exemplo de Repositório novo
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Session
1 <<<<<<< HEAD (Current Change)
2 # projeto-git
3 Exemplo de Repositório novo
4 =====
5 asuadiadygasudygasudaysgdausdygasudygsadyuasgd
6 audygsaudysagduasydgsadydgsaduagdsyugdasyudgas
7 asudygasudasygduasygduagdaudyagduasyg
8 >>>>>> Teste-1 (Incoming Change)
```

- ➡ Accept Current Change: Modificação Feita na Branch Atual
- ➡ Accept Incoming Change: Modificação vinda da Branch pedida
- ➡ Accept Both Change: Aceita as duas Modificações
- ➡ Após selecionar a sua opção, faça um `commit` novo do Merge

Utilizando Remote

- ➔ `remote` é um comando do GIT que serve para vincular um Repositório Remoto com um Repositório Local
- ➔ Remote possui os principais comandos `add`, `remove`
- ➔ Podemos usar o remote para iniciar um Repositório Local e enviar as modificações para um url específico
- ➔ Podemos usar o remote também para enviar modificações para mais de um repositório de uma vez, em diferentes sites ou no mesmo.

Utilizando Remote

Adicionando um Repositório Remoto:

```
git remote add origin url
```

- `origin` pode ser outro nome, mas quando incia um Repo vazio Localmente é bom usar o nome `origin`
- `url` é o link para o Repositório Remoto

Utilizando Remote

Removendo um Repositório Remoto:

```
git remote remove origin
```

→ `origin` é o nome do remote adicionado, podendo mudar

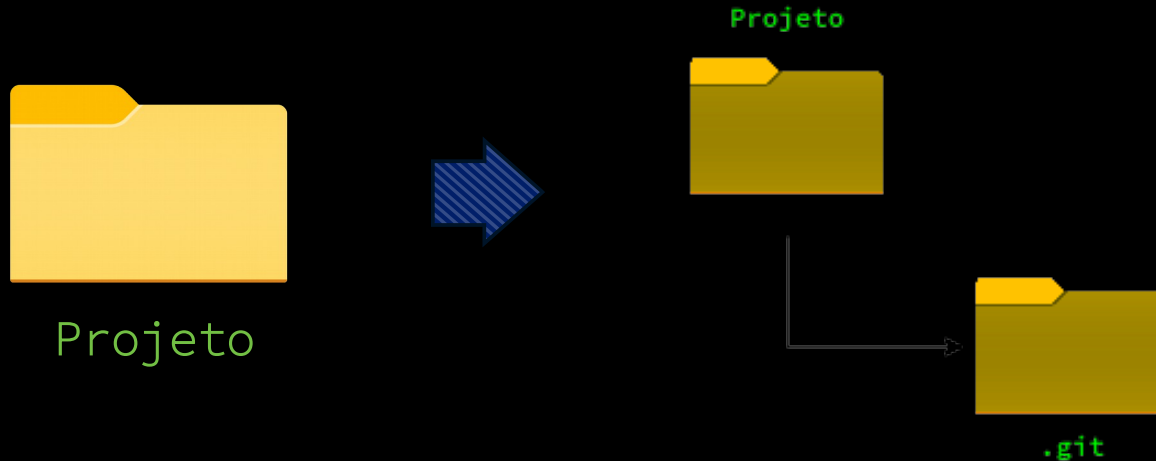
→ Mas como saber o nome do remote?

```
git remote -v
```

Entendendo Remote, agora podemos criar um Repositório de Forma Local...

→ Podemos transformar um Diretório em um Repositório, utilizando o comando:

git init



Temos Agora que pegar o URL do Repositório Remoto e configurar ele com o nosso Repositório Local

```
git remote add origin url
```

Para enviar Modificações pela primeira vez, devemos Usar outra forma de `push`

```
git push --set-upstream origin master
```

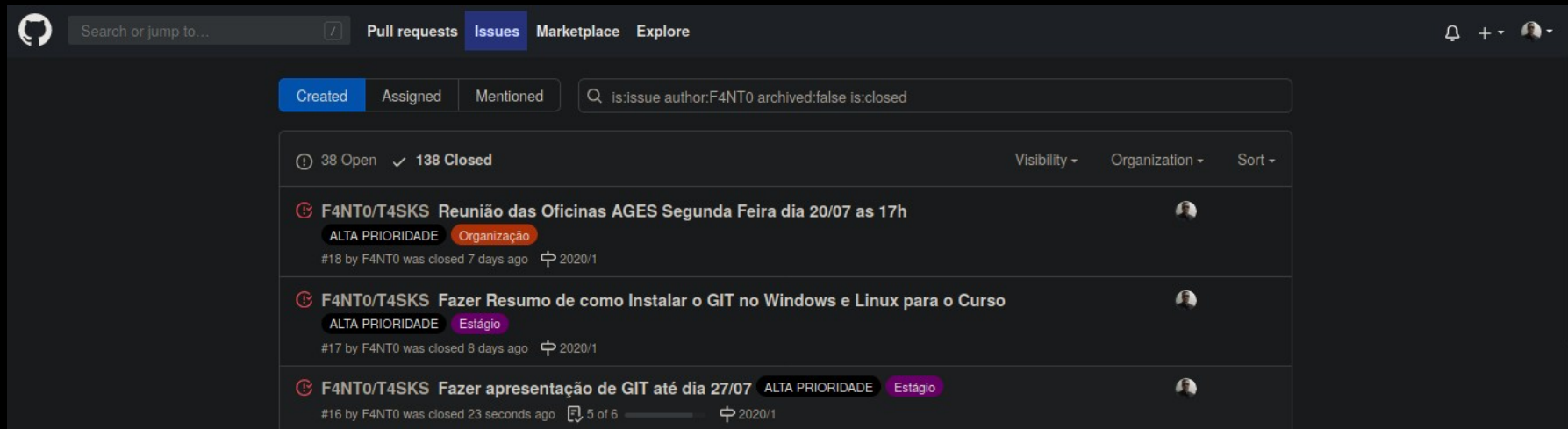
Vamos Falar agora um Pouco mais sobre
O **GITHUB!**



Entendendo algumas Palavras usadas em Repositórios Remotos...

Issues

Issues são uma ferramenta para organizar as tarefas que devem ser Feitas em um Projeto, onde pode ser aberto uma comunicação com o Desenvolvedor por pessoas de fora ou o Gerente Projeto.



The screenshot shows the GitHub interface for the 'F4NT0/T4SKS' repository. The 'Issues' tab is selected in the top navigation bar. Below the navigation bar, there are filters for 'Created', 'Assigned', and 'Mentioned'. A search bar contains the query 'is:issue author:F4NT0 archived:false is:closed'. The main content area displays a list of issues. The first issue is titled 'Reunião das Oficinas AGES Segunda Feira dia 20/07 as 17h' and is marked as 'ALTA PRIORIDADE' (High Priority) and 'Organização' (Organization). The second issue is titled 'Fazer Resumo de como Instalar o GIT no Windows e Linux para o Curso' and is marked as 'ALTA PRIORIDADE' and 'Estágio' (Stage). The third issue is titled 'Fazer apresentação de GIT até dia 27/07' and is also marked as 'ALTA PRIORIDADE' and 'Estágio'. Each issue entry includes a status indicator (e.g., '38 Open', '138 Closed'), a visibility dropdown, an organization dropdown, and a sort dropdown. The issues are listed in descending order of creation time, with the most recent issue at the bottom.

Search or jump to... Pull requests Issues Marketplace Explore

Created Assigned Mentioned Q is:issue author:F4NT0 archived:false is:closed

38 Open ✓ 138 Closed Visibility Organization Sort

F4NT0/T4SKS Reunião das Oficinas AGES Segunda Feira dia 20/07 as 17h
ALTA PRIORIDADE Organização
#18 by F4NT0 was closed 7 days ago 2020/1

F4NT0/T4SKS Fazer Resumo de como Instalar o GIT no Windows e Linux para o Curso
ALTA PRIORIDADE Estágio
#17 by F4NT0 was closed 8 days ago 2020/1

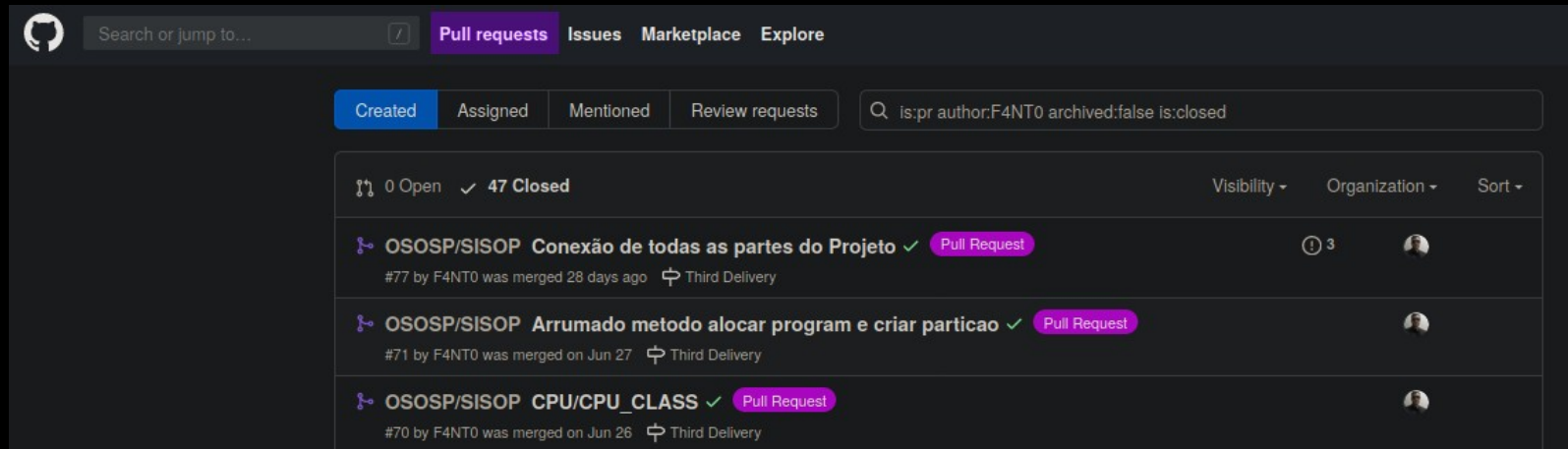
F4NT0/T4SKS Fazer apresentação de GIT até dia 27/07 ALTA PRIORIDADE Estágio
#16 by F4NT0 was closed 23 seconds ago 5 of 6 2020/1

Entendendo algumas Palavras usadas em Repositórios Remotos...

Pull Request

Pull Request é uma organização para pedir Merge de Duas Branchs
Em algumas estruturas de Organização, dois outros Desenvolvedores
Devem avaliar as Modificações antes de Fazer o Merge

Pelo Pull Request se pode fazer Merge direto pelo Github, sem
se Preocupar com resolver problemas por um editor de texto Local



The screenshot shows the GitHub interface for the OSOSP/SISOP repository. The top navigation bar includes the GitHub logo, a search bar, and tabs for Pull requests (selected), Issues, Marketplace, and Explore. Below the navigation bar, there are filters for 'Created', 'Assigned', 'Mentioned', and 'Review requests'. A search bar contains the query 'is:pr author:F4NT0 archived:false is:closed'. The main content area displays a list of pull requests. The first pull request is titled 'OSOSP/SISOP Conexão de todas as partes do Projeto' and is marked as merged. It was created by F4NT0 and merged 28 days ago. The second pull request is titled 'OSOSP/SISOP Arrumado metodo alocar program e criar particao' and is also marked as merged. It was created by F4NT0 and merged on Jun 27. The third pull request is titled 'OSOSP/SISOP CPU/CPU_CLASS' and is marked as merged. It was created by F4NT0 and merged on Jun 26. Each pull request entry includes a link to the pull request, a checkmark indicating it is merged, and a 'Pull Request' label.

Visibility	Organization	Sort
0 Open	✓ 47 Closed	
OSOSP/SISOP Conexão de todas as partes do Projeto ✓	Pull Request	3
#77 by F4NT0 was merged 28 days ago	Third Delivery	
OSOSP/SISOP Arrumado metodo alocar program e criar particao ✓	Pull Request	
#71 by F4NT0 was merged on Jun 27	Third Delivery	
OSOSP/SISOP CPU/CPU_CLASS ✓	Pull Request	
#70 by F4NT0 was merged on Jun 26	Third Delivery	

Estrutura de Organização de Commits Com Issues!

- Uma boa Prática de Criação de Commits para envio ao Github É colocando qual Issue esta sendo trabalhada
- Cada Issue e Pull Request tem um ID único junto com uma Hasthtag #
Toda vez que usarmos uma # dentro do Github ele automaticamente Vai vincular com uma Issue ou Pull Request

Se: Issue **#19** Organizar Quarto

Então: `git commit -m "#19 Limpado o Armário"`

- Quando enviado ao Github, o **#19** se torna um Link

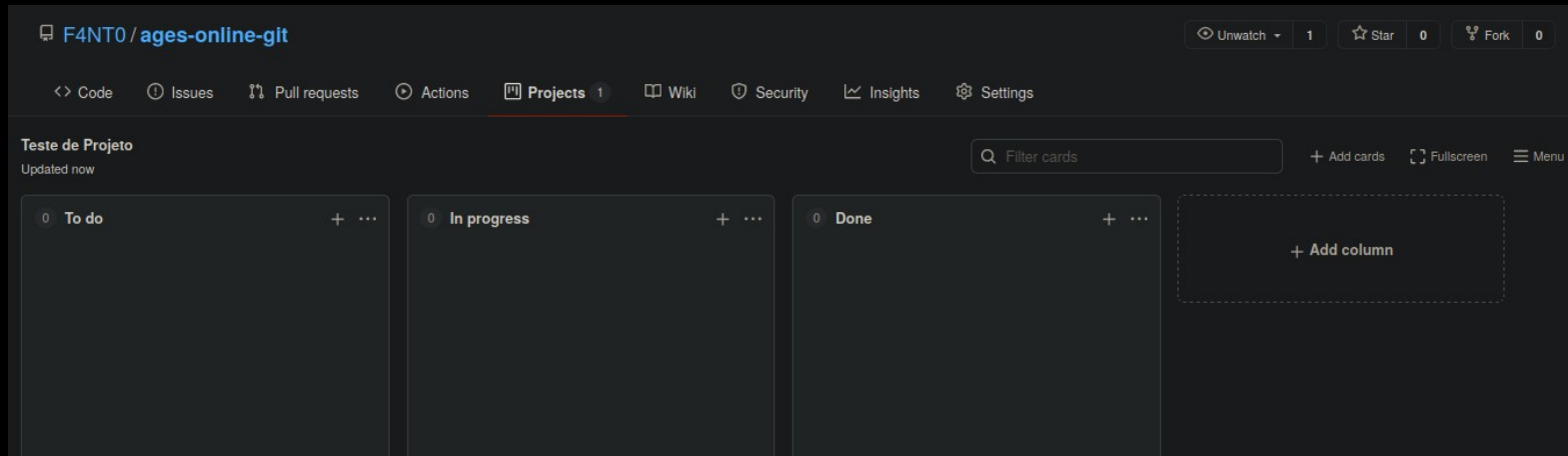
Projects no Github

- Projects é uma organização no Github das Issues de Pull Requests
- Segue um Estilo ágil de Kanban, onde Possui 3 separações Básicas:

To do: 0 que Precisa ser feito

In progress: 0 que está sendo feito

Done: 0 que foi feito



Textos em Repositórios Remotos!

- Todo Repositório Remoto possui uma Wiki, onde escrevemos tudo que é necessário em uma **Linguagem de Marcação** chamada **Markdown**
- Uma Wiki é também um Repositório, onde podemos também baixar e escrever em um Editor de Texto
- Além disso, todos os Arquivos com extensão **.md** são arquivos Em Markdown, como o Arquivo inicial **README.md**

Para mais informações e como utilizar a Linguagem,
Acessem Este Link:

https://f4nt0.github.io/PR0GR4M1NG/pages/tut_pages/home.html

Muito Obrigado e Bom Desenvolvimento!

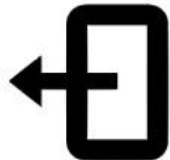
In case of fire



1. `git commit`



2. `git push`



3. `leave building`