



# git

## GIT - SISTEMA DE CONTROLE DE VERSÕES

---

SENAC RIO – LUIS PAULO JR

# O QUE É GIT

---

- Git é um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo.

<https://pt.wikipedia.org/wiki/Git>

# UMA BREVE HISTÓRIA DO GIT

---

- Assim como muitas coisas boas na vida, o Git começou com um tanto de destruição criativa e controvérsia acirrada. O kernel (núcleo) do Linux é um projeto de software de código aberto de escopo razoavelmente grande. Durante a maior parte do período de manutenção do kernel do Linux (1991-2002), as mudanças no software eram repassadas como patches e arquivos compactados. Em 2002, o projeto do kernel do Linux começou a usar um sistema DVCS proprietário chamado BitKeeper.
- Em 2005, o relacionamento entre a comunidade que desenvolvia o kernel e a empresa que desenvolvia comercialmente o BitKeeper se desfez, e o status de isento-de-pagamento da ferramenta foi revogado. Isso levou a comunidade de desenvolvedores do Linux (em particular Linus Torvalds, o criador do Linux) a desenvolver sua própria ferramenta baseada nas lições que eles aprenderam ao usar o BitKeeper. Alguns dos objetivos do novo sistema eram:
  - Velocidade
  - Design simples
  - Suporte robusto a desenvolvimento não linear (milhares de branches paralelos)
  - Totalmente distribuído
  - Capaz de lidar eficientemente com grandes projetos como o kernel do Linux (velocidade e volume de dados)
- Desde sua concepção em 2005, o Git evoluiu e amadureceu a ponto de ser um sistema fácil de usar e ainda assim mantém essas qualidades iniciais. É incrivelmente rápido, bastante eficiente com grandes projetos e possui um sistema impressionante de branching para desenvolvimento não-linear.

<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Uma-Breve-Hist%C3%B3ria-do-Git>

# INSTALANDO O GIT

---

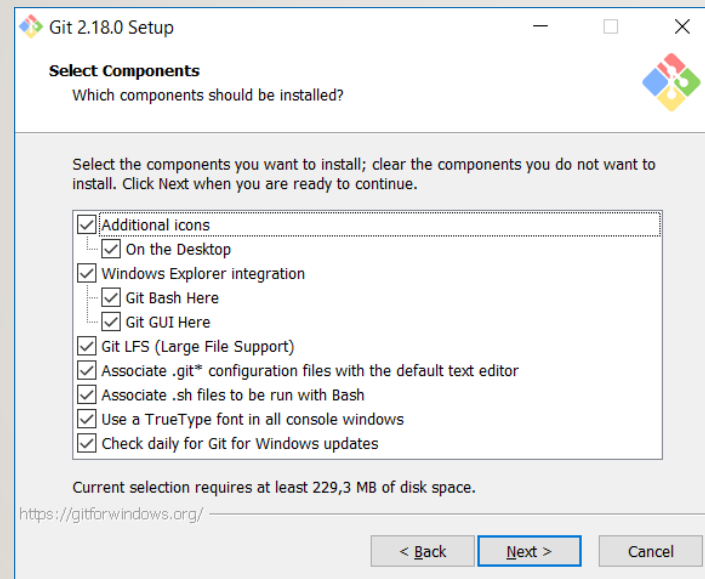
- 1ª Etapa - Baixe o instalador no endereço: <http://msysgit.github.com>



<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>

# INSTALANDO O GIT

- 2ª Etapa - Deixe como está e clique em Next



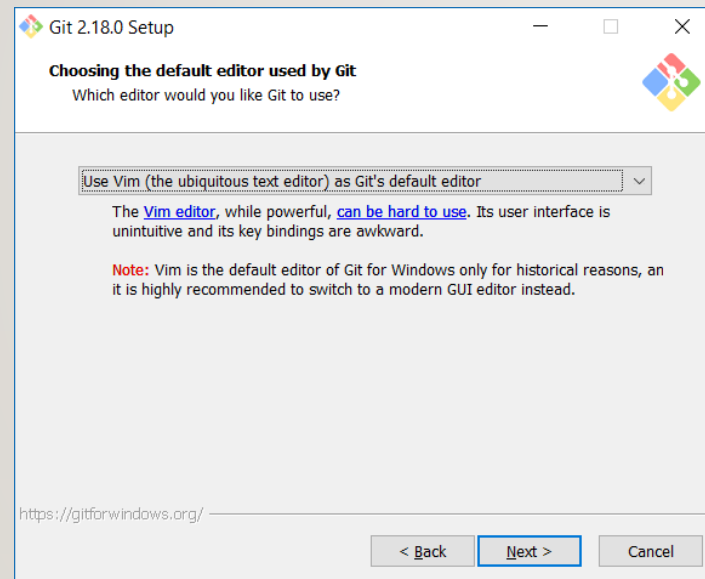
<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>



# INSTALANDO O GIT

---

- 3ª Etapa - Deixe o *Use Vim* selecionado mesmo e clique em Next

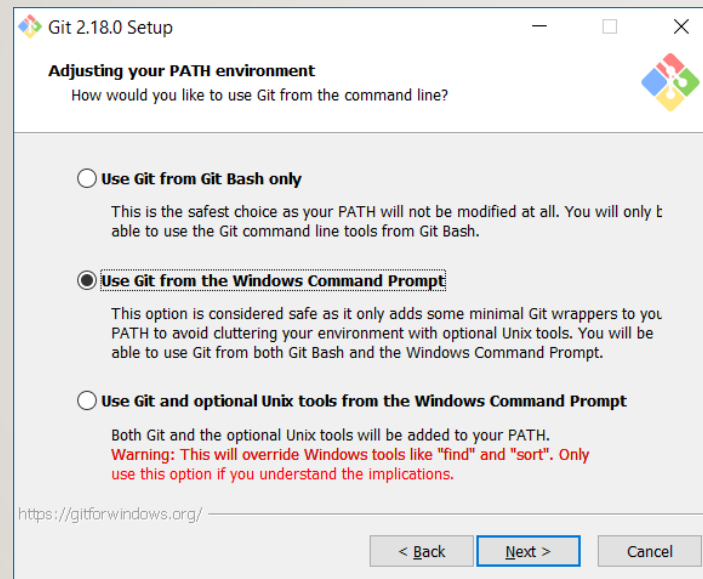


<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>

# INSTALANDO O GIT

---

- 4ª Etapa - Deixe selecionado *Use Git from the Windows Command Prompt* e clique em Next

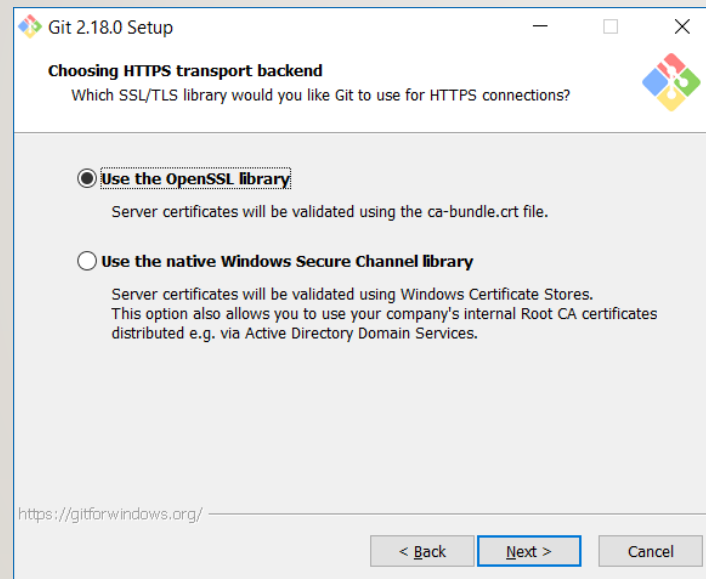


<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>

# INSTALANDO O GIT

---

- 5ª Etapa - Deixe selecionado *Use the OpenSSL library* e clique em Next



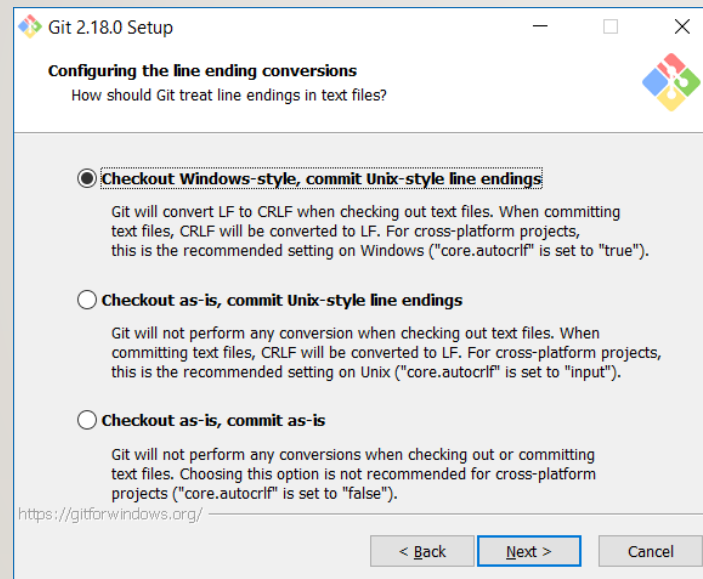
<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>



# INSTALANDO O GIT

---

- 6ª Etapa - Deixe selecionado *Checkout Windows-style* e clique em Next

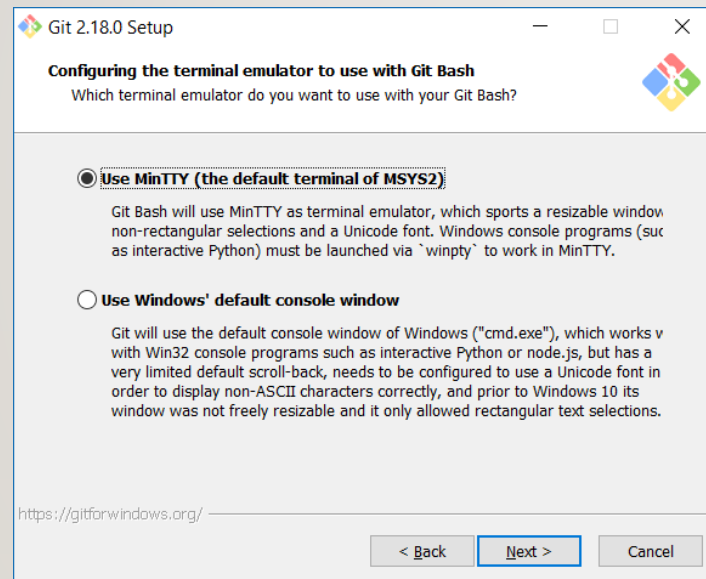


<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>

# INSTALANDO O GIT

---

- 7ª Etapa - Deixe selecionado *Use MinTTY* e clique em Next

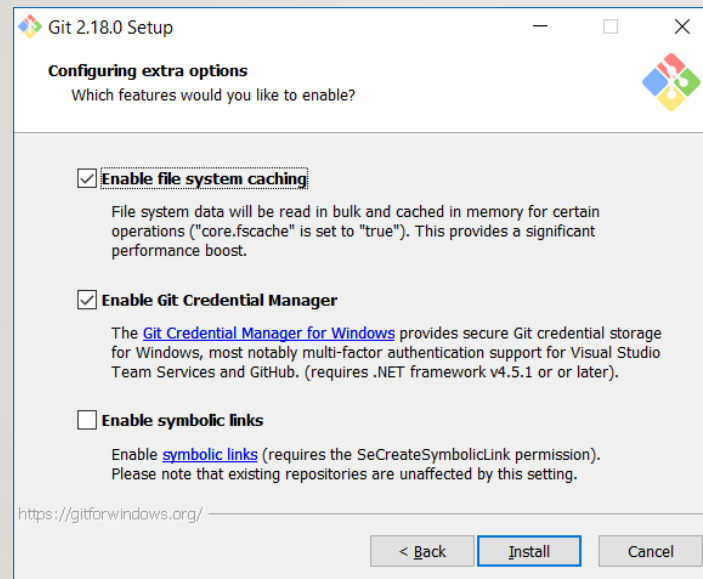


<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>

# INSTALANDO O GIT

---

- 8ª Etapa - Deixe selecionado *Use MinTTY* e clique em *Install*



<https://git-scm.com/book/pt-br/v1/Primeiros-passos-Instalando-Git>

# PRINCIPAIS COMANDOS DO GIT

---

- `git config`
  - Você pode usá-lo para configurar o nome do autor, endereço de e-mail, formatos de arquivo e muito mais.

```
git config user.name "Luis Paulo Jr"  
git config user.email "luis@senac.com"
```

- Podemos usar o parâmetro `--global` para aplicar essa configuração de autor para todos os projetos.

```
git config --global user.name "Luis Paulo Jr"  
git config --global user.email "luis@senac.com"
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git blame <nome_arquivo>`
  - Você visualizar cada linha do arquivo que foi editada por último e seu autor.

```
git blame classes/ProdutosDAO.php
```

- Podemos limitar essa visualização usando parâmetro `-L` e definir a faixa de linhas.

```
git blame -L 12,22 classes/ProdutosDAO.php
```



# PRINCIPAIS COMANDOS DO GIT

---

- `git init`
  - Usando esse comando, você garante que seu repositório git seja inicializado e crie o `.git` diretório inicial em um projeto novo ou existente. A saída será a seguinte:

```
Initialized empty Git repository in /path/.git/
```

- Você pode desfazer um `git init` com `rm -rf .git` no Linux ou `RD /s .git` .

```
rm -rd .git -- Linux  
RD /s .git -- Windows
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git clone <path>`
  - Cria uma cópia de um repositório Git remoto para o seu repositório local.

```
git clone /path/repository
```

- Além disso, você pode adicionar o **local original** como um controle remoto para que possa recuperá-lo facilmente e enviá-lo se tiver permissões. Uma vez clonado o projeto, você pode começar a trabalhar nele.

```
git clone git@github:user/repository.git
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git clone <path>`
  - Você pode clonar uma ramificação(branch) específica de cada vez `git clone -b <nome_branch><repository_url>`:

```
git clone -b nome_branch git@github:user/repository.git
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git add <nome_arquivo>`
  - Adicione um ou mais arquivos em seu working directory ao seu index.

```
git add index.php
```

- Podemos usar o parâmetro `--all` para adicionar todos os arquivos de uma só vez.

```
git add --all
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git commit`
  - Pegue todas as suas alterações escritas no index para o branch HEAD com -m “mensagem”.

```
git commit -m "Adicionado as classes modelo ao projeto"
```



# PRINCIPAIS COMANDOS DO GIT

---

- `git status`
  - Mostra a diferença de status entre um index e arquivos **working directory**. Lista os arquivos **modified** que você alterou, os **untracked** que estão apenas em seu diretório de trabalho e os arquivos **staged** que estão em estágio, pois estão prontos para serem comitados.

*On branch master*

*Initial commit*

*Untracked files:*

*(use "git add <file>..." to include in what will be committed)*

*File\_name*

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

# PRINCIPAIS COMANDOS DO GIT

---

- `git ls-files`
  - Liste todos os arquivos persistidos no git.

```
git ls-files
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git remote`
  - Mostra todas as versões remotas do seu repositório.

```
git remote
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git checkout <nome_branch>`
  - Você pode alternar de uma ramificação(branch) existente para outra ou criar uma nova ramificação e alternar para ela:

```
git checkout -b <nome_branch>
```

# PRINCIPAIS COMANDOS DO GIT

---

- git branch
  - Você pode listar todas as ramificações(branches) existentes, incluindo repositórios remotos, usando -a ou criar uma nova ramificação, se um nome de ramificação for fornecido:

```
git branch -- lista os branches do projeto  
git branch -a -- lista os branches locais e remotos  
git branch slave1 -- cria um novo branch
```



# PRINCIPAIS COMANDOS DO GIT

---

- `git push`
  - Envia todas as alterações para o repositório remoto:

```
git push origin <nome_branch>
```

- Você também pode excluir uma ramificação do seu repositório remoto.

```
git push origin :<nome_branch>
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git pull`
  - Busque e mescle suas alterações no repositório remoto para o seu diretório de trabalho:

```
git pull
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git merge <nome_branch>`
  - Mescla uma ou mais ramificações em sua ramificação atual e, se não houver conflitos, ela criará automaticamente um novo commit.

```
git merge <nome_branch>
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git diff <nome_branch>`
  - Mostrar alterações entre sua árvore de trabalho e o índice, entre duas ramificações ou alterações entre dois arquivos no disco.

```
git diff <local_branch> <remote_branch>
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git reset`
  - Redefina seu índice e diretório de trabalho para o estado de seu último commit.

```
git reset --soft origin/master
```

- Existe 2 tipos de reset:

*Irá desfazer as alterações que você fez até agora!*  
*git reset --hard*

*Se você quiser manter suas alterações*  
*git reset --soft*



# PRINCIPAIS COMANDOS DO GIT

---

- `git revert`
  - Funciona de maneira muito semelhante *git reset*, mas em vez de redefini-lo, criará um novo commit que reverte tudo o que foi introduzido pelo commit accidental.

```
git revert
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git tag`
  - Você pode usar a marcação para definir uma alteração significativa, como uma versão.

```
git tag 1.0.0 <commit_id>
```

# PRINCIPAIS COMANDOS DO GIT

---

- `git log`
  - Mostra uma listagem de commits em uma ramificação com detalhes correspondentes.

```
commit 26a9b395ffa7fbd0890d9b99cb6cf298e10ca2c0
Author: lpjunior <prof.lpjunior@gmail.com>
Date:   Wed Jul 25 19:06:59 2018 -0300
```

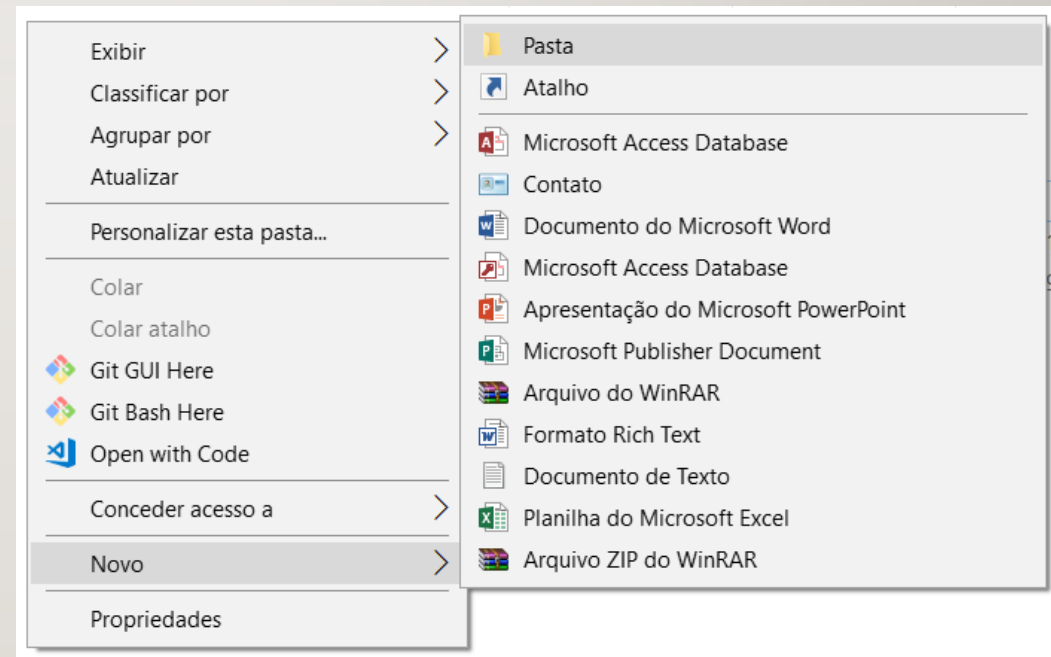
```
    Update consultas.php
```

- Podemos usar o parâmetro `--stat` para um log com detalhamento.

```
git log --stat
```

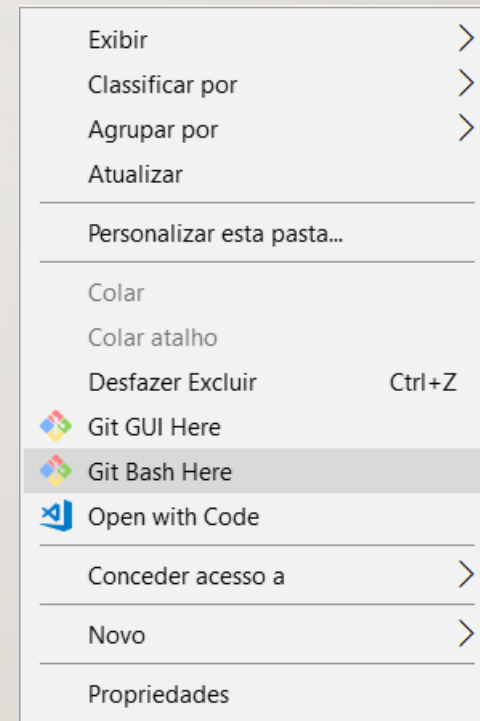
# ENVIANDO UM PROJETO PARA O GIT

- 1ª Etapa – Diretório do projeto
- Primeiramente crie uma pasta no diretório que desejar. Essa pasta será o diretório do nosso projeto.



# ENVIANDO UM PROJETO PARA O GIT

- 2ª Etapa – Abrindo o Git Bash
- Acesse a pasta criada e clique com o botão direito em qualquer área em branco da pasta.
- Com o menu aberto, vá em:  
***Git Bash Here***

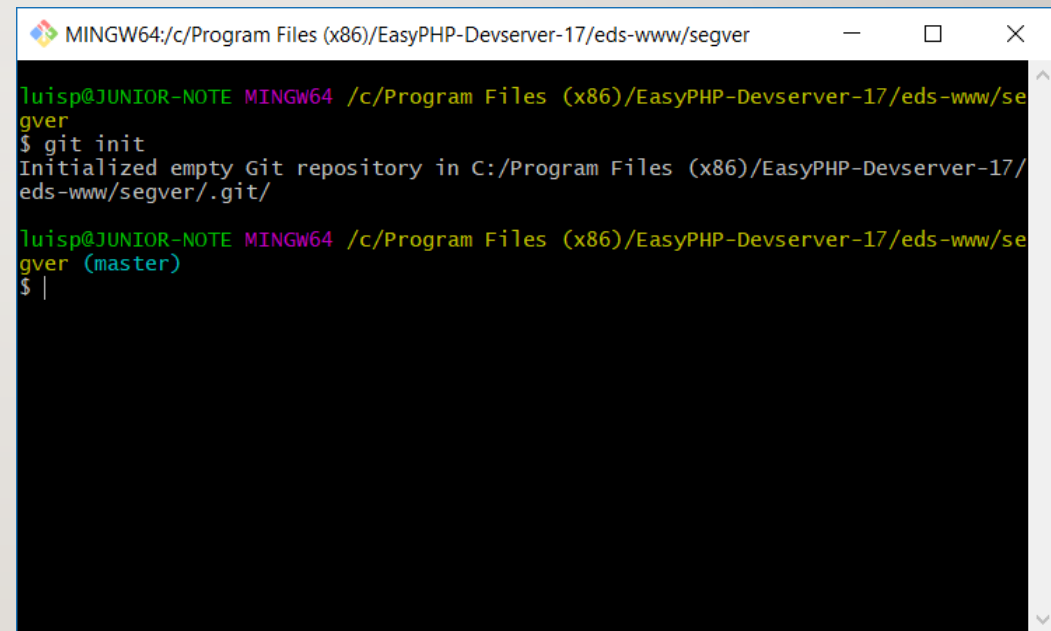




# ENVIANDO UM PROJETO PARA O GIT

---

- 3ª Etapa – Iniciando um repositório
- Com a console do git aberto, digite:  
`git init`
- Pronto, com isso foi criado o diretório `.git` e já podemos versionar todos os arquivos que colocarmos na nossa pasta.

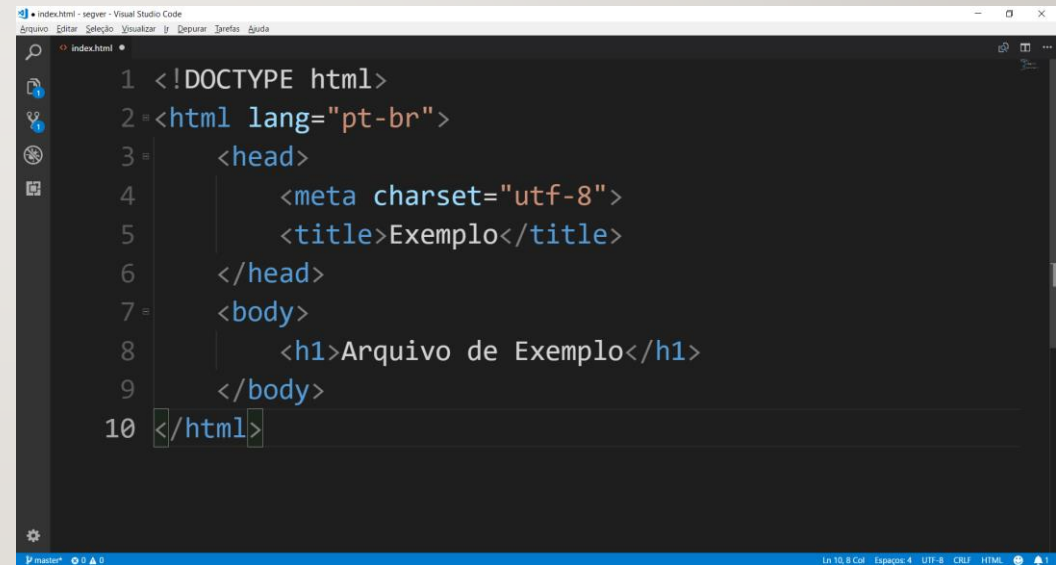


```
MINGW64:/c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
luisp@JUNIOR-NOTE MINGW64 /c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
$ git init
Initialized empty Git repository in C:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver/.git/
luisp@JUNIOR-NOTE MINGW64 /c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ |
```

# ENVIANDO UM PROJETO PARA O GIT

---

- 4ª Etapa – Crie um arquivo exemplo
- Usando o *Visual Studio Code* ou qualquer outro editor de texto, digite o código ao lado e salve como index.html na pasta que criou na etapa 1.

A screenshot of the Visual Studio Code editor interface. The title bar shows 'index.html - segue - Visual Studio Code'. The menu bar includes 'Arquivo', 'Editar', 'Seleção', 'Visualizar', 'Depurar', 'Janelas', and 'Ajuda'. The editor window displays an HTML file named 'index.html' with the following code:

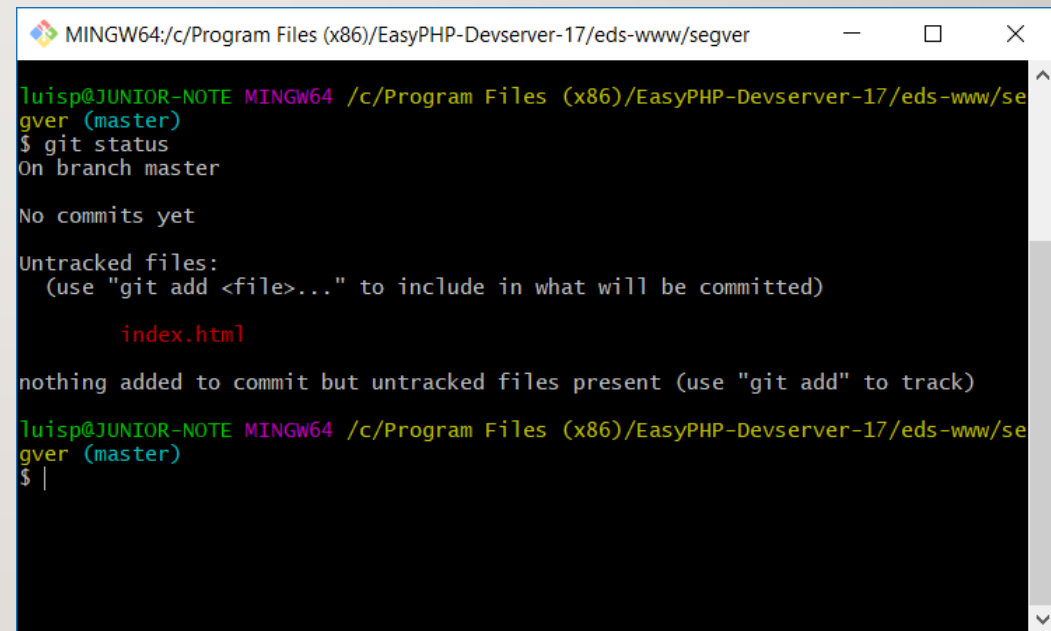
```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3   <head>
4     <meta charset="utf-8">
5     <title>Exemplo</title>
6   </head>
7   <body>
8     <h1>Arquivo de Exemplo</h1>
9   </body>
10 </html>
```

The status bar at the bottom shows 'master' on the left and 'Ln 10, 8 Col' 'Espaços: 4' 'UTF-8' 'CRLF' 'HTML' on the right.

# ENVIANDO UM PROJETO PARA O GIT

---

- 5ª Etapa – Verifique o status
- Execute o comando:  
`git status`
- Será exibido atual do seu branch com o novo arquivo em *untracked*.



```
MINGW64:/c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
luisp@JUNIOR-NOTE MINGW64 /c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/se
gver (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

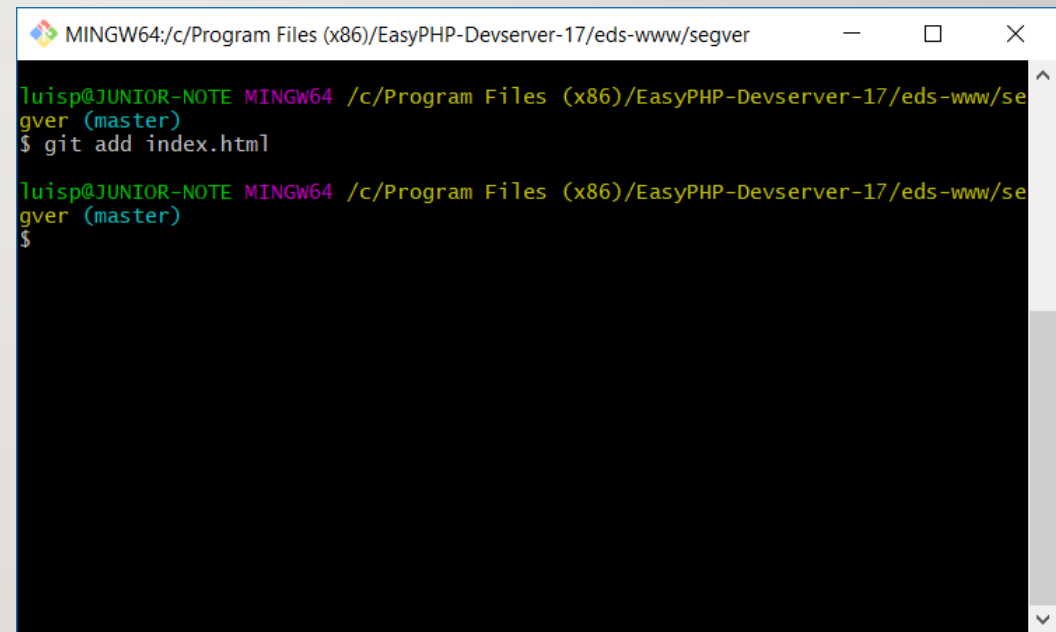
        index.html

nothing added to commit but untracked files present (use "git add" to track)
luisp@JUNIOR-NOTE MINGW64 /c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/se
gver (master)
$ |
```

# ENVIANDO UM PROJETO PARA O GIT

---

- 6ª Etapa – Adicione o arquivo ao index
- Execute o comando:  
`git add index.html`
- Será adicionado o arquivo a área de commit.

A screenshot of a Windows command prompt window titled 'MINGW64:/c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver'. The prompt shows the user 'luisp@JUNIOR-NOTE' in the 'MINGW64' environment. The command '\$ git add index.html' has been entered and executed. The output shows the prompt returning to '\$' on the next line, indicating the command was successful. The terminal has a black background with green and white text. A scrollbar is visible on the right side of the terminal window.

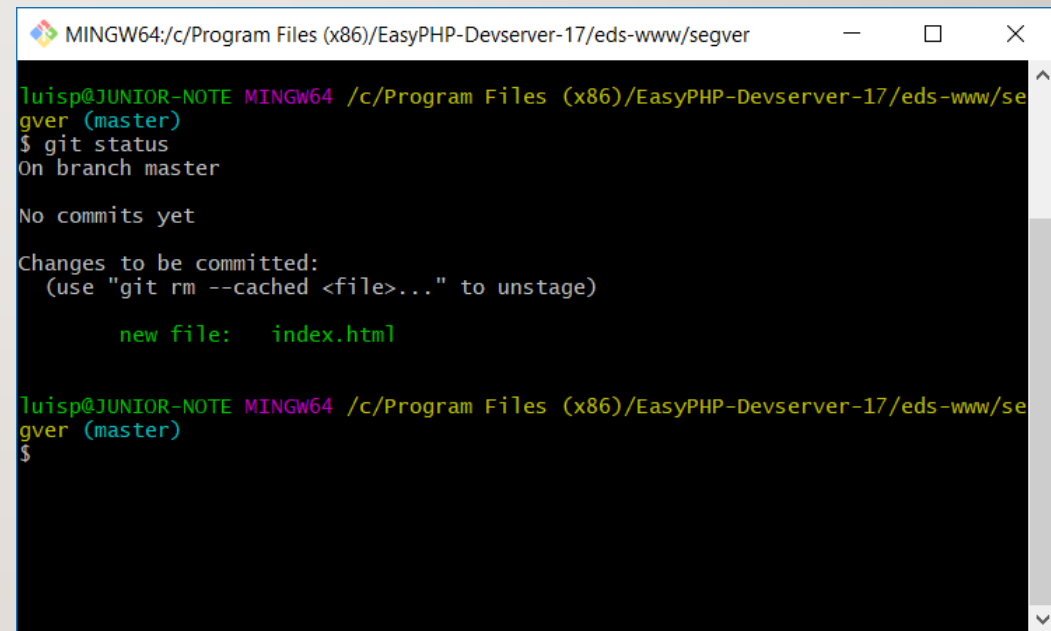
```
MINGW64:/c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/se
gver (master)
$ git add index.html

luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/se
gver (master)
$
```

# ENVIANDO UM PROJETO PARA O GIT

---

- 7ª Etapa – Verifique novamente o status
- Execute o comando:  
`git status`
- Será exibido atual do seu branch com o novo arquivo em *staged*.

A screenshot of a Windows Command Prompt window titled "MINGW64:/c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver". The prompt shows the user "luisp@JUNIOR-NOTE" in the "MINGW64" environment. The user enters the command "git status". The output shows the current branch is "master", there are no commits yet, and a new file "index.html" is staged for commit. The prompt then returns to the user's input line.

```
luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/se
gver (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   index.html

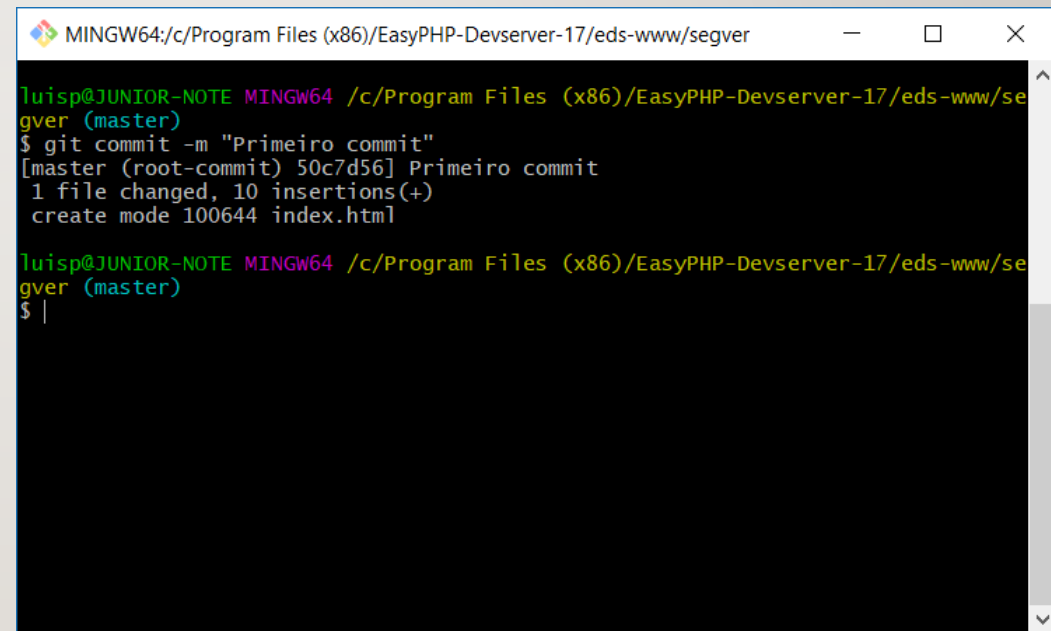
luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/se
gver (master)
$
```



# ENVIANDO UM PROJETO PARA O GIT

---

- 8ª Etapa – Envie o projeto
- Execute o comando:  
`git commit -m "Primeiro commit"`
- Será enviado o(s) arquivo(s) ao repositório git.

A screenshot of a Windows terminal window titled "MINGW64:/c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver". The terminal shows the execution of the git commit command. The prompt is "luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)". The command entered is "\$ git commit -m 'Primeiro commit'". The output shows the commit was successful: "[master (root-commit) 50c7d56] Primeiro commit", "1 file changed, 10 insertions(+)", and "create mode 100644 index.html". The prompt then returns to "\$ |".

```
MINGW64:/c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ git commit -m "Primeiro commit"
[master (root-commit) 50c7d56] Primeiro commit
1 file changed, 10 insertions(+)
create mode 100644 index.html

luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ |
```

# MICROSOFT GITHUB

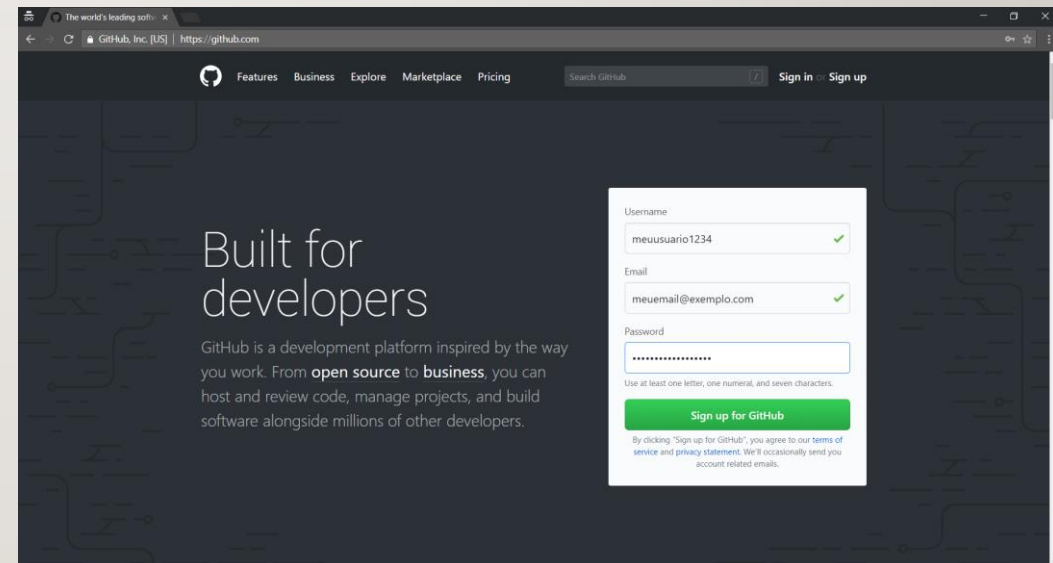
---

- É uma plataforma de hospedagem de código-fonte com controle de versão usando o **Git**. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em *projetos* privados e/ou **Open Source** de qualquer lugar do mundo. GitHub é amplamente utilizado por programadores para divulgação de seus trabalhos ou para que outros programadores contribuam com o projeto, além de promover fácil comunicação através de recursos que relatam problemas ou mesclam repositórios remotos (issues, pull request).



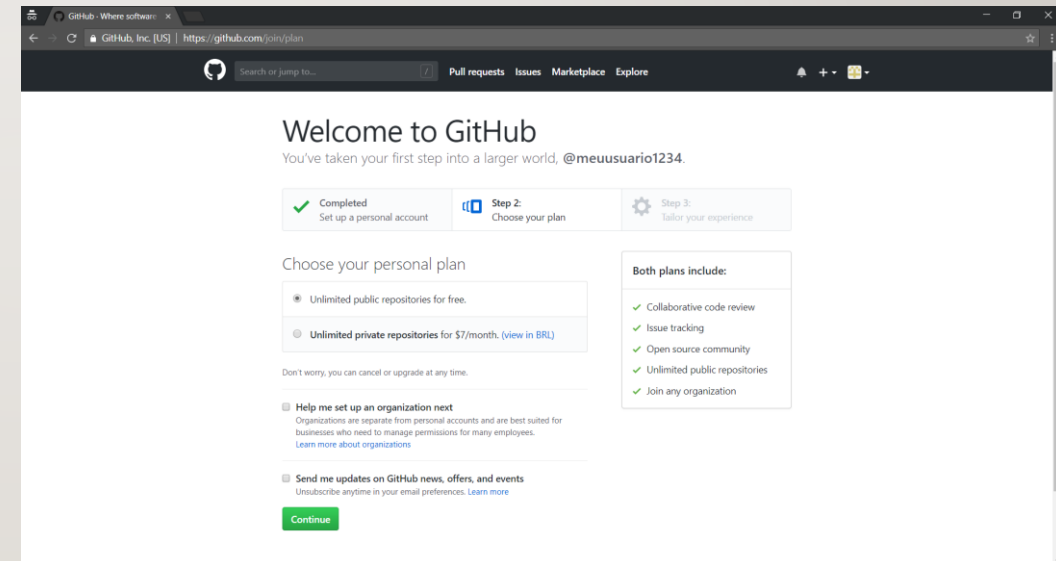
# CRIANDO UMA CONTA NO GITHUB

- 1ª Etapa – Acesse <https://github.com/>
- Preencha no formulário os dados:
  - Usuário
  - E-mail
  - Senha



# CRIANDO UMA CONTA NO GITHUB

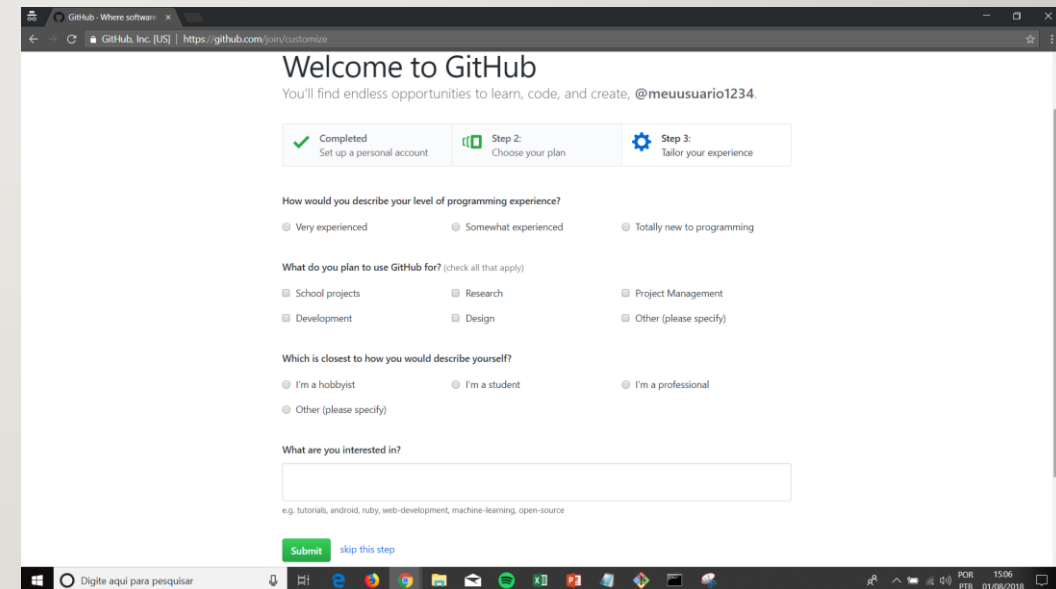
- 2ª Etapa – Selecione o tipo de plano para sua conta.
- Pode deixar selecionado *Unlimited public repositories for free*.
- Clique em *next*.





# CRIANDO UMA CONTA NO GITHUB

- 3ª Etapa – Fale um pouco sobre suas experiências.
- Em: *How would you describe your level of programming experience?*
  - *Diga seu nível de experiência em programação*
- Em: *What do you plan to use GitHub for?*
  - *Diga quais seus interesses de uso do GitHub*
- Em: *Which is closest to how you would describe yourself?*
  - *Diga como você se descreve*
- Clique em **Submit**.
- **Obs:** caso não queira preencher essas informações clique em ***skip this step*** ao lado do botão **Submit**



The screenshot shows the GitHub 'Welcome to GitHub' page for a new user, @meusuario1234. The page is titled 'Welcome to GitHub' and includes a progress bar with three steps: 'Step 1: Set up a personal account' (Completed), 'Step 2: Choose your plan' (in progress), and 'Step 3: Tailor your experience' (current step). The 'Step 3' section contains three questions with radio button options and checkboxes:

- How would you describe your level of programming experience?**
  - ☐ Very experienced
  - ☐ Somewhat experienced
  - ☐ Totally new to programming
- What do you plan to use GitHub for? (check all that apply)**
  - ☐ School projects
  - ☐ Research
  - ☐ Project Management
  - ☐ Development
  - ☐ Design
  - ☐ Other (please specify)
- Which is closest to how you would describe yourself?**
  - ☐ I'm a hobbyist
  - ☐ I'm a student
  - ☐ I'm a professional
  - ☐ Other (please specify)

Below these questions is a text input field for 'What are you interested in?' with a placeholder text: 'e.g. tutorials, android, ruby, web-development, machine-learning, open-source'. At the bottom of the form are two buttons: 'Submit' and 'skip this step'.



# CRIANDO UMA CONTA NO GITHUB

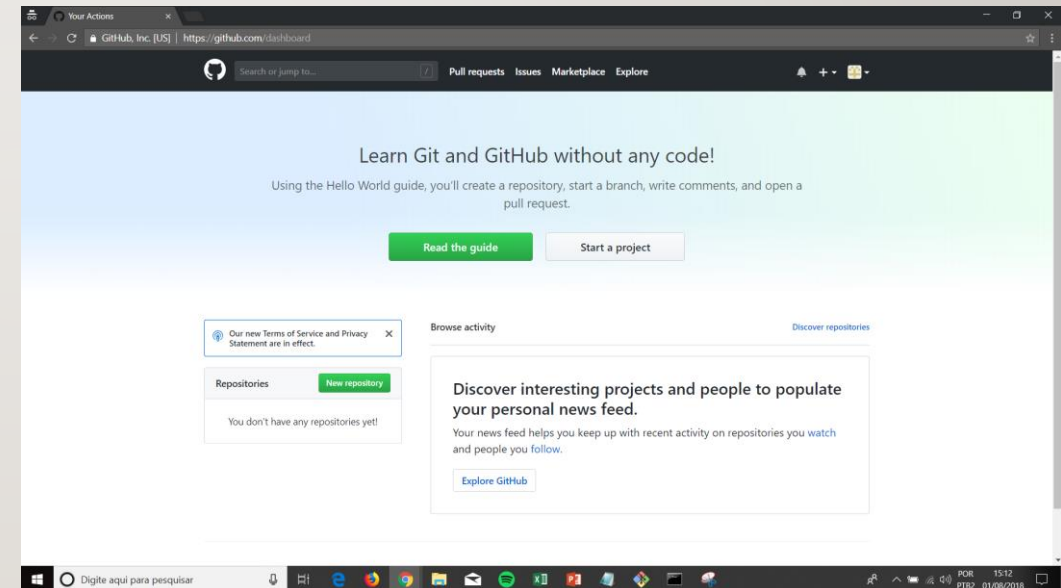
---

- 4ª Etapa – Confirme seu e-mail.
- O Github enviará um e-mail para confirmação da conta criada, acesse seu e-mail e clique no link em anexo.



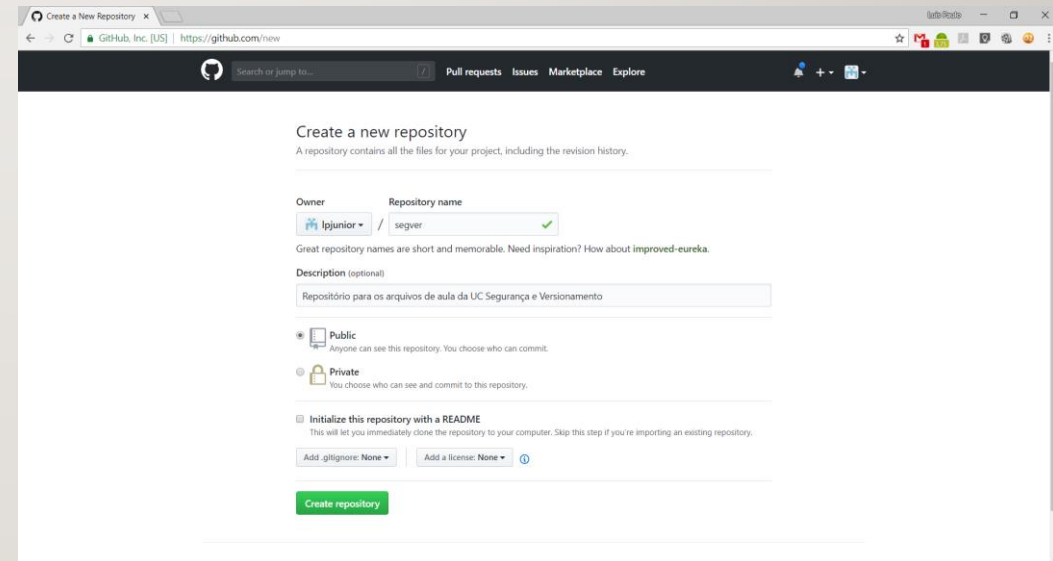
# CRIANDO UM REPOSITÓRIO NO GITHUB

- 1ª Etapa – Após efetuar login no github, na sua página principal procure por ***Start a Project*** ou ***New repository***



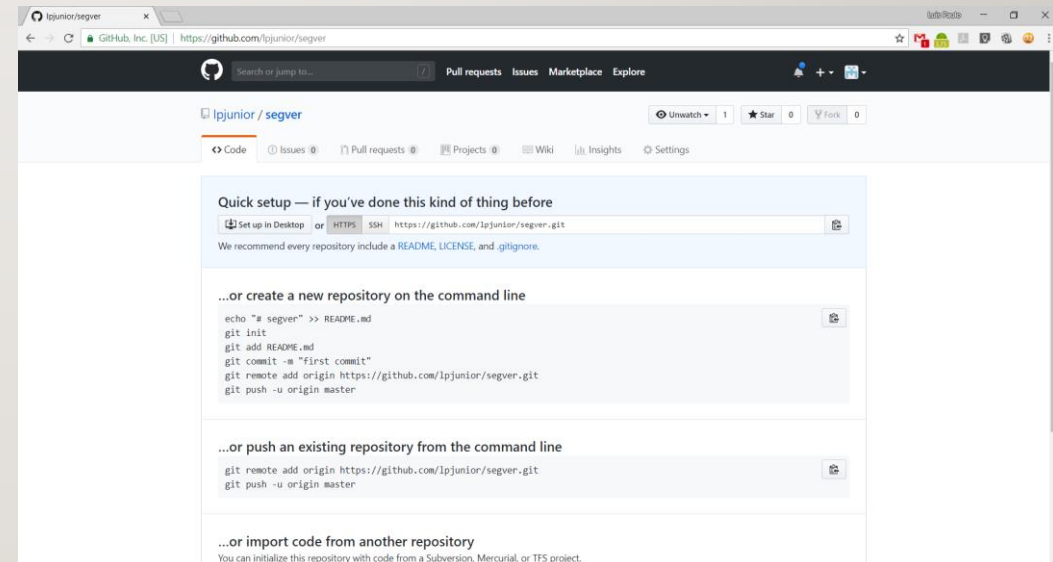
# CRIANDO UM REPOSITÓRIO NO GITHUB

- 2ª Etapa – Preencher os dados do repositório.
- Em: *Repository name*
  - *Defina um nome para seu repositório*
- Em: *Description (opcional)*
  - *Escreva uma breve informação sobre o projeto*
- Clique no **Create repository**



# CRIANDO UM REPOSITÓRIO NO GITHUB

- 3ª Etapa – Repositório criado.
- Com o repositório criado, podemos trabalhar com o git remoto, usando a url em **HTTPS** ou **SSH**

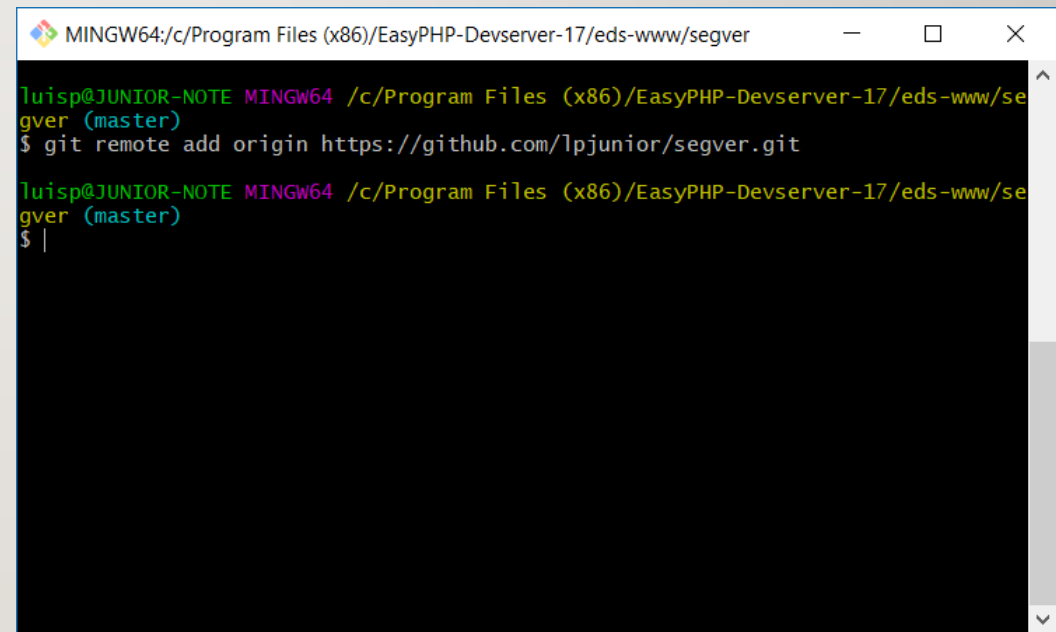




# ENVIANDO UM PROJETO PARA O GITHUB

---

- 1ª Etapa – Adicionando o endereço remoto
- Com nosso repositório criado, copia a URL HTTPS dele e digite o seguinte comando:  
`git remote add origin <URL>`
- Será adicionado ao escopo do projeto o URL do github.



```
MINGW64:/c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ git remote add origin https://github.com/lpjunior/segver.git

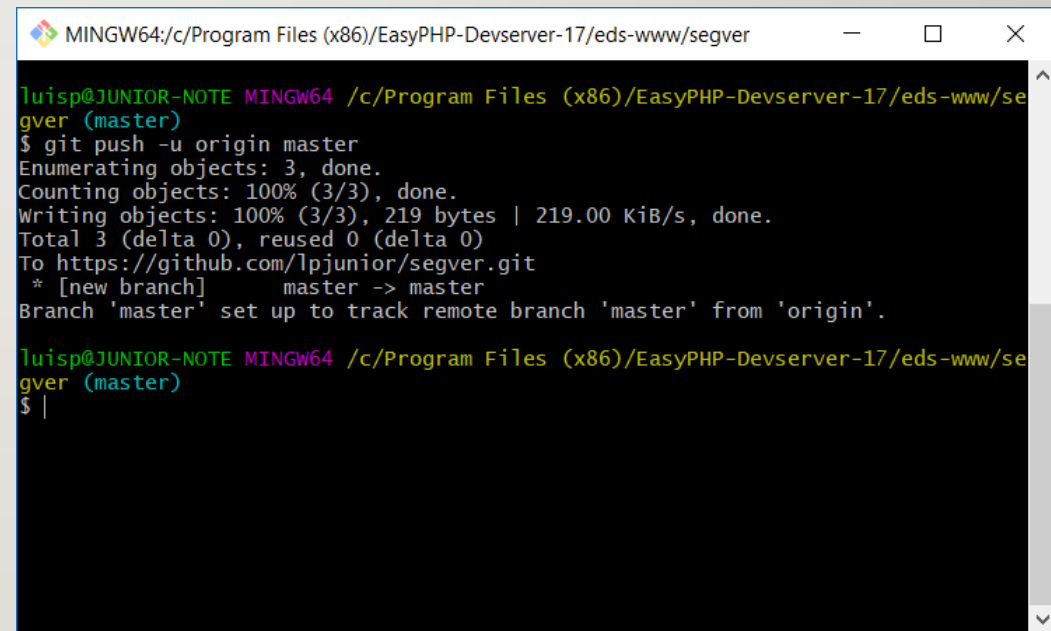
luisp@JUNIOR-NOTE MINGW64 /c/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ |
```



# ENVIANDO UM PROJETO PARA O GITHUB

---

- 2ª Etapa – Enviando para o Github
- Execute o comando :  
`git push -u origin master`
- Será enviado o projeto para o Github.

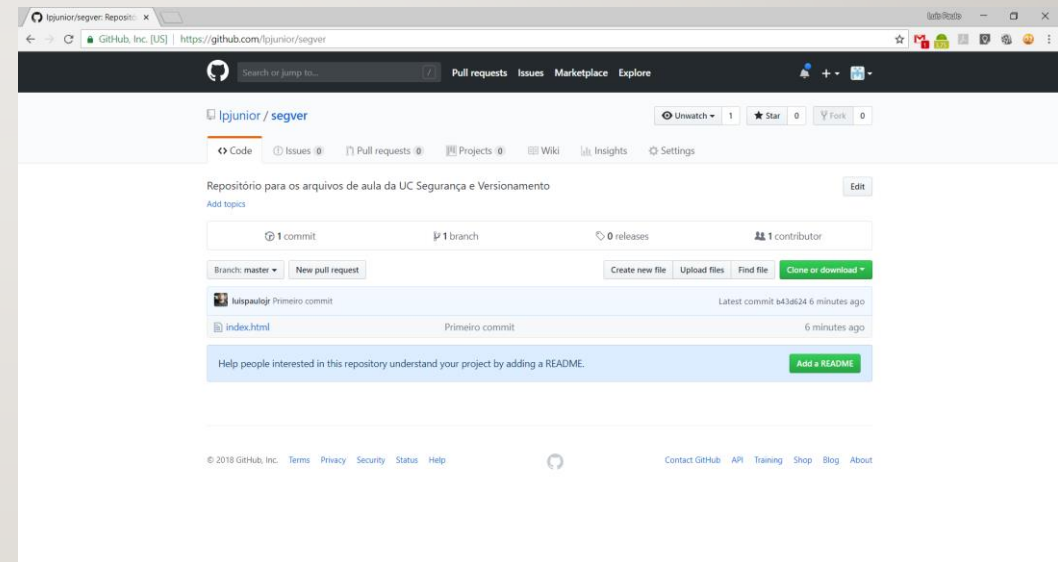


```
MINGW64:/c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver
luisp@JUNIOR-NOTE MINGW64 /c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 219 bytes | 219.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/lpjuniior/segver.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

luisp@JUNIOR-NOTE MINGW64 /c:/Program Files (x86)/EasyPHP-Devserver-17/eds-www/segver (master)
$ |
```

# ENVIANDO UM PROJETO PARA O GITHUB

- 3ª Etapa – Visualizando o projeto
- Volte a página do Github e acesse o repositório, agora a tela dele se apresenta diferente, com o(s) arquivo(s) que enviamos.



# ATLASSIAN BITBUCKET

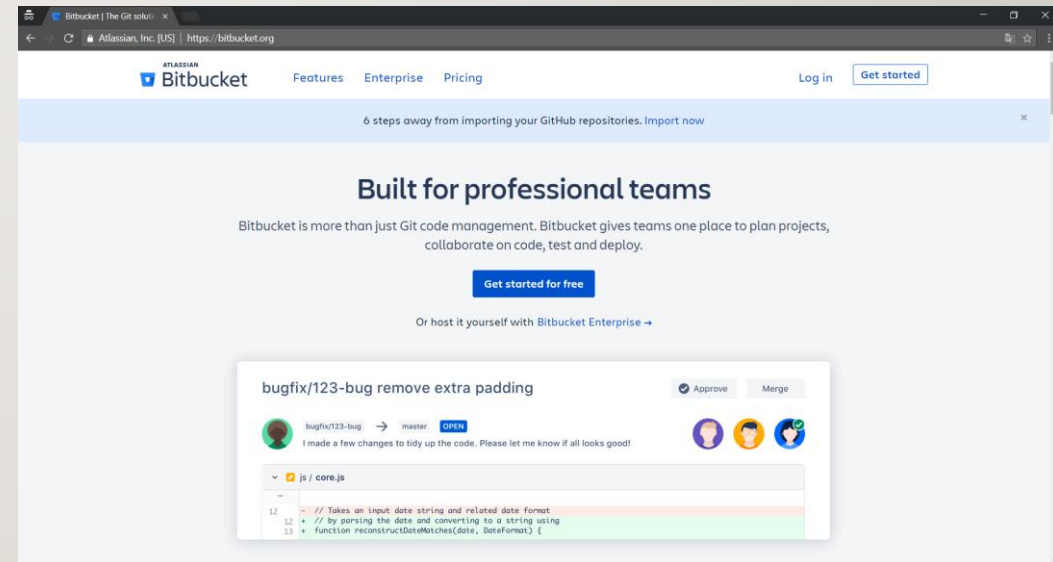
---

- É um serviço de hospedagem de projetos controlados através do Mercurial, um sistema de controle de versões distribuído. É similar ao GitHub (que utiliza Git, somente). Bitbucket têm um serviço grátis e um comercial. O serviço é escrito em Python.
- O Bitbucket também suporta repositórios usando o sistema de controle de versões Git.



# CRIANDO UMA CONTA NO GITHUB

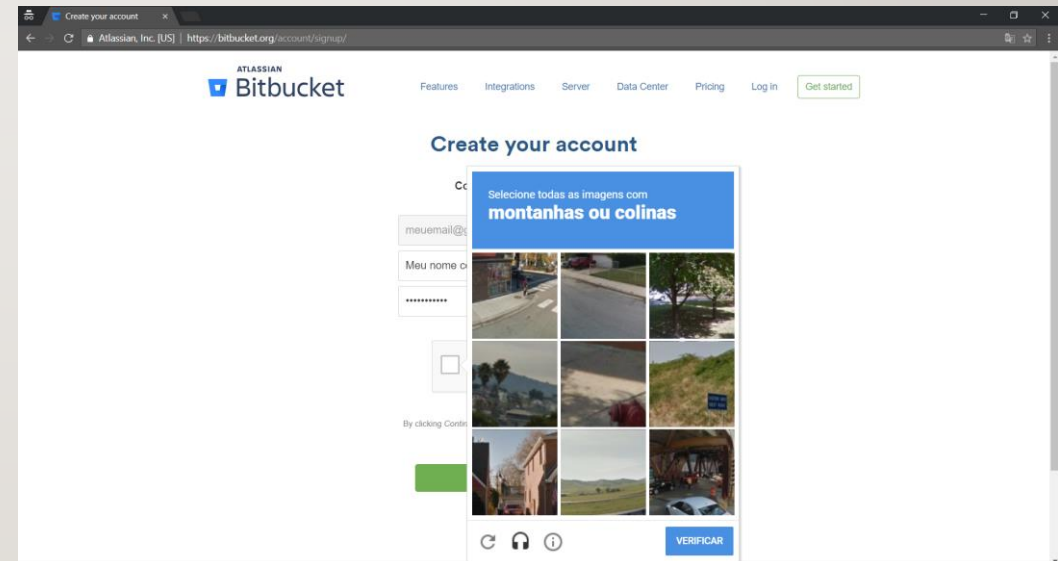
- 1ª Etapa – Acesse <https://bitbucket.org/>
- Clique em *Get started for free*





# CRIANDO UMA CONTA NO GITHUB

- 2ª Etapa – Acesse <https://bitbucket.org/>
- Preencha no formulário os dados:
  - E-mail
  - Nome Completo
  - Senha
- Confirme o recaptcha
- Feito, clique em **continue**





# CRIANDO UMA CONTA NO GITHUB

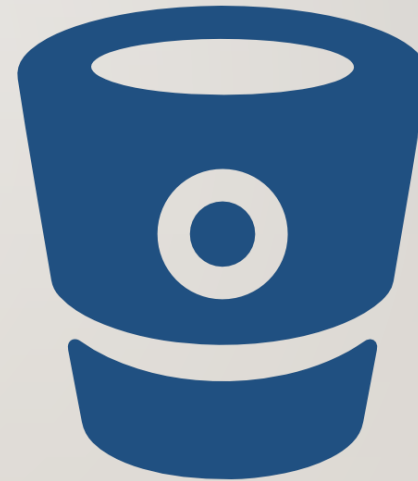
---

- 4ª Etapa – Confirme seu e-mail.
- O Bitbucket também enviará um e-mail para confirmação da conta criada, acesse seu e-mail e clique no link em anexo.



# DIFERENÇA ENTRE O GITHUB E O BITBUCK

---



# PRINCIPAIS DIFERENÇAS

---

## GITHUB

- É uma rede social de projetos, aonde você pode visualizar a quantidade de arquivos por linguagem seu projeto tem.
  - **Ex.:**
    - 10% de JavaScript
    - 45% de PHP
    - 25% de HTML
    - 15% de CSS

## BITBUCKET

- Não tem esses recursos

# PRINCIPAIS DIFERENÇAS

---

## GITHUB

- Tem como recursos um calendário de quantas alterações que fez no projeto, um histórico.

## BITBUCKET

- Não tem esse recurso



# PRINCIPAIS DIFERENÇAS

---

## GITHUB

- Você pode colaborar em outros projetos, utilizando o recurso pull request.

## BITBUCKET

- Não tem esse recurso



# PRINCIPAIS DIFERENÇAS

---

## GITHUB

- No github criamos projetos normalmente de código aberto (OpenSource), logo um projeto publico aonde as você pode compartilhar com outras pessoas. Existe a possibilidade de criar projetos privados, porém este recurso é pago

## BITBUCKET

- Podemos criar projetos privados sem custo com limite de até 5 colaboradores.

# BIBLIOGRAFIA



- 
- <https://git-scm.com/doc>
  - <https://git-scm.com/book/pt-br/v2>
  - <https://git-scm.com/book/pt-br/v1/Primeiros-passos-Sobre-Controle-de-Vers%C3%A3o>
  - <https://pt.wikipedia.org/wiki/Git>
  - <https://pt.wikipedia.org/wiki/GitHub>
  - <https://pt.wikipedia.org/wiki/Bitbucket>
  - <http://jeiks.net/wp-content/uploads/2016/11/GIT.pdf>