

**FIAP GRADUAÇÃO**

TDS

# Responsive Web Development

PROF. ALEXANDRE CARLOS  
[profalexandre.jesus@fiap.com.br](mailto:profalexandre.jesus@fiap.com.br)

# GIT E GITHUB

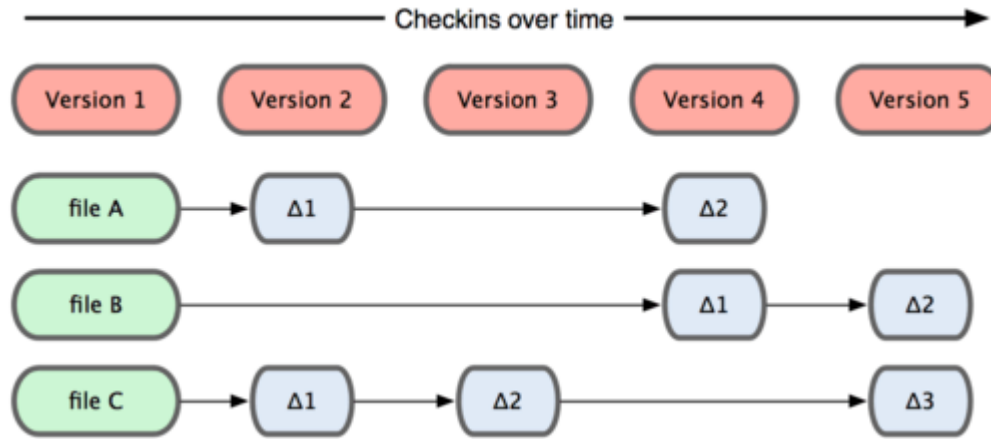


- Controle de versão
- Comandos git
- Repositórios remotos
- Github

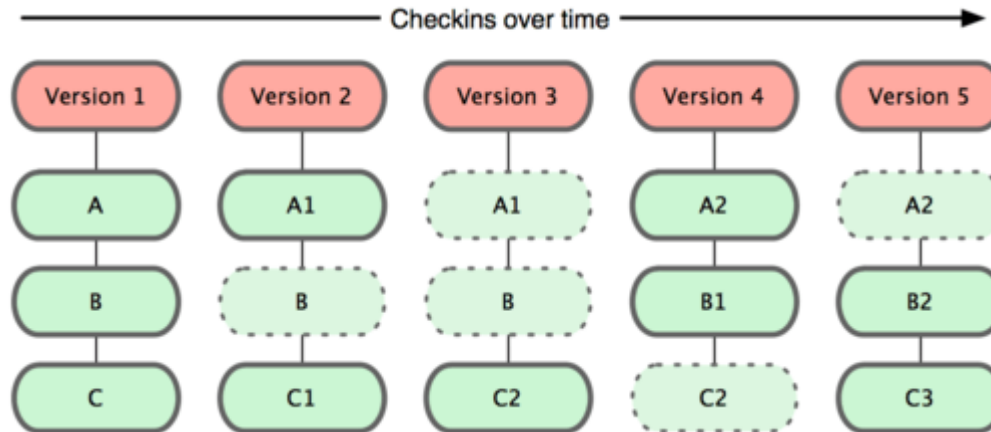
- Cópias e mais cópias de um mesmo projeto!
- Controle de versão:
  - Gerenciar diferentes versões de um documento;
  - Exemplos: CVS, **Git**, SVN, Mercurial, TFS, ClearCase.



- Outros Sistemas:



- Sistema Git:



# O QUE É O GITHUB?

- Serviço de Web compartilhado para projetos que utilizam o Git para versionamento.

- <https://github.com/>



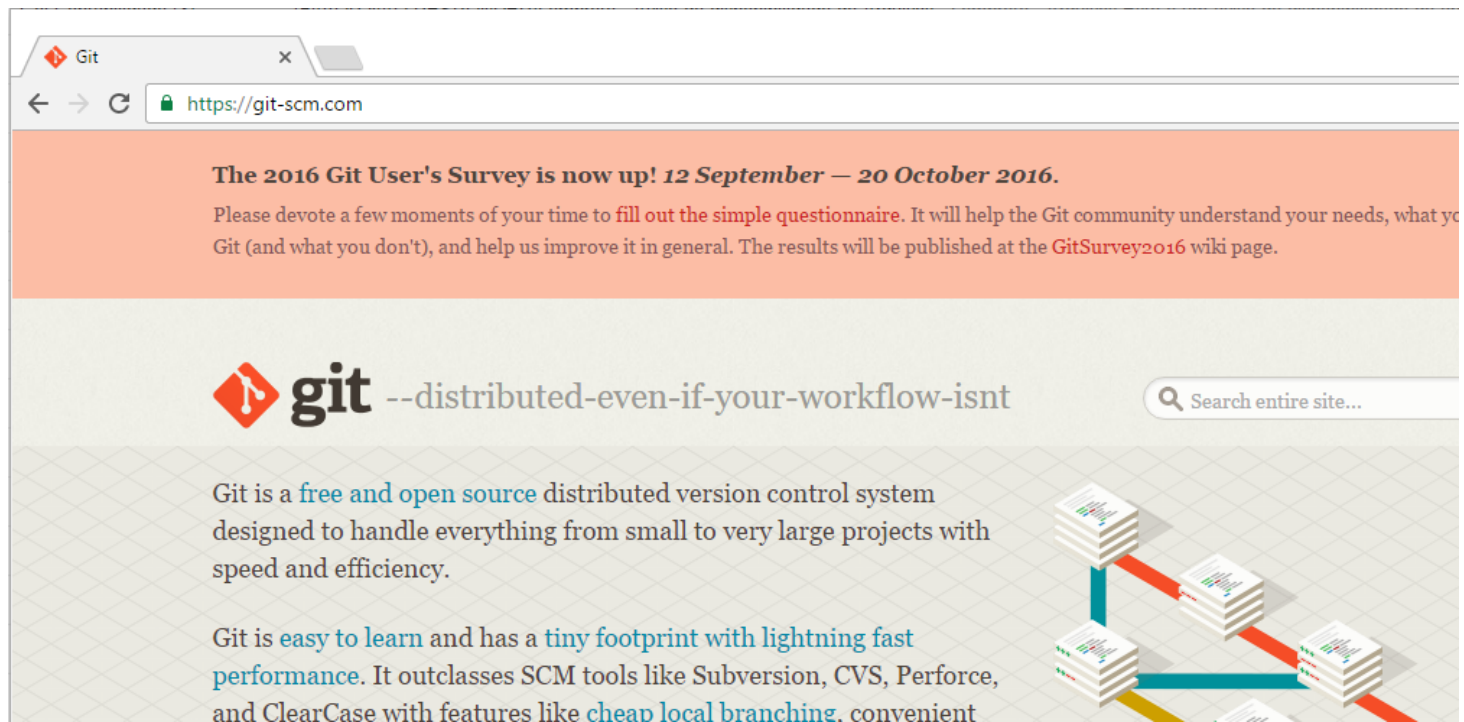
- Outra opção, com repositório privado:

- <https://bitbucket.org/>

# INSTALANDO O GIT

- Acesse o site e siga as instruções!

- <https://git-scm.com/>





- O **git config** permite ler e configurar variáveis de configuração do git. Essas variáveis podem ser armazenadas em 3 lugares diferentes.
- Primeiro precisamos definir o seu nome de usuário e endereço de e-mail.

MINGW64:/c/Users/Thiago/Desktop

```
Thiago@THIAGO MINGW64 ~/Desktop  
$ git config --global user.name "Thiago Yama"
```

Nome de usuário

```
Thiago@THIAGO MINGW64 ~/Desktop  
$ git config --global user.email thiagoyama@gmail.com
```

E-mail

```
Thiago@THIAGO MINGW64 ~/Desktop
```

# VERIFICANDO AS CONFIGURAÇÕES

MINGW64:/c/Users/Thiago/Desktop

```
Thiago@THIAGO MINGW64 ~/Desktop  
$ git config user.name  
Thiago Yama
```

Recuperando o nome de usuário

```
Thiago@THIAGO MINGW64 ~/Desktop  
$ git config user.email  
thiagoyama@gmail.com
```

Recuperando o endereço de e-mail

```
Thiago@THIAGO MINGW64 ~/Desktop  
$ git config --list  
core.symlinks=false  
core.autocrlf=true  
core.fscache=true  
color.diff=auto  
color.status=auto  
color.branch=auto
```

Recuperando todas as configurações

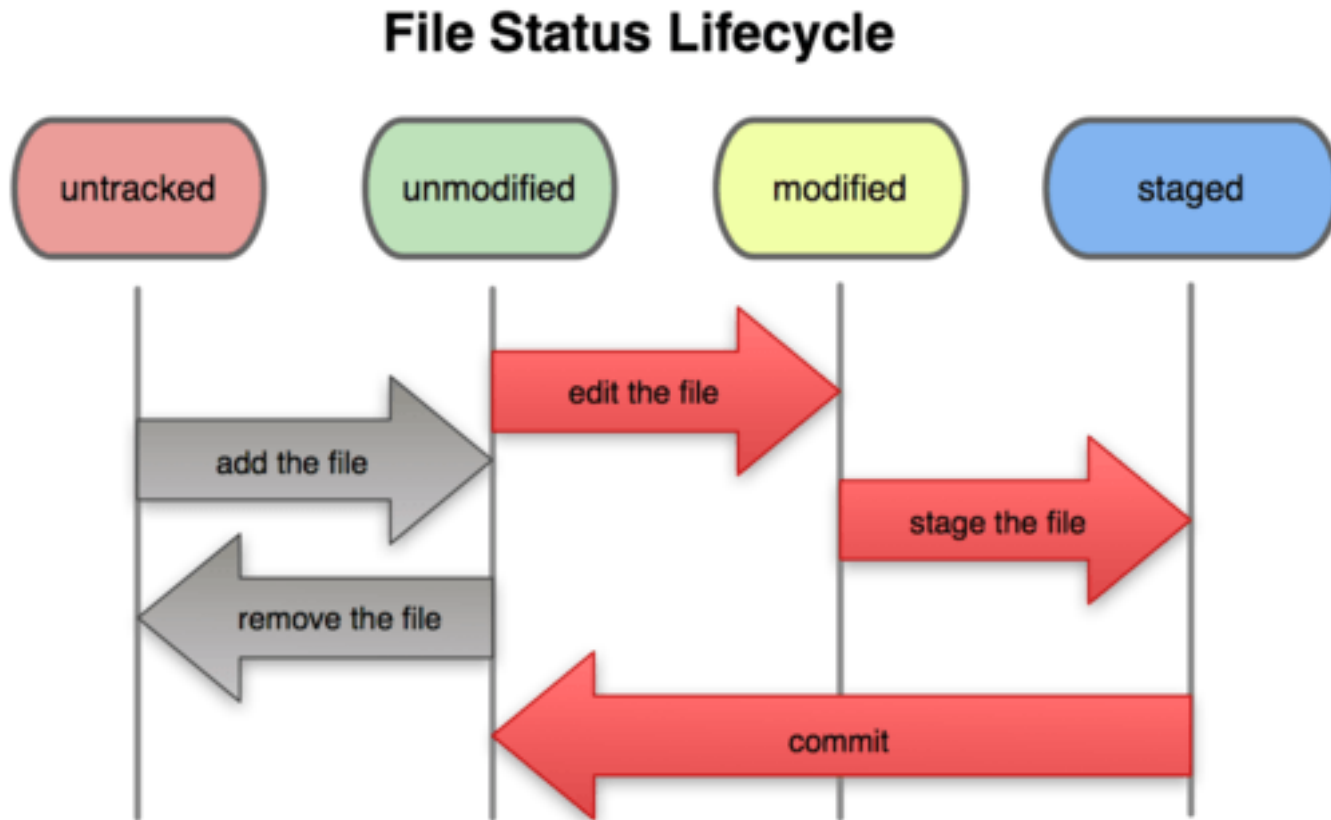
- Existem duas formas principais de obter um repositório Git:
  - Criar um novo repositório a partir de um diretório;
  - Clonar o repositório Git existente de um servidor;
- OBS: Em ambos os casos, o branch principal chama-se **master**. Não se preocupe, veremos sobre branch mais adiante.

- Primeiro navegue até o diretório e utilize o comando **git init**
  - Um diretório `.git` será adicionado com os arquivos necessários para controlar o repositório.

```
Thiago@THIAGO MINGW64 ~/Desktop
$ cd projetos/

Thiago@THIAGO MINGW64 ~/Desktop/projetos
$ git init
Initialized empty Git repository in C:/Users/Thiago/Desktop/projetos/.git/

Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$
```



- **git status** – comando para verificar os estados dos arquivos;
- **git add <arquivos>** - comando para monitorar um arquivo;

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git status
On branch master

Initial commit

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

    arquivo.txt

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

- Adicionando um arquivo e verificando o seu novo status:

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git add arquivo.txt

Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git status
On branch master

Initial commit

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

    new file:   arquivo.txt
```

- Para adicionar todos os arquivos: **git add .**

- Modificando o arquivo e verificando o status:

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git status
On branch master

Initial commit

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

        new file:   arquivo.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   arquivo.txt
```



- **git commit** – comando para gerar uma nova versão, todos os arquivos no **status stage** (selecionado) farão parte desta versão.

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git commit -m "Adicionando o arquivo.txt"
[master (root-commit) 8389495] Adicionando o arquivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 arquivo.txt
```

- **git log** – visualiza o histórico de commits.
- OBS: Para sair da listagem os logs, basta apertar a letra “q”


```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git log
commit 83894955937a368500a0e7abf59ed7052f81ae52
Author: Thiago Yama <thiagoyama@gmail.com>
Date:   Fri Oct 7 18:00:59 2016 -0300

    Adicionando o arquivo.txt
```

# DESFAZENDO AS COISAS

- **git diff** – Exibe as linhas modificadas dos arquivos alterados no projeto (Para sair, pressione a letra “q”);

```
Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git diff
diff --git a/index.html b/index.html
index b998b1f..d671771 100644
--- a/index.html
+++ b/index.html
@@ -2,7 +2,7 @@
<html>
  <head>
    <meta charset="utf-8">
-   <title>Site V 2.0</title>
+   <title>Site V 3.0</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
@@ -16,4 +16,4 @@
    <script src="js/script.js"></script>
  </body>
- </html>
\ No newline at end of file
+ </html>
```



- **git checkout <commit>** - volta para a versão de determinado commit. (Deve se usar o hash do commit);

```
Correção do numero da versão na tag title
commit 95a2090f92f4da900e8b6783a402c59158a15f95
Author: Douglas Cabral <dougscabral@gmail.com>
Date: Sat Oct 22 12:15:00 2016 -0200

Arquivos modificados para a v2

commit ce4235d838b78803472417e8580fb2b4c43752f7
Author: Douglas Cabral <dougscabral@gmail.com>
Date: Sat Oct 22 11:51:27 2016 -0200

First commit

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git checkout ce4235d838b78803472417e8580fb2b4c43752f7
Note: checking out 'ce4235d838b78803472417e8580fb2b4c43752f7'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at ce4235d... First commit
```

- **git checkout <arquivo>** - desfaz as alterações do arquivo;

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   arquivo.txt

no changes added to commit (use "git add" and/or "git commit -a")

Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git checkout arquivo.txt

Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- **git checkout <commit> <arquivo>** - volta o arquivo para determinada versão de um commit;  
(OBS: Pode ser usado para voltar arquivos excluídos);

```
Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git checkout ce4235d838b78803472417e8580fb2b4c43752f7 index.html

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$
```

- **git clean** – Remove arquivos ainda não monitorados (arquivos com o status untracked);

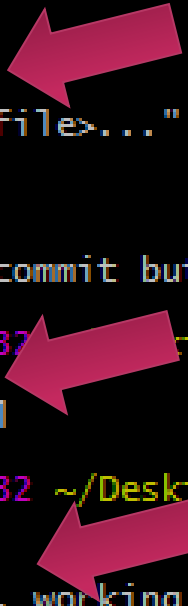
```
Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    novo.html

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

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git clean -fd
Removing novo.html

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git status
On branch master
nothing to commit, working tree clean
```





- **git reset HEAD <arquivo>** - trás o arquivo do status selecionado (stage) para modified;

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
    modified:   arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git reset HEAD arquivo.txt
```

```
Unstaged changes after reset:
```

```
M       arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git checkout -- <file>..." to discard changes in working directory)
```

```
    modified:   arquivo.txt
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

- **git reset** – desfaz um commit;
  - soft:** volta para um commit específico e deixa as alterações no status selecionado;
  - mixed:** volta para um commit específico e deixa as alterações no status modified;
  - hard:** volta para um commit específico e mata todas as alterações;

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git log
```

```
commit 21838b5928852283b8951fa1804e5c06e75d21dd
```

```
Author: Thiago Yama <thiagoyama@gmail.com>
```

```
Date:    Fri Oct 7 18:22:38 2016 -0300
```

```
    Modificação no arquivo.txt
```

```
commit 83894955937a368500a0e7abf59ed7052f81ae52
```

```
Author: Thiago Yama <thiagoyama@gmail.com>
```

```
Date:    Fri Oct 7 18:00:59 2016 -0300
```

```
    Adicionando o arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git reset --soft 83894955937a368500a0e7abf59ed7052f81ae52
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
    (use "git reset HEAD <file>..." to unstage)
```

```
    modified:   arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git log
```

```
commit bd03e3db40049a79df52f36f2aed511a40f086a3
```

```
Author: Thiago Yama <thiagoyama@gmail.com>
```

```
Date:   Fri Oct 7 18:28:08 2016 -0300
```

```
    Modificação no arquivo.txt
```

```
commit 83894955937a368500a0e7abf59ed7052f81ae52
```

```
Author: Thiago Yama <thiagoyama@gmail.com>
```

```
Date:   Fri Oct 7 18:00:59 2016 -0300
```

```
    Adicionando o arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git reset --mixed 83894955937a368500a0e7abf59ed7052f81ae52
```

```
Unstaged changes after reset:
```

```
M       arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git status
```

```
On branch master
```

```
Changes not staged for commit:
```

```
  (use "git add <file>..." to update what will be committed)
```

```
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
    modified:   arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git log
```

```
commit a1d14990bab88de6d706e4942e9b29804711ac4e
```

```
Author: Thiago Yama <thiagoyama@gmail.com>
```

```
Date:   Fri Oct 7 18:29:34 2016 -0300
```

```
    Modificação no arquivo.txt
```

```
commit 83894955937a368500a0e7abf59ed7052f81ae52
```

```
Author: Thiago Yama <thiagoyama@gmail.com>
```

```
Date:   Fri Oct 7 18:00:59 2016 -0300
```

```
    Adicionando o arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git reset --hard 83894955937a368500a0e7abf59ed7052f81ae52
```

```
HEAD is now at 8389495 Adicionando o arquivo.txt
```

```
Thiago@THIAGO MINGW64 ~/Desktop/projetos (master)
```

```
$ git status
```

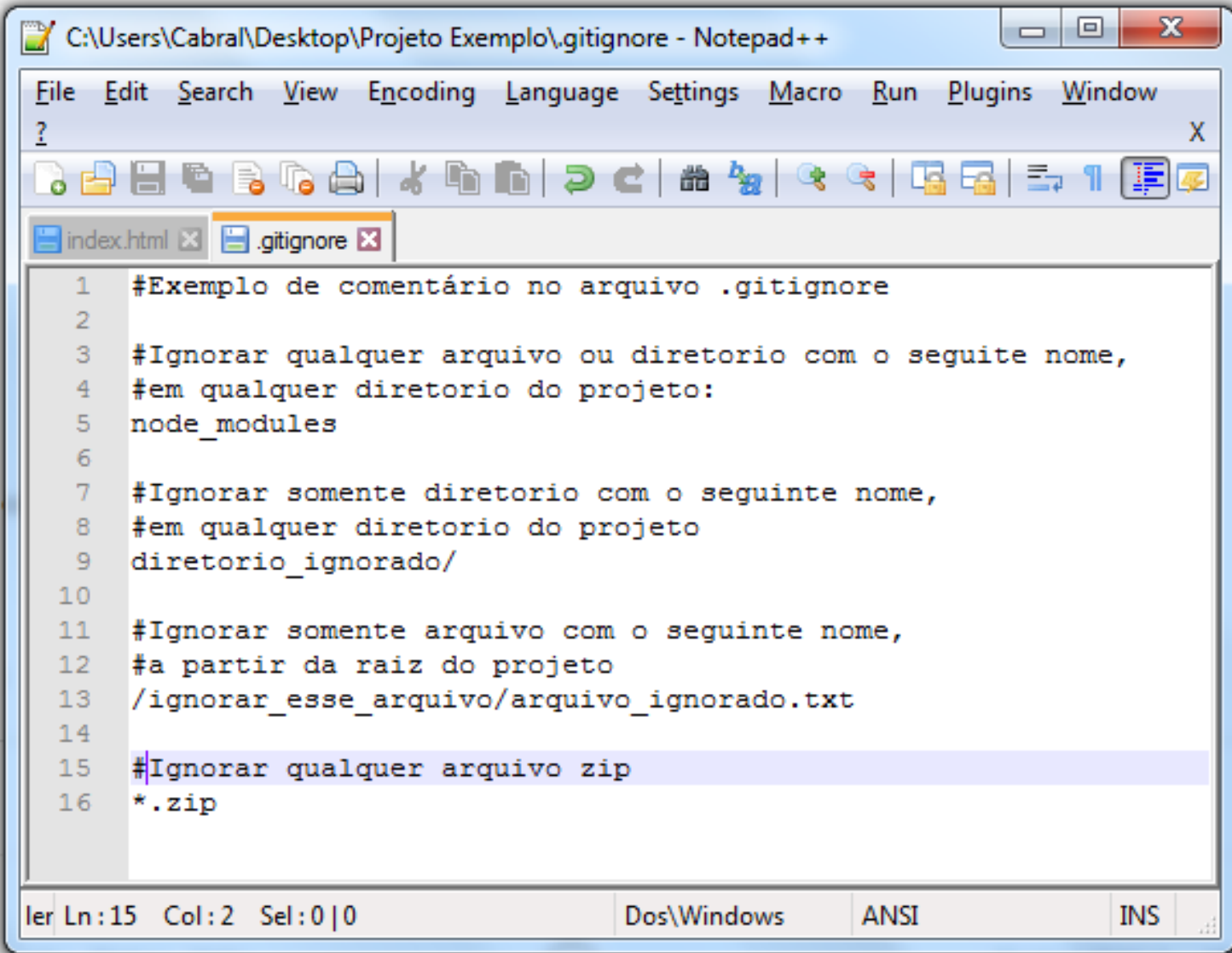
```
On branch master
```

```
nothing to commit, working tree clean
```

IGNORANDO ARQUIVOS

- **Para ignorar um arquivo, basta criar um arquivo com o nome “.gitignore” na raiz do projeto.**
  - Dentro deste arquivo deve conter o caminho dos arquivos que deverão ser ignorados no projeto.
  - Arquivos ignorados não serão mais monitorados pelo git.
  - O **ideal** é ignorar o arquivo quando ele **ainda está** no status **untracked**. Após isso é necessário remover o arquivo do cache do git usando o comando: **git rm –cached <arquivo>**

# IGNORANDO ARQUIVOS



```
C:\Users\Cabral\Desktop\Projeto Exemplo\.gitignore - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window
?
index.html .gitignore
1 #Exemplo de comentário no arquivo .gitignore
2
3 #Ignorar qualquer arquivo ou diretorio com o seguinte nome,
4 #em qualquer diretorio do projeto:
5 node_modules
6
7 #Ignorar somente diretorio com o seguinte nome,
8 #em qualquer diretorio do projeto
9 diretorio_ignorado/
10
11 #Ignorar somente arquivo com o seguinte nome,
12 #a partir da raiz do projeto
13 /ignorar_esse_arquivo/arquivo_ignorado.txt
14
15 #Ignorar qualquer arquivo zip
16 *.zip
ler Ln:15 Col:2 Sel:0 | 0 Dos\Windows ANSI INS
```



- **Para exemplos e templates de arquivos .gitignore consultar:**
- <https://www.gitignore.io/>
- <https://github.com/github/gitignore>

CRIANDO TAG

- **git tag** - são apelidos para commits. Uma das vantagens é o checkout facilitado para um commit específico através de um nome mais fácil de decorar ao invés de seu hash.
  - Com o HEAD em um commit específico, basta apenas digitar o comando com um nome como o parâmetro. (Ex: git tag <nomedatag>)
  - Usando o comando sem parâmetros, todas as tags criadas são listadas.

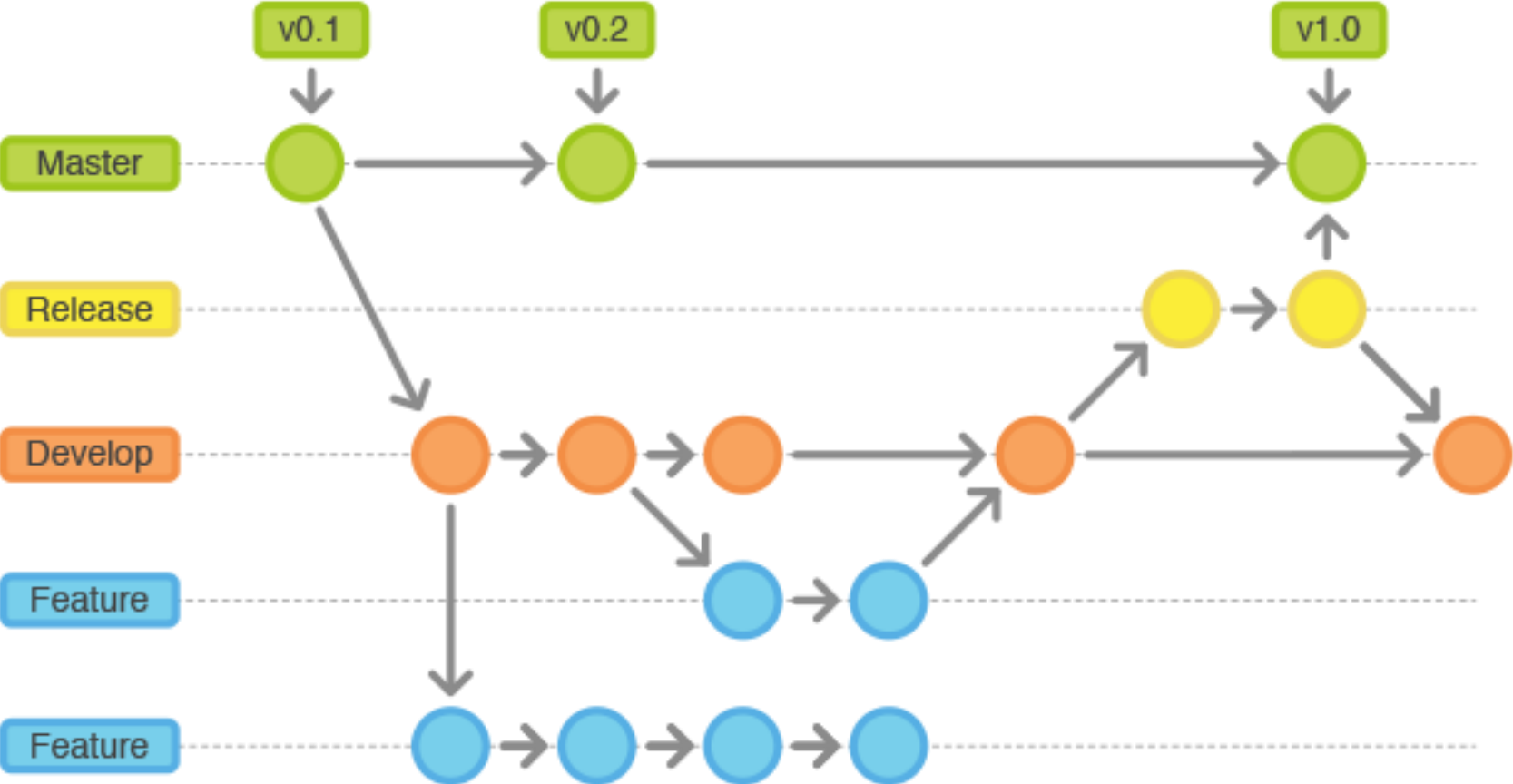
```
Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git commit -am "Inserido gitignore"
[master caed11b] Inserido gitignore
1 file changed, 2 insertions(+), 2 deletions(-)

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git tag exemplo

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ git tag
exemplo
v1
v2
```

# TRABALHANDO COM BRANCH

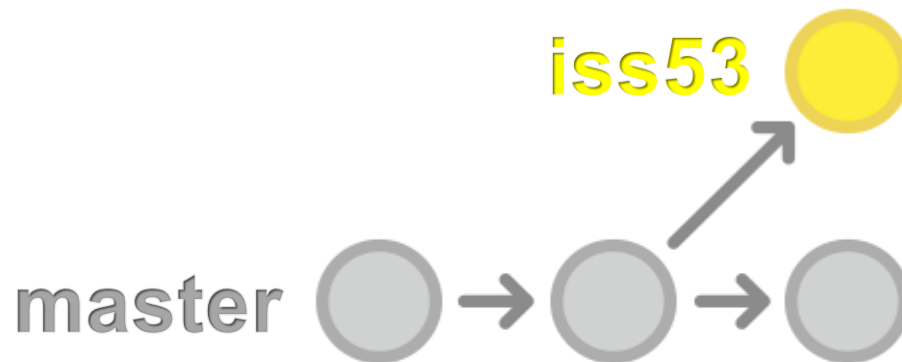
- O que é **branch** e por que utiliza-la?
  - Seções/Bifurcações no repositório;
  - Funcionam como linhas do tempo separadas.
  - Permite separar funcionalidades ou correções que ainda estão sendo programadas do código principal;



- **git branch <nome>**
  - Cria uma nova branch a partir do HEAD;
- **git checkout <nome>**
  - Muda o ponteiro para a branch;
- **git checkout -b <nome>**
  - Cria a branch e muda para ele;

```
Thiago@THIAGO MTNGW64 ~/Desktop/projetos (master)
$ git branch iss53

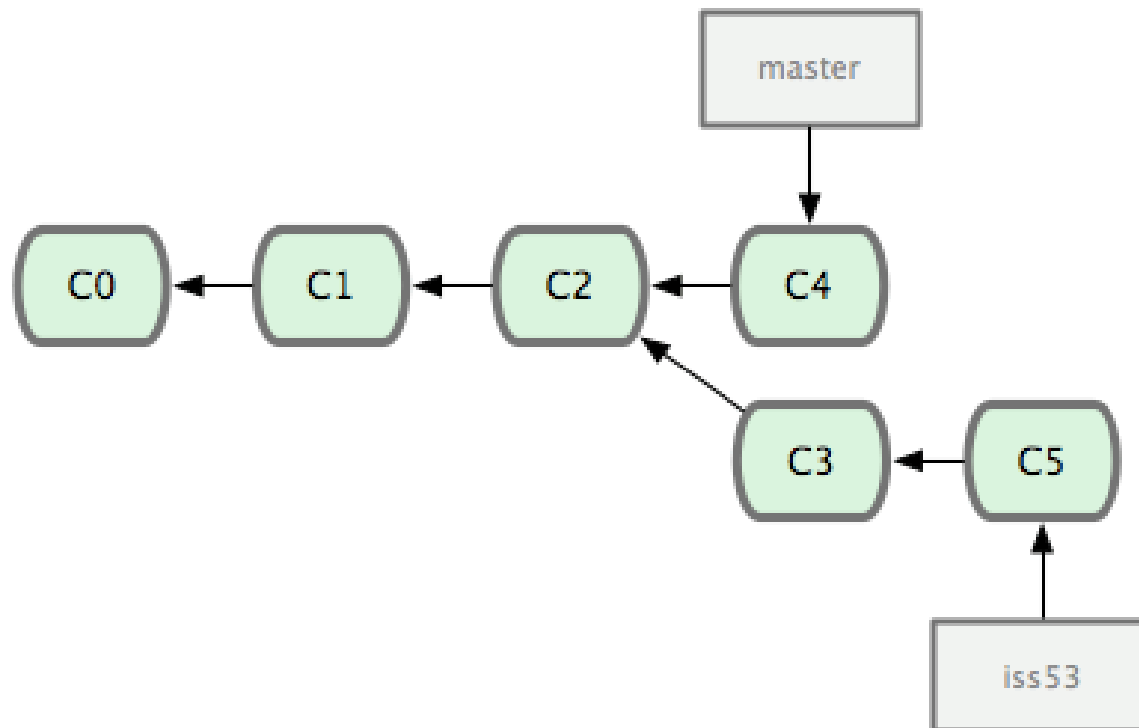
Thiago@THIAGO MTNGW64 ~/Desktop/projetos (master)
$ git checkout iss53
M      arquivo.txt
Switched to branch 'iss53'
```



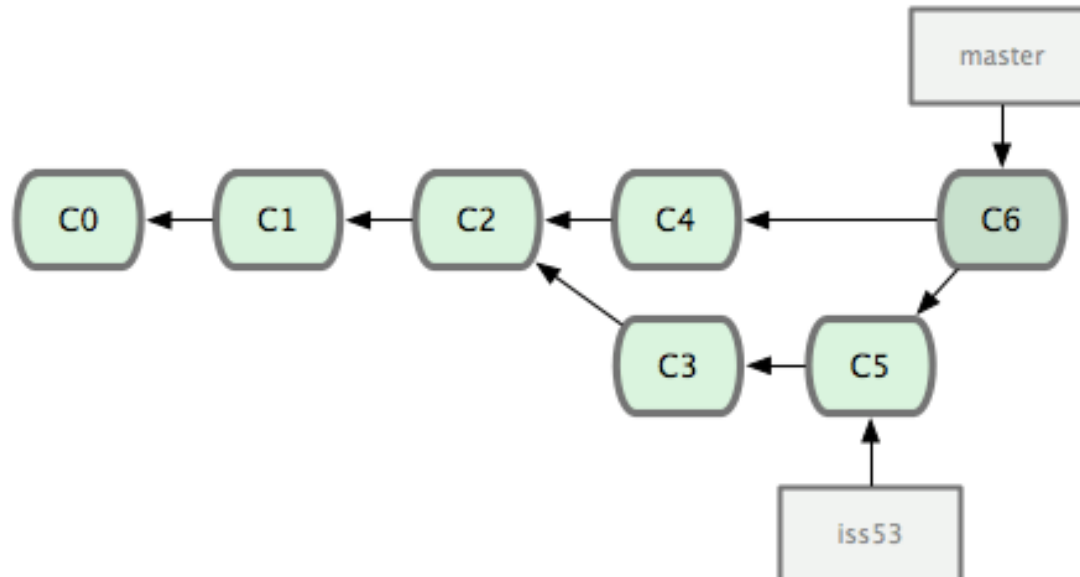


- **git merge <nome>**

- Mescla uma branch ao HEAD;



```
Thiago@THIAGO-MINGW64 ~/Desktop/projetos (master)
$ git merge iss53
Merge made by the 'recursive' strategy.
foo.txt      | 0
index.html   | 0
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 foo.txt
create mode 100644 index.html
```



- **git branch -D <nome>**
  - Apaga um branch

```
Thiago@THTAGO MTNGW64 ~/Desktop/projetos (master)
$ git branch -D iss53
Deleted branch iss53 (was 018b263).
```



As vezes ao mesclarmos uma branch ou pegarmos alterações de outros desenvolvedores podem ocorrer conflitos.

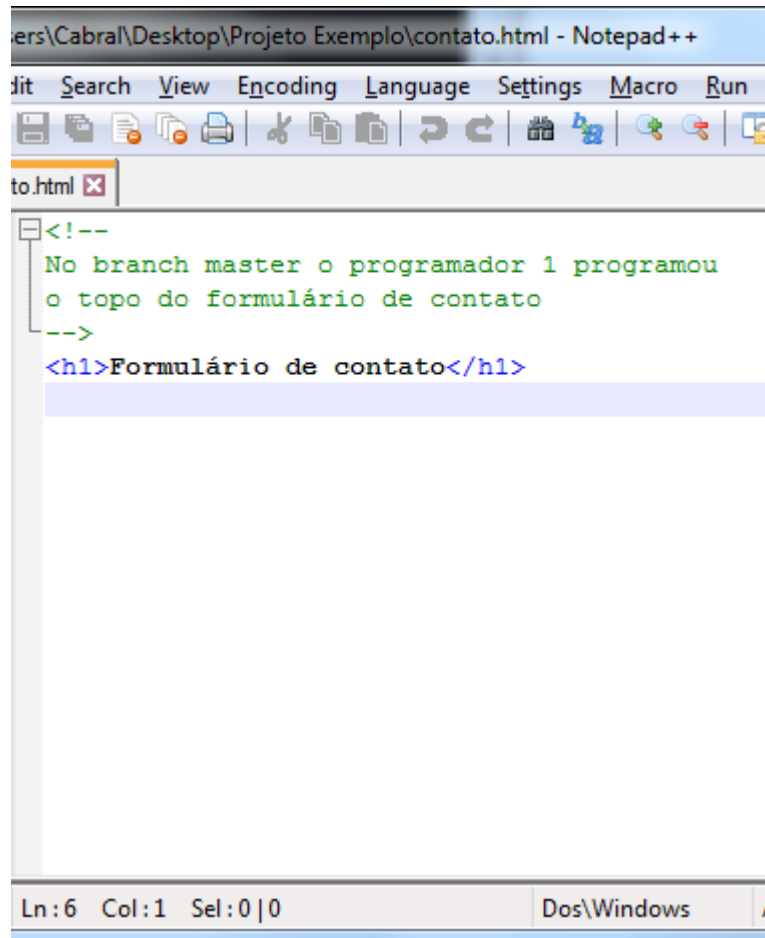
Conflito é quando o GIT não consegue decidir qual parte do código vem antes ou depois, ou se aquele trecho de código deve ficar ou sair.

Isso acontece quando um mesmo arquivo é alterado por pessoas diferentes ou em branches diferentes e depois mesclados.

```
Switched to branch 'master'  
Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)  
$ git merge dev  
Auto-merging contato.html  
CONFLICT (content): Merge conflict in contato.html  
Automatic merge failed; fix conflicts and then commit the result.
```

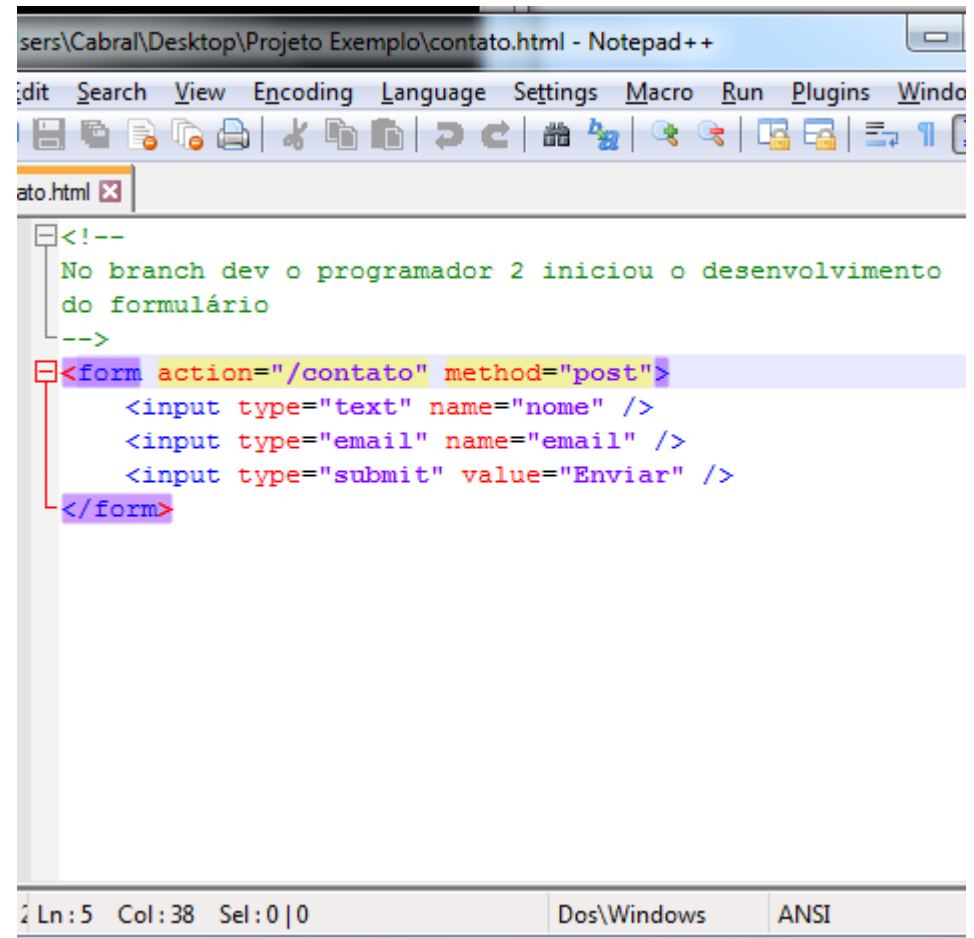
# RESOLUÇÃO DE CONFLITOS

Como foi ocasionado?



```
ers\Cabral\Desktop\Projeto Exemplo\contato.html - Notepad++
Edit Search View Encoding Language Settings Macro Run
[Icons]
to.html x
<!--
No branch master o programador 1 programou
o topo do formulário de contato
-->
<h1>Formulário de contato</h1>
```

Ln: 6 Col: 1 Sel: 0 | 0 Dos\Windows

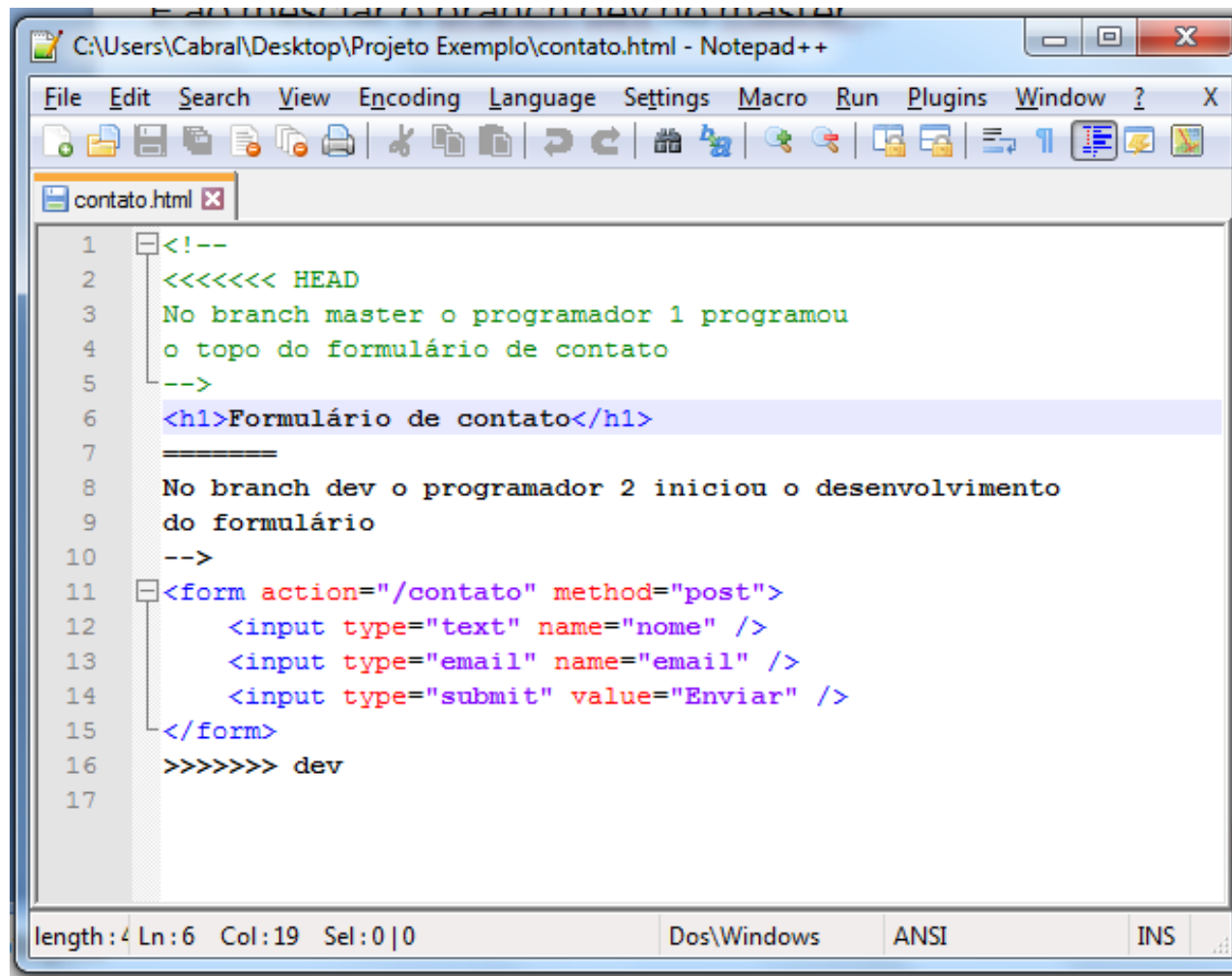


```
ers\Cabral\Desktop\Projeto Exemplo\contato.html - Notepad++
Edit Search View Encoding Language Settings Macro Run Plugins Windo
[Icons]
ato.html x
<!--
No branch dev o programador 2 iniciou o desenvolvimento
do formulário
-->
<form action="/contato" method="post">
  <input type="text" name="nome" />
  <input type="email" name="email" />
  <input type="submit" value="Enviar" />
</form>
```

Ln: 5 Col: 38 Sel: 0 | 0 Dos\Windows ANSI

# RESOLUÇÃO DE CONFLITOS

E ao mesclar o branch dev no master...



```
1 <!--
2 <<<<<<< HEAD
3 No branch master o programador 1 programou
4 o topo do formulário de contato
5 -->
6 <h1>Formulário de contato</h1>
7 =====
8 No branch dev o programador 2 iniciou o desenvolvimento
9 do formulário
10 -->
11 <form action="/contato" method="post">
12     <input type="text" name="nome" />
13     <input type="email" name="email" />
14     <input type="submit" value="Enviar" />
15 </form>
16 >>>>>>> dev
17
```

length: 4 Ln: 6 Col: 19 Sel: 0 | 0 Dos\Windows ANSI INS

# RESOLUÇÃO DE CONFLITOS

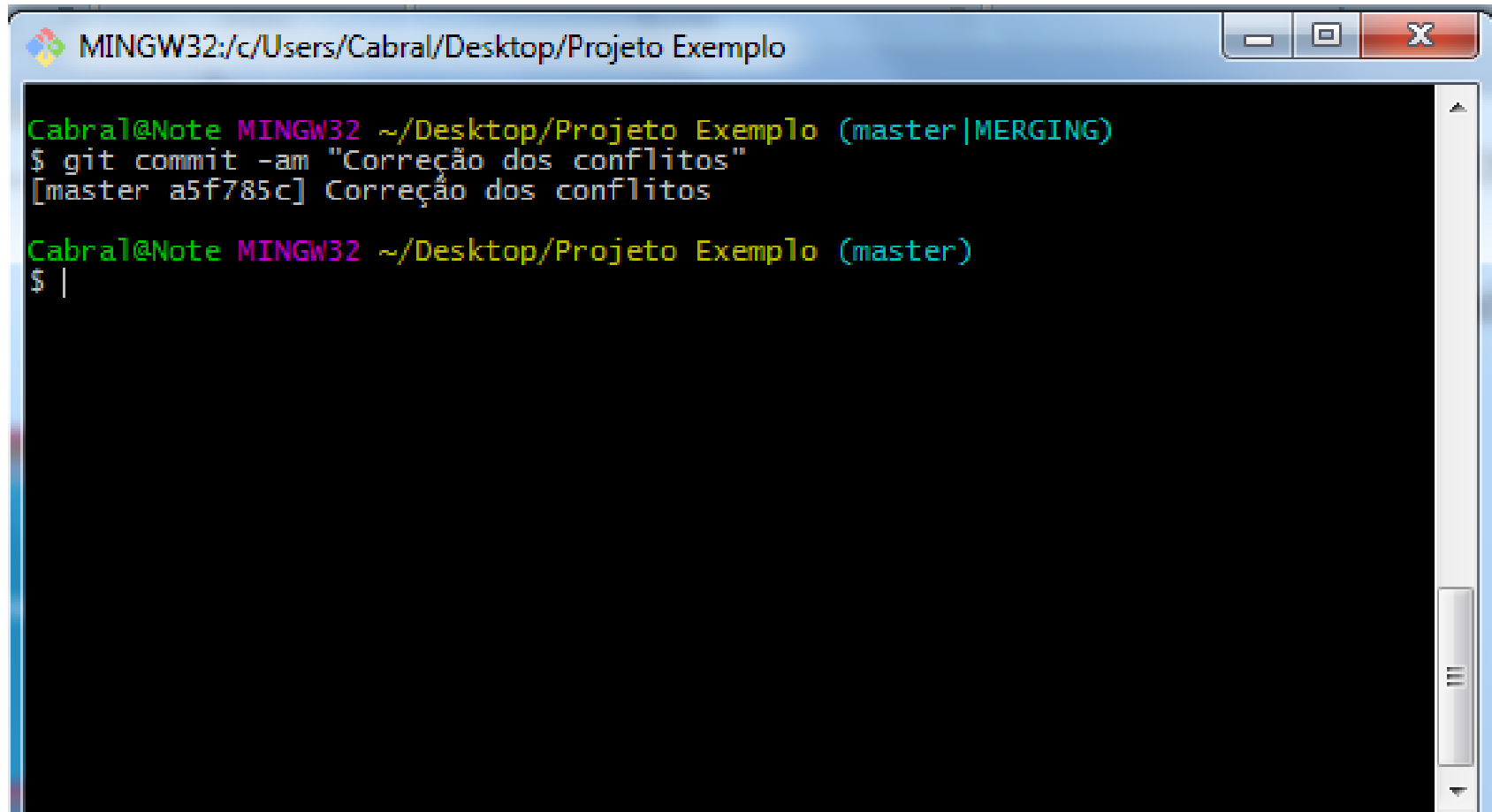
O GIT coloca por padrão o código do HEAD primeiro e o código do commit que ocasionou o conflito em seguida.

Para solucionar, deve-se editar manualmente o código-fonte para decidir qual código vem primeiro, qual sai, qual fica...

```
1 <!--  
2   No branch master o programador 1 programou  
3   o topo do formulário de contato  
4  
5   No branch dev o programador 2 iniciou o desenvolvimento  
6   do formulário  
7 -->  
8 <h1>Formulário de contato</h1>  
9  
10 <form action="/contato" method="post">  
11     <input type="text" name="nome" />  
12     <input type="email" name="email" />  
13     <input type="submit" value="Enviar" />  
14 </form>  
15
```



Após a correção de todos os conflitos, basta criar um novo commit.

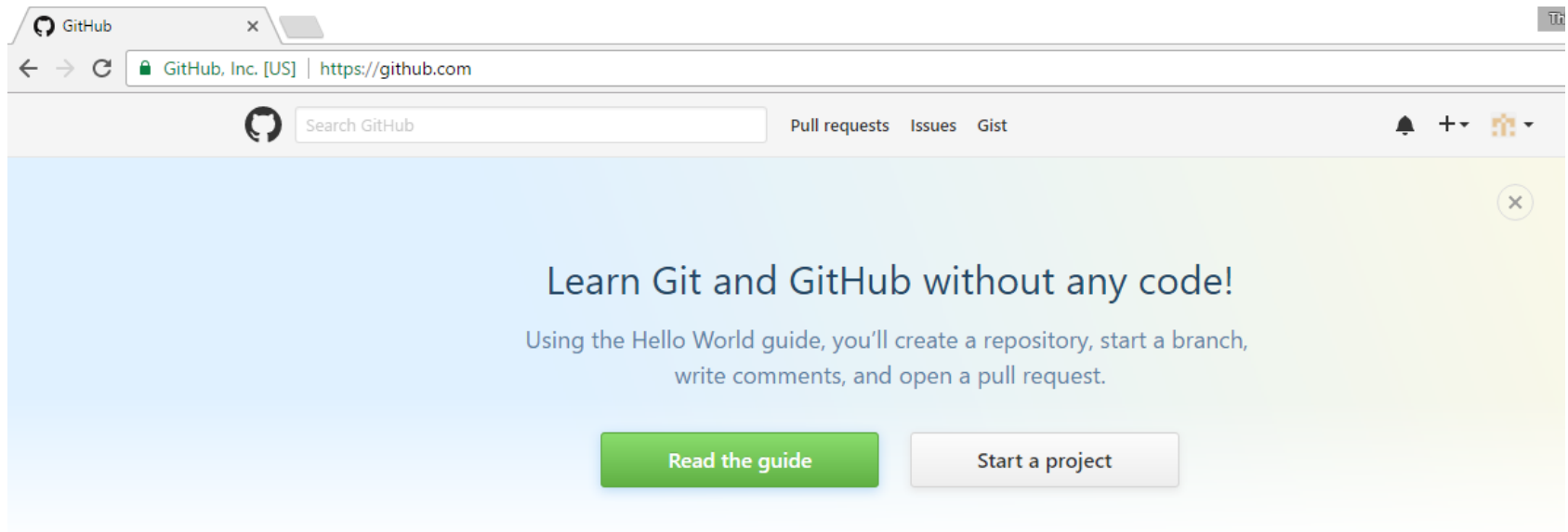
A screenshot of a Windows command prompt window titled "MINGW32:/c/Users/Cabral/Desktop/Projeto Exemplo". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt shows the following text:

```
Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master|MERGING)
$ git commit -am "Correção dos conflitos"
[master a5f785c] Correção dos conflitos

Cabral@Note MINGW32 ~/Desktop/Projeto Exemplo (master)
$ |
```



- [www.github.com](https://www.github.com)
  - Criando um repositório.



## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



thiagoyama ▾

/

Repository name

github



Great repository names are short and memorable. Need inspiration? How about **musical-potato**.

Description (optional)

Curso de git



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

- **git remote add**

- Adiciona um novo repositório remoto no Git com um nome curto;

```
Thiago@THIAGO-MINGW64 ~/Desktop/projetos (master)
$ git remote add origin https://github.com/thiagoyama/github.git
```

- **git remote -v**


- Exibe o nome e a URL do repositório remoto;

```
Thiago@THIAGO-MINGW64 ~/Desktop/projetos (master)
$ git remote -v
origin https://github.com/thiagoyama/github.git (fetch)
origin https://github.com/thiagoyama/github.git (push)
```

# ENVIANDO OS ARQUIVOS

FLAP

```
Thiago@THIAGO-MINGW64 ~/Desktop/projetos (master)
$ git push -u origin master
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 439 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/thiagoyama/github.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

 thiagoyama / github

Unwatch 1

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Pulse

Graphs

Settings

Curso de git — Edit

2 commits

1 branch

0 releases

1 contributor

Branch: master


New pull request

Create new file


Upload files

Find file

Clone or download

 thiagoyama teste

Latest commit 018b263 30 minutes ago

 arquivo.txt

teste

30 minutes ago

```
Cabral@Notebook: ~/Desktop/Projeto Exemplo (master)  
$ git push -u origin --all
```

Com o comando **git push -u origin --all** todos os branches locais são sincronizados e enviados para o repositório remoto.

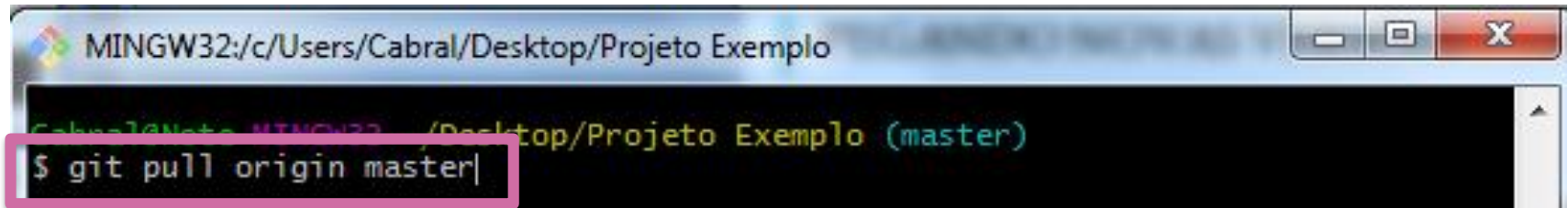
Usar apenas a primeira vez que enviar para servidor remoto.

Após usar o formato: **git push <remote> <branch>**

Ex: **git push origin master**

- **git pull <remote> <branch>**

- Atualiza o repositório local com os commits existentes no repositório remoto.



A screenshot of a Windows terminal window. The title bar shows the path 'MINGW32:/c/Users/Cabral/Desktop/Projeto Exemplo'. The terminal content shows the prompt 'Cabra1@Note: MINGW32 /Desktop/Projeto Exemplo (master)' followed by the command '\$ git pull origin master' which is highlighted with a pink rectangular box.



- **git clone**
  - Copia um repositório Git já existente;

```
Thiago@THIAGO-MINGW64 ~\Desktop/projetos-git
$ git clone https://github.com/thiagoyama/github.git
Cloning into 'github' ...
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
```



- **git rebase**
- **git fetch**
- **git show**
- **git revert**
- **git stash**
- **git cherry-pick**



- Não clonar repositórios dentro de outro repositório.
- Não manter versionado arquivos compilados (.dll, .exe, etc).
- Não manter versionado arquivos com dados sensíveis (Arquivos de configuração, senha, ambiente).
- Ignorar arquivo antes do mesmo já estar versionado.

- Deixar bem documentado as mensagens do commit e evitar mensagens genéricas como “Alterações no arquivo.txt”
- Seguir algum modelo de workflow com branches (Ex: Branch por ambiente de desenvolvimento, ou branches por funcionalidades, ou ambos)

***Copyright © 2018 Prof. Alexandre Carlos de Jesus***

***Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).***