

# RESPOSTAS

## Construção de software

### Git (exercícios)

1. `git --version`
2. Qual o efeito da execução de cada um dos comandos abaixo?
  - a. **git config -l** : lista todas as variáveis (e seus respectivos valores) setadas no arquivo "config"
  - b. **git mv a.txt b.txt** : renomeia o arquivo txt mencionado no primeiro parâmetro
  - c. **git reset --hard**: retorna os arquivos e o index para o último commit, descartando todas as alterações realizadas até o determinado commit.
  - d. **git log -27** : apresenta o log limitando o número de logs a serem apresentados em 27.
  - e. **git help** : apresenta uma lista de parâmetros para utilizar o comando git help para buscar informações sobre a utilização dos mesmos.
  - f. **git help reset** : direciona para página completa de informações sobre as possibilidades de uso do comando "reset".
  - g. **git add --all** : adiciona todos os arquivos modificados que já são "monitorados" e também que já existe alguma entrada no index. Esse comando adiciona, modifica ou remove entradas no index para se igualar à árvore que está sendo utilizada.
  - h. **git add -u** : adiciona todos os arquivos modificados e que já são "monitorados" pelo Git, ignorando os arquivos "untracked".
3. `git add [arquivo]` (ou pode-se passar parâmetros para adicionar vários) > `git commit -m "mensagem de commit"`
4. `git status`
5. `git status`
6. `git commit -m "mensagem de commit"`
7. `git checkout --teste.txt`
8. deve-se criar um arquivo **.gitignore** no diretório desejado, e dentro deste deve-se especificar que todos os arquivos dentro do diretório devem ser ignorados com o seguinte comando "nomediretorio/\*\*", assim todos os arquivos pertencentes ao diretório "nomediretorio" serão "selecionados" para serem ignorados pelo Git.
9. é necessário realizar o comando **git pull --rebase**
10. **git clone [url dorepositóriodesejado]** (url disponível na página do repositório no github)
11. `git shortlog -sne`

12. `~/.gitconfig`
13. `git init [nomedoprojeto]`
14. o nome do diretório criado é o nome informado no comando `"git init [nomedoprojeto]"`
15. `git add .`
16. SHA1 significa "Secure Hash Algorithm 1", o propósito de sua utilização pelo Git é para garantir que não haja alterações nos dados devido a corrupção nos mesmos. O SHA1 é uma tecnologia de criptografia anteriormente utilizada para garantir a segurança de dados, entretanto esta não é mais considerada segura para tratar de dados privados.
17. `git log -1 --format="%an %s %e"`
18. sim, porque o comando `git add -u` atualiza o index para se identificar com a árvore de trabalho em utilização.
19. primeiramente ele retornaria para o commit selecionado, mantendo o arquivo index e passando os arquivos alterados para "mudanças a serem commitadas". No segundo passo ele reseta o arquivo index e a árvore de trabalho sem descartar nenhuma das alterações realizadas anteriormente ao primeiro comando.
20. `git clean -f`
21. `.gitignore`
22. adicionar o prefixo `*.class` no arquivo `gitignore`.
23. <https://github.com/murilosotelo/ls2017/tree/master/jquery>

```
muril@DESKTOP-RAPJEJ2 MINGW64 ~/Desktop/Git Testes/ls2017/jquery (master)
$ git shortlog -sne
1714 John Resig <jeresig@gmail.com>
657  Timmy Willison <4timmywil@gmail.com>
579  Dave Methvin <dave.methvin@gmail.com>
336  Jörn Zaefferer <joern.zaefferer@gmail.com>
332  Julian Aubourg <aubourg.julian@gmail.com>
315  Rick Waldron <waldron.rick@gmail.com>
267  Oleg Gaidarenko <markelog@gmail.com>
262  Richard Gibson <richard.gibson@gmail.com>
250  Brandon Aaron <brandon.aaron@gmail.com>
230  Michał Gołębiowski-Owczarek <m.goleb@gmail.com>
200  Ariel Flesler <aflesler@gmail.com>
85   Mike Sherov <mike.sherov@gmail.com>
71   Colin Snover <github.com@zetafleet.com>
67   David Serduke <davidserduke@gmail.com>
59   Yehuda Katz <wycats@gmail.com>
55   Corey Frang <gnarf37@gmail.com>
47   Louis-Rémi Babé <lrbabe@gmail.com>
35   Anton Matzneller <obhvsbypqghgc@gmail.com>
34   Scott González <scott.gonzalez@gmail.com>
27   Gilles van den Hoven <gilles0181@gmail.com>
25   Timo Tijhof <krinklemail@gmail.com>
22   Robert Katić <robert.katic@gmail.com>
17   Dan Heberden <danheberden@gmail.com>
```

```
muril@DESKTOP-RAPJE12 MINGW64 ~/Desktop/Git Testes/ls2017/jquery (master)
$ git remote -v
origin https://github.com/jquery/jquery.git (fetch)
origin https://github.com/jquery/jquery.git (push)
```

- 25.
26. `git tag -l`
27. `git tag -l 2.0*`
28. cria uma assinatura (3.4-gold) para o objeto e também adiciona um comentário ("minha versão ouro") ao mesmo objeto.
29. apresenta todas as informações da tag "3.4-gold" (quem a criou, quando a criou), tanto como o comentário "minha versão ouro" e as informações do commit correspondente.
30. realiza um push da tag "3.4-gold" uma vez que o Git não faz push das tags nos pushes convencionais.
31. ele reutiliza as informações referentes ao commit passado para a realização do novo commit.
32. ele retorna para o estado em que estava no momento "HEAD", mas sem descartar o arquivo "x.txt", em vez disso ele apenas o remove da lista de "mudanças a serem commitadas", passando este a ser um arquivo "untracked".
33. o comando faz com que o arquivo "novo" sobrescreva o commitado anteriormente, o atualizando com as informações alteradas no mesmo, sem a necessidade de um novo commit.
34. **`git reset HEAD x.txt`** retorna para o estado em que estava no momento "HEAD", mas sem descartar o arquivo "x.txt", em vez disso ele apenas o remove da lista de "mudanças a serem commitadas", passando este a ser um arquivo "untracked". Enquanto **`git checkout -- a.txt`** faz com que o arquivo "novo" sobrescreva o commitado anteriormente, o atualizando com as informações alteradas no mesmo, sem a necessidade de um novo commit.