Git (exercícios)

Lista 01

1. Qual o comando para obter a versão instalada do Git?
    a. git --version
2. Qual o efeito da execução de cada um dos comandos abaixo?
    a. **git config -l**
        i. Lista todas as variáveis configuradas no arquivo de configuração do Git.
    b. **git mv a.txt b.txt**
        i. Move o arquivo a.txt e b.txt para um diretório especificado
    c. **git reset --hard**
        i. Todos os arquivos que ainda não foram commitados, mas que já estão na fila para commit são retirados dela. O repositório volta para o estado do último commit.
    d. **git log -27**
        i. Exibe o log dos últimos 27 commits
    e. **git help**
        i. Ajuda do git;
    f. **git help reset**
        i. Um manual de como usar a ajuda do Git
    g. **git add --all**
        i. Adiciona todos arquivos do diretório para fila do commit (stage)
    h. **git add -u**
        i. Atualiza os arquivos da fila do commit.
3. O fluxo “clássico” de interação com o Git é algo como “alterar um ou mais arquivos”, “acrescentar essas mudanças para serem contemplados no próximo commit” e, finalmente, executar um “commit”. Quais os comandos necessários para realizar os dois últimos “passos” desse fluxo?
    a. git add
    b. git commit
4. Qual o comando deve ser executado para identificar o que foi alterado desde o último “commit”?
    a. git log
5. Em um dado repositório, arquivos simplesmente copiados para lá, ou seja, *untracked*, podem ser exibidos/identificados com que comando?

    a. git status

6. Qual o comando para efetuar um *commit*?
    a. git commit
7. Qual o comando que devemos empregar para descartar mudanças ocorridas no arquivo teste.txt, por exemplo?
    a. `git diff | filterdiff -p 1 -x <diretorio_teste>/teste.txt`
8. O que deve ser feito para que um determinado diretório do seu repositório seja ignorado pelo Git? Faça uma busca por **.gitignore**.
    a. Para ser ignorado, o arquivo não estar commitado ou estar na fila para um commit. Verficado isso, é só acrescentá-lo no arquivo .gitignore.
9. O que acontece se o seu repositório local for acidentalmente cd removido?
    a. Os arquivos que não estavam sendo rastreados pelo git e aqueles foram marcados para um próximo commit são perdidos. É possível recuperar o repositório, caso ele tenha sido armazenado remotamente (GitHub).
10. Como clonar um repositório remoto?
    a. git clone <url_repo>
11. Em alguns cenários **git log** pode produzir extensos resultados. Se houver interesse em visualizar o histórico de um repositório, onde cada mudança é fornecida exatamente em uma única linha, qual o comando que deve ser empregado?
    a. git log --line-prefix = <prefix>
12. Em qual arquivo o Git armazena informações de configuração empregadas por usuário?
    a. .gitconfig
13. Qual o comando para criar um repositório local?
    a. git init <nome_do_projeto>
14. Qual o nome do diretório criado pelo Git quando se executa o comando **git init**?
    a. O nome passado como arguemento do comando git init.
15. Qual o comando para adicionar todos os arquivos modificados? (Aqueles para os quais **git status** identificam como **modified**?)
    a. git add .
16. O Git faz uso do valor de *hash* conhecido por SHA1. O que isto significa? Qual o propósito? O que é SHA1?
    a. SHA1 é um algoritmo de criptografia que produz hash, com o propósito de proteger os dados sensíveis e não-confidenciais dos commits.
17. Qual a palavra para indicar o último *commit* em vez do valor de *hash* SHA1 correspondente?
    a. git log -1

18. Quando se cria dois arquivos usando um editor de texto qualquer e, na sequência, executamos o comando **git add -u**, os dois arquivos criados passam de *untracked* para *new file*?
    a. Não, pois eles não tinham sido marcados para um próximo commit.
19. Qual o efeito da execução dos dois comandos abaixo, nesta ordem, em um dado repositório?
    git reset --soft HEAD~1
    git reset --hard
    a. Com o primeiro comando, é possível recuperar as informações do último commit feito. Agora, com o segundo comando, não é possível recuperar do último commit feito. Então, não é possível recuperar o último commit feito ao executar esses dois comandos em sequência.
20. Após o emprego de um ambiente integrado de desenvolvimento (IDE), é comum a criação de arquivos e diretórios. Qual o comando que podemos empregar para remover arquivos e diretórios *untracked*?
    a. git reset --hard
21. Qual o nome do arquivo no qual podemos inserir a indicação para o Git de arquivos e diretórios a serem ignorados?
    a. .gitignore
22. Quando se cria o arquivo *MinhaClasse.class* em um dado diretório e desejamos que o Git ignore não apenas este, mas arquivos *.class* em geral, por todos os membros de uma equipe que estão contribuindo com um dado projeto?
    a. Dentro do .gitignore >> *.class
23. jQuery é uma famose biblioteca em JavaScript. Consulte detalhes em http://jquery.com. O repositório Git correspondente encontra-se disponível em https://github.com/jquery/jquery.git. Faça o clone deste repositório **jqueryrepo**.
24. No repositório **jqueryrepo**, criado no passo anterior, qual o efeito do comando **git shortlog -sne**?
    a. Exibe o log do repositório com número de commits feitos por cada pessoa, o e-mail de cada um e um resumo da descrição do commit.
25. No repositório **jqueryrepo**, qual o efeito de **git remote -v**?
    a. Exibe a url remota após nome do repositório.
26. Um repositório Git pode ser etiquetado ao longo do tempo. Ou seja, *commits* específicos podem ser “marcados” ou “etiquetados” para facilitar referências posteriores. Para listar todas as “etiquetas” (*tags*) estabelecidas para um dado repositório, qual comando deve ser executado?
    a. git tag
27. Caso um dado repositório retorne muitas “marcas” ou “etiquetas” para o comando **git tag**, como retornar apenas aquelas que atendem a determinado padrão, por exemplo, iniciadas por 2.0?

a. git tag -l <pattern>

28. Qual o efeito do comando **git tag -a 3.4-gold -m “minha versão ouro”**?
    a. Cria uma tag assinada com 3.4-gold com a mensagem “minha versão ouro”
29. Após executado o comando acima, qual o efeito de **git show 3.4-gold**?
    a. Exibe todos itens de configuração presentes na tag 3.4
30. O que o comando **git push origin 3.4-gold** teria como efeito?
    a. A tag 3.4-gold seria enviada para o branch origin do repositório remoto.
31. Após executar um *commit,* qual o efeito de **git commit --amend**?
32. Esse comando corrige um último commit feito mantendo a mensagem e o autor do commit.
33. Após executar **git add x.txt**, qual o efeito de **git reset HEAD x.txt**?
    a. Esse comando reverte o commit do arquivo acima especificado.
34. Após alterar o conteúdo de um arquivo *committed* em passo anterior, qual o efeito do comando **git checkout -- a.txt**?
    a. Cria um branch no repositório de nome “a.txt”.
35. Qual a diferença entre os comandos **git reset HEAD x.txt** e **git checkout -- a.txt**?
    a. O primeiro comando reverte todas as modificações que foram feitas, mas não foram comitadas. Enquanto que o segundo, cria um branch de noem “a.txt”.

Lista 02

1. Qual o comando para que as “marcas” ou *tags* sejam enviadas para o repositório remoto? (um simples **git push** não produz este efeito)
    a. `git push origin <tag_name>`
2. Qual o nome do *branch* padrão do Git?
    a. master
3. O que o comando **git branch <branchname>** realiza?
    a. Esse comando cria um branch com o nome que foi passado no parâmetro <branchname>
4. Como criar um *branch* a partir de um *commit* específico?
    a. git branch -- merged <nome_do_commit>
5. Em um repositório, qual o efeito do comando **git branch erro1234**?
    a. Esse comando cria um branch de nome “erro1234”
6. Qual o comando para se alternar para um *branch* de nome **experimento2**?
    a. git branch -m experimento2 <novo_nome_do_branch>

7. Em um repositório com dois *branches*, **b1** e **b2**, onde **b1** é o corrente, qual o efeito do comando **git branch**?
    a. O comando lista todos os branches.
8. O que o comando **git checkout -b novobranch** faz?
    a. Realiza o merge do branch novobranch com o branch principal.
9. **Qual a função do comando git branch -d teste**?
10. Durante o desenvolvimento de um software é comum, por exemplo, utilizar um novo recurso por meio de experimentação. Talvez uma nova tecnologia, uma nova biblioteca que pode ser útil ao que está em desenvolvimento, ou até mesmo uma nova versão de um produto já empregado. Para que o uso deste novo recurso não interfira com o que é considerado pronto, um *branch* pode ser criado para a experimentação. Código que for criado para a experimentação existirá apenas no *branch* criado. Se eventualmente o experimento demonstrar um resultado satisfatório, as alterações realizadas no *branch* poderão ser incorporadas no que é considerado pronto, ou seja, no *branch* principal (*master*). Esta última ação é conhecida por *merge.* Neste item, apresente uma sequência de comandos que simula um caso simples de criação e uso seguido de *merge* empregando um *branch* para ilustrar uma experimentação conforme acima. A sequência deve incluir, obrigatoriamente: (a) criação de um ou mais *branches;* (b) chaveamento para pelo menos dois *branches* e (c) *merge*. Para simular alteração em um arquivo, basta simplesmente fornecer algo como **Arquivo <nome> é alterado**. O que foi fornecido em negrito representa uma ação que altera um arquivo cujo nome é fornecido entre o sinal de menor e o de maior.
    a. Git branch projeto1 e Git branch projeto2
    b. Git checkout projeto1 e Git checkout projeto2
    c. **Arquivo <a.txt> é alterado no branch projeto1, arquivo <a.txt> é alterado no branch projeto2.**
    d. git merge