Especialização em Desenvolvimento Web com *Frameworks* Modernos

WEB11 — Versionamento de código com Git **Lista de exercícios 01** **2022**

Aluno: **Nota:**

1. São argumentos **verdadeiros e favoráveis** à adoção de VCSs:
(não marque afirmações falsas, e não marque argumentos contrários) (/ 7)

( ) São as IDEs (*Integrated Development Environment*) mais utilizadas no planeta
( ) Os VCSs estão hoje na base do método de trabalho do mercado
( ) Exigem investimento de tempo em treinamento e uso de ferramentas específicas
( ) Resultam em ganhos de produtividade, integridade e mensurabilidade
( ) Padronizam mecanismos de *upload* e revisão de código entre colaboradores
( ) Duplicam e "zipam" a pasta do projeto automaticamente a cada alteração
( ) Exigem investimento a titulo de taxa de licenciamento do *software*

2. São VCSs *stand-alone* (isto é, softwares **independentes**, cuja funcionalidade **primária** é versionar):
(marque V para verdadeiro, F para falso) (/ 6)

( ) Git
( ) Wikipedia
( ) Mecanismo de versionamento do LibreOffice Writer
( ) SVN
( ) Mercurial
( ) Google Drive

3. Marque a opção que possui apenas pares de sinônimos verdadeiros no âmbito do Git:
(/ 1)

a) Versão = Revisão Projeto = *Commit*
b) *Commit* = Revisão *Commit* = Identificador de repositório
c) Versão = Revisão Projeto = Repositório
d) *Commit* = Revisão Revisão = Identificador de projeto
e) *Commit* = Versão Revisão = Repositório

4. Contém apenas afirmações corretas **no âmbito do Git**:
(/ 1)

a) Cada estado memorizado do projeto constitui um **repositório**; cada estado não memorizado constitui uma **versão**.
b) Versões e revisões são **sinônimos**; o termo *commit* também é utilizado para denotar **versão**.
c) O Git armazena apenas versões, **bloqueando** revisões e repositórios, o que o torna mais **ágil**.
d) No Git, os *commits* possuem identificador, autor, data e hora, **entre outros** atributos; o identificador do **repositório** é um *hash* SHA-1 de 40 caracteres.
e) **Revisões** podem ser criadas, recuperadas, versionadas, descartadas e mescladas; no Git, **essas operações com revisões** são feitas através do comando **`git checkout`**.

5. Liste o(s) comando(s) necessário(s) para instalar, numa máquina Debian ou derivada, o **Git e o suplemento** que traz a funcionalidade de autocompletar comandos do Git. Pressuponha gerenciador de pacotes **`apt`** já configurado com o repositório **`ppa:git-core/ppa`**.

(/ 5)

6. Após a instalação do Git, quais comandos devem ser executados para configurar corretamente o nome e o e-mail do desenvolvedor para todos os repositórios da máquina?
(marque V para verdadeiro, F para falso)

(/ 5)

( ) **`git config --global username “Nome do desenvolvedor”`**
( ) **`git config --global user.name “Nome do desenvolvedor”`**
( ) **`git config --global user.email “desenvolvedor@empresa”`**
( ) **`git config --global set user.name “Nome do desenvolvedor”`**
( ) **`git config --global set user.email “desenvolvedor@empresa”`**

7. Liste as **ações** e os **comandos** necessários para obter o repositório Git **fictício** do endereço https://github.com/web-git/aula1, descobrir em qual *commit* introduziu-se a última modificação no arquivo **`README.md`** e então fazer *checkout* nesse *commit* (**suponha** que o *hash* inicie com 1a2b3c4e5f).
Dica: teste os comandos em um repositório **real**, para garantir que você não esqueça de nada! Nem todos os comandos são do Git, algum(s) é(são) do próprio sistema.
Importante: pode ser necessário introduzir em sua resposta uma breve explicação de como você deve interpretar (ler) o *output* do comando para atingir a finalidade proposta no enunciado.
Exemplo: “deve-se digitar o comando **`git xyz`** e procurar pela linha onde se lê **`informacao-importante`**”.

(/ 5)