INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL, I.P.
CENTRO DE EMPREGO E FORMAÇÃO PROFISSIONAL DO PINHAL INTERIOR NORTE
SERVIÇO DE FORMAÇÃO PROFISSIONAL DE ARGANIL
C-EFPI – J+D – 2024 – *Linguagens de Programação – Programação Web*

C-EFPI | J+D | Linguagens de Programação – Programação Web

UFCD 10789

Metodologias de desenvolvimento de software

Resumo *Git*

— Endereço da ferramenta em linha [*online*] *Padlet*:

https://padlet.com/aimfonseca/ProgWeb

— Lista de procedimentos-chave já realizados:

- ✓ Criação da conta no plataforma online *GitHub* e acesso direto pelo *browser*;
- ✓ Transferência e instalação do programa do *Git Bash*;
- ✓ Transferência e instalação do programa *GitHub Desktop*;
- ✓ Transferência e instalação do *Microsoft Visual Studio Code* (o 'nosso' *IDE* de eleição… ☺);
- ✓ Criação de uma pasta apropriada no nosso computador, para guardar os repositórios que vamos criando.

Resumo *Git*

O *Git Bash* é um programa (*software*) que combina as funcionalidades do *Git* com um ambiente *Bash*.

A) Verdadeiro **B)** Falso

Multiple Choice

Resumo *Git*

— Aceda ao endereço abaixo indicado e efetue a atividade proposta:

https://h5p.org/node/1505189

— Quando tiver terminado, clique no botão 

e depois efetue uma captura de ecrã com a sua resolução;

— Guarde a imagem resultante dessa captura com o nome a iniciar em "*2024-09-27-*" e adicione as iniciais do seu nome (exemplo, para o formador seria '*2024-09-27-AF*';

— Publique a imagem na sua seção do *Padlet* (mas ***só*** quando o formador der a atividade como concluída… ☺).

Resumo *Git*

Algumas coisas (isto e aquilo...) sobre Git e derivados.

Leia as frases e arraste as palavras de forma a preencher os espaços vazios.

Git ✔ é um sistema de controlo ✔ de versões ✔ amplamente utilizado para fazer a gestão de alterações ✔ no código-fonte dos projetos ✔. Permite que várias pessoas colaborem num projeto, mantendo um histórico das modificações.

Bash ✔ é um shell (ou interpretador ✔ de comandos) muito utilizado em sistemas operativos ✔ Unix e Linux. O Bash ✔ facilita a execução de comandos, scripts e a automatização de tarefas ✔.

No contexto do ambiente Windows ✔, o Git Bash é um software ✔ que combina as funcionalidades do Git com um ambiente ✔ Bash. Como o Windows não tem o Bash ✔ de forma nativa (ao contrário do que acontece no Linux), o Git Bash é fundamental ✔ para permitir aos utilizadores trabalharem com comandos ✔ Git num ambiente de terminal ✔ semelhante ao Unix.

A

17/17

Resumo *Git*

Em determinadas circunstâncias podemos ter interesse em ignorar alguns ficheiros (ou mesmo algumas pastas).

Nesta situação, podemos (e devemos…) usar o seguinte ficheiro:

A) *.ignorarGit* **B)** *.gitignore* **C)** *.commit*

Multiple Choice

Resumo *Git*

O *GitHub Desktop* é ideal para quem deseja usufruir do *Git* e *GitHub* de forma mais visual e intuitiva, seja para melhorar a *produtividade* seja para *aprender* a usar *Git* sem os desafios iniciais da linha de comandos.

Esta afirmação é:

A) *Verdadeira*

B) *Falsa*

Multiple Choice

Resumo *Git*

— O próximo passo nesta sequência de aprendizagem passa por *apagar* um determinado ficheiro…

— Apagar um ficheiro pode acontecer de forma intencional (*por razões válidas*…) ou pode acontecer por *acidente* (quem nunca apagou nada sem querer que atire a primeira pedra… ☺);

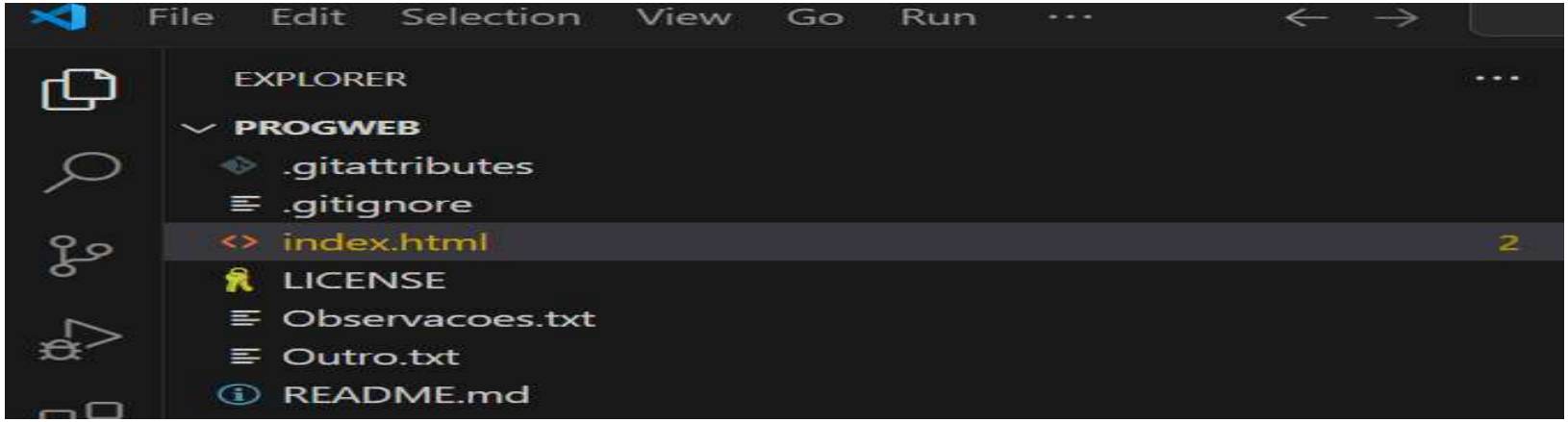
— Então, nesta próxima atividade vamos *eliminar* um ficheiro, vamos verificar que tal foi refletido no nosso projeto e depois vamos recuperar o dito ficheiro;

— Relembrar que um "*commit*" no *GitHub* é um "*instantâneo*" de todas as alterações efetuadas no nosso projeto;

— Sempre que executamos um *commit* estamos a *gravar* uma *nova* versão do nosso projeto, juntamente com uma mensagem que explica o que foi alterado (escrever esta mensagem é uma boa prática, principalmente em projetos com uma dimensão razoável…).

Resumo *Git*

— O ficheiro que vamos apagar acidentalmente é o *index.html* (que temos no repositório "*ProgramacaoWeb*" ou "*ProgWeb*" ou outro que tenha sido criado nas sessões anteriores):



— Para não nos perdermos nos passos a executar, acedam ao *link* abaixo e efetuem a transferência do vídeo "*GitProgWebDelete.mp4*".

Resumo *Git*

— Para cimentar este procedimento vamos apagar mais 2 ficheiros:

- apagamos o ficheiro *License* (no *Visual Studio Code - VSC*);

- depois, no *GitHub Desktop* executamos o *commit*, com a mensagem "*apaguei um ficheiro interessante*" e efetuamos o *Fetch Origin*;

- depois, apagamos um outro ficheiro (pode ser o *Readme.md*) no VSC;

- no *GitHub Desktop* executamos o *commit*, com a mensagem "*apaguei outro ficheiro interessante*" e efetuamos o *Fetch Origin*;

- para recuperar os ficheiros apagados *acidentalmente* (☺) procedemos da mesma forma que já fizemos;

Resumo *Git*

- neste ponto podemos/devemos efetuar um "*Pull*" [buscar – *fetch* e integrar – *merge*] se já tivermos terminamos o nosso trabalho;

- este procedimento é usado para *sincronizar* o nosso código (*local*) com as mudanças mais recentes que possam ter sido efetuadas no *GitHub;*

- isto é muito útil quando se trabalha em colaboração com outros programadores.

Resumo *Git*

O *GitHub Desktop* é uma aplicação gráfica para *Git* que se liga diretamente ao *GitHub*. É perfeito para quem quer ter uma visão geral do histórico das alterações no código. Isto é verdade. Digo eu…

Já usamos e, se não usamos, pelo menos já vimos, vários termos ou palavras ou comandos que fazem parte deste tema que estamos a desenvolver.

O que se pretende com esta atividade é colocarem "*palavras*" que vos tenham chamado mais à atenção até agora!

Word Cloud

Git Branching

— Estamos a desenvolver um website, com várias páginas *html*, *css* e outras;

— Criamos um *branch* (ramo) com o nome "*nova_pagina*" para adicionar uma nova página ao nosso projeto;

— Enquanto trabalhamos nessa página, um dos programadores da nossa esquipa deteta um *bug* numa das outras páginas;

— Bem, é melhor criar outro *branch* chamado "*corrigir_bug*" para resolver este novo problema;

— No final, quando ambos os *branches* estiverem prontos, podemos juntá-los ao *branch* principal (o tal *main* que criamos no início) e atualizar assim o nosso projeto final…

— Vamos ver uma parte de um vídeo interessante, obtido em:

https://youtu.be/8Dd7KRpKeaE?si=8IuIDTI5FKEZUBUd

Git Branching

Git Branching

— Para cimentar este conceito vamos criar 2 (dois) novos *branches*:

- *nova_pagina* e *novo_css*;

— Em cada um dos *branch* vamos criar um qualquer ficheiro, do tipo *html* ou *css*;

— No final do nosso trabalho (praticar, neste caso…) efetuamos o *merge* (juntar) em cada um dos *branch* para o *branch* principal (o tal *main*…) de forma a que os novos ficheiros passem a fazer parte do nosso projeto.

— Dúvidas?

— Já agora nestes situações temos um procedimento chamado *pull request*, que basicamente não é mais do que a *verificação, confirmação* e *aceitação* das alterações efetuadas…