UNIVERSIDADE DA BEIRA INTERIOR
DEPARTAMENTO DE INFORMÁTICA

Guia Laboratorial #1

Curso: *Plataformas de Desenvolvimento Web* – Professor: *Dr. João Neves*

1. Git

Este guia laboratorial pretende que o aluno crie o seu primeiro repositório git, e que simultaneamente hospede o mesmo no GitHub. O Git é um sistema de controlo de versões que permite registar as alterações ao código fonte de um software. Por outro lado, o GitHub é uma plataforma que permite hospedar repositórios na cloud.

Passo 1

Comece por instalar o git no seu computador. Escreva abaixo os passos que foi necessário fazer para instalar este software.

1º-Acessar o site "https://git-scm.com/downloads" e instalar a versão 2.33.00 para o SO Windows
2º- Prosseguir com a instalação concordando com todas as definições pré-estabelecidas

Passo 2

Verifique qual a versão do git que acabou de instalar, e configure o nome e email do seu utilizador no git.

Versão: 2.33.00 - Windows
Utilizador: ACER@DESKTOP-LS1K2BI
Passe: *******

Passo 3

Crie um repositório git, adicione um ficheiro README com algum texto, e de seguida execute o comando que permite ver o estado do repositório.

git init -> Cria um repositório git
git status -> permite ver o estado do repositório

Passo 4

Adicione agora o ficheiro criado à staging area do git.

```
git add README.txt
```

Passo 5

Remova agora o ficheiro da staging area do git.

```
git rm --cached README.txt
```

Passo 6

Adicione agora o seguinte ficheiro HTML ao seu repositório.

```
<!DOCTYPE html>
<html>
<head>
    <title>PDW Assignment 1</title>
</head>
<body>

    <h1>ABBA Songs</h1>
    <hr>

  <ul>
    <li>The Winner Takes it All</li>
    <li>S.O.S</li>
    <li>One of Us</li>
  </ul>

</body>
</html>
```

Passo 7

Adicione agora todos os ficheiros que estão na stagging area à história de modificações do repositório.

```

```

Passo 8

Até agora o repo criado está apenas na nossa máquina local. De modo a torná-lo acessível a partir de outros dispositivos vamos hospedar o nosso repo no GitHub. Para isso, crie uma conta no GitHub, e de seguida crie também um repositório.

Repositorio remoto --> PDW

Passo 9

Crie agora a ligação do seu repo local ao repo remoto, e de seguida faça o envio dos ficheiros locais para o repo remoto. Caso não consiga fazer login através de password leia com atenção a seguinte página https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/.

```
git remote add origin https://github.com/Henrique44609/PDW.git
git branch -M main
git push -u origin main
```

2. Colaborando no mesmo repositório

Até agora vimos como criar um repositório local e remoto, adicionando ficheiros ao mesmo e mantendo um registo das modificações efetuadas. Contudo, é muito incomum que uma só pessoa trabalhe num projeto/repositório, o que torna crucial perceber como usar o git para permitir múltiplos utilizadores contribuírem com alterações para um repositório.

Passo 1

Peça a um colega que utilize um comando git para copiar o seu projeto/repositório para a máquina local dele.

```
Numa pasta á parte:
git clone https://github.com/LuisAlmeida101/PDW.git
```

Passo 2

Peça ao colega que edite o ficheiro HTML alterando a tag *ul* para *ol*, e de seguida peça-lhe que registe a alteração realizada com a mensagem "mudança do tipo de lista", atualizando o repositório remoto.

Passo 3

Agora, no seu repositório local, altere a lista do ficheiro HTML para um lista descritiva, e de seguida registe as alterações no repositório local com a mensagem "alteração para lista descritiva".

```
<dl>
  <dt>S.O.S</dt>
  <dd> - a dramatic song </dd>
  <dt>One of Us</dt>
  <dd> - a sad song </dd>
  <dt>The Winner Takes it All</dt>
  <dd> - a super sad song </dd>
</dl>
```

Passo 4

Neste momento, o seu repositório local tem um registo de alterações diferente do que está no repositório remoto. Como irá o git tratar desta questão quando tentar enviar as alterações para o repositório remoto? Após tentar submeter o seu repo local para o repo remoto, siga a sugestão do git e tente primeiro integrar as alterações remotas com as alterações locais.

Passo 5

Quando se tenta integrar as alterações remotas no nosso repositório local, o git tenta automaticamente fazer *merge* entre as diferentes modificações realizadas nos dois repositórios. Contudo, existem situações em que não é possível fazer isto de forma automática pelo fato de as modificações serem sobre as mesmas secções de código. Verifique o que aconteceu ao ficheiro .html em conflito, e tente perceber o significado das *tags* adicionadas ao ficheiro.

```
<<<<<<< HEAD
TEXT 1
=======
```

```
TEXT 2
>>>>>>> new_branch_to_merge_later
```

A tag "<<<<<<<" sinaliza o começo do conflito de merge no arquivo
De seguida, a tag "<<<<HEAD" onde serão vistas as alterações do branch HEAD (TEXTO1). "==========" diz respeito á zona de delimitação entre as minhas alterações e as alterações no outro branch. Por último,

a tag ">>>>>>>>(Nome do Branch)" que sinaliza as alterações do outro Branch (TEXTO2)

Passo 6

Investigue como pode resolver o conflito e apresente os comandos que permitem por exemplo aceitar as modificações do seu colega em detrimento das suas.

Para solucionarmos o conflito, é necessário escolher se desejamos excluir ou manter o arquivo removido num novo commit

Passo 7

Neste exemplo, os conflitos são simples de resolver, contudo, num projeto real, a complexidade da resolução seria muito maior. Assim, é mais prático usar uma ferramenta que permita decidir como resolver cada conflito através de uma interface gráfica. Investigue que ferramenta poderá utilizar para este propósito, e resolva o conflito.

Passo 8

A ferramenta de diff e merge usada é chamada de "Meld"

Após ter efetuado a resolução do conflito, o último commit do seu repo e do repo remoto deve indicar que este resultou de um merge entre dois commits anteriores (confirme que isto foi mesmo assim). Apesar tudo parecer certo, tal como na vida, nem sempre os projetos seguem um caminho linear, existem indecisões e arrependimentos. Felizmente, o git dá-nos a possibilidade voltar atrás no nosso registo de modificações. Assim, pesquise que comando poderá fazer com que o nosso repo volte ao estado anterior ao commit realizado pelo seu colega.

O comando usado para revisar commits antigo é o "git log" que apresenta uma lista de commits mais recentes. Caso o commit requerido se encontre noutra ramificação, executamos o comando "git log --branches=*, permitindo-nos visitar outra ramificação. Após encontrar-mos o commit procurado, executamos o comando

"git checkout" que fará com que seja carregada uma captura instantânea do commit salva na máquina de desenvolvimento.

3. Markdown

Markdown é uma linguagem de marcação para a criação de texto formatado. Esta linguagem é hoje em dia amplamente usada para a criação de ficheiros README e para escrever a documentação de projetos. Assim, vamos criar um README para o seu projeto usando Markdown.

Passo 1

Pesquise como usar markdown no seu README e tente replicar o exemplo abaixo.

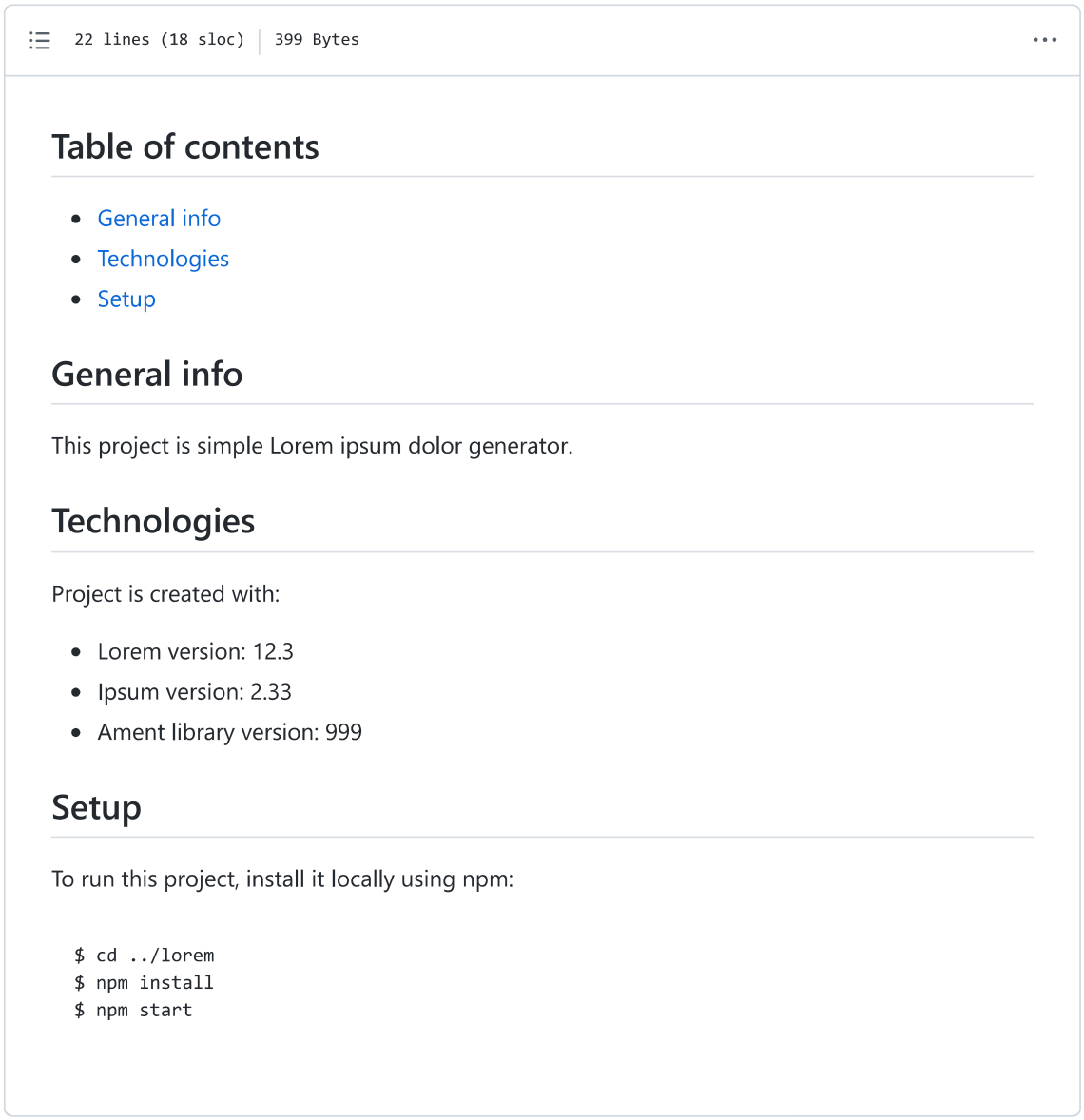

Figure 1: Exemplo de um ficheiro README criado usando Markdown.

EM BAIXO...

```
# Table of contents
* [General info](##general-info)
*  [Technologies](##technologies)
*  [Setup](##setup)

## General info

This project is simple Lorem Ipsum dolor generator

## Technologies

Project is created with:
* Lorem version: 12.3
* Ipsum version: 2.33
* Ament Library version: 999

## Setup
To run this project, install it locally using npm:

$ cd ../lorem
$ npm install
$ npm start
```