Parte Única — Questões Teóricas e Comandos Git Responda todas as questões abaixo no mesmo arquivo de entrega.

1. O que é Git e por que ele é considerado um sistema de controle de versão distribuído?

R=O Git é um Sistema de Controle de Versão (VCS) usado para rastrear e gerenciar todas as alterações no código de um projeto ao longo do tempo.

O Git é um Sistema de Controle de Versão (VCS) usado para rastrear e gerenciar todas as alterações no código de um projeto ao longo do tempo.

Ele é considerado Distribuído porque cada desenvolvedor tem uma cópia completa do histórico do projeto (o repositório) em seu computador local.

2. Para que servem os seguintes comandos: git init , git status e git config ?

R=git init (iniciar um novo repo Git na sua maquina)

git status (verificar o status dos seu arquivos no repo local informado quais arq. Foram modificados , se ha commits pendentes e em qual branch você esta trabalhando)

git config (Para configurar o git na sua maquina definindo suas variáveis de controle)

3. Explique o papel dos comandos git add e git commit dentro do processo de versionamento.

R= git add (adc. Os arquivos para ficarem prontos para subir para o repositório)

git commit(e o comentário que colocamos para saber oque foi feito nessa alteração)

4. Criação de repositório remoto: Explique o fluxo completo para criar um repositório no GitHub e conectar seu projeto local a ele.

R=Criação do Repositório no GitHub (Remoto)

1. **Acessar e Criar:** Faça login no GitHub, clique no ícone `"+"` (Novo repositório) no canto superior direito.
2. **Nome e Opções:** Dê um **nome** ao repositório (ex: `meu-projeto-js`). Escolha se ele será **Público** (visível a todos) ou **Privado**.
3. **Inicialização:** Deixe as opções "Add a README file" e "Add .gitignore" **desmarcadas** se você já tiver um projeto local existente.
4. **Finalizar:** Clique em **"Create repository"**. O GitHub exibirá uma tela com instruções (os comandos que você usará a seguir).

5. Qual a diferença entre git clone e git pull ?

R=git clone (e para quando ainda não temos o projeto na nossa maquina e queremos copiar ele para fazer algo com ele)

git pull(e para quando você está trabalhando numa branch ou com mais pessoas e precisa atualizar o código que já consta na sua maquina)

6. O que é uma branch e por que ela é importante no desenvolvimento colaborativo?

R= e uma ramificaçao como se fosse um galho de uma arvore ela e importante pois através de galhos cada desenvolvedor trabalha na sua fecture sem um atrapalhar o ou outro, ou mexer no código em produção e quebrar ele. e no fim depois do codigo ser testado e revisado se faz o merge para juntar e ficar tudo organizado.

7. Escreva os comandos Git para realizar as seguintes ações: Criar uma branch chamada desenvolvimento Trocar para essa nova branch Voltar para a branch principal Fazer merge da branch desenvolvimento na branch principal

R= git branch desenvolvimento

```
git checkout desenvolvimento
git checkout main
git merge desenvolvimento
```

8. O que é um conflito de merge? Em que situações ele costuma acontecer?

R=Um **conflito de *merge*** ocorre quando o **Git não consegue combinar automaticamente** as alterações feitas na **mesma linha ou bloco de código** em duas *branches* diferentes. O Git para o processo e exige que o desenvolvedor **resolva manualmente** qual versão do código deve ser mantida.

Costuma acontecer principalmente quando:

1. **Dois desenvolvedores** modificam as **mesmas linhas** de um arquivo.
2. Um desenvolvedor **renomeia** um arquivo enquanto outro o **modifica**.

9. Para que serve o arquivo .gitignore e por que ele é importante em projetos?

R= são importantes pois esses são os arquivos de configuração e de dados sensíveis ficam tudo ai dentro para que não pesse no github com coisas desnecessárias e não exponha dados sensiveis

10. Qual a diferença entre os comandos git log e git diff ?

R=Git log (mostra tudo sobre o log que foi feito nesse repositorio = historico)
git dif (te diz **EXATAMENTE O QUE** foi mudado no conteúdo dos arquivos.)

11. Explique resumidamente como publicar um projeto estático usando GitHub Pages.

R=**Crie e Sincronize:** Certifique-se de que seu projeto estático (que deve incluir um arquivo **`index.html`** no diretório raiz) esteja **commitado** e **enviado** (`push`) para o seu repositório no GitHub.

---

2. Configuração no GitHub

1. **Acesse as Configurações:** No seu repositório do GitHub, clique na aba **"Settings"** (Configurações).
2. **Vá para Pages:** No menu lateral esquerdo, clique em **"Pages"**.
3. **Selecione a Fonte:** Em **"Source"** (Fonte), mantenha a opção **"Deploy from a branch"** (Implantar de um *branch*).
4. **Escolha o Branch:** No dropdown **"Branch"** (Ramo), selecione o *branch* que contém seu código final (geralmente **`main`** ou **`master`**). **Salve:** Clique em **"Save"**.

3. Publicação

1. **Aguarde:** O GitHub leva alguns minutos para processar os arquivos.
2. **Confirmação:** Após a publicação, a página de **Pages** mostrará uma mensagem como: "Your site is published at **`https://www.odoo.com/pt_BR/app/website`**".
3. **Acesse:** Seu projeto estático estará acessível publicamente no formato: `https://[seu-usuario].github.io/[nome-do-repositorio]`.

12. Cite duas boas práticas ao escrever mensagens de commit.

R=**Linha de Assunto (Corpo) Curta e Imperativa (Obrigatório):**

- A primeira linha da mensagem deve ser um resumo conciso (idealmente **menos de 50 caracteres**) e escrita no **modo imperativo** (como se fosse uma ordem).
- **Corpo da Mensagem Detalhado**
- Deixe uma **linha em branco** após o assunto.
- No corpo da mensagem, explique o **porquê** da alteração e forneça contexto, como a **motivação** (qual *bug* ou *issue* resolveu) e os **efeitos colaterais** ou decisões tomadas. Mantenha o corpo em parágrafos de texto normal, idealmente **quebrando linhas a cada 72 caracteres**.

13. Explique, com suas palavras, a diferença entre fork e pull request no GitHub.

R=Fork**:** Você faz uma cópia do projeto (Projeto Original -> Seu Fork).

- Desenvolvimento**:** Você faz alterações na sua cópia.

  Pull Request: Você envia um pedido de volta: "Por favor, peguem minhas mudanças e mesclem no projeto de vocês."

14. Escreva os comandos Git para: Enviar sua branch atual para o repositório remoto Trazer atualizações remotas para o repositório local

R=git push origin main
git pull origin main

Entrega Envie um único arquivo contendo todas as respostas: atividade_git.pdf , atividade_git.docx ou atividade_git.txt (Opcionalmente compactado em .zip ou .rar)