GIT

📌 O que é Git e para que serve?

Sistema de **controlo de versões** que permite gerir mudanças em código de forma eficiente. Usado para:

✅ **Acompanhar mudanças no código** ao longo do tempo.

✅ **Trabalhar em equipa** sem sobrescrever o trabalho dos outros.

✅ **Reverter alterações** caso algo dê errado.

✅ **Criar diferentes versões do projeto** (branches) para testar novas ideias.

🔷 Principais conceitos do Git

1️⃣ Repositório (`repository`)

É onde o código-fonte e o histórico de versões são armazenados. Pode estar **localmente** (no seu computador) ou **remotamente** (GitHub).

🔷 **Exemplo de criação de um repositório local:**

```
git init
```

Isto cria um repositório Git na pasta atual.

2️⃣ Commit

Cada **commit** representa uma versão do código. Ele guarda as alterações feitas desde o último commit.

🔷 **Criar um commit:**

```
git add .  # Adiciona os arquivos para o commit
git commit -m "Descrição das mudanças"
```

📌 O `-m "mensagem"` é uma descrição curta do que foi alterado.

3️⃣ Branch

Um **branch** (ramo) é uma linha separada de desenvolvimento, permitindo que diferentes pessoas trabalhem em **funcionalidades sem afetar o código principal**.

4️⃣ Merge

O **merge** combina as mudanças de um branch com outro, normalmente unindo uma funcionalidade nova ao código principal (`main` ou `master`).

🔷 **Fazendo um merge:**

```
git checkout main
git merge nova-funcionalidade
```

📌 Isso **traz as alterações** do branch `nova-funcionalidade` para `main`.

5️⃣ Clone

O **clone** cria uma cópia exata de um repositório remoto no seu computador.

🔷 **Exemplo:**

```
git clone https://github.com/usuario/repo.git
```

Isto copia todo o repositório do GitHub para a nossa máquina.

6️⃣ Push

O **push** envia commits do repositório local para um repositório remoto (GitHub).

```
git push origin main
```

📌 `origin` representa o repositório remoto, e `main` é o branch para onde estamos a enviar as mudanças.

7️⃣ Pull

O **pull** copia as mudanças do repositório remoto e aplica-as no código local.

🔷 **Atualizar o código local:**

```
git pull origin main
```

📌 Isto garante que temos a versão mais recente do código antes de fazermos novas mudanças.

8️⃣ Stash

O **stash** guarda temporariamente mudanças não commitadas, permitindo que troquemos de branch sem perder o que já se fez.

🔷 **Guardar alterações:**

```
git stash
```

🔷 **Recuperar alterações guardadas:**

```
git stash pop
```

📌 Fluxo de trabalho básico no Git

1️⃣ **Criar um repositório** ou **clonar** um existente.

2️⃣ **Criar e modificar arquivos.**

3️⃣ **Adicionar as mudanças para commit: Ao executar git add, as mudanças vão para a staging area (local onde ficam os arquivos que serão incluidos no próximo commit).**

4️⃣ **Fazer o commit: saída da staging area para o repositório local.**

5️⃣ **Fazer um push: Enviar para o repositório remoto.**

6️⃣ **Antes de fazer um push, é necessário fazer um pull das mudanças do repositório remoto.**

📌 Conclusão

- **Git** é um sistema de controlo de versão para **gerir código e trabalhar em equipa.**
- **Principais comandos:**

- `git init` → Inicia um repositório.
- `git add .` → Adiciona mudanças para commit.
- `git commit -m "mensagem"` → Guarda uma versão do código.
- `git branch nome` → Cria um novo branch.
- `git merge nome` → Junta branches.
- `git push` / `git pull` → Envia/recebe mudanças do repositório remoto.
- `git clone` → Clona um repositório remoto na nossa máquina.

Como visualizar o histórico de commits?

✅ **Resposta esperada:**

```
git log
```

📌 **Perguntas relacionadas:**

- Como ver as diferenças entre dois commits? (`git diff`)

Como sincronizar o código com o repositório remoto?

✅ **Resposta esperada:**

```
git pull origin main  # Baixa as últimas mudanças
git push origin main  # Envia novas mudanças
```
