Processo de Versionamento de Projeto

Plataforma: GitHub

Sistemas operacionais: Windows e Linux

Conceitos necessários

Commits

Um commit, ou o ato de “commitar”, se trata de agrupar alterações realizadas em seu código sob um contexto.

Quando realizamos um commit, cujo comando você verá mais a frente, o Git armazena um objeto de commit que possui um ponteiro para o snapshot da versão atual do código. Além disso, este mesmo objeto de commit armazena outro ponteiro que aponta para o commit anterior a ele, possibilitando ao Git saber a ordem em que as alterações aconteceram.

Portanto, você consegue acessar qualquer versão do código simplesmente indicando para o Git o identificador do commit desejado. Chamamos este identificador de hash.

Branches

Podemos considerar as branches como diretórios que estão sempre apontando para o último commit de que possuem conhecimento.

Por exemplo:

Digamos que estamos em nossa branch master, a branch criada pelo git como padrão para o projeto. Ao criar uma nova branch chamada “funcionalidade_caso_de_uso” exemplo: funcionalidade_cadastro_de_receitas, para trabalhar em alguma funcionalidade nova por exemplo, criaremos uma “cópia” da branch master que tem conhecimento dos mesmos commits que ela.

No entanto, a partir do momento em que alterarmos algum código na branch “funcionalidade_caso_de_uso" e realizarmos um commit com estas alterações, ela passa a apontar para este último commit enquanto a branch master continua apontando para o commit anterior.

A branch master neste caso, só tomará conhecimento das alterações feitas nesta outra branch caso solicitemos ao git para realizar a mescla do conteúdo da nossa nova branch com ela.

Passo a passo

1. Instalação da plataforma

- Instalação em Linux:
- Comando de instalação: $ sudo apt install git-all
- Instalação em Windows:
- Link para download: https://git-scm.com/downloads

2. Criação de repositório para o projeto

- Criação do diretório: mkdir git-nome_do_projeto. Exemplo: mkdir git-master-meal.

- Navegação para o diretório criado: cd git-nome_do_projeto. Exemplo: cd git-master-meal.
- Inicializar o GitHub: git init

3. Clonagem do repositório nos computadores dos demais integrantes:
- Clonagem do repositório a partir da url: git clone url_do_projeto. Exemplo: git clone https://github.com/deprao/master-meal.git
- Navegação para o diretório criado pelo git-clone: cd nome_do_projeto. Exemplo: master-meal
- Listagem dos arquivos do diretório em um arquivo examplo.txt:
- Comando em Linux: ls -l
- Comando em Windows: dir /s/b >titulo_do_arquivo.txt

4. Após a conclusão de uma tarefa, o programador deve criar uma nova branch em seu repositório local, fazer seu checkout e fazer commit das alterações, para que o histórico seja guardado no repositório local:
- Criação de branch: git branch nome_da_branch. Exemplo: git branch funcionalidade_cadastro_de_receitas
- Acessar branch criada: git checkout nome_da_branch. Exemplo: git checkout funcionalidade_cadastro_de_receitas
- Realizar commit das alterações: git commit -m "comentario"

5. depois que a tarefa for concluída e o commit for feito, é preciso fazer checkout para a branch principal e atualizar o repositório local com os documentos mais recentes:
- Voltar para a branch principal: git checkout nome_da_branch_principal. Exemplo: git checkout master-meal
- Incorpora alterações de um repositório remoto no branch atual: git pull
- Mesclar branches: git merge nome_da_branch_principal nome_da_branch_atual. Exemplo: git merge master-meal funcionalidade_cadastro_de_receitas Transferência de commits do repositório local a um remoto: git push

6. Apagar a branch criada (se desejar) e passar para a próxima tarefa da lista;
- Deletar branch: git branch -d nome_da_branch. Exemplo: git branch -d funcionalidade_cadastro_de_receitas

7. E assim sucessivamente até que o projeto seja finalizado.

Gerenciamento de Configuração do git

Repositório de diagramas

Separando em duas pastas: Análise e projetos. Dentro de cada pasta haverá diagramas gerais e outras pastas relativas aos casos de uso com os diagrama específicos.

Exemplo:

- **Análise(diretório)**
  - Diagrama caso de uso (arquivo)
  - Diagrama de classe (arquivo)
  - Login (diretório)
    - Diagrama de sequência (arquivo)
    - Diagrama de sequência alternativo (arquivo)
    - Diagrama de sequência de exceção (arquivo)
  - Cadastro (diretório)

- **Projeto (diretório)**
  - Diagrama caso de uso (arquivo)
  - Diagrama Entidade-Relacionamento (arquivo)
  - Diagrama de classe (arquivo)
  - Diagrama de pacotes (arquivo)
  - Login(diretório)
    - Diagrama de sequência (arquivo)
    - Diagrama de sequência alternativo (arquivo)
    - Diagrama de sequência de exceção (arquivo)
  - Cadastro(diretório)

Organizações de Commits

Haverá 3 tipos diferentes de commits.

- ‘fix’ – Utilizado para indicar que o commit soluciona algum problema.
- ‘feat’ – Utilizado para indicar que o commit adiciona um novo recurso.
- ‘docs’ – Utilizado para indicar que haverá adicionar/alterar documentos que não envolverem diretamente o código.

Exemplo: git commit – m “fix” Erro ao efetuar login”

Diagrama do processo de versionamento