Git e Github: Controle e compartilhe seu código

- Esse curso capacita a:
  - O que é Git e Github;
  - Entender um sistema de controle de versão;
  - Salvar e recuperar código em diferentes versões;
  - Resolver merges e conflitos;
  - Trabalhar com diferentes branches;

Aulas:

1. **O que é Git?**
   - Um sistema de controle de versões ajuda a manter o histórico de alterações, a ter controle sobre cada alteração no código, ajuda para que uma alteração alheia na influencie na alteração realizada por outra;
   - **git init // git status;**

2. **Iniciando os trabalhos:**
   - **git rm –cached <file>** [Serve para destrackear um arquivo];
   - **git restore <file>** [Após alguma alteração em algo commitado mas sem ainda estar trackeado, possível retorna antes da alteração];
   - **git commit –m "Message";**
   - **git log –p** [Exibe a alteração nos arquivos dos commits];
   - Devemos gerar um commit sempre que a nossa base de código está em um estado do qual gostaríamos de nos lembrar. Nunca devemos ter commits de códigos que não funcionam, e também não é interessante deixar para commitar apenas no final de uma feature;

3. **Compartilhando o trabalho:**
   - **git init –bare** [Cria um repositório remoto só q local];
   - **git clone {caminho} <nomeDaPastaDestino>;**
   - **git pull** [Atualiza o repositório, fazer isso depois de ter feito um git clone];

4. **Trabalhando em equipe:**
   - **git branch <nome>** [Cria novo branch];
   - **git checkout –b <nome>** [Cria e muda de branch];
   - **git merge <branch>** [Faz merge com o branch atual com o digitado];
   - **git log –graph** [Visualiza os branch no projeto];
   - Merge junta os trabalhos e gera um "merge commit". Rebase aplica os commits de outra branch na branch atual;

5. **Manipulando as versões:**
    - **git revert <hashDoCommit>** [Gera um commit, mas retorna para antes do commit];
    - **git stash // stash list** [Armazena temporariamente algumas de nossas alterações];
    - **git stash apply <numero> // stash drop <numero>;**
    - **git stash pop** [Retira a última coisa salvada lá e injeta no código];
    - **git checkout <hash>** [Viaja a HEAD no tempo para um commit específico, para as alterações serem realmente salvas a partir desse ponto, necessário criar um novo branche aí sim commitar algo novo];
        - Que, para desfazer uma alteração antes de adicioná-la para commit (com git add), podemos utilizar o comando **git checkout -- <arquivos>**;
        - Que, para desfazer uma alteração após adiciona-la para commit, antes precisamos executar o **git reset HEAD <arquivos>** e depois podemos desfazê-las com **git checkout -- <arquivos>;**
        - Que, para revertermos as alterações realizadas em um commit, o comando **git revert** pode ser a solução. Esse comando gera um novo commit informando que alterações foram desfeitas;

6. **Gerando entregas:**
    - **git diff <hash> .. <hash>** [Visualizar todas alterações entre os hashs informados];
    - **git diff <branch1> .. <branch2>;**
    - **git tag –a <nomeDaVersão (v0.1.0) –m "Mensagem";**
    - **git push tag <nomeDaTag>;**
    - **git commit –amend –m "Mensagem"** [Muda a mensagem do ultimo commit];
    - **git revert HEAD~1** [Apagar o ultimo commit];