



Versionamento de código

R esumo


Em termos simples, o versionamento de código é um processo que permite controlar e gerenciar mudanças em um projeto, mantendo um histórico completo de todas as alterações no código-fonte. Antes do versionamento de código, os desenvolvedores salvavam diferentes versões de um projeto em diferentes pastas, o que podia resultar em confusão e perda de controle.

As vantagens do versionamento de código incluem:

- Rastreabilidade: você pode manter um histórico de todas as mudanças que fez.
- Organização: facilita a manutenção do seu código.
- Colaboração: permite o trabalho em conjunto com outras pessoas.
- Segurança: reduz o risco de perda de mudanças importantes.

Existem várias ferramentas de versionamento de código disponíveis. Três das mais famosas são Subversion (mantida pela Apache Foundation), Mercurial e Git (desenvolvido por Linus Torvalds, criador do kernel do Linux).

O Git é atualmente o sistema de controle de versões mais amplamente utilizado. Foi criado por Linus Torvalds com a intenção de ser rápido, simples e com forte suporte para desenvolvimento não linear. Você pode aprender mais sobre a história do Git em seu site oficial: git-scm.com.

Existem várias plataformas baseadas em Git, incluindo GitHub, GitLab e BitBucket. Neste curso, nos concentraremos no uso do GitHub, que atualmente é mantido  pela Microsoft.

Abaixo listo o passo a passo para instalação do Git e como usá-lo para versionamento de código.

Passo 1: Instalação do Git

1.1: Primeiramente, você precisa baixar o Git para o seu computador. Para fazer isso, acesse o site oficial do projeto. É recomendado baixar diretamente do site oficial para evitar problemas com versões não oficiais ou desatualizadas.

1.2: No site do Git, você será automaticamente direcionado para a versão de download correspondente ao seu sistema operacional. Por exemplo, se estiver usando o Windows, o site sugerirá o download do Git para Windows.

1.3: Clique em “Download” e escolha a versão do sistema operacional com base em 32-bits ou 64-bits.


1.4: Após o download, clique no arquivo .exe para iniciar a instalação.

Passo 2: Configuração do Git

2.1: Durante a instalação, você será guiado através de uma série de opções de configuração. Por padrão, você pode simplesmente aceitar as configurações recomendadas.

2.2: Certifique-se de escolher um editor de texto que você está confortável para usar durante o processo de commit. O Vim é a escolha padrão, mas você pode selecionar outro se preferir.

2.3: Durante a instalação, você terá a opção de alterar a terminologia usada pelo Git. Em vez de “master”, você pode optar por usar “main”.

2.4: Após completar a instalação, você pode verificar se tudo correu bem abrindo o terminal do Visual Studio Code (VS Code) e digitando “git version”. Se o Git está  instalado corretamente, ele mostrará a versão que você acabou de instalar.

Passo 3: Configuração do usuário Git

3.1: Para trabalhar com projetos Git, é importante configurar o seu nome de usuário e e-mail. Isso ajuda a identificar quem fez modificações específicas em um projeto.

3.2: No terminal do VS Code, digite os seguintes comandos:

- Para configurar o nome de usuário, digite: `git config --global user.name “seu nome”`
- Para configurar o e-mail, digite: `git config --global user.email “seu e-mail”`

Nota: Substitua “seu nome” e “seu e-mail” pelos seus dados reais.


Com isso, você concluiu a instalação e configuração inicial do Git no seu computador. A partir daqui você está pronto para começar a trabalhar com o controle de versão de código! Lembre-se de que todas as configurações que você definiu são globais, ou seja, se aplicam a todos os projetos neste computador. No entanto, essas configurações podem ser sobrescritas em projetos específicos, se necessário.

Agora para abrir o projeto, adicionar arquivos, verificar status, fazer alterações e rastreá-las, bem como verificar o histórico no Git, siga os passos a seguir.

Passo 4: Abertura do Projeto no VS Code

Abra o Visual Studio Code. Clique em “File” (Arquivo), depois “Open Folder” (Abrir Pasta). Navegue até a pasta do seu projeto e selecione-a. O projeto agora está aberto no VS Code.

Passo 5: Iniciar o Git no Projeto

Abra o terminal do VS Code (Menu “View” > “Terminal” ou use o atalho Ctrl + `). Digite `git init`. Isso inicia o versionamento de código do Git para o seu projeto.  executar o comando `git init`, uma pasta oculta chamada `.git` é criada na pasta do projeto. Essa pasta armazena todo o histórico de versões do seu projeto.

4.3: Adicionar Arquivos ao Git

No terminal do VS Code, digite `git add .` Esse comando adiciona todos os arquivos e pastas do projeto ao Git para serem monitorados. O comando `git add .` não é o mais apropriado para todos os casos. Por exemplo, diretórios como o `vendor`, que normalmente contêm bibliotecas, não são versionados. Mas, para fins didáticos, usaremos esse comando para adicionar todos os arquivos neste momento.

4.4: Verificar o Status do Git: No terminal do VS Code, digite `git status`. Este comando exibe o status atual do projeto no Git. Seus arquivos agora estão listados como “new file”, pois foram adicionados ao Git.

4.5: Fazer o Primeiro Commit

No terminal do VS Code, digite `git commit -m “commit inicial”`. O comando `git commit` persiste todas as alterações rastreadas. O parâmetro `-m` é seguido pela mensagem do commit entre aspas, que descreve as mudanças feitas. Você agora criou a primeira versão do seu projeto.

4.6: Fazer Alterações no Projeto e Rastreá-las

Faça algumas alterações em seus arquivos. Por exemplo, você pode mudar o texto de algumas seções. Após fazer as alterações, digite `git status` novamente para ver o status atual. Seus arquivos modificados agora estão listados como “modificados”. Você pode usar `git diff nome_do_arquivo` para ver exatamente o que foi modificado em cada arquivo. Quando estiver satisfeito com as alterações, digite `git add nome_do_arquivo` para rastrear as alterações. Digite `git commit -m “descrição das alterações”` para persistir as alterações.

4.7: Verificar o Histórico do Git



No terminal do VS Code, digite `git log`. Este comando exibe o histórico de todos os commits feitos no projeto.

Lembre-se, os comandos mais importantes para iniciar o rastreamento do seu projeto são `git init`, `git add`, `git commit` e `git status`. Utilize `git log` para revisar o histórico do projeto.


Conteúdo Bônus

Se você quiser aprender mais sobre Git e GitHub. Primeiro, procure pela documentação oficial de ambos. Procure no seu navegador favorito por “Git documentation” e por GitHub Documentation”.

Ainda, existem muitos tutoriais online disponíveis gratuitamente que abrangem desde conceitos básicos até tópicos avançados de Git e GitHub. Alguns sites populares com tutoriais sobre Git e GitHub incluem o GitHub Learning Lab e o Git Tower’s Git Tutorial. Há, também, vários livros disponíveis que abordam o Git e o GitHub em detalhes. Alguns exemplos populares incluem “Pro Git” de Scott Chacon e Ben Straub e “GitHub Essentials” de Achilleas Pipinellis.

Lembrando que a prática é fundamental para o aprendizado do Git e do GitHub. Portanto, além de estudar a teoria, é recomendado que você utilize essas ferramentas em projetos reais para ganhar experiência e familiaridade com seus recursos e funcionalidades.

Referências Bibliográficas

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação**: a construção de algoritmos e estruturas de dados c  aplicações em Python. 4. ed. São Paulo: Grupo A, 2022.

ARAÚJO, Sandro de. **Linguagem de programação** (ADS). 1. ed. São Paulo: Contentus, 2020.

Atividade Prática 13 - Versionamento de código

Título da Prática: Prática de Versionamento de Código com Git e GitHub


Objetivos: O aluno deverá demonstrar compreensão e habilidade prática em utilizar o Git para versionamento de código, seguindo um fluxo de trabalho que envolve criação de repositório local, rastreamento e commit de mudanças, e finalmente, a sincronização com um repositório remoto no GitHub.

Materiais, Métodos e Ferramentas:

- Computador com acesso à internet
- Git instalado e configurado (conforme explicado na aula)
- Conta no GitHub
- Editor de código Visual Studio Code (ou outro de sua preferência)

Atividade Prática

Enunciado da atividade:

1. Comece criando uma nova pasta em seu computador para o projeto. Crie alguns arquivos HTML, CSS e Bootstrap básicos dentro dela. 
2. Inicie o Git no projeto (Passo 5 do texto-base).
3. Adicione todos os arquivos ao Git (Passo 4.3 do texto-base).
4. Realize o seu primeiro commit (Passo 4.5 do texto-base). A mensagem do commit deve ser “commit inicial”.
5. Faça algumas modificações nos arquivos do projeto. Por exemplo, você pode adicionar um novo elemento HTML ou alterar algum estilo CSS.
6. Adicione as alterações ao Git e faça um novo commit com uma mensagem descritiva (Passo 4.6 do texto-base).
7. Verifique o histórico de commits para garantir que suas alterações foram devidamente rastreadas (Passo 4.7 do texto-base).
8. Crie um novo repositório no GitHub. Não adicione um README, .gitignore ou uma licença neste momento.
9. Conecte seu repositório local ao repositório remoto que você acabou de criar no GitHub. Para fazer isso, use o comando `git remote add origin URL_do_seu_repositório`.
10. Finalmente, faça um push das alterações para o GitHub com o comando `git push -u origin master`.
11. Verifique se as alterações estão refletidas no repositório do GitHub visitando a página do seu repositório no navegador.

Resolução comentada:

1. A pasta foi criada e os arquivos básicos de um projeto web foram adicionados.



2. O comando `git init` foi usado para inicializar o Git no projeto.

3. O comando `git add .` foi usado para adicionar todos os arquivos ao Git.

4. O comando `git commit -m "commit inicial"` foi usado para realizar o primeiro commit.

5. Algumas alterações foram feitas nos arquivos. Por exemplo, um novo elemento de título foi adicionado ao arquivo HTML.

6. O comando `git add .` foi usado novamente para adicionar as novas alterações ao Git. O comando `git commit -m "adicionado título ao HTML"` foi usado para fazer um commit das novas alterações.

7. O comando `git log` foi usado para verificar o histórico de commits. Os dois commits estão presentes no histórico.

8. Um novo repositório foi criado no GitHub sem nenhum arquivo extra.

9. O comando `git remote add origin URL_do_repositório` foi usado para conectar o repositório local ao remoto.

10. O comando `git push -u origin master` foi usado para enviar as alterações para o GitHub.

11. Visitando a página do repositório no GitHub, é possível ver que os arquivos e os commits foram sincronizados corretamente.

Ir para exercício