Conteúdo do dia!

Site:Geração TechImpresso por:JOÃO VITOR DE MELO FREITASCurso:Formação em Desenvolvedor Web - OnlineData:quinta-feira, 18 jul. 2024, 22:37

Livro: Conteúdo do dia!

Índice

1. Introdução ao Git e GitHub

- 1.1. Introdução ao Git e GitHub
- 1.2. Integração com GitHub
- 1.3. Trabalhando com Branches no Git
- 1.4. Continuação: Trabalhando com Branches no Git
- 1.5. Unindo Branches no Git
- 1.6. Unindo Branches no GitHub com Pull Requests
- 1.7. Mantendo o Repositório Local Atualizado com Git Pull

1. Introdução ao Git e GitHub

O que é Git?

Git é um sistema de controle de versão distribuído que permite que vários desenvolvedores trabalhem simultaneamente em um projeto. Ele mantém um histórico de todas as alterações feitas no código, facilitando a recuperação de versões anteriores e a colaboração em equipe.

O que é GitHub?

GitHub é uma plataforma de hospedagem de código-fonte que utiliza o Git como sistema de controle de versão. Ele oferece uma interface web para gerenciar repositórios Git, além de recursos de colaboração, como pull requests, issues e integração contínua.

Benefícios do Git:

- Histórico de Alterações: Registra todas as mudanças feitas no projeto.
- Trabalho em Equipe: Permite que vários desenvolvedores trabalhem no mesmo projeto simultaneamente.
- Branches: Facilita o desenvolvimento de novas funcionalidades sem interferir no código principal.
- Backup: Armazena cópias do código em diferentes locais, prevenindo a perda de dados.

1.1. Introdução ao Git e GitHub

O que é Git?

Git é um sistema de controle de versão distribuído que permite que vários desenvolvedores trabalhem simultaneamente em um projeto. Ele mantém um histórico de todas as alterações feitas no código, facilitando a recuperação de versões anteriores e a colaboração em equipe.

O que é GitHub?

GitHub é uma plataforma de hospedagem de código-fonte que utiliza o Git como sistema de controle de versão. Ele oferece uma interface web para gerenciar repositórios Git, além de recursos de colaboração, como pull requests, issues e integração contínua.

Benefícios do Git:

- Histórico de Alterações: Registra todas as mudanças feitas no projeto.
- Trabalho em Equipe: Permite que vários desenvolvedores trabalhem no mesmo projeto simultaneamente.
- Branches: Facilita o desenvolvimento de novas funcionalidades sem interferir no código principal.
- Backup: Armazena cópias do código em diferentes locais, prevenindo a perda de dados.

Configurando o Git

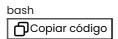
Instalação do Git

Para instalar o Git, siga os passos abaixo de acordo com o seu sistema operacional:

- Windows:
 - 1. Acesse <u>git-scm.com</u> e baixe o instalador.
 - 2. Execute o instalador e siga as instruções na tela.
 - 3. Durante a instalação, escolha o editor de texto padrão e as opções de configuração padrão.
- · Mac:
 - 1. Abra o Terminal.
 - 2. Execute o comando: brew install git.
- Linux:
 - 1. Abra o Terminal.
 - 2. Execute o comando: sudo apt-get install git (Debian/Ubuntu) ou sudo yum install git (Fedora).

Configurações Iniciais

Após instalar o Git, configure seu nome de usuário e e-mail, que serão usados nas suas commit messages:



git config --global user.name "Seu Nome" git config --global user.email "seuemail@exemplo.com"

Criando e Utilizando Repositórios Criando um Novo Repositório Local

Para criar um novo repositório local, siga os passos abaixo:

1. Navegue até a pasta onde deseja criar o repositório:



cd /caminho/para/pasta

2. Inicialize o repositório:

bash Copiar código git init

Comandos Básicos do Git

- git add .: Adiciona todas as alterações para serem commitadas.
- git commit -m "Mensagem do commit": Realiza um commit com a mensagem especificada.
- git status: Verifica o status atual do repositório.
- git log: Exibe o histórico de commits.

Conectando ao GitHub

Para conectar seu repositório local ao GitHub:

- 1. Crie um novo repositório no GitHub.
- 2. No terminal, adicione o repositório remoto:

bash Copiar código

git remote add origin https://github.com/usuario/nome-do-repositorio.git

3. Envie suas alterações para o GitHub:

bash Copiar código

git push -u origin main

Colaboração com GitHub Criando uma Conta no GitHub

- 1. Acesse <u>github.com</u> e clique em "Sign up".
- 2. Siga as instruções para criar sua conta.

Criando Repositórios no GitHub

- 1. Clique em "New repository".
- 2. Preencha o nome do repositório e a descrição.
- 3. Escolha se o repositório será público ou privado e clique em "Create repository".

Gerenciando Branches e Pull Requests

• Branches: Utilize branches para desenvolver novas funcionalidades sem interferir no código principal.

bash Copiar código

git checkout -b nova-feature

• **Pull Requests**: Ao concluir o desenvolvimento de uma funcionalidade, abra um pull request no GitHub para revisar e mesclar as alterações no branch principal.



Prepare-se para uma aula prática e interativa, onde aprenderemos a configurar, utilizar e colaborar em projetos utilizando Git e GitHub. Vamos juntos dar os primeiros passos no controle de versão e no desenvolvimento colaborativo!

1.2. Integração com GitHub

Introdução

Continuando nosso assunto sobre Git, vamos explorar a integração com o GitHub e como utilizar essas ferramentas para gerenciar projetos de forma eficiente. Vamos abordar várias situações comuns e como lidar com elas utilizando Git e GitHub.

Integração com GitHub

Situações Comuns

- 1. Início de um Projeto: Você sabe que vai utilizar GitHub desde o início.
- 2. Projeto em Andamento: Você já começou um projeto e decide guardá-lo no GitHub.
- 3. Colaboração: Você precisa que outras pessoas interajam com o projeto.

Passos Iniciais

- 1. Criando um Repositório no GitHub:
 - o Acesse sua conta no GitHub e vá para a seção de repositórios.
 - o Clique em "New" para criar um novo repositório.
 - o Dê um nome ao repositório, como "git_aula1", evite espaços e use letras minúsculas.
 - o Escolha entre repositório público ou privado.
 - o Clique em "Create repository".
- 2. Configurando o Repositório Local:
 - o Crie uma pasta no seu computador para o projeto.
 - Abra a pasta e inicie o terminal (Git Bash).
 - Navegue até a pasta com cd /caminho/para/pasta.
- 3. Clonando o Repositório:
 - o Copie a URL do repositório do GitHub.
 - No terminal, execute o comando git clone [URL].

Estrutura do Repositório

- Local: Seu computador onde os arquivos estão armazenados.
- Remoto: O repositório no GitHub.

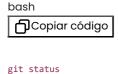
Gerenciamento de Arquivos e Pastas

• Iniciando um Repositório Local:



git init

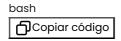
• Verificando o Status do Repositório:



• Adicionando Arquivos ao Staging:



• Comitando Alterações:



git commit -m "Mensagem do commit"

Sincronizando com o GitHub

1. Configurar Nome e E-mail:

bash Copiar código

git config --global user.name "Seu Nome" git config --global user.email "seuemail@exemplo.com"

2. Adicionar Repositório Remoto:

bash Copiar código

git remote add origin [URL]

3. Enviando Alterações para o GitHub:

bash Copiar código

git push -u origin main

Prática

Criando e Editando Arquivos

- 1. Criar um Arquivo HTML:
 - No terminal, abra o VSCode:

bash Copiar código

code .

o Crie um arquivo index.html e adicione o conteúdo:

html
Copiar código

<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width,
initial-scale=1.0"> <title>Document</title> </head> <body> Olá Mundo </body> </html>

Comitando e Enviando Alterações

1. Adicionar Arquivo ao Staging:

bash Copiar código

git add index.html

2. Commitar Alterações:

bash Copiar código

git commit -m "Adicionar arquivo index.html"

3. Enviar para o GitHub:



git push

Resolvendo Problemas Comuns

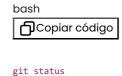
- Configuração de Usuário: Se você não configurou seu nome e e-mail, o Git pedirá essas informações ao tentar fazer um commit.
- Autenticação no GitHub: Caso não esteja logado no seu navegador padrão, o Git solicitará que você faça login.

Finalização

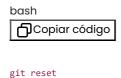
- Verificar no GitHub: Após enviar os arquivos, verifique no GitHub se as alterações foram aplicadas corretamente.
- Configuração Permanente: Configure seu nome e e-mail uma vez para evitar problemas futuros.

Comandos Úteis

• Verificar Status:



• Resetar Staging:



• Adicionar Todos os Arquivos:





Lembre-se de sempre ler as mensagens de erro e sugestões que o Git fornece, pois elas podem ajudar a resolver problemas e configurar corretamente seu ambiente de desenvolvimento. Nos vemos na próxima aula, onde continuaremos explorando mais funcionalidades e boas práticas com Git e GitHub!

1.3. Trabalhando com Branches no Git

Criando e Trabalhando com Branches

Vamos imaginar que temos nosso projeto git_aula1 com um arquivo index.html. O código está funcionando, mas o gerente de projeto solicitou uma alteração específica. Em vez de fazer a alteração diretamente no projeto principal, vamos criar uma nova versão (branch) do projeto para trabalhar nessa alteração. Isso nos permite manter a versão original intacta e experimentar sem comprometer a estabilidade do código principal.

Passos para Criar e Trabalhar com uma Nova Branch

1. Verificando o Status Atual

Certifique-se de que seu repositório local está atualizado e que você está na branch principal (main ou master):

bash
Copiar código
git status

2. Criando uma Nova Branch

Vamos criar uma nova branch chamada alternativa_a:



• checkout -b: Cria uma nova branch e muda para ela.

3. Verificando a Branch Atual

Confirme que você está na nova branch:



Você verá uma lista de branches e a alternativa_a estará destacada.

4. Fazendo Alterações na Nova Branch

Agora, vamos fazer as alterações solicitadas no arquivo index.html:

- Abra o arquivo index.html no seu editor de código.
- Faça as alterações necessárias. Por exemplo, mude o conteúdo para:

```
Copiar código
```

<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Document</title> </head> <body> Alteração na branch alternativa_a </body> </html>

5. Adicionando e Comitando as Alterações

Adicione as alterações ao staging e faça um commit:



git add index.html git commit -m "Alteração na branch alternativa_a"

6. Enviando a Nova Branch para o GitHub

Para enviar a nova branch ao repositório remoto no GitHub, use o comando push:



• -u origin alternativa_a: Configura a branch remota para fazer o tracking da branch local.

Benefícios de Utilizar Branches

- Isolamento: As alterações feitas em uma branch não afetam outras branches.
- Paralelismo: Permite que múltiplas funcionalidades sejam desenvolvidas simultaneamente.
- Segurança: Preserva a estabilidade do código principal enquanto experimenta novas funcionalidades.

Trocando de Branch e Mesclando Alterações

1. Mudando de Branch

Para voltar à branch principal (main):



git checkout main

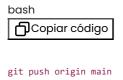
2. Mesclando Alterações

Depois de concluir e testar as alterações na alternativa_a, você pode mesclá-las na branch principal:



3. Enviando a Branch Principal Atualizada

Finalmente, envie as alterações da branch principal para o GitHub:





Conclusão

Trabalhar com branches permite gerenciar versões do seu projeto de forma eficaz, mantendo o código principal estável enquanto implementa novas funcionalidades ou experimenta mudanças. Nos vemos na próxima aula, onde continuaremos explorando mais funcionalidades do Git e GitHub!

1.4. Continuação: Trabalhando com Branches no Git

Criando e Sincronizando Novas Branches

Olá pessoal, tudo bem com vocês? Vamos continuar nosso tema de Git. Hoje quero mostrar para vocês como trabalhar com múltiplas branches no nosso repositório. Vamos aprender a criar uma nova branch a partir de outra branch já existente e sincronizar branches entre o repositório local e o remoto.

Criando uma Nova Branch no GitHub

Passos para Criar uma Nova Branch

- 1. Verificando as Branches Existentes:
 - o No GitHub, acesse o seu repositório.
 - o Clique em "Branches" para ver as branches existentes. No nosso caso, temos main e alternativa_a.
- 2. Criando a Branch Alternativa B:
 - o Clique em "New branch".
 - Nomeie a nova branch como alternativa_b.
 - o Selecione a branch base. Vamos criar a partir da alternativa_a.
 - o Clique em "Create branch".

Editando a Nova Branch

- 1. Editando o Arquivo na Branch Alternativa B:
 - Acesse a branch alternativa_b.
 - o Clique no arquivo index.html e selecione "Edit this file".
 - o Modifique o conteúdo para:



<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width,
initial-scale=1.0"> <title>Document</title> </head> <body> Alterado para a versão B </body> </html>

- Adicione uma mensagem de commit:
 - git commit -m "Nova versão atualizada"
- o Clique em "Commit changes".

Sincronizando Branches Entre Local e Remoto

Agora que criamos e modificamos a branch alternativa_b no GitHub, precisamos trazê-la para nosso repositório local.

Passos para Sincronizar Branches

- 1. Verificando as Branches Locais:
 - o No terminal do VSCode, verifique as branches locais:

Copiar código

git branch

• Você verá que temos main e alternativa_a.

2. Trazendo as Branches Remotas:

o Para sincronizar todas as branches remotas com o repositório local, use:

Copiar código

git fetch

o Este comando atualiza as referências locais das branches remotas.

3. Mudando para a Nova Branch:

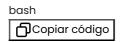
• Agora, vamos mudar para a nova branch alternativa_b que trouxemos do remoto:



git checkout alternativa_b

4. Verificando a Mudança:

• Após mudar para a branch alternativa_b, verifique o conteúdo do arquivo index.html:



<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width,
initial-scale=1.0"> <title>Document</title> </head> <body> Alterado para a versão B </body> </html>



Conclusão

Agora temos três branches sincronizadas entre o repositório local e o GitHub: main, alternativa_a e alternativa_b. Aprendemos como criar branches no GitHub, editá-las diretamente na interface do GitHub e sincronizá-las com o repositório local utilizando comandos Git. Nos vemos na próxima aula, onde continuaremos explorando mais funcionalidades e boas práticas com Git e GitHub!

Espero vocês na próxima aula!

1.5. Unindo Branches no Git

Unindo Branches no Git

Vamos continuar explorando o Git e aprender como unir branches. Imagine que você está trabalhando na branch alternativa_b, mas precisa trazer as alterações dessa branch para alternativa_a. Vamos ver como fazer isso.

Passos para Unir Branches

1. Posicione-se na Branch que Vai Receber as Alterações

Primeiro, mude para a branch alternativa_a, que receberá as alterações de alternativa_b:



2. Unindo as Branches

Com a branch alternativa_a selecionada, execute o comando git merge para unir as alterações da alternativa_b:



• Descrição do Comando: Este comando traz todas as alterações de alternativa_b para alternativa_a.

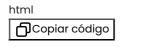
3. Verificando o Merge

Após executar o comando, o Git indicará quais alterações foram aplicadas. Se tudo correr bem, você verá uma mensagem informando que as branches foram unidas com sucesso.

Confirmando as Alterações

1. Verifique o Arquivo:

 Abra o arquivo index.html na branch alternativa_a para confirmar que as alterações de alternativa_b foram aplicadas:



<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width,
initial-scale=1.0"> <title>Document</title> </head> <body> Alterado para a versão B </body> </html>

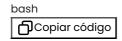
2. Commit das Alterações:

o Se necessário, adicione as alterações ao staging e faça um commit para registrar a união das branches:



3. Enviar as Alterações para o Repositório Remoto:

o Envie as alterações da branch alternativa_a para o GitHub:



git push origin alternativa_a

Conclusão

Você aprendeu como unir branches no Git usando o comando git merge. Este processo é útil quando você precisa consolidar mudanças de diferentes branches em uma única branch, mantendo seu repositório organizado e atualizado.

Resumo dos Passos:

1. Mude para a Branch de Destino:





3. Confirme e Commit as Alterações:



4. Envie para o GitHub:





Nos vemos na próxima aula, onde continuaremos a explorar mais funcionalidades do Git e GitHub. Até lá!

1.6. Unindo Branches no GitHub com Pull Requests

Unindo Branches Remotamente com Pull Requests

Olá pessoal, tudo bem com vocês? Vamos continuar nosso tema de Git e GitHub. Na última aula, aprendemos a unir branches localmente. Hoje, veremos como fazer isso diretamente no GitHub usando a ferramenta de Pull Requests.

Passos para Unir Branches Remotamente Usando Pull Requests

1. Acessando as Branches no GitHub

Primeiro, acesse o seu repositório no GitHub. Você verá uma sugestão para comparar e criar um Pull Request entre branches, se houver alterações não mescladas.

2. Criando um Pull Request

1. Comparar Branches:

- o Clique em "Compare & pull request" ou vá até a aba "Pull requests" e clique em "New pull request".
- Escolha a branch base (por exemplo, main) e a branch de comparação (por exemplo, alternativa_a).

2. Verificando Diferenças:

o Verifique as diferenças entre as branches para garantir que está mesclando corretamente.

3. Criando o Pull Request:

- o Dê um título ao Pull Request, como "Juntando a branch alternativa_a com main".
- o Adicione uma descrição detalhada do que foi feito, se necessário.
- o Clique em "Create pull request".

3. Avaliando o Pull Request

1. Verificação de Conflitos:

- o O GitHub automaticamente verifica se há conflitos entre as branches.
- Se não houver conflitos, você verá uma mensagem verde indicando que a mesclagem pode ser feita sem problemas.
- Se houver conflitos, o GitHub solicitará que você resolva os conflitos antes de continuar.

2. Aprovação do Pull Request:

 Em um ambiente colaborativo, outro desenvolvedor deve revisar e aprovar o Pull Request para garantir que as mudanças são válidas e não introduzem problemas.

4. Mesclando o Pull Request

1. Mesclagem:

- o Após a revisão, clique em "Merge pull request" e confirme a mesclagem.
- o Isso unirá as alterações da branch alternativa_a na branch main.

2. Verificação Pós-Mesclagem:

o Após a mesclagem, vá até a aba "Code" e selecione a branch main para verificar as alterações aplicadas.

Conclusão

Com a mesclagem de branches através de Pull Requests, você pode gerenciar alterações de maneira eficiente e colaborativa diretamente no GitHub. Este processo é especialmente útil quando se trabalha em equipe, pois permite revisões e aprovações antes de aplicar alterações ao branch principal.

Resumo dos Passos:

1. Comparar e Criar Pull Request:

o Comparar branches e criar um Pull Request no GitHub.

2. Verificar e Resolver Conflitos:

• Verificar automaticamente conflitos e resolvê-los, se necessário.

3. Aprovar e Mesclar:

- o Outro desenvolvedor revisa e aprova o Pull Request.
- Mesclar as alterações na branch principal (main).

4. Verificação Pós-Mesclagem:

o Confirmar que as alterações foram aplicadas corretamente.



Nos encontramos na próxima aula, onde continuaremos a explorar mais funcionalidades e boas práticas com Git e GitHub. Até lá!

1.7. Mantendo o Repositório Local Atualizado com Git Pull

Atualizando o Repositório Local com git pull

Olá pessoal, tudo bem com vocês? Vamos continuar com mais uma aula de Git. Na aula passada, fizemos o merge da branch alternativa_a com a branch main no repositório remoto. Agora, precisamos atualizar o nosso repositório local para refletir essas mudanças.

Passos para Atualizar o Repositório Local

1. Utilizando git pull

Uma forma simples de atualizar o repositório local é usar o comando git pull. Este comando "puxa" as alterações do repositório remoto para o local.

- 1. Abrir o Terminal:
 - No VSCode, abra o terminal integrado com `Ctrl + ``.
- 2. Executar git pull:
 - o Posicione-se na branch que deseja atualizar:



• Execute o comando git pull para trazer as atualizações do remoto:



• Este comando trará todas as alterações da branch main do repositório remoto para a branch main do repositório local.

2. Verificando as Alterações

Após o git pull, verifique se o arquivo index.html foi atualizado conforme esperado:



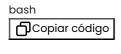
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Document</title> </head> <body> Alterado para a versão B </body> </html>

Usando git pull para Mesclar Branches Remotas Direto no Local

Situação: Trazer Alterações de alternativa_b para main

Suponha que você já saiba que todas as alterações na branch alternativa_b são corretas e deseja mesclá-las diretamente na branch main do seu repositório local.

1. Posicione-se na Branch main:



git checkout main

2. Executar git pull com Mesclagem de Branch:



git pull origin alternativa_b

• Este comando trará as alterações de alternativa_b do repositório remoto e as mesclará diretamente na branch main do repositório local.

Verificação Pós-Mesclagem

Verifique o arquivo index.html para confirmar as alterações:



<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Document</title> </head> <body> Alterado mais uma vez </bdy> </html>

Sincronizando com o Repositório Remoto

Após mesclar as alterações localmente, envie as atualizações de volta para o repositório remoto:

1. Adicionar e Comitar Alterações:



git add . git commit -m "Mesclando alterações de alternativa b na main"

2. Enviar Alterações para o Repositório Remoto:



git push origin main



Conclusão

Aprendemos a atualizar o repositório local usando git pull e a mesclar branches remotas diretamente no local. Este processo é útil para manter o repositório local sempre sincronizado com o remoto.

Resumo dos Passos:

1. Atualizar Branch Local:

bash Copiar código git pull origin main

2. Mesclar Branch Remota no Local:

bash Copiar código

git checkout main git pull origin alternativa_b

3. Enviar Atualizações para o Remoto:



 $\hbox{\tt git add . git commit -m "Mesclando alterações de alternativa_b na main" git push origin main } \\$

Nos vemos na próxima aula para explorar mais funcionalidades e boas práticas com Git e GitHub. Até lá!