

ANTES DO PUSH: EIS A QUESTÃO DO COMMIT

O BÁSICO DE GIT E GITHUB QUE TODO DEV PRECISA SABER



JOICY LARA

Antes de Começar...

Por que este eBook foi criado e como ele pode te ajudar com Git e GitHub

Este eBook é resultado de um desafio do curso "Introdução à Engenharia de Prompts", com foco em aplicar IA para criar conteúdo útil.

O tema escolhido foi Git e GitHub para iniciantes, com explicações simples, exemplos práticos e comandos essenciais para quem está começando na programação.

01

O que é Versionamento de Código?

Capítulo 1 - O que é Versionamento de Código?

Antes de entender o Git, é essencial saber o que é versionamento.

Versionamento é o processo de **controlar, registrar e organizar as alterações** feitas em um projeto ao **longo do tempo**.

Capítulo 1 - O que é Versionamento de Código?

Com ele, você pode:

- Saber quem fez cada mudança;
- Voltar no tempo para uma versão anterior;
- Comparar diferenças entre versões;
- Trabalhar em múltiplas ideias ao mesmo tempo (com branches);
- Evitar que alterações de outras pessoas apaguem o seu trabalho.

Capítulo 1 - O que é Versionamento de Código?

Sem versionamento, os riscos são altos: perder código, sobrescrever trabalho de colegas, não saber mais o que funcionava antes ou como resolver um erro que surgiu depois de alguma alteração.

Capítulo 1 - O que é Versionamento de Código?

Por isso, o versionamento é uma das habilidades mais importantes para qualquer desenvolvedor, e o Git é a ferramenta mais usada para isso no mundo.

02

Por que aprender Git e GitHub?

Capítulo 2 - Por que aprender Git e GitHub?

Imagine escrever um trabalho enorme na faculdade, salvar como “*versão_final_1*”, “*versão_final_de_verdade*”, “*versão_final_mesmo.docx*”... Parece familiar?

É **exatamente** esse **problema** que o **Git resolve**.

Capítulo 2 - Por que aprender Git e GitHub?



Git é um sistema de controle de versão. Ele permite que você salve o histórico do seu projeto, acompanhe as alterações e volte no tempo quando algo dá errado.

Capítulo 2- Por que aprender Git e GitHub?



Já o GitHub é uma plataforma online que armazena seus projetos Git. Com ele, você pode colaborar com outras pessoas, mostrar seu código para o mundo ou manter seus repositórios privados e organizados.

03

O que é Git?

Capítulo 3 - O que é Git?

Git é um sistema que registra todas as modificações feitas no seu projeto ao longo do tempo.

Ele permite que você salve versões do seu código, volte para estados anteriores, teste novas ideias sem medo e trabalhe em equipe sem bagunçar tudo.

Capítulo 3 - O que é Git?

Pense no Git como uma máquina do tempo do seu projeto. Cada vez que você faz um commit, está salvando um “checkpoint” que pode ser acessado no futuro.

Além disso, o Git funciona localmente. Ou seja, mesmo sem internet, você pode usar todos os recursos da ferramenta. Só quando for compartilhar no GitHub é que precisa estar online.

04

O que é GitHub?

Capítulo 4 - O que é GitHub?

O GitHub é um serviço de hospedagem para repositórios Git. Ele funciona como um arquivo na nuvem para o seu código, mas vai muito além disso.

Capítulo 4 - O que é GitHub?

Com o GitHub, você pode:

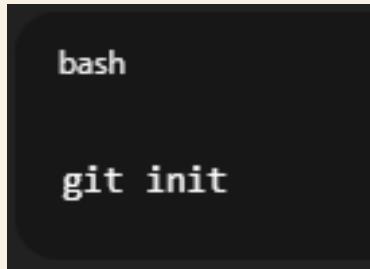
- Compartilhar seu projeto com outras pessoas;
- Colaborar em equipe com controle de quem alterou o quê;
- Criar portfólio para mostrar seu trabalho;
- Contribuir em projetos de código aberto;
- Documentar seu projeto e acompanhar issues (problemas e melhorias sugeridas).

05

Começando um projeto com Git

Capítulo 5 - Começando um projeto com Git

Antes de usar o Git, é necessário iniciar um repositório na pasta do seu projeto.



Esse comando cria uma pasta oculta chamada *.git*, onde o Git vai guardar todo o histórico de mudanças. A partir daí, tudo o que você alterar pode ser registrado.

Capítulo 5 - Começando um projeto com Git

Também é importante configurar seu nome e email:

```
bash
```

```
git config --global user.name "Seu Nome"  
git config --global user.email "seu@email.com"
```

Essas informações são usadas para identificar quem fez cada alteração.

06

O ciclo do Git

add, commit, push

Capítulo 6 - O ciclo do Git: *add, commit, push*

O ciclo básico do Git segue três passos:

1. *git add* — prepara os arquivos para serem salvos
2. *git commit* — salva o estado atual com uma mensagem
3. *git push* — envia tudo para o GitHub

Capítulo 6 - O ciclo do Git: *add, commit, push*

Pense em uma encomenda:

- *add* embala os arquivos,
- *commit* sela a caixa com uma etiqueta,
- *push* leva ao correio (GitHub).



07

git commit:
**Salvando um
momento no tempo**

Capítulo 7 - *git commit*: Salvando um momento no tempo

O commit é o coração do Git. Cada commit representa uma foto do seu projeto naquele momento.

Antes de um commit, você precisa usar git add para indicar quais arquivos vão ser incluídos.

Capítulo 7 - *git commit*: Salvando um momento no tempo

Exemplo prático:

```
bash
```

```
git add index.html  
git commit -m "Cria a estrutura inicial da página"
```

A flag *-m* permite escrever uma mensagem curta que descreve o que foi feito.



Dica: Escreva mensagens claras e específicas. Isso ajuda na leitura do histórico e facilita o trabalho em equipe.

Capítulo 7 - *git commit*: Salvando um momento no tempo

Usar padrões nas mensagens de commit ajuda a manter o projeto organizado e fácil de entender.

Um formato comum é tipo: descrição, como *feat*: adiciona login ou *fix*: corrige erro no botão. Isso facilita a leitura do histórico e melhora a colaboração, especialmente em equipe.

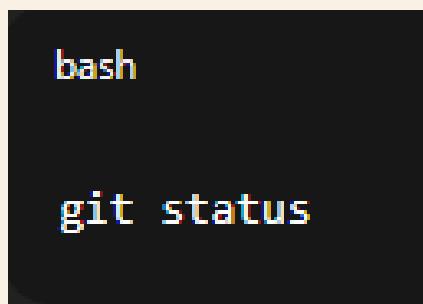
08

Inspecionando o estado do projeto

Capítulo 8 - *git status* e *git log*: Observando o estado do projeto

O comando *git status* mostra:

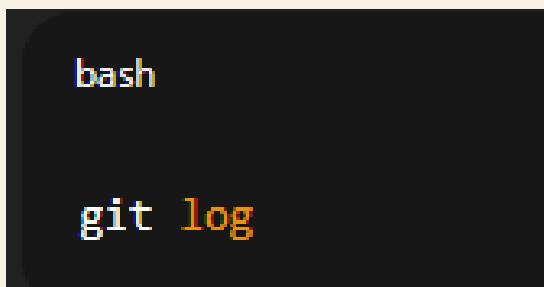
- Quais arquivos foram modificados;
- Quais estão prontos para commit;
- Quais ainda não foram adicionados.



Capítulo 8 - *git status* e *git log*: Observando o estado do projeto

Já o *git log* mostra o histórico de commits

- Inclui:
- Autor da mudança
- Data
- Mensagem do commit
- Hash (código único que identifica o commit)



09

O que são branches?

Capítulo 9 - O que são branches?

Branches (ou ramificações) são como linhas alternativas de desenvolvimento. Você pode testar novas funcionalidades sem impactar o projeto principal.

```
bash
```

```
git branch nova-funcionalidade  
git checkout nova-funcionalidade
```

Capítulo 9 - O que são branches?

Imagine que você quer testar uma nova receita, mas sem estragar o prato principal. Você cria uma nova versão (branch), experimenta à vontade e depois decide se quer unir (merge) com o original.

10

Enviando seu projeto para o GitHub

Capítulo 10 - Enviando seu projeto para o GitHub

Depois de iniciar um repositório no GitHub, você conecta seu projeto local com ele:

```
git remote add origin https://github.com/seunome/repositorio.git  
git push -u origin main
```

A partir daí, qualquer git push vai atualizar o repositório remoto.

11

Baixando e atualizando com o GitHub

Capítulo 11 - Baixando e atualizando com o GitHub

Para baixar um projeto existente, use:

```
bash  
  
git clone https://github.com/usuario/repositorio.git
```

Para atualizar seu projeto local com mudanças do GitHub, use:

```
bash  
  
git pull origin main
```

12

Resolvendo conflitos

Capítulo 12- Resolvendo conflitos

Conflitos acontecem quando duas pessoas editam a mesma linha de um arquivo em branches diferentes. O Git não sabe qual versão manter e pede sua ajuda.

Capítulo 12 - Resolvendo conflitos

Nesse caso:

- O Git mostrará o conflito no *git status*
- Você verá o conteúdo duplicado no arquivo
- Edite manualmente para corrigir
- Faça *git add* e *git commit* normalmente

13

Ignorando arquivos com .gitignore

Capítulo 13 - Ignorando arquivos com `.gitignore`

O arquivo `.gitignore` serve para excluir arquivos que você não quer enviar ao GitHub, como arquivos temporários, segredos, pastas de dependências.

Capítulo 13 - Ignorando arquivos com .gitignore

Exemplo:

```
bash

node_modules/
.env
*.log
```

14

Boas práticas com Git e GitHub

Capítulo 14 - Boas práticas com Git e GitHub

- Commits frequentes:
não espere juntar muita coisa para fazer commit.
- Mensagens claras:
facilite a leitura do histórico.
- Use branches para novas features ou correções.

Capítulo 14 - Boas práticas com Git e GitHub

- Sempre puxe (*git pull*) antes de empurrar (*git push*).
- Use *.gitignore* para manter o repositório limpo.
- Colabore via pull requests no GitHub, especialmente em times.

15

Versione sua
evolução como dev

Capítulo 15 - Versione sua evolução como dev

Aprender Git e GitHub é como aprender a escrever com clareza.

Mais do que comandos, essas ferramentas ensinam organização, versionamento, responsabilidade e colaboração.

Capítulo 15 - Versione sua evolução como dev

Comece usando Git nos seus projetos pessoais. Quando você menos perceber, já estará lidando com branches, commits e merges com naturalidade.

16

Para ir além: Links úteis sobre Git, GitHub e comandos

Capítulo 16 - Links úteis sobre Git, GitHub e comandos

1. Documentação oficial do Git (em português)
2. Guia de referência rápida do Git
3. Curso gratuito de Git e GitHub na plataforma Curso em Vídeo
4. Padrões de commit
5. Lista de comandos úteis do Git

16

Para ir além: Links úteis sobre Git, GitHub e comandos

AGRADECIMENTO

Obrigada por ler este
eBook!

Espero que ele tenha te
ajudado a entender melhor
os conceitos de Git e
GitHub, e que sirva como
base para seus próximos
commits na jornada dev.

CONTATO



GitHub:<https://github.com/Joicylara>



LinkedIn:<https://www.linkedin.com/in/joicy-kelly-dev/>