Guia Pratico de Comandos do Git com Exemplos Reais em Java (GitHub e GitLab)

O Git é um sistema de controle de versão indispensável no desenvolvimento de software, permitindo que equipes gerenciem mudanças no código de maneira eficiente.

GitHub e GitLab são ferramentas baseadas no Git que potencializam esse processo, oferecendo funcionalidades colaborativas e de automação.

Este guia explora os comandos mais usados no Git, contextualizando cada um com exemplos reais em projetos Java.

O que é GitHub?

O **GitHub**, fundado em 2008 por **Tom Preston-Werner**, **Chris Wanstrath**, **PJ Hyett** e **Scott Chacon**, é uma das maiores plataformas para hospedagem de código.

Adquirido pela Microsoft em 2018, ele é amplamente utilizado para gerenciar projetos de código aberto e privados.

Exemplo real: O **Spring Framework**, um dos frameworks mais populares para desenvolvimento Java, é gerenciado no GitHub.

Ele utiliza *pull requests*, revisões de código e tags para organizar versões.

O que é GitLab?

O **GitLab**, criado em 2011 por **Dmitriy Zaporozhets** e **Valery Sizov**, é conhecido por ser uma solução completa para DevOps, oferecendo CI/CD integrado, gerenciamento de segurança e rastreamento de problemas.

Comandos do Git com Exemplos Reais em Java

Agora que entendemos as ferramentas, vamos explorar os principais comandos do Git com cenários reais baseados em um projeto Java.

1. git init

Descrição: Inicializa um repositório Git no diretório atual

Cenário real: Você está iniciando um projeto de uma API REST em Java para gerenciar uma biblioteca.

Exemplo:

bash
mkdir biblioteca-api
cd biblioteca-api
git init

Boa prática: Garanta que o diretório inicial só contenha arquivos relevantes ao projeto.

2. git clone

Descrição: Faz uma cópia de um repositório remoto para a máquina local.

Cenário real: Sua equipe está desenvolvendo um sistema de reservas e você precisa clonar o repositório existente no GitLab.

Exemplo:

bash
git clone https://gitlab.com/empresa/sistema-reservas.git

Boa prática: Use URLs SSH sempre que possível para evitar problemas de autenticação:

bash
git clone git@gitlab.com:empresa/sistema-reservas.git

3. git status

Descrição: Mostra o status dos arquivos no repositório (modificados, novos ou não rastreados).

Cenário real: Você modificou o arquivo LivroService.java para corrigir um problema no sistema de empréstimos e deseja verificar seu status.

Exemplo:

bash
git status

- ***Saída esperada**:*

  plaintext
  Copiar código
  On branch main
  Changes not staged for commit:
  modified: src/main/java/com/empresa/LivroService.java

4. git add

Descrição: Adiciona arquivos ao *staging area* para o próximo commit.

Cenário real: Você corrigiu um bug na classe LivroService.java e quer preparar a mudança para commit.

Exemplo:

bash
git add src/main/java/com/empresa/LivroService.java

Ou para adicionar todos os arquivos alterados:

bash
git add .

Boa prática: Use git add . com cautela e sempre revise as mudanças com git status.

5. git commit

Descrição: Salva alterações no histórico do repositório.

Cenário real: Após corrigir o bug, você faz um commit com uma mensagem detalhada.

Exemplo:

bash
git commit -m "Corrige bug na lógica de renovação de empréstimos em LivroService"

Boa prática: Escreva mensagens de commit que expliquem o "quê" e, se possível, o "porquê" das alterações.

6. git push

Descrição: Envia os commits locais para o repositório remoto.

Cenário real: Você finalizou uma nova funcionalidade e precisa enviá-la para o repositório principal no GitHub.

Exemplo:

bash
git push origin main

Boa prática: Antes de fazer o *push*, garanta que sua branch está atualizada usando git pull.

7. git pull

Descrição: Atualiza o repositório local com alterações do remoto.

Cenário real: Um colega adicionou a funcionalidade de relatórios ao projeto, e você precisa integrá-la ao seu ambiente local.

Exemplo:

bash
git pull origin main

- **Boa prática**: Sempre execute este comando antes de começar a trabalhar no código para evitar conflitos.

8. git branch

Descrição: Gerencia branches no repositório.

Cenário real: Você está implementando a funcionalidade de cálculo de multas por atraso.

Exemplo:
Criar uma branch para a funcionalidade:

bash
git branch feature/calculo-multas

Trocar para a nova branch:

bash
git checkout feature/calculo-multas

Boa prática: Use convenções de nomenclatura como feature/ ou bugfix/ para identificar o objetivo da branch.

9. git merge

Descrição: Mescla alterações de uma branch em outra.

Cenário real: Após concluir o cálculo de multas, você quer integrá-lo à branch principal.

Exemplo:

bash
git checkout main
git merge feature/calculo-multas

Boa prática: Antes de mesclar, revise o código com *Pull Requests* ou *Merge Requests*.

10. git stash

Descrição: Salva mudanças temporárias sem fazer commit.

Cenário real: Você está desenvolvendo um novo endpoint para a API, mas precisa interromper para corrigir um bug crítico.

Exemplo:

bash
git stash
 # Resolve o bug
git stash apply

***Boa prática**:* Nomeie os *stashes* para facilitar a organização:

bash
git stash save "WIP: endpoint para renovação de empréstimos"

Exemplo Prático Completo: Desenvolvimento de uma API em Java

1. **Inicialização do projeto**:

   bash
   git init

   Criação de uma API Java básica com o Spring Boot.

2. **Criação de uma branch para nova funcionalidade**:

   bash
   git branch feature/cadastro-usuarios
   git checkout feature/cadastro-usuarios

3. **Adição de mudanças**: Após implementar o UsuarioController.java:

   bash
   git add src/main/java/com/empresa/UsuarioController.java
   git commit -m "Adiciona endpoint para cadastro de usuários"

4. **Mesclagem após testes**:

   bash
   git checkout main
   git merge feature/cadastro-usuarios
   git push origin main

Boas Práticas no Uso do Git em Projetos Java

Escreva mensagens de commit detalhadas: Use mensagens como *"Adiciona validação de dados no endpoint de login"* para facilitar a rastreabilidade.

Utilize branches específicas: Por exemplo, feature/autenticacao para novas funcionalidades ou bugfix/corrige-login para correções.

Use arquivos .gitignore: Exclua arquivos irrelevantes, como target/ (compilados do Maven) e application.properties (configurações locais). **Exemplo de .gitignore para Java**:

kotlin
**.class*
target/
.idea/
**.iml*
application.properties

Git, GitHub e GitLab são indispensáveis para projetos Java modernos.

Comandos como git add, git commit e git push, combinados com boas práticas como uso de branches e mensagens claras, garantem um desenvolvimento colaborativo eficiente, seja implementando novas funcionalidades ou corrigindo bugs, esses conceitos são fundamentais para gerenciar projetos com sucesso.

EducaCiência FastCode para a comunidade