SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

# CCF 322 - Engenharia de Software II

# PADRÕES DE COMMIT

## Equipe 5º Ano - “BYTECRAFT”

Florestal - MG
2025

# Conceitos e Boas Práticas de Commit

Este guia tem como objetivo apresentar os principais comandos de **Git** e **GitHub** que serão utilizados ao longo do desenvolvimento do projeto.
Os comandos descritos podem ser executados diretamente no terminal do seu sistema operacional, desde que o **Git** esteja instalado. Caso ainda não tenha o Git instalado, acesse o link "Git-Download" para realizar a instalação.
Se você ainda não tem familiaridade com o terminal, ou prefere uma interface gráfica, recomendamos o uso do **GitHub Desktop** ou do próprio site do **GitHub** como alternativas mais acessíveis.

## 1. O Básico do Git e GitHub

O **Git** é um sistema de controle de versão local, enquanto o **GitHub** é um serviço online para hospedar repositórios Git e facilitar colaboração.

## 2. Fluxo básico para trabalhar com Git/GitHub

2.1 Clonar um repositório existente

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

2.2 Criar uma nova branch, para evitar trabalhar na main diretamente, seguindo o fluxo de trabalho descrito na documentação do projeto (GitFlow).

```shell
git checkout -b nome-da-branch
```

2.3 Verificar o status dos arquivos

```shell
git status
```

2.4 Adicionar arquivos ao controle

```shell
git add nome-do-arquivo
# ou todos os arquivos modificados
git add .
```

2.5 Criar um commit

```shell
git commit -m "mensagem clara do que foi feito"
```

2.6 Enviar para o GitHub

```shell
git push origin nome-da-branch
```

2.7 Criar um Pull Request **no GitHub** para revisão e integração, selecionando corretamente a branch de origem e de destino conforme descrito no fluxo de trabalho (GitFlow).

## 3. Palavras-chave especiais nos commits

As *issues* no GitHub são um recurso para registrar tarefas, bugs, sugestões ou discussões sobre um projeto.

Cada issue recebe um número único (#n) e pode conter:

- Título e descrição do problema ou sugestão
- Comentários de colaboradores
- Etiquetas (labels) para categorizar (ex.: bug, melhoria, documentação)
- Atribuição de responsáveis
- Relação com commits e pull requests

Quando usamos palavras-chave como fixes #n ou closes #n em commits ou pull requests, o GitHub vincula automaticamente a mudança à issue e a fecha quando o pull request é mesclado.

O GitHub reconhece palavras-chave que, quando usadas junto ao número de uma *issue*, executam ações automáticas.

**Principais palavras-chave**

| **Palavra-chave** | **Efeito** |
|---|---|
| fixes #n ou fixed #n | Fecha a *issue* ao mesclar o commit/pull requests |
| closes #n | Fecha a *issue* ao mesclar |
| resolves #n | Fecha a *issue* ao mesclar |
| refs #n ou references #n | Apenas referencia a *issue*, sem fechar |
| see #n | Aponta para a *issue*, sem fechar |

**Exemplo de commit usando palavra-chave:**

```shell
git commit -m "fix: corrige bug no cadastro de usuários. fixes #23"
```

Quando o pull request que contém esse commit for aceito na branch principal, a *issue* nº 23 será fechada automaticamente.

---

## 4. Padrões de escrita para commits

Uma mensagem de commit **clara** ajuda qualquer pessoa (inclusive você no futuro) a entender o que foi feito e por quê.

**Estrutura recomendada**

```shell
<tipo>: <descrição curta no imperativo>

<descrição detalhada opcional>
```

**Tipos comuns (inspirados no Conventional Commits)**

| Tipo | Uso |
|---|---|
| feat | Nova funcionalidade |
| fix | Correção de bug |
| docs | Alteração apenas em documentação |
| style | Mudanças de formatação/estilo sem alterar lógica |
| refactor | Refatoração de código sem alterar comportamento |
| test | Adição ou modificação de testes |
| chore | Tarefas de manutenção ou configuração |

**Exemplo de commit com descrição complementar:**

```shell
git commit -m "fix: corrige bug no cadastro de usuários" \
```

```
-m "Adiciona validação de cpf único para evitar duplicidade.
fixes #23"
```