

GUIA DEFINITIVO DE GIT E GITHUB

Do zero ao uso correto (sem cometer erros comuns)

Autor: Marcos André (*uso pessoal/estudos*)

1. CONCEITOS FUNDAMENTAIS (ENTENDA ANTES DE USAR)

O que é Git?

Git é um **sistema de controle de versão**.

Ele guarda o histórico do seu projeto, permitindo:

- salvar versões
 - voltar no tempo
 - trabalhar em equipe
 - enviar código para o GitHub
-

O que é GitHub?

GitHub é um **repositório remoto** (na nuvem).

Ele **não substitui o Git**, ele **recebe o Git**.

Git = local (seu PC)

GitHub = remoto (internet)

Conceitos IMPORTANTES

Termo Significado

repositório projeto versionado

commit “foto” do projeto

branch linha de desenvolvimento

main branch principal

origin nome padrão do remoto

upstream ligação entre branch local e remota

2. ESTRUTURA CORRETA DE PROJETO

Lugar CERTO para o Git

Sempre na **raiz do projeto**:

```
BlazorVendasADS/
    ├── BlazorVendasADS/
    ├── Entities/
    ├── Program.cs
    ├── BlazorVendasADS.sln
    └── .gitignore
```

 **ERRADO:**

BlazorVendasADS/Entities/.git

3. PRIMEIRO COMMIT (DO JEITO CERTO)

 **Passo a passo COMPLETO**

1 Entrar na pasta raiz

cd caminho/do/projeto

2 Iniciar o Git

git init

3 Criar o .gitignore (OBRIGATÓRIO)

touch .gitignore

Conteúdo recomendado para .NET / Blazor:

bin/

obj/

.vs/

*.user

*.suo

 Isso evita subir lixo de build.

4 Adicionar arquivos

git add .

O ponto (.) significa: **tudo que está aqui**, respeitando o .gitignore.

5 Criar o primeiro commit

```
git commit -m "Primeiro commit do projeto"
```

6 Conectar ao GitHub

```
git remote add origin https://github.com/SEU_USUARIO/SEU_REPO.git
```

Verificar:

```
git remote -v
```

7 Usar branch padrão main

```
git branch -M main
```

8 Enviar para o GitHub (criar upstream)

```
git push --set-upstream origin main
```

 **Aqui o projeto fica oficialmente sincronizado.**

4. SEGUNDO COMMIT (E TODOS OS OUTROS)

Fluxo PADRÃO (90% dos casos)

```
git status
```

```
git add .
```

```
git commit -m "Mensagem clara do que foi feito"
```

```
git push
```

Commit de arquivo específico

```
git add Produto.cs
```

```
git commit -m "Ajusta entidade Produto"
```

```
git push
```

5. PADRÃO DE MENSAGENS (PROFISSIONAL)

 **Bons exemplos**

Adiciona entidades do domínio

Cria tela de cadastro de clientes

Ajusta validação de login

Refatora serviço de vendas

Maus exemplos

teste

commit 2

arrumando coisa

aaa

6. ERROS COMUNS E SOLUÇÕES

Erro: no upstream branch

Causa: branch não ligada ao GitHub

 Solução:

git push --set-upstream origin main

Erro: rejected (fetch first)

Causa: GitHub tem commits que você não tem

 Solução:

git pull origin main --allow-unrelated-histories

git push

Erro: src refspec main does not match any

Causa: branch main não existe localmente

 Solução:

git branch

git branch -M main

Subi só README ou arquivo errado

Causa: uso errado do git add

✓ Correção:

```
git add .
git commit -m "Adiciona projeto completo"
git push
```

📌 7. COMANDOS IMPORTANTES PARA CONSULTA

```
git status      # ver estado do projeto
git branch     # ver branch atual
git branch -vv  # ver upstream
git log        # ver histórico
git pull        # atualizar projeto
git push        # enviar alterações
```

📌 8. REGRAS DE OURO (DECORA ISSO)

- ✓ Git sempre na **raiz**
 - ✓ Sempre usar git status antes do commit
 - ✓ .gitignore antes do primeiro commit
 - ✓ git add . para projetos completos
 - ✓ Um commit = uma ideia
 - ✗ Nunca usar git push -f sem saber
 - ✗ Nunca versionar bin, obj, .vs
-

📌 9. MAPA MENTAL DO GIT

```
git init → inicia controle
git add  → prepara arquivos
git commit → salva versão
git push  → envia ao GitHub
git pull  → atualiza do GitHub
```

📌 10. PRÓXIMO NÍVEL (QUANDO QUISER)

- Branch dev + main (padrão empresa)

- Conventional Commits
 - GitFlow
 - Git no Visual Studio
 - Trabalhar em equipe sem conflito
-

CONCLUSÃO

Se você seguir **esse guia**, você:

- não perde histórico
- não quebra repositório
- trabalha no padrão de mercado
- evita **100% dos erros** que você cometeu hoje