# Introdução ao Git e GitHub

Controle de versões e colaboração em projetos de programação

Alexandre Rebechi da Silva
IFSul – Campus Passo Fundo

# O Que Vamos Aprender Hoje

## Objetivo

Compreender o funcionamento e a importância do Git e do GitHub no desenvolvimento de software

## Git

Sistema de controle de versão que funciona localmente no seu computador

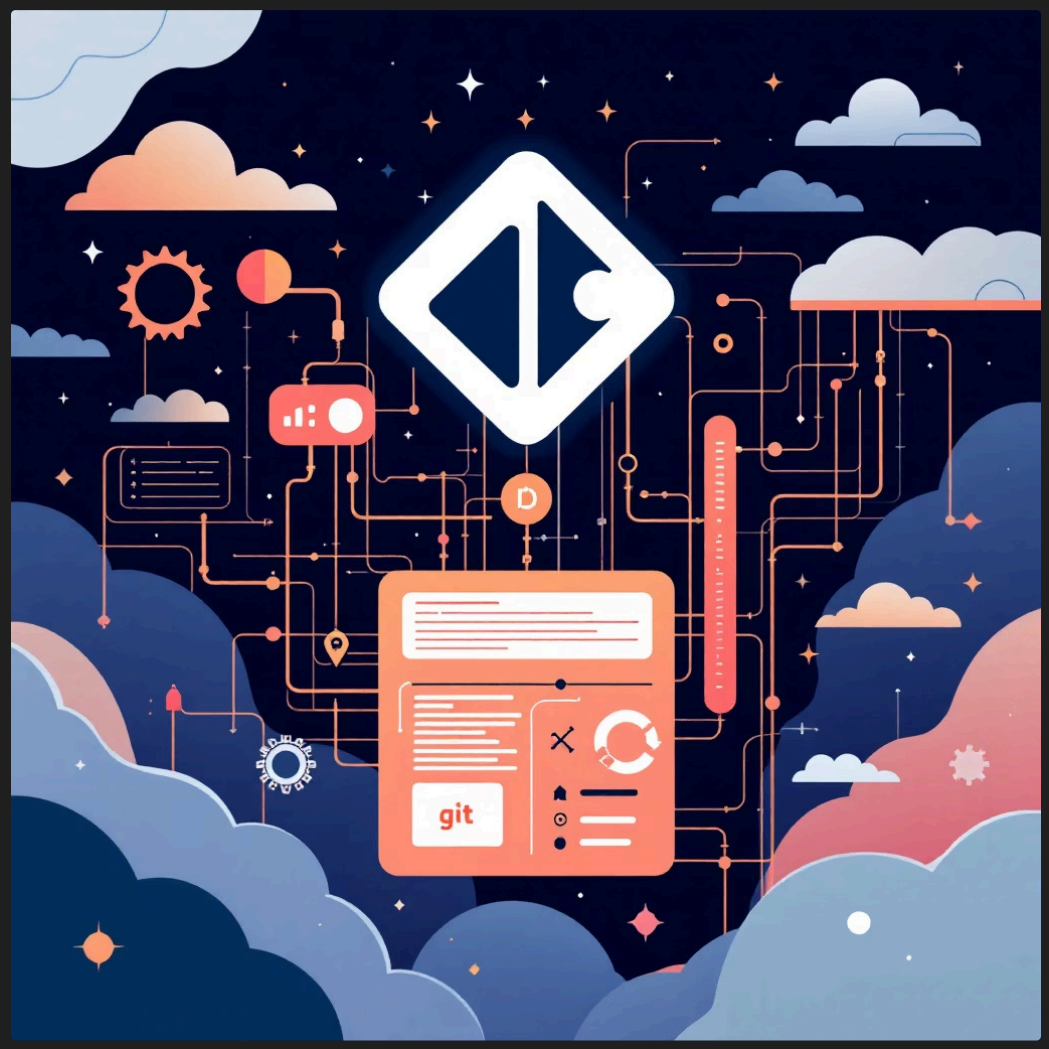
## GitHub

Plataforma online que hospeda repositórios Git e permite colaboração entre desenvolvedores

Essas ferramentas são essenciais para qualquer pessoa que trabalha com programação e desenvolvimento de projetos.

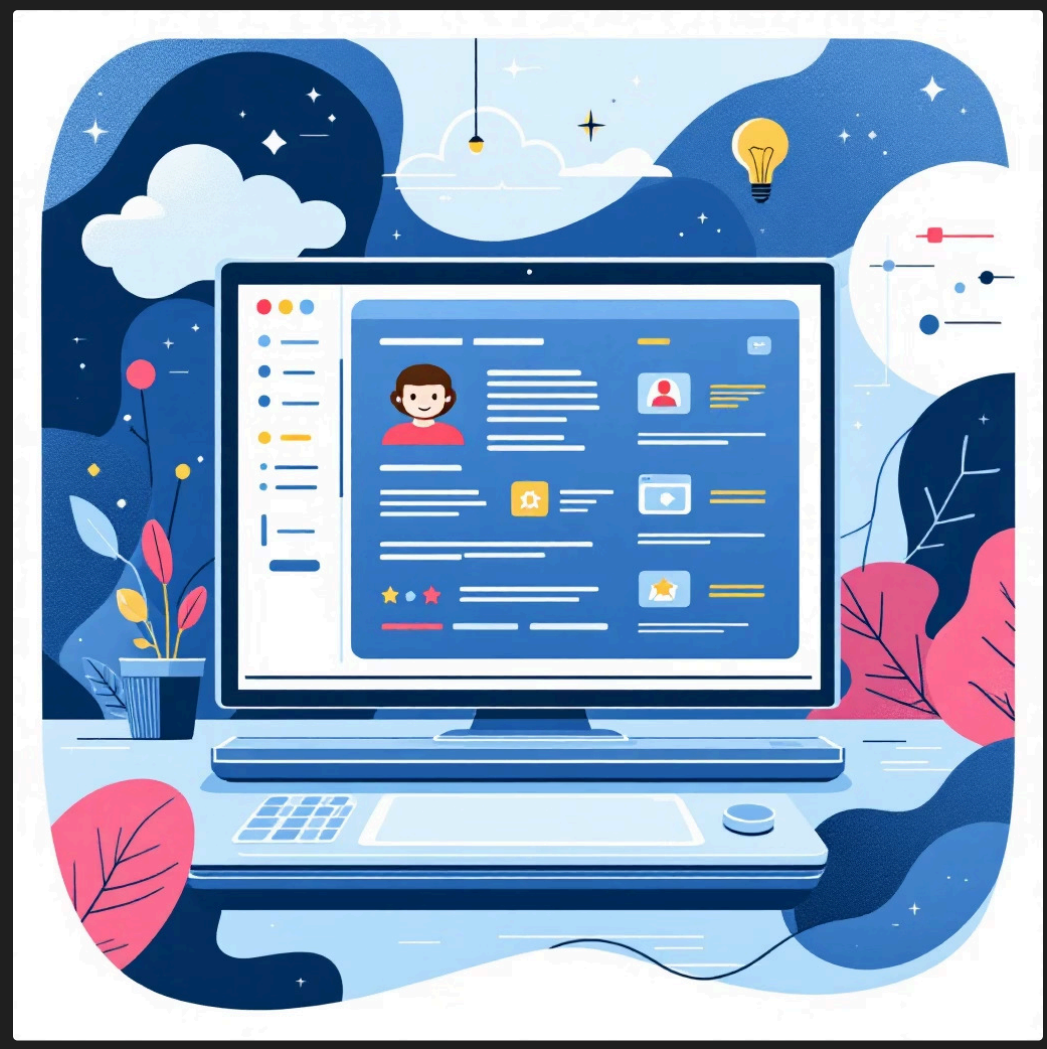
# Git vs GitHub: Entenda a Diferença

## Git



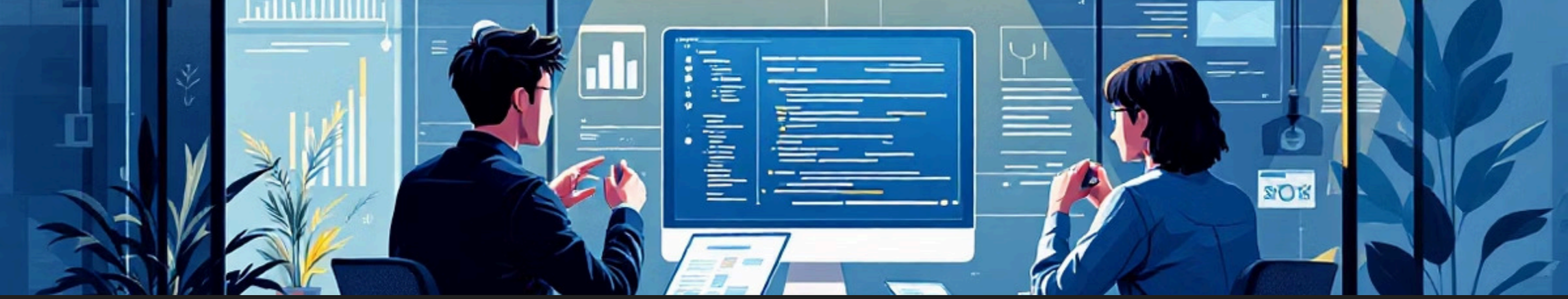

- Repositório local no seu computador
- Controle de versões offline
- Gerencia histórico de mudanças
- Trabalha com branches locais

## GitHub



- Repositório remoto na nuvem
- Hospedagem e compartilhamento
- Interface web intuitiva
- Facilita trabalho em equipe

**Em resumo:** Git faz o versionamento local do seu código, enquanto GitHub armazena tudo na nuvem e facilita a colaboração.



# Por Que Usar Git e GitHub?

### Controle de Histórico

Registre cada alteração feita no código e volte para versões anteriores quando necessário

### Ramificação de Projetos

Crie branches para desenvolver funcionalidades sem afetar o código principal

### Segurança e Organização

Mantenha seu código organizado e protegido contra perda de dados

1

### Hospedagem

Armazene código-fonte na nuvem

2

### Colaboração

Forks e Pull Requests facilitam contribuições

3

### Gestão

Issues para bugs e melhorias

# Preparando Seu Ambiente

01

## Instale as Ferramentas

Git, Visual Studio Code e GitHub Desktop são as ferramentas essenciais para começar

02

## Crie Sua Conta no GitHub

Escolha um nome profissional, use um e-mail válido e crie uma senha segura

03

## Configure Seu Perfil

Selecione seu tipo (professor/aluno), preencha bio, localidade e adicione um site se tiver

**Dica:** Use seu nome real ou um username profissional no GitHub – ele será seu cartão de visitas no mundo da programação!

# Comandos Essenciais do Git

### git init

Inicializa um novo repositório Git

### git status

Verifica o status dos arquivos

### git add .

Adiciona todos os arquivos

### git commit -m "mensagem"

Cria um commit com descrição

### git branch -M main

Define a branch principal

### git push -u origin main

Envia alterações para o GitHub

### git pull origin main

Atualiza repositório local com mudanças remotas

### git log

Mostra o histórico completo de commits

**Importante:** Sempre faça `git pull` antes de `git push` para evitar conflitos!

# Trabalhando com Branches

Branches permitem criar novas linhas de desenvolvimento sem afetar o código principal. É como ter versões paralelas do seu projeto.

1 **Criar Branch**

`git checkout -b nova-branch`

Cria e muda para nova branch

2 **Mudar de Branch**

`git checkout main`

Volta para a branch principal

3 **Unir Branches**

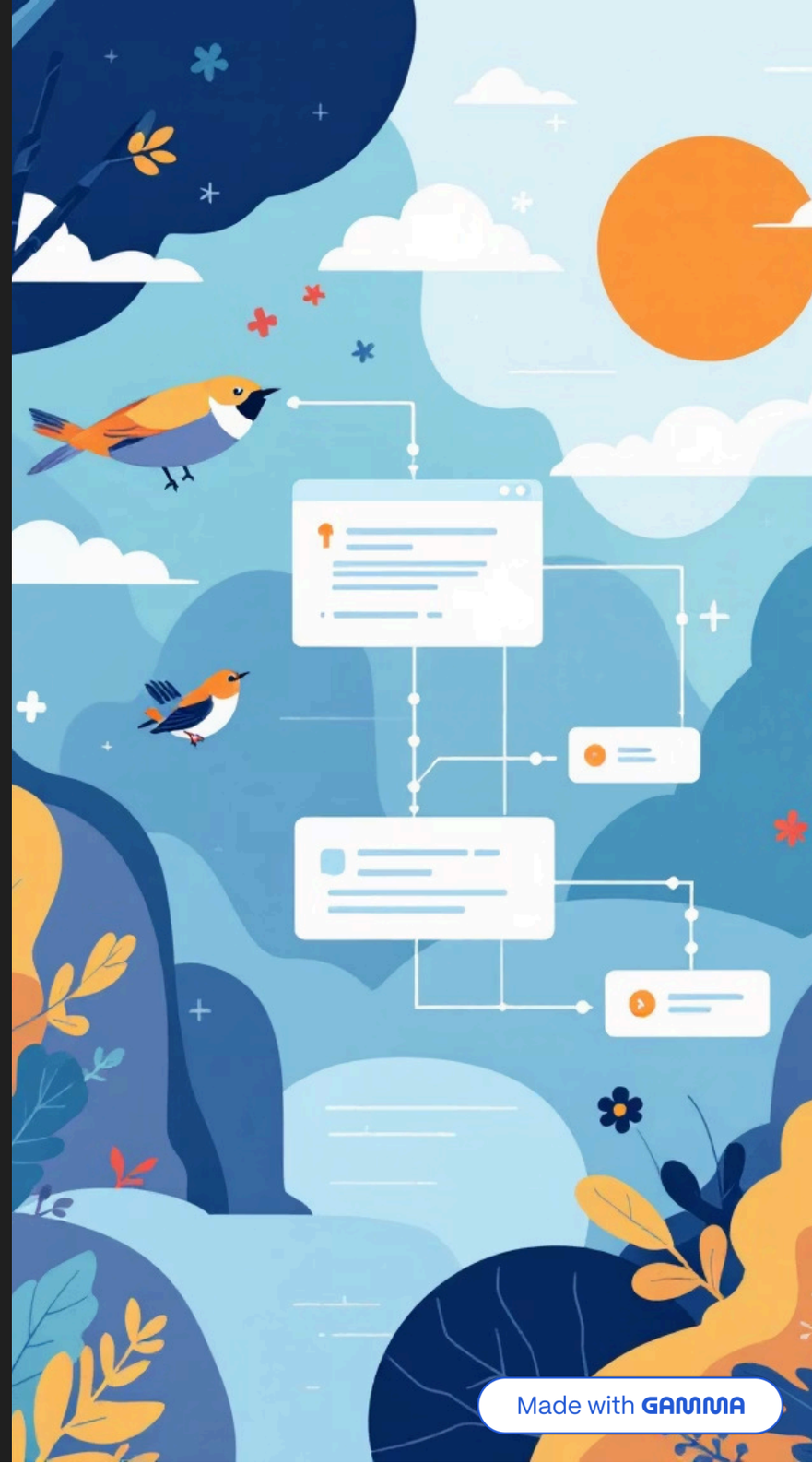
`git merge nome-da-branch`

Integra alterações à branch atual

4 **Clonar Projeto**

`git clone URL`

Baixa projeto do GitHub



# GitHub Desktop e Markdown

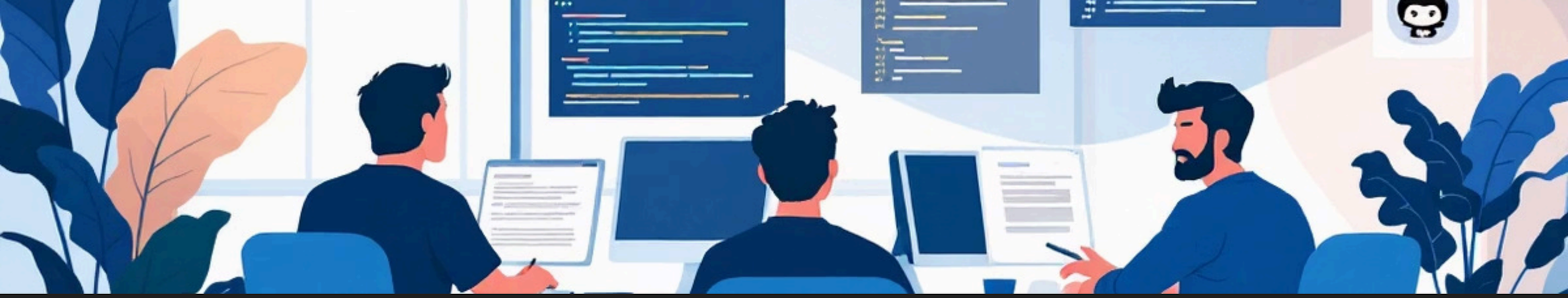
## Atalhos do GitHub Desktop


- **Ctrl + N** → Criar novo repositório
- **Ctrl + Shift + A** → Abrir no VS Code
- **Ctrl + Enter** → Fazer commit
- **Ctrl + P** → Push para GitHub

## Markdown Básico

Use Markdown para README, issues e Pull Requests:

```
**negrito** ou __negrito__
*itálico* ou _itálico_
# Título principal
## Subtítulo
- Item de lista
@usuário → menciona alguém
:emoji: → insere emoji
```



# Colaboração: Issues, Forks e Pull Requests

### Issues

Relate erros, tire dúvidas ou sugira melhorias. Use `@usuário` para mencionar colaboradores específicos.

### Fork

Cria uma cópia do repositório de outra pessoa na sua conta, permitindo que você experimente livremente.

### Pull Request

Propõe integrar suas mudanças ao projeto original. É como dizer: "olha essas melhorias que fiz!"

Essas ferramentas transformam o GitHub em uma verdadeira rede social de programadores, onde todos podem contribuir e aprender juntos.

# Boas Práticas e Conclusão

### Nunca comite direto na main

Use branches para desenvolver funcionalidades e manter a branch principal estável

### Escreva mensagens de commit claras

Descreva o que foi alterado de forma objetiva e compreensível

### Documente com README.md

Um bom README explica o que é o projeto, como instalar e usar

### Use issues para discussões

Mantenha todas as conversas sobre bugs e melhorias organizadas

> "A melhor forma de aprender Git e GitHub é praticando."

**Conclusão:** Git e GitHub são ferramentas essenciais para o desenvolvimento moderno. Elas garantem organização, segurança e facilitam a colaboração. Comece hoje mesmo a usá-las nos seus projetos!