## [ Uma breve citação para começar ]

“Isso aqui vai ser uma palestra de GitHub intermediário.”

— Um sábio contemporâneo (2025)

*Paráfrase livre de uma lenda que se foi...*

2025-10-09

# [ Capacitação GitHub ]

for_code[ ]

# [ Capacitação GitHub ]

Ronivaldo D. Andrade - Vice Diretor de Projetos – 9 de outubro de 2025



[ Capacitação GitHub ]
Ronivaldo D. Andrade - Vice Diretor de Projetos – 9 de outubro de 2025

Slide 01:

**Boa noite a todos!**

# [ O que é GitHub? ]

- Plataforma de hospedagem de código-fonte
- Utiliza sistema de controle de versão Git
- Permite colaboração em projetos
- Oferece recursos como:
  - GitHub Pages
  - GitHub Actions
  - Pull Requests
  - Code Review

Mão na massa!
No Capítulo 2 do Guia, a partir do Item 2.2, temos um guia resumido de como criar sua primeira conta no GitHub.

[ * ]





Slide 02:
**O GitHub é muito mais que um simples repositório de código.**
É uma plataforma completa que permite desenvolvedores colaborarem de forma eficiente, com ferramentas poderosas para versionamento, automação e deploy de projetos.

# [ GitHub Student Developer Pack ]

## Benefícios para Estudantes

- **Conta GitHub Pro gratuita**
- **3.000 minutos** do GitHub Actions (vs 2.000)
- **2GB** de armazenamento (vs 500MB)
- **GitHub Copilot** incluso
- **180 horas** de Codespaces (vs 120)
- **Confira o Guia, Capítulo 2, Item 2.2.1 para maiores informações.**

## Como obter

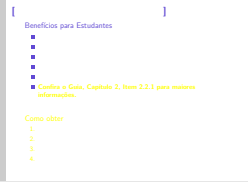
1. Adicionar e-mail acadêmico
2. Acessar education.github.com/pack
3. Enviar comprovante de matrícula
4. Aguardar a análise do documento e a aprovação

[ * ]



2025-10-09

# [ Capacitação GitHub ]




Slide 03:
**O GitHub Student Developer Pack é um programa incrível** que oferece benefícios de conta Pro gratuitamente para estudantes. Com ele você tem mais recursos para CI/CD, armazenamento e até o GitHub Copilot para ajudar na programação.

# [ Instalação e Configuração ]

**Ferramentas Necessárias:**

- Winget - Gerenciador de pacotes
- Git - Controle de versão
- GitHub CLI - Interface de linha de comando

**Comandos de Instalação:**

- `winget install Git.Git`
- `winget install GitHub.cli`
- `gh auth login`

## Configuração Básica do Git

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

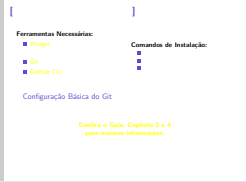
**Confira o Guia, Capítulo 3 e 4 para maiores informações.**

[ * ]



2025-10-09

# [ Capacitação GitHub ]




Slide 04:
**Vamos instalar as ferramentas essenciais usando o Winget.**
Configurem o Git com nome e e-mail - isso aparecerá nos commits.
Autentiquem no GitHub CLI, é mais prático que ficar digitando token.

# [ Comandos Git Essenciais ]

**Comandos Básicos:**

- `git init` - Inicializa repositório
- `git clone` - Clona repositório remoto
- `git add` - Adiciona arquivos ao stage
- `git commit` - Salva alterações
- `git push` - Envia para o remoto
- `git pull` - Atualiza do remoto

**Fluxo Básico:**

1. `git add .`
2. `git commit -m "msg"`
3. `git push`

**Branching:**

- `git branch`
- `git checkout -b`
- `git merge`

[ * ]





Slide 05:
**Estes são os comandos Git que vocês vão usar diariamente.**
O fluxo básico é: adicionar mudanças, commitar com mensagem clara, e enviar para o repositório remoto. Trabalhem com branches para organizar features diferentes.

# [ Git LFS - Arquivos Grandes ]

## O que é Git LFS?

Extensão do Git para versionar arquivos grandes (imagens, vídeos, modelos)

- **Problema:** Git tradicional não lida bem com arquivos >100MB
- **Solução:** Armazena apenas ponteiros no repositório
- **Conteúdo real** fica em servidor separado

## Como usar

```
git lfs install
git lfs track "*.psd"
git lfs track "*.zip"
git add .gitattributes

git add arquivo-grande.zip
```







Slide 06:
**Git LFS resolve um problema comum: arquivos grandes.**
Em vez de armazenar o arquivo inteiro no histórico,
ele guarda apenas um ponteiro, mantendo o repositório leve.
Essencial para projetos com assets pesados.

# [ Commits Profissionais ]

## Boas Práticas

- **Commits atômicos** - Uma mudança lógica por commit
- **Mensagens claras** - Assunto + descrição (se necessário)
- **Modo imperativo** - "Adiciona", "Corrige", "Remove"
- **Referência a issues** - "Closes #42", "Fixes #123"

## Exemplo de Commit

git commit -m "<tipo>(escopo): <descrição>"
**git commit -m "feat(login): adiciona autenticação via OAuth2"**

Implementa sistema de login usando OAuth2 do Google.
Inclui validação de tokens e tratamento de erros.

Closes #15

[ * ]





Slide 07:
**Commits bem feitos facilitam a vida de toda a equipe.**
Façam commits pequenos e focados, com mensagens claras que expliquem O QUE foi feito e POR QUE foi feito.
Usem padrões como Conventional Commits.

# [ Merge vs Rebase ]

**Merge**

- Preserva histórico completo
- Cria commit de merge
- Mais seguro para histórico compartilhado
- `git merge feature-branch`

**Rebase**

- Histórico linear e limpo
- Reescreve histórico
- Ideal para branches locais
- `git rebase main`

Importante!

- **Nunca** faça rebase em histórico compartilhado
- Use `--force-with-lease` em vez de `--force`
- Combine com a equipe qual estratégia usar

[ * ]






Slide 08:
**Merge e Rebase são duas formas de integrar branches.**
Merge é mais seguro e preserva todo o histórico.
Rebase deixa o histórico mais limpo mas reescreve commits.
Cuidado com rebase em branches compartilhadas!

# [ Pull Requests ]

## O que são Pull Requests?

Solicitação formal para integrar mudanças de uma branch em outra, com revisão de código

**Fluxo Básico:**

1. Criar branch para feature/bugfix
2. Desenvolver e commitar localmente
3. Fazer push da branch
4. Criar PR no GitHub
5. Revisão e aprovação
6. Merge no branch principal

[ * ]

## Via GitHub CLI

```
gh pr create --title "Minha feature" --body "Descrição"
```






Slide 09:
**Pull Requests são o coração da colaboração no GitHub.**
Permitem revisão de código, discussão sobre implementação e integração controlada de mudanças.
Podem ser criados tanto pela web quanto pela CLI.

# [ GitHub Pages & Actions ]

**GitHub Pages**

- Hospedagem gratuita de sites estáticos
- Suporte a domínios customizados
- Integração com Jekyll, Hugo, etc.
- Atualização automática

**GitHub Actions**

- Automação de CI/CD
- Testes automatizados
- Deploy automático
- Workflows customizáveis

## Exemplo de Integração

Push na main → Tests → Build → Deploy no GitHub Pages







Slide 10:
**GitHub Pages e Actions são recursos poderosos.**
Pages permite hospedar sites diretamente do repositório.
Actions automatiza testes, builds e deploys.
Juntos, criam um pipeline completo de desenvolvimento.

# [ Assinatura GPG de Commits ]

## Por que assinar commits?

- **Autenticidade** - Confirma que você fez o commit
- **Integridade** - Garante que não foi alterado
- **Verificação** - Mostra "Verified"no GitHub

**Configuração:**

1. Gerar chave GPG: `gpg --full-generate-key`
2. Configurar Git: `git config --global user.signingkey ID`
3. Adicionar chave pública no GitHub
4. Commitar com `git commit -S -m "msg"`

[ * ]






Slide 11:
**Assinar commits com GPG adiciona uma camada de segurança.**
Garante que os commits foram realmente feitos por você
e não foram adulterados. No GitHub, aparecem como "Verified".
É uma prática profissional importante.

# [ Exercícios Práticos ]

## O que vamos praticar?

- Criar repositório via GitHub CLI
- Clonar e fazer mudanças
- Trabalhar com branches
- Fazer Pull Requests
- Configurar GitHub Pages
- Criar workflow básico no Actions

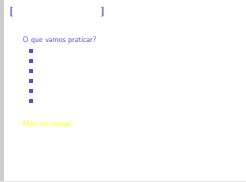
## Mão na massa!

Agora acessem o capitulo 9 do Guia que preparei, pois lá tem o passo a passo para alguns exercícios.

[ * ]








Slide 12:
**Agora é hora de colocar a mão na massa!**
Vamos praticar desde a criação do repositório até deploy automático com GitHub Pages.
Preparem suas máquinas e vamos codar!

for_code[ ]

# Obrigado!

Capacitação em GitHub

Vice Diretor de Projetos: Ronivaldo D. Andrade

9 de outubro de 2025



_ [ ]
Obrigado!
Capacitação em GitHub
Vice Diretor de Projetos: Ronivaldo D. Andrade
9 de outubro de 2025