# Newsletter Engenharia de Software

## Resumos e Materiais de Estudo

*Seu Nome*

*Engenharia de Software 2026*

# Table of contents

# 1. 👋 Bem-vindo à Newsletter de Engenharia de Software!

Este é um projeto pessoal que eu criei para documentar e compartilhar com a turma os conteúdos do curso de Engenharia de Software. A ideia é simples: **ensinar é a melhor forma de aprender**.

## 1.1 📬 O que você encontra aqui

**🔥 Resumos Semanais**

Toda semana, resumos das aulas com explicações claras, exemplos práticos e códigos.

**📑 Aulas Organizadas**

Conteúdo organizado por semana e aula, fácil de acompanhar e revisar.

**🖍️ Materiais de Apoio**

Cheatsheets, vídeos, ferramentas e templates prontos para impressão.

## 1.2 🚀 Por onde começar?

| Se você quer... | Vá para... |
|---|---|
| Acompanhar as aulas na ordem | 📅 2026 |
| Ver a semana atual | Semana 01 |
| Baixar materiais de estudo | 🧠 Materiais |

## 1.3 📆 Última Atualização

| Semana | Aulas | Data |
|---|---|---|
| Semana 01 | GitHub, Git Terminal, Arquitetura de Hardware | Fevereiro 2026 |

## 1.4 💡 Sobre este projeto

*"A melhor maneira de aprender é ensinar."* — Frank Oppenheimer

Este projeto nasceu da vontade de fixar melhor os conteúdos da faculdade e, ao mesmo tempo, ajudar os colegas de turma.

**Feito com ☕ (muito café) por um estudante de Engenharia de Software**

2026-02-18

2026-02-17

# 2. 📅 2026

## 2.1 📅 Ano 2026

Bem-vindo ao conteúdo do primeiro ano de Engenharia de Software!

### 2.1.1 📚 Semanas de Aula

| Semana | Tema Principal | Aulas |
|---|---|---|
| Semana 01 | Git, GitHub e Arquitetura de Hardware | 3 aulas |
| Semana 02 | *Em breve* | - |
| Semana 03 | *Em breve* | - |

### 2.1.2 🎯 Disciplinas do Semestre

Este semestre estamos cursando:

- 🌐 **Ferramentas WEB e UX** — HTML, CSS, JavaScript, Figma
- 🖥️ **Sistemas Operacionais** — Linux, terminal, processos
- 📝 **Linguagem e Pesquisa** — Escrita acadêmica, metodologia
- 💻 **Fundamentos da Programação** — Lógica, Python
- 🚀 **PAC** — Projeto integrador

### 2.1.3 📊 Progresso

```
Semana 01: ████████████████████████████ 100%
Semana 02: ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ 0%
Semana 03: ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ 0%
```

> **Dica**
>
> Use o menu lateral para navegar entre as semanas e aulas!

2026-02-18

2026-02-17

# 2.2 Semana 01

## 2.2.1 📆 Semana 01

**Período**

**Data:** Fevereiro de 2026
**Tema Principal:** Git, GitHub e Arquitetura de Hardware

---

### 🎯 Objetivo da Semana

Nesta primeira semana, aprendemos os fundamentos essenciais para qualquer desenvolvedor:

- **Controle de versão** com Git e GitHub
- **Conhecimento de hardware** — o que tem dentro de um computador e como funciona

---

### 📚 Aulas da Semana

| # | Aula | Descrição | Status |
|---|---|---|---|
| 1 | GitHub - Introdução | O que é Git/GitHub, criação de conta, conceitos básicos | ✅ |
| 2 | Git pelo Terminal | Comandos práticos, workflow completo, branches | ✅ |
| 3 | Arquitetura de Hardware | Componentes do PC, montagem, Kernel, manutenção | ✅ |

---

### 🧠 Resumo dos Conceitos

#### 🔀 GIT E GITHUB

**O que é Git?**

- Sistema de controle de versão
- Guarda histórico de todas as alterações do código
- Permite voltar a versões anteriores
- Funciona offline no seu computador

**O que é GitHub?**

- Plataforma na nuvem para hospedar repositórios
- Funciona como portfólio e rede social de devs
- Empresas olham seu GitHub antes de contratar

**Comandos essenciais:**

```
git init              # Inicia repositório
git add .             # Adiciona arquivos ao stage
git commit -m "msg"   # Salva alterações
git push              # Envia para GitHub
git pull              # Baixa do GitHub
git checkout -b nome  # Cria nova branch
git merge nome        # Mescla branch
```

---

🖥️ ARQUITETURA DE HARDWARE

**Componentes principais de um PC:**

| Componente | Função |
|---|---|
| **CPU (Processador)** | Cérebro do computador — executa instruções e cálculos |
| **RAM (Memória)** | Armazena dados temporários enquanto o PC está ligado |
| **HD / SSD** | Armazena arquivos permanentemente |
| **Placa-mãe** | Conecta todos os componentes |
| **Fonte (PSU)** | Fornece energia para o PC |
| **GPU (Placa de Vídeo)** | Processa gráficos e imagens |
| **Cooler** | Resfria o processador |

**Analogia importante:**

- **RAM** = Mesa de trabalho (espaço para trabalhar agora)
- **HD/SSD** = Armário (guarda arquivos permanentemente)

---

🧠 O KERNEL

O **Kernel** é o núcleo do sistema operacional — o intermediário entre software e hardware.

**Funções do Kernel:**

- Gerenciar processos (qual programa usa a CPU)
- Gerenciar memória (distribuir RAM)
- Gerenciar dispositivos (comunicar com hardware)
- Gerenciar arquivos (organizar dados no disco)
- Segurança (impedir acessos não autorizados)

**Analogia:** O Kernel é como um gerente de hotel — distribui quartos, organiza filas e garante segurança.

---

🔧 PROBLEMAS COMUNS DE HARDWARE

| Problema | Causa Provável | Solução |
|---|---|---|
| PC não liga | Fonte/cabos soltos | Verificar conexões |
| Liga mas sem imagem | RAM mal encaixada | Reencaixar RAM |
| Desliga sozinho | Superaquecimento | Limpar poeira |
| Muito lento | HD antigo / pouca RAM | Trocar por SSD / adicionar RAM |

**Dica de ouro:** Quando não aparecer vídeo, tire a RAM, limpe os contatos com borracha branca e recoloque. Resolve 70% dos casos!

---

### 📎 Materiais de Apoio

- 📄 Cheatsheet Git
- 📄 Cheatsheet Linux

---

### ✅ Checklist da Semana

GIT E GITHUB

- [ ] Criei minha conta no GitHub
- [ ] Instalei o Git no computador
- [ ] Configurei nome e email
- [ ] Fiz meu primeiro commit
- [ ] Enviei código para o GitHub com `git push`
- [ ] Entendi o conceito de branches

HARDWARE

- [ ] Sei identificar os principais componentes de um PC
- [ ] Entendo a função de cada componente
- [ ] Sei o básico de montagem/desmontagem
- [ ] Conheço problemas comuns e suas soluções
- [ ] Entendo o que é o Kernel e sua função

---

### 🎬 Vídeos Recomendados

GIT E GITHUB

| Vídeo | Canal |
|---|---|
| Git e GitHub para Iniciantes | Rafaella Ballerini |
| Como usar Git na prática | Código Fonte TV |

HARDWARE

| Vídeo | Canal |
|---|---|
| Como montar um PC | Adrenaline |
| O que é o Kernel | Fabio Akita |

---

**Dúvidas?**

Me procura no grupo da turma! 💬

2026-02-18

2026-02-17

## 2.2.2 Semana 01 — Git e GitHub: Primeiros Passos

**Informações da Aula**

**Disciplina:** Ferramentas WEB e UX / Sistemas Operacionais
**Data:** Fevereiro de 2026
**Professor:** Ícaro
**Tema:** Introdução ao Git e GitHub

---

### 🎯 O que você vai aprender

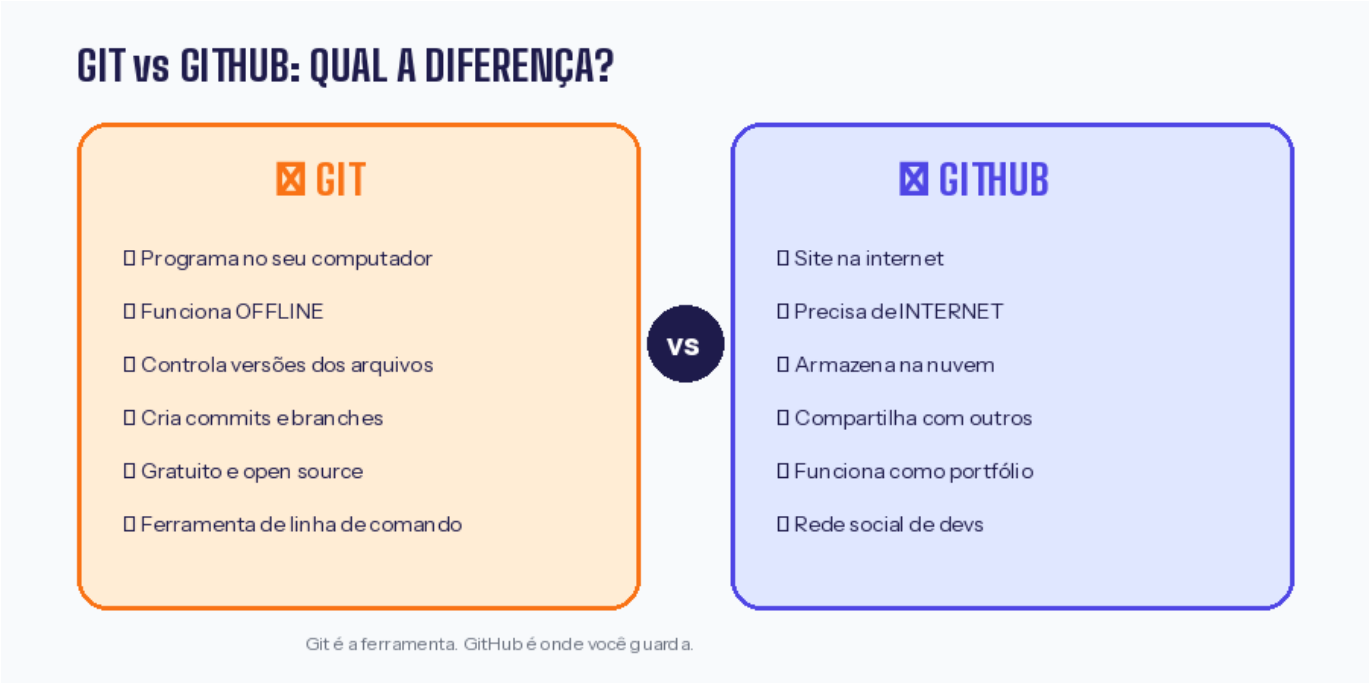
Nesta aula, demos os primeiros passos no mundo do controle de versão. Ao final, você será capaz de:

- ✅ Entender o que é Git e por que ele é essencial
- ✅ Criar uma conta no GitHub
- ✅ Instalar e configurar o Git
- ✅ Conectar o VS Code ao GitHub
- ✅ Fazer seus primeiros commits
- ✅ Criar e entender branches

---

### 🤔 O que é Git? E GitHub?

Antes de começar, vamos entender a diferença:




| Git | GitHub |
|---|---|
| É um **programa** instalado no seu computador | É um **site** na internet |
| Controla as versões dos seus arquivos | Armazena seus projetos na nuvem |
| Funciona offline | Precisa de internet |
| É a **ferramenta** | É o **serviço de hospedagem** |

> 🔥 **Analogia simples**
>
> Pense assim:
>
> • **Git** = O programa Word no seu PC
> • **GitHub** = O Google Drive onde você salva seus documentos
>
> Você usa o Git para controlar seu código, e o GitHub para guardar e compartilhar na nuvem.

---

### 🎬 Por que usar controle de versão?

Você já passou por isso?

```
📁 Meu Projeto
├── trabalho.doc
├── trabalho-final.doc
├── trabalho-final-v2.doc
├── trabalho-final-v2-REVISADO.doc
├── trabalho-final-v2-REVISADO-AGORA-VAI.doc
└── trabalho-final-v2-REVISADO-AGORA-VAI-FINAL-MESMO.doc
```

😅 Com o Git, você teria apenas:

```
📁 Meu Projeto
└── trabalho.doc
    └── (histórico completo de todas as versões guardado pelo Git)
```

BENEFÍCIOS DO GIT:

1. **Histórico completo** — Nunca perde nada, pode voltar a qualquer versão
2. **Trabalho em equipe** — Várias pessoas editando o mesmo projeto
3. **Backup automático** — Seu código está seguro no GitHub
4. **Portfólio profissional** — Empresas olham seu GitHub antes de contratar

---

### 🌐 GitHub como Rede Social e Currículo

O GitHub não é só para guardar código. Ele funciona como:

📋 CURRÍCULO DE DESENVOLVEDOR

- Mostra **todos os seus projetos** públicos
- Exibe seu **histórico de contribuições** (os quadradinhos verdes)
- Empresas **avaliam candidatos** pelo GitHub
- Você pode fixar seus **melhores projetos** no perfil

🤝 REDE SOCIAL DE DESENVOLVEDORES

- Seguir outros programadores
- Dar ⭐ (estrelas) em projetos que você gosta
- Contribuir com projetos de outras pessoas
- Participar de discussões

> ⚠️ **Dica importante**
>
> Mantenha seu GitHub ativo! Mesmo projetos pequenos da faculdade contam. Recrutadores olham a consistência (quadradinhos verdes frequentes) mais do que projetos gigantes.

---

### 🖥️ Instalação do Git

NO WINDOWS

1. Acesse: git-scm.com/downloads
2. Clique em **"Download for Windows"**
3. Execute o instalador
4. **Importante:** Nas opções, mantenha tudo padrão, mas em "Default editor" escolha **Visual Studio Code**
5. Finalize a instalação

NO LINUX (UBUNTU/MINT)

Abra o terminal e execute:

```
sudo apt update
sudo apt install git -y
```

VERIFICAR SE INSTALOU

```
git --version
```

Deve aparecer algo como: `git version 2.43.0`

---

### ⚙️ Configuração Inicial do Git

Após instalar, você precisa dizer ao Git quem você é. Abra o terminal e execute:

```
# Configurar seu nome (use seu nome real)
git config --global user.name "Seu Nome Completo"

# Configurar seu email (use o MESMO email do GitHub)
git config --global user.email "seu.email@exemplo.com"

# Configurar o VS Code como editor padrão
git config --global core.editor "code --wait"

# Configurar o nome padrão da branch principal
git config --global init.defaultBranch main
```

#### VERIFICAR SUAS CONFIGURAÇÕES

```
git config --list
```

---

### 📝 Criando sua Conta no GitHub

1. Acesse: github.com
2. Clique em **"Sign up"**
3. Preencha:
   - **Email:** Use um email que você acessa sempre
   - **Password:** Senha forte (guarde bem!)
   - **Username:** Escolha com cuidado! Este será seu "nome artístico" como dev
4. Complete a verificação
5. Escolha o plano **Free** (é suficiente!)

> **🔥 Dica para o username**
>
> - ✅ Bom: `joaosilva`, `maria-dev`, `carlos123`
> - ❌ Evite: `xX_destroyer_Xx`, `gatinha2005`, `asdfgh`
>
> Lembre-se: recrutadores vão ver isso!

---

### 🔗 Conectando VS Code ao GitHub

#### PASSO 1: ABRA O VS CODE

#### PASSO 2: INSTALE A EXTENSÃO DO GITHUB

1. Clique no ícone de extensões (quadradinhos) na barra lateral
2. Pesquise: **"GitHub Pull Requests"**
3. Instale a extensão oficial da Microsoft

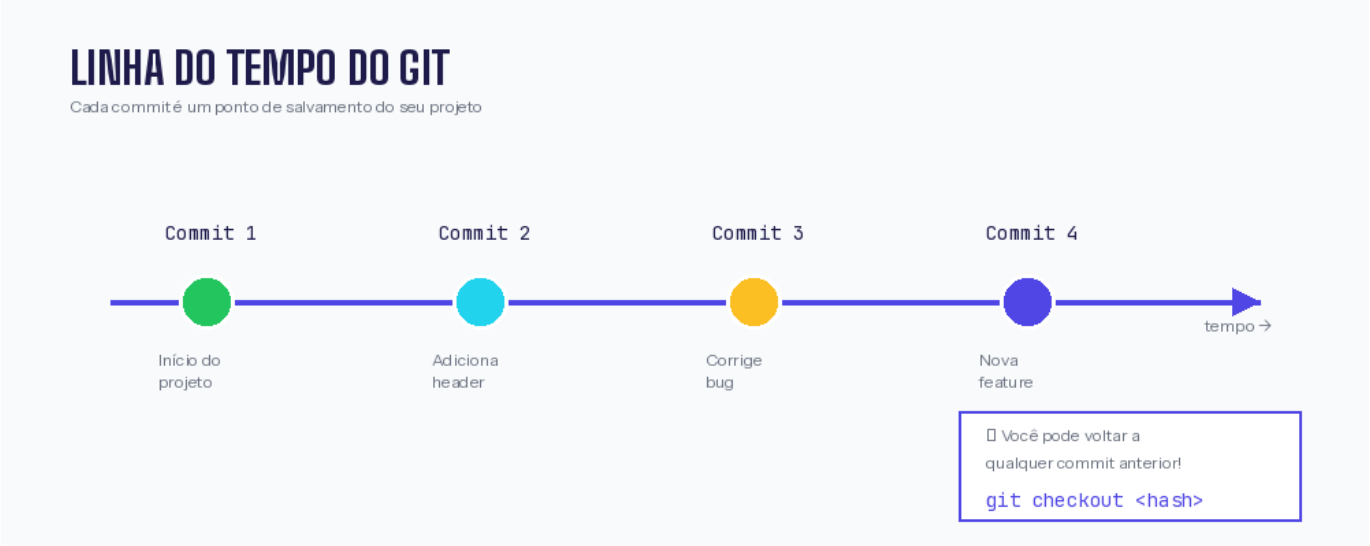
#### PASSO 3: FAÇA LOGIN

1. Clique no ícone de pessoa no canto inferior esquerdo
2. Clique em **"Sign in to GitHub"**
3. O navegador vai abrir, autorize o acesso
4. Pronto! VS Code conectado ao GitHub

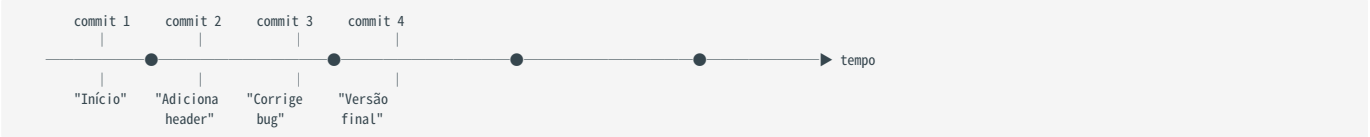
---

### 📊 Entendendo a Linha do Tempo do Git

Esta é a parte mais importante para entender o Git!




IMAGINE UMA LINHA DO TEMPO:




Cada **bolinha (●)** é um **commit** = um "ponto de salvamento" do seu projeto.

O QUE É UM COMMIT?

Um commit é como uma **foto** do seu projeto naquele momento. Ele guarda:

- 📁 Todos os arquivos
- 📅 Data e hora
- 👤 Quem fez
- 💬 Uma mensagem explicando o que mudou

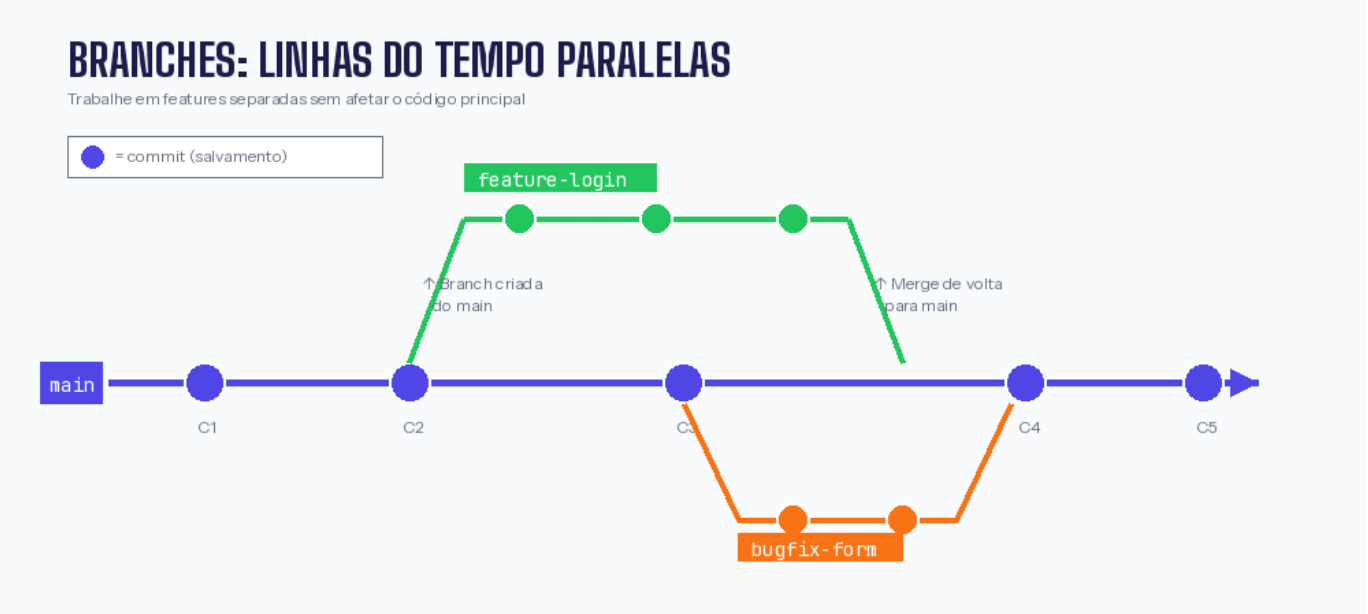
**Exemplo de histórico**

```
abc1234 - Maria - "Adiciona página de contato"
def5678 - João  - "Corrige erro no formulário"
ghi9012 - Maria - "Melhora estilo do botão"
```

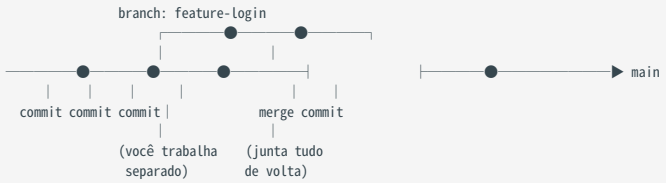
---

### 🌿 O que são Branches?

Branch significa "galho" em inglês. É como se você criasse uma **linha do tempo alternativa** para testar coisas sem afetar o projeto principal.




VISUALIZAÇÃO:




POR QUE USAR BRANCHES?

1. **Testar sem medo** — Se der errado, é só apagar a branch
2. **Trabalho em equipe** — Cada pessoa trabalha em sua branch
3. **Organização** — Separa funcionalidades diferentes

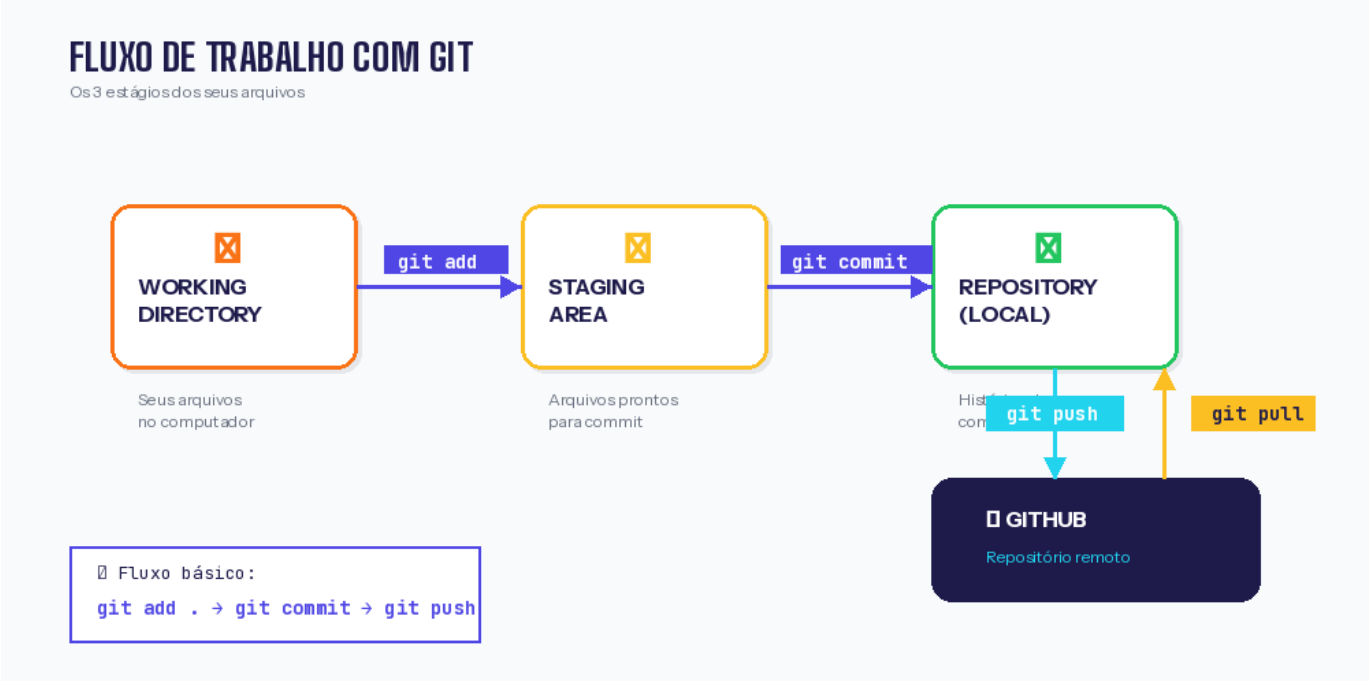
COMANDOS BÁSICOS DE BRANCH

```
# Ver em qual branch você está
git branch

# Criar uma nova branch
git branch nome-da-branch

# Mudar para outra branch
git checkout nome-da-branch

# Criar E mudar para nova branch (atalho)
git checkout -b nome-da-branch
```

---

### 🔄 Fluxo de Trabalho: Os 3 Estágios

Entenda como seus arquivos "viajam" pelo Git:




1. **Working Directory** — Seus arquivos normais no computador
2. **Staging Area** — Arquivos marcados para o próximo commit ( `git add` )
3. **Repository** — Histórico de commits salvos ( `git commit` )
4. **GitHub** — Cópia na nuvem ( `git push` )

---

### 🚀 Seu Primeiro Repositório: Passo a Passo

Vamos criar seu primeiro projeto do zero!

**PASSO 1: CRIAR PASTA DO PROJETO**

```
# Criar pasta
mkdir meu-primeiro-repo

# Entrar na pasta
cd meu-primeiro-repo
```

**PASSO 2: INICIALIZAR O GIT**

```
git init
```

Isso cria uma pasta oculta `.git` que guarda todo o histórico.

**PASSO 3: CRIAR UM ARQUIVO**

```
# Criar arquivo README
echo "# Meu Primeiro Repositório" > README.md
echo "Estou aprendendo Git!" >> README.md
```

**PASSO 4: VERIFICAR STATUS**

```
git status
```

Vai mostrar que `README.md` está "Untracked" (não rastreado).

**PASSO 5: ADICIONAR ARQUIVO AO STAGE**

```
git add README.md
```

```
# Ou para adicionar TODOS os arquivos:
git add .
```

**PASSO 6: FAZER O COMMIT**

```
git commit -m "Meu primeiro commit! 🎉"
```

**PASSO 7: CRIAR REPOSITÓRIO NO GITHUB**

1. Vá em github.com
2. Clique no **"+" → "New repository"**
3. Nome: `meu-primeiro-repo`
4. **NÃO** marque "Add README"
5. Clique em **"Create repository"**

**PASSO 8: CONECTAR E ENVIAR**

```
# Conectar ao GitHub
git remote add origin https://github.com/SEU_USUARIO/meu-primeiro-repo.git

# Enviar para o GitHub
git push -u origin main
```

**🎉 PARABÉNS! SEU CÓDIGO ESTÁ NO GITHUB!**

---

### 📋 Resumo dos Comandos

| Comando | O que faz |
|---|---|
| `git init` | Inicia repositório na pasta |
| `git status` | Mostra situação dos arquivos |
| `git add .` | Adiciona todos ao stage |
| `git commit -m "msg"` | Salva com mensagem |
| `git push` | Envia para o GitHub |
| `git pull` | Baixa do GitHub |
| `git branch` | Lista branches |
| `git checkout -b nome` | Cria e muda de branch |

---

### 🎬 Vídeos Recomendados

Para fixar o conteúdo, assista estes vídeos:

| Vídeo | Canal | Duração | Link |
|---|---|---|---|
| Git e GitHub para Iniciantes | Rafaella Ballerini | 34 min | YouTube |
| Como usar Git na prática | Rafaella Ballerini | 40 min | YouTube |
| Entendendo GIT | Fábio Akita | 60 min | YouTube |

---

### ✅ Checklist da Aula

Antes de ir para a próxima aula, confirme que você:

- [ ] Entendi a diferença entre Git e GitHub
- [ ] Criei minha conta no GitHub
- [ ] Instalei o Git no meu computador
- [ ] Configurei nome e email no Git
- [ ] Conectei o VS Code ao GitHub
- [ ] Criei meu primeiro repositório
- [ ] Fiz meu primeiro commit
- [ ] Enviei para o GitHub com `git push`
- [ ] Entendi o conceito de branches

---

### 🆘 Problemas Comuns

**"FATAL: NOT A GIT REPOSITORY"**

Você não está numa pasta com Git iniciado. Execute `git init` primeiro.

**"ERROR: FAILED TO PUSH"**

Tente `git pull` primeiro, depois `git push` novamente.

**"AUTHENTICATION FAILED"**

Use um Personal Access Token em vez de senha. Veja o guia no material.

---

> **Dúvidas?**
>
> Não entendeu algo? Me procura no grupo da turma ou abre uma issue no repositório!

2026-02-19

2026-02-17

## 2.2.3 Semana 02 — Git na Prática: Dominando o Terminal

> **Informações da Aula**
>
> **Disciplina:** Sistemas Operacionais / Ferramentas WEB
> **Data:** Fevereiro de 2026
> **Tema:** Git e GitHub pelo Terminal (Linux e Windows)
> **Obs:** Não tivemos essa aula. É um bônus de um apaixonado por Linux
> **Pré-requisito:** Ter completado a Aula 01 (conta no GitHub criada)

---

### 🎯 Objetivo da Aula

Nesta aula você vai aprender a usar Git e GitHub **100% pelo terminal**, sem depender de interface gráfica. Isso é essencial porque:

- 🖥️ Servidores não têm interface gráfica
- ⚡ Terminal é mais rápido que clicar em botões
- 💼 Entrevistas técnicas pedem conhecimento de terminal
- 🛠️ Dá mais controle sobre o que você está fazendo

---

### 🖥️ Abrindo o Terminal

NO LINUX (UBUNTU/MINT)

**Atalho:** `Ctrl + Alt + T`

Ou: Menu → Terminal

NO WINDOWS

**Opção 1 — Git Bash (Recomendado)**

Após instalar o Git, procure por "Git Bash" no menu iniciar.

**Opção 2 — PowerShell**

Clique direito no menu iniciar → "Terminal" ou "PowerShell"

**Opção 3 — CMD**

Pressione `Win + R`, digite `cmd`, pressione Enter.

> **Recomendação para Windows**
>
> Use o **Git Bash**! Ele simula um terminal Linux e os comandos são idênticos aos do Linux.

---

### ⚙️ Configuração Inicial (Primeira vez apenas)

Antes de usar o Git, você precisa se identificar. Execute estes comandos:

```
# Configurar seu nome (aparece nos commits)
git config --global user.name "Seu Nome Completo"

# Configurar seu email (MESMO do GitHub!)
git config --global user.email "seu.email@exemplo.com"

# Definir branch padrão como 'main'
```

```
git config --global init.defaultBranch main

# Configurar editor padrão (opcional)
git config --global core.editor "nano"
```

#### VERIFICAR SE CONFIGUROU CERTO

```
git config --list
```

Deve mostrar:

```
user.name=Seu Nome Completo
user.email=seu.email@exemplo.com
init.defaultbranch=main
```

---

### 📁 Projeto Prático: Criando um Portfólio

Vamos criar um projeto real do zero! Um portfólio pessoal simples.

#### PASSO 1 — CRIAR A PASTA DO PROJETO

```
# Ir para a pasta home
cd ~

# Criar pasta do projeto
mkdir meu-portfolio

# Entrar na pasta
cd meu-portfolio

# Verificar onde você está
pwd
```

**Saída esperada:**

```
/home/seu-usuario/meu-portfolio
```

---

#### PASSO 2 — INICIALIZAR O GIT

```
git init
```

**Saída esperada:**

```
Initialized empty Git repository in /home/seu-usuario/meu-portfolio/.git/
```

> 🖊️ **O que aconteceu?**
>
> O Git criou uma pasta oculta chamada `.git` dentro do seu projeto. É lá que ele guarda todo o histórico.

Para ver a pasta oculta:

```
ls -la
```

---

#### PASSO 3 — CRIAR O PRIMEIRO ARQUIVO

Vamos criar um README.md (arquivo de apresentação do projeto):

```
# Criar arquivo com echo
echo "# Meu Portfólio" > README.md
echo "" >> README.md
echo "Olá! Sou estudante de Engenharia de Software." >> README.md
echo "" >> README.md
echo "## Sobre mim" >> README.md
echo "- 🎓 Cursando Engenharia de Software" >> README.md
```

```
echo "- 💻 Aprendendo Git e GitHub" >> README.md
echo "- 🌱 Em constante evolução" >> README.md
```

**Verificar o conteúdo:**

```
cat README.md
```

**Saída esperada:**

```
# Meu Portfólio

Olá! Sou estudante de Engenharia de Software.

## Sobre mim
- 🎓 Cursando Engenharia de Software
- 💻 Aprendendo Git e GitHub
- 🌱 Em constante evolução
```

---

PASSO 4 — VERIFICAR O STATUS

```
git status
```

**Saída esperada:**

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present
```

**Traduzindo**

• **Untracked files** = Arquivos que o Git ainda não está monitorando

• O Git está te dizendo: "Ei, tem um arquivo novo aqui, quer que eu cuide dele?"

---

PASSO 5 — ADICIONAR AO STAGE (GIT ADD)

```
# Adicionar um arquivo específico
git add README.md

# OU adicionar todos os arquivos de uma vez
git add .
```

**Verificar novamente:**

```
git status
```

**Saída esperada:**

```
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
```

**Mudou de cor!**

Agora o arquivo está em **verde** (Changes to be committed). Isso significa que está pronto para o commit.

---

PASSO 6 — FAZER O COMMIT

```
git commit -m "Cria README com informações iniciais"
```

**Saída esperada:**

```
[main (root-commit) abc1234] Cria README com informações iniciais
 1 file changed, 8 insertions(+)
 create mode 100644 README.md
```


**Boas práticas para mensagens de commit**

- ✅ `"Adiciona página de contato"`
- ✅ `"Corrige erro no formulário de login"`
- ✅ `"Atualiza estilo do botão principal"`
- ❌ `"Alterações"`
- ❌ `"asdfgh"`
- ❌ `"commit"`

---

PASSO 7 — VER O HISTÓRICO

```
git log
```

**Saída esperada:**

```
commit abc1234def5678... (HEAD -> main)
Author: Seu Nome <seu.email@exemplo.com>
Date:   Mon Feb 17 10:30:00 2026 -0300

    Cria README com informações iniciais
```

**Versão resumida:**

```
git log --oneline
```

**Saída:**

```
abc1234 Cria README com informações iniciais
```

---

### 🌐 Conectando ao GitHub

Agora vamos enviar seu projeto para o GitHub!

#### PASSO 1 — CRIAR REPOSITÓRIO NO GITHUB

1. Acesse github.com
2. Clique no **"+"** → **"New repository"**
3. Preencha:

- **Repository name:** `meu-portfolio`
- **Description:** `Meu portfólio pessoal`
- **Public** ✓
- **NÃO marque** "Add a README file" (já temos!)

4. Clique **"Create repository"**

#### PASSO 2 — COPIAR O LINK DO REPOSITÓRIO

O GitHub vai mostrar instruções. Copie o link HTTPS:

```
https://github.com/SEU_USUARIO/meu-portfolio.git
```

#### PASSO 3 — CONECTAR REPOSITÓRIO LOCAL AO GITHUB

```
git remote add origin https://github.com/SEU_USUARIO/meu-portfolio.git
```

**Verificar se conectou:**

```
git remote -v
```

**Saída esperada:**

```
origin  https://github.com/SEU_USUARIO/meu-portfolio.git (fetch)
origin  https://github.com/SEU_USUARIO/meu-portfolio.git (push)
```

#### PASSO 4 — ENVIAR PARA O GITHUB (PUSH)

```
git push -u origin main
```

**O terminal vai pedir:**

```
Username for 'https://github.com': seu_usuario
Password for 'https://github.com': (cole seu token aqui)
```

> ⚠️ **ATENÇÃO: Não é sua senha!**
>
> O GitHub não aceita mais senha comum. Você precisa usar um **Personal Access Token**.

#### PASSO 5 — CRIAR O TOKEN (SE AINDA NÃO TIVER)

1. No GitHub: Foto de perfil → **Settings**

2. Menu lateral → **Developer settings**
3. **Personal access tokens → Tokens (classic)**
4. **Generate new token → Generate new token (classic)**
5. Preencha:

- **Note:** `terminal`
- **Expiration:** 90 days
- **Marque:** ☑️ `repo`

6. **Generate token**
7. **COPIE O TOKEN** (começa com `ghp_...`)

> ⚡ **Copie agora!**
>
> O token só aparece uma vez. Se perder, terá que criar outro.

PASSO 6 — COLAR O TOKEN

Quando pedir "Password", cole o token e pressione Enter.

**Não vai aparecer nada na tela** (é por segurança). Só cole e aperte Enter.

**Saída de sucesso:**

```
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 312 bytes | 312.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/SEU_USUARIO/meu-portfolio.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

🎉 VERIFIQUE NO GITHUB!

Acesse: `https://github.com/SEU_USUARIO/meu-portfolio`

Seu código está lá!

---

### 📝 Fazendo Alterações e Novos Commits

Vamos adicionar mais conteúdo ao projeto.

CRIAR ARQUIVO DE PROJETOS

```
# Criar arquivo
echo "# Meus Projetos" > projetos.md
echo "" >> projetos.md
echo "## 1. Newsletter Engenharia de Software" >> projetos.md
echo "Site com resumos semanais das aulas." >> projetos.md
echo "" >> projetos.md
echo "## 2. Meu Portfólio" >> projetos.md
echo "Este repositório!" >> projetos.md
```

VERIFICAR STATUS

```
git status
```

**Saída:**

```
On branch main
Untracked files:
        projetos.md
```

ADICIONAR E COMMITAR

```
git add projetos.md
git commit -m "Adiciona lista de projetos"
```

ENVIAR PARA O GITHUB

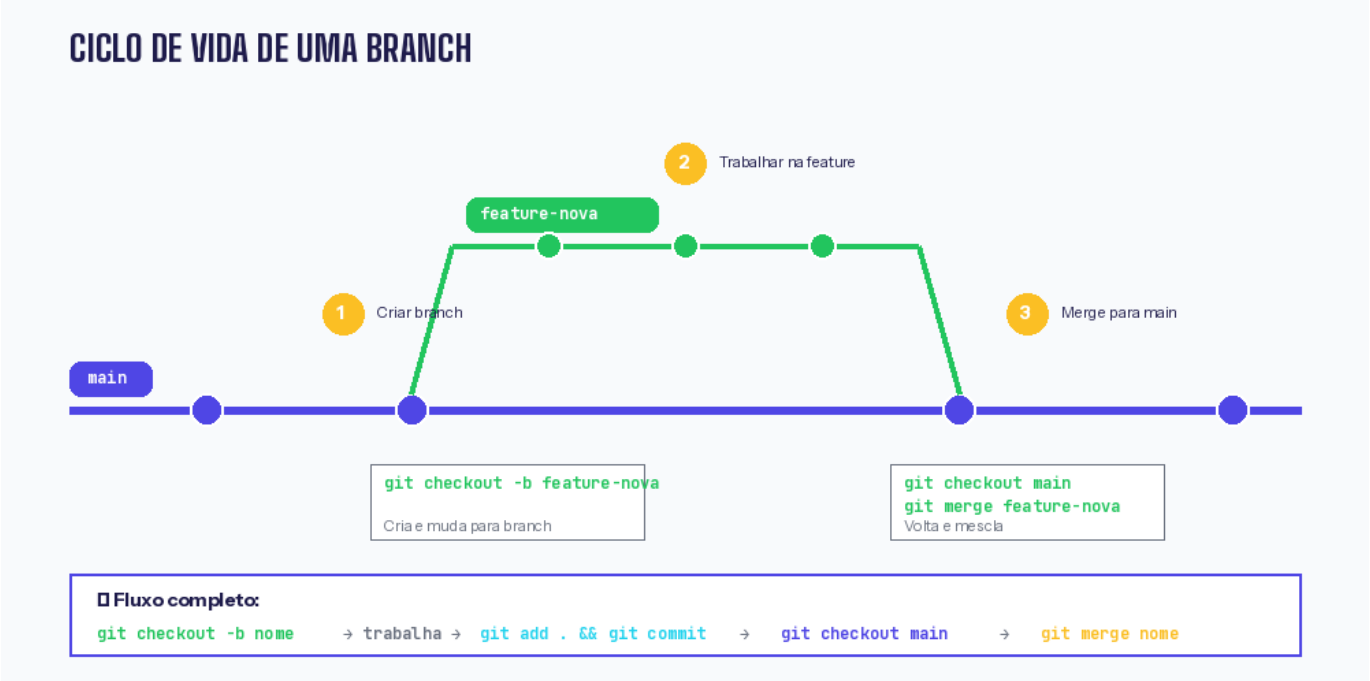
```
git push
```

Pronto! Não precisa mais do `-u origin main`, só `git push`.

---

### 🌿 Trabalhando com Branches

Branches permitem trabalhar em funcionalidades sem afetar o código principal.




CRIAR UMA BRANCH

```
# Criar e mudar para nova branch
git checkout -b pagina-contato
```

**Saída:**

```
Switched to a new branch 'pagina-contato'
```

VER BRANCHES EXISTENTES

```
git branch
```

**Saída:**

```
  main
* pagina-contato
```

O `*` indica em qual branch você está.

FAZER ALTERAÇÕES NA BRANCH

```
# Criar arquivo de contato
echo "# Contato" > contato.md
echo "" >> contato.md
echo "- Email: seu.email@exemplo.com" >> contato.md
```

```
echo "- GitHub: github.com/seu_usuario" >> contato.md
echo "- LinkedIn: linkedin.com/in/seu_usuario" >> contato.md

# Adicionar e commitar
git add contato.md
git commit -m "Adiciona página de contato"
```

VOLTAR PARA MAIN

```
git checkout main
```

**Verifique:**

```
ls
```

O arquivo `contato.md` sumiu! Ele só existe na branch `pagina-contato`.

MESCLAR (MERGE) A BRANCH

```
# Estando na main, trazer as mudanças da outra branch
git merge pagina-contato
```

**Saída:**

```
Updating abc1234..def5678
Fast-forward
 contato.md | 5 +++++
 1 file changed, 5 insertions(+)
 create mode 100644 contato.md
```

Agora o `contato.md` está na main!

DELETAR A BRANCH (OPCIONAL)

```
git branch -d pagina-contato
```

ENVIAR TUDO PARA O GITHUB

```
git push
```

---

### 📥 Baixando Atualizações (Pull)

Se você ou outra pessoa fez alterações pelo GitHub, baixe assim:

```
git pull
```

Isso baixa e mescla automaticamente.

---

### 🔁 Resumo: Fluxo de Trabalho Diário

```
# 1. Verificar status
git status

# 2. Adicionar arquivos modificados
git add .

# 3. Criar commit
git commit -m "Descrição clara do que foi feito"

# 4. Enviar para o GitHub
git push
```

---

### 📋 Comandos Completos


CONFIGURAÇÃO

| Comando | Descrição |
|---|---|
| `git config --global user.name "Nome"` | Define seu nome |
| `git config --global user.email "email"` | Define seu email |
| `git config --list` | Lista configurações |

BÁSICO

| Comando | Descrição |
|---|---|
| `git init` | Inicia repositório |
| `git status` | Mostra status dos arquivos |
| `git add arquivo` | Adiciona arquivo ao stage |
| `git add .` | Adiciona todos os arquivos |
| `git commit -m "msg"` | Cria commit |
| `git log` | Mostra histórico |
| `git log --oneline` | Histórico resumido |

REMOTO (GITHUB)

| Comando | Descrição |
|---|---|
| `git remote add origin URL` | Conecta ao GitHub |
| `git remote -v` | Lista conexões |
| `git push` | Envia para GitHub |
| `git push -u origin main` | Primeiro push |
| `git pull` | Baixa do GitHub |
| `git clone URL` | Clona repositório |

BRANCHES

| Comando | Descrição |
|---|---|
| `git branch` | Lista branches |
| `git branch nome` | Cria branch |
| `git checkout nome` | Muda de branch |
| `git checkout -b nome` | Cria e muda |
| `git merge nome` | Mescla branch |
| `git branch -d nome` | Deleta branch |

---

### 🎬 Vídeos Recomendados

| Vídeo | Canal | Link |
|---|---|---|
| Git e Github com o VSCode no GNU/Linux | Bora para Prática | YouTube |

---

### 🛠️ Exercício Prático

**Faça sozinho:**

1. Crie uma pasta `exercicio-git`
2. Inicialize o Git
3. Crie um arquivo `ola.txt` com seu nome
4. Faça commit
5. Crie uma branch `feature-idade`
6. Adicione sua idade ao arquivo
7. Faça commit na branch
8. Volte para main
9. Faça merge
10. Crie repositório no GitHub e envie

**Comandos que você vai usar:**

```
mkdir exercicio-git
cd exercicio-git
git init
echo "Meu nome é ..." > ola.txt
```

```
git add .
git commit -m "Primeiro commit"
git checkout -b feature-idade
echo "Tenho XX anos" >> ola.txt
git add .
git commit -m "Adiciona idade"
git checkout main
git merge feature-idade
git remote add origin https://github.com/SEU_USUARIO/exercicio-git.git
git push -u origin main
```

---

### ✅ Checklist da Aula

- [ ] Sei abrir o terminal no meu sistema
- [ ] Configurei nome e email no Git
- [ ] Criei um repositório do zero com `git init`
- [ ] Fiz commits pelo terminal
- [ ] Conectei ao GitHub com `git remote`
- [ ] Enviei código com `git push`
- [ ] Criei e mesclei branches
- [ ] Entendo o fluxo: add → commit → push

---

### 🆘 Problemas Comuns

#### "FATAL: NOT A GIT REPOSITORY"

Você não está numa pasta com Git. Execute `git init` ou entre na pasta certa.

#### "ERROR: FAILED TO PUSH SOME REFS"

O GitHub tem commits que você não tem. Execute:

```
git pull --rebase
git push
```

#### "AUTHENTICATION FAILED"

Token errado ou expirado. Crie um novo token.

#### "PERMISSION DENIED"

No Linux, pode ser problema de permissão:

```
sudo chown -R $USER:$USER .git
```

---

> **Dúvidas?**
>
> Pratique os comandos! A melhor forma de aprender é fazendo. Se travar, me chama no grupo!

2026-02-19

2026-02-17

## 2.2.4 Aula — Arquitetura de Hardware: Conhecendo o Computador por Dentro

**Informações da Aula**

**Disciplina:** Sistemas Operacionais
**Data:** Fevereiro de 2026
**Tipo:** Aula Prática
**Professor:** Time Robótica
**Tema:** Desmontagem, identificação de componentes e remontagem de CPU

---

### 🎯 Objetivo da Aula

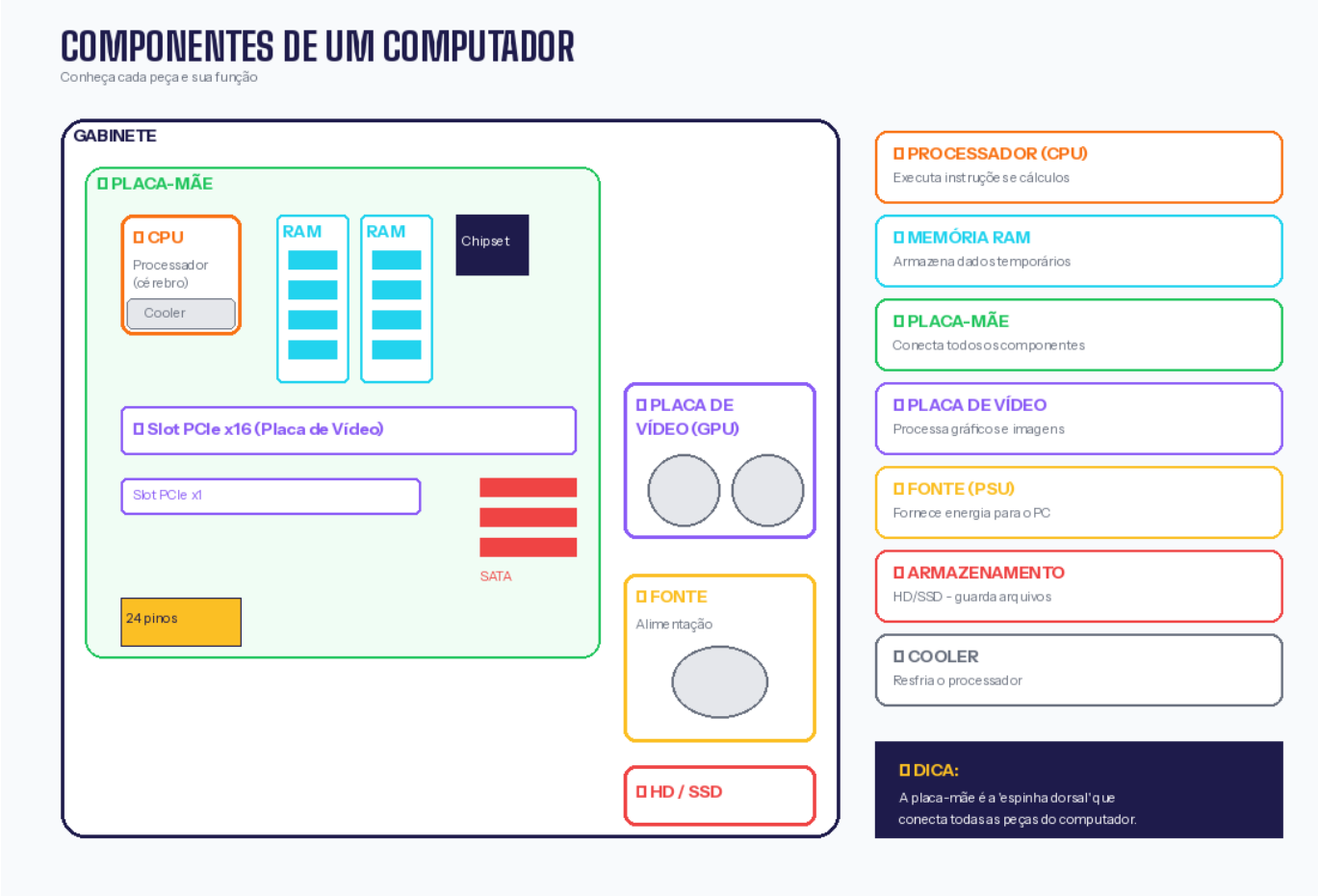
Nesta aula prática, você aprendeu a:

- ✅ Identificar os principais componentes de um computador
- ✅ Entender a função de cada peça
- ✅ Desmontar e remontar um PC com segurança
- ✅ Conectar corretamente os cabos para gerar vídeo
- ✅ Compreender o papel do Kernel no sistema

---

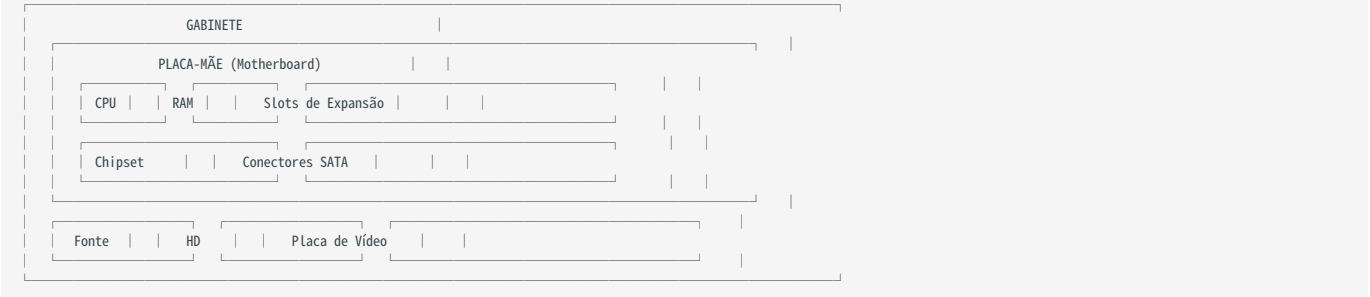
### 🖥️ Anatomia de um Computador

Antes de colocar a mão na massa, vamos entender o que tem dentro de um computador:




VISÃO GERAL

Um computador é composto por **hardware** (parte física) e **software** (programas). Nesta aula, focamos no hardware.




---

## 🧩 Componentes Principais

1. 🧠 PROCESSADOR (CPU)

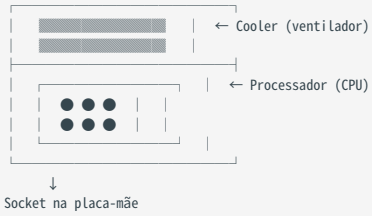
**O que é:** O "cérebro" do computador. Executa todas as instruções e cálculos.

**Onde fica:** Encaixado no socket da placa-mãe, coberto pelo cooler.

**Como identificar:** Chip quadrado com muitos pinos embaixo (ou contatos dourados).

| Característica | Descrição |
|---|---|
| **Função** | Processar instruções e fazer cálculos |
| **Marcas comuns** | Intel (Core i3, i5, i7, i9) e AMD (Ryzen) |
| **Medida de velocidade** | GHz (Gigahertz) |
| **Núcleos** | Quanto mais, melhor para multitarefas |

> ⚠️ **Cuidado!**
>
> • Nunca toque nos pinos/contatos do processador
> • O processador esquenta MUITO — sempre precisa de cooler
> • A pasta térmica é essencial para dissipar o calor




---

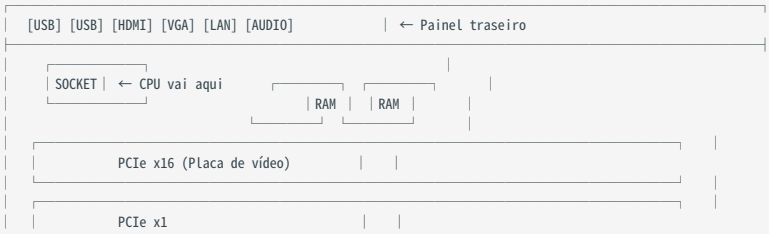
#### 2. 🎛️ PLACA-MÃE (MOTHERBOARD)

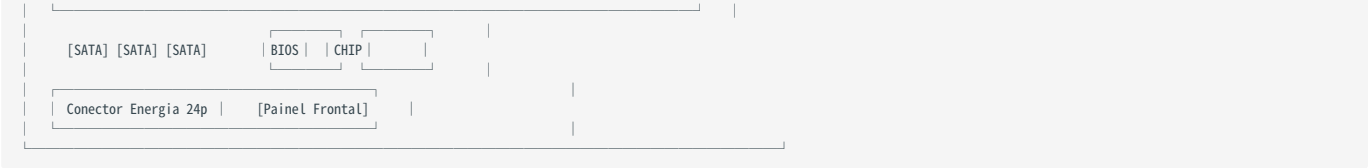
**O que é:** A "espinha dorsal" do computador. Conecta todos os componentes.

**Onde fica:** Presa ao gabinete com parafusos.

**Como identificar:** Maior placa verde/preta dentro do gabinete.

| Componente na placa | Função |
|---|---|
| **Socket** | Encaixe do processador |
| **Slots de RAM** | Encaixe das memórias |
| **Slots PCIe** | Placa de vídeo e outras placas |
| **Conectores SATA** | Ligar HD/SSD |
| **Conectores de energia** | Receber energia da fonte |
| **Portas traseiras** | USB, HDMI, VGA, Ethernet, áudio |
| **Chipset** | Controla a comunicação entre componentes |
| **BIOS/UEFI** | Firmware que inicia o computador |






---

#### 3. 💾 MEMÓRIA RAM

**O que é:** Memória temporária (volátil). Guarda dados enquanto o PC está ligado.

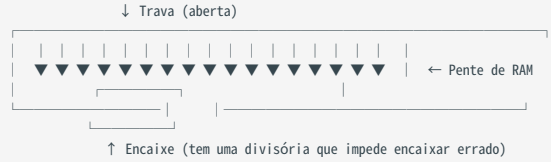
**Onde fica:** Slots alongados próximos ao processador.

**Como identificar:** Pentes retangulares finos.

| Característica | Descrição |
|---|---|
| **Função** | Armazenar dados em uso no momento |
| **Volátil** | Perde tudo quando desliga |
| **Capacidade comum** | 4GB, 8GB, 16GB, 32GB |
| **Tipos** | DDR3, DDR4, DDR5 (não são compatíveis entre si!) |

> 🔥 **Analogia**
>
> • **RAM** = Mesa de trabalho (espaço para trabalhar agora)
> • **HD/SSD** = Armário (guarda arquivos permanentemente)
>
> Quanto maior a mesa (RAM), mais coisas você pode fazer ao mesmo tempo!

**Como encaixar:**




---

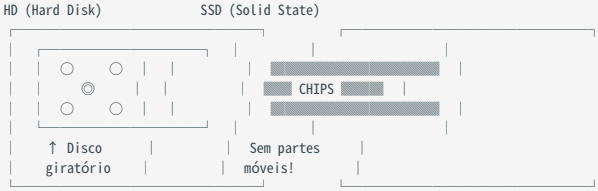
#### 4. 💿 ARMAZENAMENTO (HD E SSD)

**O que é:** Onde ficam guardados seus arquivos, sistema operacional e programas.

**Tipos:**

| Tipo | Tecnologia | Velocidade | Durabilidade |
|---|---|---|---|
| **HD (Hard Disk)** | Disco magnético giratório | Lento | Frágil (tem peças móveis) |
| **SSD (Solid State)** | Memória flash (chips) | Muito rápido | Resistente |
| **NVMe** | SSD conectado direto na placa-mãe | Ultra rápido | Resistente |

> **Dica de upgrade**
>
> Trocar o HD por um SSD é a melhor melhoria custo-benefício! O PC fica MUITO mais rápido.




---

#### 5. ⚡ FONTE DE ALIMENTAÇÃO (PSU)

**O que é:** Converte a energia da tomada (AC) para energia que o PC usa (DC).

**Onde fica:** Geralmente na parte superior ou inferior traseira do gabinete.

**Como identificar:** Caixa de metal com ventilador e muitos cabos coloridos.

| Conector | Função |
|---|---|
| **24 pinos** | Energia principal da placa-mãe |
| **4/8 pinos** | Energia do processador |
| **6/8 pinos PCIe** | Energia da placa de vídeo |
| **SATA** | Energia para HD/SSD |
| **Molex** | Energia para ventoinhas e acessórios |

> **Segurança**
>
> • SEMPRE desligue da tomada antes de mexer na fonte
> • Nunca abra a fonte — ela armazena energia mesmo desligada
> • Use fonte de qualidade — fonte ruim pode queimar o PC inteiro

---

#### 6. 🎮 PLACA DE VÍDEO (GPU)

**O que é:** Processa gráficos e imagens. Essencial para jogos e edição de vídeo.

**Onde fica:** Slot PCIe x16 da placa-mãe (o maior slot).

**Como identificar:** Placa grande com ventiladores, saídas de vídeo (HDMI, DisplayPort).

| Tipo | Descrição |
|---|---|
| **Integrada** | Embutida no processador. Básica. |
| **Dedicada** | Placa separada. Muito mais potente. |

**Marcas:** NVIDIA (GeForce) e AMD (Radeon)

**Vídeo integrado vs dedicado**

- **Integrado:** Suficiente para uso básico, Office, vídeos
- **Dedicado:** Necessário para jogos, 3D, edição de vídeo

---

#### 7. 🌀 SISTEMA DE REFRIGERAÇÃO

**O que é:** Mantém os componentes em temperatura segura.

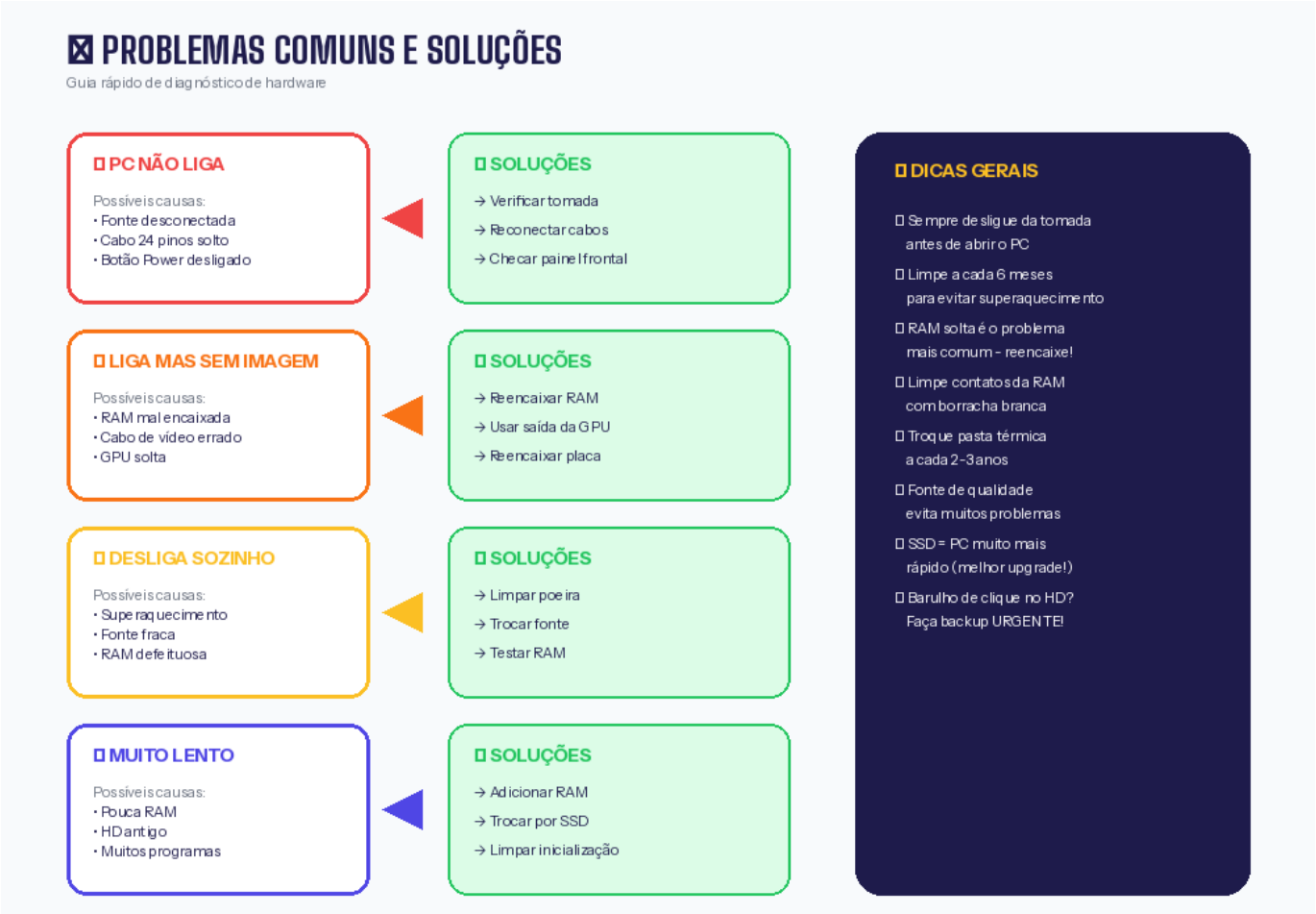
**Tipos:**

| Tipo | Descrição |
|---|---|
| **Cooler de CPU** | Ventilador + dissipador sobre o processador |
| **Fans do gabinete** | Ventoinhas que circulam ar |
| **Water Cooler** | Sistema de refrigeração a água (avançado) |

**Pasta térmica:** Gel que melhora a transferência de calor entre CPU e cooler. Precisa trocar a cada 2-3 anos.

---

### 🔧 Problemas Comuns e Soluções



❌ PC NÃO LIGA

| Possível causa | Solução |
|---|---|
| Fonte não conectada na tomada | Verificar tomada e chave da fonte (I/O) |
| Cabo de energia da placa-mãe solto | Reconectar o cabo de 24 pinos |
| Botão Power desconectado | Verificar fios do painel frontal |
| Fonte queimada | Testar com outra fonte |
| RAM mal encaixada | Remover e recolocar a RAM |

**Teste básico:** Se ao ligar as ventoinhas giram, a fonte está ok.

---

❌ PC LIGA MAS NÃO EXIBE IMAGEM

| Possível causa | Solução |
|---|---|
| Monitor no cabo errado | Conectar na placa de vídeo (não na placa-mãe) |
| RAM mal encaixada | Tirar e recolocar a RAM com firmeza |
| Placa de vídeo mal encaixada | Reencaixar a placa de vídeo |
| Cabo de vídeo com defeito | Testar outro cabo HDMI/VGA |
| Monitor desligado/entrada errada | Verificar botão e input do monitor |

🔥 **Dica de ouro**

Quando não aparecer vídeo, **tire a RAM, limpe os contatos** com borracha branca e recoloque. Resolve 70% dos casos!

---

❌ PC DESLIGA SOZINHO / REINICIA

| Possível causa | Solução |
|---|---|
| Superaquecimento | Limpar poeira, trocar pasta térmica |
| Fonte fraca/defeituosa | Trocar a fonte |
| RAM com defeito | Testar um pente de cada vez |
| Sistema operacional corrompido | Reinstalar o SO |

---

❌ PC MUITO LENTO

| Possível causa | Solução |
|---|---|
| Pouca RAM | Adicionar mais memória RAM |
| HD antigo | Trocar HD por SSD |
| Muitos programas iniciando | Desativar programas na inicialização |
| Vírus/malware | Verificar com antivírus |
| Superaquecimento | Limpar e melhorar ventilação |

---

❌ BARULHOS ESTRANHOS

| Barulho | Causa provável | Solução |
|---|---|---|
| Click click | HD com defeito | Fazer backup URGENTE e trocar HD |
| Zumbido alto | Ventoinha suja/com defeito | Limpar ou trocar ventoinha |
| Chiado | Fonte com problema | Trocar fonte |
| Vibração | Parafuso solto ou ventoinha desbalanceada | Apertar parafusos |

---

### 🧹 Limpeza e Conservação

POR QUE LIMPAR?

- Poeira acumulada = superaquecimento
- Superaquecimento = lentidão e travamentos
- Casos extremos = componentes queimados

COM QUE FREQUÊNCIA?

| Ambiente | Frequência |
|---|---|
| Casa limpa, sem pets | A cada 6-12 meses |
| Casa com pets/poeira | A cada 3-6 meses |
| Ambiente industrial | Mensal |

COMO LIMPAR

**Materiais necessários:**

- ✅ Pano de microfibra
- ✅ Pincel antiestático (ou pincel de maquiagem limpo)
- ✅ Ar comprimido (lata ou compressor)
- ✅ Álcool isopropílico (70% ou mais)
- ✅ Borracha branca (para contatos da RAM)

**Passo a passo:**

```
1. DESLIGUE o PC e tire da tomada
2. Aguarde 5 minutos (capacitores descarregarem)
3. Abra o gabinete (geralmente parafusos atrás)
4. Use ar comprimido para soprar a poeira
   - Segure as ventoinhas para não girarem!
5. Use o pincel para poeira mais grudada
6. Limpe os contatos da RAM com borracha branca
7. Se for trocar pasta térmica:
   - Limpe a antiga com álcool isopropílico
   - Aplique quantidade do tamanho de um grão de arroz
8. Feche o gabinete
```

**NUNCA faça isso!**

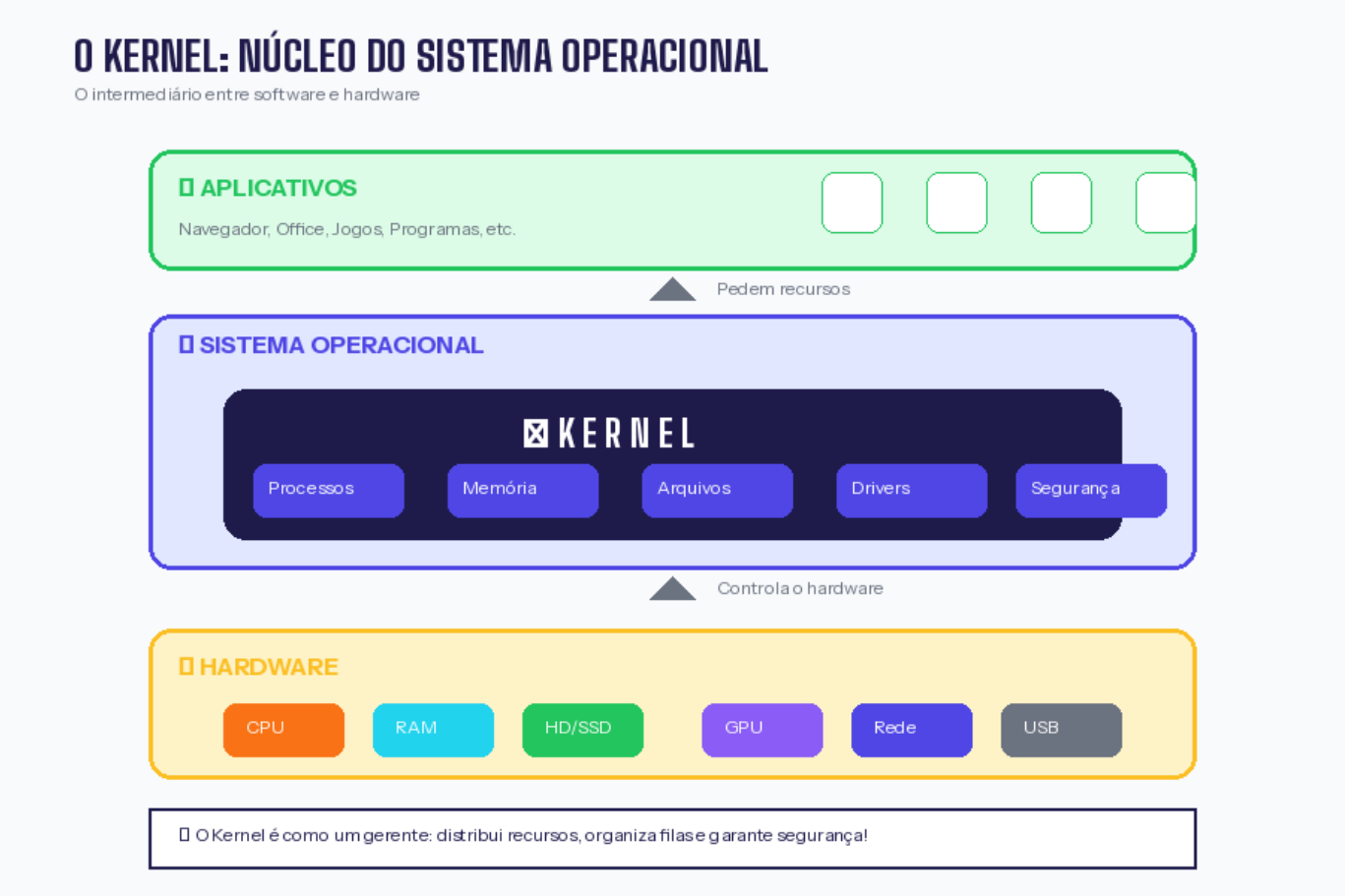
- ❌ Usar aspirador de pó (gera estática)
- ❌ Usar pano molhado com água
- ❌ Soprar com a boca (umidade!)
- ❌ Limpar com o PC ligado

---

### 🧠 O Kernel: O Coração do Sistema Operacional

O QUE É O KERNEL?

O **Kernel** é o núcleo do sistema operacional. Ele é o intermediário entre o **hardware** (peças físicas) e o **software** (programas).




```
┌─────────────────────────────────────────┐
│              APLICATIVOS                │
│     (Navegador, Word, Jogos, etc.)      │
├─────────────────────────────────────────┤
│           SISTEMA OPERACIONAL           │
│   ┌─────────────────────────────────┐   │
│   │          K E R N E L            │   │
│   │  (Gerencia tudo que acontece embaixo)  │   │
│   └─────────────────────────────────┘   │
├─────────────────────────────────────────┤
│              HARDWARE                   │
│     (CPU, RAM, HD, Placa de Vídeo, etc.)  │
└─────────────────────────────────────────┘
```

FUNÇÕES DO KERNEL

| Função | Descrição |
|---|---|
| **Gerenciar processos** | Decide qual programa usa a CPU e quando |
| **Gerenciar memória** | Distribui RAM entre os programas |
| **Gerenciar dispositivos** | Comunica com hardware (drivers) |
| **Gerenciar arquivos** | Organiza dados no HD/SSD |
| **Segurança** | Impede que programas acessem áreas proibidas |

#### ANALOGIA: O KERNEL É COMO UM GERENTE DE HOTEL

```
🏨 HOTEL = COMPUTADOR

👔 Gerente (Kernel):
   - Distribui quartos (memória) para hóspedes (programas)
   - Organiza quem usa o elevador (CPU) e quando
   - Cuida da segurança (não deixa hóspede entrar no quarto errado)
   - Controla os funcionários (drivers de hardware)

🧳 Hóspedes (Aplicativos):
   - Querem usar recursos do hotel
   - Precisam pedir ao gerente
   - Não podem fazer o que quiserem

🔧 Funcionários (Drivers):
   - Fazem o trabalho pesado
   - Recebem ordens do gerente
```

#### TIPOS DE KERNEL

| Tipo | Descrição | Exemplo |
|---|---|---|
| **Monolítico** | Tudo junto no kernel (mais rápido) | Linux |
| **Microkernel** | Mínimo no kernel, resto fora (mais seguro) | MINIX |
| **Híbrido** | Mistura dos dois | Windows, macOS |

#### KERNEL NO LINUX

No Linux, o kernel é literalmente chamado **Linux** (criado por Linus Torvalds em 1991).

```
# Ver versão do kernel no Linux
uname -r

# Saída exemplo:
5.15.0-92-generic
```

O "Linux" que usamos (Ubuntu, Mint) é na verdade:

```
Ubuntu = Kernel Linux + Programas GNU + Interface Gráfica + Aplicativos
```

Por isso muitos chamam de **GNU/Linux**.

---

### 🔌 Conexões: Montando o PC

#### ORDEM RECOMENDADA DE MONTAGEM

```
1. Instalar CPU na placa-mãe
2. Instalar cooler da CPU (com pasta térmica)
3. Instalar memória RAM
4. Instalar placa-mãe no gabinete
5. Conectar fonte de alimentação
6. Instalar HD/SSD
7. Instalar placa de vídeo
8. Conectar cabos do painel frontal
9. Conectar cabos de energia
10. Conectar cabos de dados (SATA)
11. Fechar e testar
```

CABOS ESSENCIAIS

| Cabo | De → Para | Obrigatório? |
|---|---|---|
| 24 pinos | Fonte → Placa-mãe | ✅ Sim |
| 4/8 pinos CPU | Fonte → Placa-mãe (perto da CPU) | ✅ Sim |
| SATA dados | HD/SSD → Placa-mãe | ✅ Sim |
| SATA energia | Fonte → HD/SSD | ✅ Sim |
| 6/8 pinos PCIe | Fonte → Placa de vídeo | Se tiver placa dedicada |
| Painel frontal | Gabinete → Placa-mãe | ✅ Para botão ligar |
| HDMI/VGA | Placa de vídeo → Monitor | ✅ Sim |

---

### 📋 Checklist de Montagem

Antes de ligar, verifique:

- [ ] CPU encaixada corretamente (seta alinhada)
- [ ] Cooler bem preso e com pasta térmica
- [ ] RAM encaixada até ouvir "click"
- [ ] Cabo 24 pinos conectado na placa-mãe
- [ ] Cabo 4/8 pinos da CPU conectado
- [ ] HD/SSD com cabo de dados E energia
- [ ] Placa de vídeo bem encaixada no slot
- [ ] Cabos do painel frontal conectados
- [ ] Nenhum parafuso solto dentro do gabinete
- [ ] Chave da fonte na posição "I" (ligado)

---

### 🎬 Vídeos Recomendados

| Vídeo | Canal | Descrição |
|---|---|---|
| Como montar um PC | Adrenaline | Tutorial completo de montagem |
| Entenda cada peça do PC | Diolinux | Explicação de componentes |
| O que é o Kernel | ByteMonk | Explicação técnica acessível |
| Limpeza de PC | Peperaio Hardware | Tutorial de limpeza |

---

### ✅ Checklist da Aula

- [ ] Sei identificar os principais componentes de um PC
- [ ] Entendo a função de cada componente
- [ ] Sei desmontar e remontar um computador básico
- [ ] Conheço problemas comuns e suas soluções
- [ ] Sei como fazer limpeza básica
- [ ] Entendo o que é o Kernel e sua função

### 📝 Glossário

| Termo | Significado |
|---|---|
| **CPU** | Central Processing Unit (processador) |
| **GPU** | Graphics Processing Unit (processador gráfico) |
| **RAM** | Random Access Memory (memória volátil) |
| **HD** | Hard Disk (disco rígido) |
| **SSD** | Solid State Drive (disco de estado sólido) |
| **PSU** | Power Supply Unit (fonte de alimentação) |
| **BIOS** | Basic Input/Output System (firmware básico) |
| **UEFI** | Unified Extensible Firmware Interface (BIOS moderno) |
| **Socket** | Encaixe do processador na placa-mãe |
| **Chipset** | Conjunto de chips que controla comunicação |
| **Driver** | Programa que permite o SO comunicar com hardware |
| **Kernel** | Núcleo do sistema operacional |

**Dúvidas?**

Hardware parece complicado no início, mas com prática fica natural! Qualquer dúvida, chama no grupo! 🔧

2026-02-18

2026-02-17

# 3. 🧠 Materiais

## 3.1 🧠 Materiais de Estudo

Materiais prontos para usar, estudar e imprimir.

### 3.1.1 📋 Cheatsheets

Resumos de uma página com os comandos e conceitos mais importantes.

| Material | Descrição | Download |
|---|---|---|
| Comandos Linux | Terminal essencial | PDF |
| HTML & CSS | Tags e propriedades | PDF |
| Python Básico | Sintaxe e estruturas | PDF |
| Git & GitHub | Comandos de versionamento | PDF |
| Sql | SELECT, INSERT, UPDATE, DELETE, JOIN, funções | PDF |
| VsCode | Atalhos + Extensões | PDF |
| Markdown | Títulos, listas, links, tabelas, código | PDF |

### 3.1.2 📝 Templates

Templates prontos para ajudar nos seus estudos.

| Material | Descrição | Download |
|---|---|---|
| Método Cornell | Templates para anotações de aula | PDF |

### 3.1.3 💡 Como usar

1. Clique no link **PDF** para baixar
2. Imprima ou use no tablet
3. Os cheatsheets cabem em 1 página A4

### 3.1.4 🆕 Sugerir Material

Sentiu falta de algo? Me avisa no grupo da turma ou abre uma issue no GitHub!

2026-02-19

2026-02-17

# 4. 🎬 Entretenimento

## 4.1 Filmes, Séries e Canais

```
# 🎬 Entretenimento Dev
```

Filmes, séries, documentários e canais para quem curte tecnologia.

---

### 4.1.1 🎥 Filmes

**Obrigatórios**

| Filme | Ano | Sobre o que é |
|---|---|---|
| **A Rede Social** | 2010 | A criação do Facebook por Mark Zuckerberg |
| **O Jogo da Imitação** | 2014 | Alan Turing e a máquina que quebrou a Enigma |
| **Os Piratas do Vale do Silício** | 1999 | A rivalidade entre Steve Jobs e Bill Gates |
| **Jobs** | 2013 | A história de Steve Jobs |
| **O Quinto Poder** | 2013 | A história do WikiLeaks e Julian Assange. **Um dos meus Filmes favoritos** |

**Ficção Científica Tech**

| Filme | Ano | Sobre o que é |
|---|---|---|
| **Her** | 2013 | Um homem se apaixona por uma IA |
| **Ex Machina** | 2015 | Teste de Turing com uma androide |
| **Matrix** | 1999 | Clássico sobre realidade simulada |
| **Blade Runner 2049** | 2017 | Androides e o que significa ser humano |
| **O Homem Bicentenário** | 1999 | Um robô que quer se tornar humano |
| **Eu, Robô** | 2004 | IA e as três leis da robótica |
| **Tron: O Legado** | 2010 | Dentro de um mundo digital |
| **O Exterminador do Futuro** | 1984 | IA que domina o mundo |

**Hackers e Cibercultura**

| Filme | Ano | Sobre o que é |
|---|---|---|
| **Hackers** | 1995 | Clássico dos anos 90 sobre hackers |
| **WarGames** | 1983 | Adolescente quase começa Guerra Nuclear |
| **Snowden** | 2016 | A história de Edward Snowden |
| **Who Am I** | 2014 | Grupo de hackers alemães (muito bom!) |
| **O Código** | 2001 | Documentário sobre Linux e open source. **Maravilhoso** |

---

### 4.1.2 📺 Séries

**Essenciais**

| Série | Onde assistir | Sobre o que é |
|---|---|---|
| **Mr. Robot** | Prime Video | Hacker com transtornos psicológicos. Técnica REAL! |
| **Silicon Valley** | HBO Max | Comédia sobre startups no Vale do Silício |
| **Halt and Catch Fire** | - | Anos 80, a corrida dos PCs e da internet |
| **The IT Crowd** | Netflix | Comédia britânica sobre o suporte de TI |

### 4.1.3 🎞️ Documentários

| Documentário | Ano | Tema |
|---|---|---|
| **O Dilema das Redes** | 2020 | Como redes sociais manipulam você |
| **The Great Hack** | 2019 | Escândalo Cambridge Analytica |
| **Lo and Behold** | 2016 | A história e futuro da internet |
| **O Código** | 2001 | Linux, open source e Linus Torvalds |
| **Revolution OS** | 2001 | História do movimento open source |
| **TPB AFK** | 2013 | The Pirate Bay e seus fundadores |
| **Zero Days** | 2016 | Stuxnet e guerra cibernética |
| **AlphaGo** | 2017 | IA do Google que venceu o melhor jogador de Go |
| **Inside Bill's Brain** | 2019 | A mente de Bill Gates |

### 4.1.4 📺 Canais do YouTube

**Brasileiros**

| Canal | Conteúdo |
|---|---|
| **Fabio Akita** | Carreira, história da computação, programação |
| **Código Fonte TV** | Notícias tech, tutoriais, entrevistas |
| **Yudi Ganeko** | Vlog da vida de um Programador |
| **Diolinux** | Linux, open source, tutoriais |
| **Filipe Deschamps** | Programação, carreira, notícias |
| **Rafaella Ballerini** | Tutoriais para iniciantes |
| **Rocketseat** | Tutoriais React, Node, mobile |
| **Curso em Vídeo** | Cursos completos gratuitos |
| **Programador BR** | Carreira, dicas, mercado de trabalho |
| **Lucas Montano** | Carreira, entrevistas, mercado |
| **Attekita Dev** | Dev frontend, carreira |

**Gringos**

| Canal | Conteúdo |
|---|---|
| **Fireship** | Vídeos curtos e diretos sobre tech |
| **Traversy Media** | Tutoriais web completos |
| **The Coding Train** | Programação criativa, divertido |
| **CS Dojo** | Algoritmos e estruturas de dados |
| **Tech With Tim** | Python, machine learning |
| **Computerphile** | Ciência da computação explicada |
| **Numberphile** | Matemática interessante |
| **3Blue1Brown** | Matemática visual incrível |

### 4.1.5 🎧 Podcasts

| Podcast | Tema |
|---|---|
| **Hipsters.tech** | Tecnologia, carreira, tendências |
| **Syntax** | Web development |

### 4.1.6 📚 Bônus: Livros de Ficção e Não Ficção

Como leitor voraz, não poderia deixar de indicar alguns livros sensacionais.

| Livro | Autor | Sobre |
|---|---|---|
| **Neuromancer** | William Gibson | O livro que criou o cyberpunk |
| **O Guia do Mochileiro das Galáxias** | Douglas Adams | Ficção científica cômica |
| **Eu, Robô** | Isaac Asimov | Contos sobre robôs e IA |
| **O Marciano** | Andy Weir | Sobrevivência em Marte com ciência real |
| **Cypherpunks** | Julian Assange | Liberdade nas redes |
| **Eterna Vigilância** | Edward Snowden | Como montei e desvendei o maior sistema de espionagem do mundo |

### 4.1.7 🏁 Por onde começar?

**Se tem pouco tempo:**

1. Assista **A Rede Social** (filme)
2. Comece **Mr. Robot** (série)
3. Inscreva-se no **Fireship** (YouTube)

**Se quer se aprofundar:**

1. Veja **O Dilema das Redes** (documentário)
2. Maratone **Silicon Valley** (série)
3. Ouça **Hipsters.tech** (podcast)

---

> **Dica**
>
> Assistir conteúdo em inglês ajuda MUITO na carreira de dev. Comece com legendas em português, depois em inglês, depois sem legendas!

---

Conhece algo legal que não está na lista? Me avisa no grupo! 🎬

2026-02-23

2026-02-19