AULA 1 · MATERIAL DO ALUNO

Ferramentas, contas e *setup*

Siga na ordem, de cima pra baixo. No fim você tem o ambiente pronto pra construir o **PostZap** — o SaaS de gestão de conteúdo: Kanban, aprovação no Telegram + link do cliente, e publicação automática.

Cada ferramenta tem 3 coisas: **o que é**, **como instalar** e **como conferir** se deu certo.

01 AS 4 FERRAMENTAS BASE

Instale nesta ordem

1 Node.js

O motor que roda tudo.

INSTALAR — BAIXE A VERSÃO LTS EM NODEJS.ORG, OU:

```
# Windows
winget install OpenJS.NodeJS.LTS
# Mac
brew install node
```

✓ conferir: `node -v` responde um número (ex. v20.x)

2 Git

Guarda o histórico do código e sobe pro GitHub.

INSTALAR — GIT-SCM.COM, OU:

```
# Windows
winget install Git.Git
# Mac
brew install git
```

✓ conferir: `git --version`

3 Cursor

O editor onde a gente trabalha (com IA embutida).

INSTALAR

```
# baixe em cursor.com, instale e faça login
```

✓ conferir: o Cursor abre e você está logado

4 Claude Code

O agente que constrói o SaaS com você.

INSTALAR (NO TERMINAL)

```
npm install -g @anthropic-ai/claude-code
```

ENTRAR

```
# digite claude e faça login quando pedir
claude
```

✓ conferir: `claude --version`

Fim da Parte 1: rode `node -v`, `git --version` e `claude --version`. Se os três respondem, a base está pronta.

Ponytail & Playwright

5 Ponytail — a IA escreve o mínimo de código

Menos código = menos bug, menos custo, mais rápido. É um plugin do Claude Code.

> ⚠ **Atenção — é aqui que todo mundo erra:** são **dois comandos**, e cada um vai numa **mensagem separada** dentro do Claude Code. Não cole os dois juntos.

1 abra o Claude Code (digite `claude`) e envie só isto:

```
/plugin marketplace add DietrichGebert/ponytail
```

2 espere responder e, em outra mensagem, envie só isto:

```
/plugin install ponytail@ponytail
```

✓ conferir: rode `/plugin` e veja o `ponytail` na lista

> 💡 Pelo app do Claude Code (desktop): botão **+** ao lado da caixa de prompt → **Plugins** → **Add plugin** → escolha o ponytail.

6 Playwright — a IA testa no navegador e se corrige

Você instala dentro do projeto (nas próximas aulas). Guarde os comandos:

```
npm i -D @playwright/test
npx playwright install
```

> As CLIs do Supabase, Vercel e GitHub a gente instala quando precisar (aulas de banco e deploy). Não precisa agora.

Monte o "cofre" antes de codar

Crie **todas** as contas e junte **todos** os tokens ANTES de programar. Assim a IA não trava no meio pedindo senha.

Supabase — banco, login e storage (grátis)

supabase.com → login → New project → nome `postzap` → guarde a senha do banco → Create. Depois vá em Project Settings → API e copie:

NO SUPABASE	VIRA NO .ENV
Project URL	`NEXT_PUBLIC_SUPABASE_URL`
anon public	`NEXT_PUBLIC_SUPABASE_ANON_KEY`
service_role (secreta!)	`SUPABASE_SERVICE_ROLE_KEY`

Upload-Post — publica nas redes

app.upload-post.com → entre com e-mail ou Google → copie a API Key → `UPLOAD_POST_API_KEY`.

Telegram — a aprovação interna da equipe (grátis)

1. Fale com o **@BotFather** no Telegram.
2. Envie `/newbot` → escolha um nome → um @username terminando em `bot`.
3. Copie o **token** → `TELEGRAM_BOT_TOKEN`.

CRON_SECRET — uma senha forte (você gera)

```
node -e "console.log(crypto.randomUUID())"
```

Copie o valor → `CRON_SECRET`.

04 O ARQUIVO .ENV.LOCAL

Cole suas chaves aqui

Crie um arquivo `.env.local` na raiz do projeto e preencha com os valores que copiou:

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=
NEXT_PUBLIC_SUPABASE_ANON_KEY=
SUPABASE_SERVICE_ROLE_KEY=        # secreta (só no servidor)
# Agendador
CRON_SECRET=
# Upload-Post
UPLOAD_POST_API_KEY=
# Telegram
TELEGRAM_BOT_TOKEN=
# Opcionais (mock se vazio): e-mail e WhatsApp
RESEND_API_KEY=
N8N_NOTIFICACAO_WEBHOOK=
```

🔒 As 4 regras de ouro pra nunca vazar segredo

1. **O .env.local NUNCA vai pro GitHub** — confira que o `.gitignore` tem `.env*`.
2. **Segredo nunca leva NEXT_PUBLIC_** — só a URL e a anon do Supabase podem ser públicas.
3. **O que vai pro repositório é o .env.example** — as mesmas chaves, sem os valores.
4. **Em produção, os segredos vão nas variáveis do Vercel** (nas próximas aulas).

05 A GUARDRAIL

O CLAUDE.md inicial

O arquivo de regras que segura a IA na linha. Crie `CLAUDE.md` na raiz do projeto e cole:

```
# CLAUDE.md — Regras do projeto (PostZap)

## Stack fixa (não trocar)
Next.js + React + TypeScript + Tailwind + Supabase (Postgres + Auth + RLS) + Verc

## Regras que valem sempre
- Multi-tenant: cada cliente só vê os dados dele (org_id em toda tabela e query,
- Segredos só no servidor. NUNCA usar NEXT_PUBLIC_ em chave ou token secreto.
- Nunca imprimir, logar ou commitar segredo. Nunca commitar o .env.
- Pensar antes de codar: o MENOR código que resolve (sem overengineering).
```

✓ ANTES DE IR PRA AULA 2

Checklist

- [ ] `node -v`, `git --version` e `claude --version` respondem
- [ ] Ponytail instalado (aparece em `/plugin`)
- [ ] Conta Supabase criada + os 3 valores copiados
- [ ] Upload-Post criado + API Key copiada
- [ ] Bot do Telegram criado + token copiado
- [ ] `CRON_SECRET` gerado
- [ ] `.env.local` montado + `.gitignore` com `.env*`
- [ ] `CLAUDE.md` colado na raiz

Próxima aula: a gente gera as 14 fases do projeto com um prompt só e começa a construir. 🚀