

BIUD Talks



Claude Code: nível avançado

Thiago Prado - Dev. FullStack

Cronograma do Programa

22/04	Wanderson	Inteligência Artificial para Dev
29/04	Praxedes	Testes Unitários
06/05	Luk	Arquitetura de Software
13/05	Regis	Integração de Sistemas
20/05	Rogério	Root Cause Analysis (RCA)
⚡ 27/05	Gustavo	Metodologia Ágil
⚡ 03/06	Luk	CR – Code Review

Cronograma do Programa

10/06	Thiago	IA na prática: Claude Code
17/06	Guilherme	Segurança de dados no ambiente corporativo e pessoal
24/06	Jean	Machine Learning
01/07	Santiago	CNV - Comunicação não Violenta no ambiente de trabalho
08/07	Rogério	Saúde Mental
15/07	Guilherme	DevOps e Infraestrutura
22/07	Gustavo	Ambidestria Organizacional

Cronograma do Programa

29/07

Santiago

Mindset Ágil

05/08

Jean

RAG

12/08

????

Pode ser você! Envie seu tema para o Rogério.

Estrutura de cada Sessão

Apresentação

20-30 minutos

Conteúdo e expertise do palestrante

Debate & Discussão

30 minutos

Dúvidas, insights e networking

Moderador: Rogério

Responsável por mediar o debate e garantir engajamento de todos os participantes

Os 3 pilares



- **Estratégias** — contexto, plano, paralelismo e spec-driven (GSD)
- **Comandos “/”** — o que existe e os que mais economizam tempo
- **A pasta .claude** — memória e configuração que o time compartilha

Foco: ir além do básico que o time já domina.

Contexto: o que mais pesa



- **Context rot** — a qualidade cai conforme a janela de contexto enche
- **/context** — veja onde o contexto está sendo gasto
- **/compact** — resume e libere espaço (rode por volta de 70%)
- **/clear** — recomece limpo a cada nova tarefa

```
claude - /context

> /context

Context window
[Progress bar: 38% · 76k / 200k]

System prompt      12k
CLAUDE.md + rules  9k
Conversa           46k
Ferramentas        9k

↳ Dica: /compact para liberar espaço
```

Planejar antes de executar



- **/plan** — planejamento local: rápido, no terminal, para tarefas focadas
- **/ultraplan** — planejamento na nuvem: mudanças grandes, terminal livre

O Claude desenha o plano e você aprova antes de qualquer alteração.





Paralelizar: /agents e /batch

- **/agents** — delegue tarefas a subagentes, cada um em contexto isolado
- **/batch** — orquestra uma mudança grande em 5–30 unidades, em paralelo

O /batch roda sobre subagentes: 1 worktree e 1 PR por unidade.



Spec-Driven com GSD



- **GSD (Get Shit Done)** — sistema open-source que roda sobre o Claude Code
- **Spec → planos atômicos** — cada um executado em contexto novo e limpo
- **Subagentes em waves** — paraleliza sozinho (alternativa ao /batch)
- **Fluxo** — /gsd-new-project → discuss → plan → execute

Vence o “context rot”; instala skills em .claude/skills/.

```
claude — /gsd-execute-phase

> /gsd-execute-phase 1

Wave 1 — 3 planos em paralelo

● auth-model          ✓ done          PR #142
● auth-api             ⚙ running
● auth-tests           ⚙ running

↳ Cada plano roda em um contexto novo e limpo.
```



Revisão e segurança

- **/diff** — veja exatamente o que mudou antes de commitar
- **/code-review** — revisa o diff; --fix aplica as correções
- **/security-review** — caça injeção, auth e exposição de dados
- **/rewind** — desfaz código e conversa até um checkpoint

Comandos “/” úteis



- **/context** • **/compact** • **/clear** — higiene de contexto
- **/btw** — pergunta paralela, sem poluir o histórico
- **/model** • **/effort** — ajusta modelo e nível de raciocínio
- **/diff** • **/code-review** • **/security-review** — antes de entregar

Digite / na sessão para ver e filtrar tudo.

```
claude

> /

/clear          Recomeça com contexto limpo
/compact        Resume e libera contexto
/context        Mostra o uso do contexto
/btw           Pergunta paralela, sem poluir o histórico
/model          Troca o modelo da sessão
/diff           Abre o visualizador de mudanças

digite para filtrar...
```



A pasta .claude: arquitetura

- **No projeto** — versione no Git para o time herdar o mesmo comportamento
- **No ~/.claude** — sua configuração pessoal, vale em todos os projetos
- **Precedência** — organização > sessão (flags) > projeto > global

```
claude — .claude/

> tree .claude

projeto/
├── CLAUDE.md           memória do projeto
├── .mcp.json           servidores MCP
└── .claude/
    ├── settings.json   settings.local.json
    ├── rules/ skills/ commands/
    ├── agents/ agent-memory/
    └── output-styles/

~/.claude/ (global — todos os projetos)
CLAUDE.md · settings.json · skills/ · themes/
projects/ (memória + transcripts)
```



A pasta .claude: o que cada arquivo faz

- **CLAUDE.md** — instruções e contexto lidos toda sessão
- **rules/** — instruções por tópico, opcionalmente por caminho
- **settings.json** — permissões, hooks, variáveis e modelo padrão
- **skills/** • **commands/** — fluxos reutilizáveis (/nome ou automáticos)
- **agents/** • **agent-memory/** — subagentes e sua memória persistente
- **.mcp.json** • **output-styles/** — servidores MCP e estilos de resposta

Conectores: Jira + GitHub



- **Jira (via MCP)** — o Claude lê e age nos tickets direto, sem copiar e colar
- **GitHub (via gh CLI)** — usa o gh que o time já tem: PRs, issues e commits
- **Juntos** — “implemente o ENG-4521 e abra um PR”: do ticket ao PR

O .mcp.json compartilha a conexão com o time; o login continua individual.

```
claude - conectores

> Implemente o ENG-4521 e abra um PR

✓ Jira ENG-4521                lido via MCP
  "Corrigir validação do formulário de login"

✓ Alteração implementada e testada

✓ gh pr create                  → PR #318 aberto

↳ Do ticket ao PR, sem sair do terminal.
```

Como o Claude lida com a memória



- **Carrega sozinho** — CLAUDE.md, rules/ e a auto-memória, a cada sessão
- **Você pluga** — PROGRESS.md e PLAN.md (via CLAUDE.md / @menção)
- **Auto-memória** — o Claude anota e relê notas entre sessões

Rode /memory para ver e gerenciar o que está ativo.

```
claude — /memory

> /memory

Arquivos de memória carregados
✓ CLAUDE.md           projeto
✓ ~/.claude/CLAUDE.md global
✓ rules/*.md          2 arquivos

Auto-memória: ON    · 3 notas salvas

↳ ~/.claude/projects/<projeto>/memory/
```

Automação: do /loop à nuvem



- **Na sessão** — /loop repete um prompt; some quando você fecha
- **No Desktop (Cowork)** — tarefas recorrentes: relatórios e briefings prontos
- **Na nuvem / GitHub Actions** — roda sem depender da sua máquina ligada

Escolha a camada conforme a durabilidade que a automação precisa.

claude – automação		
Sessão /loop · CLI	Desktop Cowork	Nuvem cloud · Actions
• Leve, na sua sessão • Enquanto você trabalha • Some ao fechar	• Recorrente, persistente • Relatórios e briefings • App aberto p/ rodar	• Roda sem sua máquina • Usa seus conectores • Automação de time



Pare hoje, continue amanhã

- **Tudo é salvo no disco** — útil quando bate o limite ou o cansaço
- **Ao encerrar** — `/rename`, atualize o `PROGRESS.md`, `/recap` ou `/export`
- **Ao voltar** — `claude -c`, `/resume` (pelo nome) ou `claude -r`

Às vezes recomeçar limpo + `PROGRESS.md` rende mais que um transcript longo.

Abrindo o Debate



Questões para reflexão:

1. Qual foi o principal aprendizado para você?
2. Como você pode aplicar isso em seu dia a dia?
3. Quais foram as dúvidas geradas?
4. Que conexões você vê com outros tópicos?

Obrigado!

Até o próximo episódio da BIUD Talks

Inovando juntos | Crescendo juntos