🥋 DOJO DE AGENTES

Agentes de IA & Fluxos

🚀 Antigravity • Agents • Rules • Skills • Workflows • Canonical Cycle

📅 Agenda do Dia

1. **Introdução** aos Agentes de IA
   - O que são e como funcionam?
2. **Setup do Agente** 🛠️
   - Rules, Skills e Workflows
3. **Metodologias** 📚
   - SDD, MCP e Memória
4. **Antigravity** (Exemplos funcionando)
5. **Canonical Cycle** 🔄
6. **Debate e Próximos Passos** 🚀

🤖 1. Agentes de IA

O que são?

Sistemas de software autônomos capazes de perceber seu ambiente, raciocinar sobre ele e tomar ações para atingir objetivos específicos. Diferente de um "chatbot" passivo, um agente tem **autonomia** e **capacidade de execução**.

Para que servem?

Para automatizar tarefas complexas que exigem tomada de decisão, uso de ferramentas e múltiplos passos, atuando como um "parceiro de trabalho".

> **Exemplo Simples:**
> Um agente que monitora seu repositório git, detecta PRs abertos, roda os testes e, se falhar, comenta sugerindo a correção. O chat do antigravity pode ser usado para interagir com o agente.

🧠 2. Por que configurar o Agente?

1. Eficiência de Contexto

Em vez de repetir *"eu uso Clean Code e testes em Jest"* todo dia, isso vira uma **Rule** permanente.

2. Memória de Longo Prazo

O agente "lembra" como operar seu sistema e ferramentas via **Skills**, sem você precisar colar documentação técnica a cada prompt.

3. Comandos Naturais

Transforme prompts gigantes e detalhados em pedidos simples do dia a dia.

❌ **Antes (Verboso)**

"Por favor, faça o deploy, lembrando de rodar testes, verificar a tag, confirmar no chat..."

✅ **Depois (Natural)**

"Faça o deploy."

📂 A Estrutura `.agent`

A "bancada" do seu Agente é organizada em pastas específicas para garantir modularidade:

```
.agent/
 ├── rules/        # 🧠 Contexto e Diretrizes (O que saber?)
 ├── skills/       # 🛠️ Ferramentas (Como fazer?)
 └── workflows/    # 📋 Processos (O que fazer passo-a-passo?)
```

Se não está aqui, o agente **não sabe**.

📏 3. Rules (Regras)

O que são?

Diretrizes explícitas e imutáveis que governam o comportamento do agente. São o "contexto sistêmico" que garante segurança, conformidade e estilo.

Para que servem?

Para evitar que o agente tome ações indesejadas (ex: apagar banco de produção) e para forçar padrões de qualidade (ex: sempre escrever testes).

“ Exemplo Simples:

```
Regra: "Nunca altere arquivos de configuração (.env, docker-compose) sem pedir confirmação explícita ao usuário."
```

”

🛠️ 4. Skills (Habilidades)

O que são?

Pacotes de conhecimento operacional. Uma Skill ensina ao agente **como** usar uma ferramenta específica, uma API ou executar um procedimento técnico complexo.

Para que servem?

Para expandir o "canivete suíço" do agente. Em vez de apenas gerar texto, ele aprende a interagir com o mundo real (Jira, AWS, Kubernetes).

“ **Exemplo Simples:**

Skill: `git-ops`

Ensina o agente a fazer `git checkout -b`, `git add`, `git commit` seguindo o padrão Conventional Commits do time. ”

🔀 5. Workflows (Fluxos)

O que são?

Sequências orquestradas de tarefas. Um Workflow é um "roteiro" passo-a-passo que o agente segue para completar um objetivo maior, garantindo consistência no processo.

Para que servem?

Para padronizar processos repetitivos e propensos a erro humano, como deploys, onboadings ou migrações de banco de dados.

Exemplo Simples:
Workflow: `deploy-production`

1. Acessar job no Jenkins.
2. Identificar a tag de release (user input).
3. Preencher parâmetros de build.
4. Submeter deploy e notificar Discord.

📜 6. Spec Driven Development (SDD)

O problema do Chat:
Conversar "de boca" com a IA gera loops de erro, esquecimento de contexto e alucinações ("telefone sem fio").

A solução SDD:

1. Escreva o que você quer num arquivo Markdown (Spec).
2. A IA lê a Spec.
3. A IA implementa baseado na Spec (não na sua memória).

🔌 7. MCP: Model Context Protocol

Como conectar a IA ao mundo real?
O **MCP** é um padrão aberto que permite ao Agente atuar fora da IDE.

- **Ler tickets no Jira** 🎫
- **Consultar documentação no Confluence** 📚
- **Acessar Banco de Dados** 🗄️
- **Pesquisar na Web** 🌐

É a ponte segura entre o "cérebro" da IA e os dados da sua empresa.

🧠 8. Memória por Arquivo (vs Chat)

A Filosofia "Manus":
O chat é efêmero, volátil e não confiável para guardar decisões.

- **Chat**: Apenas para pedir, ajustar e comandar.
- **Arquivo (.md)**: A única fonte de verdade e persistência.

Se você não escreveu no arquivo, para a IA, **não existe**.
Mantenha a documentação viva e o contexto salvo.

🌌 9. Antigravity (Google)

O que é?

Um agente avançado de codificação desenvolvido pelo Google Deepmind. Ele opera direto na sua IDE e terminal, combinando LLMs poderosos com acesso seguro a ferramentas do sistema (shell, editor de arquivos, browser).

Para que serve?

Para atuar como um engenheiro de software pleno/sênior em par com você. Ele entende contexto de projeto, navega em arquivos, refatora código e até corrige bugs de forma autônoma.

> **Exemplo Simples:**
> "Antigravity, refatore este controller para usar o padrão Repository e crie os testes unitários para a nova classe."

🔄 O Canonical Cycle

Framework de Consistência e Verdade

🏛️ Filosofia: O Contrato IA-Humano

1. **IA nunca decide a verdade**: Apenas propõe interpretações.
2. **Humano nunca reinterpreta raw material sozinho**: Usa o fluxo estruturado.
3. **Canonical Material é o Ponto de Controle**: Onde a verdade é firmada.

Os 3 Pilares

- 🏗️ **Bancada (Workspace Agent)**: Contexto de código e produto.
- 🔄 **Regras (Canonical Agent)**: Guardião do processo.
- 🌐 **Externo (MCPs)**: Dados do Jira, Web, Docs.

🌊 O Fluxo do Ciclo

A Jornada da Informação:

`Raw` ➡️ `Filtered` ➡️ `Canonical` ➡️ `Artifacts` ➡️ `Delivery`

- 🧱 **Raw**: Dados brutos (reuniões, anotações). *Sem verdade.*
- 🔍 **Filtered**: IA estrutura e organiza. *Proposta.*
- 🏛️ **Canonical**: Humano aprova. **VERDADE.**
- 📄 **Artifacts**: Tickets, Docs, Código. *Descartáveis.*
- 🚀 **Delivery**: O mundo real (Deploy, PR, Publish).

👥 Roles Sequenciais

O ciclo conecta especialistas em carrossel. O artefato de um vira o **Raw** do outro.

1. 🕵️ **Analista**: Entende o negócio → *Requisitos*
2. 🎨 **Designer**: Cria a experiência → *Protótipos*
3. 🏛️ **Arquiteto**: Define a estrutura → *ADRs / Diagramas*

4. ⚙️ **Engenheiro**: Planeja a mudança → *Tasks Técnicas*
5. 💻 **Desenvolvedor**: Codifica → *Commits / PRs*

> **Atalho**: Roles como Designer e Arquiteto são opcionais e puláveis.

📜 As Regras de Ouro

1. ❌ **Nada gera artefato sem Canonical.** (Sem atalhos!)
2. ✅ **Toda decisão tem aprovação humana.**
3. 🔁 **Mudou o contexto? Novo ciclo.** (Reentrância)

4. 🔗 **Artefatos são filhos do Canonical.**
5. 🗑️ **Artefatos são descartáveis**, o Canonical é eterno (na pasta `archives/`).

```
archives/
  └── ciclo_01/
      ├── analista/
      ├── engenheiro/
      └── desenvolvedor/
```