MÓDULO 10

Mission Control
Seu Painel Operacional

Interface visual para monitorar e gerenciar seu agente — mesmo usando Telegram

DURAÇÃO	FORMATO	OPÇÕES
10 min	Demo	4 ferramentas

Por Que Um Painel Visual?

Você conversa com o agente no Telegram. Então por que precisaria de uma interface visual?

Telegram = interação. Dashboard = visão

Chat é ótimo pra conversar. Mas pra ver o estado do sistema num relance? Painel visual ganha.

O Que Um Painel Resolve

Visão Geral Rápida

Quantos crons rodaram hoje? Quais falharam? Quantas tasks pendentes? Um relance responde tudo.

Tracking de Métricas

MRR, usuários ativos, peças de conteúdo produzidas, uptime de serviços — tudo num lugar.

Organização Visual

Kanban de tasks, pipeline de conteúdo, status de projetos — mais fácil de escanear que texto.

Alertas Visuais

Serviço down? Custo explodindo? Deadline próximo? Fica óbvio com cores e badges.

Exemplo Real: Amora Mission Control

SEÇÃO	O QUE MOSTRA
Tasks	Kanban: Backlog → In Progress → Done → Blocked
Memory	Insights recentes, decisions, lessons extraídas
Crons	Status de cada cron (última execução, próximo run, erros)

SEÇÃO	O QUE MOSTRA
Content HQ	229 packs, 773 peças — pipeline de produção visual
Health	Uptime de gateway, Telegram, skills, APIs

4 Opções de Ferramentas

Da mais simples à mais customizável:

Opção A: Google Sheets (Iniciante)

✅ Prós

- Todo mundo sabe usar
- Gráficos nativos
- Grátis
- Integra com GOG skill (Google Calendar/Drive)

❌ Contras

- Não é um banco de dados de verdade
- Limites de linhas (~10 milhões, mas lento antes disso)
- Sem real-time
- Design básico

🎯 Melhor pra:

Prototipagem rápida, dashboards simples, quem quer resultado em 30 minutos.

Opção B: Notion (Intermediário)

✅ Prós

- Interface bonita e familiar
- Databases relacionais
- Mobile nativo
- Compartilhamento fácil
- Skill disponível no ClawHub

❌ Contras

- API com rate limits (3 req/sec)
- Dados na nuvem (não self-hosted)
- Custo se precisar de features avançadas

Melhor pra:

Quem já é power user do Notion e quer integrar agente com workspace existente.

Opção C: NocoDB (Avançado Iniciante)

□ Prós

- Self-hosted (seus dados, seu controle)
- Interface Notion-like
- API REST completa
- Grátis e open-source
- Relações entre tabelas
- Docker Compose = setup rápido

□ Contras

- Requer conhecimento básico de Docker
- Menos customizável que código próprio
- Comunidade menor que Notion

□ **Melhor pra:**

Quem quer resultado rápido com controle total dos dados. Recomendado para a maioria.

Opção D: Custom (Next.js + Supabase)

□ Prós

- Controle total da UI/UX
- Real-time com WebSockets
- Grátis pra hospedar (Vercel + Supabase)
- Escalável e profissional
- Aprende tech stack moderno

□ Contras

- Requer conhecimento de dev full-stack
- Tempo de setup: horas/dias
- Manutenção contínua

Melhor pra:

Devs que querem algo sob medida e estão dispostos a investir tempo.

">

Estrutura de Dados (Universal)

Independente da ferramenta escolhida, essas são as "tabelas" essenciais:

1. Tasks

- `id` — identificador único
- `title` — descrição da task
- `status` — pending | in_progress | done | blocked
- `priority` — low | medium | high | urgent
- `created_at` — timestamp
- `due_date` — deadline (opcional)
- `assigned_to` — agente ou humano

2. Memory Events

- `id` — identificador único
- `date` — quando aconteceu
- `category` — decision | lesson | insight | observation
- `content` — o que foi aprendido/decidido
- `source` — main | heartbeat | cron | sub-agent

3. Cron Jobs

- `id` — identificador único
- `name` — nome do cron
- `schedule` — expressão cron ou intervalo
- `last_run` — última execução
- `next_run` — próxima execução
- `status` — active | paused | failed
- `last_result` — success | error (+ mensagem)

4. Health Checks

- `id` — identificador único
- `service` — gateway | telegram | skill_X | api_Y
- `status` — healthy | degraded | down
- `last_check` — timestamp
- `uptime_%` — disponibilidade (últimos 7 dias)

5. Agent Stats (Métricas)

- `id` — identificador único
- `date` — dia
- `messages_sent` — contagem

- `skills_used` — lista de skills acionadas
- `errors` — contagem de erros
- `model_tokens_used` — total de tokens
- `uptime_hours` — horas ativas

Conectar o Agente ao Painel

Depois de escolher a ferramenta e criar as tabelas, o agente precisa de funções pra escrever dados.

Funções Essenciais

JAVASCRIPT

```
// Exemplo: funções helper pro agente

dashboard_log_task(title, priority)
  // Cria tarefa no painel

dashboard_update_task(id, status)
  // Marca como done/blocked

dashboard_log_memory(content, category)
  // Salva insight importante

dashboard_update_health(service, status)
  // Atualiza status de serviço

dashboard_get_stats()
  // Retorna resumo do dia
```

Crons Necessários

Crons isolados que atualizam o painel automaticamente:

1. **stats-collector (a cada 1h)**
   Coleta métricas (uptime, erros, skills usadas) e escreve no painel

2. **health-monitor (a cada 15min)**
   Testa se gateway/telegram/skills estão respondendo e atualiza health table

3. **task-reminder (a cada 4h)**
   Verifica tarefas com due_date próximo e avisa no Telegram

⚠ **IMPORTANTE: Use sessionTarget: isolated**

Crons no main session (systemEvent) NÃO executam. Use `isolated` + `agentTurn` + `announce`.

Personalizar por Perfil

O que você mostra no painel depende do seu trabalho:

□ Founder

- MRR, ARR, churn, CAC, LTV
- Pipeline de vendas (leads → prospects → clientes)
- OKRs da semana/mês
- Runway (meses de caixa restante)

□□ Dev

- Deploys recentes (sucesso/falha)
- Erros críticos (via Sentry/Logtail)
- PRs pendentes de review
- Uptime dos serviços (99.9% SLA?)
- Cobertura de testes

□ Criador

- Conteúdos publicados essa semana
- Performance (views, engajamento, CTR)
- Pipeline de produção (ideias → roteiro → gravação → edição → publicado)
- Estoque de conteúdo pronto (buffer)

□ Produtividade

- Reuniões de hoje
- Tasks completadas vs pendentes (burn-down chart)
- Hábitos rastreados (meditação, exercício, deep work)
- Tempo em deep work vs shallow work

> □ **Dica**
>
> Não tente rastrear TUDO. Escolha 5-10 métricas que realmente importam. Menos é mais.

☐ Checkpoint do Módulo 10

- [ ] Ferramenta escolhida (Sheets/Notion/NocoDB/Custom)
- [ ] Estrutura de dados criada (Tasks, Memory, Crons, Health, Stats)
- [ ] Funções helper pro agente configuradas
- [ ] Crons de atualização agendados (stats, health, reminders)
- [ ] Widgets personalizados pro seu perfil
- [ ] Primeiro teste: agente logou task e apareceu no painel

☐ Prompt para o Agente

Cole este prompt no chat do seu OpenClaw depois de concluir o módulo:

Acabei de assistir o Módulo 10 sobre Mission Control. Preciso de um painel visual pra gerenciar você e ver o estado do sistema num relance. **Me ajude a escolher e configurar a ferramenta certa: 1. Perguntas pra me conhecer:** - Qual meu nível técnico? (iniciante/intermediário/avançado) - Já uso alguma ferramenta? (Notion, Sheets, etc.) - Prefiro algo rápido ou algo perfeito? - Vou gerenciar só você ou uma equipe de agentes? **2. Baseado nas minhas respostas, recomende:** - Qual ferramenta (Sheets/Notion/NocoDB/Custom) - Por que essa escolha faz sentido pra mim - Prós e contras específicos pro meu caso **3. Setup completo:** - Me guie passo a passo na configuração - Crie as tabelas necessárias (Tasks, Memory, Crons, Health, Stats) - Configure as funções helper - Crie os crons de atualização - Personalize os widgets pro meu perfil (founder/dev/creator/produtividade) **4. Primeiro teste:** - Logue uma tarefa de teste - Mostre ela aparecendo no painel - Marque como "done" via você - Confirme a atualização **5. Documente:** - Crie `docs/mission-control-setup.md` com tudo que configuramos - Credenciais no 1Password (nunca hardcoded) - Como acessar o painel Vamos construir meu Mission Control?

PRÓXIMO MÓDULO

Módulo 11 — Wrap-up & Próximos Passos