MÓDULO 04

 Módulo Diferenciador — Ninguém Ensina Isso

Memória: O Segredo que Ninguém Ensina

Como fazer seu agente lembrar de tudo — decisões, lições, contexto — entre sessões

 25 minutos Demo + Diagramas 8+ arquivos de template

O Problema: AIzheimer Digital

Todo agente AI nasce com amnésia. A cada nova sessão, ele esquece TUDO. Isso é um problema fundamental.

Sem memória estruturada

- Repete as mesmas perguntas todo dia
- Esquece decisões importantes
- Não aprende com erros
- Perde contexto de projetos
- Você vira o backup humano do agente

Com memória estruturada

- Lembra de decisões da semana passada
- Evolui a cada interação
- Consulta lições antes de sugerir
- Mantém estado de projetos
- Funciona como um verdadeiro COO

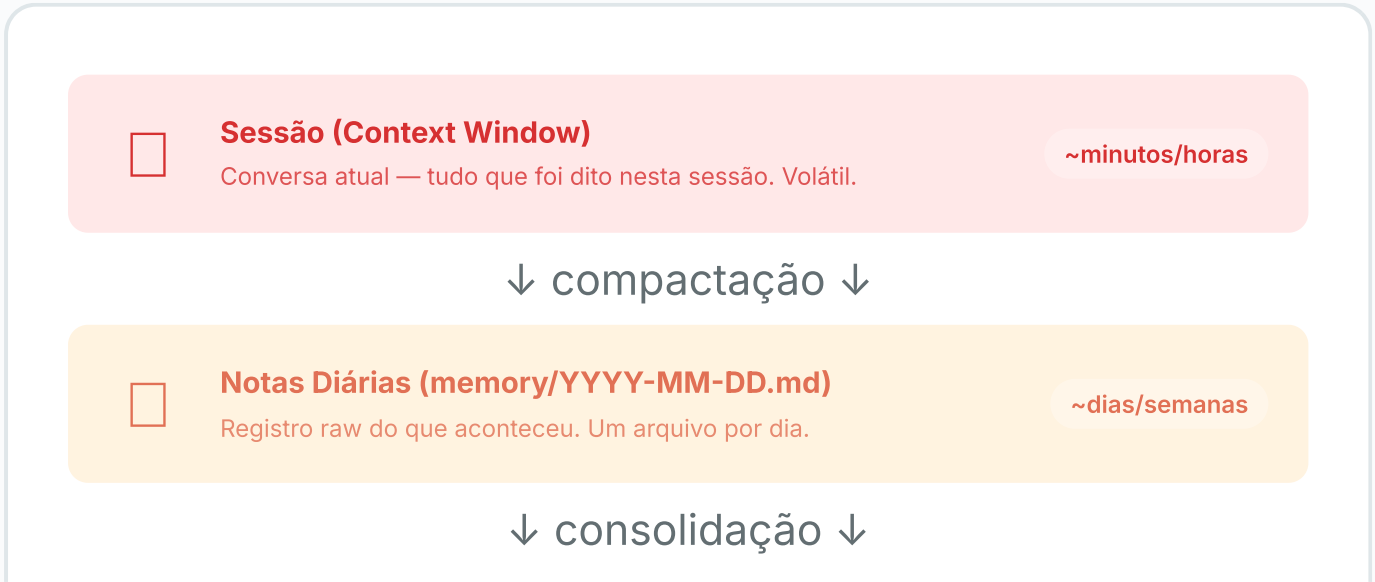
Caso Real

Dia 2 do uso: Token overflow de 173k+ tokens — o agente tentou carregar TUDO na memória a cada mensagem. Resultado: travou, custou caro, e perdeu contexto.

Dia 13 do uso: Com memória estruturada, a Amora lembrou de uma decisão do Dia 4 sem ninguém pedir. "Você decidiu no dia 4 que posts de LinkedIn devem ter tom conversacional, não corporativo." → Isso é memória funcionando.

Arquitetura de Memória em Camadas

A solução é um sistema em camadas, do volátil ao permanente:



Topic Files (memory/decisions.md, lessons.md, etc.)

Informação curada e organizada por tema.

~meses

↓ destilação ↓

MEMORY.md (Índice Central)

Sabedoria destilada. O que realmente importa a longo prazo.

permanente

Por que camadas?

Agentes têm um limite de contexto (tokens). Se você carrega TUDO a cada mensagem, estoura o limite e paga caro. Com camadas, o agente carrega só o essencial e busca o resto sob demanda via `memory_search()` .

Os 5 Topic Files

Cada arquivo tem um propósito claro. Juntos, formam a memória de longo prazo do seu agente.

decisions.md

Decisões permanentes — nunca perder. Coisas que você decidiu e não quer rediscutir.

```
## 2026-02-04
- LinkedIn: tom conversacional, NÃO corporativo
- Crons: SEMPRE isolated + agentTurn
- Sub-agents: NUNCA autonomia total
```

lessons.md

Lições aprendidas — erros, padrões, insights. Categorizadas por duração.

```
### Estratégicas (permanentes)
- Extrair lições ANTES de compactar

### Táticas (30 dias)
- Rate limit YouTube: 2 req/min
```

projects.md

Projetos ativos — estado atual, próximos passos, blockers.

```
## Curso OpenClaw
Status: gravando módulo 4
Próximo: finalizar PDFs
Blocker: nenhum
```

people.md

Pessoas — equipe, parceiros, contatos. Contexto relacional.

```
## Bruno
- Founder, foco em AI/SaaS
- Prefere comunicação direta
- Timezone: America/Sao_Paulo
```

pending.md

Pendências — coisas aguardando input do usuário. Limpa quando resolvidas.

```
- [ ] Aprovar design do dashboard (esperando desde 02/14)
- [ ] Decidir stack do Mission Control: Supabase vs NocoDB?
- [x] ~~Escolher modelo padrão~~ → Sonnet (resolvido 02/10)
```

Estrutura de Pastas Completa

TREE

```
workspace/ ├── MEMORY.md # Índice central — sabedoria destilada ├── SOUL.md # Personalidade do agente ├── USER.md # Sobre o usuário ├── HEARTBEAT.md # Checklist periódico └── memory/ ├── decisions.md # Decisões permanentes ├── lessons.md # Lições (estratégicas + táticas) ├── projects.md # Projetos ativos ├── people.md # Contatos e equipe ├── pending.md # Aguardando input ├── feedback/ # Feedback
```

```
loops (approve/reject) | ├── content.json | └── tasks.json ├── 2026-02-14.md # Nota diária (raw) ├── 2026-02-15.md └── 2026-02-16.md
```

⚡ Compactação: O Momento Crítico

Quando o contexto enche, o OpenClaw compacta (resume) a conversa. Se não extrair informação antes, ela se perde.

□ REGRA INVIOLÁVEL

ANTES de cada compactação, o agente DEVE extrair:

1. **Decisões** → salvar em `memory/decisions.md`
2. **Lições aprendidas** → salvar em `memory/lessons.md`
3. **Updates de projetos** → atualizar `memory/projects.md`
4. **Pessoas mencionadas** → atualizar `memory/people.md`
5. **Pendências** → atualizar `memory/pending.md`

Se não extrair antes de compactar, perde 80% do valor da conversa.

⚙ Configuração de Compactação

```json
{ "compaction": { "mode": "default" }, "contextTokens": 160000, // Limite de contexto "reserveTokensFloor": 30000 // Reserva pra raciocínio }
```

> ⚠ **Por que 30k de reserva?**
>
> Se o agente usar todo o contexto pra ler dados, sobra pouco espaço pra "pensar". A reserva garante que ele sempre tem espaço pra raciocinar. Sem isso, respostas ficam cortadas ou superficiais.

□ Memory Search: Busca Sob Demanda

Em vez de carregar tudo na memória, o agente busca quando precisa:

Como funciona na prática

1. Você pergunta: *"Qual foi a decisão sobre o tom do LinkedIn?"*
2. Agente roda: `memory_search("decisão tom LinkedIn")`
3. Retorna chunks relevantes (~400 tokens cada)
4. Agente usa `memory_get(path, from, lines)` pra puxar o trecho exato

5. Responde com contexto: *"No dia 4, você decidiu tom conversacional, não corporativo."*

```
# Indexar todos os arquivos de memória $ openclaw memory index --all # Testar busca
$ openclaw memory search "decisão LinkedIn"
```
TERMINAL

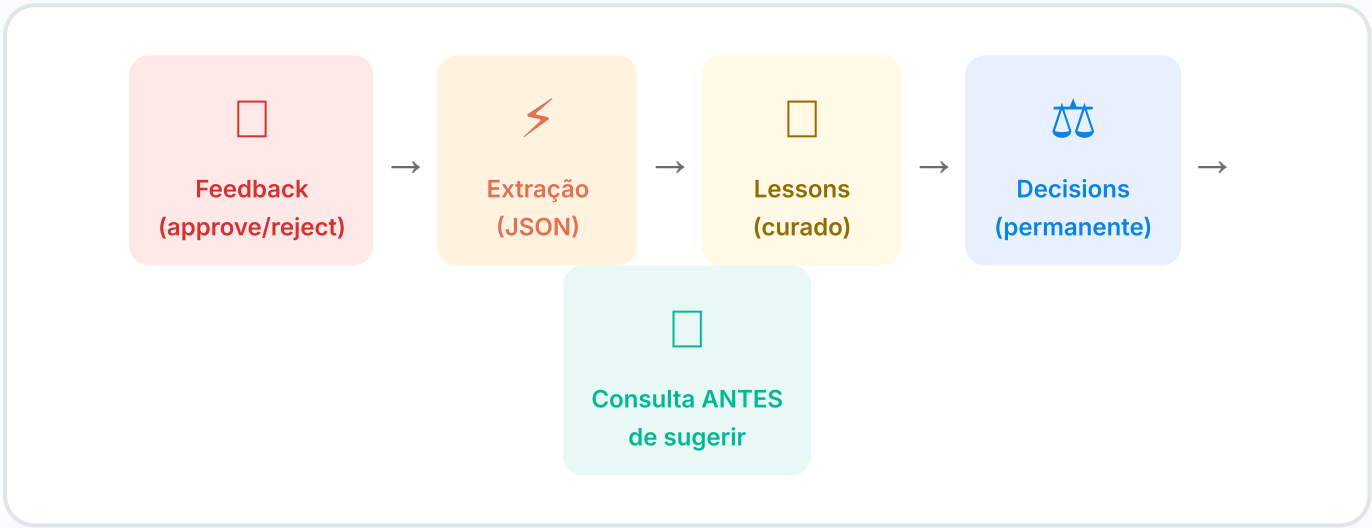
Economia de tokens:

Carregar MEMORY.md inteiro a cada mensagem = ~50KB = □. Usar memory_search sob demanda = ~400 tokens por busca = □. A diferença entre gastar $2-3/dia e $0.10/dia.

Feedback Loops: Evolução Real

O segredo para o agente melhorar com o tempo: capturar approve/reject em cada interação.

Como Funciona



Exemplo Prático: Feedback em Ação

1. Agente sugere algo:

> *"Para o post de LinkedIn, sugiro formato carrossel com 8 slides e tom executivo formal."*

2. Você rejeita:

> *"Não, muito formal. Posts de LinkedIn devem ser conversacionais, como se tivesse falando com um amigo."*

3. Agente registra feedback:

FEEDBACK/CONTENT.JSON

```
{ "entries": [ { "date": "2026-02-04", "context": "Sugeri formato carrossel com tom executivo formal", "decision": "reject", "reason": "Tom muito formal. LinkedIn deve ser conversacional.", "tags": ["linkedin", "tom", "conteúdo"] } ] }
```

4. Na próxima vez, o agente CONSULTA antes de sugerir:

> □ *"Para o post de LinkedIn, vou usar tom conversacional (decisão de 02/04). Formato: texto direto com storytelling, sem carrossel formal."*
>
> → **Evolução real. O agente não repete o mesmo erro.**

□ Consolidação Periódica

A cada poucos dias (ou via heartbeat), o agente deve:

Ação	Frequência	O Que Faz
□ Notas diárias → Topic files	A cada 2-3 dias	Extrai insights das notas brutas
□ Classificar lições	Semanal	Estratégica (permanente) vs Tática (30 dias)
□ Limpar pending	Semanal	Remove itens resolvidos
□ Atualizar MEMORY.md	Quinzenal	Destila sabedoria dos topic files
□ Expirar lições táticas	Mensal	Remove lições com >30 dias

□ **Automatize via heartbeat!**

Configure no HEARTBEAT.md para o agente fazer consolidação automaticamente a cada poucos dias. Sem intervenção manual — o agente cuida da própria memória.

Otimização: De $3/dia Para $0.10/dia

A configuração de inicialização de sessão é o maior fator de custo. Acerte isso e economize 95%.

O Que NÃO Carregar Automaticamente

Configuração ingênua

- Carrega MEMORY.md inteiro (~50KB)
- Carrega histórico de todas as sessões
- Carrega todos os topic files
- Carrega outputs anteriores
- ~$2-3/dia em tokens

Configuração otimizada

- SOUL.md (personalidade)
- USER.md (contexto do usuário)
- IDENTITY.md (dados concretos)
- memory/YYYY-MM-DD.md (só hoje+ontem)
- Resto via memory_search() sob demanda
- ~$0.10/dia em tokens

Configuração do AGENTS.md

Adicione estas regras no AGENTS.md do seu agente:

```
## Every Session Before doing anything else: 1. Read `SOUL.md` — this is who you are 2. Read `USER.md` — this is who you're helping 3. Read `memory/YYYY-MM-DD.md` (today + yesterday) for recent context 4. **If in MAIN SESSION** (direct chat): Also read `MEMORY.md` ## Memory Rules - Use `memory_search()` for anything NOT in active context - Use `memory_get()` to pull ONLY the needed lines - NEVER load all topic files at session start - BEFORE each compaction: extract lessons, decisions, pending - This rule is **INVIOLABLE**
```
MARKDOWN

Memory Search: Provider Chain

O OpenClaw tenta embeddings nesta ordem:

Provider	Tipo	Custo	Qualidade
Local GGUF	Roda na VPS	Grátis	□□□
OpenAI	API	$0.0001/1k tokens	□□□□

Provider	Tipo	Custo	Qualidade
Gemini	API	Grátis (free tier)	□□□□
Voyage	API	$0.0001/1k tokens	□□□□□

□ **Recomendação:**

Deixe no `auto`. O OpenClaw escolhe o melhor provider disponível. Se tiver API key da OpenAI configurada, usa ela pra embeddings — é baratíssimo.

□ O HEARTBEAT.md Ideal

Configure o heartbeat pra incluir manutenção de memória:

HEARTBEAT.MD

```
# HEARTBEAT.md - Checklist Periódico ## Checks (a cada heartbeat) - [ ] Emails urgentes? - [ ] Eventos nas próximas 24-48h? - [ ] Projetos parados há >5 dias? ## Memória (a cada 2-3 dias) - [ ] Consolidar notas diárias → topic files - [ ] Classificar novas lições (estratégica vs tática) - [ ] Limpar pending.md (itens resolvidos) ## Memória profunda (quinzenal) - [ ] Atualizar MEMORY.md com insights destilados - [ ] Expirar lições táticas com >30 dias - [ ] Revisar decisions.md — ainda fazem sentido?
```

□ Checkpoint — Módulo 4 Completo

- ☐ Pasta `memory/` criada com os 5 topic files
- ☐ MEMORY.md criado como índice central
- ☐ Regra de extração antes de compactação configurada no AGENTS.md
- ☐ contextTokens e reserveTokensFloor configurados
- ☐ Memory search indexado (`openclaw memory index --all`)
- ☐ HEARTBEAT.md configurado com checks de memória
- ☐ Feedback loops configurados (pelo menos 1 domínio)
- ☐ Teste: agente "lembrou" de algo de uma sessão anterior □

□ Prompt Para Seu Agente — Cole no Chat

Cole este prompt junto com os arquivos de template. O agente implementa tudo automaticamente.

□ Copie tudo dentro da caixa abaixo e cole no Telegram pro seu agente

□ Anexe junto: os templates de memory/ que vieram no kit do módulo

Acabei de assistir o Módulo 4 do curso sobre memória. Preciso que implemente o sistema de memória completo.

O que preciso que faça:

1. **Me explique como funciona sua memória** — o problema (Alzheimer entre sessões) e a solução (memória em camadas)

2. **Crie a estrutura de memória:**
- memory/ com: decisions.md, lessons.md, pending.md, people.md, projects.md
- MEMORY.md como índice
- Configurar notas diárias automáticas

3. **Configure a regra de extração antes de compactação** — NUNCA compactar sem extrair lições, decisões e pendências primeiro. Isso é INVIOLÁVEL.

4. **Configure o HEARTBEAT.md** — inclua checks de memória (consolidação, limpeza, expiração)

5. **Configure compactação:** contextTokens: 160.000, reserveTokensFloor: 30.000

6. **Configure memory search** — indexe tudo com `openclaw memory index --all`

7. **Configure feedback loops** — crie memory/feedback/ com pelo menos content.json e tasks.json

Regras:
- Me explique cada arquivo e PRA QUE SERVE antes de criar
- Me mostre um exemplo prático de como uma decisão vira memória permanente
- Regra INVIOLÁVEL: antes de CADA compactação, rodar checklist de extração
- No final, simule uma compactação pra eu ver o processo funcionando

□ **Próximo Módulo:**

Integrações & Crons — Conectar seu agente ao Google Calendar, 1Password, e automatizar tarefas recorrentes. A memória que você acabou de configurar é o que faz tudo isso funcionar entre sessões.