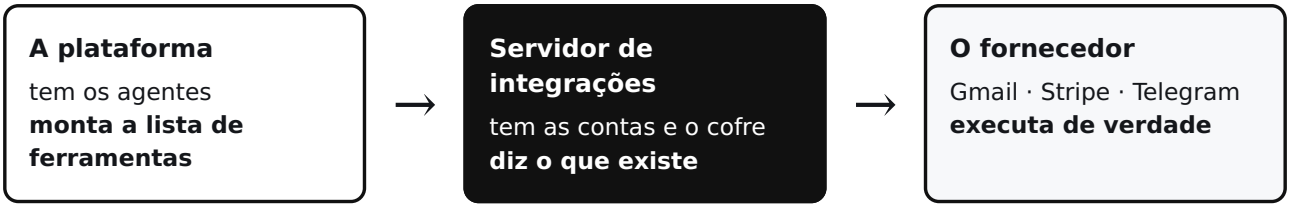
ARQUITECTURA · GUIA DO ZERO

O servidor de integrações

Como é que um agente descobre que uma ferramenta existe, quem lhe diz isso, e porque é que não é preciso um standard para o fazer.



A PERGUNTA QUE ESTE DOCUMENTO RESPONDE

“Quando o utilizador liga uma integração, como é que o agente passa a saber que ela existe — se não estamos a usar o MCP, que é o standard?”

Versão 0.1 · 26 de novembro de 2026

Complementa: “Do zero ao primeiro método” (os passos práticos) e “O cofre por dentro” (a base de dados).

1

As quatro peças

Antes de falar de descoberta, é preciso saber quem guarda o quê. São quatro peças com papéis distintos, e nenhuma sabe demasiado sobre as outras.

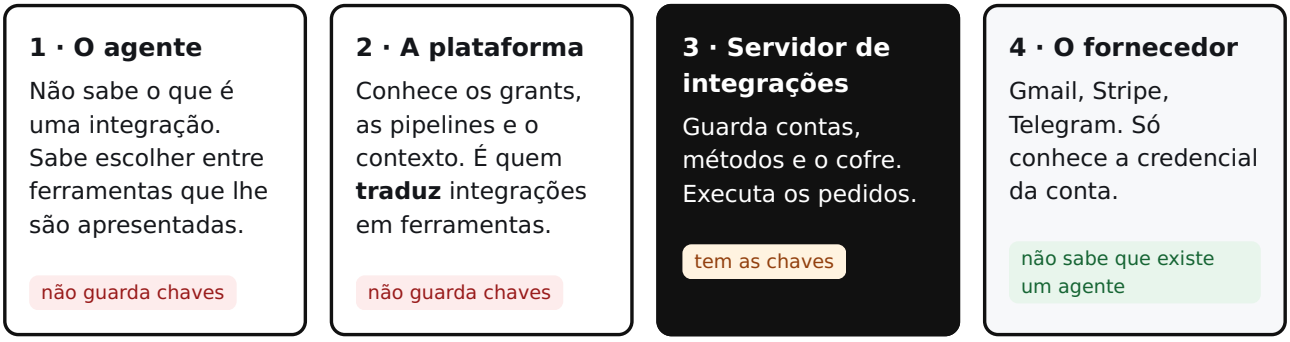


Fig. 1 — quatro papéis. As chaves vivem num só.

O que cada peça pode dizer sobre as outras

PEÇA	SABE	NÃO SABE
Agente	nome, descrição e campos das ferramentas que lhe deram	que existe cofre, contas, ou servidor
Plataforma	quem pode usar o quê (grants)	as chaves — só pede a execução
Servidor de integrações	contas, métodos, segredos	o que o agente estava a tentar fazer
Fornecedor	a conta que fez o pedido	que houve um agente, uma run, uma pipeline

> A separação é o que permite auditar: cada peça só vê a fatia que precisa.

2 · o anúncio →

2

O “anúncio”: o que o servidor diz que existe

Quando a plataforma se liga ao servidor de integrações, a primeira coisa que faz é perguntar **o que existe para este workspace**.

```
GET /micp/integrations        (autenticado como o workspace "ANATOMIA")

[
  { "integration": "gmail",
    "accounts": ["elizeu", "suporte"],
    "methods": [
      { "id": "send", "description": "Envia um email a partir da conta indicada.",
        "input": { "account": "account", "to": "email", "subject": "text", "body": "text" },
        "output": { "sent": "boolean", "id": "text" } },
      { "id": "read", "description": "Lê as últimas mensagens de uma pasta.",
        "input": { "account": "account", "folder": "text", "limit": "number" },
        "output": { "messages": "list" } }
    ] },
  { "integration": "sheets", "accounts": ["vendas"], "methods": [ … ] }
]
```

Este anúncio já vem filtrado. O servidor só lista o que foi habilitado para aquele workspace — e, dentro disso, a plataforma só entrega ao agente o que os **grants** permitirem. Duas filtragens, em série:



Fig. 2 — três filtros até chegar ao agente. Nenhum deles acontece no modelo.

PROPRIEDADE IMPORTANTE

O anúncio não contém segredos, nem nomes de contas alheias, nem métodos que aquele workspace não tem. É metadado: nomes, descrições, campos. Por isso pode ser pedido quantas vezes for preciso.

> O anúncio é o “menu”. Não abre a cozinha.

3

Traduzir o anúncio em ferramentas do modelo

Aqui está a peça que a plataforma tem de implementar. O modelo não entende "integrações" — entende uma lista de ferramentas, cada uma com nome, descrição e schema.

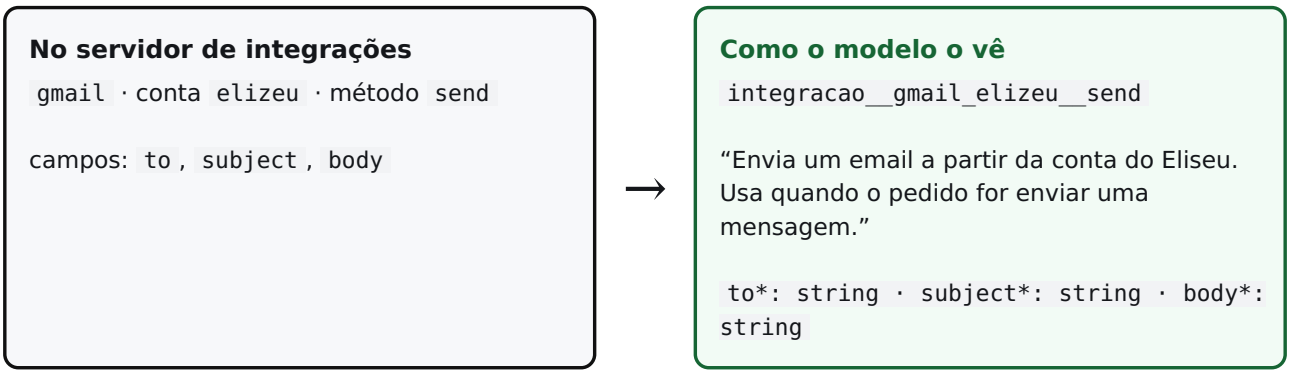


Fig. 3 — a tradução. Nome estável, descrição em linguagem natural, schema dos campos.

A regra de tradução, em quatro linhas

1. **nome** = `integracao__{alias}__{metodo}` — estável, sem acentos, sem colisões
2. **descrição** = a do método, mais o alias da conta (para o agente saber de quem é o email)
3. **parâmetros** = os campos do método, com tipos e obrigatórios — **sem** o campo `account`, que fica fixo no nome
4. **execução** = o pedido vai ao servidor de integrações, assinado, com o alias e o método

Porque é que o campo `account` desaparece

Porque a conta já está no nome da ferramenta. Se o modelo tivesse de a escolher, teria de saber que contas existem — e isso é informação de outro nível. Com a conta no nome, cada conta é uma ferramenta distinta, e o agente escolhe pelo que o utilizador disse ("o email do Eliseu").

E SE O NOME REPETIR?

O alias resolve. Duas contas Gmail chamam-se ambas "gmail", mas os alias são `elizeu` e `suporte` — e é o alias que entra no nome da ferramenta. Nomes humanos podem repetir-se; alias não.

> O modelo nunca vê a palavra "integração". Vê ferramentas.

4

Então preciso de MCP? Não.

Esta é a dúvida legítima: “os agentes recebem as ferramentas automaticamente por MCP; se não uso MCP, tenho de fazer isso eu?” — sim, tens. E é pouco.

	COM MCP	COM MICP
Quem sabe o que existe	o servidor MCP	o servidor de integrações
Quem entrega ao modelo	o cliente MCP (já existe)	**a plataforma** (monta a lista)
Transporte	protocolo MCP	HTTP + JSON assinado
Credenciais	o servidor MCP tem-nas	o cofre tem-nas, com grants por agente
Trabalho a fazer	nenhum (o cliente trata)	**~100 linhas:** pedir, filtrar, traduzir

O que a plataforma tem de implementar, concretamente:

1. Guardar o URL do servidor de integrações (ou assumir o modo embutido, com URL vazio).
2. Pedir o anúncio uma vez por execução (com cache de alguns minutos).
3. Cruz com os grants do agente: só o que ele pode usar.
4. Traduzir cada método numa ferramenta (nome, descrição, schema).
5. Quando o modelo chama a ferramenta, encaminhar o pedido assinado ao servidor e devolver o resultado.

DUAS VANTAGENS DE NÃO DEPENDER DE MCP

1. Funciona com qualquer modelo e qualquer cliente — não só os que falam MCP.
2. O grant é verificado antes de a ferramenta existir para o modelo: o que ele não pode usar nunca entra no prompt. Com MCP, o servidor mostra tudo o que tem e a recusa vem depois.

> MCP seria um adaptador a mais. A plataforma já fala a língua que o modelo entende.

5

O custo de anunciar tudo

Cada ferramenta entregue ao modelo ocupa espaço no prompt. Não é teórico: mede-se.

INTEGRAÇÕES LIGADAS	FERRAMENTAS	TOKENS APROXIMADOS SÓ EM SCHEMAS
3	8	~1 000
12	40	~5 000
40	140	~18 000

Aos 18 mil tokens, o problema deixa de ser o custo e passa a ser a **qualidade**: demasiadas opções confundem a escolha. A solução é a mesma que os assistentes grandes usam — **duas camadas**:

Camada 1 · catálogo curto

No prompt do sistema, uma lista de **uma linha por integração**:

`gmail (enviar, ler) · sheets (ler) · telegram (enviar)`

~150 tokens, e o agente já sabe o que existe.

Camada 2 · procurar e usar

Uma ferramenta `procurar_integracao("email do elizeu")` que devolve os **schemas completos** das 1–2 relevantes.

Só essas entram no contexto.

Fig. 4 — catálogo sempre, schemas a pedido. É o que mantém o contexto pequeno com 40 integrações.

REGRA PRÁTICA

Até ~10 ferramentas, entrega tudo. A partir daí, catálogo + procura. Não é preferência: é a diferença entre uma escolha certeira e uma escolha aleatória entre 140 opções.

> Anunciar tudo é fácil de implementar e caro de usar. Duas camadas resolvem as duas coisas.

RESUMO

A arquitectura numa página

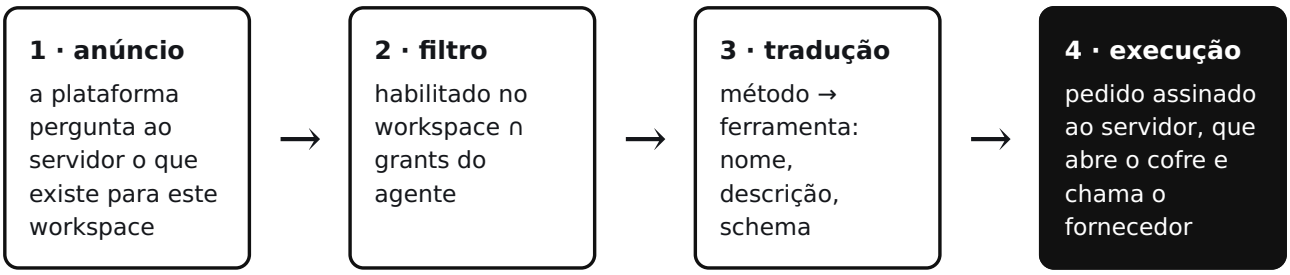


Fig. 5 — o ciclo completo. Começa no anúncio, acaba no fornecedor.

Cinco frases para guardar

1. O agente não sabe o que é uma integração — recebe ferramentas.
2. O **anúncio** vem do servidor de integrações e já vem filtrado por workspace.
3. A **plataforma** cruza o anúncio com os grants e traduz em ferramentas.
4. O **modelo** escolhe uma ferramenta; a plataforma encaminha; o servidor executa.
5. O **standard** não é preciso: o modelo só precisa de nome, descrição e schema.

Checklist do que implementar

#	ITEM	ESFORÇO
1	Campo do URL do servidor (vazio = embutido)	pequeno
2	Pedir o anúncio, com cache	pequeno
3	Filtrar por grants e traduzir em ferramentas	médio
4	Encaminhar a chamada assinada e devolver o resultado	médio
5	Catálogo curto + `procurar_integracao`	pequeno

E a resposta directa à tua pergunta: **sim, tens de implementar a entrega das ferramentas — e são cinco passos, não um protocolo novo.** O MCP já resolve isto para quem o usa; nós resolvemos com o mesmo esforço que faria falta para o integrar, e ficamos com o cofre, os grants e a auditoria que o MCP não tem.