STICKYBIT

GUIA EXECUTIVO · ABRIL 2026

RAG Hierárquico *Agentic*

O guia executivo para líderes de TI que precisam de IA empresarial confiável em produção — não em demonstrações.

ESCRITO PARA

CTOs · Gerentes de TI
PMs · PMOs

LEITURA

40 minutos

STICKYBIT

stickybit.com.br

STICKYBIT · GUIA EXECUTIVO

Índice

PARTE 01

Sumário Executivo

Quatro parágrafos e quatro números para levar à próxima reunião de comitê.

O que mudou

Entre 2023 e 2025, RAG virou sinônimo de "IA empresarial confiável". A promessa: acoplar um LLM à base de conhecimento da empresa e obter respostas precisas com rastreabilidade. Milhares de projetos foram iniciados com essa arquitetura.

O que não está funcionando

Em 2026, a indústria relata um padrão consistente: os assistentes de IA de primeira geração (arquitetura de busca única + modelo de linguagem) respondem bem perguntas diretas e falham exatamente nas perguntas que justificariam o investimento — aquelas que cruzam dados estruturados com documentos, exigem múltiplos passos de raciocínio ou precisam de rastreabilidade granular para auditoria. Taxa de respostas inventadas entre 25% e 30% em avaliações corporativas. Inaceitável para compliance, jurídico, saúde ou finanças.

O que vem depois

A arquitetura hierárquica *agentic* (Protocol-H e variantes) resolve o problema trocando o modelo "um LLM faz tudo" por uma topologia inspirada em organizações: um **supervisor estratégico** decompõe a pergunta, roteia para **workers especializados** (SQL, documentos, APIs) e sintetiza a resposta com rastreabilidade completa. Adiciona um mecanismo de *reflective retry* que recupera erros automaticamente.

84,5%	−75%	100%	2,4×
Acurácia em perguntas que cruzam múltiplas fontes (vs. 45,2% da solução convencional)	Redução na taxa de alucinação (7,1% vs. 28,5%)	Rastreabilidade por fonte em cada resposta	Payback médio em meses para casos de uso de alto impacto

Para quem faz sentido

Empresas com dados heterogêneos (bancos + documentos + APIs), perguntas que envolvem múltiplas fontes, exigência de auditoria e tolerância baixa a erros. Setores prioritários: jurídico e compliance, finanças, saúde, seguros, energia, governo.

Para quem não faz sentido (ainda)

Casos de uso simples (FAQ, atendimento básico), volumes muito baixos (menos de 1.000 consultas/mês), ausência de dados estruturados, ou equipes sem capacidade de operar sistemas de IA em produção.

> *A pergunta de decisão não é "devemos adotar RAG?". É:* ***"dos nossos casos de uso atuais, quais estão pagando o preço da alucinação sem que ninguém tenha medido?"***
>
> — O que este guia ajuda você a responder

PARTE 02

Por que agora?

Três forças convergem em 2026 para tornar esta a conversa central em IA empresarial.

O momento do mercado

1. O fim da fase "demo"

Entre 2023 e 2025, a maioria dos projetos de IA empresarial ficou em prova de conceito. Em 2026, conselhos de administração estão cobrando ROI documentado. Um thread de 461 pontos no Hacker News em março de 2026 perguntava: *"existe evidência de que agentic coding funciona?"*. A pergunta ecoa em todas as reuniões de steering committee. Demos não bastam mais — o mercado quer dados.

2. A maturação da stack

LangGraph, LlamaIndex, Qdrant, pgvector, Model Context Protocol (MCP) — todos atingiram GA entre 2024 e 2025. O que antes exigia meses de engenharia custom hoje é configuração. A janela entre "early adopter" e "commodity" está se fechando: quem não adotar arquiteturas avançadas nos próximos 12 meses vai comprar SaaS caro de terceiros depois.

3. O custo da alucinação ficou visível

Casos públicos de respostas incorretas geradas por IA em contextos legais, médicos e financeiros criaram pressão regulatória. A discussão deixou de ser "vamos medir depois" para "quantos erros aceitáveis por mil consultas?". A resposta em setores regulados é próxima de zero — um número que as soluções convencionais de IA corporativa não alcançam.

A anatomia da falha

Três histórias reais, anonimizadas, de empresas que implementaram assistentes de IA de primeira geração e descobriram os limites da abordagem.

CASO A — SEGURADORA, EQUIPE DE SINISTROS

Implantaram um assistente de IA sobre 400 mil documentos de apólices e laudos. Funcionou bem para perguntas diretas como *"qual é a franquia para furto?"*. Falhou em *"quantos sinistros do tipo X tivemos no último trimestre e o que as apólices dizem sobre cobertura em feriado?"* — a primeira parte exige o banco de sinistros, a segunda exige o documento contratual. O sistema inventava totais que "pareciam plausíveis". Desativado após 6 meses. Prejuízo: R$ 380 mil em implementação mais custo de oportunidade.

CASO B — ESCRITÓRIO DE ADVOCACIA, ÁREA DE COMPLIANCE

Implantaram um assistente inteligente sobre jurisprudência e contratos para consultas internas. Taxa de acerto em perguntas isoladas: 85%. Em perguntas que exigiam cruzar duas ou mais fontes: 41%. O escritório manteve o sistema, mas com um aviso: *"sempre cheque as fontes manualmente"*. Isso eliminou 70% do valor prometido — se você precisa checar tudo, não ganhou tempo.

CASO C — OPERADORA DE SAÚDE, PROTOCOLOS MÉDICOS

Assistente de IA sobre protocolos clínicos e histórico de atendimentos. Em testes, o sistema ocasionalmente retornava combinações de medicamentos contraindicadas porque misturava o contexto de dois pacientes diferentes na busca. O caso mais grave foi capturado em QA antes de ir a produção. O projeto foi congelado e reiniciado com uma arquitetura que garante rastreabilidade granular.

O padrão comum. Em todos os casos, os assistentes de primeira geração funcionaram em demonstrações e falharam exatamente nos casos de uso que justificariam o investimento. A causa raiz não é o modelo de linguagem — é a arquitetura da solução. Um único ponto de busca não consegue coordenar múltiplas fontes de raciocínio, e é exatamente disso que decisões corporativas precisam.

> *Pense em um estagiário brilhante que responde a tudo com um único "olhar rápido" em um só lugar. É assim que a maioria dos assistentes de IA corporativos está construída hoje. O que a sua empresa precisa é de um analista sênior.*

O custo real da alucinação

Quanto custa uma resposta errada? Depende do setor, mas aqui está uma referência honesta:

SETOR	CUSTO ESTIMADO POR INCIDENTE GRAVE
Jurídico (peça processual errada)	R$ 50 mil a R$ 2 milhões
Saúde (recomendação clínica incorreta)	R$ 200 mil a R$ 10 milhões + vidas
Finanças (decisão de crédito errada)	R$ 10 mil a R$ 500 mil
Compliance (relatório regulatório incorreto)	R$ 100 mil a R$ 5 milhões
Atendimento ao cliente (informação errada)	R$ 50 a R$ 5 mil

Em setores regulados, uma única alucinação grave paga o investimento inteiro em arquitetura hierárquica. Em setores menos críticos, o cálculo é sobre volume: um sistema que atende 100 mil consultas/mês com 5% de erro gera 5 mil respostas incorretas mensais. A pergunta não é "isso é muito?" — é "quanto vale cada uma dessas 5 mil respostas?".

PARTE 03

A arquitetura em linguagem de negócio

Uma analogia organizacional que funciona tanto em reunião técnica quanto em comitê executivo.

A analogia do analista sênior

Imagine que você contrata um analista sênior para responder perguntas complexas da empresa. Ele tem acesso ao banco de dados (ERP, CRM, sistema de sinistros), à base de documentos (contratos, políticas, manuais) e a APIs externas (cotações, regulações). Como esse analista trabalha quando recebe uma pergunta difícil?

1. **Ele pensa antes de agir.** Lê a pergunta, identifica as peças de informação que precisa, decide onde buscar cada uma.
2. **Ele delega para especialistas.** Pede ao DBA uma query, pede ao paralegal para localizar a cláusula, pede ao analista de mercado o dado atualizado.
3. **Ele sintetiza.** Junta as partes, verifica consistência, formula a resposta final.
4. **Ele recupera quando algo dá errado.** Se a query SQL retornar erro, ele refaz. Se o documento não for encontrado, ele tenta outra busca.
5. **Ele documenta as fontes.** A resposta vem com rastreabilidade — "esse número vem da query X, essa cláusula vem do contrato Y".

> *RAG simples é o estagiário que faz tudo em um passo só. RAG hierárquico agentic é o analista sênior em software.*

Os componentes principais

O Supervisor

Cérebro estratégico do sistema. Recebe a pergunta do usuário e toma três decisões: **decomposição** (quantos passos a pergunta exige), **roteamento** (qual worker é responsável por cada passo) e **síntese** (como combinar os resultados). O supervisor não acessa dados diretamente — ele apenas orquestra. É deliberadamente "burro" para fontes de informação; sua inteligência está na coordenação.

Em linguagem de negócio: o supervisor é o gerente que não executa, mas garante que tudo seja feito na ordem certa, pelas pessoas certas, e que o resultado final seja coerente.

O SQL Worker

Especialista em dados estruturados. Sabe ler o schema do banco, validar queries antes de executá-las, aplicar regras de segurança e retornar resultados formatados. Proteções nativas que você exigiria do seu DBA: queries parametrizadas (sem injeção), timeouts (nenhuma query trava o sistema), limites de linhas, política de schema configurável.

Em linguagem de negócio: é o DBA virtual que sabe as regras da casa e nunca escreve uma query perigosa.

O Vector Worker (ou Document Worker)

Especialista em documentos. Executa busca semântica combinada com busca léxica (BM25 + vetores densos). Filtra por relevância e retorna trechos pertinentes com metadados de origem. A busca híbrida aumenta significativamente a precisão em consultas jurídicas, médicas e técnicas onde termos exatos importam.

Em linguagem de negócio: é o paralegal virtual que entende o conceito da pergunta **e** sabe procurar por termos exatos em documentos.

O Reflective Retry — o diferencial silencioso

Esta é a parte que quase ninguém destaca, mas que separa projetos que funcionam de projetos que fracassam em produção.

> O PROBLEMA
>
> Em RAG simples, quando algo dá errado — query com erro de sintaxe, documento correto fora do retrieval, LLM com dados incompletos — o sistema **retorna uma resposta ruim sem avisar**. O usuário confia, age, e só descobre o erro depois.

A SOLUÇÃO HIERÁRQUICA

Cada worker sabe detectar quando falhou. Aciona um mecanismo de retry reflexivo: um nó especializado analisa o erro, propõe uma correção (por exemplo, reescrever a query com a sintaxe correta para aquele dialeto) e tenta de novo. Se falhar três vezes, o supervisor toma uma decisão consciente — retornar resposta parcial com aviso, ou não responder.

Em linguagem de negócio: é a diferença entre um analista que "acha que encontrou" e um analista que "verificou, deu errado, tentou de novo, e reportou o que conseguiu e o que não conseguiu". A segunda abordagem é a única que funciona em ambientes onde confiança é o produto.

O fluxo de uma pergunta real

Vamos acompanhar uma pergunta multi-hop do começo ao fim:

> *Quais fornecedores tiveram mais de 3 atrasos nos últimos 6 meses, e o que os contratos deles dizem sobre penalidades por atraso?*

Passo 1 — Supervisor recebe. Identifica duas partes: (a) dados de atrasos → SQL Worker e (b) cláusulas contratuais → Vector Worker. Decide executar em sequência — primeiro identifica os fornecedores, depois busca os contratos.

Passo 2 — SQL Worker. Introspecciona o schema, descobre a tabela `fornecimentos`, constrói uma query parametrizada com filtro de data e agrupamento. Executa e retorna: *"Fornecedor A: 5 atrasos. Fornecedor B: 4. Fornecedor C: 3."*

Passo 3 — Supervisor sintetiza parcial. Constrói três sub-perguntas para o Vector Worker: *"O que o contrato com Fornecedor A diz sobre penalidades por atraso?"* — e assim por diante.

Passo 4 — Vector Worker. Para cada fornecedor, executa busca híbrida filtrada por nome. Retorna trechos com metadados (arquivo, página, cláusula).

Passo 5 — Síntese final.

> *Três fornecedores excederam o limite: A (5 atrasos), B (4) e C (3). Cláusulas de penalidade: [link A], [link B], [link C]. Observação: o contrato do Fornecedor C vence em 60 dias, vale considerar antes de aplicar penalidade.*

Rastreabilidade: cada número tem link para a query que o gerou. Cada cláusula tem link para o parágrafo exato no PDF. Um auditor pode reconstruir toda a cadeia de raciocínio.

Com RAG simples, a mesma pergunta geralmente retorna algo como *"alguns fornecedores tiveram atrasos. Os contratos preveem multas conforme cláusula específica."* — genérico, inútil, e potencialmente inventado.

PARTE 04

Quando usar (e quando não)

Uma matriz de decisão honesta. Se você está sendo pressionado a adotar IA em todo lugar, esta seção é a sua defesa.

Sinais de que você precisa desta arquitetura

- [ ] **Dados heterogêneos.** Sua informação está em bancos estruturados, em documentos e em APIs — e as perguntas importantes cruzam essas fontes.
- [ ] **Perguntas em múltiplas etapas.** As consultas que importam exigem vários passos de raciocínio: filtrar, buscar, comparar, concluir.
- [ ] **Auditoria obrigatória.** Setor regulado (LGPD, BACEN, ANS, ANVISA) e necessidade de justificar cada resposta com fonte rastreável.
- [ ] **Tolerância baixa a erro.** Uma resposta errada pode causar prejuízo material, legal ou reputacional.
- [ ] **Volume relevante.** Mais de 5 mil consultas por mês ou mais de 50 usuários ativos no sistema.
- [ ] **Assistente de IA convencional já foi tentado e decepcionou.** Você conhece os limites por experiência própria.
- [ ] **Equipe técnica madura.** Engenheiros capazes de operar sistemas em produção — ou um parceiro que os tem.

ESCALA DE DECISÃO

4 ou mais marcados: esta abordagem merece uma conversa séria. Inicie um discovery.
2 ou 3: pode ser prematuro. Considere otimizar a solução atual ou validar com um piloto de baixo custo.
0 ou 1: um assistente de IA convencional (ou mesmo um LLM bem-orientado) provavelmente atende. Não complique.

Sinais de que você ainda não precisa

FAQ puro

Se 90% das perguntas são variações de 30 perguntas conhecidas, use busca com refinamento automático. Você não precisa de uma arquitetura com agentes especializados.

Volume muito baixo

Menos de 1.000 consultas por mês raramente justifica a complexidade. Foco em qualidade manual, não em automação.

Documentos homogêneos, sem dados estruturados

Se tudo está em PDF e não há banco de dados para cruzar, uma solução convencional de busca inteligente provavelmente basta.

Equipe sem capacidade operacional

Se seu time de TI não tem ninguém capaz de operar sistemas em produção com observabilidade, o projeto vai falhar na operação — não na arquitetura.

Orçamento apertado sem prioridade estratégica

Se você está fazendo "porque é IA", pare. Encontre primeiro um caso de uso com ROI claro.

Matriz de decisão rápida

DIMENSÃO	ASSISTENTE CONVENCIONAL	ARQUITETURA HIERÁRQUICA
Complexidade da pergunta	Direta, um passo	Múltiplas etapas, múltiplas fontes
Tipo de dado	Documentos homogêneos	Heterogêneos (bancos + docs + APIs)
Tolerância a erro	Alta (informativo)	Baixa (decisão crítica)
Rastreabilidade	Desejável	Obrigatória
Volume	< 5k consultas/mês	> 5k consultas/mês
Orçamento piloto	R$ 20–60 mil	R$ 80–250 mil
Tempo até o primeiro valor	2–4 semanas	8–12 semanas
Maturidade do time	Básica	Intermediária/avançada

PARTE 05

ROI e métricas de sucesso

Se você não tem baseline, você não tem projeto. Esta é a parte que o CFO vai ler primeiro.

Antes de falar de ROI, uma regra: você não está comprando uma tecnologia — está contratando um "analista sênior em software" que trabalha 24 horas por dia, acessa todas as fontes da empresa e documenta cada resposta com a fonte. É assim que você deve modelar o retorno.

O que medir

MÉTRICAS DE QUALIDADE — O QUE IMPORTA PARA O USUÁRIO

MÉTRICA	O QUE MEDE	META COM ARQUITETURA CORRETA
Acurácia em perguntas encadeadas	% de respostas corretas em questões complexas	≥ 80%
Taxa de resposta inventada	% com informação incorreta apresentada como verdadeira	≤ 10%
Rastreabilidade	% de respostas com fonte verificável	100%
"Não sei" correto	% de vezes que o sistema admite não ter resposta	≥ 95%
Satisfação do usuário (CSAT)	NPS específico do sistema	≥ 70

MÉTRICAS DE OPERAÇÃO — O QUE IMPORTA PARA A TI

MÉTRICA	O QUE MEDE	META TÍPICA
Tempo de resposta (pior caso)	Tempo máximo em 95% das consultas	≤ 4 segundos
Disponibilidade	Uptime do sistema	≥ 99,5%
Custo por consulta	R$ de IA + infraestrutura por pergunta	R$ 0,05 – 0,50
Auto-recuperação de erros	% de falhas recuperadas sem intervenção	≥ 80%

Modelo simplificado de ROI

Fórmula básica:

```
ROI = (Benefícios anuais - Custo anual) / Custo anual
```

Exemplo concreto — compliance jurídico

Cenário: 20 analistas que levam em média 40 minutos por consulta. Volume: 15 mil consultas/ano. Custo hora do analista: R$ 150.

SITUAÇÃO ATUAL (SEM IA)

Tempo gasto: 15.000 × 40 min = **10.000 horas/ano**
Custo analista: 10.000 × R$ 150 = **R$ 1,5 milhão/ano**
Erros de compliance custosos: 3/ano × R$ 300 mil = **R$ 900 mil/ano**
Total: R$ 2,4 milhões/ano

SITUAÇÃO COM A NOVA PLATAFORMA

Tempo por consulta: 40 min → 8 min (analista valida resposta)
Custo analista: **R$ 300 mil/ano**
Custo da plataforma: **R$ 180 mil/ano** (R$ 15k/mês)
Erros de compliance: 0,5/ano × R$ 300 mil = **R$ 150 mil/ano**
Total: R$ 630 mil/ano

> *Economia anual: R$ 1,77 milhão. Investimento inicial: R$ 350 mil.* ***Payback em 2,4 meses. ROI ano 1: 406%.***
>
> — Exemplo conservador em caso de uso de alto impacto

Onde os cálculos de ROI geralmente erram

Erro 1 — Subestimar o custo de operação

Plataformas de IA em produção custam mais do que em prova de conceito. Orçamento de operação deve incluir monitoramento, alertas, revisão humana por amostragem e treinamento contínuo do sistema.

Erro 2 — Superestimar o ganho de tempo

Se a IA responde em 5 segundos mas o analista ainda precisa validar, o ganho real é o tempo de validação comparado ao tempo de pesquisa — não o tempo total.

Erro 3 — Ignorar a curva de adoção

No primeiro mês, a adoção é baixa. Ganhos reais começam entre o mês 3 e o mês 6. Calcule o payback considerando essa curva.

Erro 4 — Não modelar o custo das respostas erradas

Se você não incluir o custo de respostas incorretas no modelo, a comparação é desonesta — a solução mais simples parece mais barata do que é.

PARTE 06

Roadmap em 4 fases

Nenhum projeto de IA empresarial deveria começar com "vamos comprar uma plataforma". Este roadmap protege você de erros caros.

Fase 1 — Discovery

2 a 3 semanas

Objetivo: mapear casos de uso reais, dados disponíveis e capacidade do time. Nenhuma linha de código.

ENTREGÁVEIS

- Inventário de casos de uso candidatos, priorizados por valor × viabilidade.
- Mapa de fontes de dados (estruturadas e não estruturadas) com qualidade, acessibilidade e governança.
- Avaliação da equipe: quem pode operar, quem precisa de capacitação, onde há lacunas.
- Matriz de riscos inicial (LGPD, vendor lock-in, compliance).
- Baseline das métricas atuais (tempo, taxa de erro, satisfação).
- Decisão "go / no-go" para a Fase 2 com critérios objetivos.

Custo típico: R$ 20 a R$ 50 mil com consultoria especializada.
Armadilha: pular esta fase "porque já sabemos o que queremos". 80% dos projetos que pulam escolhem o caso de uso errado.

Fase 2 — POC estruturada

4 a 6 semanas

Objetivo: construir prova de conceito do menor caso de uso viável, com métricas comparáveis ao baseline.

ENTREGÁVEIS

- Sistema funcional em ambiente controlado (ainda não produção).
- Supervisor + dois agentes especialistas (banco de dados + documentos) cobrindo um subconjunto de dados.
- Conjunto de teste com ≥ 50 perguntas representativas, julgadas por especialistas do domínio.
- Relatório comparativo: solução convencional × arquitetura hierárquica × baseline manual.
- Demonstração ao-vivo para stakeholders.

Custo típico: R$ 60 a R$ 150 mil.
Critério de sucesso: acurácia > 75% E alucinação < 15%. Se não bater, não avance.

Fase 3 — Piloto em produção

6 a 10 semanas

Objetivo: operar em produção com grupo controlado de usuários reais. Medir adoção, qualidade e operação.

ENTREGÁVEIS

- Sistema em produção com observabilidade completa (logs, métricas, alertas).
- Grupo piloto de 20 a 50 usuários reais.
- Revisão humana amostral (≥ 5% das respostas na primeira semana).
- Ciclo de feedback para ajustar os agentes e a orquestração.
- Dashboard de KPIs para stakeholders; relatório mensal × baseline.

Custo típico: R$ 80 a R$ 200 mil + operação mensal.
Critério de sucesso: NPS ≥ 50 no piloto E redução comprovada de tempo/erro E uptime ≥ 99%.

Fase 4 — Rollout e operação

Contínuo

Objetivo: expandir gradualmente para a base completa de usuários, com melhoria contínua.

ENTREGÁVEIS

- Ondas de rollout (por área, geografia ou caso de uso).
- Operação contínua com SLA definido.
- Governança: comitê mensal de revisão, políticas de atualização, plano de resposta a incidentes.
- Atualizações trimestrais da base de conhecimento e dos modelos.
- Avaliação anual de ROI real × projetado.

Custo típico: R$ 15 a R$ 60 mil/mês.
Armadilha: achar que "pronto, tá em produção, não precisa mais de atenção". Sistemas de IA degradam sem manutenção.

Timeline consolidado

PERÍODO	FASE	MARCO
Mês 1	Discovery	Casos de uso priorizados + baseline
Mês 2–3	POC estruturada	Primeiro valor mensurável
Mês 4–6	Piloto em produção	Primeiro valor em produção
Mês 7+	Rollout e operação	Escala completa

PARTE 07

Riscos e mitigação

Uma matriz honesta é a diferença entre navegar imprevistos e fracassar por surpresas previsíveis.

Matriz de riscos

#	RISCO	PROB.	IMPACTO	MITIGAÇÃO
1	Vazamento via LLM externo	Média	Crítico	LLM local ou API com DPA e não-treinamento.
2	Resposta inventada em decisão crítica	Alta*	Crítico	Arquitetura hierárquica + auto-recuperação de erros + revisão humana amostral.
3	Vendor lock-in em plataforma	Alta	Alto	Stack open-source (LangGraph, Qdrant) e padrões abertos (MCP).
4	Degradação do modelo	Média	Médio	Monitoramento contínuo + re-avaliação trimestral.
5	Resistência dos usuários	Alta	Alto	Envolvimento no piloto, treinamento, comunicação clara.
6	Custo operacional fora do previsto	Média	Médio	Limites por usuário/área; monitoramento de custo por query.
7	Time interno não consegue operar	Média	Alto	Transferência de conhecimento estruturada.
8	Compliance LGPD em dúvida	Média	Crítico	DPO envolvido desde o Discovery; on-premises para dados sensíveis.
9	Latência inaceitável em pico	Baixa	Médio	Teste de carga na Fase 3; caching de frequentes.
10	Falha de provedor de LLM	Baixa	Alto	Multi-provider (fallback entre 2+).

* Alta sem arquitetura hierárquica. Baixa com arquitetura correta.

LGPD e compliance — o que importa

Dados em repouso

Se os documentos indexados contêm dados pessoais, o vector database precisa estar em ambiente com controles adequados — criptografia, auditoria de acesso, retenção controlada.

Dados em trânsito

Se você usa LLM de terceiros (OpenAI, Anthropic, Google), cada consulta envia trechos de documentos pela rede. Exija DPA, residência de dados e garantia contratual de não-treinamento no seu conteúdo.

Direito ao esquecimento

Se um titular exerce direito de exclusão, seu sistema precisa remover os dados dele não apenas do banco original, mas também do vector database e dos caches. Inclua isso no processo desde o início.

Rastreabilidade de decisões automatizadas

A LGPD (art. 20) garante o direito à revisão de decisões automatizadas. A arquitetura hierárquica atende naturalmente esse requisito por oferecer rastreabilidade por fonte.

Alternativa radical: Local AI

Em casos de extrema sensibilidade (saúde, jurídico, governo), considere LLMs on-premises ou em nuvem soberana. A arquitetura hierárquica funciona tanto com LLMs externos quanto com modelos locais (Llama 3, Mistral, Qwen).

Governança desde o dia 1

Comitê de IA responsável. Não espere ter problemas para criar. Reunião mensal com TI, jurídico, compliance, área de negócio e um champion da IA.

Políticas escritas: quais casos de uso são permitidos? Qual é o processo quando um usuário reporta um erro? Como tratamos respostas contendo dados sensíveis? Quem pode adicionar fontes à base de conhecimento?

Auditoria periódica. A cada trimestre, amostre 100 respostas e revise com especialistas do domínio. Compare com o trimestre anterior. Ajuste.

PARTE 08

Build vs Buy

A pergunta aparece cedo em todo projeto. A resposta é "depende" — mas existe um framework para decidir bem.

As três opções

Opção A — SaaS pronto

Plataformas como Glean, Notion AI Enterprise, Amazon Q. Pagamento por usuário, integração acelerada, menos controle.

Opção B — Stack open-source self-hosted

LangGraph + Qdrant + Ollama (ou API de LLM). Controle total, custo de operação, exige equipe madura.

Opção C — Consultoria especializada + stack custom

Parceiro constrói a solução com stack open-source, transfere conhecimento e opera em modelo misto. Equilíbrio entre velocidade e controle.

Critérios de decisão

CRITÉRIO	SAAS	OPEN-SOURCE	CONSULTORIA
Time de engenharia maduro	Não	Sim	Parcial
Casos de uso padronizados	Sim	—	—
Dados altamente sensíveis	Cuidado	Sim	Sim
Orçamento CAPEX preferido	Não	Sim	Sim
Prazo agressivo (< 3 meses)	Sim	Difícil	Sim
Diferencial competitivo na IA	Não	Sim	Sim
Volume muito alto	Caro	Mais barato	Mais barato
Integração profunda com sistemas próprios	Limitado	Total	Total

A ARMADILHA DO SAAS GENÉRICO

Muitas empresas escolhem SaaS achando que é "mais simples" e descobrem depois que: o custo por usuário fica inviável na escala; a integração com sistemas internos é limitada; os dados acabam em infraestrutura que o jurídico não aprova; não há acesso à arquitetura hierárquica real — apenas assistentes convencionais com marketing diferente.

Regra prática: se seu caso de uso é estratégico, envolve dados sensíveis ou exige customização, SaaS é quase sempre o caminho errado no médio prazo. Use SaaS para prototipagem, não para operação estratégica.

A stack open-source recomendada (2026)

- **Orquestração:** LangGraph — supervisor + workers com state management determinístico.
- **Vector DB:** Qdrant ou pgvector (se já usa PostgreSQL).
- **Busca híbrida:** BM25 via OpenSearch/Elasticsearch + embeddings.
- **LLM:** OpenAI GPT-4/5 via API (com DPA), Anthropic Claude (com DPA), ou Llama 3 / Mistral on-premises.
- **Embeddings:** OpenAI, Cohere ou modelos locais (BGE, E5).
- **Observabilidade:** LangSmith, Helicone, ou stack própria com OpenTelemetry.
- **Avaliação:** RAGAS, TruLens ou frameworks custom.

Tudo tem documentação pública, comunidade ativa e caminho de evolução. Nenhum componente é proprietário de um único vendor.

PARTE 09

Checklist de avaliação

Use em reuniões de discovery, avaliação de fornecedores ou revisão de propostas.

Avaliação do caso de uso

- [ ] O problema atual está medido? (Baseline de tempo, erro, satisfação)
- [ ] O caso de uso envolve múltiplas fontes de dados?
- [ ] As perguntas típicas são em múltiplas etapas ou diretas?
- [ ] Há tolerância a erro definida? Qual?
- [ ] O valor por pergunta está estimado?
- [ ] Há volume suficiente para justificar a complexidade?
- [ ] Há stakeholders claros e patrocínio executivo?

Avaliação de dados

- [ ] Os dados estruturados estão acessíveis via API ou query direta?
- [ ] Os documentos estão em formato processável (PDF com texto, não imagem)?
- [ ] Há processo de atualização dos dados?
- [ ] A qualidade dos dados é conhecida? (Duplicação, inconsistência, lacunas)
- [ ] Há governança sobre quem pode acessar o quê?
- [ ] LGPD foi avaliada? DPO foi envolvido?

Avaliação do time e operação

- [ ] Temos engenheiros capazes de operar sistemas distribuídos?
- [ ] Temos processo de observabilidade (logs, métricas, alertas)?
- [ ] Há capacidade de revisão humana amostral na operação?
- [ ] Há plano de treinamento para usuários?

- [ ] Há canal de feedback claro?
- [ ] Há orçamento de operação contínua, não só de implementação?

Avaliação de fornecedores

- [ ] O fornecedor mostra arquitetura hierárquica real ou só marketing?
- [ ] Há referências públicas de projetos em produção similares?
- [ ] O contrato permite portabilidade de dados e modelos (anti-lock-in)?
- [ ] Há SLAs de disponibilidade e acurácia?
- [ ] Há auditoria externa de segurança?
- [ ] O fornecedor usa LLMs próprios ou de terceiros? Com que garantias?
- [ ] Há plano claro de resposta a incidentes?
- [ ] Há transferência de conhecimento estruturada?

Avaliação de riscos

- [ ] Matriz de riscos formal foi construída?
- [ ] Cada risco tem owner e plano de mitigação?
- [ ] Há plano de rollback caso o piloto falhe?
- [ ] Há cenário de descontinuidade (o que fazer se o fornecedor sair do ar)?
- [ ] O comitê de IA responsável está formado?

PARTE 10

Glossário

Os termos essenciais para conversar de igual para igual com sua equipe técnica.

Agente (agentic)

Componente de software que toma decisões baseadas em contexto, podendo usar ferramentas e executar múltiplos passos. Diferente de uma função determinística.

Alucinação

Resposta gerada por LLM que contém informação incorreta apresentada como verdadeira. O problema central que arquiteturas avançadas buscam reduzir.

Busca híbrida

Combinação de busca léxica (por palavras-chave, tipicamente BM25) e busca semântica (por vetores densos). Captura tanto coincidência exata quanto similaridade conceitual.

Chunk

Trecho de um documento (tipicamente 500 a 2000 caracteres) usado como unidade de indexação no vector database.

DPA (Data Processing Agreement)

Contrato que define responsabilidades do processador de dados (ex.: OpenAI) perante o controlador (sua empresa).

Embedding

Representação numérica (vetor) de um texto, usada para busca semântica.

GraphRAG

Variação de RAG que usa knowledge graph além de vetores. Boa para perguntas relacionais complexas. Não substitui RAG hierárquico agentic, pode compor com ele.

LLM

Large Language Model — modelo de linguagem de grande porte (GPT-4, Claude, Llama, Mistral, etc.).

Multi-hop

Pergunta que exige múltiplos passos de raciocínio para ser respondida.

MCP (Model Context Protocol)

Padrão aberto (Anthropic, 2024) para integrar agentes de IA com ferramentas externas. Adoção crescente.

RAG (Retrieval Augmented Generation)

Arquitetura que combina recuperação de documentos relevantes com geração de linguagem por LLM.

Reflective retry

Mecanismo pelo qual um sistema detecta seu próprio erro, analisa a causa e tenta de novo com uma correção. Diferencial da arquitetura hierárquica.

Supervisor

Componente que orquestra workers especializados, decompõe perguntas e sintetiza resultados.

Vector database

Banco de dados otimizado para armazenar e buscar vetores de embedding. Exemplos: Qdrant, Pinecone, Weaviate, pgvector.

Worker

Componente especializado em uma modalidade (SQL, documentos, APIs) acionado pelo supervisor.

Quer conversar?

A Stickybit é uma consultoria especializada em IA empresarial de alto impacto. Construímos sistemas que sobrevivem em produção — não demos. Nosso método combina discovery rigoroso, POCs com métricas comparáveis a baseline, implementação com stack open-source e transferência de conhecimento estruturada.

Se este guia gerou perguntas específicas sobre o seu contexto, agende uma conversa de 45 minutos sem compromisso. Vamos mapear os seus casos de uso mais promissores e honestamente dizer se faz sentido avançar.

stickybit.com.br/rag-hierarquico-agentes

Este guia é fruto de análise independente baseada em literatura pública (InfoQ, benchmarks EntQA, documentação de frameworks open-source) e experiência prática da Stickybit em projetos com clientes. Os dados de ROI, custos e prazos são referências de mercado — o seu caso específico deve ser avaliado individualmente.