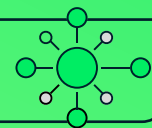




Database Digest

ISSUE 02



A camada de
inteligência
unificada:
viabilizando
a era agêntica



© 2026 MongoDB, Inc.
Todos os direitos reservados.

Publicado desde 2025
pela MongoDB,
1633 Broadway, 38th Floor
New York, NY 10019, EUA

Editores colaboradores: Boris Bialek,
Sebastian Rojas Arbulu, Daniel Jamir

Criação de marca:
Andrew Considine, Hans Bennewitz

Reconhecimentos

Soluções para a indústria: Taylor Hedgecock, Gabriela Preiss, Genevieve Broadhead, Peyman Parsi, Humza Akhtar, Francesc Mateu, Jeff Needham, Benjamin Lorenz, Thorsten Walther, Rama Gabbita, Ken Wiebke, Diego Albano, Ainhua Múgica, Raphael Schor

Relações com os desenvolvedores: Apoorva Joshi, Franck Pachot, Mikiko Bazeley

Educação: Raghu Viswanathan, Rita Rodrigues, Thalita Carrico, Emily Pope

Marketing de Produto: Jad Jarouche

Marketing corporativo global: Joanne Johnson, Laila McDonald, Kevin O'Rourke

Arquitetos de Soluções: Diana Annie Jenosh, Ashwin Gangadhar

Marketing internacional: May Petry, Penta Stanley, Cathy Gallego, Ilana Matro

Centro de Demanda Global: Steve Jurczak, Jack Yallop, Brandon Lee

Carta do Editor



Boris Bialek

Vice-presidente
de soluções para o
setor da MongoDB

Em 35 anos no ramo, nunca vi nada igual a agora. Normalmente, alguns setores fazem grandes mudanças, enquanto outros observam, se perguntando se precisam agir. Desta vez, não. Bancos, setor automotivo, setor de seguros, manufatura e muito mais estão avançando ao mesmo tempo, e a maioria deles, como um cliente me disse, está apostando todas as fichas.

Isso levou a algumas decisões peculiares de arquitetura. No início do ano, trabalhei com um cliente cuja prova de conceito para um fluxo de trabalho único e de agente limitado envolveu 14 fornecedores diferentes. Ele tinha bastante orgulho disso. Perguntei, tranquilamente, como ele planejava colocar tudo em produção. Ele olhou para mim como se nunca tivesse pensado nisso e disse: “Ainda não chegamos a essa parte.” Mostrei a ele um concorrente que já estava em produção após oito semanas. Ele se calou.

Esse breve momento privado é, a meu ver, o momento definidor da IA agêntica em 2026: uma equipe inteligente descobrindo que criou uma ótima demo que não pode ser transformada em negócio.

Nosso presidente e diretor executivo, CJ Desai, disse no MongoDB.local de Londres, em maio de 2026: “A parte mais difícil de administrar os agentes em produção não é o modelo. É a camada de dados por baixo dele.” Ninguém na sala precisou de mais explicações.

Ele tinha motivos para ser direto. A Gartner espera que 40% dos projetos de IA agêntica sejam cancelados até 2027. A Deloitte descobriu que apenas 11% das empresas usam sistemas agênticos em produção, o que significa que 89% ainda estão aquém disso. O motivo, quase sem exceção, não é o modelo. É o que está por trás de tudo: segurança adicionada tardiamente, trilhas de auditoria que ninguém planejou, dados em tempo real que ficam em quatro sistemas ao mesmo tempo e chegam ao agente meio passo atrasados.

Considere a alternativa. O maior distribuidor farmacêutico dos Estados Unidos movimenta 1,2 bilhão de contêineres por ano, e por trás de

cada um desses contêineres há um paciente aguardando o medicamento certo na dosagem certa no dia certo. A Lei de Segurança da Cadeia de Suprimentos de Medicamentos dos EUA (Drug Supply Chain Security Act) não se importa com a inteligência do seu modelo, e sim com a sua capacidade de rastrear qualquer contêiner, de ponta a ponta, em tempo real, o tempo todo. Alguns anos atrás, a McKesson fez uma escolha arquitetônica e discreta: aposentar uma stack legada monolítica e passar a operar sobre uma plataforma de dados unificada. Eles ampliaram as operações em 300 vezes. Ainda são responsáveis pela segurança dos pacientes. Essa confiança não foi conquistada por um agente, mas pelo que fica por baixo dele.

Ao longo de toda a edição, defendemos a necessidade dessa base sólida. A seção 01 mede o custo real da arquitetura fragmentada e o que a McKesson criou no lugar. A seção 02 aborda o que substitui a fragmentação: a camada de inteligência unificada, demonstrada nos setores onde ela apresenta resultados positivos em primeiro lugar. A seção 03 questiona como deixar os agentes sobre a sua base realmente precisos, e a LG U+ oferece uma prévia de como é isso na prática. Na seção 04, chegamos ao clímax: a transição de sistemas de registro para sistemas de ação.

Vou encerrar com um momento que me marcou. No dia em que a equipe do CTO de um grande varejista entrou em operação, ele me disse que o projeto que criamos juntos era obsoleto. Por um instante, fiquei de coração partido. Então, ele explicou: “Boris, é obsoleto porque está em operação. Mas os dados eram do MongoDB. Essa é a minha base. Posso trocar o LLM e o modelo a qualquer momento. Posso trocar o framework por cima. Os dados são meus.”

O que aprendemos nessas grandes mudanças é que as escolhas arquitetônicas feitas agora durarão mais do que as ferramentas de IA para as quais foram escolhidas. No fim das contas, é por isso que a camada de dados é a decisão que vale a pena acertar.





SEÇÃO UM

Quando a IA supera a Stack

O custo oculto da arquitetura fragmentada.

p. 6

SEÇÃO DOIS

A camada de inteligência unificada

Uma única plataforma para todo o ciclo de vida da recuperação.

p. 15

SEÇÃO TRÊS

Precisão em escala

Fundamentando agentes de IA na verdade operacional em tempo real..

p. 23

SEÇÃO QUATRO

Construindo a era da IA

Dos sistemas de registro aos sistemas de ação.

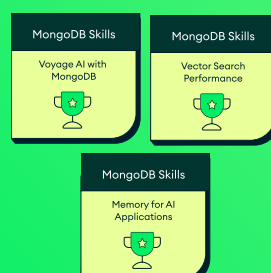
p. 31



LIVRO EM DESTAQUE

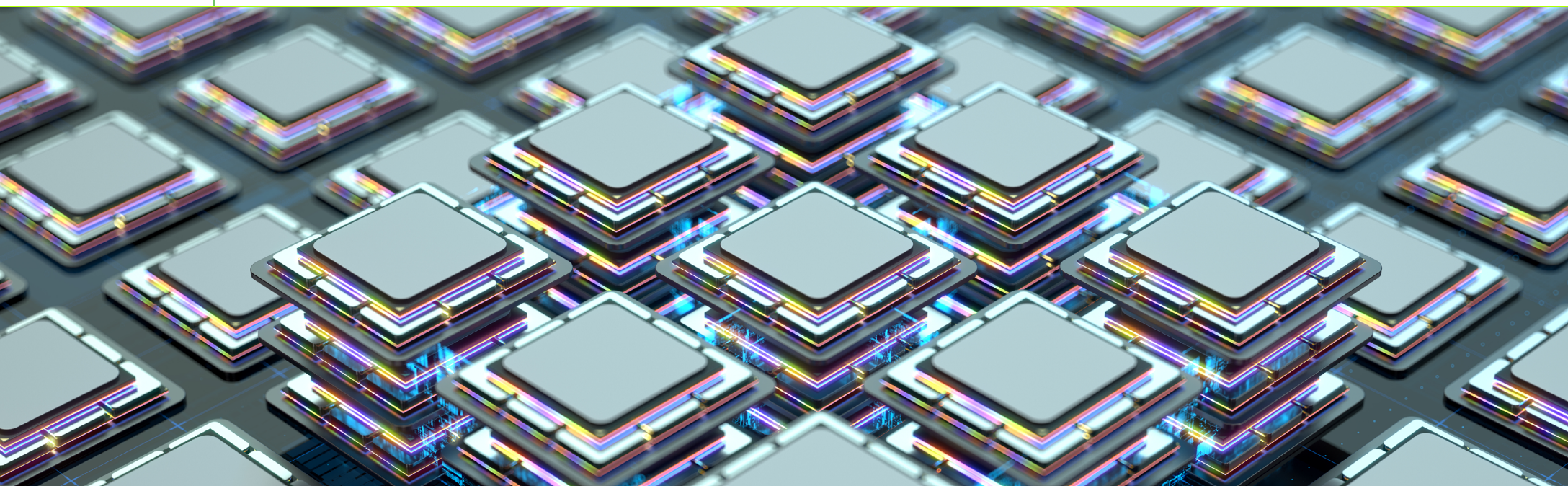
Architectures for the Intelligent AI-Ready Enterprise

Frameworks práticos para criar arquiteturas prontas para IA que oferecem valor duradouro aos negócios.



APRENDIZAGEM E CERTIFICAÇÃO

Três novos skill badges de IA do MongoDB: Voyage AI com MongoDB, Desempenho do MongoDB Vector Search e Memória para Aplicações de IA.



SEÇÃO UM

Quando a IA supera a stack

O custo oculto da arquitetura fragmentada

Duas empresas iniciam o mesmo projeto de IA no mesmo dia.

Até o fim do trimestre, um deles terá colocado um agente funcional em produção. É fundamentado nos dados operacionais da empresa.

Tem memória entre as sessões. Foi aprovado na revisão de segurança. O custo é previsível.

Dezoito meses depois, o segundo continua na fase piloto. O agente funciona no ambiente de demo. Gera alucinações no ambiente de staging. O armazenamento vetorial e o armazenamento operacional começaram a divergir – e ninguém sabe o responsável pelo problema.

O departamento de compras está questionando se a fatura da nuvem deveria mesmo ser essa.

O jurídico está perguntando como auditar algo que ninguém sabe descrever completamente.

O mesmo talento. Os mesmos modelos. O mesmo orçamento. A única variável é a arquitetura inicial.

Essa lacuna agora tem um nome: chamamos de **arrasto arquitetônico**, o peso cumulativo que uma arquitetura fragmentada impõe a todas as equipes que tentam implantar IA sobre ela. [Uma nova pesquisa](#) da International Data Corporation (IDC), comissionada pela MongoDB e conduzida com 1.400 organizações em oito mercados da Ásia-Pacífico, chegou a um cálculo: 43% das organizações dizem que a arquitetura existente é um grande obstáculo na criação de novas aplicações sem uma modernização extensa. Os sistemas são rígidos, caros e lentos demais. E o custo de não fazer nada aumenta: a IDC prevê que as organizações que não conseguirem lidar com a dívida técnica terão uma taxa 50% maior de falhas na IA até 2027.

Essa taxa de falhas não é um problema na qualidade do modelo. É o que acontece quando os agentes encontram uma stack que foi feita para recuperar, não para agir. O chatbot lê. O agente lê, decide, grava, transaciona e muda de estado – e depois grava mais um pouco, pois cada decisão que o agente toma deixa um rastro. Logs de decisões. Estados intermediários. Registros de auditoria. A memória do agente, que precisa sobreviver à próxima sessão. O arrasto é o que acontece quando uma pilha montada para responder a perguntas é solicitada a fazer tudo de uma só vez.

Isso atinge quatro lugares ao mesmo tempo. O armazenamento vetorial não pode

gravar, mas o agente precisa mudar de estado. O armazenamento operacional e o armazenamento vetorial não podem compartilhar uma transação, mas a ação do agente deve ser atômica, ou é parcialmente concluída: o cliente é cobrado, o pedido não é reservado. A sincronização entre eles atrasa, e o agente decide com base na verdade de ontem. E, quando um regulador pergunta o que o agente fez, a trilha de auditoria abrange quatro das seis etapas, pois nenhum sistema viu a sequência inteira.

A lacuna entre as duas empresas não é o talento ou o orçamento. É que um deles modernizou a camada de dados ao criar o agente. A outra tentou depois e perdeu um ano com o trabalho de integração que deveria ter acontecido primeiro.

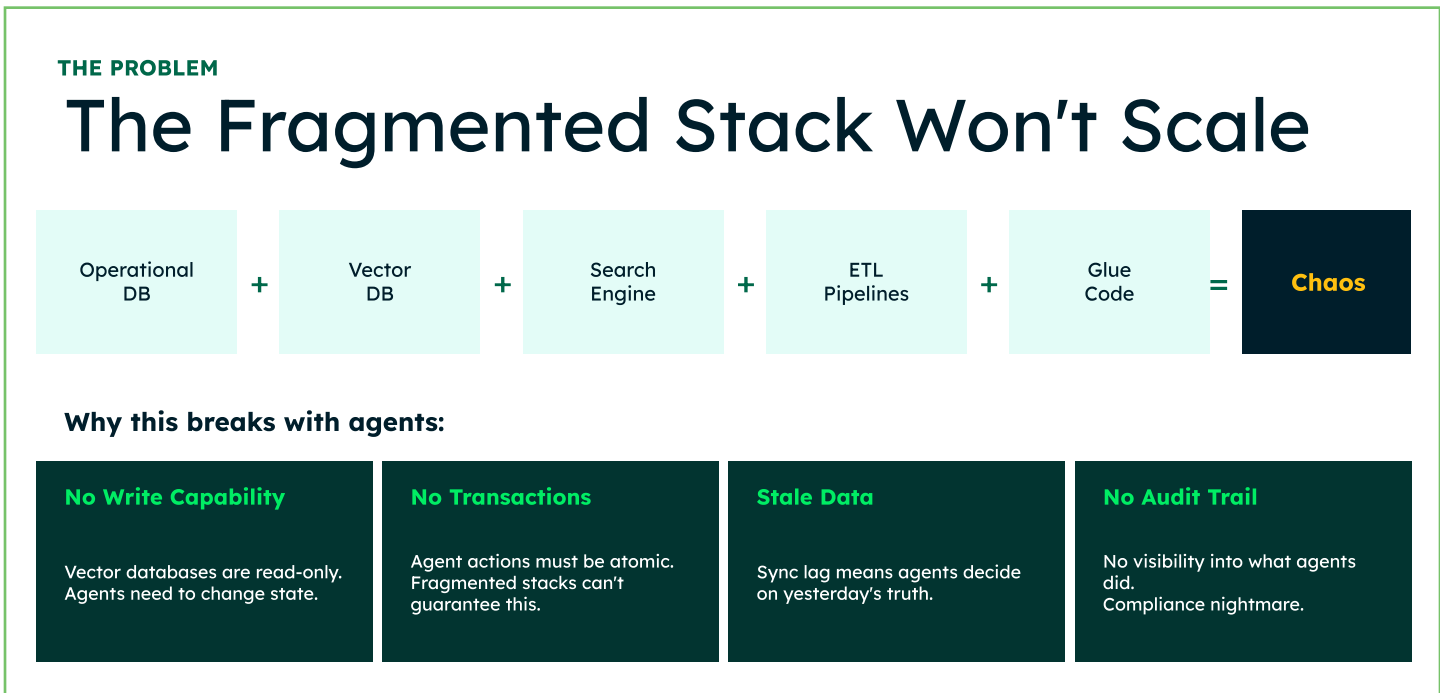
Ele não aparece como um item do orçamento, mas como um roteiro mais lento, análises de segurança mais dolorosas e os projetos de IA que somem silenciosamente da próxima revisão trimestral.



A empresa quer avançar rapidamente, mas os sistemas por baixo dizem não. Algo que deveria levar duas semanas leva seis meses.

Thorsten Walther *Diretor executivo da CXO Advisory Asia do MongoDB*

Figura 1.
A stack fragmentada e por que ela rompe com os agentes



O restante da seção analisa o problema em duas partes.

Primeiro, o diagnóstico. Por que arquiteturas divididas – armazenamentos vetoriais e operacionais separados, conectados por pipelines de extração, transformação e carregamento (ETL) – impõem um custo que só aparece em larga escala. Por que a mesma incompatibilidade aparece no nível do modelo de dados sempre que as cargas de trabalho de documento são forçadas nos mecanismos relacionais. E uma

estrutura de decisão honesta para escolher entre Postgres e MongoDB, de um arquiteto que passou 30 anos nos dois lados desse debate.

Em seguida, o exemplo de uma empresa que abordou essa questão. A McKesson movimentou 1,2 bilhão de contêineres de medicamentos por ano e decidiu desde cedo que a camada de dados era a parte da stack que valia a pena acertar. O que eles criaram, e o que mudou por causa dessa criação, é o estudo de caso que encerra a questão.



**LEIA A
PESQUISA
DA IDC**

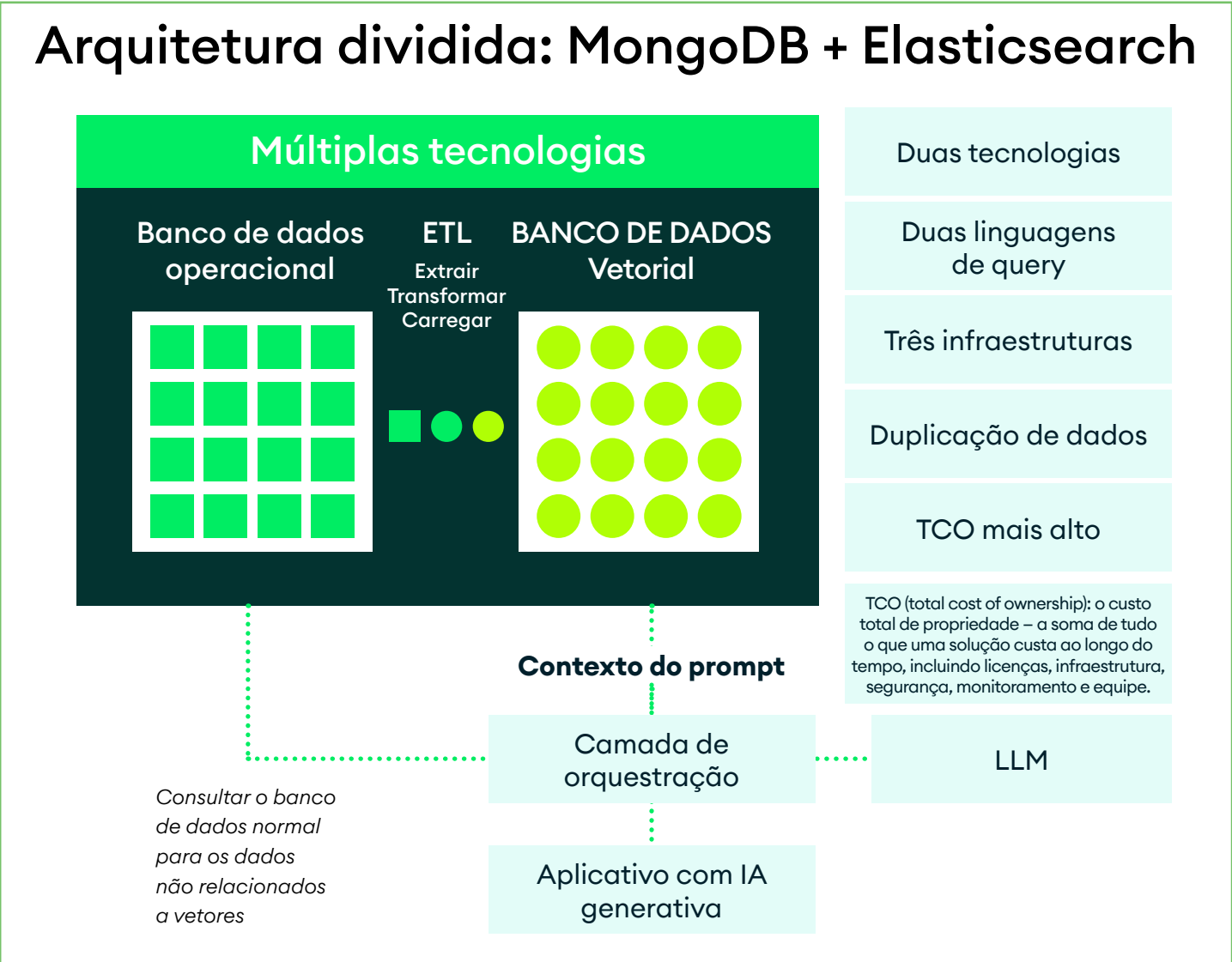
Arquitetura de banco de dados estratégico para IA: unificada x dividida

Toda revisão de arquitetura em um projeto de IA agêntica termina na mesma pergunta: a stack que já temos já cobre o que vamos construir? É tentador responder a essa pergunta comparando o desempenho dos bancos de dados vetoriais. A questão mais profunda é saber se sua arquitetura consegue tratar, em uma única consulta e com as mesmas garantias, tanto os dados quanto o contexto semântico.

É isso que um agente realmente precisa. É a parte da stack que a maioria das equipes resolve por último, depois que o resto do sistema já ganhou forma. O banco de dados operacional é tratado como um dado. De acordo com a [Harvard Business Review Analytic Services](#), apenas 15% das organizações consideram seu alicerce de dados pronto para a IA agêntica. O resultado é uma arquitetura dividida que parece razoável no quadro branco e começa a desmoronar na produção.

Vale a pena comparar abertamente os dois padrões. Na arquitetura unificada, uma única plataforma lida com dados operacionais e pesquisa vetorial em conjunto. Na arquitetura dividida, um armazenamento vetorial dedicado (Pinecone, Weaviate ou um mecanismo de pesquisa como Elasticsearch) fica ao lado do banco de dados operacional, com um pipeline ETL mantendo os dois sincronizados. Um emparelhamento comum, mostrado na Figura 2, é o MongoDB como o sistema de registro para documentos de produtos e o Elasticsearch como o mecanismo de pesquisa vetorial para os embeddings. Ambos funcionam na demonstração. As diferenças só aparecem quando você escala.

Figura 2.
Arquitetura dividida:
banco de dados
operacional MongoDB +
armazenamento vetorial
Elasticsearch



Os pontos reais de falha da abordagem dividida são previsíveis. A complexidade de sincronização aumenta a cada novo campo. Quando um documento é excluído do armazenamento operacional, o embedding vetorial correspondente pode permanecer como um documento fantasma, retornando resultados que não existem mais. As transações atômicas não são opção, pois dois sistemas não podem compartilhar um limite de transação.

Os custos operacionais se acumulam de forma igualmente silenciosa. A velocidade do desenvolvedor sofre porque a maior parte do esforço de engenharia é direcionada ao código de reconciliação e não às funcionalidades. Estratégias separadas de backup, procedimentos separados de failover, stacks separadas de monitoramento e escalas de plantão separadas. E a conta chega

todo mês: duas tecnologias de banco de dados, duas linguagens de query, três infraestruturas para proteger e pagar, dados duplicados e custo total de propriedade mais alto.

O caso unificado não é abstrato. Quando o documento e o embedding ficam juntos, uma atualização é uma gravação atômica, uma exclusão remove ambos, e os mesmos controles de acesso protegem ambos. Plataformas como o MongoDB Atlas, que adotam a abordagem unificada, usam nós de pesquisa dedicados para isolar cargas de trabalho vetoriais da camada operacional para serem dimensionadas de forma independente. Unificação não é abrir mão da margem de desempenho que as cargas de trabalho vetoriais exigem. O custo total de propriedade reflete essa simplicidade: uma plataforma para pagar, uma única plataforma para pagar, proteger e operar.

Figura 3. Arquitetura dividida: o custo oculto.

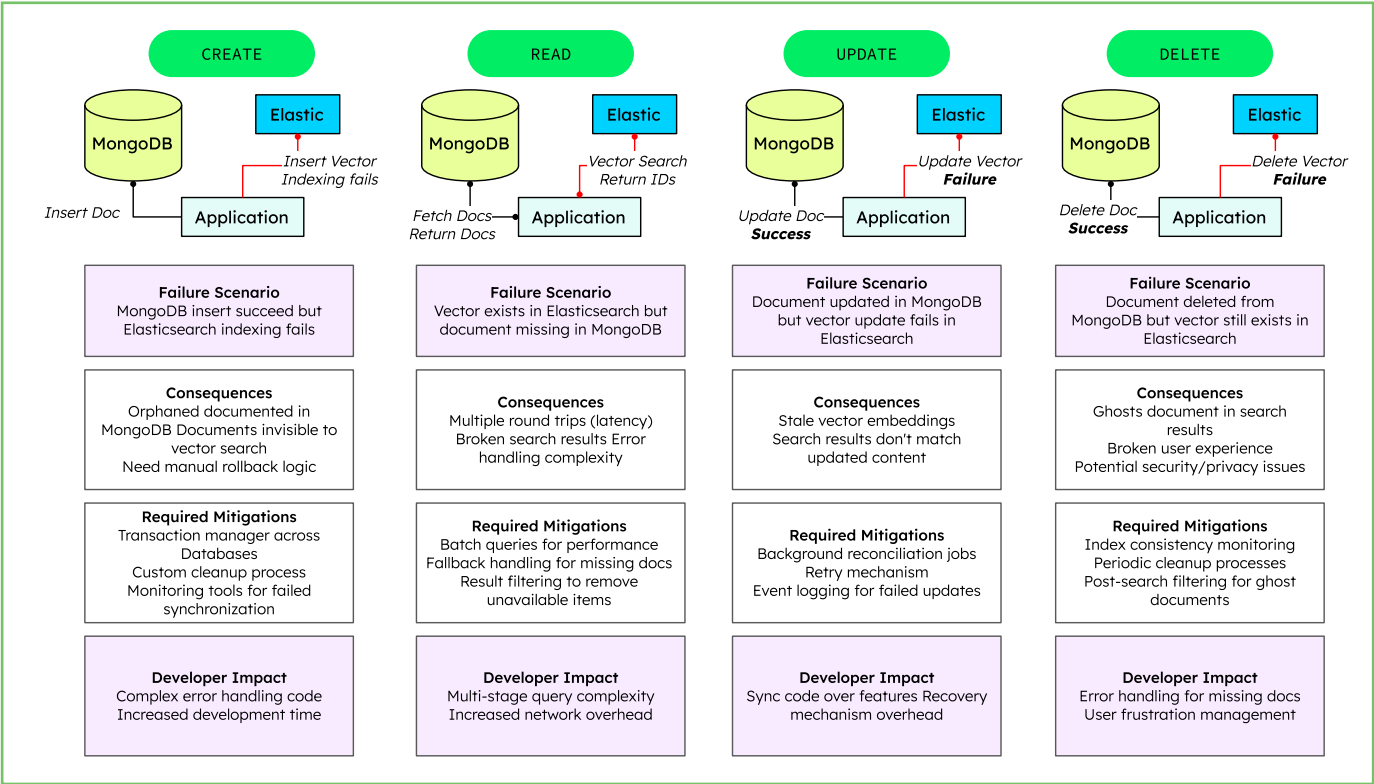
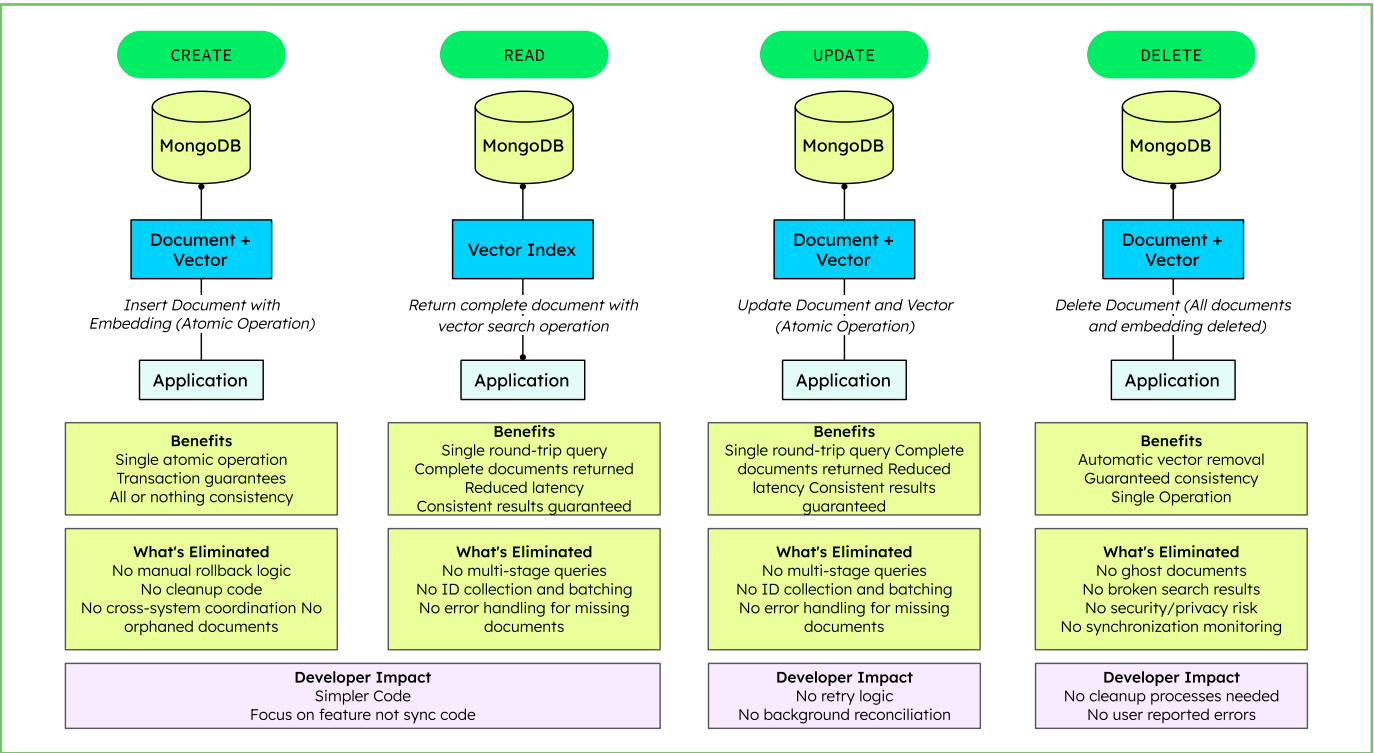


Figura 4. Operações CRUD em uma arquitetura unificada: MongoDB Atlas com pesquisa vetorial.



LEIA O ARTIGO



MongoDB x SQL: além dos mitos

A stack web moderna fala uma linguagem: JSON (JavaScript Object Notation), o formato de texto leve que navegadores, APIs e aplicativos móveis usam para trocar dados. O navegador envia JSON. A API processa objetos JavaScript. O aplicativo móvel espera JSON. Filas de mensagens, WebSockets, microsserviços: tudo JSON. Escolha um banco de dados cujo modelo de dados corresponda ao resto da stack, e uma categoria inteira de trabalho desaparece. Escolha um que não corresponda, e sua equipe passará o resto da vida traduzindo entre formatos. É o que Tim Carter Clausen, que escreve como [The Decipherist](#) e opera o MongoDB em produção há uma década, escreve em um artigo que vale a pena ler na íntegra.

A cada poucos meses, uma publicação no blog viraliza: “Por que voltamos para o Postgres.” Os comentários se enchem dos mesmos pontos. *O MongoDB não dimensiona. Não há junções. As transações são fracas.* A resposta de Clausen não é apontar os pontos fortes de ambos os bancos de dados. Ele aborda cada crítica em sequência, com dez anos de dados de produção.

Dois temas interessam aos líderes técnicos que avaliam uma decisão quanto à stack em 2026.

O primeiro é o pipeline JSON. Uma típica query SQL faz oito alterações de formato em uma única viagem de ida e volta:

1. O cliente envia JSON.
2. A API analisa um objeto JavaScript.
3. Uma ferramenta de mapeamento objeto-relacional (ORM) decompõe esse objeto em linhas espalhadas por tabelas normalizadas.
4. O banco de dados faz seu trabalho.
5. As linhas são remontadas.
6. São mapeadas de volta em um objeto.
7. O objeto é serializado de volta para JSON.
8. A resposta é enviada.

O equivalente do MongoDB são quatro transformações, e Clausen afirma que quatro é um exagero, pois a forma dos dados nunca muda de fato. Cada mudança de formato ao longo do fluxo SQL é CPU, memória, latência e um lugar onde o bug pode se esconder. Nada disso melhora o produto. Afinal, tudo tem que continuar funcionando.

Em segundo lugar, vem a evolução do esquema. Renomear um campo, adicionar um objeto aninhado ou reestruturar um documento acontece ao vivo no MongoDB, com tempo de inatividade zero. Em uma grande tabela relacional, a mesma operação pode bloquear gravações por minutos, às vezes horas. Para equipes que lançam semanalmente, a diferença não é teórica. É a diferença entre adicionar um campo à tarde e agendar um período de manutenção no próximo trimestre.

O diagnóstico de Clausen é direto. A equipe que culpa o MongoDB pelas falhas geralmente é a que o construiu como o SQL: coleções modeladas como tabelas, Mongoose usado como ORM, chamadas de `find()` espalhadas por arquivos, nada desnormalizado, nada indexado. Em seguida, uma publicação de 3.000 palavras sobre como o banco de dados é o problema. “Não é”, escreve ele. “O problema são eles.”

Nada disso quer dizer que os mecanismos relacionais sejam obsoletos. Eles continuam sendo a opção certa para muitas cargas de trabalho, e as grandes empresas continuarão a executá-los por décadas. A questão é mais específica. Quando a carga de trabalho tem formato JSON, é orientada a documentos e itera semanalmente, as suposições por trás do design relacional deixam de ser neutras. O que levanta outra questão: os mecanismos relacionais modernos realmente preenchem essa lacuna, ou apenas a encobrem? Postgres com suporte nativo a JSON é o principal exemplo.



**LEIA O
ARTIGO**

Postgres ou MongoDB?

A árvore de decisões do arquiteto

Há uma objeção óbvia: mecanismos relacionais modernos agora aceitam colunas JSON, índices vetoriais e campos de array. Uma implantação do Postgres com JSONB e pgvector pode levar uma empresa para a era da IA sem forçar uma migração da camada de dados, certo? A pergunta é justa e vale a pena ser respondida por mérito técnico, não por lealdade tribal. Franck Pachot, veterano de 30 anos da Oracle, Postgres, YugabyteDB e MongoDB, além de AWS Data Hero e Oracle Certified Master, aborda exatamente essa questão em uma palestra que vale a pena acompanhar na íntegra. O framework dele não é partidário; ele recomenda Postgres em algumas cargas de trabalho mesmo em uma função no MongoDB. Esse equilíbrio é o que faz valer a pena.

Duas observações técnicas mudam a comparação da preferência para a arquitetura.

A primeira é o que acontece em um documento de 10 MB em cada mecanismo. Insira-o no MongoDB, e um rastreamento do Linux mostra chamadas de gravação de 10 MB cada uma, porque o documento permanece junto como uma folha de B-tree. Insira-o no Postgres, e o mesmo rastreamento mostra gravações de 8 KB, porque o Postgres divide o documento em blocos de 8 KB usando o TOAST. A segunda é o que “uma única tabela” realmente significa. Uma tabela Postgres com uma coluna JSON não é uma tabela nos bastidores. São duas tabelas e dois índices: a tabela principal com o índice de chave primária e uma tabela de partes TOAST com o próprio índice. A busca de um documento grande é, mecanicamente, uma junção de loops aninhados entre ambos.

Figura 5.
A realidade física de uma tabela Postgres que contém uma coluna JSON. Uma declaração CREATE TABLE, quatro objetos por trás: uma tabela de tuplas, uma tabela de partes TOAST e um índice para cada um.



A mesma divergência aparece na camada de indexação. Considere uma consulta familiar para alguém que administra um sistema operacional: devolva os últimos 10 pedidos de um determinado produto em um determinado país. No modelo de documento, um único índice composto resolve isso diretamente, incluindo os campos aninhados dentro de arrays. Em um sistema relacional com os mesmos dados que JSON, o equivalente normalmente requer um índice GIN para o conteúdo do array, um índice B-tree separado para os campos escalares e uma classificação que o planejador não pode evitar. O plano lê mais linhas do que o necessário e apresenta desempenho desfavorável à medida que o conjunto de dados cresce. Nada disso fica visível na camada da aplicação. Tudo fica visível na fatura da nuvem.

A perspicácia estrutural por trás de ambos os pontos é a localidade dos dados. No modelo de documento, o modelo lógico e o modelo físico são iguais. A forma que a aplicação grava é a forma que o banco de dados armazena, e a forma que o banco de dados armazena é a forma que a camada de recuperação retorna a um agente de IA. Essa equivalência não é um slogan. É o que permite que os dados, o índice e o caminho de recuperação sejam um só, sem os pipelines de sincronização, lógica de dupla escrita e trabalhos de reconciliação que, de outra forma,

arquiteturas fragmentadas precisam se aproximar do mesmo resultado.

O framework de decisão de Pachot aponta para as observações técnicas. Um banco de dados central que atende a muitos aplicativos, onde os casos de uso ainda não são todos conhecidos, é uma tarefa relacional, e o Postgres é a resposta relacional certa. Uma única aplicação criada com base em um modelo de domínio e persistida conforme a aplicação o interpreta é uma tarefa de documento, e é aí que o MongoDB entra. Ambos os mecanismos são respostas diretas a perguntas diferentes. A pergunta a fazer a seguir é qual deles descreve o trabalho que está sendo construído hoje. São microserviços com contextos limitados, esquemas que iteram semanalmente, ou objetos de domínio persistidos da maneira que a aplicação pensa sobre eles?

A conclusão de Pachot merece ser citada em qualquer conversa sobre banco de dados em 2026: “Um banco de dados é bom e rápido se você usar corretamente, e o mais importante é escolher um que você conheça ou que queira aprender.”

A responsabilidade volta para onde pertence: ao arquiteto, não ao mecanismo. A próxima seção mostra como é essa responsabilidade quando a carga de trabalho é a segurança dos pacientes na escala de 1,2 bilhão de contêineres por ano.



**ASSISTA
À PALESTRA**





McKesson garante compliance em larga escala com o MongoDB

Os argumentos de arquitetura são abstratos até que o maior distribuidor de medicamentos dos Estados Unidos aposte a segurança dos pacientes neles.

Quando a Lei de Segurança da Cadeia de Suprimentos de Medicamentos dos EUA entrou totalmente em vigor, a McKesson enfrentou um mandato inegociável: rastrear anualmente mais de 1,2 bilhão de números de série na cadeia de suprimentos, com cada varredura autenticando um contêiner até o fabricante, em tempo real. A configuração atual deles, SAP e Postgres, tinha dificuldades com a flexibilidade e a escala exigidas pela regulamentação.

A estrutura do principal arquiteto de TI da McKesson, Brian Schmidt, foi caracteristicamente operacional: os dados que acompanham cada contêiner de medicamento são tão críticos quanto o próprio medicamento. Medicamentos que chegam sem dados válidos e entregáveis não podem ser liberados para o paciente que precisa deles.

A McKesson escolheu o MongoDB como base para dois novos sistemas. O **Repositório Central de Dados** gerencia os dados de números de série dos 350 mil clientes que a McKesson atende diariamente. O **Repositório Distribuído de Números de Série** é uma aplicação distribuída que lida com as chamadas de validação em toda a rede de centros de distribuição da McKesson. O modelo de documento se mostrou ideal para os dados hierárquicos da cadeia de suprimentos, onde uma única remessa pode conter dezenas de produtos,

cada um com a própria procedência, lote, data de validade e histórico de rastreabilidade. Modelar isso em tabelas planas traz um grafo de junções que seria um problema de latência por si só. Modelar como documentos significou que o formato dos dados refletia o objeto de negócio.

A descrição do próprio Schmidt sobre como entrar em produção mereceu um destaque na seção: “O mapa não mostrou nenhum problema.” Para uma empresa top 10 da Fortune 500, operando sob um prazo federal de conformidade e fornecendo medicamentos para um terço do país, nenhum problema no mapa é uma conquista técnica significativa. É a diferença entre uma implementação regulatória bem-sucedida e uma falha pública com consequências para a segurança dos pacientes.

A conclusão aqui não é que o setor de saúde é excepcionalmente difícil (embora seja). É que a compliance em larga escala e a agilidade arquitetônica não são forças opostas. A McKesson modernizou-se para compliance, não para IA. A base que a equipe criou é o que eles agora usam para explorar a IA generativa e a próxima classe de casos de uso. A opção arquitetônica que viabilizou a conformidade em larga escala é a mesma que viabiliza de fato a próxima carga de trabalho.

“A escala que alcançamos com o MongoDB é impressionante. É um momento de muito orgulho para todos nós.”

Upendra Kulkarni, Gerente principal de produtos da McKesson



[LEIA A HISTÓRIA](#)



[ASSISTA AO KEYNOTE](#)

Do arrasto arquitetônico à camada de inteligência unificada

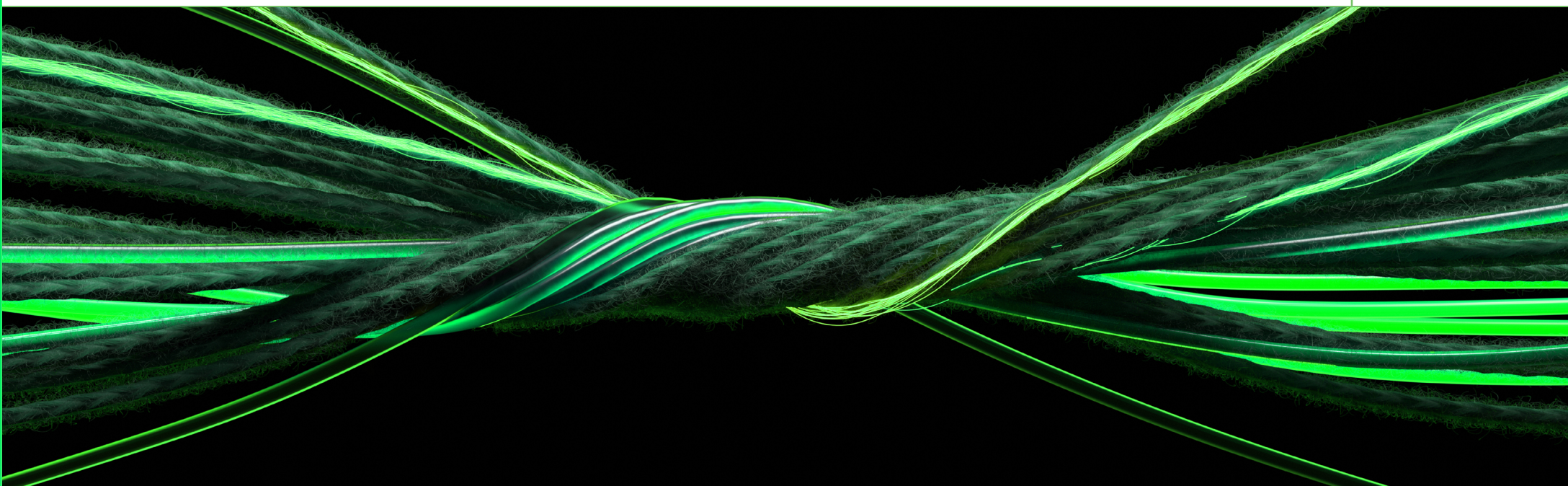
A Seção 01 foi o diagnóstico. O arrasto arquitetônico é real, a pesquisa do IDC o confirma, e a McKesson mostra como reduzir isso em larga escala. Porém, o diagnóstico sem um plano de tratamento é apenas uma auditoria.

A Seção 02 revela o plano de tratamento.

Uma vez que você aceita que juntar armazenamentos separados e adaptar cargas de trabalho de documentos em tabelas relacionais impõe o mesmo custo em moedas diferentes, as perguntas se tornam práticas:

1. Como é a camada de inteligência unificada?
2. Quais cargas de trabalho ela precisa absorver?
3. Como é operar uma quando sua empresa depende de agentes que atuam na produção todos os dias?

Nas próximas dez páginas, responderemos às três perguntas, apresentadas nos setores em que o padrão unificado traz retorno primeiro: manufatura, varejo, serviços financeiros e automotivo.



SEÇÃO DOIS

A camada de inteligência unificada

Uma única plataforma para o ciclo completo de recuperação.

A camada de inteligência unificada não é uma ideia. É uma lista de funcionalidades que precisam estar no mesmo lugar. Esta seção detalha um de cada vez, mostrando por que cada um é importante quando o agente precisa agir com base no que diz.

O padrão arquitetônico em torno do qual este volume foi construído tem um nome: **Converged Datastore para IA agêntica (um armazenamento de dados convergente)**. As entidades sobre as quais uma empresa

opera, os embeddings que os agentes de IA analisam e o estado que esses agentes acumulam ao longo das interações ficam no mesmo lugar: sob uma única API, uma única linguagem de consulta e um único modelo de segurança. Dados operacionais, pesquisa vetorial, pesquisa full-text search, geração de embeddings, memória do agente e processamento de fluxo deixam de ser seis problemas de integração para serem uma única plataforma.

Os modelos continuarão melhorando. Os frameworks continuarão em constante rotatividade. A camada de dados é a parte que precisa durar mais que ambos. Arquitete bem, e as trocas de modelos e as migrações de framework serão refatorações em vez de reconstruções. Arquitete mal, e cada nova onda de ferramentas de IA força a migração da plataforma.

A Seção 01 abordou esse caso no nível da arquitetura: dividida x unificada, com o arrasto arquitetônico recaindo sobre quem optou pela divisão. O mesmo contraste se desenvolve na própria carga de trabalho da IA, funcionalidade por funcionalidade.

MongoDB Data Platform



MCP



Database



Text



Vector



Embedding
models

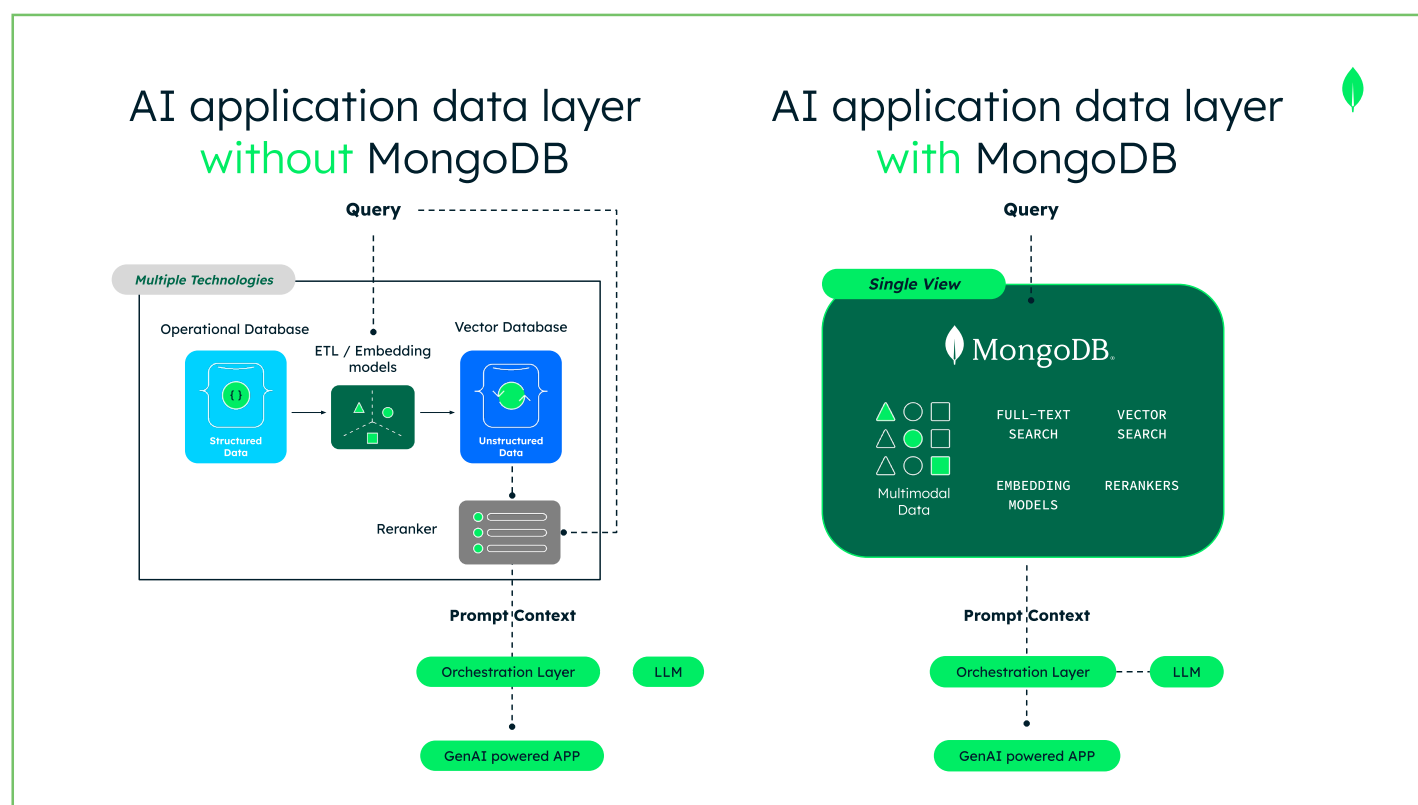


Rerankers



Local
dev

Figura 6.
A camada de dados da aplicação de IA, antes e depois da consolidação. À esquerda, múltiplas tecnologias integradas via pipelines ETL: banco de dados operacional, modelos de embedding, banco de dados vetorial e reranker. À direita, as mesmas funcionalidades nativas de um único repositório.



O que se segue é uma visão geral de quatro setores. A própria IA é agnóstica ao domínio: um robô de solda relata torque, um investigador de fraudes pesquisa transcrições, um técnico associa um conector corroído a uma árvore de falhas, um agente recupera o histórico de pedidos de um cliente. Vocabulários diferentes, usuários diferentes, a mesma base por trás. Então, esta seção percorre deliberadamente os setores, mostrando o que muda na carga de trabalho e o que não muda.

Começamos com a manufatura, onde o padrão Unified Namespace já existe na maioria dos chãos de fábrica e precisa apenas de memória para ficar pronto para o agente. Migramos para o varejo, onde o comércio unificado resolve um problema de silo de dados de uma década e, além disso, viabiliza o agente de IA que lida com consultas de suporte. Em seguida, os

serviços financeiros, onde uma plataforma que alimenta a pesquisa, a similaridade semântica, a análise de grafos e o aprendizado de máquina simultaneamente é a única resposta verdadeira para o crime financeiro em velocidade de máquina em velocidade de máquina. Fechamos com o setor automotivo, onde o diagnóstico pós-venda é um dos poucos lugares em que uma camada de inteligência unificada traz retorno dentro de um único ciclo de produto.

O fio condutor entre os quatro é o mesmo: a consolidação compensa em lugares que parecem não relacionados até que você olhe para o diagrama de arquitetura.

A Seção 01 terminou com uma pergunta: o que realmente se ganha ao pagar para reduzir o arrasto arquitetônico? A Seção 02 tem a resposta.



**LEIA O BLOG
CONVERGED DATASTORE**



**ASSISTA À ENTREVISTA
DA REDMONK**

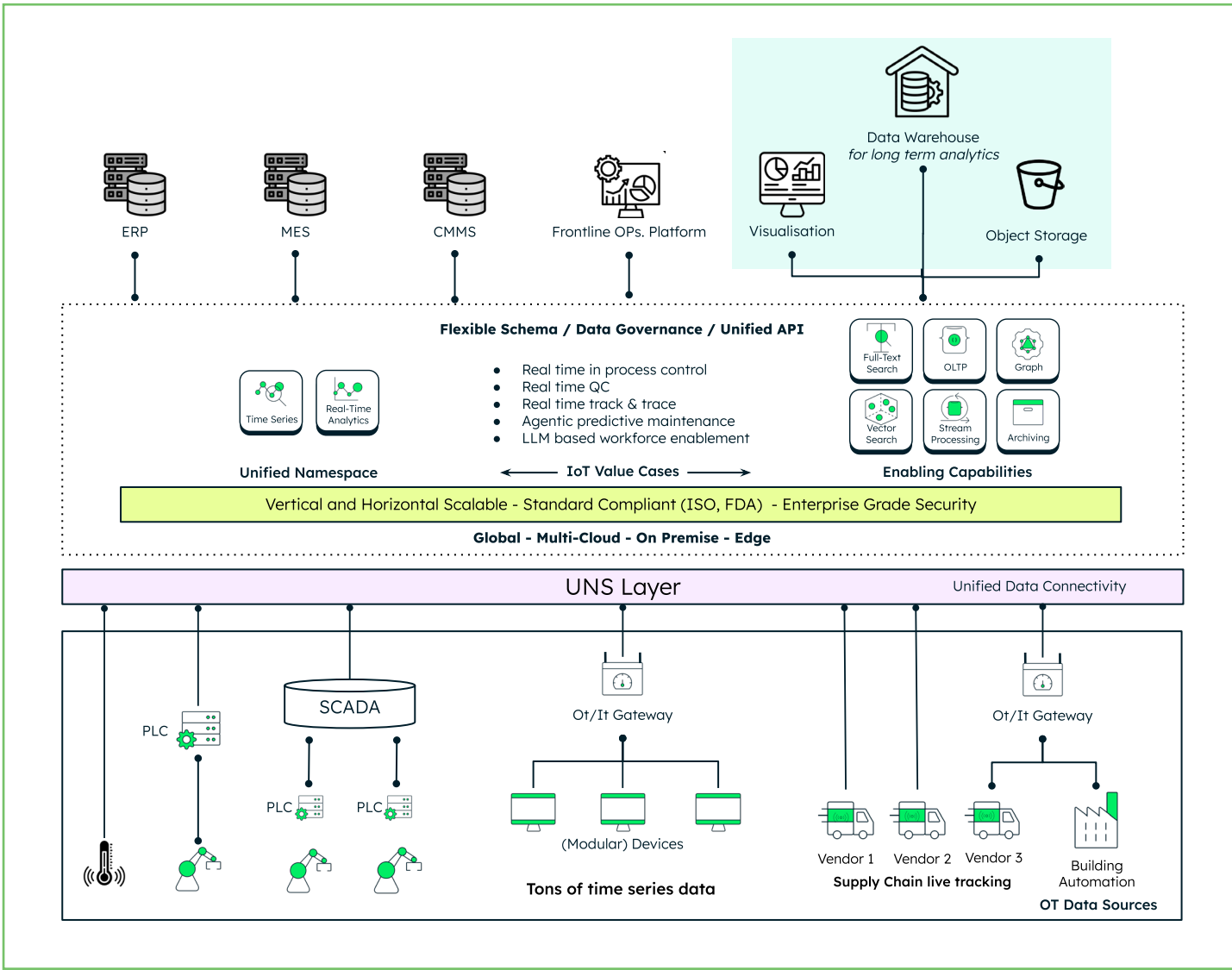
A camada de dados na manufatura precisa de memória para IA agêntica

O padrão Unified Namespace existe na maioria dos chãos de fábrica modernos, às vezes com nomes diferentes. É uma única hierarquia semântica no qual cada controlador, sensor e sistema de registro publica dados de produção ao vivo, e todo assinante lê no mesmo lugar. É a resposta arquitetônica para uma geração de integrações ponto a ponto que ninguém quer manter. Mas o namespace por si só não levará os fabricantes para a era agêntica. O namespace é transitório. O agente precisa de memória.

Os controladores lógicos programáveis (CLPs), os pequenos computadores que operam máquinas individuais – juntamente com sensores e sistemas de controle supervisório – publicam

no namespace; o sistema de execução de manufatura (MES), o planejamento de recursos empresariais (ERP) e os sistemas de qualidade se inscrevem nele. Isso basta para operar uma fábrica moderna. Não basta para executar um agente sobre ela. Um agente que responde a uma variação de tolerância, programa manutenções não planejadas ou coordena uma troca de linha não precisa apenas do estado atual do ambiente. São necessários os incidentes anteriores, os dados de qualidade da semana passada, as anotações do operador do último turno e os embeddings dos manuais de manutenção consultados na última vez. O namespace publica o estado atual. A memória guarda o passado.

Figura 7. Arquitetura unificada vinculando dados OT via namespace unificado a uma plataforma de dados industrial dimensionável e em tempo real.



É aqui que o Mong memória. Os mesmos documentos que contêm a telemetria operacional também contêm embeddings, histórico de conversas e o rastreamento de auditoria das decisões do agente. A leitura de torque de um robô de solda fica ao lado da ordem de trabalho, do número da peça, do operador e do resultado de qualidade. Então, quando a linha de soldagem mostra um desvio no torque em três turnos, o agente pode extrair todas as soldas anteriores com assinaturas similares de desvio, compará-las com os registros de manutenção e notas do operador, e abrir uma ordem de trabalho de manutenção antes da próxima inspeção com falha. Ele indica o componente suspeito, o técnico com a certificação correta e uma janela

na agenda de produção quando a linha pode ser parada.

As funcionalidades que viabilizam isso vêm do mesmo mecanismo. [As coleções de séries temporais](#) lidam com os fluxos de sensores de alta cardinalidade. [A pesquisa vetorial](#) lida com a recuperação semântica de manuais e relatórios de incidentes. [O Stream Processing](#) lida com o enriquecimento em tempo real conforme os dados chegam. Juntas, essas funcionalidades transformam o namespace de um feed em uma fundação. A próxima carga de trabalho de IA que a fábrica adicionar cai nele, não em um novo.



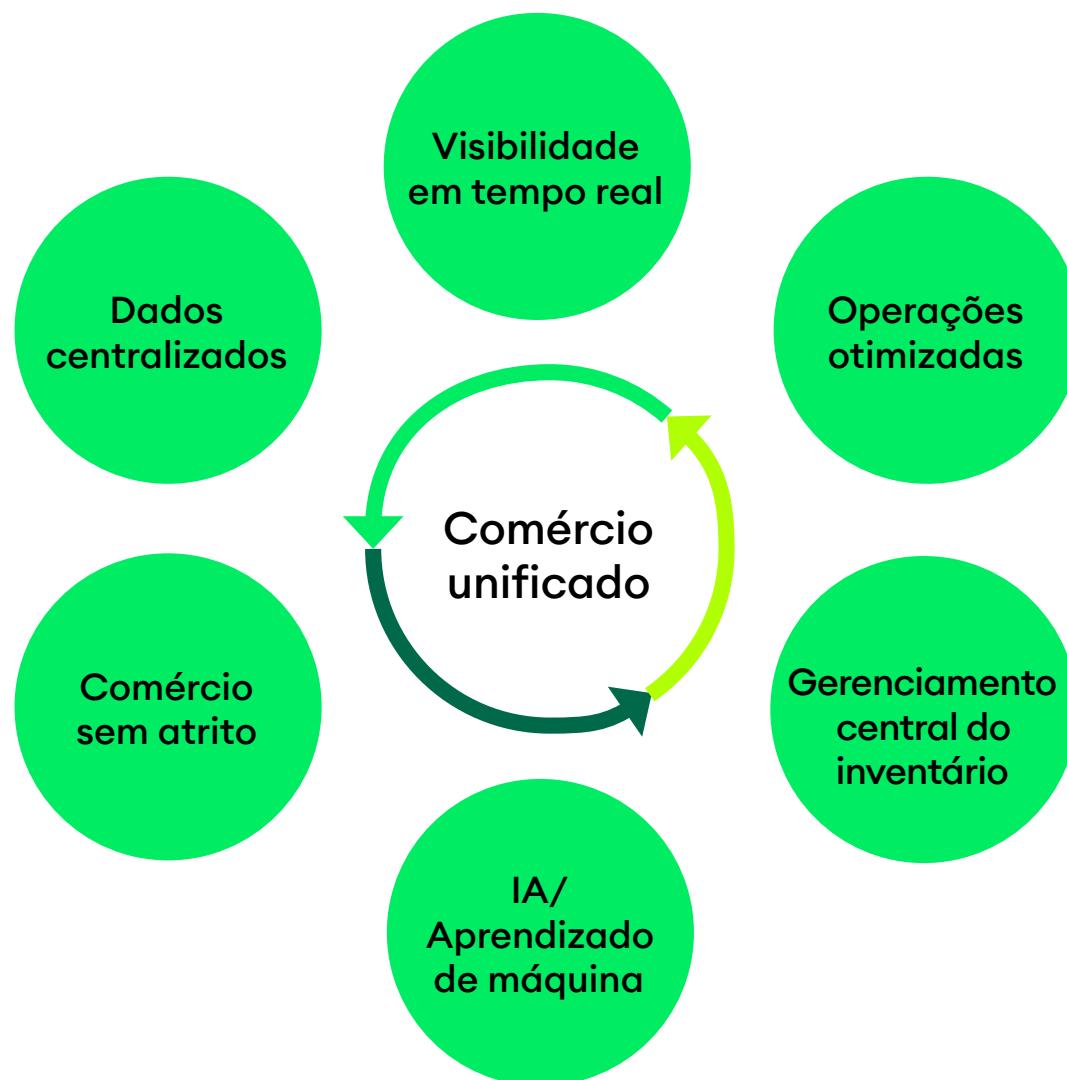
**BAIXE O
WHITE PAPER**



Comércio unificado com pesquisa inteligente

As equipes de varejo passaram a maior parte da última década tentando oferecer uma experiência única na web, aparelhos móveis, loja física e central de atendimento. Os dados sempre estiveram lá, mas existiam em vários sistemas diferentes. O resultado foi um cliente que pode ver um conjunto de inventário online e outro diferente na loja, recomendações que se contrapõem ou uma barra de pesquisa que retorna resultados diferentes no aplicativo e no desktop.

O comércio unificado resolve os desafios visíveis na superfície integrando os dados que estão por baixo. A plataforma de dados MongoDB lida com catálogo de produtos, inventário, preços, histórico de pedidos e personalização. Já a camada de pesquisa inteligente combina precisão lexical, significado semântico e sinais geoespaciais: a mesma consulta que classifica os produtos por relevância também pode apresentar a loja mais próxima com estoque.



**CONHEÇA
A SOLUÇÃO**

O que é possível uma vez que os dados são unificados é mais interessante do que o que é resolvido. A pesquisa híbrida retorna resultados que respeitam a precisão das palavras-chave e o significado semântico. A personalização se baseia no histórico multicanal sem nenhum processo ETL realizado durante a noite. A mesma camada

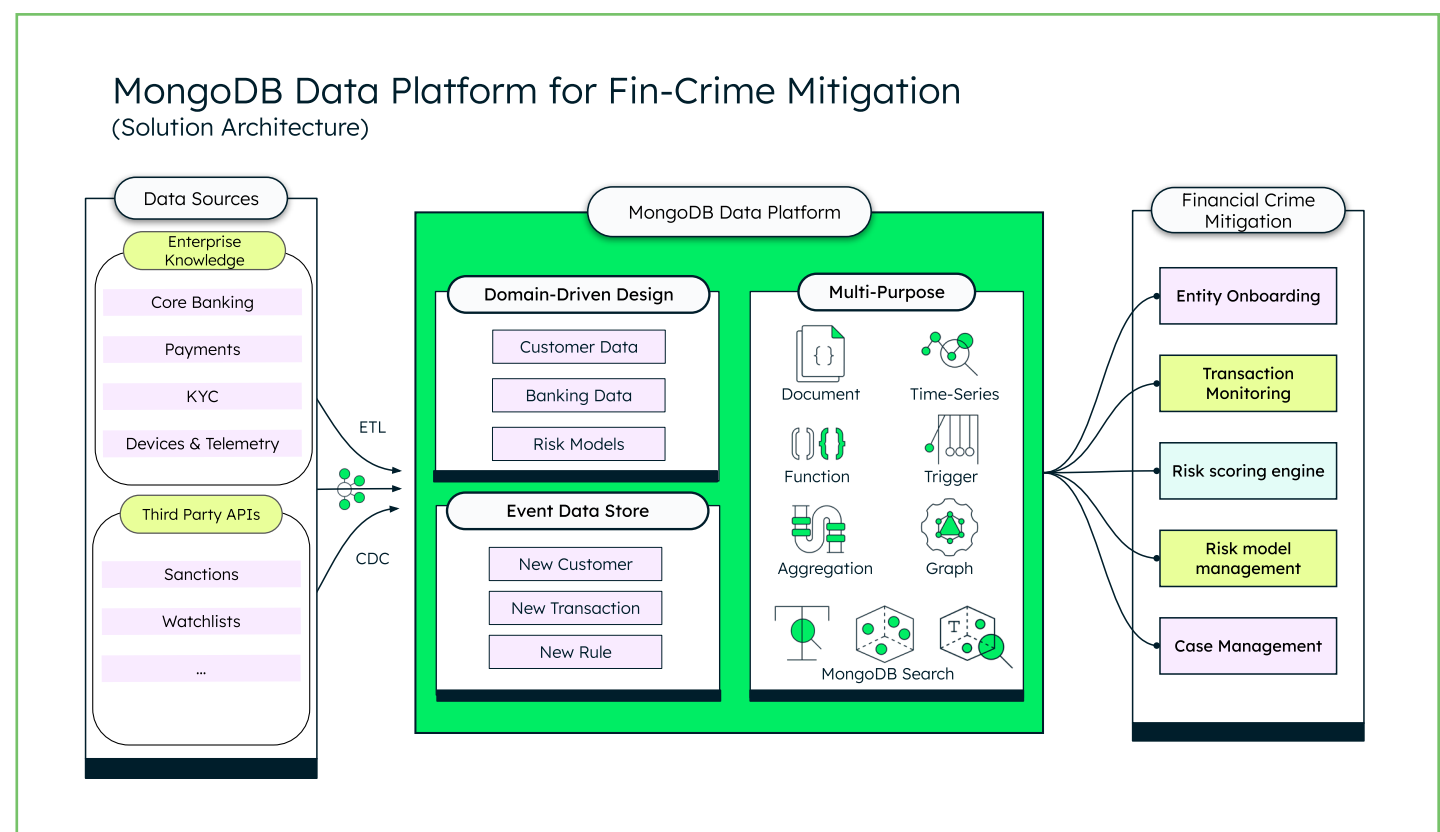
de dados que alimenta a vitrine da loja também alimenta um agente de IA que lida com consultas de suporte, pois só há uma fonte da verdade para ancorar ambos. O padrão que traz retorno para a vitrine da loja também traz retorno na velocidade da máquina, para cargas de trabalho como a detecção de crimes financeiros.

Criando uma plataforma de mitigação de crimes financeiros com o MongoDB

A detecção de crimes financeiros sempre esteve na interseção de dois problemas nos dados. O primeiro é estruturado: transações, hierarquia das contas, campos “Conheça seu cliente (KYC)” e listas de sanções. O segundo é não estruturado: anotações, transcrições e documentos de casos, além da ampla variedade de detalhes contextual que explica por que uma transação é suspeita. Durante décadas, os bancos tentaram resolver as duas metades em sistemas separados e conciliá-las no nível de alerta. Os investigadores pagaram o custo, mais visivelmente nos 90% dos alertas do sistema legado, que acabam sendo falsos positivos, sobrecarregando as equipes de conformidade enquanto ameaças reais passam despercebidas.

As apostas superaram a arquitetura. As instituições financeiras globais perdem mais de US\$ 2 trilhões com crimes financeiros a cada ano, um número projetado para quase triplicar, para US\$ 6 trilhões até 2030, e adversários auxiliados por IA já operam na velocidade da máquina. A solução é uma única plataforma que reúne, lado a lado, pesquisa, similaridade semântica, análise de grafos, regras e aprendizado de máquina. Controles apropriados de acesso são o padrão confiável para qualquer plataforma que lide com esses dados tão confidenciais. A pesquisa híbrida permite que os investigadores perguntem “*Quem mais movimentou dinheiro neste aparelho nos últimos 90 dias?*” e “*quem mais usou linguagem semelhante a este caso nas transcrições de suporte?*” da mesma interface de consulta, contra a mesma fonte da verdade.

Figura 8.
Arquitetura de alto nível de uma plataforma unificada de combate ao crime financeiro com MongoDB.



LEIA O BLOG

A fragmentação não deixa uma plataforma de crime financeiro mais segura. Deixa a detecção mais lenta, a auditoria mais difícil e a operação mais cara, um custo que a próxima etapa acompanha nos diagnósticos automotivos e na linha de montagem.

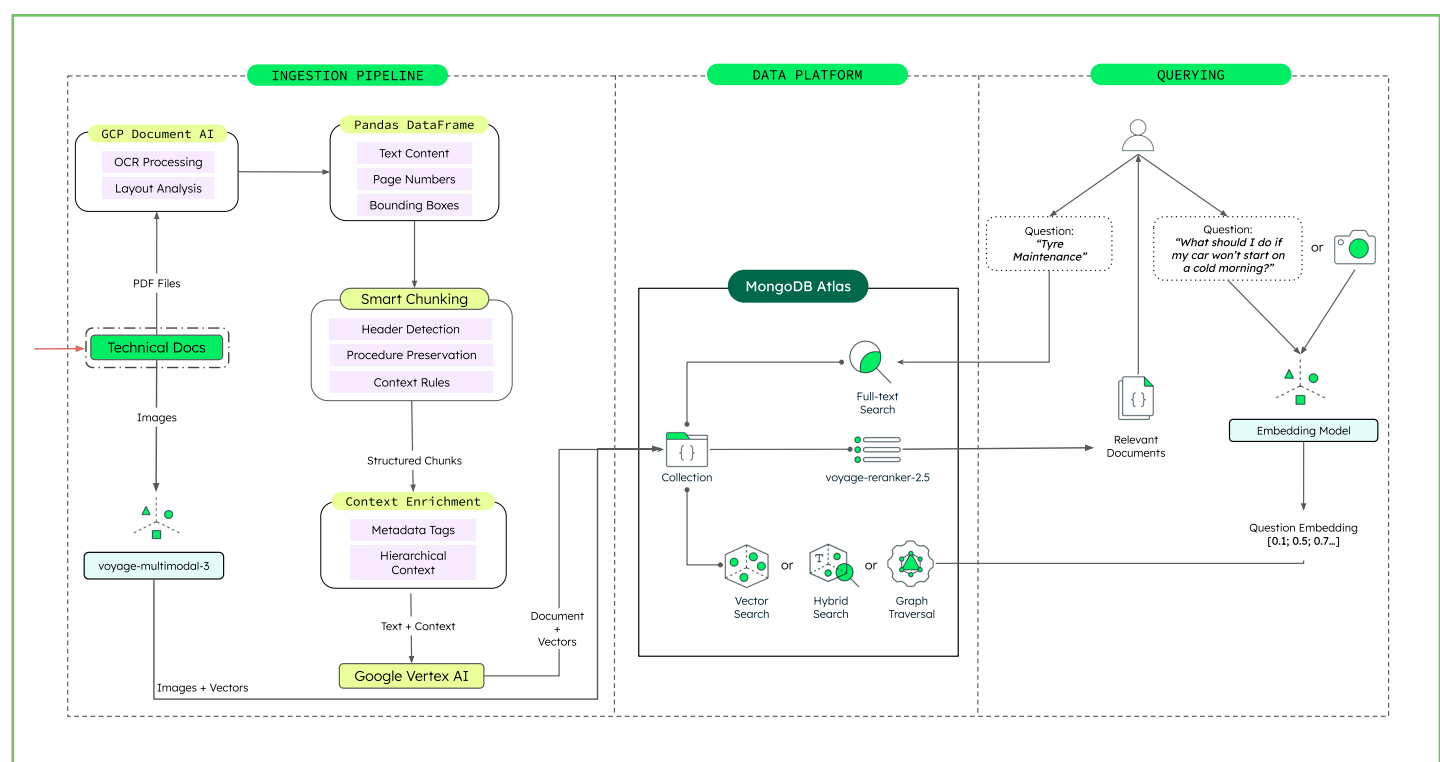
Dentro de um aplicativo inteligente de diagnóstico automotivo pós-venda

A conversa de diagnóstico que costumava ocorrer entre um consultor de serviço, um técnico e um encadernador de manuais de reparo está acontecendo cada vez mais entre um tablet na área de serviço e um agente de IA que leu todos os manuais e boletins emitidos para essa plataforma. A arquitetura que faz isso funcionar é mais interessante do que o frontend sugere. O que está em jogo é ainda maior. Os técnicos podem gastar até 30% do tempo pesquisando informações em vez de consertar veículos, e os eventos “No Fault Found” (NFF), em que uma parte funcional é trocada porque o diagnóstico foi inconclusivo, representam quase 30% dos custos de garantia.

O padrão começa nos manuais. A documentação dos serviços e os boletins técnicos do fabricante de equipamentos originais (OEM, a própria montadora) são processados por meio de reconhecimento óptico de caracteres (OCR), divididos em partes (chunking) para preservar a integridade do

procedimento e incorporados com modelos de texto e multimodais, para que o técnico possa descrever um sintoma em palavras ou tirar uma foto de um conector corroído. A camada de dados é unificada dentro do MongoDB Atlas: embeddings de texto, embeddings de imagem multimodal e o grafo de relacionamento entre sintomas, sistemas e causas-raiz ficam todos na mesma coleção. O agente usa pesquisa vetorial para encontrar a seção relevante do manual, percorre a árvore de falhas com \$graphLookup (operador do MongoDB para consultar recursivamente dados conectados) para sugerir o próximo passo lógico e aplica um reranker da Voyage AI que coloca avisos de segurança e correções verificadas no topo. Os mecânicos pensam em sistemas, não em palavras-chave: uma bateria descarregada pode ser causada pela trava do porta-malas com consumo parasítico, e um backend unificado de grafo e vetor é o que permite ao agente sugerir checar a trava quando o técnico procura a bateria.

Figura 9. Arquitetura vetorial unificada, gráfica e multimodal para diagnóstico automotivo no MongoDB Atlas. O pipeline de ingestão processa manuais via OCR e gera embeddings; as consultas podem ser texto ou imagens.



As equipes automotivas são um dos poucos lugares onde uma camada unificada de inteligência retorna dentro de um único ciclo de produto: diagnóstico mais rápido, menos escalonamentos, reparos mais rápidos resolvidos na primeira visita e reduções significativas no custo da garantia NFF. A mudança arquitetônica é o pré-requisito, não a consequência.



**CRIE O
APLICATIVO**

Da camada de inteligência unificada à precisão em escala

A seção 02 era a arquitetura. O Converged Datastore é real, quatro setores provaram que ele absorve diferentes cargas de trabalho sob o mesmo teto, e o fio condutor que liga manufatura, varejo, serviços financeiros e automotivo é o mesmo argumento arquitetônico em diferentes vocabulários. Porém, uma arquitetura sem confiança é apenas um diagrama.

A seção 03 trata da disciplina que conquista a confiança. Uma camada unificada pode armazenar dados operacionais, vetores, pesquisa vetorial e memória do agente em um único lugar. Mas, se os resultados que ela retorna a um agente estão desatualizadas, incompletas ou imprecisas, o agente alucina, e a elegância da arquitetura vira um problema para o cliente. Quando você aceita que as alucinações são um problema na recuperação e não um problema no modelo, estas perguntas são práticas:

O que é preciso para fundamentar, em tempo real, cada resposta dos agentes na verdade operacional?

Como manter embeddings, recuperações e ranqueamentos rápidos e baratos sem perder a precisão?

O que a camada de dados precisa entregar antes que um agente de produção esteja liberado para implantação?

Nas próximas oito páginas, respondemos às três perguntas: o Voyage 4 elevando o limite máximo de precisão, as decisões de engenharia que fazem da recuperação um problema de otimizar três fatores ao mesmo tempo, em vez de apenas dois, e a central de atendimento da LG U+, que já executa tudo isso em produção com 3,5 milhões de chamadas por mês.





SEÇÃO TRÊS

Precisão em escala

Fundamentando agentes de IA na verdade operacional em tempo real.

A Seção 02 apresentou os argumentos para uma camada de dados unificada. Esta seção aborda o que essa camada precisa oferecer antes que um agente criado sobre ela possa ser considerado confiável em produção.

Alucinações não são apenas um problema de modelo. Elas são um problema de recuperação. Quando um modelo de linguagem inventa um número, uma entidade ou uma política que não existe, quase sempre é porque a recuperação que alimentou o prompt estava imprecisa, desatualizada ou incompleta. Existe um nome para a distância entre um sistema que retorna uma resposta e um sistema cujas respostas podem ser transformadas em ação: a **lacuna de confiança (Trust Gap)**. Essa lacuna não é abstrata. Um agente raciocinando sobre uma captura dos dados do cliente do dia anterior, um vetor que está atrasado em relação ao registro que descreve ou uma recuperação cuja fundamentação se desviou

silenciosamente porque o modelo de embedding foi trocado e os dados não foram reprocessados: nenhum desses casos produz uma resposta visivelmente errada. Eles produzem uma resposta confiante. Uma alucinação em uma demo isolada é uma curiosidade. A mesma alucinação aplicada a milhares de interações com clientes por minuto vira um incidente.

A MongoDB passou os dois últimos eventos MongoDB.local lançando os componentes necessários para fechar essa lacuna. No MongoDB.local de São Francisco, em janeiro de 2026, a mensagem era entregar mais rápido: a série [Voyage 4](#) elevou o limite máximo de precisão da recuperação, a [Embedding and Reranking API](#) trouxe modelos de recuperação de alta fidelidade diretamente para a camada de dados, e os [embeddings automatizados da Voyage AI](#) passaram a ser gerados no momento em que os dados são gravados para que os agentes sempre tenham contexto atualizado. No MongoDB.local de Londres, em maio de 2026, a mensagem passou a ser executar agentes em produção em escala empresarial: o MongoDB 8.3 entrou em disponibilidade geral (GA), com até 45% mais leituras e 35% mais gravações em relação ao 8.0, e o [LangGraph.js Long-Term Memory Store](#) também entrou em disponibilidade geral, dando às equipes de JavaScript e TypeScript a mesma memória persistente que os desenvolvedores Python já tinham

CONTINUA NA PRÓXIMA PÁGINA





Outubro de 2024

Disponível hoje

8.0 → 8.3

35%

Mais taxa de
transferência
de gravação

45%

Mais taxa de
transferência
de leitura

15%

Mais taxa de
transferência
de transações

Dois eventos, um argumento. A precisão é uma propriedade de arquitetura. Chame isso de Integridade Contextual: a disciplina de manter dados operacionais, embeddings, índices de pesquisa e estado do agente automaticamente consistentes, em vez de reconciliá-los depois. Quando esses quatro elementos ficam no mesmo registro, servido pelo mesmo caminho de

query, não há uma etapa de sincronização pela qual a precisão possa escapar. A recuperação recebida por um agente é fundamentada na verdade operacional do mesmo instante em que foi emitida. Os trade-offs que permanecem (quantização, dimensionalidade e escolha de embedding) são ajustados sobre uma base cuja precisão não está se degradando por baixo deles.



**RESUMO DO .LOCAL
EM SÃO FRANCISCO**



**RESUMO DO .LOCAL
EM LONDRES**

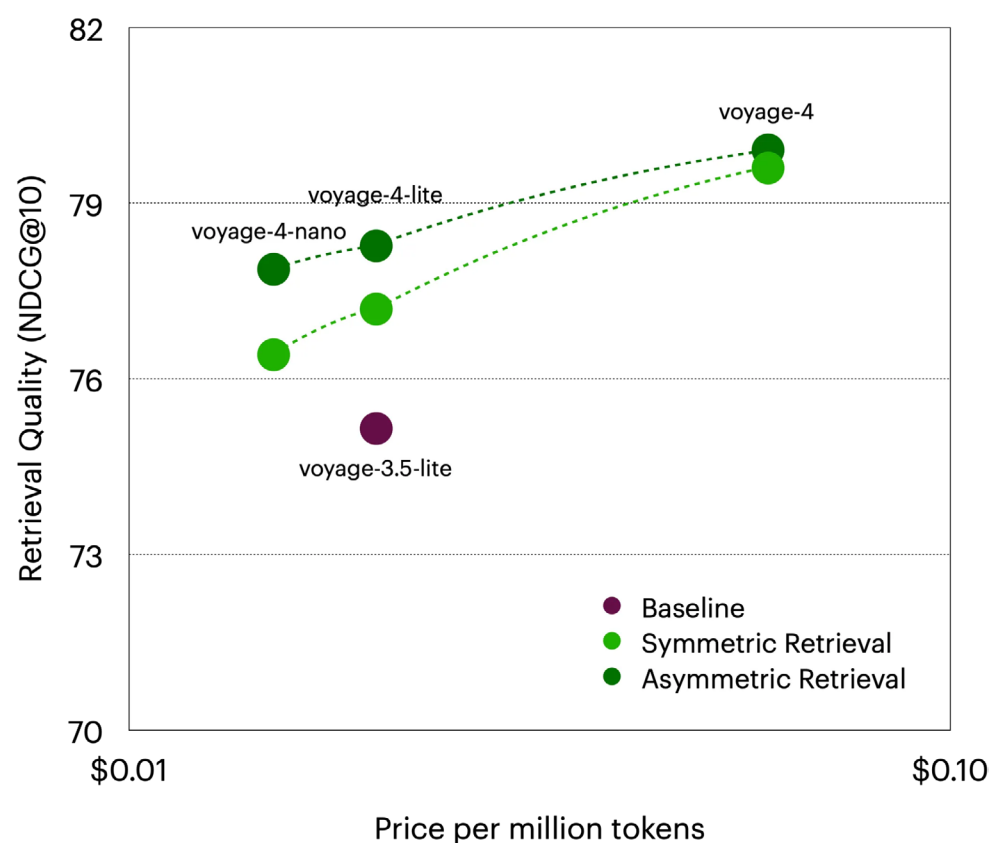
A série Voyage 4 já está disponível

O Voyage 4 é uma família, não um modelo. A série vem em quatro tamanhos: voyage-4-large, voyage-4, voyage-4-lite e o voyage-4-nano, de pesos abertos (open-weights). O destaque é a precisão da recuperação. Os modelos de embedding da Voyage AI lideram o Retrieval Embedding Benchmark (RTEB), superando modelos comparáveis da Gemini, Cohere e OpenAI nos tipos de tarefa que aparecem em RAG corporativo. O ranking do RTEB continua sendo o mais próximo que o setor tem de uma medida objetiva de quão bem um agente encontra o que precisa.

A inovação de arquitetura é o espaço compartilhado de embeddings, o primeiro do setor. Gerações anteriores exigiam modelos idênticos dos dois lados de uma query, o que fazia com que as equipes ficassem presas a um único perfil de precisão e custo ao criar o índice.

Os modelos Voyage 4 são interoperáveis: todos os quatro geram embeddings compatíveis. O padrão de produção recomendado, a recuperação assimétrica, decorre disso. Incorpore o corpus de documentos uma única vez com voyage-4-large para obter a máxima precisão, com custo único. Incorpore queries continuamente com voyage-4-lite ou voyage-4-nano para manter baixa a latência e o custo por query. A precisão do modelo maior fica no índice. A economia do modelo menor fica no serving. A abertura da seção mencionou “o modelo de embedding foi trocado e os dados não foram reprocessados” como um dos modos de falha que produzem a lacuna de confiança. O espaço compartilhado de embeddings remove a pré-condição para essa falha: nada precisa ser reprocessado, porque o novo modelo fala a mesma linguagem de coordenadas que o anterior.

Figura 10. A recuperação assimétrica eleva a qualidade pelo mesmo custo.



O próprio modelo voyage-4-large usa uma arquitetura de mixture-of-experts (MoE) que entrega precisão de última geração com um custo de serving cerca de 40% menor do que o de modelos densos comparáveis. É o primeiro modelo de embedding de nível de produção a

usar MoE. Para cargas de trabalho agênticas de alto volume, em que um único agente pode emitir dezenas de recuperações por tarefa, essa relação de custo se multiplica rapidamente. Mas o modelo é apenas o limite máximo. Atingir esse limite em produção é um problema de engenharia.



**LEIA O
ANÚNCIO**



**GANHE O
SELO DE HABILIDADE**

Como criar uma recuperação de informações rápida, econômica e precisa

O triângulo formado por velocidade, custo e precisão costuma ser tratado como uma escolha em que só é possível maximizar dois desses fatores. Os pipelines de recuperação de informações são um dos poucos casos em que é possível alcançar os três ao mesmo tempo, porque escolhas equivocadas feitas no início cristalizam custos e latências que nenhum upgrade de modelo consegue corrigir depois. Em termos práticos, a recuperação de informações em produção precisa resolver simultaneamente três dimensões: precisão (o contexto correto, fundamentado na verdade operacional existente no momento em que a query é realizada), latência (uma resposta correta que chega depois que o usuário já seguiu em frente é, para fins operacionais, uma resposta errada) e custo (um pipeline executado a cada consulta e repetido diversas vezes durante uma interação agêntica não consegue dimensionar economicamente se cada chamada for cara).

O Voyage 4 eleva o limite da precisão. O trabalho de engenharia necessário para alcançar esse nível em produção abrange três otimizações, cada uma voltada para um gargalo diferente. **A redução de dimensionalidade** usa o Matryoshka Representation Learning para truncar embeddings preservando o significado: 50% menos armazenamento e memória, queries 40% mais rápidas e retenção de 98% da precisão de referência. **A quantização binária** comprime vetores de alta precisão para um bit por dimensão: 96% menos memória, queries 60% mais rápidas e retenção de 99% da precisão. **A recuperação assimétrica** combina um modelo maior para documentos com um modelo menor para queries: custo de embedding por query 50% menor com o voyage-4 e até 83% menor com o voyage-4-lite. Números concretos, não impressões.

Figura 11.
Técnicas de otimização de recuperação avaliadas no conjunto de dados NFCorpus. As três abordagens preservam de 98% a 99% da precisão de referência, ao mesmo tempo que proporcionam melhorias de latência de 40% a 60%.

Otimização	NDCG@10 médio	Latência P50 (ms)	Latência P95 (ms)
Nenhuma (linha de base)	0,65	1,772	2,11
Redução de dimensionalidade	0,643	1,049	1,265
Quantização binária	0,646	0,718	0,841
Recuperação assimétrica	0,64	1,722	2,006

A história do dimensionamento de custos é o ponto que vale a pena levar para a revisão de arquitetura. Com 10.000 queries por dia, a recuperação assimétrica com voyage-4-lite gera uma economia de aproximadamente US\$ 11.000 por ano em comparação com a linha de base. Com 1 milhão de queries por dia, a mesma configuração gera uma economia superior a US\$ 1 milhão por ano. E quanto maior esse número se torna, mais importante passa a ser a camada de dados, porque cada otimização precisa se integrar de forma consistente sem obrigar a equipe a reestruturar toda a plataforma por baixo dela.

A principal conclusão é que a precisão em escala resulta de um conjunto de decisões, não de uma única otimização. Escolher o modelo de embeddings correto define o limite máximo de precisão. Escolher a dimensionalidade correta define os limites de armazenamento e memória. Escolher a estratégia de quantização correta define os limites de latência. Escolher a estratégia correta de embeddings no lado da query define o custo por query. Escolher a camada de dados correta faz com que todas essas decisões trabalhem juntas, em vez de competir entre si.



LEIA O BLOG

Gerador de relatórios médicos com tecnologia de IA

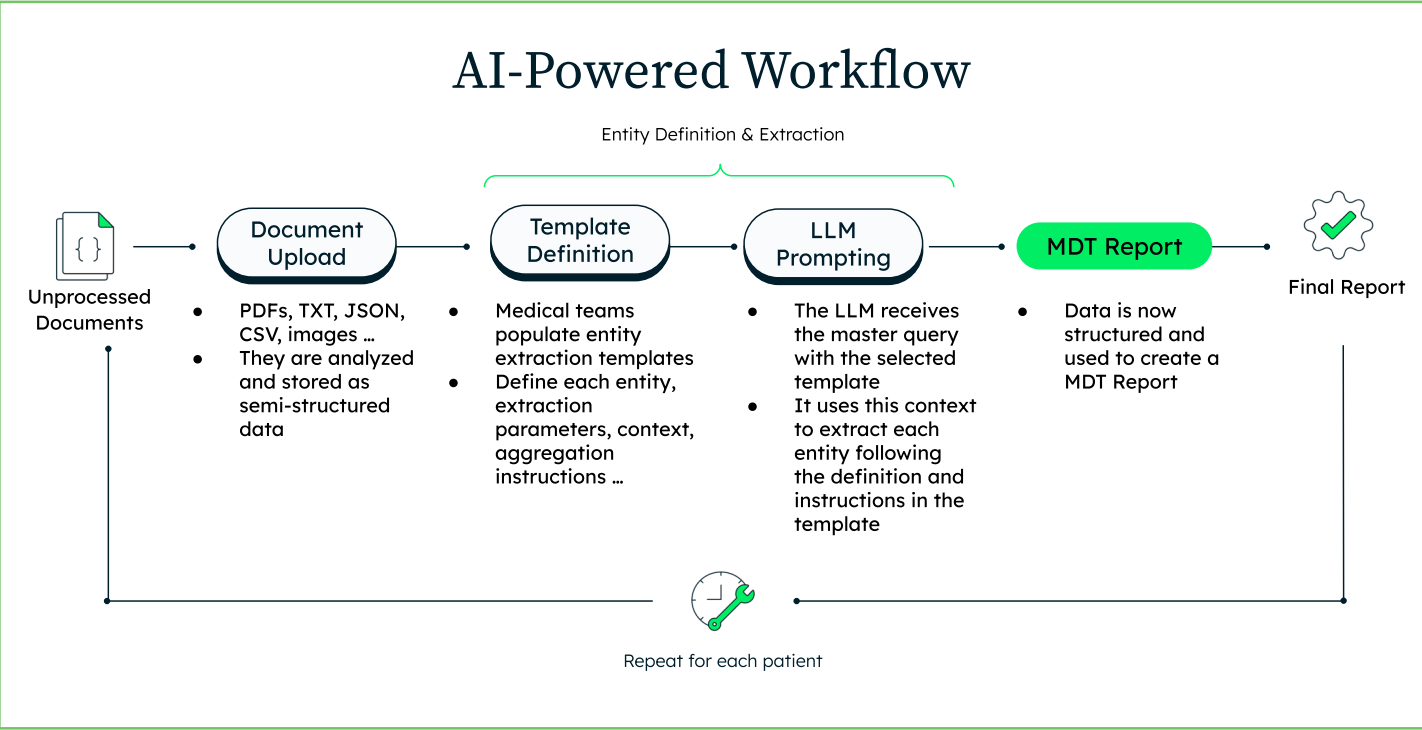
A documentação clínica é a parte da medicina que consome mais tempo e gera mais esgotamento profissional. Em unidades especializadas, como oncologia, os pacientes chegam com um longo histórico de relatórios digitalizados, PDFs, laudos patológicos, resumos de exames de imagem e resultados laboratoriais espalhados por sistemas desconectados. Antes de cada reunião da equipe multidisciplinar, coordenadores e residentes passam horas reconstruindo a história clínica: extraíndo fatos, resolvendo contradições, normalizando terminologias e montando uma linha do tempo coerente. O problema de dados por trás da crise da documentação é o que torna difícil cumprir a promessa da IA.

A arquitetura de referência para um gerador de relatórios médicos com tecnologia de IA combina dados estruturados de prontuários eletrônicos de saúde (EHR), anotações clínicas não estruturadas, metadados de imagens e a literatura médica relevante em um único pipeline de recuperação.

O MongoDB atua como a camada de dados unificada, armazenando registros de pacientes, embeddings de conteúdo clínico e a trilha de auditoria necessária para a governança clínica.

A abordagem adotada parte de uma visão clara sobre precisão. Cada etapa de geração é fundamentada em fontes recuperadas. Cada fonte recuperada pode ser rastreada até sua origem. Cada edição feita por um profissional de saúde retorna ao sistema como um sinal. O objetivo não é tirar o profissional de saúde do processo. O objetivo é fazer com que o trabalho desse profissional seja o trabalho que exige julgamento criterioso, não a transcrição.

Figura 12.
Diagrama do
fluxo de trabalho
de IA



**EXPLORE A
SOLUÇÃO**

A jornada do construtor: criando aplicativos de IA com o MongoDB

Para engenheiros cansados de conteúdo conceitual sobre IA e que querem ver algo sendo criado de ponta a ponta, a palestra Builder's Journey, apresentada no MongoDB.local São Francisco, é uma das mais interessantes. Ela mostra, ao vivo, a criação de uma aplicação de produtividade para desenvolvedores em duas evoluções: a V1 entrega um assistente de IA para brainstorming de projetos com pesquisa semântica sobre trabalhos anteriores; a V2 adiciona ingestão de imagens, pesquisa multimodal em textos e imagens e “memórias”: preferências e detalhes de implementação extraídos que personalizam proativamente o próximo projeto. A mudança arquitetônica entre a V1 e a V2 consiste em alterar um único campo de 1 para 2. As mesmas três coleções. As mesmas queries. O mesmo cluster. O que torna essa palestra útil para líderes técnicos, mesmo para aqueles que não programam no dia a dia, é o argumento arquitetônico implícito e outro explícito. No lado implícito, cada etapa reutiliza a mesma camada de dados. O mesmo cluster Atlas alimenta dados operacionais, pesquisa vetorial,

memória de agentes e o pipeline de recuperação. Não existe um momento de “e agora adicionamos um banco de dados vetoriais”. No lado explícito, a palestra faz uma pausa para abordar a pesquisa vetorial filtrada e explica por que o MongoDB primeiro filtra por ID do usuário e depois executa a pesquisa vetorial, enquanto o Postgres executa a pesquisa vetorial primeiro e aplica os filtros depois. Isso significa que resultados relevantes podem ser excluídos do conjunto de candidatos antes mesmo de serem avaliados. Uma pequena decisão de arquitetura. Uma consequência real para a precisão em escala.

Também é uma ferramenta útil de onboarding. Mostrar a palestra a um novo integrante da equipe faz com que ele se torne produtivo mais rápido do que com qualquer diagrama de arquitetura, porque ele vê as decisões sendo tomadas no contexto e os trade-offs sendo evitados. A palestra ensina o padrão. A próxima história mostra esse padrão lidando com uma carga real de produção.



**ASSISTA
À PALESTRA**



LG U+ dimensiona o atendimento ao cliente com IA usando o MongoDB Atlas

Argumentos sobre arquitetura encontram um limite diferente quando a carga de trabalho é uma central de atendimento que recebe 3,5 milhões de chamadas por mês.

A LG U+, operadora de telecomunicações sul-coreana que administra um dos maiores centros de atendimento ao cliente do país, decidiu criar o Agent Assist: uma ferramenta de IA que fornece a 4.000 agentes humanos transcrição em tempo real, categorização automática das solicitações dos clientes e resumos contextuais inteligentes durante chamadas ao vivo. O banco de dados relacional utilizado não conseguia acomodar os dados vetoriais gerados pelos recursos de IA, os testes que a equipe precisava executar nesses recursos durante os atendimentos e as queries operacionais exigidas pelo serviço, tudo no mesmo caminho.

A LG U+ criou o Agent Assist do zero no MongoDB Atlas e adotou o Atlas Vector Search logo nas fases iniciais do desenvolvimento. Ao migrar os

dados operacionais do Postgres para o mesmo armazenamento que continha os embeddings, a equipe eliminou completamente a camada de sincronização entre os sistemas vetoriais e operacionais. O modelo documental acomodou os diferentes formatos dos dados de atendimento, e o \$vectorSearch executou similaridade semântica diretamente nas mesmas coleções que os agentes já consultavam para obter respostas transacionais.

O serviço foi lançado quatro meses após o início do projeto. O tempo de processamento por chamada caiu 7%. A eficiência de recursos aumentou 30%. Mesmo sob carga, a latência permaneceu abaixo de um segundo, e o dimensionamento automático absorveu os picos matinais, quando o volume de chamadas aumenta. Inicialmente implantado para 1.200 agentes, o serviço agora lida com mais de um milhão de consultas de usuários por semana e está sendo ampliado para atender 2.300 agentes.

“Gerenciar dados vetoriais e operacionais no MongoDB trouxe novas possibilidades para nossa equipe.”

Minkyu Ha, Gerente Sênior de Engenharia de Software, LG U+

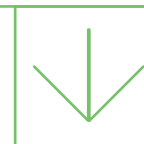
A decisão que encerra essa história tem mais peso do que os percentuais apresentados. Além do Agent Assist, a LG U+ está expandindo a mesma arquitetura para seus callbots e chatbots, ampliando o serviço para o mercado B2B e trabalhando com a MongoDB em uma modernização mais ampla da camada de dados, que inclui a migração de aplicações locais de grande porte para o Atlas. A escolha arquitetônica feita para uma única aplicação tornou-se a base de todo o portfólio de IA da empresa.

É assim que a Integridade Contextual se manifesta na escala de uma operadora nacional de telecomunicações. Embeddings vetoriais e dados operacionais ficam no mesmo registro. As informações recuperadas pela IA e apresentadas a um agente humano durante uma chamada de atendimento estão fundamentadas na verdade operacional daquele exato momento. A lacuna de confiança apresentada no início desta seção não é uma questão de gestão; é uma falha que a própria arquitetura elimina.



**LEIA A
HISTÓRIA**

CONTINUA NA PRÓXIMA PÁGINA





O caso da LG U+ mostra exatamente o que a camada de dados precisa oferecer antes que um agente de produção possa ser implantado com segurança.

- O índice de pesquisa da IA e os dados operacionais em tempo real da empresa precisam estar no mesmo lugar, para que o agente não responda com base em uma versão da verdade enquanto atua sobre outra.
- As queries semânticas precisam retornar resultados em menos de um segundo, mesmo sob cargas reais de produção. Qualquer coisa mais lenta do que isso faz com que o agente deixe de ser útil.
- A plataforma precisa escalar automaticamente quando o volume de chamadas aumenta, sem exigir que a equipe reconstrua a solução para absorver os picos de demanda.
- E o modelo de dados precisa ser flexível o suficiente para manter qualquer forma que a conversa assuma, sem uma migração de esquema toda vez que o negócio aprende algo novo. Nada disso é um problema de qualidade do modelo.

Tudo isso é arquitetura, definida antes mesmo de o primeiro prompt ser escrito.

Da precisão em escala para sistemas de ação

A Seção 03 tratou de fundamentação. O Voyage 4 elevou o limite máximo de precisão, e as técnicas de engenharia por trás dele (redução de dimensionalidade, quantização e recuperação assimétrica) tornaram esse nível de precisão viável em produção. A LG U+ mostrou então como a recuperação fundamentada funciona em escala de produção, com 3,5 milhões de chamadas por mês operando sobre uma camada de dados unificada. Um agente fundamentado, que recupera informações a partir da verdade operacional no exato momento em que a pergunta é feita, é algo muito além de um chatbot sofisticado. É um agente pronto para agir.

Quando a precisão em escala é resolvida, a pergunta muda novamente. O que uma empresa realmente pode construir que antes não conseguia? A Seção 04 traz a resposta.



SEÇÃO QUATRO

Construindo a era da IA

Dos sistemas de registro aos sistemas de ação.

Existe uma diferença entre um agente que resume uma solicitação de sinistro e um que realiza o pagamento. Entre um agente que recomenda uma rota de pagamento e um que executa a transferência. Entre um agente que elabora uma decisão de crédito e um que libera os recursos do empréstimo. No momento em que um agente passa a executar transações, os requisitos se multiplicam. A memória precisa ser durável entre sessões. O estado precisa sobreviver a falhas. As permissões precisam ser auditáveis. A precisão era a metade mais fácil do problema. A ação é a parte mais difícil.

O setor já deu um nome a essa mudança: **sistemas de registro versus sistemas de ação**. Um sistema de registro armazena, recupera e apresenta informações. Um sistema de ação altera o estado do mundo em que opera. Ele confirma transações, aciona fluxos de trabalho, agenda compromissos, atualiza registros contábeis e realiza pagamentos. A Seção 03 aperfeiçoou a primeira

metade. Esta seção trata do que muda quando a precisão deixa de ser a única coisa que pode falhar.

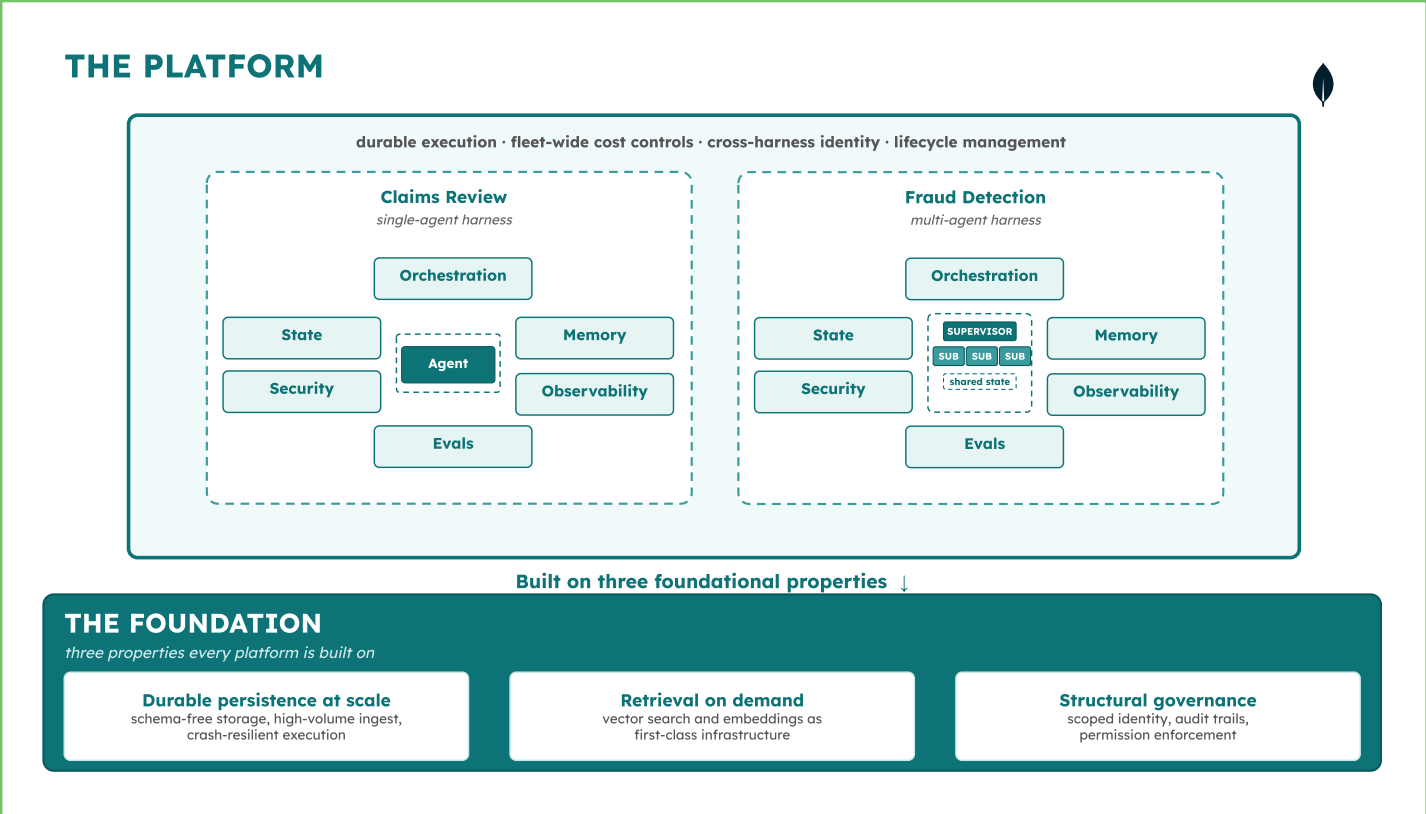
As apostas têm um número. [A Gartner](#) prevê que 40% das aplicações corporativas incorporarão agentes de IA até o final de 2026. E 40% dos projetos de IA com agentes serão cancelados até o final de 2027. A simetria não é acidental. Os cancelamentos não decorrem de um problema de qualidade dos modelos. Projetos corporativos de agentes costumam estagnar em três pontos previsíveis: na revisão de segurança, porque não existe um conjunto padronizado de controles para sistemas autônomos; nos limites de custo, quando os gastos com tokens superam o valor gerado para o negócio; e na fronteira de integração, quando a camada de dados não consegue acompanhar tudo o que o agente precisa saber.

Sistemas de agentes em produção têm mais em comum com sistemas de aprendizado de máquina em produção do que a maioria das equipes percebe. O modelo recebe a atenção. A infraestrutura ao redor do modelo determina se algo realmente funciona: memória, estado, orquestração, observabilidade, avaliações, segurança e a camada de dados que alimenta tudo isso. A série da MongoDB [Designing an Agentic Platform](#) desenvolve essa arquitetura em profundidade. Esta é sua estrutura.

CONTINUA NA PRÓXIMA PÁGINA



Figura 13.
Uma plataforma
permite a execução
de diversos agentes
em um conjunto
variado de casos
de uso, domínios
e frameworks.



Cada caixa na figura é um harness, o runtime que conduz um agente, ou um conjunto coordenado de agentes, por seu ciclo de decisão. As seis disciplinas de engenharia dentro dele, orquestração, estado, memória, segurança, observabilidade e avaliações, são onde a maior parte do trabalho de produção realmente acontece. A plataforma fica acima delas, executando muitos harnesses para muitas equipes ao longo do tempo.

Três propriedades fundamentais sustentam a plataforma. Persistência durável em escala, para os dados sem esquema gerados pelos harnesses; Recuperação sob demanda, por meio de embeddings e pesquisa vetorial, para que o contexto esteja disponível no instante em que um agente precisa dele. E governança estrutural, com identidade delimitada, trilhas de auditoria e aplicação de permissões fornecidas uma única vez na plataforma, em vez de serem acopladas a cada harness. Uma plataforma sem qualquer uma das três produz harnesses que parecem funcionar em condições controladas e falham sob carga de produção.



**LEIA A SÉRIE
SOBRE A
PLATAFORMA**



Os agentes devem ser invisíveis: objetos de negócio importam mais do que memória de IA

Há um erro sutil de arquitetura no centro da maioria das primeiras plataformas de agentes. As equipes tratam a memória do agente como a abstração central, com todo o restante organizado em torno dela: o cliente, a apólice, o sinistro e a transação. Isso funciona em uma demo porque ela só precisa de um agente e de um fluxo de trabalho. Mas falha em produção porque empresas reais têm muitos agentes, muitos fluxos de trabalho e uma única verdade subjacente: o objeto de negócio.

Os agentes devem ser invisíveis. O cliente não interage com um agente. Em seguros, o cliente interage com sua apólice, seu sinistro e sua conta. No varejo, com seu pedido. Na área da saúde, com seu caso. O sistema simplesmente usa agentes para atuar sobre esses objetos. Organize a arquitetura em torno de objetos de negócio, e não da memória dos agentes, e três coisas acontecem. Novos agentes se conectam a objetos existentes sem exigir uma nova arquitetura da camada de dados. Vários agentes podem se coordenar em torno do mesmo objeto sem interferir uns nos outros. A trilha de auditoria

se agrega naturalmente no nível do objeto, onde reguladores e operadores esperam encontrá-la.

A camada de dados necessária para isso tem um nome: a **camada de dados de inteligência**, onde objetos de negócio, embeddings, memórias de agentes, estado do fluxo de trabalho e trilha de auditoria ficam em um único lugar coerente. O modelo de documento da MongoDB é uma escolha natural. Objetos de negócio são, por natureza, aninhados. Controles de acesso podem ser aplicados no nível do campo. E a mesma interface de query responde a perguntas operacionais, semânticas e históricas. Esse é o padrão Converged Datastore aplicado ao caso em que ele gera retorno mais rápido.

A conclusão prática é projetar primeiro o modelo de objetos. Agentes entram e saem. Os objetos, não. Acerte o modelo de objetos, e cada agente subsequente se beneficia da mesma base. Erre, e cada agente se torna sua própria rearquitetura.

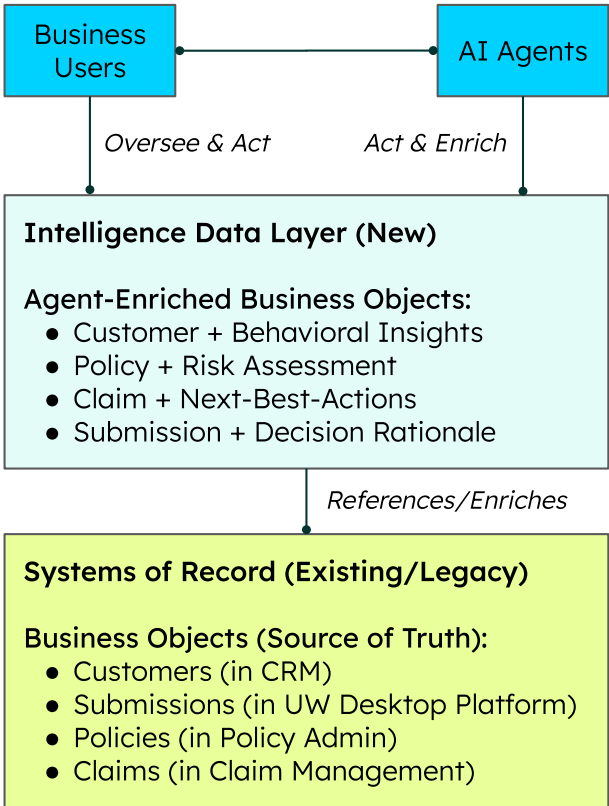


Figura 14.
O sistema de ação: viabilizando supervisão e ação mais inteligentes para usuários de negócios



[LEIA A PARTE 1](#)



[LEIA A PARTE 2](#)

Crie agentes de IA que valham a pena manter: o Canvas Framework

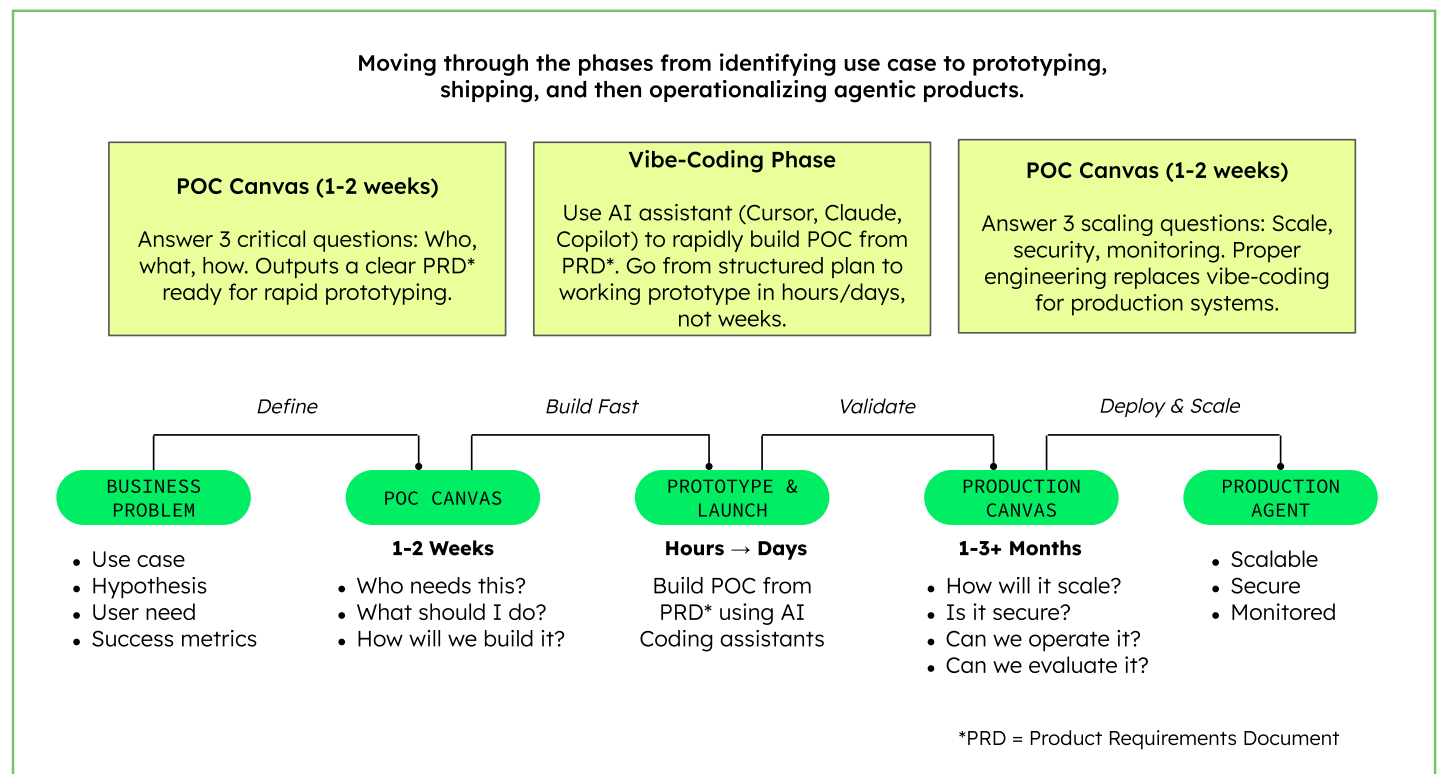
A maioria dos projetos de agentes falha da mesma forma. A equipe começa escolhendo um framework, escolhe um modelo, cria algo interessante e percebe tarde demais que ninguém precisava daquilo que foi construído. [As pesquisas da McKinsey](#) são consistentes nesse ponto. Quase 80% das empresas relatam usar IA generativa, mas menos de 10% dos casos de uso passam da fase piloto. O padrão é tão comum que pesquisadores do MIT deram um nome a ele: o abismo da IA generativa.

O Canvas Framework é uma resposta direta a esse padrão. Mikiko Bazeley, especialista em relações com desenvolvedores da MongoDB, modelou-o a partir do modelo de negócios canvas. A estrutura força as equipes a responder às perguntas certas na ordem certa: primeiro produto, depois agente, depois dados e, por fim, modelo. A premissa é que o design de agentes fica entre produto e dados, determinando de qual conhecimento o

agente precisa e quais modelos podem entregá-lo. Inverta a ordem, como faz a maioria das equipes, e o resultado é uma tecnologia em busca de um problema.

O canvas de prova de conceito (POC) é composto por oito blocos focados em validação rápida, organizados em quatro fases: produto, agente, dados e integração de modelos. O canvas de produção adiciona 11 blocos distribuídos em cinco fases: caso de negócio e escala, arquitetura robusta de agentes, infraestrutura de dados unificada, operações de modelos e o conjunto de requisitos que as POCs costumam ignorar, mas que as empresas exigem. Revisão de segurança, limites de custo, orçamentos de latência, observabilidade, governança, pipelines de avaliação e estratégia de ajuste fino. O framework obriga as equipes a planejar tudo isso antes de dimensionar, e não depois.

Figura 15. O Agentic AI Canvas Framework. Uma abordagem estruturada em cinco fases, que avança da definição do problema de negócios para POC, protótipo, canvas de produção e implantação de agentes em produção.



O que o framework acerta, e o que a maior parte das orientações sobre agentes deixa passar, é que a escolha da camada de dados é fundamental. Equipes que tentam adiar essa decisão acabam fazendo uma rearquitetura no meio do caminho. Equipes que escolhem uma plataforma unificada desde o primeiro dia dedicam seu tempo à

engenharia de prompts e à orquestração, em vez de ao trabalho braçal de infraestrutura. O Canvas trata isso como uma decisão estrutural, não como uma preferência de ferramenta.

As próximas partes mostram sistemas de ação em produção, em serviços financeiros, pagamentos, cadeia de suprimentos e crédito.



**LEIA O
BLOG**

Revelando inteligência documental em serviços financeiros com IA agêntica

Serviços financeiros funcionam com base em documentos. Arquivos de empréstimos, pacotes de KYC, cartas de mandato, confirmações de derivativos e atestados de conformidade. O trabalho de extrair estrutura desses documentos historicamente tem sido um trabalho manual de baixo retorno. A IA com agentes muda essa economia, mas somente se a camada de dados conseguir dar suporte à ingestão, extração e reconciliação multiagentes em escala.

A arquitetura de referência para inteligência documental usa o MongoDB como a base de dados unificada para um sistema multiagente supervisor. Um agente supervisor coordena quatro agentes filhos especializados que trabalham de forma colaborativa: um Scanner identifica documentos relevantes, um Avaliador atribui pontuações de relevância para o caso de uso, um Extrator extrai conteúdo e o prepara para divisão em partes menores, e um Processador gera embeddings para tarefas posteriores. Um agente

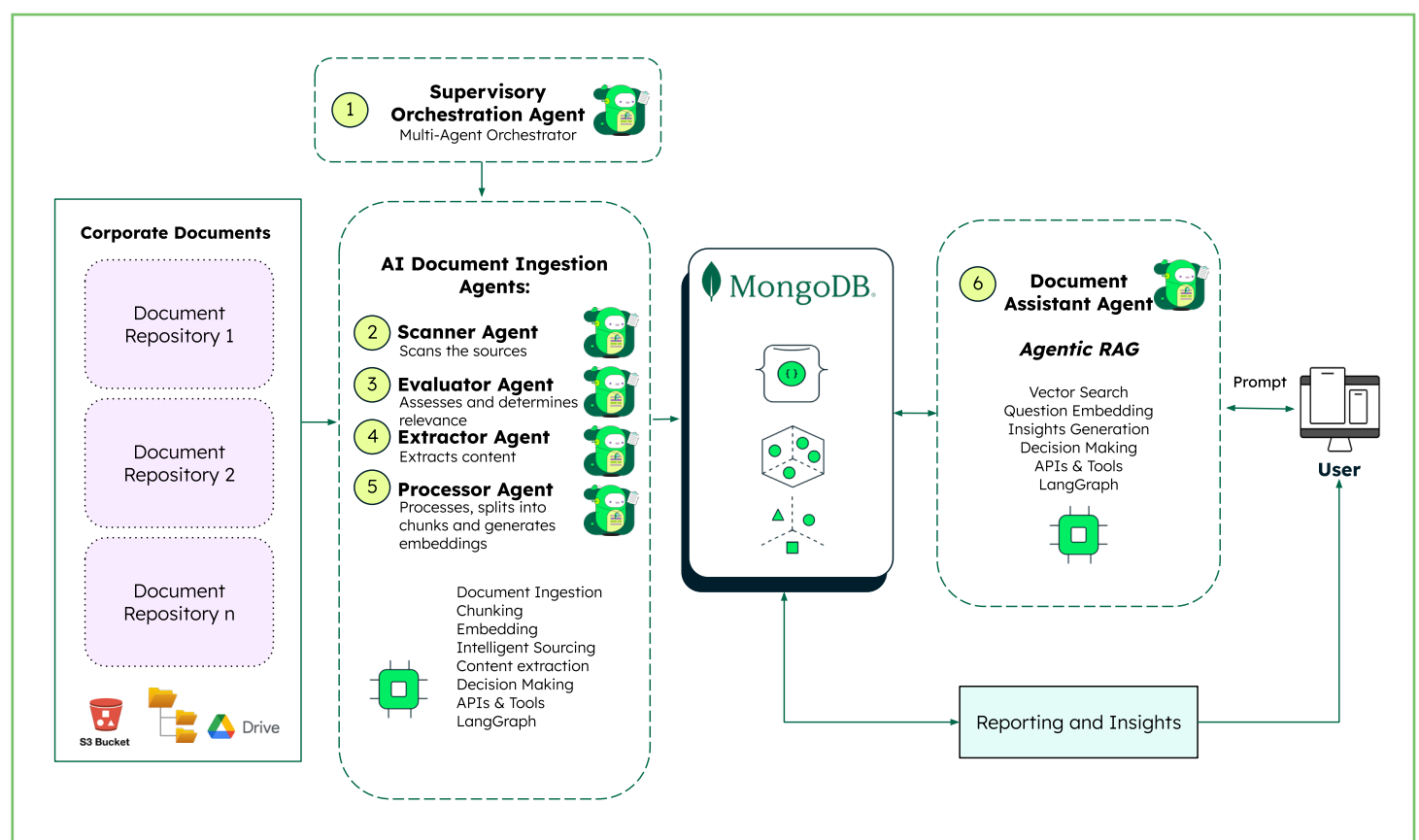
Assistente de Documentos de segundo nível então lida com RAG com agentes, decidindo quando e como recuperar informações com base na query do usuário. O modelo voyage-context-3 da Voyage AI produz embeddings de partes contextualizadas que capturam o contexto local e global dos documentos, proporcionando uma precisão de recuperação significativamente maior a um custo mais baixo.

A memória dos agentes é a parte que, discretamente, faz a maior parte do trabalho. Cada agente se lembra de interações anteriores, aprende com correções e constrói contexto ao longo do tempo. Essa memória precisa existir em algum lugar durável, com controles de acesso e trilhas de auditoria que atendam à governança de serviços financeiros. A função da MongoDB é ser esse lugar: o documento, os embeddings, a memória dos agentes e a trilha de auditoria em uma única camada coerente.

Figura 16.
Arquitetura de
processamento
inteligente de
documentos
com o MongoDB



**LEIA O
ARTIGO**

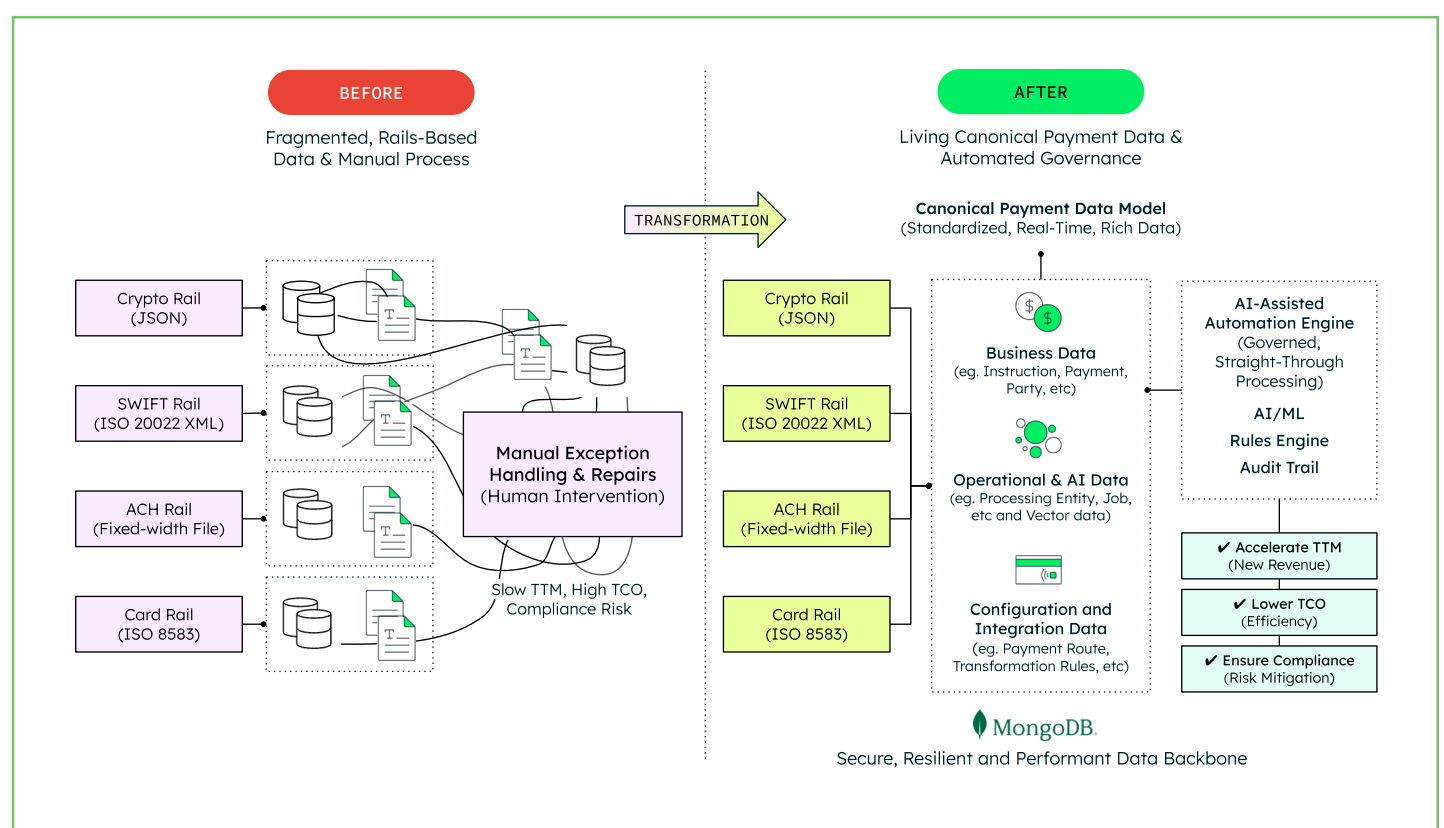


Desbloqueando o poder dos agentes para modernizar sistemas globais de pagamentos

Sistemas de pagamento são o segmento da infraestrutura financeira em que a modernização é mais difícil e o custo de erro é mais alto. Os trilhos que movimentam dinheiro foram criados, cada um, para uma era diferente: o SWIFT usa ISO 20022 XML, o ACH usa arquivos de largura fixa, as redes de cartão usam ISO 8583, e as criptomoedas usam JSON. A maioria dos bancos os conecta com integrações ponto a ponto e remenda as lacunas com tratamento humano de exceções. Só os pagamentos malsucedidos drenam mais de US\$ 100 bilhões da economia global todos os anos. O resultado é menor tempo de entrada no mercado, menor custo de propriedade e compliance tratada como uma propriedade dos dados, e não como uma força-tarefa trimestral.

A solução não é substituir os trilhos. É reunir tudo o que eles emitem em um único modelo canônico de pagamentos: dados de negócios, como instruções e partes; dados operacionais e de IA, como tarefas de processamento e embeddings vetoriais; e dados de configuração, como rotas e regras de transformação. Sobre esse modelo, um mecanismo de automação com IA/aprendizado de máquina, regras e uma trilha de auditoria integrada pode finalmente realizar trabalhos úteis: roteamento, reconciliação, triagem de fraude, resolução de exceções e comunicação com o cliente.

Figura 17.
Da proliferação de trilhos legados para uma plataforma canônica de pagamentos com agentes.



**LEIA
O BLOG**

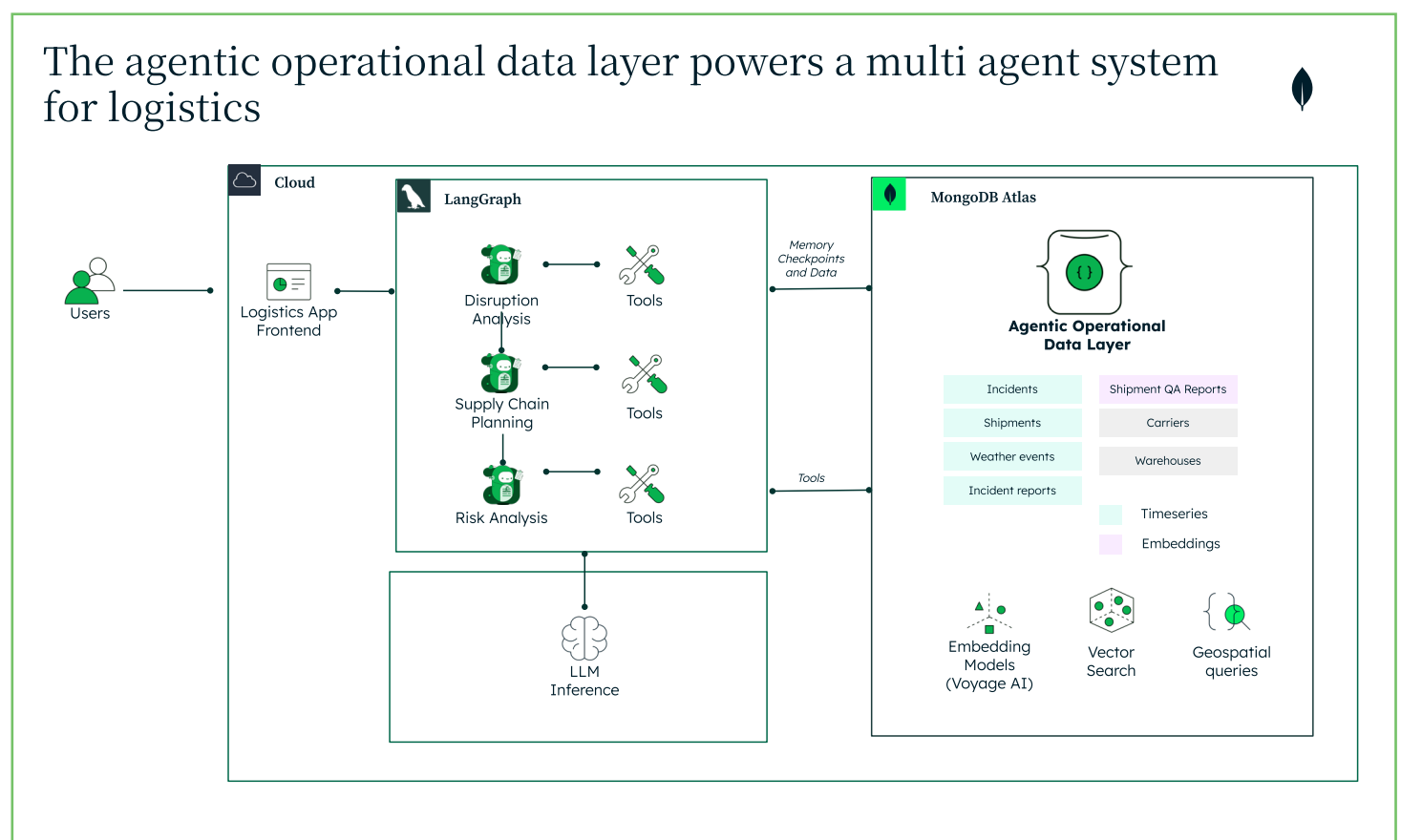
O resultado é maior tempo de entrada no mercado, menor custo de propriedade e compliance tratada como uma propriedade dos dados, e não como uma força-tarefa trimestral. O que muda é o que o banco consegue entregar.

Sistemas multiagente para gerenciamento de interrupções na cadeia de suprimentos

A disrupção da cadeia de suprimentos é o problema multiagente canônico. Um evento climático em um porto, uma ação trabalhista em outro, um problema de qualidade em um fornecedor de segundo nível e um pico de demanda em um mercado importante interagem entre si. Nenhuma equipe humana consegue manter a visão completa em tempo real. Nenhum agente isolado consegue, tampouco. O custo de errar é concreto: estima-se que as disrupções da cadeia de suprimentos custem US\$ 184 bilhões por ano às empresas.

Na prática, o trabalho se divide em funções especializadas. A análise de interrupções monitora incidentes à medida que eles acontecem. O planejamento da cadeia de suprimentos modela o impacto nas etapas seguintes OR modela os efeitos em cascata e propõe mitigações. A análise de riscos pontua a exposição e destaca o que precisa ser escalado. O LangGraph (um framework de orquestração para fluxos de trabalho multiagente) faz o roteamento entre essas funções, mas o entendimento compartilhado dos agentes sobre o mundo fica na camada de dados abaixo: um único armazenamento em que remessas, clima, transportadoras, hubs de frota e relatórios de incidentes incorporados ficam juntos.

Figura 18. Diagrama da arquitetura da solução. Três agentes especializados, *Disruption Analysis*, *Supply Chain Planning* e *Risk Analysis*, são orquestrados por meio do LangGraph, com o MongoDB Atlas atuando como a camada de dados operacionais com agentes para remessas, eventos climáticos, transportadoras, armazéns, embeddings vetoriais e queries geoespaciais.



**EXPLORE A
SOLUÇÃO**

O que o MongoDB Atlas acrescenta é mais do que armazenamento. Eventos climáticos entram em fluxo como séries temporais. Remessas e armazéns carregam coordenadas sobre as quais queries geoespaciais conseguem raciocinar. Relatórios históricos de incidentes são incorporados com a Voyage AI para que a

pesquisa vetorial consiga revelar o que os agentes precisam saber sobre disrupções anteriores. E os checkpoints de memória que o LangGraph grava entre as etapas dos agentes ficam no mesmo armazenamento, o que faz com que a trilha de postmortem se acumule como um registro estruturado, e não como uma conversa no Slack.

A jornada moderna de crédito digital de ponta a ponta, impulsionada pelo MongoDB e pela IA com agentes

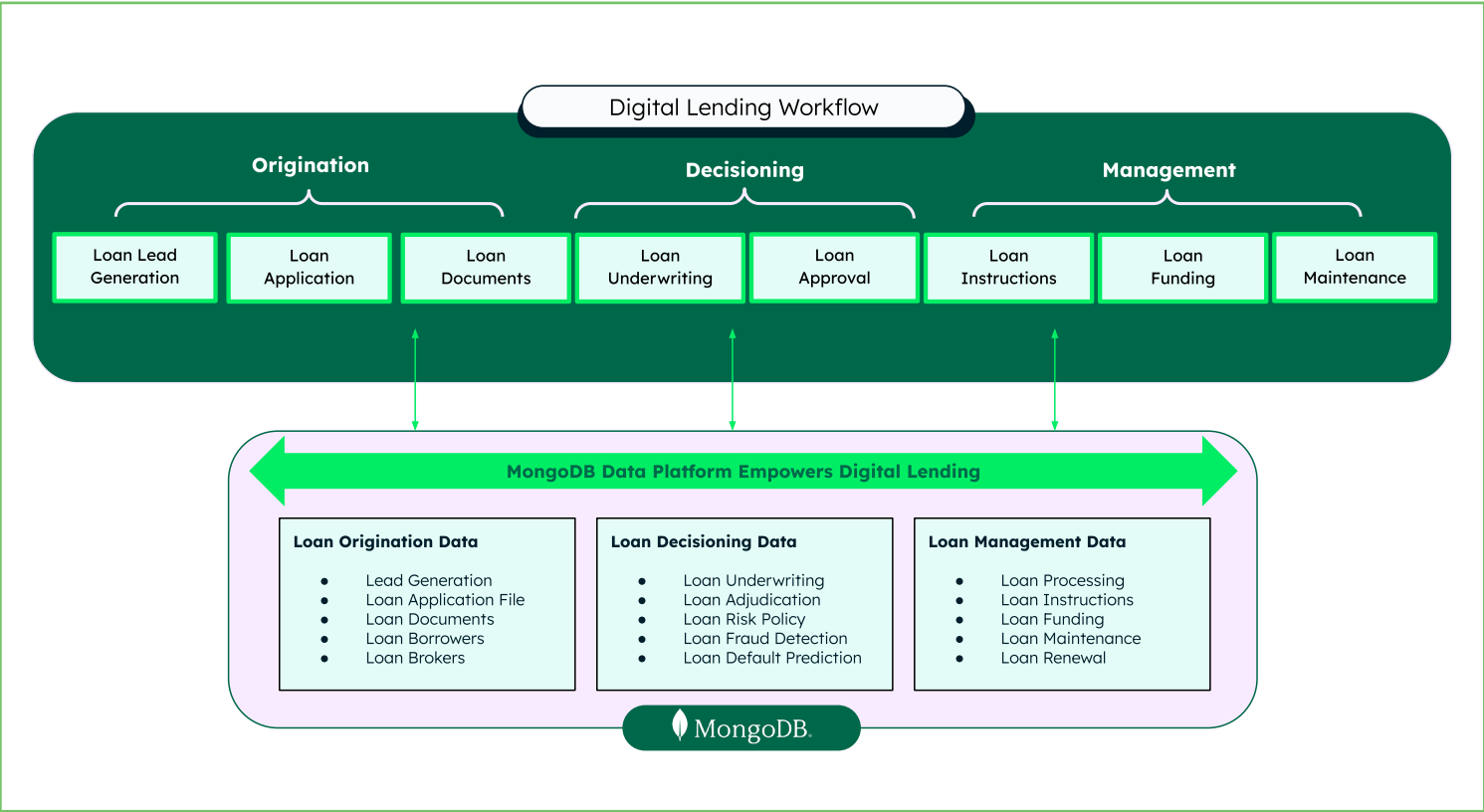
O crédito é o exemplo clássico de fluxo de trabalho financeiro que há duas décadas recebe a promessa de automação de ponta a ponta, mas que, na prática, entregou principalmente automação parcial em muitos silos. Os motivos são conhecidos. Os dados vivem em uma dúzia de sistemas, as etapas de compliance exigem análise humana em várias fases, e a experiência do cliente é moldada pelo sistema que, por acaso, é responsável pela pior transferência.

A IA com agentes muda o gargalo. Uma jornada moderna de crédito pode ser projetada como um conjunto coordenado de agentes que lidam com todo o ciclo de vida, da originação à renovação. Um **agente de solicitação de empréstimo** atua como a porta de entrada inteligente, orientando tomadores, validando e enriquecendo dados em tempo real e extraindo insights de documentos para produzir um perfil de empréstimo pronto para decisão. Um **agente de subscrição** combina regras de negócios, modelos de aprendizado de máquina e dados alternativos para avaliar continuamente a

capacidade de crédito, encaminhando exceções e casos de alto risco a um revisor humano para aprovação final. Um **agente de renovação** monitora o ciclo de vida após a liberação dos recursos, prevendo riscos de refinanciamento, dificuldades financeiras e churn, além de acionar ofertas personalizadas ou intervenções antecipadas. Cada agente é delimitado, cada um é observável, e todos compartilham uma única visão da solicitação como objeto de negócio. O cliente vê uma experiência coerente. O credor vê uma trilha de auditoria.

A arquitetura importa porque a concessão de crédito exige análises complexas e tomadas de decisão criteriosas. A camada de dados determina onde a automação é segura, onde a revisão humana é necessária e, em última instância, onde está a diferença entre uma jornada que leva minutos e outra que leva semanas. Esse padrão se estende além da concessão de crédito para qualquer fluxo de trabalho financeiro baseado em documentos que precise equilibrar velocidade e conformidade regulatória.

Figura 19.
Fluxo de trabalho de IA com agentes: crédito digital



**LEIA O
ARTIGO**

O teste de portabilidade

Os casos de uso desta seção mostraram como são os sistemas de ação no mundo real: pagamentos que reconciliam, cadeias de suprimentos que respondem, jornadas de crédito que avançam da entrada à decisão e agentes que fazem mais do que recuperar informações. Mas, quando esses sistemas começam a importar, surge outra pergunta. Não apenas o que o agente fará, mas onde a empresa precisará que ele seja executado?

Sistemas corporativos com agentes raramente ficam confinados a um único ambiente bem delimitado. Eles podem abranger nuvens, regiões, plataformas de modelos, ferramentas de desenvolvimento, dados operacionais e limites de segurança. Algumas cargas de trabalho se

movem por latência. Outras por residência de dados. Outras porque determinado serviço de IA, modelo ou opção de computação está disponível em outro lugar. Outras porque o próprio negócio muda: um novo mercado, uma nova aquisição, uma nova exigência regulatória e um novo padrão de plataforma.

Isso torna a portabilidade parte da arquitetura, não uma consideração posterior. Se o armazenamento operacional, a pesquisa vetorial, a memória do agente e o modelo de segurança precisarem ser movidos separadamente, a stack se fragmenta novamente. Se esses elementos puderem se mover juntos, o sistema de ação pode acompanhar a empresa.

Sua plataforma de dados acompanha você

A primeira década da computação em nuvem foi sobre escolher uma. A segunda década é sobre operar em várias. Aquisições adicionam uma nuvem que a equipe não escolheu. Reguladores regionais obrigam cargas de trabalho a permanecer no país, no provedor que tiver a zona local. As melhores ferramentas de IA para uma carga de trabalho estão em uma nuvem diferente das melhores ferramentas de IA para a próxima. E a própria capacidade se tornou uma restrição: as equipes recorrem ao seu hyperscaler e recebem a resposta de que não há GPUs disponíveis na região necessária, ou que não há capacidade disponível.

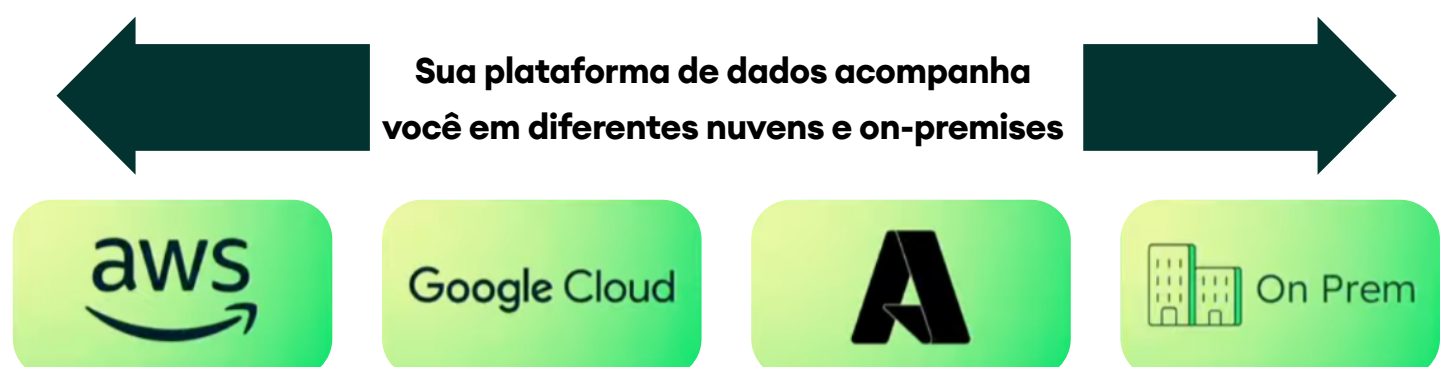
Um padrão já está surgindo em produção. Uma empresa de modelos de fronteira fica sem capacidade de GPU na AWS, transfere parte da carga de trabalho para o Google Cloud para usar TPUs e acaba executando sua camada de dados distribuída entre as duas nuvens. A decisão foi imposta pela física, não por preferência. A arquitetura absorveu essa mudança porque a camada de dados já era independente de nuvem.

Um único MongoDB Atlas cluster pode abranger AWS, Azure e Google Cloud ao mesmo tempo. O mesmo mecanismo, incluindo pesquisa de texto completo e pesquisa vetorial, agora é executado no MongoDB Enterprise Advanced para implantações on-premises e air-gapped (fisicamente isoladas), quando requisitos de residência ou soberania de dados exigem esse modelo.

Isso protege contra a classe mais cara de erro na nuvem: criar uma arquitetura de IA que funciona em uma nuvem e desmorona quando a empresa adiciona, substitui ou se divide entre outra. O ano de 2025 terminou com três interrupções de hyperscalers que merecem ser mencionadas. Google Cloud em junho. AWS US-EAST-1 em 20 de outubro. Azure Front Door nove dias depois. Cada uma tirou do ar clientes que haviam apostado em um único provedor. Cada uma foi contornável para clientes cuja camada de dados abrangia mais de um provedor.



[LEIA O ARTIGO](#)



MongoDB como ledger de mandatos para comércio agêntico: A2A, AP2 e UCP

Este é o item que mais olha para o futuro nessa edição, por isso encerra a seção. O comércio com agentes é o próximo ponto de inflexão para o varejo. O cliente delega cada vez mais a decisão de compra a um agente. O agente compra em nome do cliente, avalia opções, conclui o checkout e acompanha até a conclusão. Três protocolos emergentes estão moldando a forma como essa interação acontece: comunicação Agent-to-Agent (A2A), Agent Payments Protocol (AP2) e Universal Commerce Protocol (UCP).

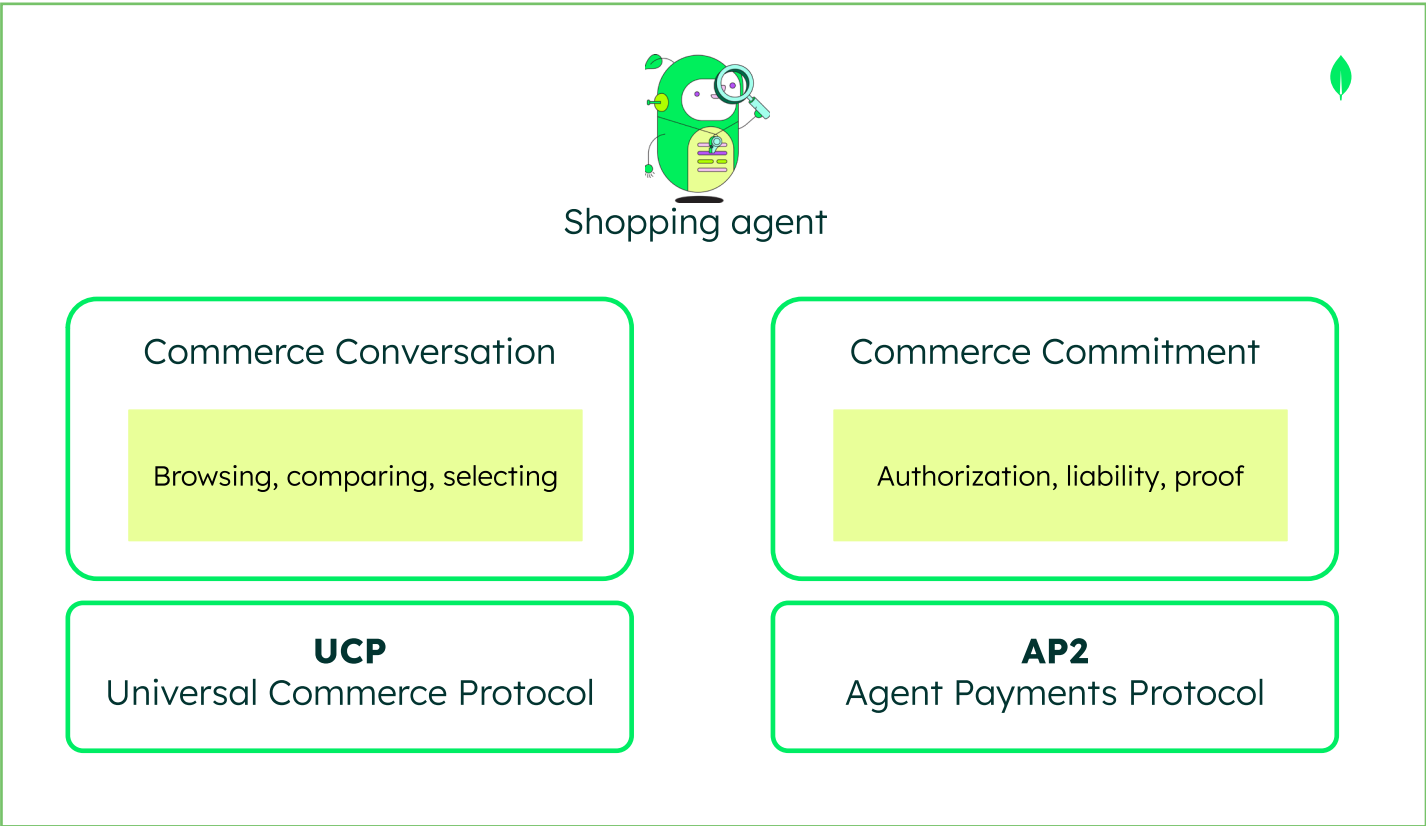
Cada um desses protocolos exige um ledger que capture a autorização do cliente, as ações do agente e as respostas do vendedor com detalhes suficientes para dar suporte a reconciliação, resolução de disputas e auditoria. O MongoDB atua como o **Mandate Ledger (ledger de mandatos)** desse ecossistema emergente. Um mandato é o registro durável e inviolável de uma transação com agentes, e o AP2 define três tipos: mandatos de intenção, o que o agente tem permissão para fazer;

mandatos de carrinho, o que está sendo comprado; e mandatos de pagamento, como e quando os recursos podem ser transferidos. O ledger que os armazena precisa ser consultável em dimensões estruturadas e semânticas, somente anexável na camada da aplicação e capaz de absorver novos tipos de mandato conforme os protocolos evoluem. É exatamente isso que um armazenamento unificado de documentos com pesquisa vetorial entrega de forma nativa.

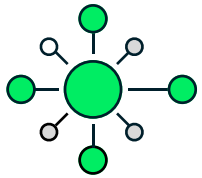
Para as equipes de varejo, a conclusão é que ser compatível com agentes não é apenas um problema de integração. É uma oportunidade de arquitetura. As marcas com maior probabilidade de oferecer suporte a transações com agentes em 2027 são aquelas cujos ledgers já são coerentes hoje.

O arco se fecha aqui. Começamos com o peso que arquiteturas fragmentadas impõem a todos os projetos. Terminamos com uma arquitetura que não apenas absorve a era dos agentes, mas lhe dá sustentação.

Figura 20.
Conversa
UCP versus
compromisso AP2



LEIA O
ARTIGO



O que vai sobreviver ao agente

No início desta edição, duas equipes iniciaram o mesmo projeto de IA no mesmo dia. Ao final do trimestre, uma já havia colocado a solução em produção. A outra tinha uma demo impressionante e nenhum caminho para a produção. É uma cena familiar hoje: o agente funciona, os slides impressionam e o modelo se comporta bem pelo tempo suficiente para convencer a sala. Então chegam as perguntas reais. Quem é responsável pelos dados? A resposta pode ser auditada? O que acontece quando o cliente está esperando, o regulador está observando e o sistema precisa agir?

Ao longo destas páginas, a resposta foi a mesma. A McKesson não protegeu a segurança dos pacientes com um prompt inteligente. A LG U+ não melhorou o atendimento ao cliente em tempo real mantendo inteligência e operações separadas. As instituições financeiras, fabricantes, varejistas e construtores apresentados nesta edição não estão esperando o modelo perfeito. Eles estão fazendo algo mais discreto e mais durável: construindo a camada que permite que

a inteligência se torne confiável o suficiente para ser utilizada.

Não será o espetáculo do agente, mas a fundação sobre a qual ele foi construído. A memória que persiste. As permissões que acompanham a ação. A verdade operacional que chega no momento certo. A trilha de auditoria que comprova o que aconteceu depois.

Os modelos serão substituídos. Os frameworks serão renomeados. Os protocolos vão amadurecer.

As arquiteturas vencedoras serão aquelas capazes de absorver todas essas mudanças sem exigir que a empresa comece tudo de novo.

A era agêntica não será impulsionada apenas por agentes. Ela será impulsionada pela camada de dados que torna seguro confiar neles.

→ As páginas finais desta edição incluem um livro de arquitetura e três skill badges para leitores prontos para começar.



Architectures for the Intelligent AI-Ready Enterprise

Frameworks práticos para a criação de arquiteturas prontas para IA que ofereçam valor comercial contínuo.

Sobre este recurso

Entregue isto ao arquiteto que terminar a leitura desta edição e perguntar, com razão, onde está o restante do conteúdo.

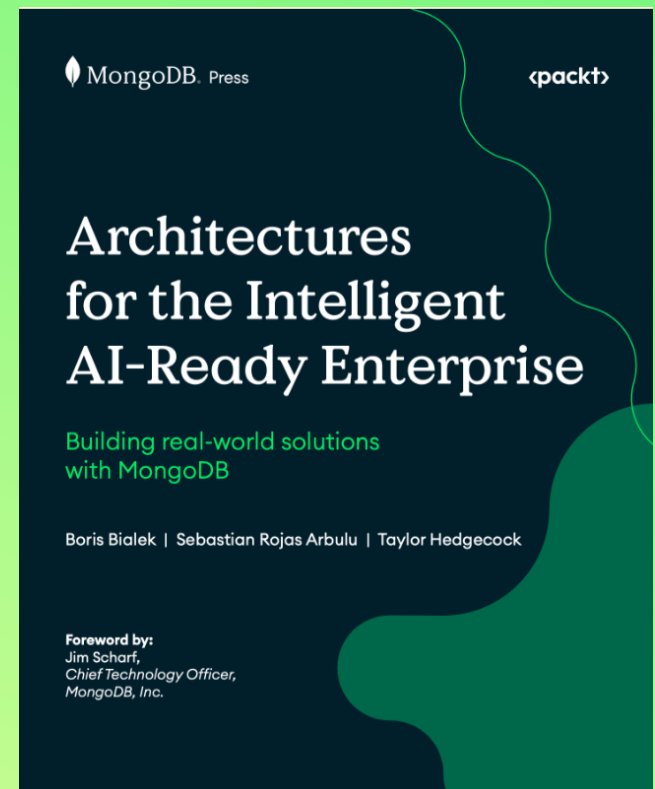
A revista apresenta, em formato narrativo, o argumento a favor de uma camada unificada de inteligência. O livro é a arquitetura prática por trás dessa visão: mais de 500 páginas de padrões, estudos de caso e capítulos para os quais uma edição deste tamanho não tem espaço.

Mais de 25 casos de uso de IA em mais de seis setores. Mais de 15 estudos de caso corporativos, incluindo Novo Nordisk e Base39. As implementações de parceiros com Iguazio by McKinsey & Company, Cognigy, RegData, Fireworks AI, Dataworkz e Encore, das quais esta edição pôde apresentar apenas trechos.

Capítulos fundamentais sobre o que um Sistema de Ação realmente exige, o caminho de modernização que leva sistemas legados a um estado preparado para IA sem começar do zero, a proteção semântica de dados que permite que a IA opere em setores regulados e os padrões de agentes e multiagentes que transformam uma resposta em uma ação.

Com prefácio de Jim Scharf, diretor de tecnologia da MongoDB.

Crie para durar mais do que o modelo.



[SOLICITE O RECURSO](#)



[LEIA A AMOSTRA GRÁTIS](#)

Skill Badges

Credenciais práticas que levam os leitores do conceito à implementação funcional. Cada badge combina um módulo específico com um laboratório prático, obtido em menos de uma hora, útil para o restante de um projeto de IA.



**GANHE
O SELO**



Voyage AI com MongoDB

Construa aplicações de IA usando embeddings Voyage AI armazenados e pesquisados nativamente no MongoDB Atlas. Cobre modelos de embedding, ingestão e fundamentos de pesquisa semântica. O caminho mais rápido de “já ouvi falar de embeddings” até “implementei uma pesquisa semântica funcional”.



**GANHE
O SELO**

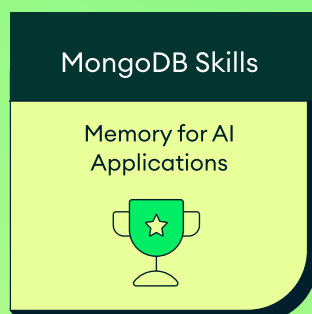


Desempenho da pesquisa vetorial

Use nós de pesquisa, técnicas de quantização e visualizações para dimensionar cargas de trabalho vetoriais do protótipo à produção no MongoDB Atlas. O selo que se paga na primeira vez que você evita uma regressão do desempenho na produção.

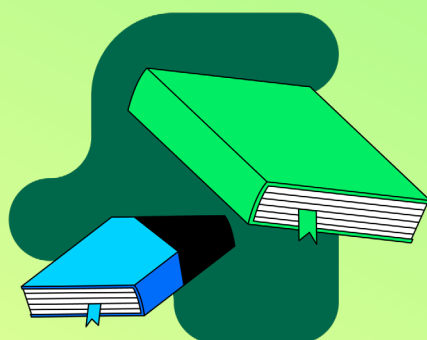


**GANHE
O SELO**



Memória para aplicações com IA

Crie memória durável e recuperável para agentes de IA. Contexto de curto prazo, recuperação de longo prazo e padrões que mantêm os sistemas multiagentes coerentes ao longo do tempo. Se seus agentes se esquecem demais, comece com esse módulo.



**AVANCE NA
CARREIRA**

A MongoDB Education ajuda os estudantes a desenvolver habilidades práticas do MongoDB em treinamentos gratuitos e flexíveis, caminhos de aprendizagem baseados em funções, skill badges e certificações. Com conteúdo auto guiado, laboratórios práticos e instrução especializada, a MongoDB University facilita o aprendizado, a validação das habilidades e o desenvolvimento.

Recursos adicionais

Para onde ir a seguir, dependendo do que você quer fazer.



**VISITE A
CENTRAL**

MongoDB Atlas Architecture Center

Melhores práticas e orientações para projetar e operar o MongoDB Atlas em escala empresarial. Estruturado em torno do Atlas Well-Architected Framework: segurança, eficiência operacional, desempenho, confiabilidade e otimização dos custos. A referência que arquitetos e engenheiros de plataforma consultam quando precisam de um projeto coeso.



**CONSULTE AS
HISTÓRIAS DOS
CLIENTES**

Histórias de sucesso de clientes

Além da McKesson, LG U+ e as outras apresentadas nesta edição, a MongoDB publica novas histórias de sucesso de clientes todos os meses. A biblioteca completa é pesquisável por setor, caso de uso e resultado. Um bom ponto de partida quando você está construindo internamente um caso de negócios e precisa de um cliente de referência no seu setor.



**COMECE A
APRENDER**

Central de Aprendizagem de IA

Um roteiro selecionado de tutoriais, implementações de referência e skill badges para desenvolvedores que criam aplicações de IA no MongoDB. Organizado por nível de habilidade (skill), a partir do “só preciso lançar um protótipo RAG” até “estou projetando uma plataforma multi-agente para produção”.



Quer aprender mais?
Fale conosco.



 MongoDB