Digibee

IA na integração corporativa

Quando ela é a escolha certa, quando não é e o que realmente significa uma plataforma de integração AI-native

A versão resumida

Agentes de programação, como o Claude Code, são excelentes para trabalhos pontuais de integração: migrações únicas, sistemas bem documentados e scripts de baixo risco. Eles deixam a desejar quando o assunto são as necessidades da integração corporativa: execuções agendadas, lógica de retentativa, trilhas de auditoria, gerenciamento de credenciais e monitoramento capaz de identificar falhas silenciosas. Essa diferença não é um problema de maturidade que será resolvido com modelos melhores. Ela é estrutural.

Quando um modelo atinge o limite do fenômeno conhecido como *lost in the middle* em uma janela de contexto muito longa, ele passa a recorrer aos dados com os quais foi treinado e produz código que passa nos testes, mas falha silenciosamente em produção, às vezes apenas seis meses depois.

Neste material, apresentamos um framework de decisão para ajudar a classificar os itens do backlog de acordo com o que eles realmente precisam, e não com a ferramenta que está em evidência no momento.

A lógica determinística continua sendo a melhor escolha quando não há margem para falhas, quando um regulador pode exigir uma trilha de auditoria ou quando um desenvolvedor experiente consegue escrever as regras em uma tarde. A IA agêntica faz sentido quando as entradas são imprevisíveis ou quando a tarefa exige um nível de julgamento que um conjunto de regras não consegue representar.

A maior parte do trabalho, porém, fica entre esses dois extremos: um modelo que chamamos de Deterministic Plus, em que um pipeline governado faz o trabalho pesado e um LLM fica responsável apenas por uma etapa específica e limitada, como identificar exceções em uma nota fiscal ou fazer a triagem de um chamado de suporte.

Na prática, a maior parte do backlog continua pertencendo ao lado determinístico. Adicionar IA a um processo que já funciona não o torna melhor; apenas torna sua governança mais complexa.

SUMÁRIO EXECUTIVO

Na última seção, mostramos o que a Digibee construiu para se tornar verdadeiramente AI-native.

Isso incluiu o desenvolvimento de uma nova interface pensada especificamente para agentes e a incorporação de conhecimento especializado em integração a agentes dedicados. Também propomos um teste simples: pergunte a qualquer fornecedor de integração se ele reconstruiu sua plataforma para agentes ou se apenas adicionou um chatbot.

LIMITES

Porque agentes de programação não conseguem dar conta de integrações corporativas e por que essa lacuna é estrutural.

FRAMEWORK

Organize o backlog de acordo com a necessidade: determinístico, agêntico ou determinístico-plus.

AI-NATIVE

O que é necessário para reconstruir uma plataforma de verdade para agentes e como isso muda a forma de avaliar fornecedores.

SUMÁRIO

O que você encontrará neste material

INTRODUÇÃO

O verdadeiro desafio não é adotar IA. É entender onde ela realmente faz sentido.

Conselhos de administração, CEOs e outros executivos estão impulsionando estratégias AI-first, enquanto diferentes áreas das empresas passam a acreditar que agentes de IA podem resolver qualquer problema de software. Ao mesmo tempo, o volume de iniciativas cresce sem parar, aumentando o backlog das equipes de integração, que já não conseguem acompanhar a demanda.

Nesse cenário, especialistas em integração passaram a receber ferramentas de IA genéricas, desenvolvidas para projetos greenfield (criados do zero), com a expectativa de entregar resultados mais rapidamente. Na prática, porém, essas ferramentas oferecem pouca ajuda quando aplicadas ao contexto da integração empresarial.

Neste guia, analisamos a relação entre IA e integração sob três perspectivas.

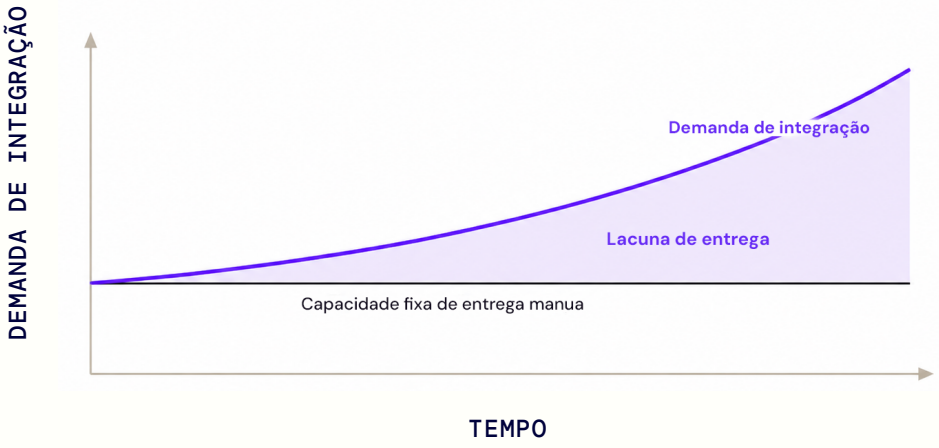
A demanda continua crescendo.

Os agentes genéricos não acompanham.

- RISK Treinar a IA com padrões e melhores práticas
- RISK Acesso seguro a sistemas e dados
- RISK Decisões governadas
- RISK Facilidade de colaboração entre equipes
- RISK Eficiência por meio da reutilização
- RISK Padronização dos protocolos da organização
- RISK Suporte a SLAs para casos de uso de integração

A LACUNA DE ENTREGA

A demanda continua crescendo. A capacidade manual, não.



CH. 01

Onde os agentes falham

Onde agentes de programação genéricos falham em integrações corporativas e por que essa lacuna é estrutural, não temporária.

CH. 02

Um framework de decisão

Como associar cada tarefa à solução certa: determinística, agêntica ou "determinística-plus".

CH. 03

Construindo uma plataforma AI-native

Nossa jornada na construção de uma plataforma AI-native e os desafios que tivemos que resolver para chegar até aqui.

A maioria das organizações encara a adoção da IA como uma decisão binária: usar ou não usar.

Mas o verdadeiro desafio não é esse.
O que realmente importa é entender onde a IA agrega valor e onde ela não agrega.

CAPÍTULO 01

Por que agentes de programação não conseguem lidar com integração empresarial

Na Digibee, todas as equipes utilizam o Claude Code. Fazemos questão de começar por esse ponto porque o que você vai ler a seguir não é uma crítica feita por quem não conhece a ferramenta.

INTEGRAÇÃO DE SISTEMAS EXIGE UMA ABORDAGEM DIFERENTE

Os agentes de programação mudaram a forma como nossos engenheiros trabalham. Eles aceleram protótipos, reduzem o esforço cognitivo em tarefas repetitivas e aumentam significativamente a produtividade.

Se a sua equipe ainda não utiliza esse tipo de ferramenta, provavelmente está deixando ganhos reais de produtividade na mesa.

O problema é que o sucesso desses agentes no desenvolvimento de software greenfield criou a impressão de que eles podem ser aplicados a qualquer desafio tecnológico. Integração empresarial é uma categoria completamente diferente de trabalho. Ela possui requisitos, restrições e formas de falha que agentes de programação simplesmente não foram projetados para resolver.

Em que agentes de programação realmente são bons?

Agentes de programação genéricos funcionam muito bem em cenários bastante específicos:

- Os sistemas envolvidos são bem documentados;
- O trabalho será executado apenas uma vez;
- Eventuais falhas têm baixo impacto e são facilmente recuperáveis;
- Não existe risco para ambientes de produção durante os testes.

Migrar dados de uma API REST para um arquivo CSV? Ótimo caso de uso.
Criar rapidamente um script para consumir um endpoint público? Perfeito.
Montar um processo temporário de ETL sem dependências críticas? Também faz sentido.

Os problemas começam quando a integração precisa operar em condições muito mais complexas.

AS EMPRESAS PRECISAM DE INTEGRAÇÕES QUE:

- Funcionem de forma confiável, seja em execuções agendadas ou acionadas por eventos em tempo real;
- Contem com lógica de retentativa e mecanismos de recuperação em caso de falha;
- Mantenham trilhas de auditoria mostrando o que foi transferido, quando e por quê;
- Gerenciem credenciais sem espalhar configurações personalizadas por dezenas de ambientes;
- Monitorem falhas silenciosas antes que elas interrompam um processo de negócio;
- Possam ser mantidas, evoluídas e compreendidas por outras pessoas além de quem as desenvolveu.

Agentes de programação não entregam nada disso. Eles geram código. E esse código precisa ser hospedado em algum lugar, protegido, monitorado e mantido por alguém. Quando essas responsabilidades não são tratadas adequadamente, o resultado pode ser um processo caro, difícil de manter ou até mesmo um problema de conformidade.

Por que essa diferença é estrutural

É natural imaginar que essas limitações desaparecerão à medida que os modelos evoluírem. Mas o problema não é circunstancial. Ele é estrutural. A integração empresarial depende de conhecimento acumulado ao longo dos anos, incorporado em conectores e componentes prontos que entendem detalhes como a forma como o NetSuite executa determinadas operações, como o SAP trata idempotência ou como o Salesforce organiza seu modelo de dados. Um agente de programação precisa raciocinar sobre todas essas particularidades do zero, toda vez que gera uma integração.

É justamente aí que surgem bugs sutis, que aparecem apenas em produção.

Agentes de programação precisam redescobrir cada sistema

Os sistemas com os quais as empresas precisam se integrar, em muitos casos, são mal documentados. Essa é a realidade de sistemas legados, contratos de dados construídos ao longo de anos e ambientes que acumularam décadas de complexidade.

Grande parte do comportamento que realmente importa só aparece quando o sistema está sob carga. E, normalmente, isso não está documentado. Mesmo quando existe documentação, o agente pode simplesmente deixar de utilizá-la.

Pesquisadores já demonstraram que LLMs podem perder informações que aparecem no meio de uma janela de contexto muito extensa um fenômeno conhecido como lost in the middle. Quando isso acontece, o modelo passa a se apoiar principalmente nos próprios dados de treinamento. Quanto mais específico ou menos conhecido for um pacote, uma API ou um sistema empresarial, maior a chance de o agente produzir um código que simplesmente não funciona naquele ambiente.

Agentes não aprendem automaticamente com isso. Eles não acumulam conhecimento entre diferentes sessões. Toda vez que recebem uma nova solicitação, precisam interpretar novamente regras complexas de sequenciamento, comportamentos específicos de idempotência e diversas outras particularidades.

O resultado costuma ser um código altamente personalizado, difícil de entender e ainda mais difícil de governar de forma consistente em toda a empresa.

O pior cenário não é aquele em que a integração quebra logo após o deploy.

É quando ela continua funcionando aparentemente bem, mas passa a corromper registros ou gravar transações duplicadas em situações muito específicas, que só aparecem sob carga real. E, muitas vezes, o CIO só descobre o problema meses depois.

> O EFEITO "LOST IN THE MIDDLE"
>
> LLMs podem perder de vista informações localizadas no meio de uma janela de contexto extensa e substituí-las pelo conhecimento presente em seus dados de treinamento.
> Quando isso acontece em APIs empresariais pouco conhecidas ou em sistemas legados, o resultado pode ser um código que passa nos testes, mas falha em produção.

Agentes de código criam código. Integrações exigem mais

Alguém precisa provisionar a infraestrutura, garantir a disponibilidade, gerenciar a escalabilidade, implementar lógicas de retenção, criar mecanismos de recuperação de falhas e garantir visibilidade sobre o que acontece em produção.

Você pode tentar contornar cada uma dessas lacunas individualmente por meio de prompts. Mas, nesse ponto, você já está construindo não apenas uma integração, mas também tudo o que é necessário para mantê-la funcionando, cada elemento com sua própria carga de manutenção.

Agentes de código também são otimizados para iterar rapidamente, não para falhar de forma segura. Quando uma integração falha em produção, pagamentos deixam de ser processados, cotações de frete deixam de ser realizadas ou pedidos deixam de ser enviados. Um agente que gera lógica de negócio sem logs, alertas ou circuit breakers entrega apenas metade da solução, justamente a parte mais fácil.

APIs mudam. Tokens expiram. Sistemas são atualizados. Um agente não mantém uma relação com aquilo que construiu nem se lembra do motivo por trás das decisões tomadas.

O conhecimento sobre a integração fica concentrado na pessoa que criou os prompts. Quando essa pessoa se ausenta, mesmo que seja apenas durante as férias, o contexto por trás de cada decisão de arquitetura também deixa de estar disponível.

> O CUSTO ACUMULADO
>
> **“Uma integração criada com um agente de código é administrável. Cem delas, cada uma com sua própria lógica de retenção, gerenciamento de credenciais e premissas sobre tratamento de erros, representam cem bases de código diferentes para governar.”**

Agentes de programação não são responsáveis pelo que constroem

Dívida técnica é visível. Dívida de governança, não. Ela só se torna evidente quando acontece um incidente de segurança, uma auditoria aponta uma não conformidade ou chega o último dia de trabalho da pessoa que criou aquela integração.

Sem um artefato intermediário, não há como auditar

Agentes de programação genéricos geram código diretamente a partir de um prompt.
Não existe um artefato intermediário. Não há uma especificação funcional, um documento de mapeamento nem um registro estruturado dos requisitos, dos casos de exceção ou do comportamento esperado diante de erros.

Essa é uma lacuna importante de controle para qualquer equipe que precise entender, auditar ou modificar uma integração depois que ela entra em produção.

Toda integração que movimenta dados entre sistemas precisa manter um registro confiável do que foi transferido, quando isso aconteceu e por quê.
O código gerado, por si só, não fornece esse nível de rastreabilidade.

Credenciais sem um responsável

Chaves de API, tokens OAuth e credenciais de serviço precisam ser armazenados, protegidos, ter seus escopos definidos corretamente e passar por rotação periódica. O código gerado não toma nenhuma decisão sobre isso e tampouco assume essa responsabilidade.

Cada integração personalizada passa a ser um novo ativo que precisa ser protegido, uma nova superfície de ataque e uma nova dependência sem um modelo claro de governança.

O custo cresce rapidamente

Cem integrações, cada uma com sua própria lógica de retentativa, seu próprio tratamento de erros e suas próprias premissas sobre credenciais, representam cem bases de código diferentes para proteger, atualizar e manter. O custo cresce rapidamente.

DÍVIDA DE GOVERNANÇA, ITEM POR ITEM

- ☑ Nenhum documento de especificação ou mapeamento
- ☑ Nenhum registro estruturado de casos excepcionais
- ☑ Nenhum modelo definido com responsáveis pela rotação de credenciais
- ☑ Nenhum registro confiável do que foi alterado e por quê

O que as skills resolvem e o que elas não resolvem

Equipes que utilizam o Claude Code costumam recorrer às chamadas skills, que são bibliotecas de exemplos e padrões documentados que ajudam o agente a produzir resultados melhores em tarefas bem definidas. Aplicar esse conceito ao universo da integração significaria pesquisar como cada sistema se comporta além do que está documentado: casos de exceção, limites de requisição, particularidades entre versões e inúmeros outros detalhes. E esse não é um trabalho pontual. sistemas evoluem constantemente, e qualquer skill precisaria evoluir junto com eles.

Mesmo skills muito bem desenvolvidas não eliminam o risco de gerar código com falhas. Os LLMs são probabilísticos por natureza. Eles podem produzir um código perfeito para um prompt e, logo em seguida, gerar outro que passa em todos os testes, mas falha silenciosamente em produção. Violações de idempotência, por exemplo, normalmente não geram erros imediatos. Gravações duplicadas continuam se acumulando até que, dias depois, os processos de reconciliação deixam de fechar.

Uma skill pode documentar como reagir a uma falha. Mas ela não é capaz de identificar que essa falha aconteceu. O agente não acompanha a integração em execução. Uma plataforma, sim. Ela monitora falhas silenciosas, dispara alertas antes que os processos de reconciliação sejam comprometidos e mantém toda a camada operacional funcionando entre uma entrega e outra.

Ferramentas excelentes. Mas não para este trabalho.

Agentes de programação deixam a desejar quando o assunto é integração empresarial recorrente, crítica e em larga escala. Não porque a tecnologia não seja impressionante, mas porque o problema simplesmente não se encaixa na ferramenta. Sabendo disso, a pergunta deixa de ser se a IA deve ou não fazer parte da integração. A pergunta passa a ser: em quais situações ela faz sentido e em quais não faz?

A seguir: um framework para avaliar cada item do seu backlog. →

CAPÍTULO 02

IA não é tudo ou nada

Como partir dos resultados, e não da solução, ao resolver desafios de integração.
Você tem um ano inteiro de trabalho acumulado no backlog. Como decidir qual é a solução certa para cada problema?

Comece pelos resultados, não pela tecnologia

Você tem um backlog para um ano inteiro, está correndo para entender onde a IA realmente agrega valor e, ao mesmo tempo, precisa lidar com limitações reais de sistemas, equipes e dados.

Como decidir qual é a solução certa para cada problema?

Analise o backlog item por item. Algumas tarefas se encaixam perfeitamente em fluxos determinísticos, nos quais a IA apenas adicionaria risco, custo e complexidade. Outras apresentam problemas genuinamente ambíguos, nos quais a IA agentic viabiliza níveis de automação que antes simplesmente não eram possíveis. E um número surpreendente de casos fica entre esses dois extremos.

É natural ter preferência por uma plataforma, uma linguagem ou pela tecnologia mais recente. Mas, antes de escolher a ferramenta, pergunte qual resultado o processo precisa entregar.

SINAIS DE QUE VALE A PENA CONSTRUIR DE FORMA DETERMINÍSTICA

Tolerância zero a falhas

Com 99% de precisão, um processo executado 100.000 vezes ainda produzirá 1.000 erros. Para determinados casos de uso, esse custo é simplesmente inaceitável.

Explicabilidade das decisões

Se um regulador ou órgão de compliance puder questionar por que o sistema tomou determinada decisão automatizada, mantenha a lógica determinística. Ela produz uma trilha de auditoria. LLMs geram resultados probabilísticos e, em alguns casos, podem desrespeitar instruções. Quando isso acontece, entender o motivo pode ser extremamente difícil ou até impossível.

Lógica simples de A para B

Se um desenvolvedor experiente consegue implementar as regras em uma tarde, implemente as regras. Mantenha o simples simples e reserve a IA para os problemas realmente complexos.

A MATEMÁTICA DOS “99% DE PRECISÃO”

Com 99% de precisão, um processo executado 100.000 vezes estará errado 1.000 vezes. Imagine as consequências de 1.000 folhas de pagamento processadas incorretamente.

SINAIS DE QUE A IA AGÊNTICA AGREGA VALOR

Entradas imprevisíveis

Se o projeto precisa interpretar documentos em formatos variados, solicitações em linguagem natural ou outras fontes de dados não estruturadas, os LLMs normalmente são a única solução viável.

Tomada de decisão contextual em tempo de execução

Um LLM consegue raciocinar sobre entradas ambíguas de uma forma que conjuntos de regras não conseguem. Se o sistema precisa exercer julgamento, ele precisa de IA.

Pontos de atenção em casos menos óbvios

- **Throughput e latência:** Processos de alto volume e com requisitos rigorosos de tempo de resposta tendem a favorecer abordagens determinísticas. Inferência de IA adiciona latência e custo quando executada em escala.
- **Previsibilidade de custos:** Workflows agentic carregam custos operacionais variáveis em qualquer escala. Modele esses custos cuidadosamente antes de seguir por esse caminho.
- **Custo total de propriedade (TCO):** Pipelines desenvolvidos em código costumam exigir mais tempo da equipe no início. Um agente criado rapidamente pode acabar custando mais ao longo do tempo. Um pipeline que leva algumas horas para ser construído, mas falha em 2% das execuções, é realmente mais barato do que outro que leva uma semana para ficar pronto e nunca falha? Depende.

ALERTA PARA CUSTOS FORA DE CONTROLE

O caso do agente que consumiu US$ 47 mil em 11 dias foi extremo. Ainda assim, workflows agentic carregam custos operacionais variáveis em qualquer escala. Se previsibilidade orçamentária é importante para o seu negócio, modele esses custos antes de tomar uma decisão. (Fonte)

O espectro

O trabalho moderno de integração existe em um espectro que vai do determinístico ao agêntico. Entre esses dois extremos existe uma ampla faixa intermediária, na qual uma base determinística é complementada por etapas pontuais com IA agêntico. Esse framework pode ser aplicado tanto a um workflow específico quanto ao portfólio inteiro de integrações de uma organização.

Determinístico

Execução confiável, auditável e repetível a baixo custo. A base das integrações corporativas.

Determinístico Plus

Um pipeline governado, aprimorado com uma ou mais etapas agênticas delimitadas que agregam valor claro.

Agêntico

Lida com ambiguidade, inputs variáveis, síntese e raciocínio, com resultados e custos variáveis.

Hoje, na maioria das empresas com as quais trabalhamos, a maior parte do backlog ainda pertence ao **lado determinístico** e isso faz sentido. São problemas de integração conhecidos e já resolvidos. Workflows agêntico abrem novas oportunidades de automação, e nossos clientes vêm encontrando cada vez mais maneiras de adicionar capacidades agêntico a pipelines determinísticos sem comprometer sua base.

Auxílio à decisão

	Determinístico	Determinístico Plus	Agêntico
USE QUANDO	Os requisitos são estáveis, as entradas são estruturadas, governança é essencial e falhas não são uma opção.	Um workflow governado pode gerar mais valor ao incorporar uma etapa agentic específica. Casos de exceção ou determinados resultados se beneficiam do julgamento da IA.	O problema exige julgamento, síntese ou criatividade, e alguma variabilidade nos resultados é aceitável.
EXEMPLOS COMUNS	■ Fluxos de recuperação de senha ■ Exportação de logs para auditoria de compliance ■ Escalonamento de alertas de fraude bancária	■ Roteamento de chamados de Service Desk com notas de triagem geradas por IA ■ Processamento de faturas com sinalização de exceções por LLM ■ Geração de release notes a partir de dados estruturados de commits	■ Resumo de contratos e identificação de riscos ■ Elaboração de respostas para RFPs ■ Geração de conteúdo para redes sociais a partir de um prompt
FIQUE ATENTO	Não subestimar o valor da IA em casos de exceção e em entradas não estruturadas.	Não permitir que a etapa agêntico ultrapasse seu papel delimitado.	Custos variáveis em escala, requisitos de auditoria e modos de falha silenciosos.

Workflows determinísticos e Workflows agênticos

Antes de combiná-los, vale entender com precisão onde cada extremo desse espectro se destaca e quais são os custos envolvidos em cada abordagem.

Workflows Determinísticos

A base das integrações corporativas. Eles oferecem execução confiável, auditável e replicável a baixo custo. Quando os requisitos são estáveis e os inputs são estruturados, pipelines de código quase sempre são a escolha certa, embora muitas vezes sejam subvalorizados no atual cenário de IA.

- Workflows de recuperação de senha
- Exportações de logs de auditoria de compliance
- Escalonamento de alertas de fraude bancária

Workflows Agêntico

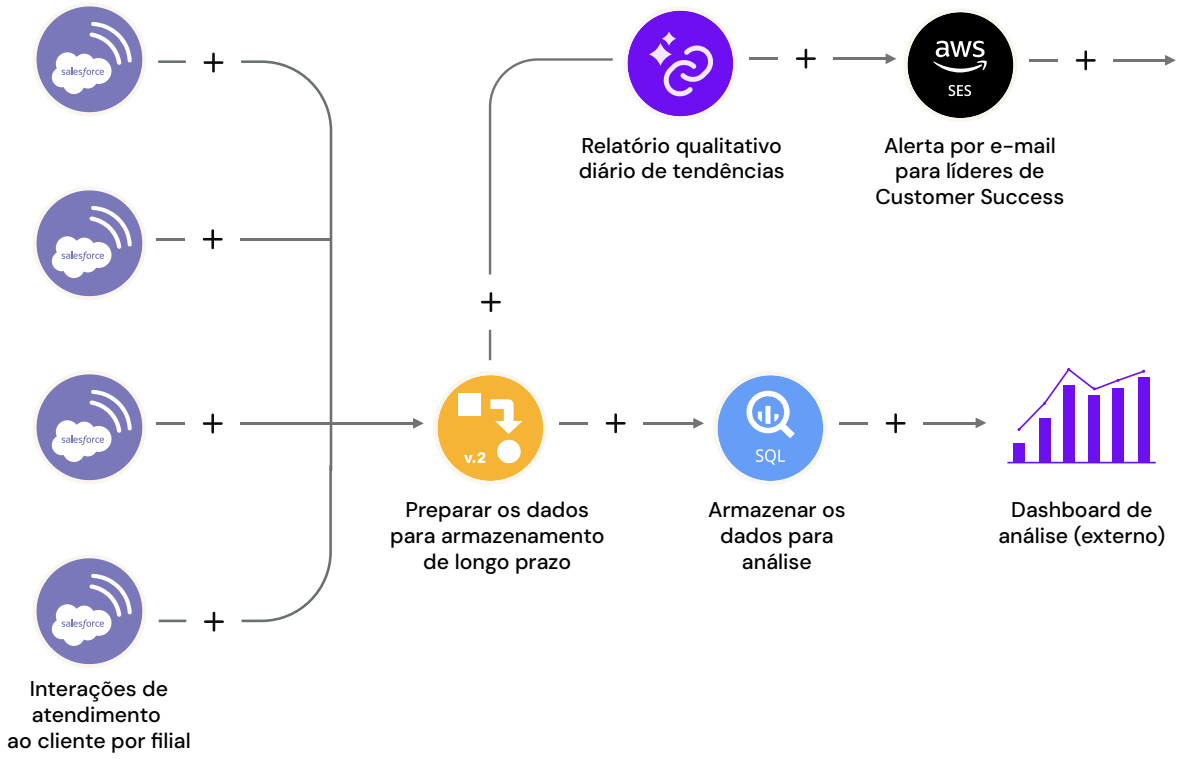
Lidam com o que o código não consegue: ambiguidade, inputs variáveis, síntese e raciocínio. Eles permitem automatizar decisões que antes dependiam de pessoas, muitas vezes especialistas caros e com disponibilidade limitada, para realizar tarefas repetitivas e trabalhosas. O trade-off é que os resultados são inerentemente variáveis, a execução custa mais e o trabalho é mais difícil de auditar e monitorar.

- Resumo de contratos e identificação de riscos
- Elaboração de respostas a RFPs
- Geração de posts para redes sociais

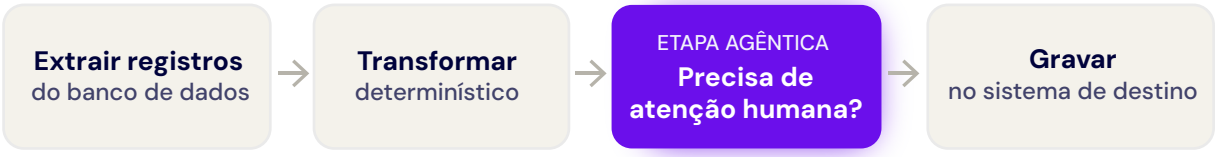
Determinístico Plus

A maioria dos workflows de integração deve começar como pipelines determinísticos. O conceito de Determinístico Plus descreve o que acontece quando um workflow comprovado e governado é aprimorado com uma ou mais etapas agêntico que entregam valor claro e bem delimitado.

Não se trata de uma divisão meio a meio. O pipeline determinístico continua sendo a espinha dorsal da solução. A etapa agêntico entra apenas para agregar valor de forma pontual.



UM PIPELINE GOVERNADO, UMA ETAPA DE IA DELIMITADA



A IA atua em uma única etapa. Todo o restante permanece previsível, auditável e barato de executar.

Uma abordagem complementar: workflows separados

Outra forma de aplicar esse padrão é manter workflows distintos para responsabilidades distintas. Um pipeline totalmente determinístico faz a ingestão ou movimentação de um lote de dados. Em seguida, ele aciona um workflow agêntico separado, responsável apenas por analisar esse lote e gerar insights. Um de nossos clientes, por exemplo, está experimentando um workflow centralizado de avaliação para medir o desempenho de outros workflows em todo o seu ambiente de integração. A lógica da integração permanece limpa e previsível; a IA atua apenas onde a variabilidade é aceitável.

O modelo Determinístico Plus permite que as organizações capturem o valor da IA sem expor sua infraestrutura crítica aos modos de falha característicos dos workflows agêntico. E é assim que a maioria dos ambientes de integração deve evoluir: de forma incremental, deliberada e com a governança preservada.
Esse é o padrão para o qual a arquitetura da Digibee foi concebida.

A MELHOR ESTRATÉGIA É UMA ESTRATÉGIA DELIBERADA.

Os líderes de integração mais preparados não são aqueles que fazem mais coisas com IA. São aqueles que entendem claramente os trade-offs. Se um workflow determinístico funciona bem, não há motivo para alterá-lo.
Seu backlog não precisa ser um backlog de IA. Ele precisa ser um backlog resolvido.

CAPÍTULO 03

O que realmente significa ser AI-native

Há um ano, nos fizemos uma pergunta simples: como construiríamos nossa plataforma se estivéssemos começando do zero hoje?
Foi dessa pergunta que nasceu a plataforma que construímos e é esse raciocínio que vamos compartilhar neste capítulo.

O trabalho de engenharia por trás do conceito

Fundamos a Digibee para aproveitar ao máximo o potencial da computação em nuvem justamente quando ela estava redefinindo o que era possível para softwares B2B. Naquela época, muitas plataformas legadas de iPaaS se apresentavam como cloud-native, mas continuavam executando arquiteturas de lift-and-shift. Nossos clientes perceberam essa diferença na prática desde o início. E não estamos dispostos a aceitar menos do que isso na era da IA.

Reavaliamos a Digibee a partir dos primeiros princípios. Nosso objetivo sempre foi permitir que profissionais de integração construíssem workflows de nível empresarial, com gerenciamento de credenciais, segurança e infraestrutura em nuvem com escalabilidade automática, na maior velocidade que a tecnologia disponível permitisse. Com a chegada da IA agêntico, o trabalho deixa de ser apenas execução e passa a envolver delegação e estratégia. Para transformar essa visão em realidade, tivemos que resolver três desafios:

01 Tornar a experiência da plataforma tão intuitiva para agentes quanto é para pessoas.

02 Especializar e capacitar agentes de IA para atuar como engenheiros de integração experientes.

03 Evoluir a forma como as pessoas trabalham com essa nova força de trabalho digital.

Acesso para agentes

Pessoas precisam de interfaces visuais: menus, botões e campos de formulário. Agentes de IA precisam de pontos de interação atômicos e autoexplicativos. Diante de vários endpoints com nomes ambíguos, um LLM tende a escolher um deles com confiança e seguir em frente. Por isso, criamos APIs e ferramentas baseadas em MCP documentadas com uma linguagem clara, autocontida e pensada para um leitor que não dispõe de nenhum contexto adicional.

Entrega de contexto

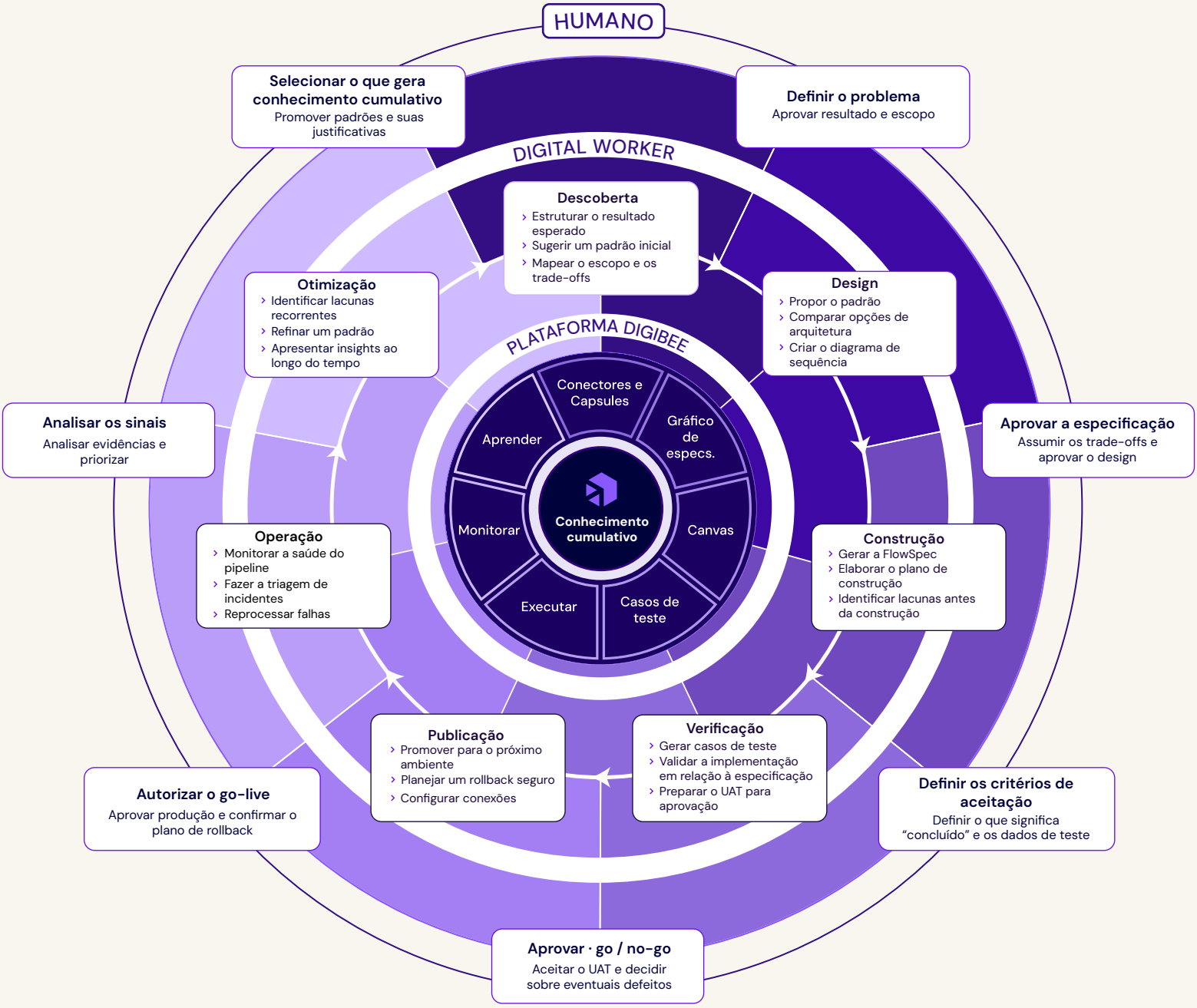
Quando um usuário desenvolve um workflow no canvas da Digibee, a plataforma exibe a mensagem de erro do conector que falhou. Poderíamos fornecer todo o log de execução, mas optamos por não fazer isso. Usuários não precisam desse volume de informação, e o log completo apenas tornaria mais difícil encontrar o que realmente importa.

Já os LLMs se beneficiam de analisar todo o histórico de execução. Um agente nem sempre compreende qual era a função de cada etapa anterior à falha, e esse contexto adicional melhora sua capacidade de raciocínio. Usuários humanos constroem contexto de diversas formas: pela representação visual do workflow e pela experiência acumulada em integrações semelhantes. Agentes, por outro lado, dependem exclusivamente das informações recebidas em tempo de execução. Por isso, projetamos nossos serviços voltados para agentes para entregar contexto profundo, estruturado e otimizado para consumo por LLMs.

Especialização em integração

Agentes de programação são excelentes quando trabalham dentro de uma base de código. Já a integração corporativa acontece no nível do conhecimento organizacional. Os sistemas envolvidos costumam ser pouco documentados, e os dados de treinamento dos LLMs contêm pouquíssimos exemplos de como integrar sistemas legados específicos e pouco conhecidos. Na Digibee, esse conhecimento é codificado em um documento chamado FlowSpec, que descreve de forma estruturada como uma integração deve funcionar. É pouco provável que qualquer grande LLM tenha visto mais do que um número muito pequeno desses documentos durante seu treinamento.

Construir um workflow envolve diversas tarefas distintas. Obrigar um único agente a executar todas elas compromete a qualidade do resultado: o contexto cresce demais e o agente perde coerência. Por isso, transformamos nosso conhecimento em conjuntos específicos de prompts e os distribuímos entre subagentes especializados, cada um responsável por uma etapa do processo e capaz de produzir resultados em nível de especialista. O usuário nunca vê essa complexidade. Ele interage apenas com um único Digital Worker, que coordena os subagentes e reúne suas contribuições em um projeto completo e coeso de ponta a ponta.



Leitura do centro para fora: superfície da plataforma → Digital Worker (etapa) → validação humana (verificação)

Colaboração entre pessoas e IA

Nossa abordagem utiliza IA para elaborar projetos e workflows, sempre em colaboração com o usuário. Cabe ao usuário definir os objetivos, orientar o projeto e validar o resultado. O papel do especialista em integração não desaparece; ele evolui. Para tornar isso possível, desenvolvemos interfaces específicas para duas etapas: a definição da especificação da integração e a revisão do que foi construído pelo Digital Worker.

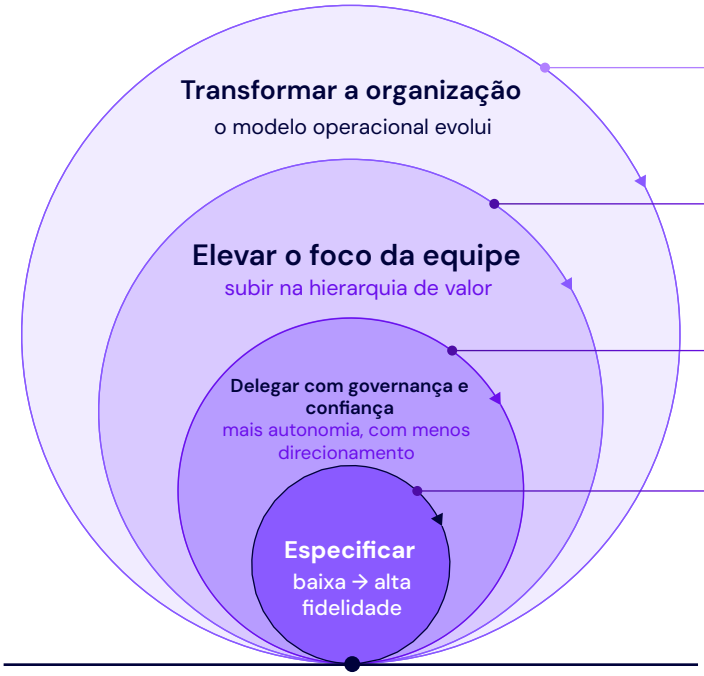
Validar o resultado foi a parte mais simples, pois já contávamos com o canvas da Digibee. Para o planejamento colaborativo, criamos uma nova interface que combina uma janela de conversa com um documento dinâmico organizado em múltiplas abas. O usuário começa descrevendo seu objetivo em alto nível e, a partir daí, trabalha em conjunto com o Digital Worker para construir uma especificação detalhada do projeto, contemplando casos de exceção, condições de disparo e estratégias de tratamento de erros.

Integração em um nível mais estratégico

Para os usuários da Digibee, ser AI-native significa trabalhar em um nível mais estratégico. Embora continuem revisando e ajustando o que o Digital Worker produz no canvas, eles deixam de gastar tempo arrastando e configurando manualmente cada componente da integração. Ao reduzir o tempo dedicado à execução operacional e acelerar a eliminação do backlog, os especialistas em integração passam a concentrar seus esforços em questões de maior impacto para o negócio: quais processos podem ser centralizados em cápsulas reutilizáveis? Onde ainda existem oportunidades de gerar valor que ninguém explorou? Historicamente, as equipes de integração raramente tiveram tempo para responder a esse tipo de pergunta.

SUBINDO NA HIERARQUIA DE VALOR

Cada ciclo se traduz em entregas reais, elevando o nível de valor.



Lead time de integração

CIO / VP de Engenharia
tempo entre a solicitação e a entrada em produção

Custo da integração

Head de Integração / Finanças
esforço e custo por integração

Velocidade da integração

Arquiteto / Tech Lead
volume de entregas após a delegação

Qualidade da integração

Líder de QA / Desenvolvedor de integração
certo de primeira, com menos defeitos

cada ciclo retorna à entrega real.

20X+
MAIS RÁPIDO

Nos primeiros testes com parceiros de design, os usuários concluíram workflows até 20X+ mais rápido.

O teste decisivo

Na era da computação em nuvem, o grande sinal de alerta era o lift-and-shift: migrar aplicações legadas para a nuvem sem mudar a forma como elas funcionavam, desperdiçando os benefícios que a nuvem realmente podia oferecer.

Na era da IA, existe um equivalente muito mais simples.

PERGUNTE A QUALQUER FORNECEDOR COMO ELE REDESENHOU SUA PLATAFORMA PARA AGENTES.

Como vocês redesenharam a plataforma para atender às necessidades de usuários agênticos?

Adicionar IA a um produto existente cria exatamente os mesmos problemas que o lift-and-shift criou anos atrás. Essa abordagem impede que você aproveite os benefícios de uma reconstrução genuína e mantém sua empresa presa por mais tempo do que o esperado em um estado intermediário entre o antigo e o novo.

Faça a mesma pergunta para os seus próprios workflows. Você pode descobrir que os maiores ganhos não vêm de acelerar processos existentes, mas de questionar se esses processos ainda deveriam existir da forma como existem hoje.

Veja a plataforma de integração AI-native em ação

A Digibee é a primeira plataforma de integração corporativa AI-native. Se você é responsável pela infraestrutura de integração da sua organização e quer ver, na prática, como essa abordagem funciona, Entre em contato conosco

Agende uma demo →

Tenha acesso antecipado →