AGENTES DE IA CORPORATIVOS NAS PLATAFORMAS MICROSOFT · PLAYBOOK

A plataforma é a primeira decisão. Persona e raio de impacto decidem o resto.

Um agente de IA corporativo é um tipo diferente de software: mantém um objetivo, guarda estado entre passos, chama ferramentas para mudar o mundo e para diante de um humano quando não consegue prosseguir com segurança. Este playbook roteia cada carga de agente entre as três plataformas da Microsoft, mapeia os quatro tipos de agentes que chegam à produção, apresenta a arquitetura de referência em seis camadas, e mostra como a supervisão humana escala com a autonomia, tudo visto pelo GitHub Copilot.

AUTORA

Paula Silva

CARGO

AI-Native Software Engineer

CONTATO

in/paulanunes

TEMPO DE LEITURA

~ 45 minutos, fim a fim

3	6	4
PLATAFORMAS	CAPÍTULOS	TIPOS DE AGENTES

40%

APPS CORPORATIVOS
ATÉ 2026

CAPÍTULOS

Chapter 00

Fundamentos de agentes

O que separa um agente de IA corporativo de um chatbot, os cinco componentes que todo agente reúne, e por que a escolha de plataforma vem antes da primeira linha de código.

Fundamentos →

Chapter 01

Três plataformas, um framework

Copilot Studio para usuários de negócio, GitHub Copilot para engenheiros, Azure AI Foundry para equipes de ML e plataforma, e a matriz de seis eixos que roteia cada carga.

Plataformas →

Chapter 02

Quatro tipos de agentes

Agentes de desenvolvimento, produtividade, processos de negócio e dados, a plataforma onde cada um se encaixa, e o guardrail que cada tipo exige antes de ir para produção.

Tipos de agentes →

Chapter 03

A arquitetura de referência

A pilha de agentes enterprise em seis camadas, cinco princípios que regem cada camada, três padrões de orquestração e quatro regras de design que mantêm uma falha contida.

Arquitetura →

Chapter 04

Supervisão humana

Por que mais autonomia exige mais guardrails, os quatro quadrantes de impacto e reversibilidade, três padrões de checkpoint, e a base de segurança por baixo deles.

Supervisão →

Chapter 05

Do piloto à frota

Um caso de leasing trabalhado sob gates Human-in-the-Loop, quatro padrões verticais que reutilizam sua coreografia, e um roadmap de doze meses com critérios de saída.

Roadmap →

DICAS DE LEITURA

> **ATALHOS DE TECLADO**
>
> Aperte `/` para buscar. Use `j` e `k` para mover entre capítulos. Aperte `t` para alternar tema. Aperte `g` para ir ao topo.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

00 AGENT FOUNDATIONS

O que é, de fato, um agente de IA corporativo, e por que a plataforma é a primeira decisão.

Um agente de IA corporativo não é um chatbot mais esperto. É um tipo diferente de software. Ele mantém um objetivo, guarda estado entre passos, chama ferramentas para mudar o mundo e para diante de um humano quando não consegue prosseguir com segurança. Este capítulo define o que um agente de fato é, nomeia os cinco componentes que todo agente reúne e mostra por que a decisão de plataforma vem antes da primeira linha de código, tudo visto pelo GitHub Copilot.

Agente versus chatbot: orientado a objetivos, com estado e usando ferramentas

Comece pelo que um agente não é. Um chatbot é stateless e reativo. Ele recebe uma pergunta e devolve uma única resposta. Não guarda memória entre turnos, não consegue chamar ferramentas e não tem noção de perseguir um objetivo. Pergunte a mesma coisa duas vezes e ele começa do zero a cada vez. Esse design serve bem para perguntas frequentes e fluxos de suporte roteirizados, e ainda serve para isso. Ele não é um agente. Um agente de IA corporativo é orientado a objetivos, mantém estado e usa ferramentas. Ele recebe um objetivo, planeja uma sequência de passos, chama ferramentas, observa os resultados e itera até o objetivo ser alcançado. Quando não consegue prosseguir com segurança, escalona para um humano.

O GitHub Copilot torna a diferença concreta. O mesmo modelo no Copilot picker pode conduzir uma resposta de chat de uma vez só ou um agente de múltiplos passos. Quando o Copilot passa de uma resposta de chat para o agent mode, o modelo não muda; muda o loop ao redor dele, o estado que ele guarda e as ferramentas que ele pode chamar. No agent mode o Copilot lê seu repositório, planeja uma mudança, edita arquivos, executa o build e abre um pull request. O escalonamento faz parte do design, não é um acréscimo posterior: quando o agente

chega a uma decisão que não pode tomar com segurança, ele para e pergunta. Um agente que nunca escalona não é mais autônomo. É apenas um agente sem supervisão.

Os cinco componentes que todo agente reúne: brain, memory, tools, policy e orchestration

Todo agente, independentemente da plataforma, é construído a partir dos mesmos cinco componentes. O brain é o modelo que raciocina sobre a entrada e planeja o próximo passo, escolhido por tarefa. No GitHub Copilot o brain é o modelo que você seleciona no Copilot picker: Claude Opus 4.8 para raciocínio profundo, Claude Fable 5 ou Sonnet 5 para trabalho equilibrado, Haiku 4.5 para turnos rápidos e baratos, Gemini 3.1 Pro, ou um modelo GPT-5.x. Você escolhe por tarefa, não uma vez para o programa inteiro. A memory é o histórico de conversa, um bloco de rascunho de trabalho, um vector store para recuperação e memória episódica entre execuções. A memory é delimitada por política: um agente lembra apenas o que tem permissão para lembrar.

Tools, policy e o loop que conduz tudo

As tools são as funções que o agente pode chamar: pesquisa, consultas a banco de dados, chamadas REST, execução de código, gatilhos de workflow. Elas são catalogadas e com escopo definido pelo Model Context Protocol, para que o agente nunca converse com um endpoint aleatório. A policy é RBAC, filtros de conteúdo, limites de taxa, checkpoints de Human-in-the-Loop e trilhas de auditoria, e é inegociável. No GitHub Copilot a policy vive em .github/copilot-instructions.md e nas regras da sua organização. A orchestration é o loop plan-act-observe, a coordenação multi-agente, as retentativas e os timeouts, e é responsabilidade da plataforma. O custo acompanha tudo isso: no GitHub Copilot, o trabalho do agente é cobrado em GitHub AI Credits, em que um credit equivale a um centavo de dólar, em vigor desde 1 de junho de 2026, então o brain que você escolhe e os tokens que você gasta caem na conta. O brain raciocina, a memory persiste, as tools agem, a policy delimita, a orchestration conduz o loop.

Por que a escolha de plataforma é a primeira decisão

A pressão para colocar agentes em produção é real, e está acelerando. O Gartner projeta que 40 por cento das aplicações corporativas terão agentes de IA de tarefa específica até 2026, ante menos de 5 por cento em 2025. É um salto grande em um único ano, e é onde a maioria dos programas erra. Os times partem da ferramenta que já está aberta, em vez da decisão que ainda não tomaram. A primeira decisão não é qual modelo ou qual framework. É em qual plataforma o agente roda, porque a plataforma define o limite de tudo o que o agente pode fazer.

O portfólio de agentes da Microsoft roda sobre três plataformas, e elas são complementares, não concorrentes. O Copilot Studio é voltado a usuários de negócio e citizen developers, com autoria por arrastar e soltar e conectores nativos com o Microsoft 365. O GitHub Copilot é voltado a engenheiros de software, com agent mode, agentes personalizados e ferramentas MCP que vivem no repositório, ao lado do código que elas mudam. O Azure AI Foundry é voltado a engenheiros de ML e equipes de plataforma, com o SDK completo, o catálogo de modelos e orquestração multi-agente para agentes verticais de nível produtivo. A maioria das empresas termina rodando as três. A questão nunca é qual plataforma vence. É qual caso de uso vai para onde.

A tese do framework de decisão: a plataforma é definida por quem o agente serve e no que ele pode tocar

A tese deste playbook cabe em uma frase: a plataforma certa é definida por quem o agente serve e no que ele pode tocar. Quem ele serve é a persona. Um usuário de negócio criando um bot de helpdesk, um engenheiro refatorando um serviço e uma equipe de plataforma construindo um agente de dados regulado são três donos diferentes, com três níveis técnicos diferentes. No que ele pode tocar é o raio de impacto. Um rascunho de e-mail não é uma transferência de fundos, e um arquivo temporário não é um banco de dados de produção. Esses dois eixos, persona e raio de impacto, decidem mais do que qualquer comparação de recursos.

Leia os dois eixos juntos e a plataforma se revela. Um citizen developer automatizando um fluxo de aprovação dentro do Microsoft 365 pertence ao Copilot Studio. Um engenheiro gerando código e abrindo pull requests pertence ao GitHub Copilot, governado por .github/copilot-instructions.md e ferramentas com escopo definido por MCP. Uma equipe construindo um pipeline de dados multi-agente sob SLAs rigorosos pertence ao Azure AI Foundry, governado por Azure RBAC. Acerte a persona e o raio de impacto e tudo o que vem depois tem um lar: o modelo no Copilot picker, os checkpoints de Human-in-the-Loop, a trilha de auditoria, a conta em GitHub AI Credits. Erre os dois e nenhuma qualidade de modelo resgata o agente. A plataforma é a primeira decisão porque toda decisão posterior a herda.

> DEFINIÇÃO
>
> A tese do framework de decisão em uma frase: a plataforma certa é definida por quem o agente serve e no que ele pode tocar. A persona define o nível técnico e a superfície. O raio de impacto define os guardrails. Todo o resto, o modelo, as ferramentas, os checkpoints, o custo, herda desses dois.

Referências

Fontes primárias para a definição de agente, o portfólio de plataformas e a projeção de adoção usada neste capítulo.

- GitHub Copilot Documentation, a plataforma de referência para agent mode, agentes personalizados e o arquivo copilot-instructions usado ao longo deste playbook.
- Microsoft Copilot Studio Documentation, a plataforma low-code para usuários de negócio e citizen developers.
- Azure AI Foundry Documentation, a plataforma para orquestração multi-agente e agentes de nível produtivo.
- Gartner, task-specific AI agents in 40 percent of enterprise apps by 2026, a projeção de adoção: 40 por cento das aplicações corporativas terão agentes de IA de tarefa específica até 2026, ante menos de 5 por cento em 2025.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

01 ESCOLHA DE PLATAFORMA

Três plataformas, um framework de decisão.

O portfólio de agentes de IA da Microsoft roda sobre três plataformas, e cada uma atende a um público diferente e a uma superfície de controle diferente. O Copilot Studio é para usuários de negócio e desenvolvedores cidadãos, o GitHub Copilot é para engenheiros, e o Azure AI Foundry é para engenheiros de ML e equipes de plataforma. Elas são complementares, não concorrentes, então a decisão não é qual plataforma vence; é qual caso de uso vai para onde. A Gartner projeta que 40% dos aplicativos corporativos terão agentes de IA de tarefa específica até 2026, ante menos de 5% em 2025, o que torna essa decisão de roteamento uma que toda equipe corporativa agora precisa tomar. Este capítulo coloca as três plataformas lado a lado e dá o framework para rotear cada carga de trabalho.

Copilot Studio: para usuários de negócio e desenvolvedores cidadãos

O Copilot Studio é o ponto de entrada para quem é dono de um processo mas não escreve código. A autoria é por arrastar e soltar: você monta tópicos em um canvas visual, conecta-os a gatilhos e os testa na mesma janela. A biblioteca de conectores é a razão de ele se encaixar em ambientes Microsoft. Conectores nativos alcançam o Microsoft 365, o Dataverse e o Power Platform, então um agente pode ler uma lista do SharePoint, gravar em uma tabela do Dataverse ou iniciar um fluxo do Power Automate sem nenhum código de integração customizado. Quem constrói esses agentes são as mesmas pessoas donas do processo de negócio, o que elimina o repasse para uma equipe de engenharia.

A governança passa pelo centro de administração do Power Platform. Os administradores definem políticas de ambiente, regras de prevenção de perda de dados e limites de compartilhamento a partir de um único console, o mesmo console que já governa o Power Apps e o Power Automate no tenant. O tempo até um piloto

funcional é medido em horas a dias. A troca é teto por velocidade: o Copilot Studio é o lar certo para um assistente de RH, um bot de helpdesk de TI ou um fluxo de atendimento ao cliente em que a equipe de negócio é dona do agente, e é o lar errado para um programa que precisa de código de orquestração customizado ou de um modelo com fine-tuning.

GitHub Copilot: para engenheiros

O GitHub Copilot é a plataforma para quem escreve o código. O agent mode leva o Copilot de sugestões de uma linha para trabalho de múltiplos passos: ele planeja uma mudança, edita em vários arquivos, roda o terminal, lê a saída e itera até a tarefa passar. Você o direciona com um arquivo do repositório, .github/copilot-instructions.md, que declara as convenções, os comandos de build e as restrições que o agente deve respeitar em cada tarefa. As ferramentas do Model Context Protocol (MCP) estendem o agente para além do repositório, conectando-o a sistemas de build, rastreadores de issues e serviços internos por uma única interface padrão, em vez de uma integração sob medida por ferramenta.

A escolha de modelo fica no seletor do Copilot. A lista atual inclui Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro e a família GPT-5.x, e você troca de modelo por tarefa sem sair do editor. A cobrança roda sobre o GitHub AI Credits, em que um crédito equivale a um centavo de dólar, em vigor desde 1 de junho de 2026. Cada requisição consome créditos conforme o modelo, então a equipe orça o uso do agente como orça qualquer recurso medido, e um modelo mais pesado custa mais por tarefa do que um mais leve. A governança segue o modelo de políticas do GitHub que a organização já opera, e um piloto pode entrar em produção em dias.

Azure AI Foundry: para engenheiros de ML e equipes de plataforma

O Azure AI Foundry é para equipes que constroem agentes como produto. Ele oferece o SDK completo, um catálogo de modelos, orquestração multi-agente, avaliação e observabilidade, que é o que uma equipe de plataforma precisa para operar agentes verticais de nível produtivo em escala. O catálogo de modelos permite escolher um modelo por tarefa e trocá-lo sob controle de versão. A

orquestração multi-agente coordena agentes especializados sob um único runtime, com retentativas e timeouts tratados pela plataforma em vez de escritos à mão em cada agente.

A governança usa Azure RBAC e Azure AI Content Safety, os mesmos controles de identidade e segurança que governam o restante de um ambiente Azure. Avaliação e observabilidade são de primeira classe: a equipe define um conjunto de testes dourados, roda regressão a cada troca de modelo e rastreia cada decisão do agente para auditoria. O tempo até um piloto funcional é medido em semanas, não em dias, porque a preparação é mais pesada e a régua de revisão é mais alta. O retorno é um teto que os caminhos low-code e de IDE não alcançam, e é por isso que cargas de produção reguladas caem aqui.

A matriz de decisão, lado a lado

As três plataformas são complementares, não concorrentes, e a maioria das empresas termina utilizando as três. A questão em aberto é qual caso de uso vai para onde, e seis eixos resolvem a maior parte disso. Público: o Copilot Studio atende usuários de negócio e desenvolvedores cidadãos, o GitHub Copilot atende engenheiros, o Azure AI Foundry atende engenheiros de ML e equipes de plataforma. Nível de código: no-code e low-code, code-first na IDE, e pro-code com SDK. Conectores: Microsoft 365 e Power Platform, repositórios e MCP, e qualquer fonte via Python, REST ou MCP.

Lendo a matriz por eixo

Restam três eixos. O tempo até o piloto vai de horas a dias, a dias, a semanas, nessa ordem. A customização vai de tópicos mais conectores, a configuração de repositório e ferramentas MCP, a código completo mais fine-tuning. A governança vai do centro de administração do Power Platform, a políticas do GitHub, a Azure RBAC com Content Safety. Leia os seis eixos em conjunto e a escolha raramente fica ambígua. Combine a plataforma com quem é dono do agente e com a governança que a carga exige, e deixe o tempo até o piloto e a customização desempatarem o que restar.

DEFINIÇÃO

Studio para citizens, Copilot para engenheiros, Foundry para produção. Combine a plataforma com quem é dono do agente e com a governança que a carga exige, e deixe o tempo até o piloto e a customização desempatarem.

Referências

Fontes primárias para as três plataformas e o contexto de mercado por trás da decisão de roteamento.

- Microsoft Copilot Studio documentation, a referência para tópicos por arrastar e soltar, conectores Microsoft 365 e Power Platform, e governança pelo centro de administração.
- GitHub Copilot documentation, a referência para agent mode, custom instructions e ferramentas do Model Context Protocol.
- Azure AI Foundry documentation, a referência para o catálogo de modelos, orquestração multi-agente, avaliação e observabilidade.
- Gartner, 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, o contexto de mercado: ante menos de 5% em 2025.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

02 TIPOS DE AGENTES

Quatro tipos de agentes e onde cada um se encaixa.

A escolha de plataforma responde onde um agente roda. O tipo de agente responde qual trabalho ele faz. A maioria dos programas corporativos descobre os mesmos quatro tipos, e cada um se encaixa em uma plataforma específica por um motivo específico. Este capítulo mapeia agentes de desenvolvimento, produtividade, processos de negócio e dados para a plataforma que serve, e nomeia o guardrail que cada tipo exige antes de chegar à produção.

Quatro tipos, e por que cada um se encaixa onde se encaixa

Os quatro tipos não são uma taxonomia por si só. Eles se agrupam por quem é dono do agente, quais sistemas ele toca e quanto custa uma ação errada. Agentes de desenvolvimento servem engenheiros dentro do repositório. Agentes de produtividade servem trabalhadores do conhecimento dentro do Microsoft 365. Agentes de processos de negócio cruzam vários sistemas em nome de uma equipe de operações. Agentes de dados e analytics respondem perguntas contra um data warehouse. Uma vez que você sabe qual tipo está construindo, a escolha de plataforma do capítulo anterior praticamente se resolve sozinha.

O Gartner projeta que 40% dos aplicativos corporativos terão agentes de IA task-specific até 2026, contra menos de 5% em 2025. Task-specific é a expressão operativa. Os agentes que chegam à produção são estreitos. Fazem um tipo de trabalho, para um público, com uma superfície de controle. Os quatro tipos abaixo são as quatro formas estreitas que se repetem entre os programas. Trate-os como zonas de pouso, não como uma fronteira rígida. Um único programa costuma rodar os quatro lado a lado.

Agentes de desenvolvimento: o repositório e a IDE

Agentes de desenvolvimento vivem onde os engenheiros já trabalham: o repositório, o pull request e o editor. Correção de bugs, refatoração e geração de testes são as tarefas canônicas. Na ótica do GitHub Copilot, isso roda através do agent mode. O agente lê o .github/copilot-instructions.md para as convenções do repositório, planeja uma mudança, edita arquivos, roda o build e abre um pull request. O engenheiro permanece no loop no pull request, que é onde o code review já acontece.

O MCP é o que torna um agente de desenvolvimento útil para além do editor. Servidores Model Context Protocol conectam os sistemas que uma mudança real toca: o sistema de build, o rastreador de issues, o pipeline de deploy. Um agente que consegue ler uma verificação que falhou, abrir o ticket vinculado e correlacionar os dois escreve uma correção melhor do que um que só enxerga o diff. O modelo vem do Copilot picker. Claude Opus 4.8 ou Claude Fable 5 para as refatorações difíceis, Sonnet 5 ou Haiku 4.5 para as edições de rotina, em que latência e custo pesam mais que profundidade.

A cobrança segue o modelo

O trabalho de um agente de desenvolvimento é cobrado em GitHub AI Credits, em que um crédito equivale a um centavo de dólar desde 1 de junho de 2026. Modelos mais baratos do Copilot picker custam menos créditos por requisição. Rotear edições de rotina para o Haiku 4.5 e reservar o Opus 4.8 para as mudanças que precisam dele é uma linha na fatura, não uma questão de gosto. Por isso a escolha de modelo e o tipo de agente são decididos juntos, não isoladamente.

Agentes de produtividade: dentro do Microsoft 365

Agentes de produtividade servem trabalhadores do conhecimento onde eles já passam o dia: Teams, Outlook e SharePoint. O trabalho é notas de reunião, geração de rascunhos e busca no conteúdo que a pessoa tem permissão para ver. O Copilot Studio cria os tópicos personalizados e os conecta ao Microsoft 365, ao Dataverse e

ao Power Platform. A equipe de negócio é dona do agente, e o centro de administração do Power Platform o governa. O tempo até um piloto funcional é de horas a dias, porque o desenvolvedor cidadão nunca sai da superfície low-code.

A restrição que define esse tipo é a recuperação com escopo de identidade. Um agente de produtividade só pode expor conteúdo que o usuário solicitante já tem permissão para abrir. As notas de reunião resumem uma chamada de que o usuário participou. A busca retorna documentos que o usuário poderia ter encontrado à mão. Isso não é um recurso de conveniência. É a linha entre um assistente útil e um vazamento de dados, e é imposta na camada de plataforma, não no prompt.

Agentes de processos de negócio: através de vários sistemas

Agentes de processos de negócio executam os fluxos que mantêm uma operação em movimento: onboarding de funcionários, aprovações de compra, sinistros de seguro. Esses fluxos cruzam vários sistemas, o que os separa dos dois primeiros tipos. Um agente de sinistros lê um documento, verifica uma apólice, atualiza um registro e notifica uma pessoa, cada passo em um sistema diferente. O Copilot Studio consegue carregar a frente conversacional desse trabalho. Ele atinge seu limite quando o fluxo precisa de SLAs rigorosos, orquestração multi-agente ou um modelo com fine-tuning.

Quando precisa, o fluxo migra para o Azure AI Foundry. O Foundry dá à equipe de plataforma o SDK completo, orquestração multi-agente, harnesses de avaliação e Azure RBAC com Content Safety em cada chamada. O motivo para migrar não é ambição. É o SLA. Uma aprovação que precisa concluir dentro de um tempo limitado, com um registro de auditoria em cada passo, precisa da observabilidade e da governança que uma superfície low-code não oferece. A passagem do Copilot Studio para o Foundry é um estágio normal em um programa que amadurece, não uma reescrita.

Agentes de dados e analytics: ancorados no data warehouse

Agentes de dados e analytics transformam uma pergunta em linguagem natural em uma consulta, executam e devolvem um resumo. Traduzir um pedido em SQL, gerar um insight, sinalizar uma anomalia. O Azure AI Foundry é a plataforma, ancorado no Fabric, no Synapse ou no data warehouse que a organização já roda. A ancoragem é o ponto central. Um agente de analytics que infere um esquema em vez de lê-lo devolve números que parecem certos e estão errados.

Dois guardrails são obrigatórios para esse tipo. O primeiro é um harness de avaliação estruturado: um conjunto fixo de perguntas com respostas conhecidas que o agente precisa passar antes de cada troca de modelo, porque uma consulta que devolve números plausíveis mas incorretos é pior do que uma que falha em voz alta. O segundo é a aprovação humana antes da publicação. O agente redige o relatório. Uma pessoa o assina. O custo de um número errado em uma apresentação de board se mede em decisões, não em tokens.

> DEFINIÇÃO
>
> O padrão nos quatro tipos é o mesmo. Mais autonomia exige mais guardrails, não menos. Um agente de desenvolvimento para no pull request. Um agente de produtividade para nas permissões do próprio usuário. Um agente de processos de negócio para no SLA e no registro de auditoria. Um agente de dados para na aprovação humana antes da publicação. O guardrail não é atrito. É o que permite que o agente rode.

Referências

Fontes primárias para os quatro tipos de agentes, as plataformas onde cada um se encaixa e a projeção de mercado por trás deles.

- GitHub Copilot documentation, agent mode, instruções de repositório e MCP para agentes de desenvolvimento.

- Microsoft Copilot Studio documentation, a plataforma low-code para agentes de produtividade e de processos de negócio conversacionais.
- Azure AI Foundry documentation, a plataforma pro-code para agentes de processos de negócio e de dados sob SLAs rigorosos.
- 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, a projeção de mercado: agentes task-specific sobem de menos de 5% dos aplicativos corporativos em 2025 para 40% até 2026.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

03 ARQUITETURA DE REFERÊNCIA

A arquitetura de referência em seis camadas: a pilha de agentes enterprise.

Programas de agentes de nível produtivo são construídos sobre primitivos da plataforma, não em torno deles. A pilha tem seis camadas, da plataforma e governança na base até a superfície onde os usuários encontram o agente. Este capítulo percorre as seis camadas, enuncia os cinco princípios que regem todas elas, apresenta três padrões de orquestração e nomeia as quatro regras de design que sustentam cada camada.

Seis camadas, da plataforma e governança à superfície

A pilha se lê de baixo para cima, e cada camada depende da que está abaixo dela. A base é plataforma e governança: identidade, RBAC, Azure AI Content Safety, auditoria, observabilidade e FinOps, os mesmos controles que já regem o restante do Azure. Dentro do GitHub Copilot, o custo nesta camada é medido em GitHub AI Credits, onde um crédito equivale a um centavo de dólar, cobrado desde 1 de junho de 2026, de modo que FinOps é uma rubrica desde o primeiro piloto, não uma reflexão tardia. A segunda camada são os modelos. O catálogo do Azure AI Foundry guarda os modelos hospedados para agentes do Foundry; para o trabalho de agente conduzido a partir do GitHub Copilot, o seletor de modelos expõe Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro e a família GPT-5.x. Escolha por tarefa e faça fine-tuning apenas onde o retorno paga o pipeline. A terceira camada é o contexto: um vector store, uma camada de contexto semântico, um registro de receitas e recuperação com escopo de identidade, de modo que um agente veja apenas os dados que seu chamador tem permissão de ver.

A quarta camada são as ferramentas, expostas pelo Model Context Protocol. Cada ferramenta é catalogada, versionada e com escopo: servidores MCP, plugins nativos

e skills personalizadas, cada uma com um token que concede o acesso mais estreito de que a ferramenta precisa. A quinta camada é a orquestração: o runtime multi-agente, o planejamento, as retentativas, os checkpoints de Human-in-the-Loop e a avaliação. É a camada que transforma uma única chamada de modelo em uma sequência de passos que um revisor consegue acompanhar. A sexta camada é a superfície: Teams, Outlook, a IDE e apps personalizados. É onde os usuários encontram o agente, e é a única camada que eles veem. Uma lacuna em qualquer camada inferior aparece aqui como uma resposta errada, um registro vazado ou uma ação que ninguém aprovou.

Cinco princípios que regem cada camada

Cinco princípios valem para todas as seis camadas. Comece estreito: um caso de uso, um agente, uma equipe, e generalize apenas depois que o valor for provado, porque um agente que faz um trabalho bem vale mais do que um framework que faz dez trabalhos mal. Reutilize a governança do Azure em vez de inventar um plano de controle paralelo: a identidade, o RBAC e o Content Safety que protegem o restante do parque protegem os agentes também, e um segundo modelo de governança é uma segunda coisa a auditar. Catalogue cada ferramenta com um token com escopo e auditoria por chamada: uma ferramenta não catalogada é uma ferramenta não governada, e um token com mais acesso do que a ferramenta precisa é o raio de dano do próximo incidente.

Avalie antes de implantar: mantenha um conjunto de testes dourados e rode uma regressão a cada troca de modelo, porque mover um agente do Sonnet 5 para o Opus 4.8 muda o comportamento mesmo quando o prompt é idêntico. A avaliação roda no agent mode do GitHub Copilot contra o mesmo .github/copilot-instructions.md com que o agente é entregue, para que as condições de teste correspondam à produção. Planeje a depreciação desde o primeiro dia: versione cada prompt, mantenha o rollback disponível e trate um prompt como qualquer outro artefato implantável. Um programa de agentes que não consegue reverter uma mudança de prompt está a uma edição ruim de uma queda que não sabe explicar.

Três padrões de orquestração

A quinta camada roda um de três padrões de orquestração, e a escolha segue o formato do trabalho. O pipeline sequencial encadeia agentes em uma ordem fixa: cada agente consome a saída do agente anterior, e os repasses são os checkpoints. Serve para trabalho com etapas claras, como ler um documento, validá-lo e depois montar um resultado, em que cada etapa precisa terminar antes que a próxima comece. O padrão de fan-out e fan-in paralelo divide uma tarefa entre vários agentes que rodam ao mesmo tempo e depois combina suas saídas. Serve para trabalho que se decompõe em subtarefas independentes, como pontuar um solicitante contra várias listas de sanções, em que as subtarefas não dependem umas das outras e o passo de combinação reconcilia os resultados.

O padrão de supervisor hierárquico com workers coloca um agente supervisor no comando de um grupo de agentes worker. O supervisor planeja, delega e decide quando o objetivo foi alcançado; os workers executam tarefas estreitas e reportam de volta. Serve para trabalho aberto em que o plano não é conhecido de antemão, como uma tarefa de código que pode tocar três arquivos ou trinta. O supervisor também é o lugar natural para o checkpoint de Human-in-the-Loop: ele detém o plano, então pode pausar para aprovação antes que um worker realize uma ação irreversível. No agent mode do GitHub Copilot, esse é o padrão por trás de um coding agent que lê o repositório, rascunha uma mudança em vários arquivos e para em um gate de revisão antes de abrir um pull request.

Quatro regras de design que sustentam cada camada

Quatro regras de design mantêm a pilha sólida à medida que ela cresce. Responsabilidade única: cada agente é dono de um trabalho, e uma tarefa que precisa de três capacidades são três agentes atrás de um supervisor, não um agente com três prompts, porque um agente de responsabilidade única é um agente que você consegue testar, custear e explicar. Padrão à prova de falhas: quando um agente está incerto ou uma chamada de ferramenta falha, o padrão é parar e escalonar para um humano, nunca chutar e prosseguir. O custo de uma ação autônoma errada sobre dados regulados supera a economia de cem ações corretas, então o caminho seguro é o caminho padrão.

Auditoria imutável: cada chamada de ferramenta, escolha de modelo e decisão de Human-in-the-Loop é registrada em um registro que nenhum agente pode alterar, para que um revisor consiga reconstruir exatamente o que qualquer agente viu e fez em qualquer momento. Degradação graciosa: quando uma camada inferior está indisponível, o agente estreita seu escopo em vez de falhar por completo, de modo que um contexto que fica brevemente offline rebaixa o agente para um conjunto de ações menor e mais seguro em vez de derrubá-lo. Juntas, as quatro regras significam que uma falha em uma camada permanece contida naquela camada em vez de cascatear até a superfície.

> DEFINIÇÃO
>
> A pilha em uma frase: seis camadas, da plataforma e governança até a superfície; cinco princípios que começam estreitos e planejam a depreciação; três padrões de orquestração combinados ao formato do trabalho; e quatro regras de design, responsabilidade única, padrão à prova de falhas, auditoria imutável e degradação graciosa, que impedem que uma falha em uma camada chegue ao usuário.

Referências

Fontes primárias para as camadas de plataforma, o catálogo de modelos e a tendência de adoção enterprise.

- Azure AI Foundry documentation, o catálogo de modelos, a orquestração e a avaliação por trás das camadas de modelos e orquestração.
- Microsoft Copilot Studio documentation, a superfície low-code e a governança para agentes de propriedade do negócio.
- GitHub Copilot documentation, a plataforma de referência para agent mode, copilot-instructions.md e integração MCP.
- Gartner, 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, a tendência de adoção enterprise por trás da pilha de seis camadas: acima de menos de 5% em 2025.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

04 SUPERVISÃO HUMANA

Supervisão humana e os guardrails que escalam com a autonomia.

A autonomia não é o objetivo. Resultados confiáveis são. Um agente que pode agir sobre dados regulados, assinar documentos, movimentar dinheiro ou alterar registros de clientes precisa de checkpoints humanos explícitos, porque uma ação autônoma errada pode custar mais do que cem ações corretas economizam. Este capítulo enuncia o princípio orientador, ordena cada ação do agente em quatro quadrantes, apresenta três padrões de implementação para o checkpoint e define a base de segurança que fica por baixo deles.

O princípio orientador: mais autonomia exige mais guardrails

A regra é simples, e vai contra a intuição. Mais autonomia exige mais guardrails, não menos. Um agente que apenas sugere texto precisa de pouca supervisão. Um agente que pode agir sobre dados regulados, assinar documentos, movimentar dinheiro ou alterar registros de clientes precisa de checkpoints humanos explícitos. A razão é a assimetria. Uma ação autônoma errada pode custar mais do que cem ações corretas economizam. Um reembolso enviado para a conta errada, um contrato assinado em condições ruins, uma tabela de produção apagada: cada um desses é caro de reverter, e alguns não podem ser revertidos de forma alguma. Os guardrails existem para impedir que a ação catastrófica rara chegue à produção sem supervisão.

Esse princípio vale independentemente de qual modelo executa o agente. No GitHub Copilot em agent mode, você escolhe o modelo no Copilot picker: Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro ou um modelo GPT-5.x. Um modelo mais forte planeja melhor e age mais rápido. Ele não reduz o custo de uma ação errada. Se algo muda, um agente mais rápido amplia o raio de impacto, porque alcança o passo irreversível mais cedo. Então os guardrails escalam com a

capacidade, não contra ela. O lugar de declará-los é o .github/copilot-instructions.md, o arquivo que o agente lê em cada tarefa. Ele carrega as convenções, os limites e a regra de quando parar e perguntar a um humano.

> DEFINIÇÃO
>
> A regra que rege a supervisão humana: mais autonomia exige mais guardrails, não menos. Uma ação autônoma errada pode custar mais do que cem ações corretas economizam, então o guardrail escala com o raio de impacto, não com a confiança do modelo.

Quatro quadrantes: impacto contra reversibilidade

Nem toda ação precisa da mesma supervisão. Ordene cada ação que um agente pode tomar em dois eixos: impacto no negócio e reversibilidade. Isso gera quatro quadrantes, e cada um recebe um nível diferente de supervisão. O quadrante um é automatizar: baixo impacto, alta reversibilidade. Sugestões de mensagem de commit, resumos de rascunhos de e-mail, notas de reunião. Rodam dentro do loop do agente com log de auditoria e sem checkpoint, porque o erro é barato e fácil de desfazer. O quadrante dois é revisar: alto impacto, alta reversibilidade. Abrir um pull request, agendar uma reunião, enviar um rascunho a um cliente. Um humano aprova antes do commit ou do envio, e a ação ainda pode ser desfeita se escapar.

O quadrante três é confirmar: baixo impacto, baixa reversibilidade. Arquivar um ticket, encerrar uma conversa, excluir um arquivo temporário. O dano é pequeno, mas permanente, então uma única etapa de confirmação basta. O quadrante quatro é controle duplo: alto impacto, baixa reversibilidade. Assinar um contrato, transferir fundos, alterar permissões RBAC, excluir dados de produção, protocolar impostos. Duas pessoas aprovam, e o log de auditoria carrega evidência assinada. A maioria das ações de agentes corporativos pertence ao quadrante dois ou três, não ao quadrante um. O GitHub Copilot em agent mode encaixa diretamente nesse modelo: o agente propõe seu trabalho como um pull request, e o pull request é, ele mesmo, o checkpoint do quadrante dois, onde uma pessoa revisa o diff antes do merge.

Três padrões de implementação para o checkpoint

Três padrões cobrem a maior parte do que os quadrantes exigem. Aprovar antes é o padrão para trabalho reversível de alto impacto. O agente prepara uma ação e aguarda em um checkpoint. Nada executa até que um humano aprove, e a aprovação é registrada com a identidade do revisor. Em agent mode, isso é a revisão do pull request: o agente escreve a mudança, uma pessoa lê o diff, e o merge é a aprovação. Amostragem e revisão serve para trabalho de alto volume e baixo impacto, como etiquetagem, classificação e roteamento. O agente age continuamente, uma amostra aleatória de suas ações é auditada, e uma queda na métrica de qualidade amostrada dispara escalonamento. Você não revisa cada ação. Você revisa o suficiente para confiar na distribuição.

Controle duplo é para o quadrante irreversível. Uma pessoa inicia, uma pessoa separada aprova, e as duas nunca são a mesma identidade. Ambas as aprovações carregam assinatura criptográfica, e o par é armazenado como evidência. É obrigatório para finanças, jurídico e qualquer mudança em dados de produção. Os três padrões também moldam o custo. No GitHub Copilot, cada ação do agente é cobrada em GitHub AI Credits, em que um crédito equivale a um centavo de dólar, o modelo de cobrança em vigor desde 1º de junho de 2026. Amostragem e revisão mantém risco e gasto observáveis em alto volume, porque você acompanha os resultados amostrados e os AI Credits que eles consomem na mesma visão. As ferramentas chegam ao agente por servidores MCP, e cada servidor deve expor apenas as operações de que a tarefa precisa.

Segurança e governança: de least privilege à retenção de sete anos

Guardrails no nível da ação precisam de uma base de segurança por baixo. Least privilege vem primeiro. Um agente, e cada servidor MCP que ele chama, recebe acesso apenas aos sistemas de que a tarefa específica precisa, e nada mais. Um agente de código não precisa da API de pagamentos. Um agente de triagem não precisa de acesso de escrita à produção. A autenticação usa Azure Managed Identity, nunca uma string de conexão hardcoded em um prompt, um repositório ou um arquivo de ambiente. Managed identity significa que a credencial é emitida e

rotacionada pela plataforma, e nunca aparece em texto que o agente possa vazar. Escopo estreito e managed identity, juntos, removem as formas mais comuns de uma ação autônoma virar um incidente.

A trilha de auditoria é a outra metade. Toda ação do agente é escrita em um log imutável: qual agente, qual versão, qual ação, o input e o output como hashes em vez de dado bruto, o humano que aprovou e a duração. Dados pessoais são mascarados antes de chegar ao log. Um identificador como um CPF ou um endereço é mascarado para que o registro prove o que aconteceu sem expor com quem aconteceu. A retenção é longa. Cargas reguladas no Brasil mantêm o registro de auditoria por 2555 dias, sete anos, para casar com o prazo fiscal de retenção, guardado em Azure Immutable Storage para que ninguém possa reescrever a história depois. A política que o agente segue mora no .github/copilot-instructions.md, então o mesmo arquivo que define as convenções também codifica o limite que o agente não pode cruzar.

Referências

Fontes primárias para o modelo de supervisão, os padrões de guardrail e a base de segurança e governança.

- GitHub Copilot documentation, a plataforma de referência para agent mode, .github/copilot-instructions.md e integração de ferramentas MCP.
- Azure AI Foundry documentation, governança para agentes de produção via Azure RBAC, managed identity e Azure AI Content Safety.
- Microsoft Copilot Studio documentation, governança para agentes de propriedade do negócio pelo centro de administração do Power Platform.
- Azure, Responsible use of AI overview, orientação da Microsoft sobre supervisão humana e content safety para sistemas de IA.
- LGPD, Lei Geral de Protecao de Dados, a base legal para mascarar dados pessoais em logs de auditoria em cargas reguladas no Brasil.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

05 PARTE V, ENTREGAR E ESCALAR

Do primeiro piloto a uma frota de agentes em produção.

Um piloto prova que um agente funciona uma vez. Uma frota prova que ele funciona todo dia, sobre dados regulados, sob auditoria, em times que nunca conheceram quem construiu o primeiro. A distância entre esses dois fatos é onde a maioria dos programas de agentes trava. Este capítulo a percorre com um caso de uso trabalhado, quatro padrões verticais que reutilizam sua coreografia, e um roadmap de doze meses cujos gates de fase mantêm o programa honesto.

O caso de leasing de veículos: quatro agentes encadeados sob gates Human-in-the-Loop

Comece com um fluxo estreito o bastante para terminar e real o bastante para importar. Um cliente solicita o leasing de um veículo. Quatro agentes especializados conduzem a solicitação de ponta a ponta, cada um com uma função. O agente de intake coleta o pedido e normaliza os campos. O agente de parsing de documentos lê os arquivos enviados: comprovante de renda, identidade, endereço. O agente de validação de NF-e confere a nota fiscal eletrônica, o padrão brasileiro para documentos fiscais, contra o registro tributário que ela afirma corresponder. O agente de montagem de contrato redige o leasing a partir das entradas aprovadas. Nenhum agente é dono do fluxo inteiro. Cada um entrega um resultado tipado ao próximo, e é isso que torna a cadeia auditável.

Os gates Human-in-the-Loop ficam entre as etapas, posicionados por reversibilidade e impacto. O parsing de documentos roda na faixa de automação com log de auditoria. A validação de NF-e é um gate de revisão, porque uma correspondência tributária errada é cara e lenta de desfazer. A assinatura do contrato é controle duplo: um iniciador e um aprovador separado, nunca a mesma pessoa. Construído sobre o GitHub Copilot, cada agente é um agente personalizado com escopo definido via .github/copilot-instructions.md, com servidores MCP

conectando o repositório de documentos, a consulta tributária e o sistema de contratos. A escolha de modelo vem do seletor do Copilot a cada etapa: Haiku 4.5 para classificação barata, Opus 4.8 para o raciocínio de contrato que precisa estar certo. Cada chamada é cobrada em GitHub AI Credits, em que um crédito é um centavo de dólar, então o custo por solicitação é um número que você lê, não um chute.

Padrões verticais que reutilizam a mesma coreografia

A cadeia de leasing não é um caso isolado. A mesma coreografia, agentes especializados em sequência com gates humanos nos pontos de alto impacto, transfere para qualquer fluxo regulado e pesado em documentos. Em financeiro, a triagem de KYC e AML roda um agente de documentos que lê arquivos de onboarding, um agente de pontuação de risco que cruza nomes com listas de sanções, e um agente de escalonamento que encaminha ao compliance, com controle duplo na aprovação. Em saúde, a autorização prévia combina um agente de extração clínica com uma camada de recuperação de regras de pagadores e um checkpoint do médico antes da submissão. O tempo de autorização cai de dias para horas, e cada decisão é registrada para auditoria.

O varejo roda o mesmo formato para devoluções e logística reversa. Um agente de triagem classifica o motivo da devolução, um agente de reembolso executa a verificação de política, e um agente de logística agenda a coleta. O gate humano só é acionado em exceções acima do limite de SKU, então devoluções de rotina passam sem revisor. A manufatura aplica isso à manutenção preditiva: um agente de telemetria monitora os sinais de IoT, um agente de diagnóstico propõe ordens de serviço, e um agente de agendamento aloca os técnicos. A aprovação do gerente de planta é necessária para qualquer ordem de serviço acima de quatro horas.

A coreografia é portável; as regras de domínio não.

O que atravessa as verticais é o padrão: agentes estreitos, entregas tipadas, gates posicionados por impacto. O que não atravessa é o conteúdo de domínio, a lista de sanções, as regras de pagadores, a política de SKU, os limiares de manutenção. Mantenha esse conteúdo em arquivos de instrução versionados e ferramentas MCP,

não em prompts, e a frota de leasing de um time vira a frota de KYC do próximo em semanas, não em trimestres.

O roadmap de cinco fases e doze meses

Uma frota é construída em cinco fases ao longo de doze meses, não em um sprint heroico. Descobrir ocupa o primeiro mês: mapear casos de uso candidatos, pontuar cada um por valor versus esforço, e obter a aprovação dos stakeholders antes de alguém escrever um agente. Pilotar corre nos meses dois e três: um agente, um time, um harness de avaliação em loop fechado que mede se o agente está de fato ajudando. Endurecer preenche os meses quatro a seis: RBAC e auditoria configurados, SLA e limites de custo definidos em GitHub AI Credits, e os checkpoints Human-in-the-Loop definidos para cada ação que o agente pode executar.

Escalar corre dos meses sete a nove: um registro de ferramentas versionado, os padrões de coreografia multi-agente do caso de uso tornados reutilizáveis, e onboarding self-service para que um time novo suba a própria frota sem um engenheiro de plataforma na sala. Operar são os meses dez a doze e tudo o que vem depois: FinOps sobre o gasto de AI Credits, gestão de drift, exercícios de troca de modelo que reexecutam o harness de avaliação quando um novo modelo chega ao seletor do Copilot, e uma cadência de revisão trimestral. As fases são cumulativas. Cada uma pressupõe que a anterior foi fechada.

Critérios de saída por fase

Cada fase termina em um critério, não em uma data. Descobrir sai quando a lista curta está aprovada e o caso de uso principal tem uma métrica de sucesso mensurável. Pilotar sai quando o harness de avaliação mostra o agente atingindo essa métrica em casos reservados, não na demo. Endurecer sai quando um auditor consegue rastrear qualquer ação do agente até uma identidade e uma política. Escalar sai quando um time que você não treinou faz o próprio onboarding. Operar não tem saída; é o estado estável, e seu critério é que a revisão trimestral continue encontrando o programa saudável.

Os critérios existem porque as duas falhas mais comuns são pulos de fase. Pilotar sem endurecer produz uma demonstração: funciona na sala e cai na primeira vez que encontra dados reais e um auditor real. Escalar sem operar produz dívida: dezenas de agentes em produção sem ninguém observando o gasto, o drift ou as mudanças de modelo por baixo. Um gate de fase é uma alegação que você defende numa revisão de board. Trate-o assim e a frota chega no prazo e se mantém de pé.

> DEFINIÇÃO
>
> Pilotar sem endurecer é uma demonstração. Escalar sem operar é dívida. Cada gate de fase é uma alegação que você defende numa revisão de board, não uma data em um slide.

Referências

Fontes primárias para as escolhas de plataforma, o roadmap de produção e o contexto de mercado por trás de uma frota de agentes enterprise.

- GitHub Copilot documentation, o caminho de engenharia usado para construir os agentes do caso de uso: agent mode, agentes personalizados e MCP.
- Azure AI Foundry documentation, a plataforma de produção para as fases de endurecer, escalar e operar: catálogo de modelos, orquestração multi-agente, avaliação e observabilidade.
- Microsoft Copilot Studio documentation, o caminho low-code para agentes de propriedade do negócio nas fases de descobrir e pilotar.
- Gartner, 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026, contexto de mercado para por que a frota importa: agentes de IA específicos por tarefa projetados para 40% dos apps enterprise até 2026.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps