GESTÃO DE TOKENS E DESIGN DE AGENTES · PLAYBOOK

A cobrança por tokens recompensa a engenharia disciplinada e instrumentada.

Em 1º de junho de 2026 o GitHub Copilot aposentou os PRUs e migrou para GitHub AI Credits. Times que tratam tokens como recurso finito, aplicam context engineering e instrumentam FinOps pagam de 25 a 45 por cento menos que times que usam o Copilot como caixa-preta. A diferença não é talento. É prática.

AUTORA

Paula Silva

CARGO

Software Global Black Belt

CONTATO

paulasilva@microsoft.com

TEMPO DE LEITURA

~ 45 minutos, fim a fim

5

PARTES DO PLAYBOOK

6

CAPÍTULOS

8

PADRÕES APLICADOS

25-45%

META DE REDUÇÃO DE CUSTO

CAPÍTULOS

Chapter 00

Fundações: tokens, consumo, modelos

Como tokens são contados, para onde vão dentro de uma sessão e como o model picker do Copilot molda a conta. Sem isso, toda estimativa é chute.

Fundações →

Chapter 01

Disciplinas técnicas

Context engineering, prompt caching, design de agentes e estratégias token-eficientes. As alavancas que acumulam economia antes de mexer em orçamento.

Disciplinas →

Chapter 02

A mudança de junho de 2026

PRUs aposentados, GitHub AI Credits no lugar. O que mudou, os três caminhos para planos anuais e como migrar um time sem estouro de overage.

Mudança →

Chapter 03

Governança e FinOps

Orçamentos, alertas e atribuição por sessão, agente e repositório. A economia do Azure AI Foundry em escala enterprise.

Governança →

Chapter 04

Padrões aplicados e anti-padrões

Oito padrões aplicados de otimização e a árvore de decisão de anti-padrões. Redução de 40 a 70 por cento em cargas agênticas.

Aplicação →

Chapter 05

Apêndices: material de referência

Princípios de prompts otimizados, a referência de preços de meados de 2026, modelos de relatório FinOps e onde vive o código completo.

Referência →

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

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps

01 PARTE I, FUNDAMENTOS

Fundamentos: tokens, consumo e a escada de modelos.

O modelo nunca vê texto. Ele opera sobre sequências de inteiros, e cada inteiro tem um preço. Como texto vira token, para onde os tokens vão e qual modelo os multiplica: a fundação de todo o resto deste playbook.

O token é a unidade de trabalho, e de cobrança

Um token é a unidade atômica que um modelo de linguagem processa: um inteiro positivo que indexa uma entrada em um vocabulário fixo, tipicamente entre 32.000 e 200.000 entradas dependendo do modelo. A maioria dos LLMs modernos usa Byte Pair Encoding (BPE), um algoritmo de compressão criado por Philip Gage em 1994 e adaptado para NLP por Sennrich et al. em 2016, que descobre empiricamente unidades de subpalavra a partir do corpus de treinamento. A tokenização por subpalavras resolve três problemas de uma vez: vocabulário aberto (palavras nunca vistas se decompõem em subtokens conhecidos), compressão eficiente e multilinguismo, porque o BPE em nível de byte opera sobre os 256 bytes de UTF-8 e funciona em qualquer língua, código ou emoji.

Como cada provedor descobre o próprio vocabulário, o mesmo texto tokeniza diferente em cada modelo, mas as equivalências de bolso se mantêm. Inglês geral fica em torno de 0,75 palavra por token; o português brasileiro é cerca de 25% mais denso em tokens que o inglês; código consome mais tokens por linha que prosa; JSON estruturado é ainda mais pesado, e Unicode raro ou emojis podem custar de 2 a 4 tokens cada. Na prática, um arquivo Python de 1.000 linhas tem cerca de 12.000 tokens, um documento markdown de 2.000 palavras em português, cerca de 3.300 tokens, e uma resposta de chat de 500 palavras em inglês, cerca de 670 tokens. Com essas equivalências você estima qualquer sessão antes de executá-la; sem elas, toda estimativa de custo é chute.

Input, output e cache: três preços para três tipos de trabalho

Toda chamada a um LLM tem três categorias de tokens. O input (system prompt, ferramentas, histórico, contexto e a mensagem atual) é processado em paralelo pelo mecanismo de atenção; o trabalho é limitado por memória e é a categoria mais barata. O output é gerado token a token em modo autorregressivo, cada token novo exigindo uma passada sobre tudo o que veio antes, o que o torna tipicamente de 3 a 5 vezes mais caro que o input. A proporção não é arbitrária; é o custo computacional real da geração sequencial.

Os tokens em cache são a terceira categoria: o provedor guarda o estado interno (KV cache) de um prefixo estável e o reutiliza entre requisições. Leituras de cache custam 10% do preço de input; a escrita de cache de 5 minutos custa 25% a mais. Como system prompt, ferramentas e documentos anexados se repetem a cada chamada em cargas conversacionais e agênticas, o caching sistemático corta o custo de input em até 90% e o custo total entre 30 e 50% sem mudar mais nada. É a alavanca mais subutilizada entre times que migram para cobrança por token sem disciplina.

Um custo se esconde dentro do output: os tokens de raciocínio. Em modelos com extended thinking, o raciocínio interno gerado antes da resposta visível é cobrado como output mesmo que o usuário nunca o veja. Uma resposta visível de 500 tokens pode ter consumido 5.000 tokens invisíveis de raciocínio, custando 10 vezes mais do que aparentava; times que habilitam thinking sem essa consciência recebem contas de 5 a 10 vezes mais altas que o esperado. Habilite thinking apenas onde a qualidade extra justifica o custo.

Para onde os tokens vão: a anatomia do consumo

Tokens não são consumidos de maneira uniforme. O estudo Tokenomics (arXiv 2601.14470) mediu o consumo em desenvolvimento de software multiagente: a implementação domina com 30 a 50% do total, trabalho legítimo, enquanto o refinamento toma 15 a 30% e é o vilão habitual, iterações que um planejamento melhor, critérios de parada explícitos ou o modelo certo teriam evitado. Entender a

tarefa custa 5 a 10% quando feito uma única vez, verificação e testes tomam 15 a 25%, planejamento 5 a 15%, e a comunicação entre agentes, outros 5 a 15% onde há orquestração.

O efeito mais subestimado é o multiplicador agêntico: em uma sessão de múltiplos turnos, todo o histórico é reenviado como input a cada turno, e o input acumulado cresce quadraticamente. Sessões com mais de 10 turnos começam a sentir o efeito; passados 20 turnos, o custo é dominado pelo input acumulado e o cache vira a única alavanca prática. Quatro estratégias domam o multiplicador, em ordem de impacto: prompt caching; compactação estruturada, que comprime o histórico em um resumo e reduz tokens entre 22 e 57% sem perda de precisão; limpeza de resultados de ferramentas, que pode liberar mais de 50K tokens em sessões longas; e fan-out de subagentes, que exploram com contexto isolado e devolvem apenas resumos, cortando 70 a 90% em workflows de exploração.

> DEFINIÇÃO
>
> Sessões longas surpreendem orçamentos porque o input cresce quadraticamente com os turnos, não linearmente. Trate cada agente de longa duração como um custo que compõe: faça cache do prefixo estável, compacte o histórico, limpe resultados obsoletos e pare quando a tarefa terminar.

Modelos e multiplicadores no model picker do GitHub Copilot

A escolha de modelo é a alavanca de custo de primeira ordem: bem aplicada, reduz o custo médio entre 50 e 70% sem perda percebida de qualidade. No model picker do GitHub Copilot, só a família Claude cobre uma ordem de magnitude nos preços de lista de meados de 2026: Haiku 4.5 a US$ 1 por milhão de tokens de input e US$ 5 de output, Sonnet 5 a US$ 3 e US$ 15 (introdutório de US$ 2 e US$ 10 até agosto de 2026), Opus 4.8 a US$ 5 e US$ 25, e Claude Fable 5 a US$ 10 e US$ 50. Opus custa cerca de 1,7 vez o Sonnet; Haiku é 5 vezes mais barato que Opus. Modelos bundled vêm incluídos nos planos pagos sem consumir créditos adicionais.

A heurística tem cinco níveis. Bundled cobre cerca de 70% do trabalho diário: completions, perguntas pontuais, refatorações simples. Os baratos, classe Haiku 4.5, tomam 10 a 15%: classificação, extração, boilerplate em escala. O workhorse, Sonnet 5, toma outros 10 a 15%: debugging não trivial, refatorações multiarquivo, code review profundo, o default quando bundled não basta. Os premium, Opus 4.8 e Claude Fable 5, deveriam ser 1 a 3% do trabalho, reservados a arquitetura, debugging distribuído e decisões irreversíveis, e só depois que o workhorse se mostrou insuficiente. Long-context entra quando o problema precisa de mais de 100K tokens de contexto.

Desde a transição de 1º de junho de 2026, o GitHub Copilot cobra esse consumo em GitHub AI Credits, em que um crédito corresponde a aproximadamente um dólar de uso a preço de lista. A aritmética é direta: uma chamada com 8K de input e 1K de output no Sonnet 5 custa US$ 0,039; um time de 50 desenvolvedores fazendo 1.000 chamadas assim por dia gasta cerca de US$ 1.950 por dia, US$ 42.900 por mês em 22 dias úteis, perto de US$ 515 mil por ano. Na cobrança plana por request, todas essas chamadas custavam o mesmo; em tokens, refletem trabalho real. É por isso que estes fundamentos agora são habilidade de orçamento, não curiosidade.

Referências

Fontes primárias para tokenização, anatomia do consumo e preços de modelos.

- Neural Machine Translation of Rare Words with Subword Units, Sennrich, Haddow e Birch, ACL 2016, o artigo que adaptou o BPE para NLP e resolveu o problema do vocabulário aberto.
- Tokenomics, token consumption distribution in multi-agent SDLC systems, arXiv 2601.14470, a distribuição empírica por trás da anatomia do consumo.
- GitHub Docs, Available Copilot models, a lista autoritativa de modelos no model picker do GitHub Copilot.
- Anthropic, Models and pricing, os preços de lista da família Claude: input, output e cache.
- Anthropic Docs, Prompt caching, a mecânica de leituras e escritas de cache por trás do preço de leitura de 10%.

Paula Silva

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps

01 PARTE II, DISCIPLINAS TÉCNICAS

Disciplinas técnicas: contexto, cache, agentes e eficiência.

Quatro disciplinas separam os times que pagam o preço cheio dos que pagam uma fração pelo mesmo trabalho: engenharia de contexto, prompt caching, design de agentes e estratégias token-efficient, aplicadas sob a lente do GitHub Copilot.

Engenharia de contexto: curadoria com cirurgia

Engenharia de contexto é a disciplina de selecionar o menor conjunto de tokens de alto sinal que maximiza a probabilidade do resultado desejado. A intuição comum, de que mais contexto sempre ajuda, está errada. A curva que relaciona volume de contexto e qualidade da resposta é côncava com pico: tokens irrelevantes têm primeiro efeito neutro, depois negativo, um modo de falha chamado context rot. Times que dominam a disciplina pagam de 30 a 50% menos pelo mesmo trabalho, com qualidade igual ou melhor.

Quatro técnicas a tornam operacional. A compactação, em que o modelo comprime o próprio histórico em um resumo estruturado, entregou redução média de 22,7% de tokens no SWE-bench Lite com Claude Haiku 4.5, com 0% de perda de acurácia. A limpeza de tool results ataca o maior consumidor em fluxos agênticos: ler um arquivo de 2.000 linhas pode injetar 30.000 tokens, então mantenha apenas os últimos 5 a 10 resultados. A memória persistente tira fatos estáveis da janela, com compressão de 11x demonstrada sobre 4.182 conversas. Subagentes com fan-out e síntese central cortam fluxos de exploração em 70% ou mais. No Copilot, isso aterrissa em AGENTS.md e .github/copilot-instructions.md: um AGENTS.md com curadoria humana reduziu o tempo de execução em 28,64% e os tokens em 16,58%, enquanto um gerado por LLM teve desempenho 3% pior do que não ter arquivo.

Prompt caching: a alavanca dos 90%

O prompt caching armazena o estado de atenção calculado sobre o seu prompt para que requisições com o mesmo prefixo de tokens pulem esse trabalho. O cache opera por hash de prefixo, byte a byte: qualquer mudança em qualquer byte o invalida daquele ponto em diante. A regra é arquitetural: conteúdo estável primeiro, conteúdo dinâmico depois. System prompt, definições de ferramentas e contexto estático, como o conteúdo do AGENTS.md, ficam no topo com cache_control; o histórico da conversa e a mensagem atual ficam no fim, nunca em cache. Há mínimos: 1.024 tokens de prefixo cacheável em modelos classe Sonnet e Opus e 2.048 no Haiku 4.5.

A economia é simples. Uma leitura de cache custa 10% do input regular. Uma escrita de cache com o TTL padrão de 5 minutos custa 25% a mais, e cada requisição reativa o TTL enquanto a sessão está ativa. Em cargas conversacionais, com hit rate típico de 70 a 85% após o primeiro turno, é possível economizar até 90% do custo de input sem mudar uma única linha de prompt. O GitHub Copilot no VS Code gerencia esse cache automaticamente; você o ajuda mantendo o .github/copilot-instructions.md e o AGENTS.md estáveis, alterados em pull requests e não de forma ad hoc.

Anti-padrões que quebram o cache

Cinco erros recorrentes destroem o benefício. Um timestamp dentro do prefixo em cache muda o hash a cada requisição, então você paga o prêmio de 25% de escrita toda vez e nunca lê. cache_control em um bloco que muda, como a mensagem do usuário, faz o mesmo. Reordenar blocos entre sessões produz um miss completo. Variação cosmética, um espaço duplo ou uma palavra trocada em um prompt montado inline, invalida tudo silenciosamente; mantenha system prompts em arquivos versionados. E um prefixo abaixo do tamanho mínimo é ignorado em silêncio.

DEFINIÇÃO

A leitura de cache custa 10% do input regular; a escrita de cache de 5 minutos custa 25% a mais. Com prefixo estável, até 90% do custo de input desaparece sem tocar em uma linha de prompt. O pré-requisito é disciplina: estável primeiro, dinâmico depois, nada volátil dentro do prefixo em cache.

Design de agentes: o padrão é workflow

A taxonomia da Anthropic separa workflows, em que o código decide a sequência e o LLM faz partes do trabalho, de agentes, em que o modelo dirige o próprio processo e o uso de ferramentas. Um workflow tem custo previsível; um agente pode custar de US$ 0,30 a mais de US$ 5 na mesma tarefa. Cerca de 70% dos casos empresariais são resolvidos por workflows a uma fração do custo, então o padrão é workflow, e a agência precisa ser justificada. Cinco padrões componíveis cobrem o espaço: prompt chaining, routing, paralelização, orchestrator-workers e evaluator-optimizer. Reserve agentes de verdade, em termos de Copilot o agent mode e o Coding Agent, para espaços de solução amplos em que a sequência varia genuinamente por problema.

Quando a agência se justifica, restrinja-a. Agentes customizados definidos em arquivos .agents/ devem seguir responsabilidade única, ser read-only por padrão, carregar allowlists explícitas de ferramentas e declarar orçamentos como max_iterations e max_tokens_per_session. Aplique o mesmo tratamento ao GitHub Coding Agent na nuvem: limite tokens e iterações por tarefa, planeje com um modelo mais forte e execute com um mais barato, e exija aprovação humana em todo PR. Servidores MCP estão entre as maiores fontes de overhead de tokens; uma definição enxuta de cerca de 80 tokens vence uma verbosa de 500. Hooks em torno das chamadas de ferramentas, com cache de resultados repetidos e truncamento de saídas grandes, podem reduzir o custo de fluxos agênticos em 30 a 50%.

Estratégias token-efficient e o model picker

Menos é mais. Cortar 39% do corpo de uma skill e 48% das descrições melhorou a qualidade funcional em 2,8%: prompts compactos concentram o sinal que a atenção de fato alcança. O progressive disclosure aplica a mesma ideia estruturalmente: um núcleo pequeno sempre carregado, com checklists detalhados e exemplos buscados sob demanda. O design enxuto de ferramentas completa o conjunto: nomes curtos, descrições de uma linha, poucos parâmetros, resultados por intervalo em vez de dumps de arquivo inteiro, JSON denso em vez de prosa. Todo loop evaluator-optimizer precisa de critérios de parada explícitos: máximo de iterações, limiar de score, delta mínimo de melhoria e orçamento de tokens.

A última disciplina é o roteamento por complexidade, e no Copilot ela vive no model picker. Usar modelo premium por padrão custa em média de 3 a 5x a mais; uma etapa de classificação que custa cerca de US$ 0,00005 por chamada reduz o custo médio em 50 a 70% sem perda percebida de qualidade. Mapeie os níveis para os preços de meados de 2026: trabalho de rotina para o Haiku 4.5, a US$ 1 de input e US$ 5 de output por milhão de tokens, trabalho complexo para o Sonnet 5, a US$ 3 e US$ 15, trabalho de especialista para o Opus 4.8, a US$ 5 e US$ 25, e apenas raciocínio de fronteira para o Fable 5, a US$ 10 e US$ 50. Cada escolha premium consome AI Credits com seu próprio multiplicador, então o picker é um instrumento de orçamento, não uma preferência. Codifique a política em custom prompts em .github/prompts/, com modelo explícito e padrões applyTo.

Paula Silva

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps

01 PARTE III, A MUDANÇA

A mudança de junho de 2026: de premium requests para GitHub AI Credits.

Os PRUs foram aposentados, a cobrança por uso está ativa e um dólar em créditos compra um dólar de trabalho a preço de lista. O que mudou, o que decidir e como conduzir um time pela transição.

O que mudou em 1º de junho de 2026

Em 1º de junho de 2026, o GitHub Copilot aposentou os Premium Request Units, a unidade de cobrança por requisição dos modelos premium, e migrou todos os planos pagos para GitHub AI Credits, cobrança baseada em uso em que um dólar em créditos corresponde a cerca de um dólar de uso ao preço de lista da API de cada modelo. Um AI Credit equivale a US$ 0,01, e cada chamada a um modelo premium consome créditos na proporção dos tokens de input, output e cache processados. A mudança estrutural cabe em uma frase: sob PRUs, o custo era por intenção; sob tokens, é por trabalho realizado. Mario Rodriguez, CPO do GitHub, apresentou a causa como arquitetural, não comercial: uma pergunta rápida e uma sessão autônoma de horas custavam o mesmo, e o subsídio implícito das sessões agênticas longas deixou de ser sustentável.

O preço base dos planos não mudou. O Copilot Pro segue em US$ 10 por mês com 1.000 AI Credits incluídos, o Pro+ em US$ 39 com 3.900, o Business em US$ 19 por usuário com 1.900 e o Enterprise em US$ 39 por usuário com 3.900. Code completions inline, Next Edit Suggestions e o Copilot Chat com modelos bundled como GPT-4.1 e GPT-5 mini continuam incluídos e não consomem créditos. Duas mudanças merecem atenção especial. Primeira, o fallback automático acabou: com créditos zerados e a política de paid usage desabilitada, os modelos premium param de funcionar, o que transforma orçamento em controle de continuidade, não apenas de custo. Segunda, o Copilot Code Review agora cobra em duas dimensões, AI

Credits pelo modelo e minutos de GitHub Actions pelo runner. Os créditos são zerados no dia 1º de cada mês às 00:00 UTC e não acumulam: é usar ou perder.

> DEFINIÇÃO
>
> A âncora primária deste capítulo: em 1º de junho de 2026, o GitHub Copilot aposentou os PRUs e migrou os planos pagos para GitHub AI Credits, mudança anunciada no blog oficial do GitHub como GitHub Copilot is moving to usage-based billing. Um dólar em créditos compra aproximadamente um dólar de uso a preço de lista. Todo o resto deriva desse único fato.

Três caminhos para assinantes anuais de Pro e Pro+

Quem assinou Pro ou Pro+ anual antes de 1º de junho de 2026 recebeu uma transição diferenciada: permanece em PRUs até o vencimento do plano, mas com multiplicadores bem maiores. O Claude Opus 4.7 foi de 7,5x para 27x e o Claude Sonnet 4.6 de 1,25x para 4,5x, ambos um aumento de 3,6x, enquanto GPT-5.3 Codex e Gemini 3.1 Pro foram de 1x para 6x. A capacidade real cai na mesma proporção: 100 sessões com Opus por mês viram cerca de 28, e 100 com Codex, cerca de 17. O GitHub ofereceu três caminhos, cada um com um prazo que importava em maio.

Caminho A, manter o plano anual até o vencimento

Nenhuma ação necessária e o preço anual permanece como pago, mas a capacidade premium real cai de 3,6x a 6x e não há visibilidade granular de custo, porque a cobrança segue em PRUs. Faz sentido para quem trabalha quase sempre em modelos bundled como GPT-4.1 e GPT-5 mini, usa premium só ocasionalmente ou tem plano anual vencendo em menos de três meses.

Caminho B, converter para mensal

O GitHub oferece créditos proporcionais ao tempo restante do plano anual. Um assinante de Pro+ anual que pagou US$ 390 e tem quatro meses restantes recebe US$ 130 como 13.000 AI Credits de bônus e entra em junho com o Pro+ mensal ativo

mais o bônus, 16.900 créditos no total; de julho em diante, valem os 3.900 créditos mensais padrão. O prazo de conversão era 25 de maio. É o caminho certo para power users que dependem de premium com frequência e querem visibilidade em nível de token desde já.

Caminho C, cancelar com reembolso proporcional

Cancelar antes de 20 de maio acionava reembolso proporcional ao tempo não utilizado, US$ 130 no mesmo exemplo de quatro meses restantes. Recupera valor e deixa o usuário livre para escolher outro provedor, ao custo de perder o acesso ao Copilot, a menos que reassine no mensal. Serve para quem sai do ecossistema Copilot ou perde o acesso por troca de empresa. O mapa por persona é direto: o dev que vive no bundled fica com o Caminho A, o power user de premium com o B e quem está de saída com o C.

Migrando um time: a janela de 16 semanas

Para organizações, a transição não é um evento de um dia, e sim um processo de cerca de 16 semanas, do bill preview que o GitHub disponibilizou no início de maio de 2026 até o custo estável em setembro. Maio foi descoberta e preparação: analisar o bill preview no Billing Overview, identificar os maiores consumidores, decidir o destino dos planos anuais antes dos prazos de 20 e 25 de maio, ativar a política de premium request paid usage com teto de 20% a 30% acima do consumo histórico, configurar alertas em 50%, 75%, 90% e 100% do orçamento, designar um dono explícito de FinOps e treinar os tech leads antes de 1º de junho. Essa política é a configuração mais importante: habilitada, o trabalho continua além dos créditos incluídos como overage controlado; sem ela, os modelos premium param quando os créditos zeram.

Junho é o D-Day mais a estabilização: monitorar o consumo diariamente na primeira semana, validar os alertas e aplicar os quick wins, roteamento de modelos por complexidade e prompt caching nas cargas mais pesadas. Julho é otimização sistemática, com showback semanal aos tech leads, prompts customizados e observabilidade enterprise. Agosto é maturação, e setembro entrega o primeiro mês completo de cobrança realista. O bill preview diz a cada organização o quão urgente isso é. Projeção igual ou menor que a fatura atual em PRUs significa que a disciplina

já existe. Aumento de 30% a 100% sinaliza premium dentro de workflows agênticos longos e pede roteamento e caching imediatos. Acima de 100%, em geral é Opus em agent mode sem disciplina, e exige correção antes que vire risco operacional sério. Times que aplicaram essa disciplina capturaram redução de 25% a 45% frente ao uso sem disciplina. Quem agiu cedo capturou economia; quem esperou pagou overage.

Referências

Fontes para a mudança de junho de 2026, os GitHub AI Credits e os caminhos de migração.

- GitHub Copilot is moving to usage-based billing, GitHub Blog, abril de 2026, o anúncio oficial que aposentou os PRUs e introduziu os GitHub AI Credits.
- About billing for GitHub Copilot, GitHub Docs, preços dos planos, AI Credits incluídos e a política de paid usage.
- GitHub Copilot documentation, GitHub Docs, tabelas de preço por modelo e multiplicadores dos planos anuais, sempre confira os valores atualizados.
- FinOps Foundation, o framework por trás do cronograma de migração, do showback e da posse de orçamento neste capítulo.

Paula Silva

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps

01 PARTE IV

Governança e FinOps: opere o gasto com IA como um recurso de nuvem.

Orçamentos, alertas e atribuição de custo por sessão, agente e repositório no GitHub Copilot, e a economia do Azure AI Foundry que torna a governança viável em escala enterprise.

Inform, Optimize, Operate: o ciclo FinOps aplicado ao Copilot

Sem dados, otimização é chute. A FinOps Foundation define três fases iterativas. Inform é visibilidade granular: quem usa IA, quanto, com qual modelo, em qual feature. A stack Microsoft é concreta: o Copilot emite telemetria via OpenTelemetry para o Application Insights, o Azure Cost Management exporta dados no padrão FOCUS, o Microsoft FinOps Hub processa tudo, e Power BI ou Grafana visualizam. Optimize transforma visibilidade em oportunidades com payback estimável: prompts ineficientes, modelos mal escolhidos, falta de cache, times sem treinamento. Operate institucionaliza o ciclo: reuniões recorrentes, dashboards de KPI vivos, ajustes de orçamento baseados em dados.

Maturidade se mede em meses, não em dias. O mês 1 estabelece a linha de base: quanto consumimos hoje, e onde. Os meses 2 e 3 entregam as primeiras economias com quick wins. Os meses 4 a 6 institucionalizam a melhoria contínua. Do mês 6 em diante, o showback evolui para chargeback e as decisões passam a ser orientadas por dados.

Cinco métricas para instrumentar desde o primeiro dia

Custo por 1K tokens diz se o preço médio é justo para o seu mix; a meta é tendência de queda: mais cache, já que leituras de cache custam 10% do preço de input, roteamento para bundled e context engineering. Custo por feature diz se a capacidade devolve o ROI. Custo por usuário expõe outliers: os 10% do topo consomem tipicamente 40 a 50% do gasto, e o 1% do topo sozinho pode

consumir 10 a 15%. A taxa de acerto de cache deve passar de 60% em workloads conversacionais. A razão output/input é saudável entre 0,1 e 0,3; acima de 0,5 é sinal vermelho.

Orçamentos, alertas e atribuição por sessão, agente e repositório

São quatro níveis de controle, do mais suave ao mais invasivo. Alertas notificam em 50, 75 e 90% do orçamento mensal; são sinal para reagir, não parada. Caps interrompem os modelos premium a 100% enquanto os bundled seguem: com um cap mensal de US$ 5.000 na política de premium requests do GitHub Copilot, os desenvolvedores perdem Sonnet, Opus e Gemini Pro até o mês seguinte, mas mantêm GPT-4.1 e GPT-5 mini. Limites rígidos param tudo e devem ser raros: orçamentos fixos de POC ou contratos com cliente. Detecção de anomalias é o nível proativo: compara-se o consumo com um histórico móvel de 30 dias e sinaliza-se qualquer dia acima de 3 sigma. Um drill-down típico vai do dia anômalo aos maiores usuários, depois às sessões de um usuário, e encontra um Coding Agent que rodou 50 vezes em loop por um bug.

Nada disso funciona se cada AI Credit não for atribuível. Cada requisição deve carregar modelo, modo (ask, edit, agent ou plan), tokens por tipo, custo em USD, AI Credits e identificadores de usuário, time, feature, sessão e organização ou repositório. Essa atribuição alimenta o showback, que mostra a cada time o custo gerado sem cobrar, e o chargeback, que debita de fato o centro de custo. Sequencie: tech leads nos meses 1 a 3, engineering managers nos meses 4 a 6, público entre times depois do mês 6. Migre para chargeback só com 6 ou mais meses de showback confiável e tagging consistente, em transição gradual: 25, 50, 75 e 100% em quatro meses.

DEFINIÇÃO

O comportamento muda quando o número é visto. Showback cria visibilidade sem atrito financeiro; chargeback transforma a escolha de modelo em decisão econômica. Menos de 3 meses de showback, mais de 10% das chamadas sem tag ou times ainda resistentes significam chargeback prematuro, seja qual for a pressão do financeiro.

A economia do Azure AI Foundry em escala enterprise

O GitHub Copilot resolve a codificação na IDE; o Azure AI Foundry resolve workloads de produção e governança regulatória. É um complemento, não uma alternativa, e a escolha é por workload, não por organização. O Foundry traz um catálogo com mais de 1.800 modelos de Anthropic, OpenAI, Meta, Mistral e outros, autenticação com Microsoft Entra ID, RBAC embutido, logs de auditoria no Azure Monitor, isolamento de rede com private endpoints e certificações como HIPAA, SOC 2 Type II, ISO 27001, FedRAMP High, GDPR e LGPD. Codificação fica no Copilot, workloads regulados e RAG sensível vão para o Foundry, POCs rápidas usam a API direta.

PTU versus PAYG: a decisão de cobrança com maior impacto

PAYG cobra por token, sem mínimo nem compromisso, para carga variável ou com picos. PTU, Provisioned Throughput Units, reserva capacidade: paga-se pela capacidade, não por token, com throughput garantido e latência consistente. A regra: abaixo de US$ 5 mil por mês em um modelo, PAYG sempre; entre US$ 5 mil e US$ 30 mil, depende da latência; acima de US$ 30 mil, avalie PTU a sério; acima de US$ 100 mil, PTU quase sempre vence. A escala muda tudo: 50 milhões de tokens por mês em GPT-5 custam cerca de US$ 500 por mês em PAYG. Com 5 bilhões de tokens por mês, a conta PAYG chega a US$ 50 mil por mês, e PTUs reservadas com 30% de desconto anual saem 75% mais baratas.

Governança tem preço, e ele é pequeno

Os filtros do Azure AI Content Safety não são gratuitos, mas são baratos em relação ao risco: moderação de texto a US$ 0,40 por 1.000 chamadas, prompt shields a US$ 0,30, protected materials a US$ 0,60 e groundedness a US$ 0,75. Para 100 mil chamadas por mês com todos os filtros, são cerca de US$ 200 por mês, um erro de arredondamento perto de um incidente de compliance. Vigie o logging diagnóstico de prompts e completions completos, que pode sair caro; defina retenção. O export FOCUS coloca a cobrança do GitHub Copilot e o consumo do Foundry nos mesmos dados do Azure Cost Management, e o CFO lê um dashboard, não dois.

Referências

Fontes para o ciclo FinOps, os níveis de controle e a economia do Foundry.

- FinOps Framework, FinOps Foundation, as fases Inform, Optimize, Operate.
- FOCUS Specification, FinOps Foundation, o schema aberto de custo e uso por trás do export unificado.
- Microsoft FinOps Toolkit, Microsoft, o FinOps Hub open source com templates de Power BI e queries.
- Azure Cost Management Documentation, Microsoft, orçamentos, alertas e exports FOCUS.
- Azure AI Foundry Documentation, Microsoft, catálogo de modelos, content safety e governança enterprise.
- Provisioned Throughput Concepts, Microsoft Learn, dimensionamento de PTU e modelo de preço.

Paula Silva

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps

01 PARTE V, APLICAÇÃO PRÁTICA

Padrões aplicados e antipadrões.

Oito padrões implementáveis amanhã, os antipadrões que desfazem a economia e as árvores de decisão que tornam a disciplina de tokens um reflexo.

Oito padrões, um objetivo

Teoria sem implementação não economiza tokens. Esta parte do playbook aterrissa os capítulos anteriores em oito padrões concretos para o GitHub Copilot e as cargas agênticas ao redor dele. Cada padrão resolve um problema de custo específico e eles se compõem: o roteamento decide qual modelo responde, o cache decide quanto do input é cobrado de novo, o design de agentes decide quanto contexto cada etapa carrega, e o design de ferramentas decide o overhead fixo de cada turno.

Roteie por complexidade, planeje com o modelo premium

O padrão 1 é o roteamento dinâmico. Times definem o premium como padrão no Copilot e pagam preço premium por perguntas que um modelo incluído no plano resolveria. A solução é um classificador em dois estágios: heurísticas estáticas por palavra-chave, sem custo, e um modelo barato como fallback dinâmico quando elas não decidem. Impacto: redução de 40-70% no custo médio sem perda percebida de qualidade. O padrão 4 é plan e execute: planejar a tarefa grande com o modelo premium e executar com o modelo de trabalho. Segundo a fonte, isso sai 64% mais barato do que rodar tudo no premium, com a qualidade de plano do premium.

Cache em camadas, exato e semântico

O padrão 2 é o cache hierárquico em três camadas: cache exato indexado por hash, cache semântico por similaridade de embeddings e o prompt cache do provedor por baixo. Caches exatos falham com variação cosmética; a camada semântica captura perguntas que significam a mesma coisa. A taxa de acerto combinada chega a 60-

90%, cortando 50-80% do custo de input. O padrão 5 é o cache semântico standalone: em FAQs e perguntas recorrentes, adiciona 20-40% de acerto sobre um cache exato.

Isole contexto, especifique antes de gerar

O padrão 3 é o fan-out de subagentes. Um agente líder explorando uma base de código grande acumula 800K+ tokens de contexto. A saída: subagentes com contexto isolado e ferramentas somente leitura, cada um retornando um resumo de no máximo 2K tokens que o líder sintetiza. Impacto: redução de 70-90% em fluxos de exploração. O padrão 8 é o agente customizado com Spec-Kit: um agente do Copilot em .agents/ que lê uma CONSTITUTION, uma SPECIFICATION em notação EARS e um IMPLEMENTATION_PLAN antes de escrever código. Estrutura substitui retrabalho: 40-60% menos iterações de refinamento.

Componha ferramentas, mantenha o MCP enxuto

O padrão 6 é a composição de ferramentas. Dez ferramentas granulares com cerca de 200 tokens de definição cada impõem 2.000 tokens de overhead em todo turno; três ferramentas compostas com defaults sensatos cobrem o mesmo espaço e cortam 60-70% do overhead de system prompt. O padrão 7 estende a ideia aos servidores MCP aos quais o Copilot se conecta: nomes curtos, descrições de 5-10 palavras, 2-4 parâmetros essenciais, resultados paginados e recursos com lazy-load. Um servidor enxuto cabe inteiro em cerca de 250 tokens de definição, contra 1000+ nos servidores verbosos.

> DEFINIÇÃO
>
> O número a guardar: combinados em cargas agênticas enterprise, esses oito padrões entregam redução de 40-70% no custo total. Não há bala de prata: roteamento, cache, isolamento de contexto e disciplina de ferramentas se acumulam em cada requisição que o Copilot faz.

Os antipadrões que desfazem a economia

O catálogo lista dez antipadrões recorrentes de pilotos enterprise. O cemitério de agentes: agentes criados como prova de conceito, nunca aposentados, ainda consumindo tokens. Tudo é agente: fluxos determinísticos implementados como agentes por flexibilidade futura, quando workflow vence em 70% dos casos e a conversão de volta tipicamente economiza 50-80%. Premium como padrão: o viés de que premium é sempre melhor. O system prompt monolítico: 30K tokens com todas as convenções da empresa, o que também quebra o cache; a correção: progressive disclosure e um AGENTS.md curado com menos de 500 linhas. E o estouro de tool results: sessões acumulando 600K tokens de leituras brutas porque as ferramentas retornam arquivos inteiros.

A segunda metade: ausência de critérios de parada, em vez de combinar máximo de iterações, meta de score, delta de melhoria e orçamento de tokens. O cache que nunca cacheia, acerto abaixo de 10% porque um timestamp ou um bloco reordenado quebra o prefixo estável. Mau uso de long context, confundindo o que cabe com o que vale a pena, quando uma chamada de 1M de tokens custa $10. Multiagente sem coordenação, onde a comunicação entre agentes consome 30% dos tokens. E ausência de instrumentação: o custo real vira surpresa no fim do mês. O GitHub Copilot adiciona as próprias variantes: modo Agent para perguntas simples que o Ask com modelo incluído responde por 5-10x menos, workspace settings sem versionar que fazem o custo variar 3-5x entre times, e prompts presos em snippets locais em vez de .github/prompts/.

A matriz de modos no VS Code

O antídoto prático é uma matriz de modos: Ask com modelo incluído para perguntas, Edit para mudanças locais como docstrings e renomeações, Agent com o modelo de trabalho para features completas e migrações multiarquivo, Plan com o modelo premium para arquitetura e tradeoffs irreversíveis. Planejar faz poucas chamadas e custa pouco mesmo no premium; executar faz muitas, e por isso merece o modelo de trabalho.

Três frameworks de decisão e o rollout

Framework 1, qual modelo: modelos incluídos no plano para autocomplete e perguntas curtas; long context apenas quando o contexto excede 100K tokens; o tier barato para classificação e extração; o modelo de trabalho como padrão para refatoração e debugging; o premium apenas quando o de trabalho falhou ou o problema é genuinamente complexo. Regra de bolso: o modelo de trabalho cobre 70% dos casos não incluídos no plano, e o premium fica para os 5-10% genuinamente difíceis. Framework 2, workflow ou agente: se a sequência de passos é conhecida e estável, use workflow; adicione agência apenas quando a decomposição realmente varia por input.

O framework 3 é a ordem de otimização por ROI: roteamento por complexidade primeiro, depois prompt caching, fan-out de subagentes, limpeza de tool results, design enxuto de ferramentas, auditoria de system prompt, critérios de parada e compactação estruturada por último. Depois de 8 semanas dessa sequência, a fonte relata redução típica de 40-70% sobre a linha de base. O rollout roda em quatro fases: descoberta para medir o estado atual; quick wins como o modelo de trabalho padrão no workspace e tetos de gasto com alertas, valendo redução esperada de 25-40%; disciplinas técnicas para 10-20% adicionais; e operação FinOps contínua com dashboards e detecção de anomalias.

Referências

Fontes dos padrões, dos antipadrões e dos frameworks de decisão.

- Building effective agents, Anthropic, a base do Framework 2.
- Effective context engineering, Anthropic, a lógica do fan-out de subagentes.
- Spec-Kit, Spec-Driven Development, GitHub, a estrutura do padrão 8.
- Model Context Protocol Specification, o protocolo do servidor MCP enxuto do padrão 7.
- GitHub Docs, Best practices for Copilot at scale, a base operacional das fases de rollout.

Paula Silva

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps

01 APÊNDICES

Apêndices, o material de referência.

Quatro apêndices sustentam este playbook: prompts otimizados, um cartão de referência de preços, templates de relatório FinOps e o código completo. Este capítulo condensa cada um deles no que você precisa na hora do uso.

Apêndice A, prompts otimizados

O Apêndice A reúne os templates de prompt testados nos pilotos da América Latina em 2025 e 2026. Os princípios de design são compactos e repetíveis: declare o papel em uma linha, enumere severidades ou passos em vez de descrevê-los, fixe o formato de saída com um schema JSON ou uma instrução como apenas a docstring, e delegue as convenções do projeto ao AGENTS.md em vez de repeti-las em cada prompt. Um revisor de código escrito assim cabe em cerca de 50 tokens e substitui as versões verbosas de mais de 500 tokens que os times escrevem por instinto, sem perda de qualidade de revisão.

O mesmo apêndice traz definições enxutas de ferramentas para read, search e run_tests, além de custom prompts para .github/prompts/ cujo frontmatter fixa o modelo no model picker do GitHub Copilot por tarefa: revisão profunda de código em um modelo workhorse, design de arquitetura em um modelo premium com extended thinking, explicação de código em um modelo bundled que não consome AI Credits. Completam o conjunto templates de hooks preToolUse e postToolUse, que adicionam verificação de cache, trava de orçamento, validação de caminhos, redação de segredos e truncamento de resultados, e um template de AGENTS.md que diz aos agentes onde as coisas ficam e o que não tocar.

Apêndice B, cartão de referência de preços

O cartão de referência usa os preços de lista da Anthropic de meados de 2026, por milhão de tokens, os mesmos por trás das opções do model picker do Copilot.

Claude Fable 5 custa $10 de input e $50 de output. Opus 4.5 a 4.8 custam $5 de input e $25 de output. Sonnet 5 custa $3 e $15, com preço introdutório de $2 e $10 até agosto de 2026. Sonnet 4.6 custa $3 e $15, e Haiku 4.5 custa $1 e $5. Duas regras de cache completam o cartão: a leitura em cache custa 10% do preço de input, e a escrita de cache de 5 minutos custa o preço de input mais 25%. Para o Sonnet 4.6, isso significa $0.30 por milhão de tokens lidos do cache e $3.75 por milhão escritos.

No lado do consumo, o conversor do apêndice trata um AI Credit como aproximadamente um centavo de dólar de preço de lista. A capacidade por plano é a seguinte: Copilot Pro, a $10 por mês, inclui 1.000 AI Credits; Pro+, a $39, inclui 3.900; Business inclui 1.900 por usuário e Enterprise, 3.900 por usuário. Planos anuais ainda medidos em PRUs convertem com multiplicadores entre 3.6x e 6x, então sempre confira a tabela exata em docs.github.com antes de fechar um orçamento.

> DEFINIÇÃO
>
> Se uma cópia do apêndice de preços ainda listar o Opus a $15 de input e $75 de output, ela é anterior ao reajuste de 2026: a lista atual de Opus 4.5 a 4.8 é $5 e $25. Preços mudam; as regras de cache de 10% para leitura e mais 25% para escrita de 5 minutos têm se mantido estáveis. Verifique em docs.github.com e anthropic.com/pricing antes de publicar qualquer estimativa.

Apêndice C, templates de relatório FinOps

O Apêndice C fornece duas cadências de relatório. O relatório semanal operacional, para tech leads toda segunda-feira, acompanha o gasto total contra a semana anterior, o custo por mil tokens contra a meta abaixo de $0.005, o cache hit rate contra a meta acima de 60%, os 10 maiores consumidores com custo médio por sessão, as anomalias detectadas com suas notas de investigação e os alertas de orçamento disparados. O relatório mensal executivo, para o CFO e o VP de Engenharia, adiciona custo por desenvolvedor, utilização do orçamento, distribuição por centro de custo e por modelo, as vitórias de otimização do mês, áreas de atenção como usuários outliers e o ROI do próprio playbook.

Os templates vêm com o ferramental para alimentá-los: três dashboards de Power BI (Executive Overview, Tech Lead Operational e FinOps Detail), seis queries KQL para Application Insights cobrindo custo por time, sessões descontroladas acima de $5, cache hit rate por feature, anomalias diárias além de três desvios-padrão, distribuição de modelos por usuário e sinais de alerta na razão output/input, e um JSON de dashboard do Grafana pronto para importar, com custo total, tendência de consumo, um medidor de cache hit e uma tabela de top usuários.

Apêndice D, onde está o código completo

O Apêndice D centraliza a implementação referenciada ao longo do playbook. O model router de produção classifica cada requisição em cinco tiers, bundled, cheap, workhorse, premium e long context, tentando primeiro uma passada estática por palavras-chave, que é gratuita, e recorrendo a um classificador dinâmico no Haiku 4.5, que custa uma fração de centavo por chamada; ele carrega uma cadeia de fallback e emite cada decisão de roteamento para a telemetria. O cache hierárquico empilha três camadas: correspondência exata no Redis, correspondência semântica com limiar de similaridade de 0.93 e o cache do próprio provedor.

Dois ativos a mais fecham o apêndice: um setup de OpenTelemetry com contadores de tokens, custo e chamadas dimensionados por modelo, usuário, feature e time, alimentando as queries KQL e os dashboards do Apêndice C, e um template de servidor MCP enxuto cujas três definições de ferramenta somam cerca de 250 tokens. Um template de agente customizado mostra os guardrails em forma de arquivo: modelo fixado, lista explícita de ferramentas, context_files, max_iterations de 25, orçamento de sessão de 500.000 tokens e hooks pre e post tool. Os quatro apêndices acompanham este playbook como arquivos markdown, de A_appendix_optimized_prompts.md a D_appendix_complete_code.md, ao lado dos capítulos; copie-os para o seu repositório e adapte ao seu stack.

Paula Silva

SOFTWARE GLOBAL BLACK BELT

paulasilva@microsoft.com

v1.0.0 · 2026-06-17

Building the future of software development with AI and Agentic DevOps