ECONOMIA DE TOKENS · GITHUB COPILOT

Extraia mais de cada token.
Otimização de tokens e qualidade de agentes na era dos AI Credits.

Um briefing executivo e técnico sobre o usage-based billing, o que ele muda na sua organização de engenharia e como operar bem nele.

AUTORA
Paula Silva

CARGO
Software Global Black Belt

DURAÇÃO
60 a 90 minutos

DATA
2026-06-14

AGENDA

Sete partes, um só quadro.

PART I	A mudança, AI Credits substituem a cobrança por request em 1 de junho de 2026
PART II	Fundamentos, como LLMs, agentes, tokens e janelas de contexto funcionam de verdade
PART III	Três superfícies, VS Code, GitHub Copilot CLI e GitHub.com, uma só carteira
PART IV	Qualidade de agente primeiro, por que qualidade supera contar tokens
PART V	As seis alavancas de otimização, e como elas se acumulam
PART VI	Aplicar e governar, engenharia de contexto, orçamentos, observabilidade, adoção
PART VII	Anexo, cache, o prompt cache do GitHub Copilot versus Foundry mais Redis

PARTE

I

A mudança.

Em 1 de junho de 2026, o GitHub Copilot para de cobrar por request. O token vira a unidade de custo.

A DATA DE VIRADA

2026-06-01

Premium Request Units saem. GitHub AI Credits assumem.

1 AI Credit equivale a US$ 0,01. Tokens de entrada, saída e cache são medidos na taxa publicada de cada modelo. O custo agora segue o trabalho realizado, não a frequência com que você pergunta.

UNIDADE ANTIGA		UNIDADE NOVA
1 premium request	→	tokens, 1 cr = US$ 0,01

ANTES E DEPOIS

A unidade de custo passou de intenção para trabalho.

ANTES · PREMIUM REQUESTS

Custo por intenção

Um prompt de uma palavra e uma sessão de agente de quarenta turnos contavam igual, uma request cada. Fácil de orçar, mas desconectado do compute realmente consumido.

DEPOIS · AI CREDITS

Custo por trabalho

Tokens de entrada, saída e cache na taxa de cada modelo. Uma sessão agêntica longa é de fato muito mais cara que uma pergunta de chat. A conta agora bate com a realidade.

A EQUAÇÃO DO BILLING

Dois fatores, e você controla os dois.

A EQUAÇÃO

custo = modelo vezes tokens

custo = entrada vezes taxa-de-entrada, mais cache vezes taxa-de-cache, mais saída vezes taxa-de-saída. 1 AI Credit = US$ 0,01. Toda tática deste deck age sobre o modelo ou sobre os tokens.

O QUE CONSOME CRÉDITOS

Trabalho pesado, não autocomplete

Chat, agentes, coding agent e modelos premium consomem créditos. Code completions e Next Edit Suggestions não. A revisão de código do GitHub Copilot custa duas vezes: AI Credits mais minutos de GitHub Actions.

O QUE NÃO MUDA

Os preços base seguem, e o dia a dia continua gratuito.

01

Preços seguem iguais

Pro US$ 10, Pro+ US$ 39, Business US$ 19, Enterprise US$ 39 por usuário. Cada licença soma seu valor em créditos a um pool compartilhado.

02

Completions seguem grátis

Code completions e Next Edit Suggestions seguem ilimitadas e não consomem AI Credits. A experiência inline do dia a dia não muda.

03

Só o trabalho pesado mede

Chat, modo agente, coding agent, Spaces e modelos premium consomem créditos. É ali que o medidor realmente roda.

DO BUFFET AO À LA CARTE

Não é cobrança injusta. É a devolução de um controle que não existia.

01

O buffet escondia o preço

O prato caro sempre custou caro de produzir. O buffet de preço fixo escondia o custo de cada prato e premiava o desperdício em silêncio.

02

Agora o cardápio tem preços

O usage-based billing entrega o cardápio à la carte com os preços impressos. A cozinha não mudou. Agora você pode escolher.

03

O mercado já se moveu

FinOps 2026: 98 por cento das organizações já gerem gasto de IA, contra 31 por cento em 2024. É maturidade de mercado, não uma esquisitice do GitHub.

POR QUE O PREÇO ESTÁ EVOLUINDO

Cobrança por token bate com como o compute de IA funciona.

01

Tokens, não requests

Os provedores cobram o GitHub em tokens. Uma request pode ter centenas ou dezenas de milhares. Preço por request igual nunca bateu com o custo.

02

O mercado se move rápido

Labs lançam modelos e reprecificam o tempo todo. Um framework de token absorve isso sem revisão de preço a cada vez.

03

Destrava recursos

Janelas de contexto maiores e workflows agênticos mais ricos só fazem sentido econômico quando o custo é medido por consumo.

A LINHA DO TEMPO

Planeje contra setembro, não contra a conta amortecida de hoje.

ANÚNCIO E PREVIEW	Anunciado em 27 de abril. O preview de billing em maio é sua janela para instrumentar monitoramento e configurar orçamentos antes de qualquer cobrança.	Abr a maio
COLCHÃO NATIVO	O usage-based billing entra em 1 de junho com folga incluída que amortece os primeiros meses. Use como janela de aprendizado do consumo real.	1 jun a 31 ago
A CONTA REAL CHEGA	O colchão expira em 31 de agosto. Setembro é o primeiro mês sem amortecimento. Re-baseie os orçamentos aqui, nunca contra gasto promocional.	A partir de set

POR QUE O IMPACTO VARIA TANTO

A conta é um espelho da maturidade de plataforma, não um veredito.

01

Maturidade, não tamanho

Duas empresas sentem a mesma mudança de formas opostas. A diferença é maturidade de uso, não headcount.

02

Espelho, não veredito

Times com governança pagam proporcional ao valor. Times sem ela pagam pelo caos que o preço fixo escondia.

03

IA amplifica

O DORA 2025 mostrou que a IA não conserta um time, ela amplifica o que já existe. O usage billing só deixa isso visível na fatura.

PARTE

II

Fundamentos.

Como LLMs, agentes, tokens e janelas de contexto funcionam, o modelo mental do qual o resto depende.

O MODELO E A REGRA

Uma máquina de probabilidade de palavras, alimentada por você.

01

Ela prevê o próximo token

Um LLM é texto entra, texto sai. Não distingue relevante de irrelevante, nem alucinação de fato. Os dois vêm da mesma matemática.

02

Equilíbrio de contexto

A regra central: o mínimo de contexto possível, mas o tanto que for necessário. Demais enviesa a resposta e custa mais, de menos convida à alucinação.

03

Token é fragmento de palavra

Cerca de três quartos de uma palavra. 1M de tokens é mais ou menos a trilogia O Senhor dos Anéis mais O Hobbit. Prompts, arquivos e respostas todos consomem.

A REGRA CENTRAL

O mínimo de contexto possível, mas o tanto que for necessário.

DEMAIS

Enviesa a resposta

Informação irrelevante é pesada, não ignorada. Empurra o modelo para respostas erradas, e você paga por cada token extra, a cada turno.

DE MENOS

Convida à alucinação

Faltar contexto crítico faz o modelo preencher a lacuna, sem mensagem de erro. A matemática não distingue fato de invenção.

O QUE É UM TOKEN

Um token é fragmento de palavra, e a densidade varia.

01

Tokenização subword

Cerca de três quartos de uma palavra. O inglês fica em torno de 1,3 token por palavra.

02

O idioma importa

Português e espanhol ficam em 1,5 a 1,7 token por palavra. Código tokeniza em densidade variável.

03

Não exagere no ajuste

Você tem pouco controle sobre a tokenização. Pense no alto nível: prompts, arquivos e respostas consomem e se acumulam a cada turno.

O QUE UM AGENTE REALMENTE É

Um agente é só código falando com um modelo, muitas vezes.

01

O harness

Um agente é uma aplicação: VS Code Chat, GitHub Copilot CLI, o coding agent, até Claude Code ou Codex. Ele fala com o modelo por você, muitas vezes por tarefa.

02

O LLM

O modelo em si é só GPT, Claude ou Gemini. A interação não é mágica, é texto, e o modelo é stateless.

03

Suas alavancas

Você o influencia por três coisas: seu prompt, os arquivos do projeto, e os configs do agente, instruções, skills, MCP.

O EFEITO DO REENVIO

2M+

O modelo não lembra. Ele relê, a cada turno.

Uma sessão de 50.000 tokens em 40 turnos envia cerca de 2 milhões de tokens de entrada, mesmo que seu último prompt tenha 20 palavras. Como LLMs são stateless, a conversa inteira é reenviada a cada turno, então o inchaço de contexto é a maior fonte de desperdício.

PARA ONDE OS TOKENS DE FATO VÃO

Seu prompt é a menor fatia da conta.

01	CUSTO FIXO	**System prompt e schemas de tools**	Você não muda isso, mas é por que toda sessão começa acima de zero.
02	CRESCE A CADA TURNO	**Histórico da conversa**	A fatia que a maioria nunca pensa, e a que se acumula com o tamanho da sessão.
03	A DESCUIDADA	**Resultados de tool-call**	Leituras de arquivo, saída de shell, testes. Uma leitura descuidada de um arquivo grande viaja junto para sempre.
04	MENOR, AINDA COBRADA	**Seu prompt e a saída**	Seu prompt digitado costuma ser a menor fatia. A resposta do modelo, incluindo tokens de raciocínio, também é cobrada.

TRÊS TIPOS DE TOKEN, TRÊS COMPORTAMENTOS

Saída é o token mais caro. Cache é o mais barato.

01

Entrada

Tudo que você envia: prompt, histórico, anexos, system prompt, schemas de ferramentas. Re-cobrado a cada turno. A taxa base.

02

Saída

O que o modelo gera, incluindo tokens invisíveis de raciocínio. Tipicamente 4 a 10 vezes a taxa de entrada. Respostas prolixas custam dinheiro de verdade.

03

Cache

Um prefixo estável que o provedor reproduz do cache. Cerca de 10 por cento da entrada, um desconto de 90 por cento. A maior alavanca para trabalho agêntico.

JANELA VERSUS CONTA

Uma janela maior não é uma conta menor.

A JANELA DE CONTEXTO

O que o modelo pode segurar

1 token é cerca de três quartos de palavra. 1M de tokens é mais ou menos a trilogia O Senhor dos Anéis mais O Hobbit. A janela é a capacidade.

SUA CONTA

O que você de fato põe nela

Tokens medem linear, esteja você a 20 ou 80 por cento. Fique em 500K numa janela de 1M e paga 500K a cada turno. A compactação te salva da parede, não do custo de chegar perto.

DOIS MODOS DE FALHA DO CONTEXTO LONGO

Contexto maior é ao mesmo tempo mais caro e de menor qualidade.

PERDIDO NO MEIO

Uma curva de recall em U

Modelos usam melhor a informação no início e no fim da janela, pior no meio. Troque de tarefa no meio da sessão e o modelo pode voltar à primeira instrução. Comece uma janela nova por tarefa.

CONTEXT ROT

Qualidade cai antes do overflow

A Chroma testou 18 modelos de fronteira. Todos pioram quando a entrada cresce, muitas vezes bem antes de encher a janela. Mantenha a janela abaixo de uns 60 a 70 por cento.

PARTE

III

Três superfícies, uma carteira.

Uma conta, três comportamentos. Cada superfície tem seu fator de custo dominante e seus controles.

A PLATAFORMA GITHUB E A FAMÍLIA GITHUB COPILOT

Uma plataforma, duas classes de custo, muitos modelos.

01

A plataforma

O GitHub é repos, pull requests, issues e Actions. O GitHub Copilot é a camada de IA tecida nele, acessada pelo VS Code, pelo CLI, pela web e por outras IDEs.

02

Duas classes de custo

Grátis e ilimitado: code completions e Next Edit Suggestions. Medido em AI Credits: chat, agentes, coding agent, Spaces, modelos premium.

03

Multi-modelo

Os mesmos tiers vêm da Anthropic, OpenAI e Google, roteados por um único plano de controle e uma única unidade de cobrança, AI Credits.

UMA CARTEIRA, TRÊS COMPORTAMENTOS

A otimização é por superfície. A física é a mesma.

01

Terminal · CLI

Fator: reenvio do histórico inteiro a cada turno. Controles: /clear, /compact, /context, sub-agentes especialistas.

02

Editor · VS Code

Fator: anexos mais a proliferação de tool-calls do agente. Controles: escolha de modo, referências hash-file, instruções, escopo de MCP.

03

Web · GitHub.com

Fator: execuções do coding agent com grande raio de impacto. Controles: issues bem escopadas, Spaces enxutos, PR review.

O TERMINAL

O GitHub Copilot CLI premia a higiene de contexto.

01

Clear e compact

/clear reseta entre tarefas distintas, /compact resume no meio, /context mostra a composição, /usage mostra quanto a sessão custou.

02

Troque o modelo no meio

/model deixa você planejar num modelo forte, implementar num barato, revisar num especialista. Cada passo no tier mais barato que conclui.

03

Especialistas poupam tokens

/explore, /plan, /review e /delegate rodam em contexto isolado. A sessão principal só vê o resumo, não os arquivos que eles leram.

O EDITOR

Escolha o modo mais barato que resolve, fixe contexto estreito.

$

Ask

Somente leitura. Explicações, dúvidas de design, consulta a docs. Sem escrever arquivos. O modo mais barato e o certo para a maioria das perguntas.

$

Plan

Spec primeiro. Gera um plano revisável que você entrega ao Agent. O plano estreita o que o Agent lê e altera, e daí vem a economia.

$$$

Agent

Loop autônomo com ferramentas. Mais barato quando guiado por uma spec. Prefira hash-file a hash-codebase, e recomece em vez de aparar.

POR BAIXO DO CAPÔ NO VS CODE

Duas mecânicas que moldam a conta em silêncio.

ÍNDICE DO WORKSPACE

O que o hash-codebase consulta

O hash-codebase consulta um índice semântico, não um grep ao vivo. Para repos do GitHub é remoto e quase instantâneo desde março de 2025. Um índice frio força leituras mais amplas e caras.

COMPACTAÇÃO

/compact no chat do editor

Resume turnos antigos para liberar contexto, com uma dica do que manter. Mais barato que começar do zero quando há continuidade, muito mais barato que estourar.

A WEB

Operações maiores, menos frequentes, raio de impacto maior.

01

Coding agent

Issue para branch para PR numa VM sandbox, GA desde setembro de 2025. A qualidade da issue domina a conta, então escope bem com ponteiros de arquivo.

02

GitHub Copilot Spaces

Contexto curado entre repos e docs. Três arquivos ganham de três repos. Um Space por tipo de pergunta, atualize em vez de acumular.

03

Um arquivo, três superfícies

O copilot-instructions.md é respeitado pelo VS Code, pelo coding agent e pelo CLI. Invista nele uma vez, ganhe em todo lugar.

MESMA TAREFA, TRÊS SUPERFÍCIES, TRÊS FORMATOS DE CUSTO

O botão de custo é o mesmo: quão estreito você escopa antes do modelo ler.

01

CLI

/explore, /plan, escopo estreito. Quando você já está no terminal para uma edição rápida.

02

VS Code

O Plan gera uma spec, entregue ao Agent com escopo hash-file. Para uma mudança média que você revisa antes.

03

GitHub.com

Issue com ponteiros de arquivo, atribua @copilot, revise o PR. Para uma mudança delimitada de que você pode se afastar.

CODING AGENT, QUANDO COMPENSA

Delimitado e bem especificado, ou não pague por isso.

BOM FIT

Compensa

Subir deps, adicionar testes de um módulo, um rename no repo. Issues com ponteiros de arquivo e critérios de aceite, em repos com testes fortes e CI.

MAU FIT

Não pague por isso

Trabalho exploratório vago, arquitetura transversal, repos greenfield, repos sem testes, ou algo que deveria ter sido uma conversa de dez linhas.

GITHUB COPILOT SPACES, CONTEXTO CURADO

Três arquivos ganham de três repos.

01

O que há num Space

Repos e caminhos específicos, notas coladas, instruções em texto livre, links de docs. Enviados ao contexto no momento da consulta.

02

Alavancas de custo

Fontes estreitas, um Space por tipo de pergunta, atualize em vez de acumular. Fontes velhas custam igual a novas.

03

Histórico

Spaces saíram em maio de 2025 e substituíram os Knowledge Bases em setembro de 2025. Se você vê KB, é o termo legado.

A WEB SÃO SEIS PRODUTOS, UMA CARTEIRA

Conheça o formato de custo de cada um.

01

Chat e Edits

Q&A ancorado no repo e edições por arquivo. Os pontos de entrada mais baratos da web.

02

PR review

Auto-revisão escopada ao diff. Agora também consome minutos de Actions pela análise agêntica.

03

Spark e coding agent

Gerar apps inteiros, ou execuções issue-to-PR. O mais pesado por execução, geração longa.

MECÂNICA DE COMPACTAÇÃO DO CLI

A compactação protege a parede, não a conta.

01

Auto-compact em 80 a 95%

Background perto de 80, hard perto de 95, escalado ao modelo. Só dispara quando você está perto.

02

Tokens medem linear

Em 500K numa janela de 1M você paga 500K a cada turno, bem antes da compactação. Janelas maiores não são contas menores.

03

Context antes, resume sempre

Abaixo de 40 por cento, não pague para compactar ainda. Resuma de um summary salvo em vez de recolar contexto.

PARTE

IV

Qualidade de agente primeiro.

Por que otimizar qualidade supera otimizar custo, e por que os dois andam na mesma direção.

O REENQUADRAMENTO

Pare de apostar. Envie menos agentes que acertam.

A PERGUNTA ERRADA

Aposta com agentes

Como disparar 20 foguetes baratos na direção da Lua e torcer para um pousar. Pouco contexto, prompt preguiçoso, manda, e tenta de novo se falhar. Insustentável quando cada dev dispara dezenas de agentes por dia.

A PERGUNTA CERTA

Faça cada token contar

Aumente o valor e a qualidade de cada agente antes de enviar. Menos agentes, melhores, já significa menos tokens. Otimize o retorno por agente.

QUALIDADE GERA VALOR GERA ROI

Otimizar custo quando o valor é zero é trabalho inútil.

A FÓRMULA

ROI do agente

ROI = valor da saída menos custo de tokens, dividido pelo custo de tokens. Você não calcula limpo, mas ela guia: zerar o custo de uma saída inútil não é vitória.

A DINÂMICA

Cortar contexto faz os dois

Aumentar o valor de um agente muitas vezes se faz enviando menos tokens. Aparar contexto irrelevante é uma alavanca que melhora a qualidade e baixa o custo de uma vez.

O PROBLEMA DO ERRO COMPOSTO

Pequenos erros por passo se multiplicam no fluxo.

POR PASSO

99%

Acurácia otimista em um único passo

SE ACUMULA

50 steps

NO SUCESSO TOTAL

APÓS 50 PASSOS

61%

Mesmo com 99 por cento por passo. Com 95 por cento você cai para perto de 8 por cento.

SHIFT-LEFT, PARA AGENTES

Controles determinísticos reiniciam o relógio da acurácia.

01

Testes são determinísticos

Um teste passa ou falha, sem probabilidade. Um teste falho para o agente que derivou e o força a reconstruir sobre uma base estável.

02

53 por cento é teste

O time do GitHub Copilot CLI entrega cerca de 500 PRs por semana. A prática número um são testes, 53 por cento do código.

03

Além de testes

Linters, checadores de tipo e scanners de segurança são guard-rails que o agente executa. Sem eles, ele empilha mudança bugada sobre mudança bugada.

A TESE

Custo e qualidade andam na mesma direção.

CONTAR TOKENS

A armadilha

Espremer tokens a ponto de o agente produzir pior não economiza, você pagou por um resultado pior e pelo retrabalho que vem depois.

FAZER CONTAR

A alavanca

Cortar contexto irrelevante eleva a qualidade e baixa o custo ao mesmo tempo. Ajuste o esforço à maturidade: quanto mais agentes você roda, mais isso compensa.

AJUSTE O ESFORÇO À MATURIDADE

O quanto você otimiza depende de quantos agentes você roda.

ENGENHEIRO ASSISTIDO POR IA

Um agente, quase sync

Cerca de 10 agentes por dia, IA como assistente. Mesmo 50 por cento de economia num mês de 20 dólares dá só 10. Aprenda os fundamentos, aplique os poucos hábitos do topo.

ENGENHEIRO DE IA

Orquestrador de muitos

Dezenas ou centenas de agentes async. Cada por cento de tokens e de qualidade, lembre da matemática do erro composto, é multiplicado pela frota. O framework inteiro compensa.

TRAÇOS DE LONGO PRAZO

O que te torna valioso no desenvolvimento agêntico.

01

Habilidades analíticas

Codar nunca foi o valor, a análise foi. Dizer ao agente exatamente o que fazer, na língua do domínio, vira a habilidade mais valiosa.

02

Boa arquitetura

Domain-driven design, hexagonal, CQRS, event-driven. Arquitetura limpa dá guard-rails ao agente e reduz erros.

03

Itere nos configs

Você é um engenheiro de contexto agora. Trate erros do agente como incidentes, mantenha configs frescos, recrie por trimestre.

PARTE

V

As seis alavancas.

Seis alavancas que se acumulam, cada uma corrige um fator de gasto. Modelos de campo colocam o efeito combinado na faixa de 37 a 44 por cento.

SEIS FATORES DE GASTO DE TOKEN

Diagnostique a causa antes de aplicar o remédio.

01

Contexto, 30 a 50%

Arquivos abertos, indexação e histórico enviados a cada turno.

02

Modelo, 10 a 24x

Tier de fronteira versus leve, o spread do Opus ao mini.

03

Instruções, 10x sem cache

O mesmo contexto de projeto reenviado a cada turno sem cache.

04

Raciocínio, 10 a 80x

Traços de pensamento alto e máximo, cobrados como tokens de saída.

05

Agentes e tools, 4 a 30x

Loops de agente re-cobram o trabalho, schemas de MCP reenviados a cada turno.

06

Sessões, até 8x

O histórico se acumula entre turnos, o turno 30 sem gestão.

O FRAMEWORK

Seis fatores, seis alavancas, um efeito composto.

01

Disciplina de contexto

Anexe o menor escopo que deixa o modelo responder. O token mais barato é o que você nunca envia.

02

Roteamento de modelo

Case o modelo com a tarefa. Um spread de 10 a 24 vezes entre tiers é dinheiro real. Use Auto como padrão.

03

Engenharia de instrução

Escreva os padrões do time uma vez em arquivos cacheáveis, e reuse com cerca de 90 por cento de desconto.

04

Profundidade de raciocínio

Pensar é cobrado como saída. Padrão médio, escale só para turnos realmente difíceis.

05

Agentes e ferramentas

Apare o overhead de MCP, especialize papéis, delegue descoberta a subagentes. Governe o loop ou a conta explode.

06

Ciclo de sessão

O histórico se acumula a cada turno. Um tópico, uma sessão. Resete, não acumule.

O EFEITO COMPOSTO

~41%

As alavancas se somam. A mesma licença faz muito mais trabalho útil.

Aplicadas juntas, as alavancas chegam a cerca de 41 por cento de redução de tokens sem perda mensurável de produtividade, dentro da faixa de 37 a 44 por cento. Os números são ilustrativos, meça sua própria baseline.

REDUÇÃO DE TOKENS		TRABALHO ÚTIL POR LICENÇA
37 a 44 por cento	→	cerca de 1,6 a 1,8 vez

ALAVANCAS 1 E 2

Envie menos, e envie ao modelo certo.

ALAVANCA 1 · DISCIPLINA DE CONTEXTO

15 vezes menos entrada

Nomear os 3 arquivos que importam em vez de varrer o codebase inteiro levou um refactor real de cerca de 25.000 para 1.700 tokens de entrada por turno, com uma resposta mais afiada.

ALAVANCA 2 · ROTEAMENTO DE MODELO

6 vezes mais barato, mesma feature

Planeje num modelo poderoso, implemente num leve, revise num versátil. Uma feature de OAuth caiu de cerca de US$ 13,75 para US$ 2,30, com a mesma qualidade.

TAXAS POR MODELO, AI CREDITS POR 1M DE TOKENS

Escolher modelo é decisão de orçamento, não de estética.

MODELO	ENTRADA	SAÍDA	MELHOR USO
GPT-5 mini	25 cr	200 cr	Chat diário, completions de rotina
GPT-5	125 cr	1,000 cr	Raciocínio e código de complexidade média
Claude Sonnet 4.6	300 cr	1,500 cr	Contexto longo, trabalho de código sustentado
Claude Opus 4.7	1,500 cr	7,500 cr	Tarefas difíceis de agente, uso reservado

Nenhum modelo é gratuito no usage-based billing: todo modelo, inclusive o tier leve, consome créditos por token. Só code completions e Next Edit Suggestions não consomem créditos. Taxas e nomes de modelo são ilustrativos em meados de 2026, sempre valide contra o rate card oficial de modelos e preços do GitHub Copilot antes de fechar orçamentos.

CASE O MODELO COM A TAREFA

Três tiers, um spread de 10 a 24 vezes.

01

Leve

Q&A simples, formatação, boilerplate, refactors simples. O padrão barato para rotina.

02

Versátil

Código do dia a dia, feature, review, debugging. O cavalo de batalha onde mora o gasto.

03

Poderoso

Arquitetura, debugging difícil, security review, lógica nova. Reserve, o multiplicador é alto.

ALAVANCAS 3 E 4

Cacheie o prefixo recorrente, pague o pensamento que a tarefa exige.

ALAVANCA 3 · ENGENHARIA DE INSTRUÇÃO

Pague uma vez, reuse com 90 por cento off

Mantenha os arquivos do time estáveis para o cache valer. Acrescente regras novas no fim, nunca reordene. Em 50 devs isso poupa na ordem de US$ 32.000 por ano só no prefixo de instruções.

ALAVANCA 4 · PROFUNDIDADE DE RACIOCÍNIO

Padrão médio

Em modelos de raciocínio, esforço alto ou máximo pode multiplicar a conta de 10 a 80 vezes. Decomponha: um plano médio, vários passos baixos baratos, uma revisão média. Cerca de 7 vezes mais barato por tarefa.

ESFORÇO DE RACIOCÍNIO É COBRADO COMO SAÍDA

De LOW a MAX pode ser até 80 vezes a conta.

1x

LOW

1x. Cerca de 200 a 800 tokens de pensamento. Serve para a maioria dos passos.

2-4x

MEDIUM

2 a 4x. Cerca de 1K a 4K. O padrão certo para a maioria do código.

10-25x

HIGH

10 a 25x. Cerca de 5K a 20K. Guarde para turnos realmente difíceis.

50-80x

MAX

50 a 80x. Até 64K tokens de pensamento. Só design arquitetural e bugs sutis.

ALAVANCAS 5 E 6

Governe o loop do agente, resete a sessão.

ALAVANCA 5 · AGENTES E FERRAMENTAS

Apare, especialize, delegue

MCP apara o que é enviado, hooks aparam o que é armazenado, skills pulam a exploração. Especialize papéis e delegue descoberta a subagentes. Um debug de 30 turnos caiu de cerca de US$ 45 para US$ 6.

ALAVANCA 6 · CICLO DE SESSÃO

Um tópico, uma sessão

No turno 30 o pedaço de histórico é cerca de 90 por cento da conta de cada turno. Tópico novo, chat novo. Compacte com foco, bifurque para explorar, arquive ao terminar.

HÁBITOS QUE SOMAM POR CIMA

Restrinja a saída, use o modo Ask, adote o kit inicial.

01

Restrinja a saída

Saída é o token mais caro. Limite-a: uma frase, três bullets, só código. Só código corta a saída de 40 a 70 por cento.

02

Use o modo Ask

O modo define o número de chamadas: Ask é uma, Agent é 5 a 25. Não pague overhead de loop de agente por uma pergunta delimitada.

03

O kit inicial

Escolha três: /clear, /model, /usage. Controle de contexto, controle de custo, visibilidade de custo, um para cada dimensão.

ALAVANCA 5A, APARE O OVERHEAD DE TOOLS

Três fontes de inchaço, três alavancas.

01

MCP apara o enviado

Desabilite servidores que não precisa hoje, use um CLI para avulsos. Cerca de 5K para 1K por request.

02

Hooks aparam o armazenado

Um hook local filtra saída ruidosa antes de entrar no histórico. Cerca de 10K bruto para 200 filtrado.

03

Skills pulam a exploração

Um SKILL.md descreve a estrutura uma vez em vez de ler 5+ arquivos. Cerca de 5K para 500.

ALAVANCA 5B, AGENTES DE HANDOFF

Fixe um tier por papel, passe um payload pequeno.

01

Planner, poderoso

Tools só de leitura, produz um plano de 5 passos e notas de design. A taxa de fronteira, paga uma vez.

02

Implementer, leve

Tools de edição, executa um passo do plano por turno. O grosso do trabalho, no tier barato.

03

Reviewer, versátil

Só leitura, diff e testes. Naive 10 turnos Opus cerca de $15, handoff cerca de $2,11, ~7x mais barato.

ALAVANCA 5C, SUBAGENTES

Pague a descoberta uma vez, não a cada turno.

SEM SUBAGENTE

Re-cobrado a cada turno

O pai recarrega o mesmo conteúdo de arquivo a cada turno. Um refactor de 20 arquivos em 4 turnos envia 86K de entrada, pago quatro vezes.

COM SUBAGENTE

Lido uma vez, resumido

O subagente lê os arquivos em contexto isolado e devolve um resumo de 1K. A entrada do pai fica plana, cerca de 8K no total, umas 10 vezes menos.

SUBAGENTE VERSUS HANDOFF

Escolha pelo formato da tarefa, não por preferência.

5.1

Subagente (5.1)

Descoberta delimitada, volume de arquivos o gargalo, uma resposta, fan-out paralelo, o pai fica. Descoberta vai para subagente.

5.3

Handoff (5.3)

Build multi-fase, profundidade de raciocínio o gargalo, três ou mais fases, sequencial, o bastão passa. Build vai para handoff.

PARTE

VI

Aplicar, governar, medir.

Engenharia de contexto, orçamentos e controles, observabilidade e um playbook de adoção sequenciado.

ENGENHARIA DE CONTEXTO

Você agora é um engenheiro de contexto.

01

Instruções persistentes

O copilot-instructions.md viaja em toda sessão. Mantenha pequeno, escreva você mesma, registre erros recorrentes, recrie a cada trimestre.

02

Skills e regras com escopo

Instruções com escopo por caminho e skills carregam só quando relevantes. Prompt files e chat modes limitam o toolset e o raio de impacto.

03

Agentes e subagentes

Fixe modelo e ferramentas por papel para evitar caminhos errados. Subagentes leem arquivos uma vez e devolvem só um resumo, mantendo o pai enxuto.

SEU PROMPT, SEMPRE ATIVO

Direcione desde o início, não corte palavras.

01

Seja preciso

Não 'corrija o bug', mas 'a issue 45 descreve X, corrija'. Aponte o sintoma e o arquivo exatos.

02

Adicione sinais de parada

Quando o bug estiver corrigido e os testes passarem, pare. Evita que o agente vagueie por trabalho não pedido.

03

Antecipe o contexto conhecido

Se você sabe os arquivos, docs ou skill, nomeie-os. Cada loop de descoberta poupado é token que você não paga.

PESQUISAR, PLANEJAR, IMPLEMENTAR

Trabalhe em fases, com uma janela nova entre elas.

PESQUISA	Carrega muitos arquivos, a maioria irrelevante para a implementação. Faça numa janela própria ou subagente.	Opcional
PLANO	Um modelo de raciocínio produz uma spec precisa, um to-do detalhado que faz o pensamento antes.	Uma vez
IMPLEMENTAR	Execute a spec num modelo mais barato com só o contexto relevante. Com spec clara, agentes em paralelo ficam possíveis.	Por passo

CONTROLES DETERMINÍSTICOS

Testes reiniciam o relógio da acurácia.

01

Testes são determinísticos

Passa ou falha, sem probabilidade. Um teste vermelho para o agente que derivou e força reconstruir sobre base estável.

02

53 por cento é teste

O time do GitHub Copilot CLI entrega cerca de 500 PRs por semana. A prática número um são testes, 53 por cento do código.

03

Além de testes

Linters, checadores de tipo, scanners de segurança. Qualquer guard-rail que você expresse como código, o agente executa.

CONFIGS COM ESCOPO E REUSÁVEIS

Carregue orientação só quando ela é relevante.

01

Instruções com escopo

Globs de caminho applyTo carregam só quando arquivos correspondentes são anexados, mantendo o always-on pequeno.

02

Prompt files

Prompts versionados, invocados à mão. Acaba com a deriva de cada dev ter seu prompt de estimação.

03

Chat modes

Uma allowlist estreita de tools literalmente não entra em loop de Agent, então a conta é limitada por design.

SKILLS VERSUS MCP

Ofereça comportamento sob demanda, não anuncie a cada turno.

SKILLS · CARGA PROGRESSIVA

1 a 2 KB até uma casar

Uma skill expõe só o nome e a descrição até seu prompt casar. Instale 20 skills e você soma poucos KB, não os corpos inteiros. Padrão aberto em VS Code, CLI e cloud agent.

MCP · SEMPRE ANUNCIADO

10 a 15 KB a cada turno

Cada tool de MCP habilitada envia seu schema a cada turno, usada ou não. Escope por workspace, apare para o que se usa na semana e prefira um CLI como gh para leitura pesada.

INSTRUÇÕES PERSISTENTES

Mantenha pequeno, escreva você mesma, recrie por trimestre.

MANTER

Regras que o modelo não infere

Riscos não óbvios, o framework de teste e o comando de build, regras explícitas de não mexer em arquivos e padrões. Cerca de 150 tokens, pagos uma vez e depois cacheados.

CORTAR

Tudo que o código já diz

Textos de onboarding, arquitetura em ASCII, guias de estilo inteiros, regras duplicadas. Pesquisa de 2026: arquivos de contexto gerados somam 20 a 23 por cento de tokens para cerca de menos 2 por cento de correção.

INSTRUÇÕES COMPRIMIDAS

As mesmas regras, cerca de 64 por cento menos tokens.

ANTES, ~120 TOKENS

Prosa

Sempre use pnpm, nunca npm. Testes ficam em spec/. Não modifique arquivos em generated/. Documente exports com jsdoc, e assim por diante.

DEPOIS, ~50 TOKENS

Chave-valor enxuto

pkg: pnpm. tests: spec/. no-edit: generated/. docs: jsdoc on exports. Mesmo sentido, pago a cada turno, 64 por cento mais leve.

HIGIENE DE MCP

Por workspace ganha de global, apare a lista de tools.

ESCOPE POR WORKSPACE

.vscode/mcp.json

Vive no git, revisável, escopado ao repo que precisa. Um servidor global ruim polui todo projeto.

APARE O QUE É ENVIADO

Toda tool manda seu schema

Um conjunto de 40 tools anuncia 10 a 15 KB por turno, usadas ou não. Apare para o uso semanal e prefira o CLI gh para leitura pesada.

AGENTRC

Pare de escrever a pilha de instruções na mão. Gere.

01

Readiness

Pontua o repo em 9 pilares e um modelo de maturidade de 5 níveis. Use como gate de CI que falha abaixo do nível 3.

02

Instructions

Gera copilot-instructions.md, a config de MCP, settings e AGENTS.md a partir do próprio código.

03

Eval

Re-roda casos salvos para pegar a deterioração das instruções no CI. Experimental, então fixe uma versão.

PAPÉIS E ISOLAMENTO

Fixe o papel, isole a descoberta.

AGENTES CUSTOMIZADOS

Evite caminhos errados

Fixe modelo e ferramentas por papel: Planner, Implementer, Reviewer. O ganho real não é tokens, é não dar ao agente uma ferramenta que você não quer que ele use.

SUBAGENTES

Pague a descoberta uma vez

Um subagente lê vários arquivos em contexto isolado e devolve só um resumo. O pai fica enxuto numa sessão longa, muitas vezes cerca de 10 vezes menos entrada re-cobrada.

TERRITÓRIO DE POWER USER

Táticas avançadas, para quando você orquestra muitos agentes.

01

Pensar em código

Escreva um script para filtrar a saída antes do modelo vê-la. Filtre uma resposta de API para os campos que importam, não despeje o payload inteiro.

02

CLI em vez de MCP

Para leitura pesada, um CLI conhecido como gh é mais enxuto que a superfície de tools de um servidor MCP. Menos tokens estáticos, mesma resposta.

03

Chronicle e RTK

Rode chronicle para analisar seus logs de sessão em busca de melhorias, e ferramentas como RTK para aparar saídas longas de shell ao que o agente precisa.

OS OUTROS CONFIGS

Orientação com escopo, condicional e automática.

01

Instruções com escopo

Condicionais, por caminho de arquivo. Oferecidas ao agente como skills. Comece estático, vá para escopo só quando o arquivo cresce.

02

Prompt files

Prompts reusáveis invocados à mão. Um ponto de partida comum para acionar skills ou custom agents.

03

GitHub Copilot memory

Orientação pequena, automática, always-on, aprendida do seu comportamento, aplicada entre superfícies. Cheque periodicamente.

GOVERNANÇA · O POOL COMPARTILHADO

Cada licença contribui com créditos para um pool único.

ANTES · POR ASSENTO

Saldos presos

As requests não usadas de um usuário leve não ajudavam um usuário pesado, que batia em overage enquanto outros sobravam.

DEPOIS · POOL COMPARTILHADO

Créditos fluem para a necessidade

Todos puxam de um pool primeiro. Só o gasto além do pool é gasto adicional, governado pelas camadas de orçamento. Orçamentos por usuário limitam cada pessoa.

POOL NA PRÁTICA

O uso incluído flui pela enterprise.

MODELO PRU

Saldos presos

As requests não usadas de um usuário leve não ajudavam um pesado, que batia em overage enquanto outros sobravam. Um não drenava o outro.

POOL COMPARTILHADO

Menos desperdício, uso desigual

Usuários puxam de um pool pela necessidade real, então valor não usado não fica preso. O trade-off, consumo desigual, por isso existem orçamentos por usuário.

AS QUATRO CAMADAS DE ORÇAMENTO

Guard-rails, não algemas. Qualquer orçamento em zero para o uso.

04

CAMADA · TETO GLOBAL

Orçamento enterprise

Limita o gasto adicional total além do pool. Alertas em 75, 90 e 100 por cento. Cost centers podem ser excluídos para uma unidade financiada seguir trabalhando.

03

CAMADA · POR UNIDADE

Orçamento de cost center

Aloca gasto adicional a uma org ou grupo de usuários. Pode mapear uma Azure subscription por cost center.

02

CAMADA · PADRÃO JUSTO

Orçamento universal por usuário

Um teto padrão sobre o consumo total por usuário. O jeito fácil de impedir uma pessoa de drenar o pool.

01

CAMADA · OVERRIDE DE POWER USER

Orçamento individual por usuário

Sobrepõe o padrão universal para usuários específicos. Máximo de 10.000 orçamentos na enterprise, então use overrides com parcimônia.

O PEDIDO DO DIA ZERO

150%

Ponha o orçamento por usuário em 150 por cento, e mais dois movimentos.

150 por cento é alto o bastante para nunca bloquear o trabalho normal e baixo o bastante para limitar um loop descontrolado a cerca de 12 horas. Um orçamento zero bloqueia o acesso por completo, não há fallback para um modelo mais barato.

TRÊS MOVIMENTOS, HOJE

ULB 150% · Auto padrão · arquivo de instruções → RESULTADO

cerca de 1,6 a 1,8x de throughput, mesma licença

CENÁRIOS DE ORÇAMENTO

Três formas que as quatro camadas assumem na prática.

01

Gerir uso compartilhado

Um orçamento universal por usuário define o padrão, overrides individuais elevam power users específicos. Controle base com flexibilidade por pessoa.

02

Por unidade de negócio

Orçamentos de cost center por org, com o orçamento enterprise como teto global e failsafe para quem está fora de um cost center.

03

Power users numa unidade

As quatro camadas juntas: padrão universal, overrides individuais dentro de um cost center, o orçamento da unidade e o teto enterprise, opcionalmente excluindo a unidade.

CRIANDO OS GUARD-RAILS CERTOS

Quatro perguntas antes de definir qualquer orçamento.

01

Orçamento enterprise

Quanto o negócio está disposto a gastar em serviços de IA?

02

Orçamento universal por usuário

Quanto cada engenheiro deveria poder gastar?

03

Orçamento de cost center

Qual o máximo que cada unidade ou time pode gastar?

04

Override individual

Quem precisa de exceção a esses orçamentos?

NOTIFICAÇÕES DE ORÇAMENTO

Alertas antes da parede, e a parada dura nela.

75%

Admin recebe aviso

Em 75 por cento, chega um e-mail. Comece a monitorar de perto.

90%

Decida em 90

Aumente o orçamento, ou deixe atingir o teto. Escolha consciente, não surpresa.

100%

Bloqueado em 100

Usuários param até o próximo ciclo ou um admin eleva o teto, o que leva segundos.

CONTENT EXCLUSION, A BASE DO ADMIN

Bloqueie arquivos do contexto antes de qualquer ajuste.

01

Segredos e env

.env, .env.*, secrets/*, *.pem, *.key. Nunca deixe isso fluir para o contexto.

02

Gerado e vendored

dist/, build/, node_modules/, vendor/. Ruído que o modelo nunca deveria ler.

03

Dados sensíveis

compliance/, customer-data/. Globs por repo, org ou enterprise, propagam em cerca de 30 minutos.

POLÍTICA DE MODELO E AUTO

Faça da escolha eficiente o padrão.

AUTO POR PADRÃO

Um ganho grátis de 10 por cento

O Auto escolhe um modelo apropriado e ganha cerca de 10 por cento de desconto para pagantes. Faça-o o padrão da org.

RESTRINJA PREMIUM

Aprove em etapas

Limite modelos premium para trabalho de rotina em Organization Settings, GitHub Copilot, Policies. Aprove por workflow, time e necessidade medível, não na honra.

OBSERVABILIDADE

Você não otimiza o que não consegue ver.

04

CAMADA · POR TURNO

OTel mais badge de modelo

O VS Code 1.119 emite spans OpenTelemetry com a composição de tokens e cache por turno, mais o modelo resolvido inline. Grátis, local, hoje.

03

CAMADA · POR USUÁRIO

/usage e /context

Sinais de custo na sessão e no fim dela antes da fatura. Rode /usage no fim das suas próximas três sessões para conhecer sua baseline.

02

CAMADA · POR ORG

GitHub Copilot Metrics API

Um dataset publicado, agora incluindo atividade do CLI, exportado para CSV para análise de tendência e re-baseline mensal.

01

CAMADA · DASHBOARDS

Viewer e Grafana

O Metrics Viewer open-source e os dashboards Grafana da Microsoft tornam o desperdício um gráfico compartilhável. Acompanhe a taxa de cache acima de 60 por cento.

TELEMETRIA DO VS CODE 1.119

Três ganchos que viram o custo em dado.

01

Tracing OpenTelemetry

Spans GenAI com composição de tokens e cache por turno, exportados para qualquer backend OTLP.

02

Badge de modelo e multiplicador

Mostra o modelo que o Auto resolveu e seu multiplicador inline. Ligado por padrão.

03

Background todo agent

Passa o controle da lista de tarefas a um modelo leve para o principal não pagar por isso. Desligado por padrão, ligue-o.

MEÇA O DELTA VOCÊ MESMA

Rode a mesma tarefa de dois jeitos e observe a conta.

FLUXO A · NAIVE

Agente, sem escopo

Abra o modo Agent, aponte para hash-codebase, itere 8 a 12 turnos num só chat, aceite o que vier. O caso de controle.

FLUXO B · ENGENHEIRADO

Planeje, depois escope

O Plan gera uma spec, você revisa, entrega ao Agent com escopo hash-file estreito, instruções do repo garantem convenções. Meça tokens OTel, contagem de turnos, tempo até mergeável.

NÚMEROS PARA ACOMPANHAR

Os sinais de estado estável que mantêm o programa honesto.

01

Acerto de cache acima de 60 por cento

Confirma que o prefixo de instruções está estável. Uma taxa caindo quer dizer que alguém edita arquivos do time no meio do dia e perde o desconto.

02

Overage de 10 a 15 por cento

Caindo de 30 por cento ou mais em baselines descontroladas. Revise médias mensais, não picos diários.

03

Revisão semanal do top-10

Olhe as dez sessões mais pesadas por semana. Elas revelam um padrão corrigível: um MCP sempre ligado, uma sessão nunca limpa, um modelo premium fixado para triagem.

O PLAYBOOK DE 30/60/90 DIAS

Faça na ordem. Cada fase barateia a próxima.

ESTABILIZE	Instruções nos repos principais, content exclusion no nível da org, Auto como padrão, OTel conectado para times de alto volume.	Primeiros 30 dias
PADRONIZE	Gere a pilha de instruções com AgentRC, escope MCP por repo, converta prompts em skills, suba o export de métricas e o alerta de 75 por cento.	Dias 31 a 60
ESCALE E GOVERNE	Abra issues prontas para o coding agent, adicione um gate de readiness do AgentRC no CI, re-tier os orçamentos por trimestre com telemetria real.	Dias 61 a 90

O CHECKLIST DE 30 MINUTOS DE SEGUNDA

Seis itens, cada um sob cinco minutos, num repo real.

01

Adicione copilot-instructions.md

Cinco linhas, só convenções, estilo comprimido.

02

Adicione uma instrução com escopo

Seu diretório de maior tráfego, por exemplo src/api/.

03

Mova MCP para .vscode/mcp.json

Por workspace ganha de global, apare para o uso semanal.

04

Use uma skill para tarefa recorrente

PR review, triagem, checagem de regressão.

05

Crie um GitHub Copilot Space

Sua pergunta transversal número um, três fontes no máximo.

06

Abra uma issue para o coding agent

Escopo delimitado, ponteiros de arquivo, critérios de aceite.

O ROADMAP DE TRÊS FASES

Pré-habilitação, otimização, governança.

PRÉ-HABILITAÇÃO	Cerca de duas horas. Ponha ULB 150 por cento, configure o teto de overage com alerta em 80 por cento, defina cost centers, ative o Auto na org, commite um arquivo de instruções inicial.	Dia 0
OTIMIZAÇÃO	Treine disciplina de contexto e hash-mentions, audite MCP por time, ponha o esforço em médio, distribua instruções por caminho, comece a revisão semanal do top-10.	Semanas 1-4
GOVERNANÇA	Exporte o CSV de uso, revise médias mensais, ajuste a ULB para power users justificados, cheque taxa de cache acima de 60 por cento, refine instruções e toolsets.	Mensal

A TRANSIÇÃO DE SEIS MESES

Diagnostique, resgate, ancore nos pontos de decisão.

MÊS 1 · JULHO	Os orçamentos e políticas ainda são adequados? Confirme ou ajuste cedo, enquanto o colchão está ativo.	Confirmar
MÊS 3 · SETEMBRO	Qual o orçamento realista na alocação padrão? Re-baseie e comunique. O checkpoint mais importante.	Re-baseline
MÊS 6 · DEZEMBRO	O tier ainda é ótimo? Há alavancas de renegociação? Renove, mude de tier ou renegocie.	Decidir

SEIS DIMENSÕES DE MATURIDADE, PESADAS POR ALAVANCA DE CUSTO

Diagnostique antes de prescrever. Governança de modelo lidera.

25

Governança de modelo

Peso 25. A maior alavanca isolada de custo, o modelo certo por tarefa.

20

Fundação de plataforma

Peso 20. A raiz do descontrole quando ausente.

15

Disciplina de contexto

Peso 15. Ataca o fator tokens diretamente.

15

Escopo de agente

Peso 15. Controla gasto descontrolado.

15

Primitivos de repositório

Peso 15. O começo da disciplina estrutural.

10

Controle de orçamento

Peso 10. O guard-rail contra contas de horror.

DOIS HORIZONTES

O resgate cabe na janela, a transformação realiza o valor.

RESGATE, JUN A DEZ

Quatro frentes

Governança de modelo, orçamentos como guard-rail, curadoria de contexto, primitivos de repositório. Todos agem sobre a equação modelo vezes tokens.

TRANSFORMAÇÃO

Plataforma por design

Uma plataforma sedimentada, uma camada de contexto e orquestração, um centro de controle no Microsoft Foundry. A eficiência vira propriedade da infraestrutura, não da disciplina individual.

O MODELO DE ADVISORY

Papéis claros, cadência quinzenal.

01

O GBB é o cérebro

Estratégia, diagnóstico, governança, arquitetura, o que e o porquê.

02

Parceiros são os braços

Integram e validam no ambiente do cliente, o como.

03

O cliente é dono do resultado

Dados, participação, implementação. Um checkpoint a cada 15 dias prova que o compromisso está vivo.

A CAMADA COMERCIAL, PLANOS DE PRÉ-COMPRA

Previsibilidade, dimensionada por banda de contrato.

OS TRÊS P3

GitHub, AI Credit, Agent Factory

Pré-pagos, um ano, via Azure subscriptions. Tiers compartilhados: 20k = US$ 19k (5%), 100k = US$ 90k (10%), 500k = US$ 425k (15%).

FIT POR BANDA DE ACD

Confirme antes de propor

Abaixo de 15 por cento de ACD o P3 vence, 15 a 25 só AI Credit P3 com Deal Desk, 25 por cento ou mais o ACD vence. Valide contra o P3 FAQ.

ANTI-PADRÕES A EVITAR

Nomear as armadilhas protege a credibilidade.

01

Conta alta é incompetência

Errado. A conta é espelho da maturidade de plataforma, não veredito. Cargas pesadas podem ser legítimas.

02

Orçar contra gasto promo

O número muda em setembro. Sempre orce contra a alocação real.

03

Prometer a extensão como fato

Ela é condicional. E nunca meça produtividade individual, meça entrega organizacional.

ÚLTIMAS NOTÍCIAS E ROADMAP

Cada vez mais disso vira automático e aplicável por política.

01

Auto e Hydra

O Auto roteia para o modelo capaz mais barato com desconto para pagantes. O Hydra, um classificador de intenção, escolhe o fit sem custo extra e troca só em fronteiras de cache.

02

Coding agent e Spaces

O coding agent está GA, os Spaces substituíram os Knowledge Bases, agent skills são padrão aberto e os dashboards Grafana cobrem GitHub Copilot, Claude Code e Codex.

03

O que vem aí

Roteamento ciente de token, tiers de Auto (Eco, Fast, Balanced, Max) e políticas de admin para impor o Auto. O catálogo de modelos gira sempre, trate taxas como fotos do momento.

COMO O AUTO FUNCIONA

Dois sistemas escolhem o modelo por você.

HYDRA

Classificador de intenção

Um pequeno classificador ModernBERT pontua a complexidade do prompt em raciocínio, debugging, geração de código e tools, e escolhe o mais barato que serve. Sem custo extra.

SELEÇÃO DINÂMICA

Roteamento em tempo real

Considera capacidade, latência e erros. O Auto troca de modelo só em fronteiras de cache, início da conversa e após compactação, para nunca quebrar o cache.

QUATRO ALAVANCAS DE CONFIANÇA DE CUSTO

Não é uma feature, é um roadmap que funciona junto.

01

Rotear tarefas melhor

Task intent casa o trabalho a um caminho de modelo, sem o usuário pesar trade-offs.

02

Governar por time

Enterprise Teams e cost centers atribuem orçamentos onde o uso acontece.

03

Comunicar o roadmap

Confiante, mas cauteloso quanto a tempo e roteamento futuro.

04

O resultado

Mais uso, menos surpresas, controles melhores.

MARCOS RECENTES

Por onde a plataforma andou, para ancorar a linha do tempo.

01	MAR 2025	**Índice semântico GA**	O índice instantâneo de busca semântica de código ficou GA, tornando hash-codebase rápido e preciso em repos hospedados no GitHub.
02	SET 2025	**Coding agent GA, Spaces**	O coding agent assíncrono chegou ao GA, e os GitHub Copilot Spaces substituíram os Knowledge Bases.
03	JAN 2026	**CLI aprimorado**	O GitHub Copilot CLI ganhou agentes mais ricos e gestão de contexto, e dobrou a atividade do CLI na Metrics API de uso.
04	MAI 2026	**VS Code 1.119**	Tracing OpenTelemetry, o badge de modelo e multiplicador, e o background todo agent chegaram, com a UI de UBB pré-montada.

O ROADMAP À FRENTE

Cada vez mais do que você otimiza na mão vira automático.

JUNHO	O Auto chega a mais clientes e vira a única opção para Free e EDU, com UX para explicar qual modelo foi escolhido e otimizações de cache.	Agora
JULHO EM DIANTE	Roteamento ciente de token e tiers de Auto: Eco, Fast, Balanced, Max. Um único botão para sua postura de custo versus capacidade.	A seguir
DIREÇÃO	Políticas de admin para impor o Auto e fixar sua versão. A escolha eficiente vira o padrão aplicável. Trate as taxas de modelo como fotos do momento.	Adiante

ANEXO

VII

Cache.

Dois caches que as pessoas confundem. Um você influencia, o outro você hospeda. E se Foundry mais Redis cabe atrás do VS Code e do CLI.

DOIS CACHES, UMA PALAVRA

Prompt cache reusa a mesma entrada. Cache semântico reusa uma resposta similar.

CACHE A · PROMPT CACHE DO GITHUB COPILOT

Você influencia

O provedor reproduz um prefixo estável entre turnos, os cached tokens da sua conta, cerca de 90 por cento off. O VS Code chegou a mais de 93 por cento de reuso por request. Não há botão, você mantém o prefixo estável.

CACHE B · CACHE SEMÂNTICO

Você hospeda

Devolve uma resposta guardada quando um prompt novo é semanticamente similar a um anterior. Você o constrói para seus próprios apps e agentes na Azure, ele não reduz a conta da assinatura do GitHub Copilot.

ADMINISTRANDO O CACHE A

No VS Code e no CLI, você protege o cache, não o configura.

01

Mantenha aquecido

Mantenha os arquivos de instrução e o conjunto de MCP estáveis na sessão, acrescente regras no fim e deixe o Auto trocar modelo em fronteiras de cache.

02

O que quebra

Editar instruções no meio do dia, trocar de modelo na mão ou mudar o conjunto de ferramentas reescreve o prefixo e você re-paga a preço cheio de entrada.

03

Meça

O OTel do 1.119 mostra cache read e creation por turno, o /context mostra a composição. Mire numa taxa de acerto de cache acima de 60 por cento.

CACHE B, O CACHE SEMÂNTICO QUE VOCÊ HOSPEDA

Para seus próprios agentes na Azure, não para o GitHub Copilot.

01

Azure Managed Redis

Com o módulo RediSearch, o cache externo e vector store. Habilite o RediSearch na criação.

02

Gateway API Management

O GenAI gateway na frente dos seus modelos, também disponível embutido no Foundry.

03

Políticas lookup e store

llm-semantic-cache-lookup e store comparam prompts por proximidade vetorial, devolvendo respostas similares.

GUIA DE DECISÃO

Qual cache, quando.

01

No VS Code ou no CLI

Você está no Cache A, o prompt cache. Proteja-o com disciplina de prefixo. Nada a implantar.

02

Construindo seu próprio agente

Use o Cache B, o cache semântico, no Foundry mais Redis. Uma alavanca de custo real e governável.

03

Os dois ao mesmo tempo

Comum em plataformas maduras. Coexistem e nunca se sobrepõem, camadas totalmente diferentes.

FOUNDRY MAIS AZURE MANAGED REDIS

Camada certa para seus agentes, não para o GitHub Copilot no IDE.

GITHUB COPILOT NO VS CODE E CLI

Não

O GitHub Copilot é um produto gerenciado. Você não insere seu Redis ou gateway Foundry no cache ou no billing dele. A única alavanca é a disciplina de prefixo.

SEUS AGENTES NO FOUNDRY

Sim

API Management como GenAI gateway com as políticas de cache semântico mais Azure Managed Redis e RediSearch corta tokens nos modelos que você chama direto. A plataforma do horizonte de transformação.

CONCLUSÕES

Três pilares, um modelo mental.

01

Custo é contexto

Em toda superfície, o que você envia ao chat é o que você paga. Apare, escope, recomece.

02

Específico ganha de amplo

hash-file melhor que hash-codebase, issue estreita melhor que vaga, Space focado melhor que de tudo um pouco.

03

Persistência acima de prompts

Instruções, chat modes e Spaces no git sobrevivem a qualquer prompt único.

GLOSSÁRIO, PARTE 1

Os termos que atravessam o deck.

AI Credit

A moeda do usage-based billing. 1 crédito = US$ 0,01, medido por token.

Token

Um fragmento de palavra, cerca de três quartos de uma. Entrada, saída ou cache.

Efeito do reenvio

O harness reenvia a conversa inteira a cada turno, então o contexto se acumula.

Janela de contexto

O que o modelo segura. Diferente do que você põe nela, que é o que você paga.

Prompt cache

Replay do provedor de um prefixo estável, cerca de 90 por cento off. Os cached tokens da conta.

Context rot

A qualidade cai com a entrada crescendo, muitas vezes antes de encher a janela.

GLOSSÁRIO, PARTE 2

Os configs e controles.

Auto e Hydra

O Auto roteia para o modelo capaz mais barato, com desconto para pagantes. O Hydra é seu classificador de intenção.

Coding agent

De issue a branch a PR num sandbox. GA desde setembro de 2025.

Skill

Uma capacidade portátil, sob demanda. Carrega só quando a descrição casa.

MCP

Tools externas para agentes. Cada uma anuncia seu schema a cada turno.

ULB

Orçamento por usuário, um teto sobre o consumo total. Recomendado em 150 por cento.

Content exclusion

Controle de admin que bloqueia arquivos do contexto em todas as superfícies.

REFERÊNCIAS

Valide os números contra fontes primárias.

01

GitHub

O post de usage-based billing, o rate card de modelos e preços, docs de budgets e Auto, os changelogs do coding agent e Spaces.

02

Microsoft Learn

Notas do VS Code 1.119, cache semântico e AI gateway no API Management, Azure Managed Redis, Grafana para agentes de IA.

03

Pesquisa

Liu et al. Lost in the Middle, Chroma Context Rot, o relatório DORA 2025, a pesquisa FinOps 2026.

ENCERRAMENTO

Em vez de contar tokens, faça cada token contar.

Pioneering software development with AI and Agentic DevOps.

Contato

Paula Silva
Software Global Black Belt
paulasilva@microsoft.com

Próximo passo

Rode /usage no fim das suas próximas três sessões para conhecer sua baseline, ponha o orçamento por usuário em 150 por cento, ative o Auto como padrão e commite um arquivo de instruções pequeno no seu repo mais ativo.