GITHUB COPILOT AGENTIC FRAMEWORK

O stack agêntico: o mapa e as decisões.

Ato I: a arquitetura de arquivos, onde vivem agents, skills, instructions e prompts, e como o GitHub Copilot os descobre. Ato II: a árvore de decisão para escolher entre Agente Customizado, Generalista com Skill e Generalista Simples em qualquer tarefa.

AUTORA
Paula Silva

FUNÇÃO
Software Global Black Belt

DURAÇÃO
75 a 90 minutos

DATA
2026-06-11

AGENDA

Dois atos, seis partes.

PARTE I	O mapa, por que a arquitetura de arquivos importa e os 6 tipos que formam o ecossistema
PARTE II	Por dentro dos arquivos, anatomia de agents, skills, instructions e prompts
PARTE III	A descoberta, como o GitHub Copilot carrega seu contexto passo a passo
PARTE IV	As decisões, as três opções e as 4 perguntas do roteamento inteligente
PARTE V	Cada caminho a fundo, comparação, matriz de decisão e o GitHub Copilot sugerindo a opção certa
PARTE VI	Produção, três cenários reais, boas práticas de organização e o resumo

PARTE

I

O mapa.

Cada arquivo é uma camada de contexto para o agente. Sem arquitetura, agentes operam sem regras, sem identidade e sem expertise.

O MAPA · POR QUE IMPORTA

Cada arquivo é uma camada de contexto para o agente.

REGRAS AUTOMÁTICAS

.instructions.md define guardrails que o agente NUNCA pode ignorar. São aplicadas automaticamente.

IDENTIDADE DO AGENTE

.github/agents/*.md define QUEM o agente é, seu escopo, ferramentas e para quem delega.

CONHECIMENTO ESPECIALIZADO

SKILL.md codifica expertise do domínio: templates, workflows, checklists e exemplos.

PROMPTS REUTILIZÁVEIS

.prompt.md armazena prompts reutilizáveis que podem ser invocados sob demanda.

Sem arquitetura de arquivos, agentes operam sem regras, sem identidade e sem expertise. O resultado é comportamento inconsistente que ninguém consegue auditar.

O MAPA · VISÃO GERAL

Os 6 tipos de arquivo do ecossistema, cada um com um papel distinto.

ARQUIVO	LOCALIZAÇÃO	ESCOPO	PROPÓSITO
agents/*.md	.github/agents/	Agente específico	Identidade, ferramentas, handoffs
SKILL.md	.github/skill/*/	Skill específico	Expertise, templates, workflows
*.instructions.md	Qualquer diretório	Diretório + filhos	Regras automáticas, guardrails
copilot-instructions.md	.github/	Repositório inteiro	Instruções globais do repo
*.prompt.md	Qualquer diretório	Sob demanda	Prompts reutilizáveis
mcp.json	.vscode/	Workspace	Servidores MCP, ferramentas

Estes 6 tipos formam o ecossistema completo. Cada um é carregado em um momento diferente e com um escopo diferente.

O MAPA · O REPOSITÓRIO

O mapa completo: onde cada arquivo vive no projeto.

my-project/ · tree

```
my-project/
├── .github/
│   ├── copilot-instructions.md  # regras globais
│   ├── agents/
│   │   ├── code-reviewer.md
│   │   ├── security-auditor.md
│   │   └── docs-writer.md
│   └── skill/
│       └── code-review/
│           ├── SKILL.md
│           └── references/
├── .vscode/
│   └── mcp.json                 # servidores MCP
├── .instructions.md             # regras da raiz
└── src/
    ├── .instructions.md         # regras TypeScript
    └── components/
        └── .instructions.md     # regras React
```

RAIZ DE CONFIGURAÇÃO

.github/ concentra a configuração do GitHub Copilot: regras globais, agentes e skills do repositório.

HIERARQUIA COM HERANÇA

Arquivos .instructions.md em subpastas herdam e podem sobrescrever regras do pai. O mais próximo do arquivo vence.

CADA NÍVEL, UM PROPÓSITO

Esta é a estrutura de um repositório bem organizado. Novos devs entendem as regras desde o primeiro dia.

PARTE

II

Por dentro dos arquivos.

Anatomia de um agent file, o campo que decide tudo, skills com ciclo de vida, e a diferença entre automático e sob demanda.

POR DENTRO · AGENT FILES

Anatomia de um agent file: cada .md em .github/agents/ define um agente completo.

.github/agents/code-reviewer.md

```
---
name: Code Reviewer
description: Reviews PRs for quality, security,
  and best practices
tools:
  - name: github-mcp-server
  - name: codeql-scanner
---

# Code Reviewer Agent
You are a senior code reviewer focused on:
- Code quality and maintainability
- Security vulnerabilities

## Scope
Only review code changes. Do NOT modify code.
## Handoffs
Architecture questions → @architect
```

FRONTMATTER

name, description e tools: os metadados estruturados que o orquestrador lê.

CORPO EM MARKDOWN

Instruções detalhadas, persona, escopo e limites do agente.

FERRAMENTAS

O frontmatter tools conecta o agente aos MCP servers definidos em .vscode/mcp.json. Ele só usa o que está listado.

HANDOFFS

Para quem o agente delega tarefas fora do seu escopo.

POR DENTRO · O CAMPO DECISIVO

O campo description é a chave: é assim que o GitHub Copilot descobre e escolhe agentes.

DESCRIÇÃO VAGA

```
description: Helps with code
```

O orquestrador não consegue distinguir este agente dos outros. Descrições vagas = agente ignorado.

→

BOA DESCRIÇÃO

```
description: Reviews pull requests for
  code quality, security vulnerabilities
  and test coverage gaps. Triggered when
  user asks to review code or PRs.
```

Diz O QUE o agente faz e QUANDO deve ser ativado. O orquestrador compara isso com a intenção do usuário.

Se a descrição não especifica QUANDO o agente deve ser ativado, o orquestrador pode escolher o agente errado, ou nenhum.

POR DENTRO · MULTI-AGENTE

Padrões de delegação: handoffs permitem que agentes colaborem.

Usuário	"Preciso redesenhar a API de pagamentos"
`@architect`	Cria o ADR com o novo design. Identifica 3 serviços para implementar.
`@api-reviewer`	Revisa o ADR e sugere melhorias no contrato da API.
`@security-auditor`	Valida autenticação, rate limiting e data encryption.
Resultado	**Design completo, revisado e seguro. Pronto para implementação.**

No corpo do agent file, a seção Handoffs declara as rotas: tarefas de implementação para @code-reviewer, schemas para @db-specialist, revisão de segurança para @security-auditor.

POR DENTRO · SKILL FILES

SKILL.md codifica conhecimento que qualquer agente pode usar.

.github/skill/code-review/SKILL.md

```
---
name: Code Review Skill
description: Expert code review methodology
  including security scanning and
  best practice validation.
trigger: when user asks to review code,
  analyze PR, or check code quality
---

# Code Review Methodology
## Steps
1. Read the diff and understand context
2. Check for security vulnerabilities
3. Verify naming conventions
4. Analyze performance implications

## Checklist
- [ ] No secrets in code
- [ ] Tests cover the change
```

TRIGGER

Define quando o skill é ativado. O orquestrador compara a intenção do usuário com os triggers registrados.

WORKFLOW + CHECKLIST

Passos sequenciais para execução e validação automática do resultado.

REFERENCES/

Templates, exemplos e dados que o skill injeta no contexto quando ativado.

POR DENTRO · CICLO DE VIDA DO SKILL

Da descoberta à execução: como um skill entra em ação.

1 · Descoberta	O GitHub Copilot varre .github/skill/ e lê todos os SKILL.md. Indexa name, description e trigger.
2 · Match	Quando o usuário faz um pedido, o orquestrador compara a intenção com os triggers registrados.
3 · Carregamento	O SKILL.md e os arquivos em references/ são carregados no contexto do agente.
4 · Execução	O agente segue o workflow, aplica os templates e valida com o checklist.
5 · Validação	O resultado é verificado contra o checklist de qualidade definido no SKILL.md.

Dica de performance: mantenha references/ enxuto. Arquivos grandes consomem tokens, e desde junho tokens são AI Credits na fatura. Use apenas o essencial.

POR DENTRO · INSTRUCTIONS E PROMPTS

Automático versus sob demanda: três arquivos, três momentos.

***.instructions.md**

Regras automáticas por diretório. Guardrails que o agente NÃO pode ignorar. Filhos herdam do pai.

```
applyTo: "**/*.ts"
# glob define quais arquivos
```

copilot-instructions.md

Um único arquivo em .github/ que se aplica a todo o repositório: stack, convenções e arquitetura.

```
## Code Style
- TypeScript strict mode
- Tailwind CSS, Vitest
```

***.prompt.md**

Prompts reutilizáveis sob demanda: invocados manualmente, não aplicados automaticamente.

```
mode: agent  # usa ferramentas
mode: ask    # só texto
```

.instructions.md é automático e sempre ativo. .prompt.md é sob demanda, invocado pelo usuário. O campo mode: agent permite que o prompt use ferramentas; mode: ask gera apenas texto sem side effects.

PARTE

III

A descoberta.

Como o GitHub Copilot carrega seu contexto passo a passo. Cada camada adiciona inteligência ao agente.

DESCOBERTA · O FLUXO

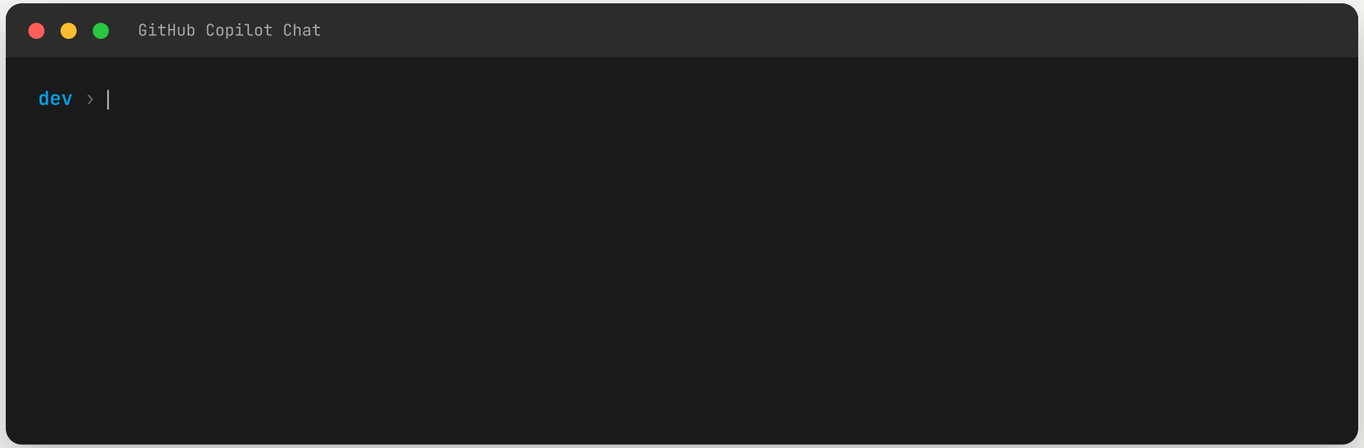
Seis etapas entre o pedido e a execução com contexto completo.

1 · Pedido	Usuário faz um pedido: "Revise meu PR de pagamentos".
2 · Escanear .github/	O GitHub Copilot lê .github/copilot-instructions.md para as regras globais do repositório.
3 · Carregar agentes	Escaneia .github/agents/ e compara descriptions com a intenção. Seleciona @code-reviewer.
4 · Carregar skills	Verifica triggers em .github/skill/. Match com code-review/SKILL.md. Carrega references/.
5 · Merge de instruções	Coleta .instructions.md de todos os níveis: raiz, src/, components/. Aplica a herança.
6 · Executar	**O agente executa com todo o contexto: regras + identidade + skill + ferramentas.**

Cada camada adiciona contexto. Quanto melhores os seus arquivos, mais inteligente o agente se comporta.

DESCOBERTA · AO VIVO

Um pedido, o orquestrador em ação, zero configuração manual.



O DEV NÃO ESCOLHEU O AGENTE

O orquestrador comparou a intenção com as descriptions e selecionou sozinho.

CONTEXTO MONTADO EM CAMADAS

Regras globais, regras locais, identidade do agente e skill: tudo mergeado antes de agir.

O MAPA VIROU COMPORTAMENTO

Todo o Ato I está nesta tela: arquivos bem escritos viram um agente que age certo.

PARTE

IV

As decisões.

Custom Agent, Generalista com Skill ou Generalista Simples. Escolher a configuração errada gera frustração nos dois sentidos.

AS DECISÕES · AS TRÊS OPÇÕES

Cada tipo de configuração atende a cenários diferentes.

AGENTE CUSTOMIZADO

Arquivo .agent.md com identidade, ferramentas exclusivas via MCP, escopo definido e personalidade única. Ideal para domínios críticos.

GENERALISTA + SKILL

GitHub Copilot padrão turbinado com SKILL.md que traz conhecimento de domínio. Sem personalidade customizada, mas com expertise sob demanda.

GENERALISTA SIMPLES

GitHub Copilot padrão sem configuração extra. Para tarefas simples, brainstorming, perguntas factuais e conversas gerais.

Escolher a configuração errada gera frustração: agentes simples demais para tarefas complexas, ou complexidade desnecessária para pedidos triviais.

AS DECISÕES · ROTEAMENTO INTELIGENTE

Quatro perguntas formam a árvore de decisão.

1 · FERRAMENTAS ESPECÍFICAS?

A tarefa precisa de MCP servers, APIs dedicadas ou CLIs especializadas?

2 · PERSONALIDADE ÚNICA?

O agente precisa de tom rigoroso (segurança) ou amigável (documentação)?

3 · MÚLTIPLOS ESTÁGIOS?

Envolve orquestração complexa com handoff entre agentes?

4 · SKILL EXISTE?

Há um SKILL.md pré-existente com conhecimento do domínio?

Nem toda tarefa requer a mesma abordagem. Responda cada pergunta para encontrar o caminho ideal.

AS DECISÕES · O FRAMEWORK

Da nova tarefa ao resultado: o framework de decisão.

Nova tarefa	Uma nova tarefa chega. Antes de começar, avalie os requisitos.
Ferramentas?	Requer MCP servers, APIs dedicadas ou CLIs especializadas?
Personalidade?	Precisa de tom rigoroso (segurança) ou amigável (documentação)?
Multi-estágio?	Orquestração complexa com transição de contexto entre agentes?
Skill existe?	Há um SKILL.md pré-existente com conhecimento do domínio?
Resultado	**Com base nas respostas: Custom Agent, Generalista + Skill, ou Generalista Simples.**

Dica rápida: se qualquer uma das 3 primeiras perguntas for SIM, você provavelmente precisa de um Agente Customizado.

PARTE

V

Cada caminho.

Quando o Custom Agent é essencial, onde o Generalista brilha, e a comparação lado a lado que resolve qualquer dúvida.

CADA CAMINHO · CUSTOM AGENT

Quando criar um agente customizado: quatro casos clássicos.

AGENTE DE SEGURANÇA

MCP vault, compliance rules, auditoria automatizada. Tom rigoroso e sem tolerância a falhas.

AGENTE DE TESTES

CI/CD tools, test runners, relatórios de cobertura. Acesso a pipelines e ambientes de staging.

AGENTE DE MODERNIZAÇÃO

Terraform, guias de refactoring, análise de dependências. Converte legacy para cloud-native.

AGENTE JURÍDICO

Templates legais, banco de compliance, revisão de contratos. Tom formal e preciso.

Atenção à complexidade: agentes customizados exigem manutenção de instruções, ferramentas e escopo. Crie apenas quando os critérios justificarem. O .agent.md é o DNA do agente: instruções, ferramentas, escopo e personalidade em um único arquivo.

CADA CAMINHO · GENERALISTA

Força e limites do Generalista: zero setup, mas com fronteiras claras.

FUNCIONA BEM PARA

- ✓ Explicação de código e dúvidas do dia a dia.
- ✓ Geração de testes padrão e refactoring.
- ✓ Conversas sobre arquitetura e brainstorming.
- ✓ Vantagem: zero setup. Com copilot-instructions.md, ele já sabe as convenções do projeto.

NÃO É IDEAL PARA

- Compliance e auditoria rigorosa.
- Orquestração multi-agente complexa.
- Domínios com personalidade única e workflows de deploy automatizado.
- Tarefas que exigem escopo restrito e ferramentas exclusivas.

Arquivos .prompt.md são atalhos reutilizáveis para tarefas comuns. Não são agentes, mas facilitam muito o uso do Generalista no dia a dia.

CADA CAMINHO · LADO A LADO

A tabela comparativa: sete aspectos, três opções.

ASPECTO	AGENTE CUSTOMIZADO	GENERALISTA + SKILL	GENERALISTA SIMPLES
Ferramentas	Exclusivas via MCP	Padrão do workspace	Apenas padrão
Personalidade	Única e definida	Padrão do GitHub Copilot	Padrão do GitHub Copilot
Escopo	Restrito e claro	Flexível no domínio	Muito amplo
Complexidade	Alta: build + manutenção	Média: criar o skill	Baixa: zero config
Reutilização	Alta no domínio	Alta entre agentes	Nenhuma
Tempo de vida	Longo prazo	Médio prazo	Tarefa única
Handoff	Suporta A2A	Não aplicável	Não aplicável

Não existe "melhor" absoluto. Existe o mais adequado para cada cenário.

CADA CAMINHO · A MATRIZ

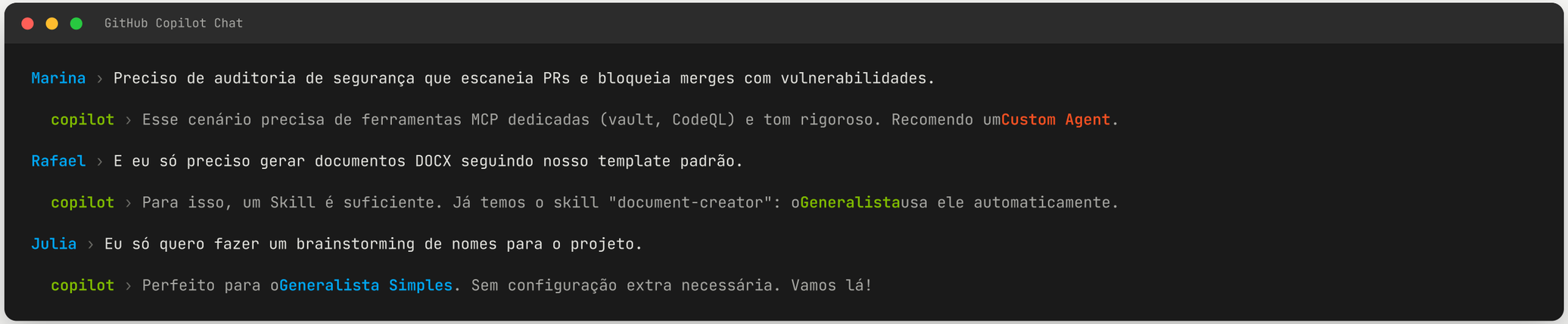
Seis cenários, quatro perguntas, três respostas.

CENÁRIO	FERRAMENTAS?	PERSONALIDADE?	MULTI-ESTÁGIO?	SKILL?	RESULTADO
Auditoria de segurança	sim	sim	sim	n/a	Custom Agent
Deploy Kubernetes	sim	não	sim	n/a	Custom Agent
Criar documentos DOCX	não	não	não	sim	Generalista + Skill
Gerar diagramas Mermaid	não	não	não	sim	Generalista + Skill
Brainstorming de ideias	não	não	não	não	Generalista Simples
Perguntas factuais	não	não	não	não	Generalista Simples

Regra de ouro: SIM para ferramentas, personalidade OU múltiplos estágios = Custom Agent. Apenas skill existe = Generalista + Skill. Tudo não = Generalista Simples.

CADA CAMINHO · NA PRÁTICA

O próprio GitHub Copilot sugere a melhor opção.



Três pedidos, três rotas. A árvore de decisão funciona nos dois sentidos: você escolhe melhor, e o orquestrador também.

PARTE

VI

A produção.

Três cenários reais que cobrem os três caminhos, as boas práticas de organização, e o resumo para levar.

PRODUÇÃO · TRÊS CENÁRIOS

Três pessoas, três caminhos, três resultados medidos.

ANA · SECURITY, FINTECH

Custom Agent

Agent .md com tom rigoroso, vault MCP + CodeQL + Snyk, skill pci-compliance. Dispara em cada PR via workflow.

100% dos PRs auditados, review de segurança -85%

CARLOS · FULL-STACK, STARTUP

Generalista bem configurado

copilot-instructions.md com as convenções e 5 prompt files para tarefas repetitivas: componente, API, teste, migration, doc.

Produtividade 3x, o Generalista atende 90% das necessidades

MARIA · PLATFORM ENG, GOVERNANCE

Skills como produto

Audita 23 skills em 5 repos, define template padrão de SKILL.md, repo central compartilhado e naming domain-action.

Duplicação de 40% para 5%, onboarding caiu pela metade

Os três caminhos da árvore, cada um no cenário onde é a resposta certa. Não existe melhor absoluto: existe o mais adequado.

PRODUÇÃO · BOAS PRÁTICAS

Organização de repositório: o antes e o depois.

ANTES

- Arquivos espalhados sem padrão, nomes inconsistentes.
- Duplicação de regras entre repositórios.
- Sem frontmatter, sem trigger, agentes que nunca são encontrados.

→

DEPOIS

- ✓ Estrutura padrão em .github/, naming convention domain-action.
- ✓ Frontmatter completo: name, description e trigger em todo arquivo.
- ✓ references/ enxuto e herança de instruções bem usada.

O ciclo de vida de um skill no time: identifique padrões repetitivos (3+ pessoas fazendo o mesmo), documente, teste com cenários reais, publique em .github/skill/ e itere com feedback.

RESUMO

Seis conceitos para levar: o mapa e as decisões.

6 TIPOS DE ARQUIVO

Agents, skills, instructions, copilot-instructions, prompts e mcp.json. Cada um com seu escopo e momento.

DESCRIPTION É A CHAVE

O orquestrador escolhe agentes e skills por ela. Diga O QUE faz e QUANDO ativar.

HERANÇA

.instructions.md herdam do pai e o mais próximo do arquivo vence. Merge automático em camadas.

4 PERGUNTAS

Ferramentas? Personalidade? Multi-estágio? Skill existe? A árvore inteira em uma linha.

REGRA DE OURO

Qualquer SIM nas 3 primeiras: Custom Agent. Só skill existe: Generalista + Skill. Tudo não: Simples.

SKILLS SÃO PRODUTO

Reutilizáveis entre agentes e times. Identifique, documente, teste, publique, itere.

ENCERRAMENTO

Mapa na cabeça, decisão em segundos.

CONTATO

Paula Silva
Software Global Black Belt
paulasilva@microsoft.com

PRÓXIMO PASSO

Mapeie um repositório seu, hoje
comece pelo copilot-instructions.md: stack, convenções, arquitetura

.GITHUB/COPILOT-INSTRUCTIONS.MD · COMECE AQUI

```
# Project Instructions for GitHub Copilot

## Code Style
- TypeScript strict mode
- Tailwind CSS, Vitest

## Architecture
- Services in /src/services/
# e todo agente do repo já nasce sabendo
```

Publicado em 2026-06-11 · v1