GITHUB COPILOT AGENTIC FRAMEWORK

El stack agéntico: el mapa y las decisiones.

Acto I: la arquitectura de archivos, dónde viven agents, skills, instructions y prompts, y cómo GitHub Copilot los descubre. Acto II: el árbol de decisión para elegir entre Agente Personalizado, Generalista con Skill y Generalista Simple en cualquier tarea.

AUTORA
Paula Silva

FUNCIÓN
Software Global Black Belt

DURACIÓN
75 a 90 minutos

FECHA
2026-06-11

AGENDA

Dos actos, seis partes.

PARTE I	El mapa, por qué importa la arquitectura de archivos y los 6 tipos que forman el ecosistema
PARTE II	Por dentro de los archivos, anatomía de agents, skills, instructions y prompts
PARTE III	El descubrimiento, cómo GitHub Copilot carga tu contexto paso a paso
PARTE IV	Las decisiones, las tres opciones y las 4 preguntas del enrutamiento inteligente
PARTE V	Cada camino a fondo, comparación, matriz de decisión y GitHub Copilot sugiriendo la opción correcta
PARTE VI	Producción, tres escenarios reales, buenas prácticas de organización y el resumen

PARTE

I

El mapa.

Cada archivo es una capa de contexto para el agente. Sin arquitectura, los agentes operan sin reglas, sin identidad y sin expertise.

EL MAPA · POR QUÉ IMPORTA

Cada archivo es una capa de contexto para el agente.

REGLAS AUTOMÁTICAS

.instructions.md define guardrails que el agente NUNCA puede ignorar. Se aplican automáticamente.

IDENTIDAD DEL AGENTE

.github/agents/*.md define QUIÉN es el agente, su alcance, herramientas y a quién delega.

CONOCIMIENTO ESPECIALIZADO

SKILL.md codifica expertise del dominio: plantillas, workflows, checklists y ejemplos.

PROMPTS REUTILIZABLES

.prompt.md almacena prompts reutilizables que pueden invocarse bajo demanda.

Sin arquitectura de archivos, los agentes operan sin reglas, sin identidad y sin expertise. El resultado es un comportamiento inconsistente que nadie puede auditar.

EL MAPA · VISIÓN GENERAL

Los 6 tipos de archivo del ecosistema, cada uno con un papel distinto.

ARCHIVO	UBICACIÓN	ALCANCE	PROPÓSITO
agents/*.md	.github/agents/	Agente específico	Identidad, herramientas, handoffs
SKILL.md	.github/skill/*/	Skill específico	Expertise, plantillas, workflows
*.instructions.md	Cualquier directorio	Directorio + hijos	Reglas automáticas, guardrails
copilot-instructions.md	.github/	Repositorio entero	Instrucciones globales del repo
*.prompt.md	Cualquier directorio	Bajo demanda	Prompts reutilizables
mcp.json	.vscode/	Workspace	Servidores MCP, herramientas

Estos 6 tipos forman el ecosistema completo. Cada uno se carga en un momento diferente y con un alcance diferente.

EL MAPA · EL REPOSITORIO

El mapa completo: dónde vive cada archivo en el proyecto.

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
```

RAÍZ DE CONFIGURACIÓN

.github/ concentra la configuración de GitHub Copilot: reglas globales, agentes y los skills del repositorio.

JERARQUÍA CON HERENCIA

Los .instructions.md en subcarpetas heredan y pueden sobrescribir reglas del padre. El más cercano al archivo gana.

CADA NIVEL, UN PROPÓSITO

Esta es la estructura de un repositorio bien organizado. Los nuevos devs entienden las reglas desde el primer día.

PARTE

II

Por dentro de los archivos.

Anatomía de un agent file, el campo que lo decide todo, skills con ciclo de vida, y la diferencia entre automático y bajo demanda.

POR DENTRO · AGENT FILES

Anatomía de un agent file: cada .md en .github/agents/ define un 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 y tools: los metadatos estructurados que lee el orquestador.

CUERPO EN MARKDOWN

Instrucciones detalladas, persona, alcance y límites del agente.

HERRAMIENTAS

El frontmatter tools conecta al agente con los MCP servers definidos en .vscode/mcp.json. Solo usa lo que está listado.

HANDOFFS

A quién delega el agente las tareas fuera de su alcance.

POR DENTRO · EL CAMPO DECISIVO

El campo description es la clave: así GitHub Copilot descubre y elige agentes.

DESCRIPCIÓN VAGA

```
description: Helps with code
```

El orquestador no puede distinguir este agente de los demás. Descripciones vagas = agente ignorado.

→

BUENA DESCRIPCIÓN

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

Dice QUÉ hace el agente y CUÁNDO debe activarse. El orquestador compara esto con la intención del usuario.

Si la descripción no especifica CUÁNDO debe activarse el agente, el orquestador puede elegir el agente equivocado, o ninguno.

POR DENTRO · MULTIAGENTE

Patrones de delegación: los handoffs permiten que los agentes colaboren.

Usuario	"Necesito rediseñar la API de pagos"
`@architect`	Crea el ADR con el nuevo diseño. Identifica 3 servicios para implementar.
`@api-reviewer`	Revisa el ADR y sugiere mejoras en el contrato de la API.
`@security-auditor`	Valida autenticación, rate limiting y data encryption.
Resultado	**Diseño completo, revisado y seguro. Listo para implementación.**

En el cuerpo del agent file, la sección Handoffs declara las rutas: tareas de implementación a @code-reviewer, schemas a @db-specialist, revisión de seguridad a @security-auditor.

POR DENTRO · SKILL FILES

SKILL.md codifica conocimiento que cualquier agente puede 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 cuándo se activa el skill. El orquestador compara la intención del usuario con los triggers registrados.

WORKFLOW + CHECKLIST

Pasos secuenciales para la ejecución y validación automática del resultado.

REFERENCES/

Plantillas, ejemplos y datos que el skill inyecta en el contexto cuando se activa.

POR DENTRO · CICLO DE VIDA DEL SKILL

Del descubrimiento a la ejecución: cómo un skill entra en acción.

1 · Descubrimiento	GitHub Copilot recorre .github/skill/ y lee todos los SKILL.md. Indexa name, description y trigger.
2 · Match	Cuando el usuario hace una petición, el orquestador compara la intención con los triggers registrados.
3 · Carga	El SKILL.md y los archivos en references/ se cargan en el contexto del agente.
4 · Ejecución	El agente sigue el workflow, aplica las plantillas y valida con el checklist.
5 · Validación	El resultado se verifica contra el checklist de calidad definido en el SKILL.md.

Consejo de performance: mantén references/ ligero. Los archivos grandes consumen tokens, y desde junio los tokens son AI Credits en la factura. Usa solo lo esencial.

POR DENTRO · INSTRUCTIONS Y PROMPTS

Automático versus bajo demanda: tres archivos, tres momentos.

***.instructions.md**

Reglas automáticas por directorio. Guardrails que el agente NO puede ignorar. Los hijos heredan del padre.

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

copilot-instructions.md

Un único archivo en .github/ que se aplica a todo el repositorio: stack, convenciones y arquitectura.

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

***.prompt.md**

Prompts reutilizables bajo demanda: invocados manualmente, no aplicados automáticamente.

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

.instructions.md es automático y siempre activo. .prompt.md es bajo demanda, invocado por el usuario. El campo mode: agent permite que el prompt use herramientas; mode: ask genera solo texto sin side effects.

PARTE

III

El descubrimiento.

Cómo GitHub Copilot carga tu contexto paso a paso. Cada capa añade inteligencia al agente.

DESCUBRIMIENTO · EL FLUJO

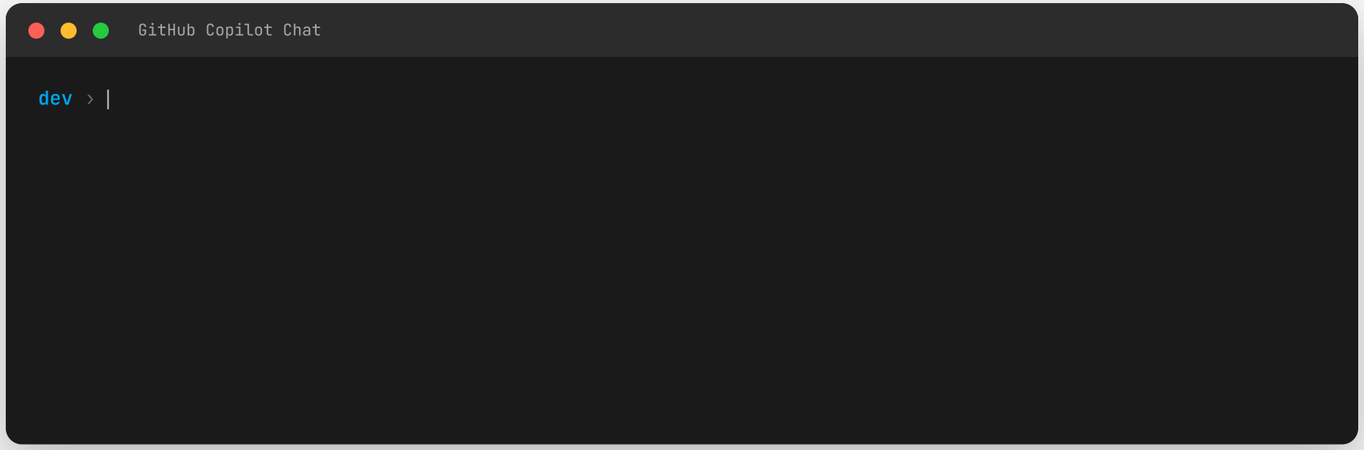
Seis etapas entre la petición y la ejecución con contexto completo.

1 · Petición	El usuario hace una petición: "Revisa mi PR de pagos".
2 · Escanear .github/	GitHub Copilot lee .github/copilot-instructions.md para las reglas globales del repositorio.
3 · Cargar agentes	Escanea .github/agents/ y compara las descriptions con la intención. Selecciona @code-reviewer.
4 · Cargar skills	Verifica triggers en .github/skill/. Match con code-review/SKILL.md. Carga references/.
5 · Merge de instrucciones	Recoge .instructions.md de todos los niveles: raíz, src/, components/. Aplica la herencia.
6 · Ejecutar	**El agente ejecuta con todo el contexto: reglas + identidad + skill + herramientas.**

Cada capa añade contexto. Cuanto mejores sean tus archivos, más inteligente se comporta el agente.

DESCUBRIMIENTO · EN VIVO

Una petición, el orquestador en acción, cero configuración manual.



EL DEV NO ELIGIÓ AL AGENTE

El orquestador comparó la intención con las descriptions y seleccionó por su cuenta.

CONTEXTO MONTADO EN CAPAS

Reglas globales, reglas locales, identidad del agente y skill: todo mergeado antes de actuar.

EL MAPA SE VOLVIÓ COMPORTAMIENTO

Todo el Acto I está en esta pantalla: archivos bien escritos se vuelven un agente que actúa bien.

PARTE

IV

Las decisiones.

Custom Agent, Generalista con Skill o Generalista Simple. Elegir la configuración equivocada genera frustración en ambos sentidos.

LAS DECISIONES · LAS TRES OPCIONES

Cada tipo de configuración atiende escenarios diferentes.

AGENTE PERSONALIZADO

Archivo .agent.md con identidad, herramientas exclusivas vía MCP, alcance definido y personalidad única. Ideal para dominios críticos.

GENERALISTA + SKILL

GitHub Copilot estándar potenciado con un SKILL.md que aporta conocimiento de dominio. Sin personalidad personalizada, pero con expertise bajo demanda.

GENERALISTA SIMPLE

GitHub Copilot estándar sin configuración extra. Para tareas simples, brainstorming, preguntas factuales y conversaciones generales.

Elegir la configuración equivocada genera frustración: agentes demasiado simples para tareas complejas, o complejidad innecesaria para peticiones triviales.

LAS DECISIONES · ENRUTAMIENTO INTELIGENTE

Cuatro preguntas forman el árbol de decisión.

1 · ¿HERRAMIENTAS ESPECÍFICAS?

¿La tarea necesita MCP servers, APIs dedicadas o CLIs especializadas?

2 · ¿PERSONALIDAD ÚNICA?

¿El agente necesita un tono riguroso (seguridad) o amigable (documentación)?

3 · ¿MÚLTIPLES ETAPAS?

¿Involucra orquestación compleja con handoff entre agentes?

4 · ¿EXISTE UN SKILL?

¿Hay un SKILL.md preexistente con conocimiento del dominio?

No toda tarea requiere el mismo enfoque. Responde cada pregunta para encontrar el camino ideal.

LAS DECISIONES · EL FRAMEWORK

De la nueva tarea al resultado: el framework de decisión.

Nueva tarea	Llega una nueva tarea. Antes de empezar, evalúa los requisitos.
¿Herramientas?	¿Requiere MCP servers, APIs dedicadas o CLIs especializadas?
¿Personalidad?	¿Necesita un tono riguroso (seguridad) o amigable (documentación)?
¿Multietapa?	¿Orquestación compleja con transición de contexto entre agentes?
¿Existe un skill?	¿Hay un SKILL.md preexistente con conocimiento del dominio?
Resultado	**Según las respuestas: Custom Agent, Generalista + Skill, o Generalista Simple.**

Consejo rápido: si cualquiera de las 3 primeras preguntas es SÍ, probablemente necesitas un Agente Personalizado.

PARTE

V

Cada camino.

Cuándo el Custom Agent es esencial, dónde brilla el Generalista, y la comparación lado a lado que resuelve cualquier duda.

CADA CAMINO · CUSTOM AGENT

Cuándo crear un agente personalizado: cuatro casos clásicos.

AGENTE DE SEGURIDAD

MCP vault, compliance rules, auditoría automatizada. Tono riguroso y sin tolerancia a fallos.

AGENTE DE PRUEBAS

Herramientas CI/CD, test runners, reportes de cobertura. Acceso a pipelines y entornos de staging.

AGENTE DE MODERNIZACIÓN

Terraform, guías de refactoring, análisis de dependencias. Convierte legacy a cloud-native.

AGENTE JURÍDICO

Plantillas legales, base de compliance, revisión de contratos. Tono formal y preciso.

Atención a la complejidad: los agentes personalizados exigen mantenimiento de instrucciones, herramientas y alcance. Créalos solo cuando los criterios lo justifiquen. El .agent.md es el ADN del agente: instrucciones, herramientas, alcance y personalidad en un único archivo.

CADA CAMINO · GENERALISTA

Fuerza y límites del Generalista: cero setup, pero con fronteras claras.

FUNCIONA BIEN PARA

- ✓ Explicación de código y dudas del día a día.
- ✓ Generación de pruebas estándar y refactoring.
- ✓ Conversaciones sobre arquitectura y brainstorming.
- ✓ Ventaja: cero setup. Con copilot-instructions.md ya conoce las convenciones del proyecto.

NO ES IDEAL PARA

- Compliance y auditoría rigurosa.
- Orquestación multiagente compleja.
- Dominios con personalidad única y workflows de deploy automatizado.
- Tareas que exigen alcance restringido y herramientas exclusivas.

Los archivos .prompt.md son atajos reutilizables para tareas comunes. No son agentes, pero facilitan mucho el uso del Generalista en el día a día.

CADA CAMINO · LADO A LADO

La tabla comparativa: siete aspectos, tres opciones.

ASPECTO	AGENTE PERSONALIZADO	GENERALISTA + SKILL	GENERALISTA SIMPLE
Herramientas	Exclusivas vía MCP	Estándar del workspace	Solo estándar
Personalidad	Única y definida	Estándar de GitHub Copilot	Estándar de GitHub Copilot
Alcance	Restringido y claro	Flexible en el dominio	Muy amplio
Complejidad	Alta: build + mantenimiento	Media: crear el skill	Baja: cero config
Reutilización	Alta en el dominio	Alta entre agentes	Ninguna
Tiempo de vida	Largo plazo	Medio plazo	Tarea única
Handoff	Soporta A2A	No aplicable	No aplicable

No existe un "mejor" absoluto. Existe el más adecuado para cada escenario.

CADA CAMINO · LA MATRIZ

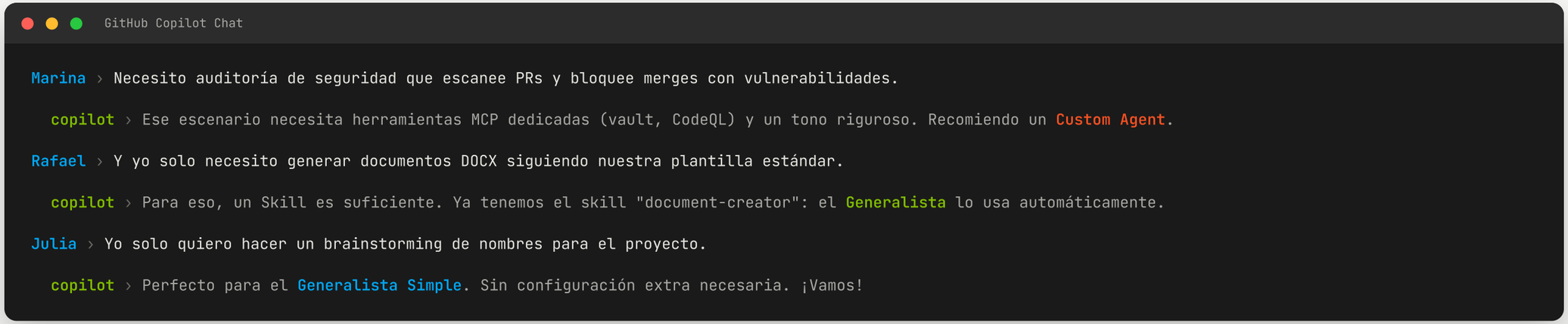
Seis escenarios, cuatro preguntas, tres respuestas.

ESCENARIO	¿HERRAMIENTAS?	¿PERSONALIDAD?	¿MULTIETAPA?	¿SKILL?	RESULTADO
Auditoría de seguridad	**sí**	**sí**	**sí**	n/a	`Custom Agent`
Deploy Kubernetes	**sí**	no	**sí**	n/a	`Custom Agent`
Crear documentos DOCX	no	no	no	**sí**	`Generalista + Skill`
Generar diagramas Mermaid	no	no	no	**sí**	`Generalista + Skill`
Brainstorming de ideas	no	no	no	no	`Generalista Simples`
Preguntas factuales	no	no	no	no	`Generalista Simples`

Regla de oro: SÍ para herramientas, personalidad O múltiples etapas = Custom Agent. Solo existe un skill = Generalista + Skill. Todo no = Generalista Simple.

CADA CAMINO · EN LA PRÁCTICA

El propio GitHub Copilot sugiere la mejor opción.



Tres peticiones, tres rutas. El árbol de decisión funciona en ambos sentidos: tú eliges mejor, y el orquestador también.

PARTE

VI

La producción.

Tres escenarios reales que cubren los tres caminos, las buenas prácticas de organización, y el resumen para llevar.

PRODUCCIÓN · TRES ESCENARIOS

Tres personas, tres caminos, tres resultados medidos.

ANA · SECURITY, FINTECH

Custom Agent

Agent .md con tono riguroso, vault MCP + CodeQL + Snyk, skill pci-compliance. Se dispara en cada PR vía workflow.

100% de los PRs auditados, revisión de seguridad -85%

CARLOS · FULL-STACK, STARTUP

Generalista bien configurado

copilot-instructions.md con las convenciones y 5 prompt files para tareas repetitivas: componente, API, prueba, migration, doc.

Productividad 3x, el Generalista cubre el 90% de sus necesidades

MARIA · PLATFORM ENG, GOVERNANCE

Skills como producto

Audita 23 skills en 5 repos, define una plantilla estándar de SKILL.md, un repo central compartido y naming domain-action.

Duplicación del 40% al 5%, onboarding reducido a la mitad

Los tres caminos del árbol, cada uno en el escenario donde es la respuesta correcta. No existe un mejor absoluto: existe el más adecuado.

PRODUCCIÓN · BUENAS PRÁCTICAS

Organización del repositorio: el antes y el después.

ANTES

- Archivos dispersos sin estándar, nombres inconsistentes.
- Reglas duplicadas entre repositorios.
- Sin frontmatter, sin trigger, agentes que nunca son encontrados.

→

DESPUÉS

- ✓ Estructura estándar en .github/, naming convention domain-action.
- ✓ Frontmatter completo: name, description y trigger en cada archivo.
- ✓ references/ ligero y herencia de instrucciones bien usada.

El ciclo de vida de un skill en el equipo: identifica patrones repetitivos (3+ personas haciendo lo mismo), documenta, prueba con escenarios reales, publica en .github/skill/ e itera con feedback.

RESUMEN

Seis conceptos para llevar: el mapa y las decisiones.

6 TIPOS DE ARCHIVO

Agents, skills, instructions, copilot-instructions, prompts y mcp.json. Cada uno con su alcance y su momento.

DESCRIPTION ES LA CLAVE

El orquestador elige agentes y skills por ella. Di QUÉ hace y CUÁNDO activarse.

HERENCIA

Los .instructions.md heredan del padre y el más cercano al archivo gana. Merge automático en capas.

4 PREGUNTAS

¿Herramientas? ¿Personalidad? ¿Multietapa? ¿Existe un skill? Todo el árbol en una línea.

REGLA DE ORO

Cualquier SÍ en las 3 primeras: Custom Agent. Solo existe un skill: Generalista + Skill. Todo no: Simple.

LOS SKILLS SON PRODUCTO

Reutilizables entre agentes y equipos. Identifica, documenta, prueba, publica, itera.

CIERRE

Mapa en la cabeza, decisión en segundos.

CONTACTO

Paula Silva
Software Global Black Belt
paulasilva@microsoft.com

PRÓXIMO PASO

Mapea un repositorio tuyo, hoy
empieza por copilot-instructions.md: stack, convenciones, arquitectura

.GITHUB/COPILOT-INSTRUCTIONS.MD · EMPIEZA AQUÍ

```
# Project Instructions for GitHub Copilot

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

## Architecture
- Services in /src/services/
# y todo agente del repo ya nace sabiéndolo
```

Publicado el 2026-06-11 · v1