MODERNIZACIÓN DE LEGADO

Modernización de legado con agentes en GitHub.

Cómo agentes especializados en GitHub Copilot ayudan a entender estates Natural, Adabas, COBOL y DB2, escribir specs auditables y construir el sistema nuevo. Con el caso real del SIFAP.

PRESENTA

Paula Silva

Software Global Black Belt · Microsoft

DURACIÓN

~2 horas

FECHA

11 jun 2026

AGENDA

Seis partes, un arco.

I	El problema, por qué la modernización tradicional dejó de cerrar
II	El giro agéntico, trabajar con agentes es diferente de usar IA
III	El legado invisible, con SIFAP como ejemplo concreto
IV	Los agentes en acción, del arqueólogo al agente en la nube
V	El sistema moderno, el legado rehecho y probado
VI	Cómo empezar el lunes por la mañana

PARTE

I

El problema.

Por qué la cuenta de la modernización tradicional dejó de cerrar.

EL COSTO INVISIBLE

Cada línea de código antiguo cobra intereses compuestos cada trimestre.

El sistema parece funcionar. El costo se esconde en operación, cumplimiento y velocidad. Cuantifiquémoslo.

75%

DEL PRESUPUESTO DE TI MANTENIENDO LO QUE YA EXISTE

10 a 15

AÑOS, EDAD MEDIA DEL CÓDIGO CRÍTICO EN ESTADOS HEREDADOS

5 a 7

AÑOS POR PROGRAMA DE MODERNIZACIÓN TRADICIONAL

70%

DE LOS PROYECTOS DE MODERNIZACIÓN NO LLEGAN AL FINAL

Fuentes de mercado, Gartner e IDC. No es catastrofismo, es la base que hace caro el statu quo.

LOS DOS CAMINOS CLÁSICOS

Reescritura desde cero y lift-and-shift tienen el mismo destino: pararse antes de llegar.

Reescritura Big Bang

equipo paralelo, stack nuevo

Rehacer el sistema desde cero mientras el legado sigue corriendo y cambiando.

- Cuando lo nuevo está listo, lo viejo ya evolucionó. No hay paridad.
- El conocimiento tácito nunca se escribió. Se va con quien se jubila.
- Años sin entregar valor. El proyecto muere antes del go-live.

Promete todo nuevo. Entrega atraso y riesgo.

Lift-and-shift

misma lógica, infra nueva

Mover el código tal cual a la nube, sin reescribir la lógica.

- La deuda técnica viaja con él. El legado ahora corre caro en la nube.
- Ninguna regla se entendió. La caja negra sigue negra.
- Modernizaste el hosting, no el sistema. El problema queda para después.

Promete rapidez. Entrega el mismo problema, con factura mayor.

YA PROBARON IA

La IA generativa de primera generación rompió tres barreras. Otras tres siguen en pie.

Chat, autocomplete y explain this code aceleran al individuo. Pero modernizar no es una tarea individual.

LO QUE LA IA TRADICIONAL RESOLVIÓ

Velocidad individual en tareas conocidas.

- Explica código antiguo en segundos.
- Genera boilerplate, tests simples, queries.
- Un dev senior se vuelve 30 a 40% más rápido en lo que ya domina.

≠

LO QUE NO RESOLVIÓ

Contexto, coordinación y continuidad.

- No conoce tu dominio hasta que le explicas todo, cada vez.
- No coordina a varias personas alrededor del mismo sistema.
- No tiene memoria entre sesiones. Cada conversación empieza de cero.

PARTE

II

El giro agéntico.

Trabajar con agentes es diferente de usar IA. La unidad de trabajo cambia.

TRES NIVELES DE ADOPCIÓN

La adopción de IA en desarrollo pasa por tres niveles distintos.

La mayoría de los equipos está estancada en el Nivel 1. El salto que importa va directo al Nivel 3.

NIVEL 1 · ASISTIDO · ×1.3

Chat, autocomplete, generación inline. La IA responde cuando se le pregunta, cada dev solo con su copiloto. Ganancia: velocidad individual. Límite: proceso y calidad siguen manuales.

NIVEL 2 · AUMENTADO · ×2.5

Modos Edits y Agent. La IA hace cambios multiarchivo, refactoriza, genera tests; el dev revisa y acepta. Ganancia: tareas enteras delegables. Límite: aún falta proceso y memoria.

NIVEL 3 · AGÉNTICO

Agentes con rol, herramientas y memoria, dirigidos por spec y gates. El equipo orquesta, valida y firma. Ganancia: el trabajo escala sin que el dev sea el cuello de botella.

QUÉ ES UN AGENTE, DE VERDAD

Agente se volvió palabra de moda. Aquí tiene una definición precisa.

Un LLM con rol definido, más herramientas, más memoria de sesión. Los agentes encadenan llamadas.

01 · ROL

Un system prompt enfocado. Eres un arqueólogo de legado, tu trabajo es extraer reglas de negocio del Natural con trazabilidad, usa esta plantilla.

02 · HERRAMIENTAS

Acciones que puede ejecutar. Leer y editar archivos, correr comandos, consultar MCP servers: GitHub, base de datos, documentación.

03 · MEMORIA

Estado entre turnos. Specs guardadas en archivo, decisiones registradas. El agente no reinicia de cero en cada conversación.

SPECS COMO UNIDAD DE TRABAJO

La spec se vuelve la unidad de trabajo. El código es la consecuencia, no el punto de partida.

El agente ejecuta lo que está escrito, no lo que imaginaste. Cuanto más formal el pedido, menos ambigüedad queda.

SIN SPEC

Refactoriza esto. Pedido vago: el agente adivina la arquitectura y el resultado no coincide con la regla del legado.

CON ALCANCE

Extrae el módulo X a su propia clase. Mejor, pero aún sin criterio de aceptación ni contrato de test.

SPEC EARS

Requisitos formales, criterios de aceptación, contratos de test, con el origen en el legado registrado. Una frase vaga no pasa.

SPEC EJECUTABLE

Con Spec-Kit, la spec dirige la ejecución por etapas, con gates. El agente solo implementa lo que la spec declara.

LA ASIGNACIÓN DE ESFUERZO CAMBIA

La spec dirigiendo agentes cambia quién hace qué dentro del equipo.

Equipo tradicional

esfuerzo en el teclado

La mayor parte del tiempo es escribir código a mano.

- 70% escribiendo código manualmente.
- 15% en reuniones para alinear quién hace qué.
- 10% revisando PRs con contexto pobre, 5% en docs que nadie lee.

Cuello de botella: el dev senior se vuelve el embudo de calidad.

Equipo agéntico

esfuerzo en el juicio

La mayor parte del tiempo es especificar, orquestar y revisar.

- Escribir specs claras y criterios de aceptación.
- Orquestar agentes y validar lo que proponen.
- Decidir las fronteras y firmar lo que entra.

El juicio humano escala mejor que la digitación.

LA TESIS

Quien moderniza el legado decide dónde entra el agente.

El equipo no escribe desde cero. El agente lee, indexa, propone. El humano revisa, valida, decide, firma. El agente no es oráculo, y el humano no es mecanógrafo. Es un emparejamiento de roles distintos sobre el mismo artefacto.

CUATRO AGENTES EN GITHUB COPILOT CHAT

@archaeologist lee el legado · @architect escribe la spec · @builder construye · @evolution operacionaliza. Cada uno con tools acotadas a su rol.

MÁS EL GITHUB COPILOT CODING AGENT EN LA NUBE

Recibe issues bien escritas por @evolution, ejecuta en background, abre PR draft. El humano revisa antes de cualquier merge.

PARTE

III

El legado invisible.

Hay mainframe corriendo cosas serias debajo de más sistemas de los que contamos. Y el talento que entiende se está jubilando.

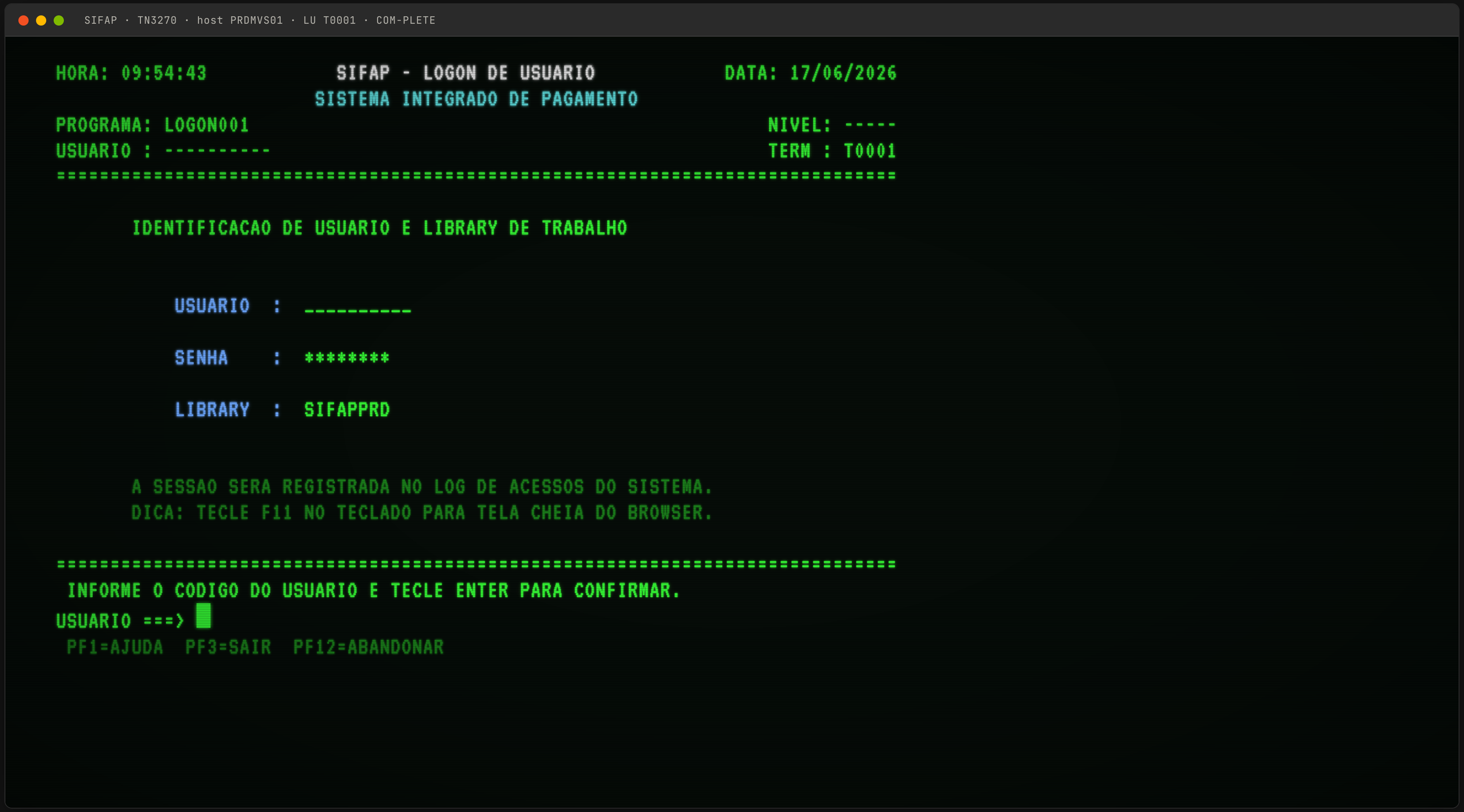
UN SISTEMA, UN EJEMPLO

SIFAP. Sistema social federal. 29 años en producción. Dos desarrolladores Natural activos.

Sistema brasileño de fiscalización y administración de pagos. 3,2 millones de beneficiarios activos, R$ 4,2 mil millones al año. 15 programas Natural, 4 DDMs Adabas. En producción desde 1996. Los dos desarrolladores Natural activos del equipo se jubilan en esta década.

~3,2 M	R$ 4,2 B	29 años	2
BENEFICIARIOS ACTIVOS	MOVIMIENTO ANUAL	EDAD DEL CÓDIGO	DEVS NATURAL ACTIVOS

Es un ejemplo. Puede ser tu sistema de facturación, tu core bancario, tu plataforma de RH. El perfil se repite en cualquier estate de esa generación.


SIFAP · TN3270 · host PRDMVS01 · LU T0001 · COM-PLETE
HORA: 09:54:43
SIFAP - LOGON DE USUARIO
DATA: 17/06/2026
SISTEMA INTEGRADO DE PAGAMENTO
PROGRAMA: LOGON001
NIVEL: -----
USUARIO : ----------
TERM : T0001
IDENTIFICACAO DE USUARIO E LIBRARY DE TRABALHO
USUARIO : _________
SENHA : ********
LIBRARY : SIFAPPRD
A SESSAO SERA REGISTRADA NO LOG DE ACESSOS DO SISTEMA.
DICA: TECLE F11 NO TECLADO PARA TELA CHEIA DO BROWSER.
INFORME O CODIGO DO USUARIO E TECLE ENTER PARA CONFIRMAR.
USUARIO ===>
PF1=AJUDA PF3=SAIR PF12=ABANDONAR

DEMO 1 · LO QUE ESTO REVELA

No se puede saber qué hace CRPGM042 solo mirando la pantalla.

ANTES

Pantalla 3270: CRPGM042 al pie, opaco

DESPUÉS

Regla documentada y trazable

Si esa regla se reescribe mal en Java, a Maria se le paga mal. Por eso la arqueología viene antes que la construcción.

PARTE

IV

Cuatro agentes en acción.

Cuatro agentes en GitHub Copilot Chat, más el GitHub Copilot Coding Agent en la nube. Cada uno con su rol.

@archaeologist

Lee el legado · solo lectura

El @archaeologist lee el legado y se niega a inventar.

QUÉ OBSERVAR

01

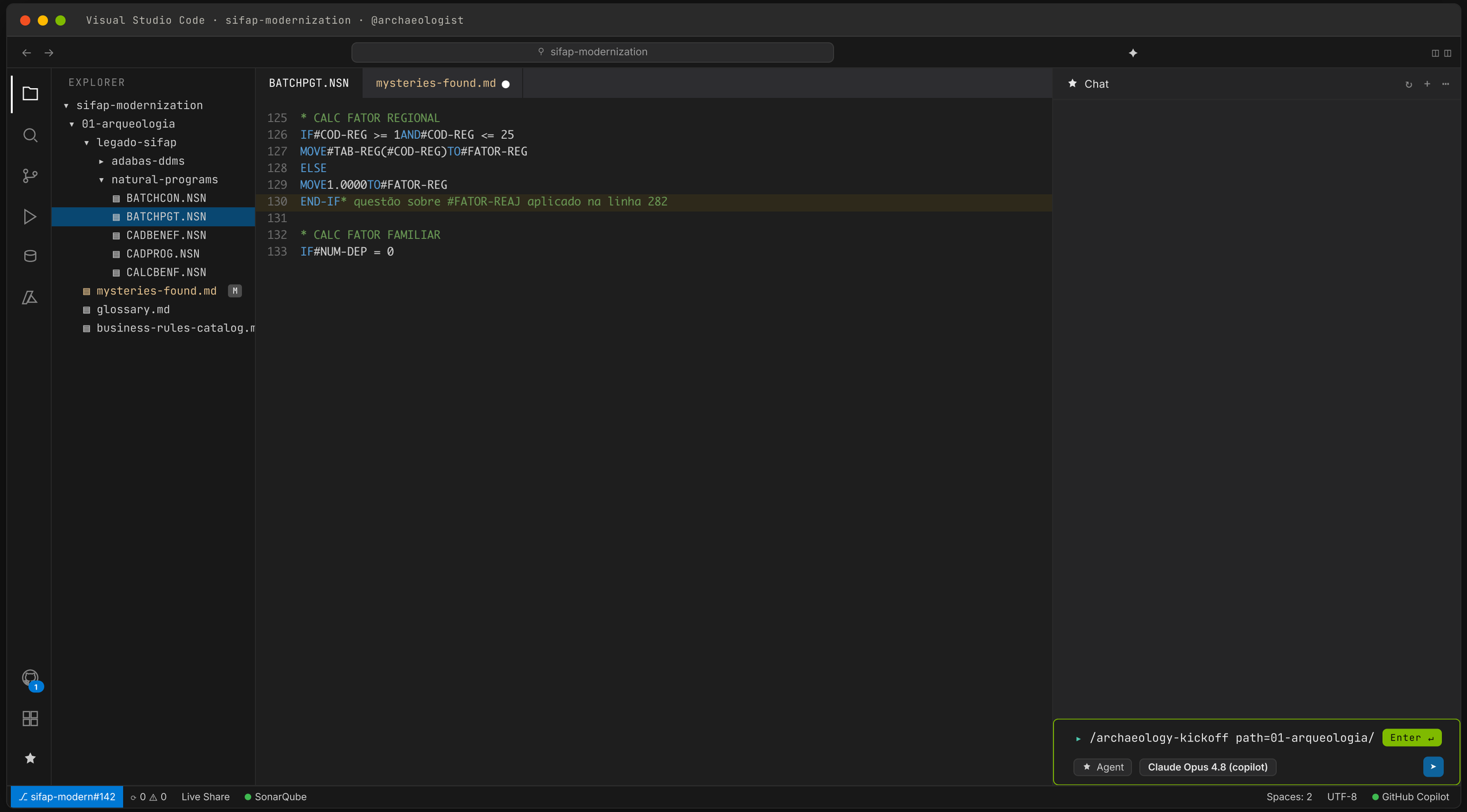
Read-only. No modifica una sola línea del legado.

02

Cuando no sabe, registra un misterio y sigue. Descubrimiento por encima de revelación.

03

Observe: pregunta de vuelta antes de responder, y abre CADPROG.NSN en la línea 81.


Visual Studio Code · sifap-modernization · @archaeologist
sifap-modernization
EXPLORER
sifap-modernization
01-arqueologia
legado-sifap
adabas-ddms
natural-programs
BATCHCON.NSN
BATCHPGT.NSN
CADBENEF.NSN
CADPROG.NSN
CALCBENF.NSN
mysteries-found.md
M
glossary.md
business-rules-catalog.m
BATCHPGT.NSN
mysteries-found.md
125 * CALC FATOR REGIONAL
126 IF#COD-REG >= 1AND#COD-REG <= 25
127 MOVE#TAB-REG(#COD-REG)TO#FATOR-REG
128 ELSE
129 MOVE1.0000TO#FATOR-REG
130 END-IF* questão sobre #FATOR-REAJ aplicado na linha 282
131
132 * CALC FATOR FAMILIAR
133 IF#NUM-DEP = 0
Chat
/archaeology-kickoff path=01-arqueologia/
Enter
Agent
Claude Opus 4.8 (copilot)
sifap-modern#142
0 0
Live Share
SonarQube
Spaces: 2
UTF-8
GitHub Copilot

LOS MISTERIOS QUE EL AGENTE EXPONE

Cuatro horas en SIFAP. 187 reglas catalogadas. 17 misterios oficiales.

187
REGLAS DE NEGOCIO CATALOGADAS

17
MISTERIOS OFICIALES REGISTRADOS

4 h
READ-ONLY, CERO CAMBIOS EN EL LEGADO

MYS-003 · Fator-K

Constante mágica 0.347215 sin origen legal en CADPROG.NSN. Probable conversión Cruzeiro Real → Real (1994). VLR-BASE ya viene multiplicado por K. Migrar mal paga dos veces.

MYS-007 · CPF 00000000000

Institucionalizado en el DDM como feature. Backdoor de registro fantasma, sin trazabilidad documentada.

MYS-008 · Región 99

COD-REGIAO=99 (INTERNACIONAL/DIPLOMÁTICO) salta todas las validaciones en VALELEG.NSN. Toda una clase nunca pasó la trilha.

MYS-010 · RELAUDIT oculta EX

Filtra silenciosamente ACAO='EX' antes de imprimir auditoría. Viola RN-011. Compliance rojo.

ETAPA 1 · /EXTRACT-BUSINESS-RULES · GENERADO 2026-05-27

Catálogo de Reglas de Negocio · SIFAP Legado

Índice consolidado. Las tablas detalladas (187 reglas con fuente archivo.NSN:Lstart-Lend, clasificación EARS y notas) viven en los 5 fragments en _fragments/, uno por par. No duplicamos las líneas aquí, para mantener una única fuente de trazabilidad.

RESUMEN GENERAL

Programas Natural leídos	15
DDMs cruzados	4
Reglas extraídas	187
Confirmadas (cruzaron con docs)	43
Inferidas (solo código, sin doc)	104
Misterios	44
Divergencias críticas doc × código	9

Razón Inferida/Confirmada ≈ 2.4×: la documentación histórica (1997/2008/2012) cubre menos de la mitad de la lógica real. La Etapa 2 trata las reglas Inferidas como candidatas a entrevista con PO/SME, no como verdad cerrada.

FRAGMENTOS POR PAR

PAR	PROGRAMAS	REGLAS	MISTERIOS
1 · Visión	`CADBENEF, CADDEPEND, CADPROG`	32	10
2 · Arquitectura	`BATCHCON, BATCHPGT, BATCHREL`	47	12
3 · Implementación	`CALCBENF, CALCCORR, CALCDSCT`	44	11
4 · Calidad	`VALBENEF, VALDOCS, VALELEG`	30	6
5 · Operaciones	`CONSBENF, RELAUDIT, RELPGT`	34	5

DIVERGENCIAS CROSS-CUTTING (BLOQUEADORES DE LA ETAPA 2)

#	TEMA	CONFLICTO
1	Límite de dependientes	RN-004 dice 3; CADDEPEND.NSN:L66-L69 acepta 5; CALCBENF acepta >3
2	Fórmula del beneficio (Factor Familiar)	REGRAS-NEGOCIO-2012 §6 = aditiva; CALCBENF.NSN = multiplicativa

DESPLAZA PARA VER LAS 187 REGLAS ↓

DONDE VIVEN LOS DATOS

El @archaeologist leyó el código. Pero los datos viven en una base que piensa distinto de todo lo que aprendiste.

Adabas no tiene tablas planas con filas. Tiene un FDT, Field Definition Table, donde un registro carga grupos enteros dentro de sí. Entender esto es entender por qué la traducción es difícil.

EL MODELO QUE CONOCES

Relacional: tablas, filas planas, claves foráneas, índices B-tree. Un dependiente es una fila en otra tabla.

- 1 fila = 1 entidad, siempre plana
- Relaciones vía JOIN entre tablas
- Índices y PKs explícitos en el schema
- NUMERIC, VARCHAR, tipos previsibles

≠

EL MODELO DE ADABAS

FDT: un registro de beneficiario carga hasta 10 dependientes DENTRO de él. Sin tabla separada, sin JOIN.

- Grupo periódico (PE): N ocurrencias en el mismo registro
- Campo multivaluado (MU): varios valores en un solo campo
- Descriptor y superdescriptor en lugar de índice
- Packed decimal: N9.2, N5.4, nunca DOUBLE

EL TERRITORIO A MAPEAR

Cuatro DDMs describen todo el SIFAP. Cada uno carga su propia rareza y sus propios misterios.

BENEFICIARIO

FNR 150

Base principal. El grupo periódico de dependientes (máx 10) y el CPF-centinela 00000000000 viven aquí.

~4,2M registros **52** campos **1 PE** · 3 superdesc

PROGRAMA-SOCIAL

FNR 151

Tabla paramétrica. El campo FATOR-K (N5.4) está aquí, con ">>> NO DOCUMENTADO <<<" grabado en el propio FDT.

45 programas **42** campos **2 PE · 1 MU**

PAGAMENTO

FNR 152

Transaccional, sin política de purga. Todos los registros desde 1998. Una máquina de estados de 7 status en un solo campo.

~180M registros **50** campos **1 PE** · 3 superdesc

AUDITORIA

FNR 153

Inmutable (INSERT ONLY), retención indefinida por ley. Pares antes/después en multivalor de hasta 20 campos.

~25M registros **34** campos **2 MU** · 3 superdesc

ANATOMÍA DE UN REGISTRO

Un registro de beneficiario, por dentro. Haz clic en cada parte para ver por qué no se vuelve una fila de tabla.

RECORD · BENEFICIARIO · ISN 0042

AB · NUM-CPF — campo plano
A 11 · descriptor (DE) · S1

DA · GRP-DEPENDENTE (PE máx 10) — grupo periódico
- occ 1 · CPF-DEP · NOME · PARENTESCO
- occ 2 · CPF-DEP · NOME · PARENTESCO
- occ 3 … até occ 10 (no mesmo registro)

EA · TIPO-DSCT-APLIC (MU máx 8) — multivalor
{ IR · JD · CS · PA · EM · TX · OU · EX }

SUPERDESCRIPTORS — descritores
S1 = AB · S2 = BG+CE · S3 = CA+CE
índices compostos calculados, não colunas

~850 bytes · packed decimals · ordem de campos congelada (Port. 847/2003)

CAMPO PLANO

El caso fácil

NUM-CPF es un campo escalar simple. Es el único tipo que se mapea directo a una columna en una tabla relacional. Todo lo que está debajo es lo que hace difícil la migración.

Adabas: `AB · A 11 · DE` → PostgreSQL: `cpf CHAR(11) PRIMARY KEY`

Haz clic en las otras partes del registro →

@dba
Mapea el schema · DDM → JPA

El @dba lee el FDT real de Adabas y traduce cada construcción extraña al mundo relacional.

QUÉ OBSERVAR

01

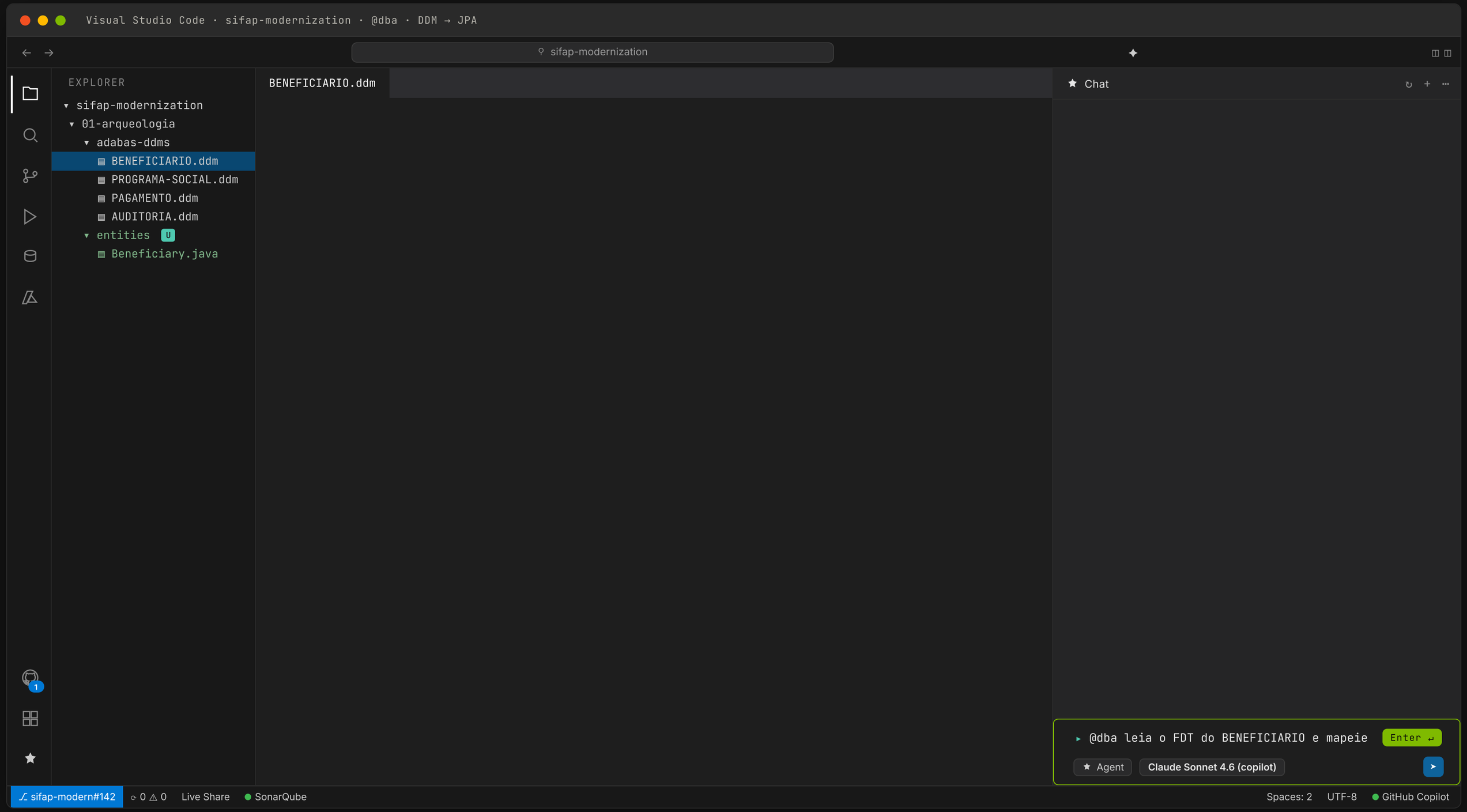
El grupo periódico de dependientes no es columna, se vuelve tabla separada con FK y constraint de cardinalidad.

02

La divergencia del límite (10/5/3) nace en el schema, no en el código. El @dba la expone, no la esconde.

03

FATOR-K aparece con el comentario ">>> NO DOCUMENTADO <<<" grabado en el FDT. Lo marca, no lo inventa.


Visual Studio Code · sifap-modernization · @dba · DDM → JPA
sifap-modernization
EXPLORER
sifap-modernization
01-arqueologia
adabas-ddms
BENEFICIARIO.ddm
PROGRAMA-SOCIAL.ddm
PAGAMENTO.ddm
AUDITORIA.ddm
entities
U
Beneficiary.java
BENEFICIARIO.ddm
Chat
@dba leia o FDT do BENEFICIARIO e mapeie
Enter
Agent
Claude Sonnet 4.6 (copilot)
sifap-modern#142
0 0
Live Share
SonarQube
Spaces: 2
UTF-8
GitHub Copilot

SUB-SECCIÓN · LO QUE ESTO REVELA

La base de datos es el artefacto más difícil de traducir, y los misterios comienzan en el schema.

ANTES

FDT Adabas: PE, MU, descriptores, packed decimals

DESPUÉS

Schema relacional + JPA, con cada divergencia rastreada

El @dba no migra la base a ciegas. Lee el FDT, reconoce lo que no tiene equivalente plano, y transforma cada rareza en una decisión explícita. FATOR-K y el límite de dependientes no son bugs a corregir en silencio: son misterios que suben para que Spec-Driven decida.

DEL DESCUBRIMIENTO A LA ESPECIFICACIÓN

187 reglas descubiertas. Ahora, ¿cómo convertirlas en algo construible sin reintroducir ambigüedad?

La respuesta no es escribir código más rápido. Es escribir la especificación primero, y dejar que el agente trabaje dentro de ella.

VIBE CODING

"Construye el cálculo del beneficio." La IA genera código al instante, adivina la arquitectura, salta los requisitos. Funciona, pero no coincide con la regla del legado. El 40% del tiempo se vuelve retrabajo.

→

DETERMINÍSTICO

La regla del CALCBENF.NSN se vuelve un EARS testeable, con source_legacy. El agente solo implementa lo que dice la spec. Lo que sale es verificable contra el legado.

DOS PIEZAS, UNA CAPA

Spec-Kit es el método. Specky es el motor. No compiten, se apilan.

Spec-Kit

github/spec-kit · open source

La metodología. Define el QUÉ.

- Notación EARS y los 6 patrones de requisito
- Secuencia con gates: specify → clarify → plan → tasks → analyze → implement
- Modelo de constitución y plantillas de prompt que la IA lee y sigue
- Se instala con la CLI specify; comandos /speckit.* en GitHub Copilot

Liviano. Ideal para aprender SDD y adoptar rápido.

Specky

paulasilvatech/specky · CLI toolkit

El motor. Fuerza el CÓMO.

- CLI toolkit: 13 agentes, 22 prompts, 8 skills, 16 hooks, 57 MCP tools
- Máquina de estados de 10 fases que bloquea saltar etapas
- Validador EARS por schema: una frase vaga no pasa
- Análisis cruzado spec↔design↔tasks y compliance (HIPAA, SOC2, GDPR)

La IA es el operador. Specky es el motor.

Spec-Kit
El método · EARS + gates en GitHub Copilot

Spec-Kit convierte los hallazgos de la arqueología en EARS rastreable.

QUÉ OBSERVAR

01

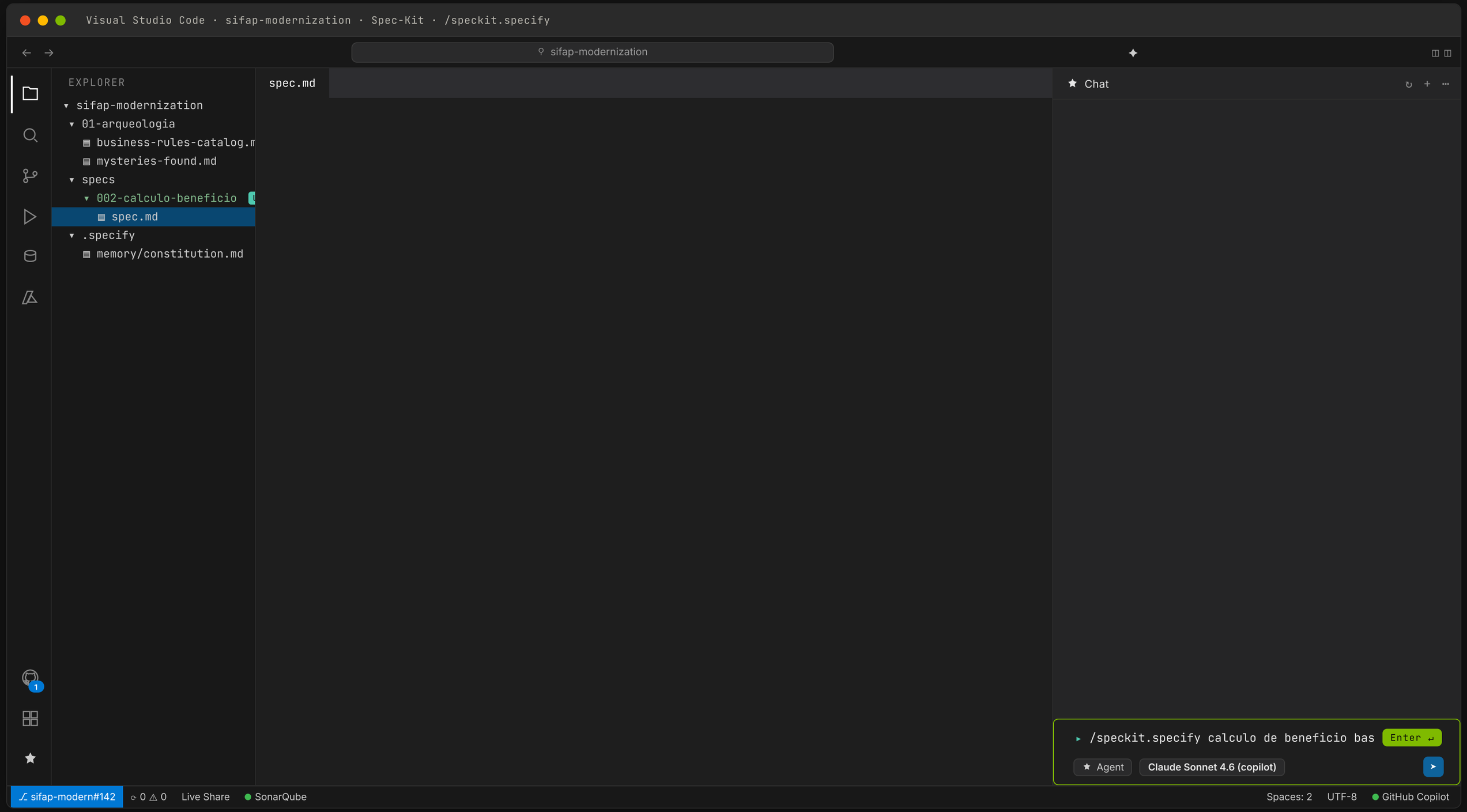
Spec-Kit es open source, del propio GitHub. La entrada son los artefactos del @archaeologist.

02

La regla del CALCBENF.NSN se vuelve REQ-CALC-001, con source_legacy apuntando a la línea exacta.

03

Lo que la IA no sabe (el Factor-K) lo marca [NEEDS CLARIFICATION], no lo inventa.


Visual Studio Code · sifap-modernization · Spec-Kit · /speckit.specify
sifap-modernization
EXPLORER
sifap-modernization
01-arqueologia
business-rules-catalog.m
mysteries-found.md
specs
002-calculo-beneficio
spec.md
.specify
memory/constitution.md
spec.md
Chat
/speckit.specify calculo de beneficio bas
Enter
Agent
Claude Sonnet 4.6 (copilot)
sifap-modern#142
0 0
Live Share
SonarQube
Spaces: 2
UTF-8
GitHub Copilot

Specky
El motor · máquina de estados + validación

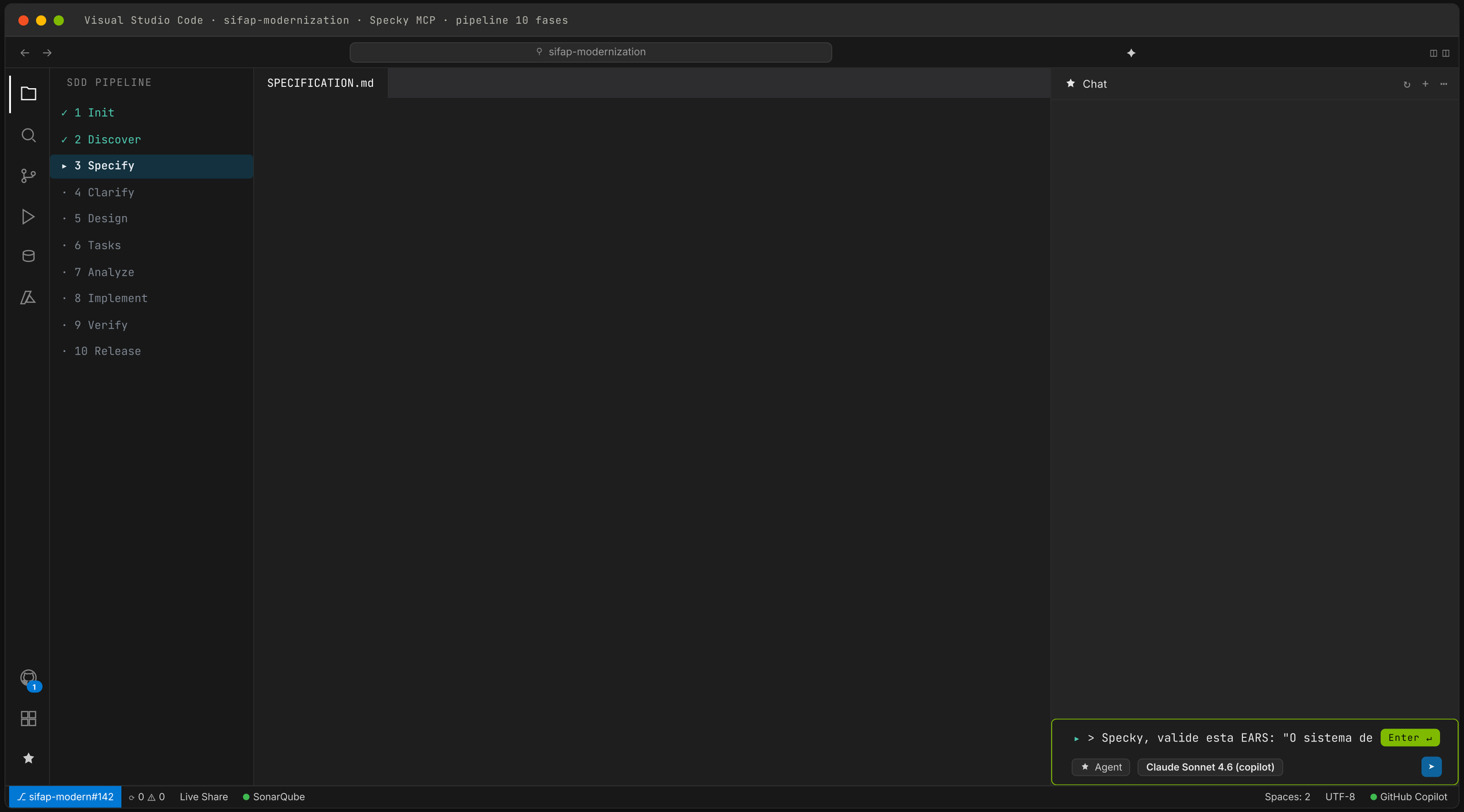
Specky no pide con educación. Fuerza el pipeline.

QUÉ OBSERVAR

01
CLI toolkit: 13 agentes, 22 prompts, 8 skills, 16 hooks, 57 MCP tools, instalado por npm.

02
El validador EARS rechaza una frase vaga a nivel de schema, no es opinión de la IA.

03
La máquina de 10 fases bloquea saltar etapas. El atajo que crea deuda no existe.


Visual Studio Code · sifap-modernization · Specky MCP · pipeline 10 fases
sifap-modernization
SDD PIPELINE
✓ 1 Init
✓ 2 Discover
▸ 3 Specify
· 4 Clarify
· 5 Design
· 6 Tasks
· 7 Analyze
· 8 Implement
· 9 Verify
· 10 Release
SPECIFICATION.md
Chat
▸ > Specky, valide esta EARS: "O sistema de
Enter ↵
Agent
Claude Sonnet 4.6 (copilot)
sifap-modern#142
0 ⚠ 0
Live Share
SonarQube
Spaces: 2
UTF-8
GitHub Copilot

SUB-SECCIÓN · LO QUE ESTO REVELA

Método más motor: la IA es el operador, Specky es el motor.

ANTES	DESPUÉS
Spec sugerida, gates opcionales	Spec validada, fases obligatorias

Spec-Kit aporta la notación y las fases. Specky las convierte en reglas que la máquina cumple. La creatividad de la IA fluye por un pipeline validado en vez de adivinanzas.

A

@architect

Escribe la spec · EARS + source_legacy

El @architect convierte los hallazgos en una spec auditable.

QUÉ OBSERVAR

01

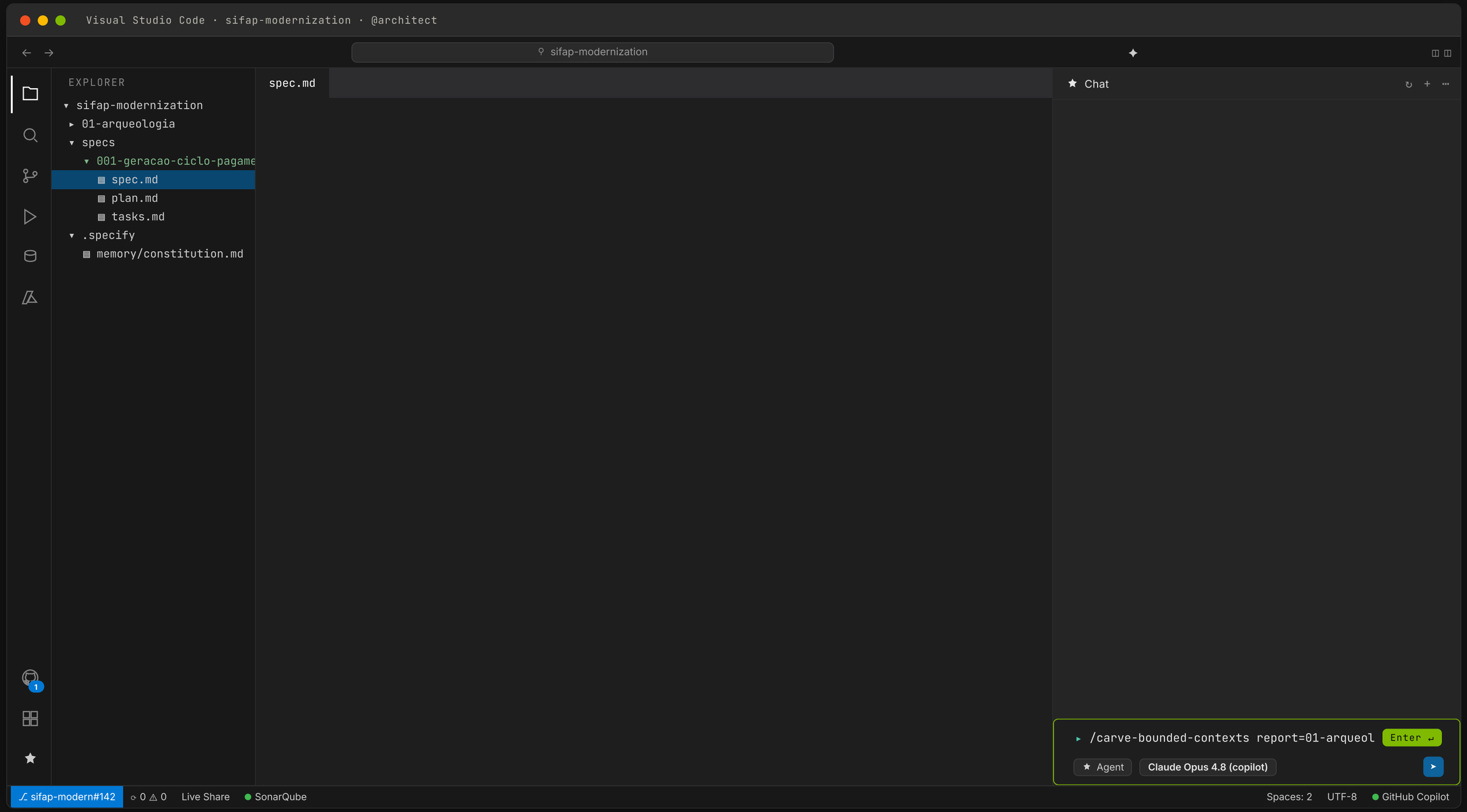
Spec-Kit (github/spec-kit) instala slash commands en GitHub Copilot. Es open source, no propietario.

02

Entrada: los artefactos del @archaeologist. Salida: REQ-PAY-001 en notación EARS.

03

Observe el source_legacy apuntando a BATCHPGT.NSN línea 120.


Visual Studio Code · sifap-modernization · @architect
sifap-modernization
EXPLORER
sifap-modernization
01-arqueologia
specs
001-geracao-ciclo-pagame
spec.md
plan.md
tasks.md
.specify
memory/constitution.md
spec.md
Chat
/carve-bounded-contexts report=01-arqueol
Enter
Agent
Claude Opus 4.8 (copilot)
sifap-modern#142
0 0
Live Share
SonarQube
Spaces: 2
UTF-8
GitHub Copilot

ETAPA 2 · ESPECIFICACIÓN

Mapa de Bounded Contexts · SIFAP 2.0

Producido por el @architect a partir de discovery-report.md §7 y dependency-map.md. Arquitectura objetivo: Modular Monolith (ver ADR-001).

EVALUACIÓN DE HIPÓTESIS

HIPÓTESIS	COHESIÓN	ACOPLAMIENTO	FREC. CAMBIO	DECISIÓN
A · Cálculo + Pago como un único contexto financiero	Baja	Medio	Divergente	Rechazado
B · Auditoría embebida en cada contexto	Alta (si aislada)	Alto (si embebida)	Estable	Rechazado
C · Registro único para Beneficiario + Dependiente + Programa	Alta	Bajo (externo)	Coherente	Aceptado

Conclusiones: separar Cálculo y Pago (unirlos crearía un módulo con dos razones de cambio). Auditoría se vuelve contexto propio con interfaz de escritura publicada. Registro es el aggregate raíz de identidad.

BOUNDED CONTEXTS FINALES

CONTEXTO	RESPONSABILIDAD	DATOS PROPIOS
1 · Registro	Ciclo de vida de beneficiarios, dependientes y programas; validación de CPF y elegibilidad.	`BENEFICIARIO (150) · DEPENDENTE · PROGRAMA_SOCIAL (151)`
2 · Cálculo	Valor base, factores (regional, etario, familia, renta, Fator-K), descuentos (cap 30% excepto judicial), corrección IPCA.	`tabelas de config versionadas · resultados efêmeros`
3 · Pago	Ciclo mensual, remesa CNAB 240, conciliación por match triple, status G/P/E/R/D.	`PAGAMENTO (152, ~180M reg, particionado)`
4 · Auditoría	Traza append-only de todas las acciones; informes incluyendo exclusiones (corrige MYS-010).	`AUDITORIA (153, append-only, particionada por ano)`

COMUNICACIÓN ENTRE CONTEXTOS

DE	A	MECANISMO	DATOS
Pagamento	Cadastro	Llamada síncrona	Beneficiarios elegibles por competencia
Pagamento	Cálculo	Llamada síncrona	(beneficiarioId, competencia) → valor calculado
Cadastro	Auditoria	Domain event	Alta/cambio/baja de beneficiario
Cálculo	Auditoria	Domain event	Evento batch (BT) de cálculo ejecutado
Pagamento	Auditoria	Domain event	Generación de ciclo + conciliación

Regla: comunicación cross-context solo vía interfaz pública o domain event, nunca acceso directo al repositorio de otro módulo (ver ADR-001, anti-corruption layer).

DESPLAZA PARA VER LOS 4 CONTEXTOS ↓

DEMO 3 · LO QUE ESTO REVELA

Sin source_legacy es una suposición. Con source_legacy es ingeniería.

ANTES

Requisito sin origen

DESPUÉS

REQ-PAY-001 → BATCHPGT.NSN#L120

Cada requisito del sistema nuevo apunta a la línea exacta del legado que lo originó. La trazabilidad queda visible y auditable.

@builder

Traduce a Java · semántico, no literal

El @builder traduce Natural a Java idiomático.

QUÉ OBSERVAR

01

No es un port línea por línea. Es traducción semántica.

02

COMPUTE se vuelve BigDecimal con scale. MOVE se vuelve Optional. FIND se vuelve repository query.

03

Cada commit cita Implements REQ-XXX. El test JUnit se genera junto.

Visual Studio Code · sifap-modernization · @builder · Claude Sonnet 4.6

sifap-modernization

BATCHPGT.NSN · Natural

```
240  * CALC FATOR REGIONAL, tabela 27 itens, fixa desde 1997
241  IF#COD-REG >= 1AND#COD-REG <= 25
242  MOVE#TAB-REG(#COD-REG)TO#FATOR-REG
243  ELSE
244  MOVE1.0000TO#FATOR-REG
245  END-IF
246
247  * CALC FATOR FAMILIAR
248  IF#NUM-DEP = 0
249  MOVE1.0000TO#FATOR-FAM
250  ELSE
251  IF#NUM-DEP <= 2
252  COMPUTE#FATOR-FAM = 1.0000 + (#NUM-DEP * 0.0500)
```

PaymentFactorService.java · Java 21 · Spring 3.3

Chat

▸ /translate-natural-to-java BATCHPGT.NSN#L240-L260 (fator regional e familiar). Implements REQ-PAY-002.

Enter ↵

Agent | Claude Sonnet 4.6 (copilot)

sifap-modern#142 | ⟳ 0 ⚠ 0 | Live Share | SonarQube | Spaces: 2 | UTF-8 | GitHub Copilot

DEMO 4 · LO QUE ESTO REVELA

El criterio de aceptación no es que compile. Es que el test de equivalencia pase.

ANTES

Compila ✓

DESPUÉS

Equivalencia semántica ✓ · 7 tests

Línea por línea obtienes código que compila. Equivalencia semántica contra el legado te da un sistema.

@evolution
Despacha a la nube · GitHub Copilot Coding Agent

El @evolution despacha la issue. El GitHub Copilot Coding Agent ejecuta en la nube.

QUÉ OBSERVAR

01

Aquí salimos de VS Code y entramos en GitHub web.

02

Issue bien estructurada: contexto, criterio de aceptación, archivos a tocar, restricciones explícitas.

03

Hand off to GitHub Copilot. El agente ejecuta en segundo plano y abre un PR draft.

@evolution · VS Code · Claude Sonnet 4.6 (copilot)

@evolution /write-github-issue

ETAPA 4 · EVOLUCIÓN · COMPLETADO POR EL EQUIPO

Informe de Experiencia con GitHub Copilot Agent

Completado por el equipo al final de la Etapa 4. Sé honesto: queremos aprender qué funciona y qué no. Las reseñas positivas forzadas no ayudan a nadie.

1 · ISSUES CREADAS Y PRS GENERADOS

ISSUE	PR	ARCHIVOS	PRUEBAS	AJUSTE MANUAL	MERGE
#142 Tope descuentos 30%	#312	4	Sí	No	Tras revisión
#143 Auditoría incluye exclusiones EX	#318	6	Sí	Menor	Tras revisión

2 · RETROSPECTIVA DEL EQUIPO (EN PALABRAS DEL EQUIPO)

Q1 Agente más útil: el @archaeologist, leyó 29 años de Natural sin inventar una regla. Ahorró semanas de lectura manual.

Q2 Falla más sorprendente: una prueba generada que pasaba sin probar nada de verdad. Atrapada en la revisión.

Q3 Qué cambiaría: issues más específicas desde el inicio. Issue vaga genera PR vago.

Q4 Confianza en producción: 7 de 10, sube con más cobertura de prueba de equivalencia.

Q5 Llevar de vuelta: spec antes de código, siempre. El gate humano no es opcional.

3 · TIPOS DE FALLA ENCONTRADOS

✗	Imports incorrectos o circulares (import alucinado)
✗	Prueba que pasaba sin ejercitar la regla
✗	Scope creep, tocó un archivo fuera del pedido
✓	Todos atrapados en la revisión humana antes del merge

4 · CALIDAD DE LOS PRS (NOTA 1-5)

CRITERIO	NOTA
Corrección del código	4
Adherencia a la arquitectura	4
Calidad de las pruebas	3
Documentación generada	4
Promedio general	3.8

DESPLAZA PARA VER LA RETROSPECTIVA ↓

DEMO 5 · LO QUE ESTO REVELA

Ningún merge sin revisión humana y security check.

ANTES

PR abierto por el agente

DESPUÉS

Merge solo tras revisión humana + security

El agente en la nube no reemplaza al desarrollador. Le quita la obligación de escribir la primera versión.

PARTE

V

El legado, rehecho.

La misma María, las mismas 14 reglas, ahora en Spring Boot y Next.js. Nada se descartó: el legado fue entendido, registrado y reconstruido.

DEMO 6 · EL DESTINO

La misma Maria, ahora en Spring Boot y Next.js.

01 Mismo CPF, mismos R$ 1.412,00, mismas reglas de elegibilidad.

02 CBENF020.NSP se convirtió en BeneficiaryController.findByCpf().

03 Las 14 reglas cubiertas por BeneficiaryServiceTest. source_legacy en cada REQ-ID.

Beneficiários
Cadastrar
Pagamentos
Ciclos
Auditoria
Relatório Gerencial

Beneficiários / 123.456.789-01

Histórico completo | Gerar 2ª via | Exportar

BENEFICIÁRIA

Maria das Graças Silva

CPF 123.456.789-01 · 64 anos · Salvador, BA

BPC ATIVO

VALOR MENSAL

R$ 1.412,00

Próximo pagamento: 02/06/2026 · agência 0001 · conta poupança

HISTÓRICO DE PAGAMENTOS

05/2026	R$ 1.412,00	PAGO
04/2026	R$ 1.412,00	PAGO
03/2026	R$ 1.412,00	PAGO
02/2026	R$ 1.412,00	PAGO
01/2026	R$ 1.412,00	PAGO
12/2025	R$ 1.412,00	PAGO

source_legacy · A mesma Maria do legado. `CBENF020.NSP` agora vive em `BeneficiaryController.findByCpf()`. Regras BR-CBENF-001..014 cobertas por `BeneficiaryServiceTest`.

@dba
Destino · Azure Database for PostgreSQL

La base ya modernizada, viva en Azure, y el propio GitHub Copilot conversa con ella.

QUÉ OBSERVAR

01

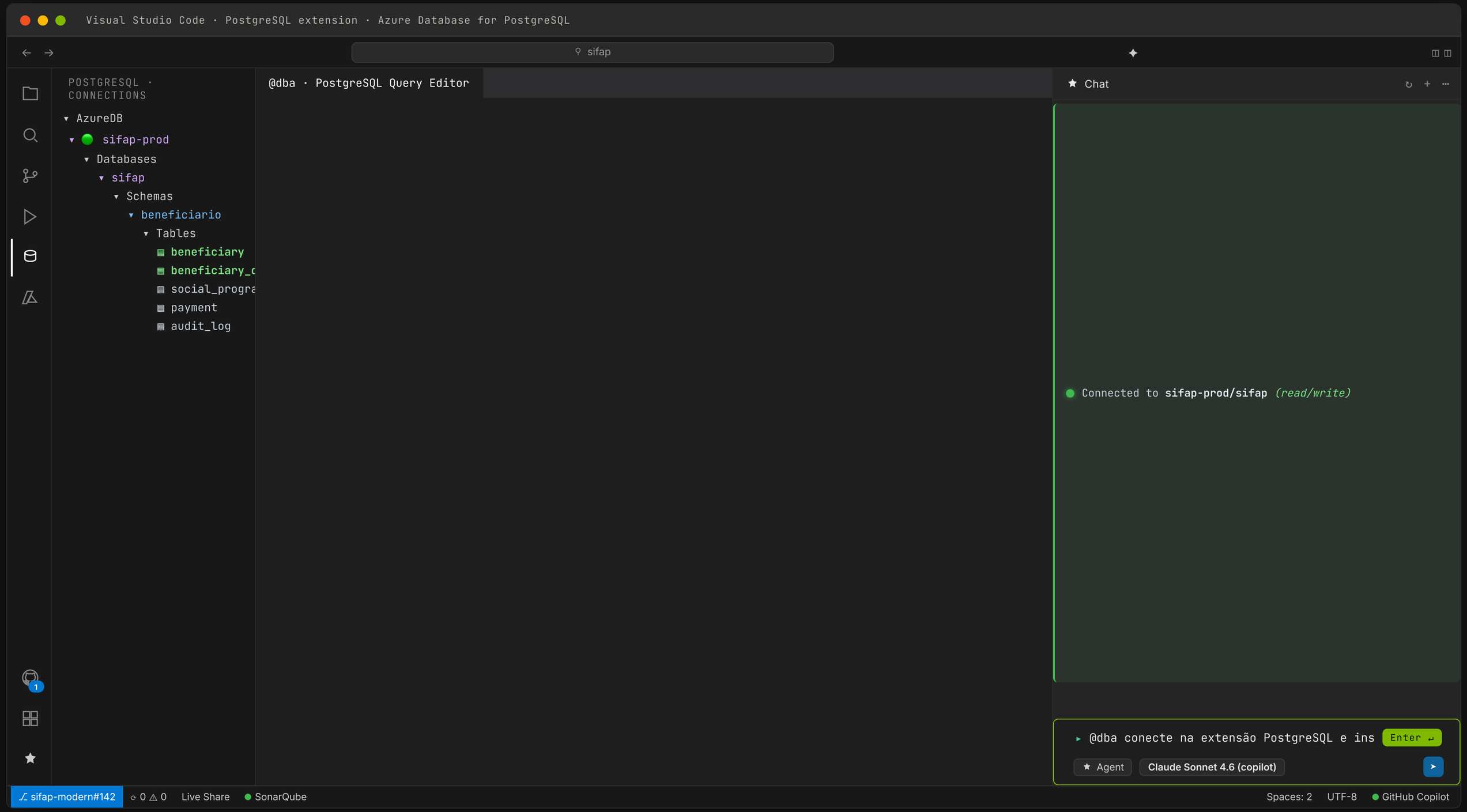
La extensión PostgreSQL conecta VS Code a Azure. El @dba opera la base de producción en lenguaje natural, sin salir del editor.

02

Schema relacional limpio: beneficiary y beneficiary_dependent ligados por FK, visible en el árbol CONNECTIONS.

03

Y el @dba renderiza el ERD completo por la extensión: cinco tablas en verde, todas las FKs dibujadas, leído directo del catálogo de Postgres.


Visual Studio Code · PostgreSQL extension · Azure Database for PostgreSQL
sifap
POSTGRESQL · CONNECTIONS
AzureDB
sifap-prod
Databases
sifap
Schemas
beneficiario
Tables
beneficiary
beneficiary_c
social_progra
payment
audit_log
@dba · PostgreSQL Query Editor
Chat
Connected to sifap-prod/sifap (read/write)
@dba conecte na extensão PostgreSQL e ins
Enter
Agent
Claude Sonnet 4.6 (copilot)
sifap-modern#142
0 0
Live Share
SonarQube
Spaces: 2
UTF-8
GitHub Copilot

DEMO 6 · LO QUE ESTO REVELA

El legado no fue descartado. Fue entendido, registrado y rehecho.

ANTES

CBENF020.NSP · legado

DESPUÉS

BeneficiaryController · con tests

Es otro tipo de modernización. Hay firma humana en cada capa, del requisito al test.

PARTE

VI

Cómo empezar.

Lunes por la mañana, con GitHub Copilot. Cuando crezca, integra Azure. Empieza simple.

QUÉ REBANADA PRIMERO

No empieces por lo más crítico. Empieza por la rebanada que prueba el método sin quebrarte.

La primera rebanada es una apuesta de aprendizaje, no de valor. Elige algo pequeño para terminar en días, real para convencer al equipo.

✓ ELIGE SI

Tiene una regla de negocio clara y testeable (ej: cálculo de descuento), pocos consumidores downstream, y alguien vivo que conoce el dominio.

✗ EVITA SI

Es el motor de pagos entero, toca 180M de registros, o nadie sabe explicar por qué funciona. Eso viene después, con el método ya probado.

→ EN SIFAP SERÍA

CALCDSCT: el cálculo de descuentos. Regla aislada, divergencia conocida (cap 30% vs judicial), y un golden master fácil de armar.

EL LUNES POR LA MAÑANA, EN LA PRÁCTICA

Tres horizontes, comandos reales. GitHub Copilot es el punto de partida; Azure entra cuando crece la complejidad.

01 Semana 1

Habilita GitHub Copilot, sube los 4 agentes custom en un repo sandbox y corre @archaeologist en la primera rebanada.

```
# habilita os agentes custom no repo
$ cp agents/*.md .github/agents/
$ gh extension install github/gh-copilot
# no GitHub Copilot Chat:
@archaeologist mapeie CALCDSCT.NSN
```

Salida: **reglas extraídas + misterios** de la rebanada.

02 Mes 1

Instala Spec-Kit, convierte los hallazgos en una spec firmada y pon un CI mínimo en GitHub Actions.

```
# instala o método e o motor
$ uvx --from specify-cli specify init
$ npm i -g specky-sdd && specky install
# gera e valida a spec
/speckit.specify · /speckit.plan
```

Salida: **spec.md rastreable** + CI verde.

03 Cuando crezca

Con el método probado, integra Azure según lo exija la rebanada: runtime, secretos y observabilidad.

```
# provisiona o que a fatia pedir
$ azd init && azd up
# AI Foundry · App Service
# Key Vault · App Insights
infra/ provisionada por IaC
```

Salida: **rebanada en producción**, observable.

LO QUE MATA EL PROYECTO

Cuatro trampas hunden la modernización de legado. Todas comparten un antídoto: entender antes de construir.

- ✕ **Big bang: reescribir todo de una vez, en paralelo a un legado que sigue cambiando.**
  → Rebana. Una regla a la vez, con golden master contra el legado. Strangler fig, no reemplazo total.

- ✕ **Confiar en el código como spec: "el sistema es la documentación, solo traduzcámoslo."**
  → El código tiene 44 misterios y 9 divergencias. Cada uno se vuelve [NEEDS CLARIFICATION], no una suposición.

- ✕ **Vibe coding en el legado: pedirle al agente que "modernice" sin spec ni test de equivalencia.**
  → Spec primero, validada por schema. El agente solo implementa lo que la EARS declara, y el test prueba paridad.

- ✕ **Corregir bugs en silencio: "esto está mal en el legado, lo arreglo en el nuevo."**
  → Ese "bug" puede ser una regla fiscal de 20 años. Escala la decisión; preserva o corrige con respaldo, nunca por corazonada.

CÓMO SABES QUE FUNCIONÓ

Modernización sin métrica es fe. Estas cuatro prueban que la rebanada está lista, y llevan el método más allá.

100%

DE LAS REGLAS CON SOURCE_LEGACY RASTREABLE HASTA ARCHIVO:LÍNEA

paridad

EN EL GOLDEN MASTER: NUEVO Y LEGADO DAN EL MISMO RESULTADO, AL CENTAVO

0

MISTERIOS SIN RESOLVER VOLVIÉNDOSE CÓDIGO, CADA UNO SE VUELVE DECISIÓN O CLARIFICACIÓN

↓

DEUDA DE INTENCIÓN BAJANDO EN CADA REBANADA, MEDIDA Y VISIBLE EN EL PANEL

¿Qué rebanada de tu legado llevarías al @archaeologist primero?

Esa es la rebanada que empieza el lunes.

CIERRE

El método es entender antes de construir.

CONTACTO

Paula Silva

Software Global Black Belt · Microsoft

paulasilva@microsoft.com

PRÓXIMO PASO

Workshop de una rebanada

Un día, equipo real, legado real, primera rebanada corriendo

.GITHUB/COPILOT · SUBE LOS 4 AGENTES EL LUNES

```
agents:
  - { name: archaeologist, tools: [read, search] }
  - { name: architect,     tools: [read, search, edit] }
  - { name: builder,       tools: [read, search, edit, execute] }
  - { name: evolution,     tools: [read, search, edit, execute, web] }
speckit: { version: latest, integration: copilot }
```