MODERNIZAÇÃO DE LEGADO

Modernização de legado com agentes no GitHub.

Como agentes especializados no GitHub Copilot ajudam a entender estates Natural, Adabas, COBOL e DB2, escrever specs auditáveis e construir o sistema novo. Com o caso real do SIFAP.

APRESENTA

Paula Silva

Software Global Black Belt · Microsoft

DURAÇÃO

~2 horas

DATA

11 jun 2026

AGENDA

Seis partes, um arco.

I	O problema, por que a modernização tradicional parou de fechar a conta
II	A virada agêntica, trabalhar com agentes é diferente de usar IA
III	O legado invisível, com SIFAP como exemplo concreto
IV	Os agentes em ação, do arqueólogo ao agente na nuvem
V	O sistema moderno, o legado refeito e testado
VI	Como começar segunda de manhã

PARTE

I

O problema.

Por que a conta da modernização tradicional parou de fechar.

O CUSTO INVISÍVEL

Cada linha de código antigo cobra juros compostos todo trimestre.

O sistema parece funcionando. O custo está escondido em operação, conformidade e velocidade. Vamos quantificar.

75%
DO ORÇAMENTO DE TI MANTENDO O QUE JÁ EXISTE

10 a 15
ANOS, IDADE MÉDIA DO CÓDIGO CRÍTICO EM ESTATES LEGADOS

5 a 7
ANOS POR PROGRAMA DE MODERNIZAÇÃO TRADICIONAL

70%
DOS PROJETOS DE MODERNIZAÇÃO NÃO CHEGAM AO FIM

Fontes de mercado, Gartner e IDC. Não é catastrofismo, é a linha de base que torna o status quo caro.

OS DOIS CAMINHOS CLÁSSICOS

Reescrita do zero e lift-and-shift têm o mesmo destino: parar antes de chegar.

Reescrita Big Bang

time paralelo, stack nova

Refazer o sistema do zero enquanto o legado segue rodando e mudando.

- Quando o novo fica pronto, o antigo já evoluiu. Não há paridade.
- O conhecimento tácito nunca foi escrito. Some com quem se aposenta.
- Anos sem entregar valor. O projeto morre antes do go-live.

Promete tudo novo. Entrega atraso e risco.

Lift-and-shift

mesma lógica, infra nova

Mover o código como está pra nuvem, sem reescrever a lógica.

- A dívida técnica viaja junto. O legado agora roda caro na nuvem.
- Nenhuma regra foi entendida. A caixa-preta continua preta.
- Modernizou a hospedagem, não o sistema. O problema fica pra depois.

Promete rapidez. Entrega o mesmo problema, com fatura maior.

VOCÊS JÁ TESTARAM IA

A IA generativa de primeira geração quebrou três barreiras. Outras três seguem de pé.

Chat, autocomplete e explain this code aceleram o indivíduo. Mas modernização não é tarefa individual.

O QUE A IA TRADICIONAL RESOLVEU

Velocidade individual em tarefas conhecidas.

- Explica código antigo em segundos.
- Gera boilerplate, testes simples, queries.
- Dev sênior fica 30 a 40% mais rápido no que já domina.

≠

O QUE ELA NÃO RESOLVEU

Contexto, coordenação e continuidade.

- Não conhece o seu domínio até você explicar tudo, toda vez.
- Não coordena várias pessoas em torno do mesmo sistema.
- Não tem memória entre sessões. Cada conversa começa do zero.

PARTE

II

A virada agêntica.

Trabalhar com agentes é diferente de usar IA. A unidade de trabalho muda.

TRÊS NÍVEIS DE ADOÇÃO

Adoção de IA em desenvolvimento passa por três níveis distintos.

A maioria das equipes está parada no Nível 1. O salto que importa é direto para o Nível 3.

NÍVEL 1 · ASSISTIDO · ×1.3

Chat, autocomplete, geração inline. A IA responde quando perguntada, cada dev sozinho com seu copiloto. Ganho: velocidade individual. Limite: processo e qualidade seguem manuais.

NÍVEL 2 · AUMENTADO · ×2.5

Modos Edits e Agent. A IA faz mudança multi-arquivo, refatora, gera testes; o dev revisa e aceita. Ganho: tarefas inteiras delegáveis. Limite: ainda falta processo e memória.

NÍVEL 3 · AGÊNTICO

Agentes com papel, ferramentas e memória, dirigidos por spec e gates. O time orquestra, valida e assina. Ganho: o trabalho escala sem o dev virar gargalo.

O QUE É UM AGENTE, DE VERDADE

Agente virou palavra da moda. Aqui ela tem definição precisa.

Um LLM com papel definido, mais ferramentas, mais memória de sessão. Agentes encadeiam chamadas.

01 · PAPEL

System prompt focado. Você é um arqueólogo de legado, seu trabalho é extrair regra de negócio do Natural com rastreabilidade, use este template.

02 · FERRAMENTAS

Ações que pode executar. Ler e editar arquivos, rodar comandos, consultar MCP servers: GitHub, banco de dados, documentação.

03 · MEMÓRIA

Estado entre turnos. Specs salvas em arquivo, decisões registradas. O agente não recomeça do zero a cada conversa.

SPECS COMO UNIDADE DE TRABALHO

A spec vira a unidade de trabalho. Código é a consequência, não o ponto de partida.

O agente executa o que está escrito, não o que você imaginou. Quanto mais formal o pedido, menos ambiguidade sobra.

SEM SPEC

Refatora isso aqui. Pedido vago: o agente adivinha a arquitetura e o resultado não bate com a regra do legado.

COM ESCOPO

Extrai o módulo X pra uma classe própria. Melhor, mas ainda sem critério de aceite nem contrato de teste.

SPEC EARS

Requisitos formais, critérios de aceite, contratos de teste, com a origem no legado registrada. Frase vaga não passa.

SPEC EXECUTÁVEL

Com Spec-Kit, a spec dirige a execução por etapas, com gates. O agente só implementa o que a spec declara.

A ALOCAÇÃO DE ESFORÇO MUDA

Spec dirigindo agentes muda quem faz o quê dentro do time.

Time tradicional

esforço no teclado

A maior parte do tempo é escrever código à mão.

- 70% escrevendo código manualmente.
- 15% em reuniões pra alinhar quem faz o quê.
- 10% revisando PRs com contexto pobre, 5% em doc que ninguém lê.

Gargalo: o dev sênior vira o funil de qualidade.

Time agêntico

esforço no julgamento

A maior parte do tempo é especificar, orquestrar e revisar.

- Escrever specs claras e critérios de aceite.
- Orquestrar agentes e validar o que eles propõem.
- Decidir as fronteiras e assinar o que entra.

O julgamento humano escala melhor que a digitação.

A TESE

Quem moderniza o legado decide onde o agente entra.

O time não escreve do zero. O agente lê, indexa, propõe. O humano revisa, valida, decide, assina. O agente não é oráculo, e o humano não é digitador. É um pareamento de papéis distintos sobre o mesmo artefato.

QUATRO AGENTES NO GITHUB COPILOT CHAT

@archaeologist lê legado · @architect escreve spec · @builder constrói · @evolution operacionaliza. Cada um com tools restritos ao seu papel.

MAIS O GITHUB COPILOT CODING AGENT NA NUVEM

Recebe issues bem escritas pelo @evolution, executa em background, abre PR draft. Humano revisa antes de qualquer merge.

PARTE

III

O legado invisível.

Tem mainframe rodando coisa séria embaixo de mais sistema do que a gente conta. E o talento que entende está saindo.

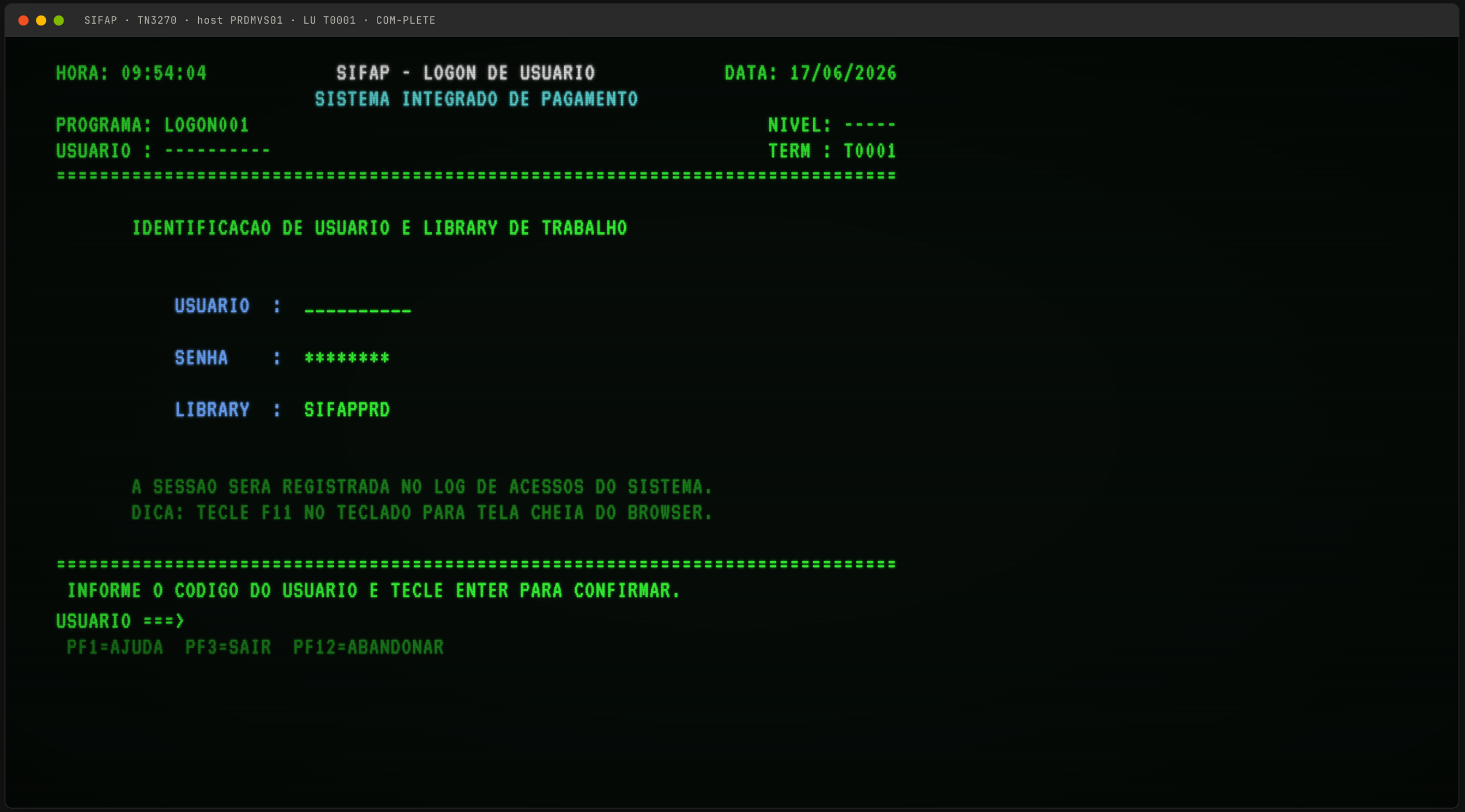
UM SISTEMA, UM EXEMPLO

SIFAP. Sistema social federal. 29 anos em produção. Dois desenvolvedores Natural ativos.

Sistema de Fiscalização e Administração de Pagamentos. Programas sociais federais brasileiros. 3,2 milhões de beneficiários, R$ 4,2 bilhões por ano. 15 programas Natural, 4 DDMs Adabas. Em produção desde 1996. Os dois desenvolvedores Natural ativos no time se aposentam nesta década.

~3,2 M	R$ 4,2 B	29 anos	2
BENEFICIÁRIOS ATIVOS	MOVIMENTAÇÃO ANUAL	IDADE DO CÓDIGO	DEVS NATURAL ATIVOS

É um exemplo. Pode ser o seu sistema de cobrança, seu core bancário, sua plataforma de RH. O perfil se repete em todo estate dessa geração.


SIFAP · TN3270 · host PRDMVS01 · LU T0001 · COM-PLETE
HORA: 09:54:04
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 · O QUE ISSO REVELA

Não dá pra saber o que o CRPGM042 faz só olhando a tela.

ANTES

Tela 3270: CRPGM042 no rodapé, opaco

DEPOIS

Regra documentada e rastreável

Se essa regra for reescrita errada em Java, a Maria recebe errado. É exatamente por isso que arqueologia vem antes de construção.

PARTE

IV

Os agentes em ação.

Quatro agentes no GitHub Copilot Chat, mais o GitHub Copilot Coding Agent na nuvem. Cada um com seu papel.

@archaeologist

Lê o legado · somente leitura

O @archaeologist lê o legado e se recusa a inventar.

O QUE OBSERVAR

01

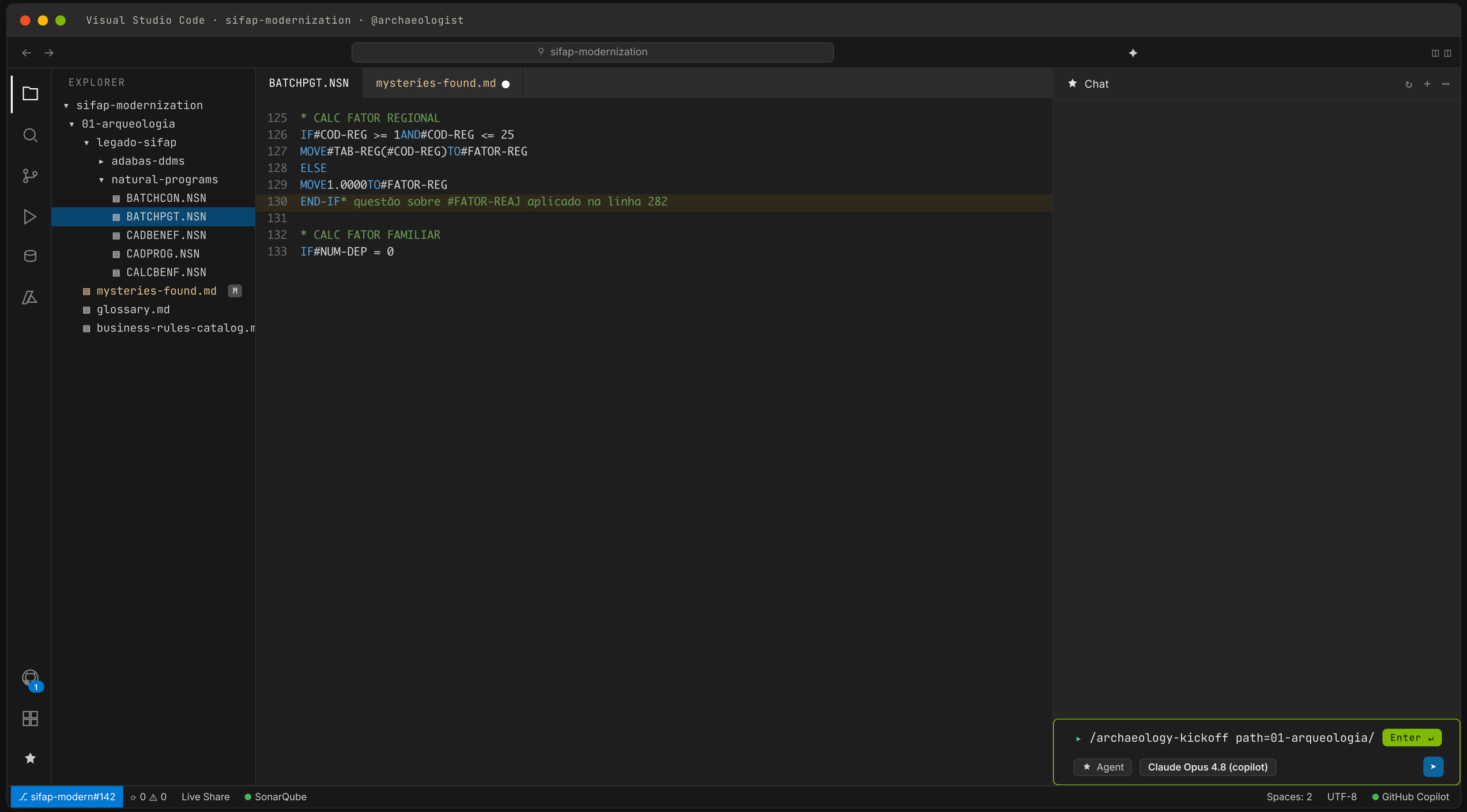
Read-only. Não modifica uma única linha do legado.

02

Quando não sabe, registra um mistério e segue. Descoberta acima de revelação.

03

Observe: ele pergunta de volta antes de responder, e abre o CADPROG.NSN na linha 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

OS MISTÉRIOS QUE O AGENTE EXPÕE

Em quatro horas de SIFAP, 187 regras catalogadas e 17 mistérios oficiais.

187
REGRAS DE NEGÓCIO CATALOGADAS

17
MISTÉRIOS OFICIAIS REGISTRADOS

4 h
READ-ONLY, ZERO ALTERAÇÃO NO LEGADO

MYS-003 · Fator-K

Constante mágica 0.347215 sem origem legal em CADPROG.NSN. Provável conversão Cruzeiro Real → Real (1994). VLR-BASE já vem multiplicado por K. Migrar errado paga duas vezes.

MYS-007 · CPF 00000000000

Institucionalizado no DDM como feature. Backdoor de cadastro fantasma, sem rastreabilidade documentada.

MYS-008 · Região 99

COD-REGIAO=99 (INTERNACIONAL/DIPLOMATICO) pula todas as validações em VALELEG.NSN. Classe inteira nunca passou pela trilha.

MYS-010 · RELAUDIT oculta EX

Filtra silenciosamente ACAO='EX' antes de imprimir auditoria. Viola RN-011. Compliance vermelho.

ESTÁGIO 1 · /EXTRACT-BUSINESS-RULES · GERADO 2026-05-27

Catálogo de Regras de Negócio · SIFAP Legado

Índice consolidado. As tabelas detalhadas (187 regras com fonte arquivo.NSN:Lstart-Lend, classificação EARS e notas) vivem nos 5 fragments em _fragments/, um por par. Não duplicamos as linhas aqui para manter rastreabilidade única.

RESUMO GERAL

Programas Natural lidos	15
DDMs cruzados	4
Regras extraídas	187
Confirmadas (cruzaram com docs)	43
Inferidas (só código, sem doc)	104
Mistérios	44
Divergências críticas doc × código	9

Razão Inferida/Confirmada ≈ 2.4×: a documentação histórica (1997/2008/2012) cobre menos de metade da lógica real. O Estágio 2 trata regras Inferidas como candidatas a entrevista com PO/SME, não como verdade pronta.

FRAGMENTOS POR PAR

PAR	PROGRAMAS	REGRAS	MISTÉRIOS
1 · Visão	`CADBENEF, CADDEPEND, CADPROG`	32	10
2 · Arquitetura	`BATCHCON, BATCHPGT, BATCHREL`	47	12
3 · Implementação	`CALCBENF, CALCCORR, CALCDSCT`	44	11
4 · Qualidade	`VALBENEF, VALDOCS, VALELEG`	30	6
5 · Operações	`CONSBENF, RELAUDIT, RELPGT`	34	5

DIVERGÊNCIAS CROSS-CUTTING (BLOQUEADORES PARA O ESTÁGIO 2)

#	TEMA	CONFLITO
1	Limite de dependentes	RN-004 diz 3; CADDEPEND.NSN:L66-L69 aceita 5; CALCBENF aceita >3
2	Fórmula do benefício (Fator Familiar)	REGRAS-NEGOCIO-2012 §6 = aditiva; CALCBENF.NSN = multiplicativa

ROLE PARA VER AS 187 REGRAS ↓

ONDE OS DADOS MORAM

O @archaeologist leu o código. Mas os dados moram num banco que pensa diferente de tudo que você aprendeu.

Adabas não tem tabelas planas com linhas. Tem um FDT, Field Definition Table, onde um registro carrega grupos inteiros dentro de si. Entender isso é entender por que a tradução é difícil.

O MODELO QUE VOCÊ CONHECE

Relacional: tabelas, linhas planas, chaves estrangeiras, índices B-tree. Um dependente é uma linha em outra tabela.

- 1 linha = 1 entidade, sempre plana
- Relações via JOIN entre tabelas
- Índices e PKs explícitos no schema
- NUMERIC, VARCHAR, tipos previsíveis

≠

O MODELO DO ADABAS

FDT: um registro de beneficiário carrega até 10 dependentes DENTRO dele. Sem tabela separada, sem JOIN.

- Grupo periódico (PE): N ocorrências no mesmo registro
- Campo multivalorado (MU): vários valores num campo só
- Descritor e superdescritor no lugar de índice
- Packed decimal: N9.2, N5.4, nunca DOUBLE

O TERRITÓRIO A MAPEAR

Quatro DDMs descrevem todo o SIFAP. Cada um carrega sua própria estranheza e seus próprios mistérios.

BENEFICIARIO

FNR 150

Base principal. Grupo periódico de dependentes (máx 10) e o CPF-sentinela 00000000000 vivem aqui.

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

PROGRAMA-SOCIAL

FNR 151

Tabela paramétrica. O campo FATOR-K (N5.4) está aqui, com ">>> NÃO DOCUMENTADO <<<" gravado no próprio FDT.

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

PAGAMENTO

FNR 152

Transacional, sem política de purge. Todos os registros desde 1998. Máquina de estados de 7 status num campo só.

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

AUDITORIA

FNR 153

Imutável (INSERT ONLY), retenção indefinida por lei. Pares antes/depois em multivalor de até 20 campos.

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

ANATOMIA DE UM REGISTRO

Um registro de beneficiário, por dentro. Clique em cada parte para ver por que ela não vira uma linha de tabela.

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

O caso fácil

NUM-CPF é um campo escalar simples. É o único tipo que vira diretamente uma coluna numa tabela relacional. Tudo abaixo dele é o que torna a migração difícil.

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

Clique nas outras partes do registro →

@dba
Mapeia o schema · DDM → JPA

O @dba lê o FDT real do Adabas e traduz cada construção estranha para o mundo relacional.

O QUE OBSERVAR

01

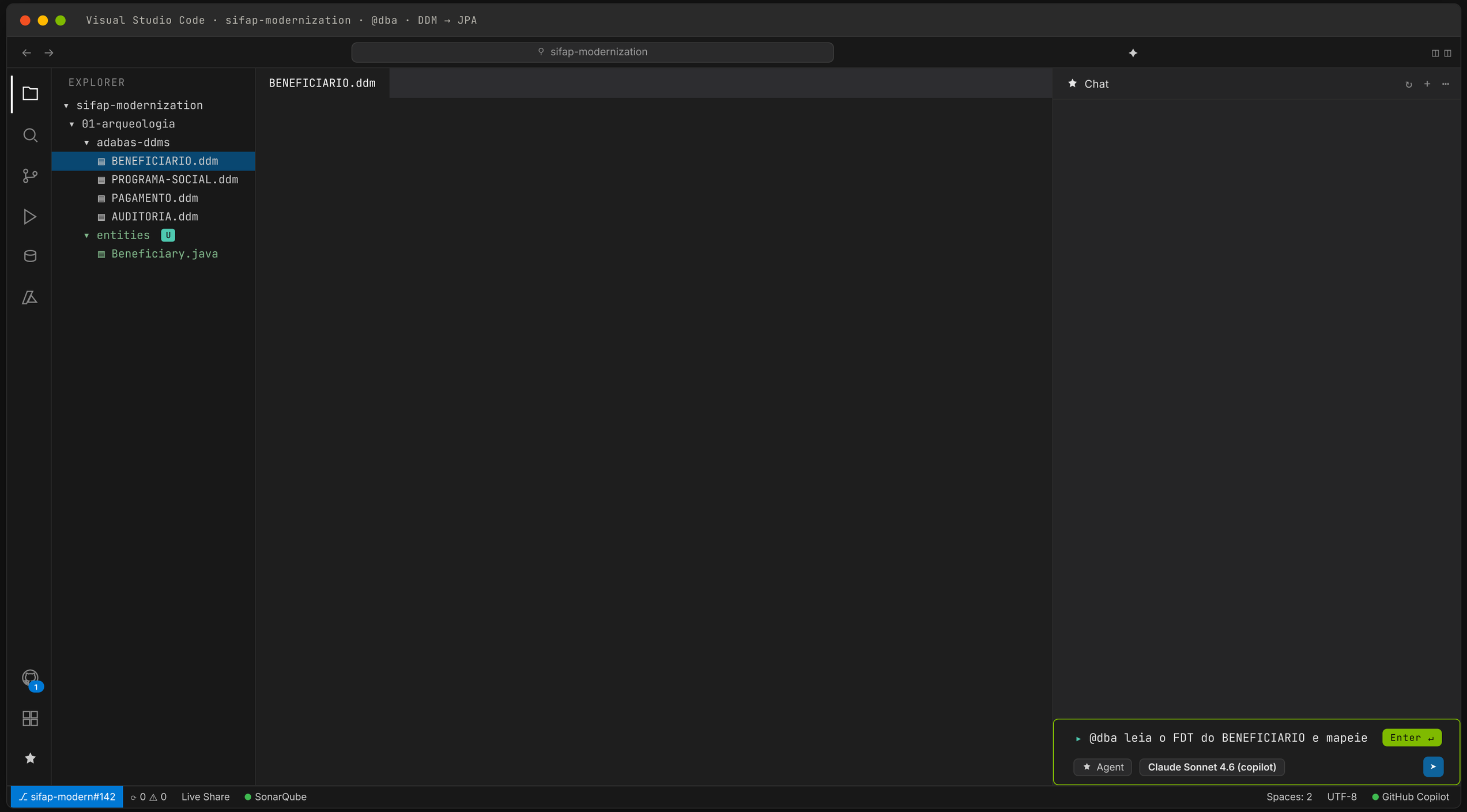
O grupo periódico de dependentes não é coluna, vira tabela separada com FK e constraint de cardinalidade.

02

A divergência do limite (10/5/3) nasce no schema, não no código. O @dba a expõe, não a esconde.

03

O Fator-K aparece com o comentário ">>> NÃO DOCUMENTADO <<<" gravado no FDT. Ele marca, não 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-SEÇÃO · O QUE ISSO REVELA

O banco é o artefato mais difícil de traduzir, e os mistérios começam no schema.

ANTES

FDT Adabas: PE, MU, descritores, packed decimals

DEPOIS

Schema relacional + JPA, com cada divergência rastreada

O @dba não migra o banco às cegas. Ele lê o FDT, reconhece o que não tem equivalente plano, e transforma cada estranheza numa decisão explícita. O Fator-K e o limite de dependentes não são bugs a corrigir em silêncio: são mistérios que sobem para o Spec-Driven decidir.

DA DESCOBERTA À ESPECIFICAÇÃO

187 regras descobertas. Agora, como transformá-las em algo construível sem reintroduzir ambiguidade?

A resposta não é escrever código mais rápido. É escrever a especificação primeiro, e deixar o agente trabalhar dentro dela.

VIBE CODING

"Construa o cálculo do benefício." A IA gera código na hora, adivinha a arquitetura, pula os requisitos. Funciona, mas não bate com a regra do legado. 40% do tempo vira retrabalho.

→

DETERMINÍSTICO

A regra do CALCBENF.NSN vira uma EARS testável, com source_legacy. O agente só implementa o que a spec diz. O que sai é verificável contra o legado.

DUAS PEÇAS, UMA CAMADA

Spec-Kit é o método. Specky é o motor. Não competem, se empilham.

Spec-Kit

github/spec-kit · open source

A metodologia. Define o QUÊ.

- Notação EARS e os 6 padrões de requisito
- Sequência com gates: specify → clarify → plan → tasks → analyze → implement
- Modelo de constituição e templates de prompt que a IA lê e segue
- Instala via specify CLI; comandos /speckit.* no GitHub Copilot

Leve. Ideal para aprender SDD e adotar rápido.

Specky

paulasilvatech/specky · CLI toolkit

O motor. Força o COMO.

- CLI toolkit: 13 agentes, 22 prompts, 8 skills, 16 hooks, 57 MCP tools
- Máquina de estados de 10 fases que bloqueia pular etapa
- Validador EARS por schema: frase vaga não passa
- Análise cruzada spec↔design↔tasks e compliance (HIPAA, SOC2, GDPR)

A IA é o operador. Specky é o motor.

Spec-Kit
O método · EARS + gates no GitHub Copilot

O Spec-Kit transforma os achados da arqueologia em EARS rastreável.

O QUE OBSERVAR

01

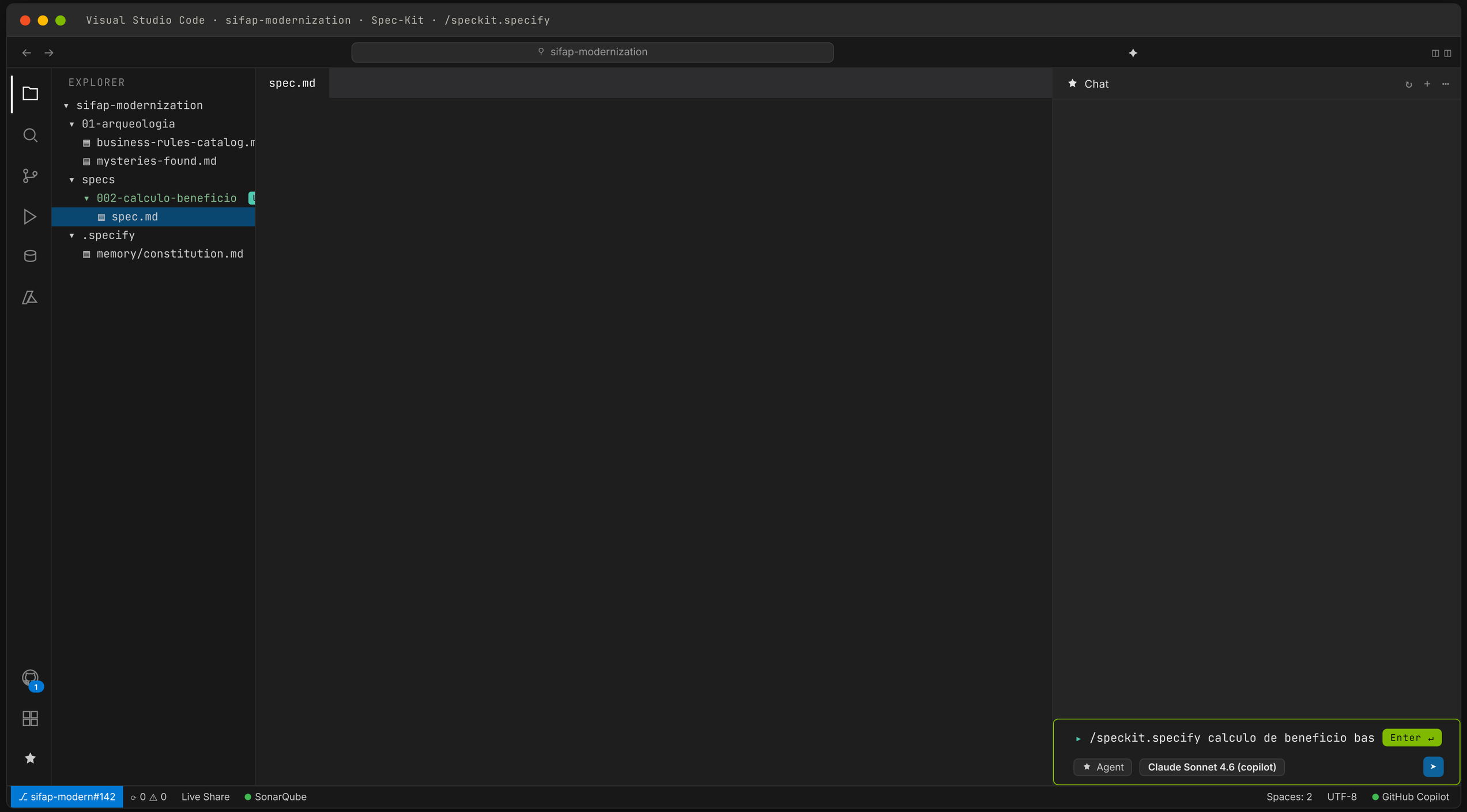
Spec-Kit é open source, do próprio GitHub. A entrada são os artefatos do @archaeologist.

02

A regra do CALCBENF.NSN vira REQ-CALC-001, com source_legacy apontando pra linha exata.

03

O que a IA não sabe (o Fator-K) ela marca [NEEDS CLARIFICATION], não 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
O motor · máquina de estados + validação

O Specky não pede educadamente. Ele força o pipeline.

O QUE OBSERVAR

01

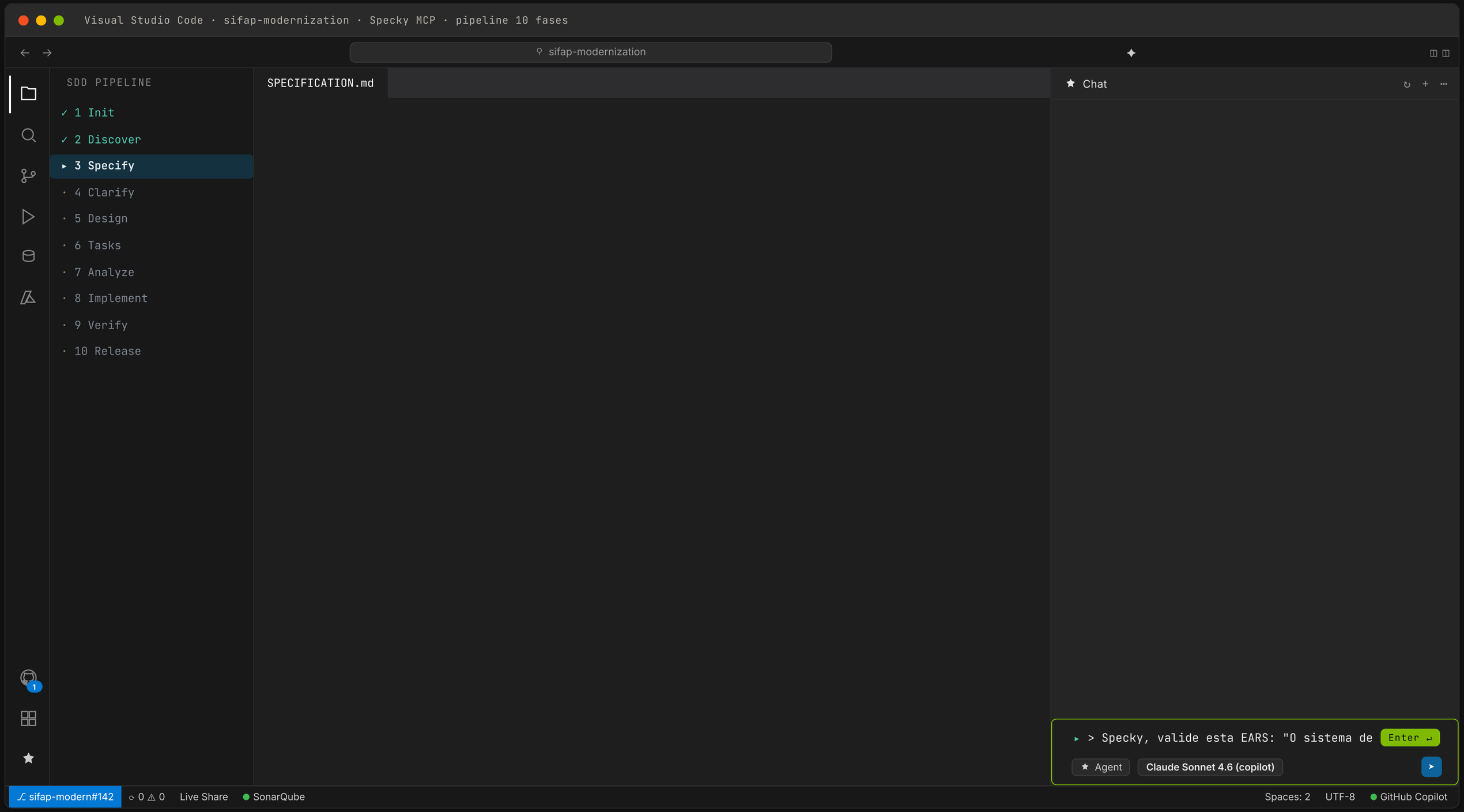
CLI toolkit: 13 agentes, 22 prompts, 8 skills, 16 hooks, 57 MCP tools, instalado por npm.

02

O validador EARS rejeita frase vaga por schema, não é opinião da IA.

03

A máquina de 10 fases bloqueia pular etapa. O atalho que cria dívida não 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-SEÇÃO · O QUE ISSO REVELA

Método mais motor: a IA é o operador, o Specky é o engine.

ANTES	DEPOIS
Spec sugerida, gates opcionais	Spec validada, fases obrigatórias

Spec-Kit dá a notação e as fases. Specky transforma isso em regras que a máquina cumpre. A criatividade da IA passa por um pipeline validado, não por chute.

A

@architect

Escreve a spec · EARS + source_legacy

O @architect transforma os achados em spec auditável.

O QUE OBSERVAR

01

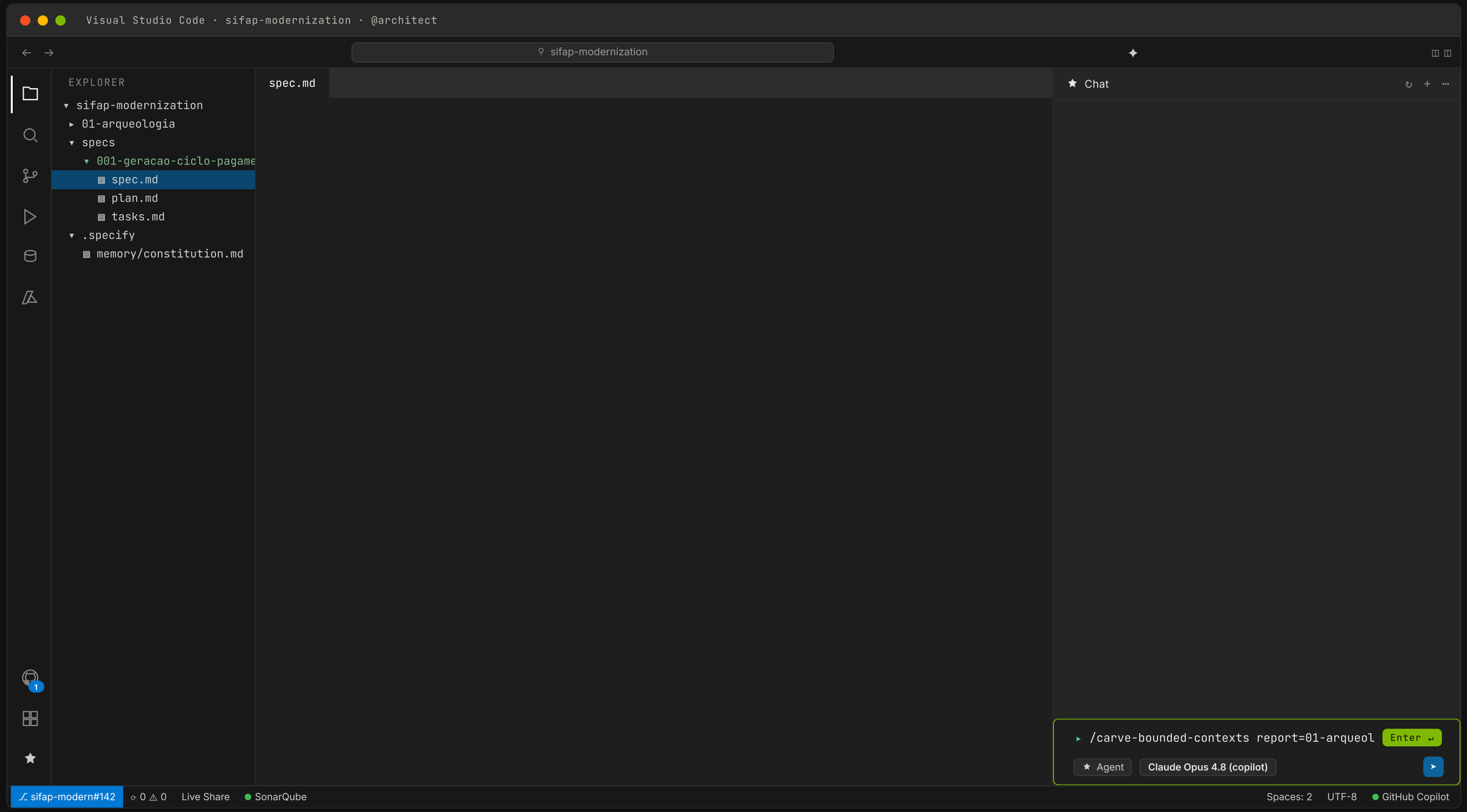
Spec-Kit (github/spec-kit) instala slash commands no GitHub Copilot. É open source, não é proprietário.

02

Entrada: os artefatos do @archaeologist. Saída: REQ-PAY-001 em padrão EARS.

03

Observe o source_legacy apontando pra BATCHPGT.NSN linha 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

ESTÁGIO 2 · ESPECIFICAÇÃO

Mapa de Bounded Contexts · SIFAP 2.0

Produzido pelo @architect a partir de discovery-report.md §7 e dependency-map.md. Arquitetura-alvo: Modular Monolith (ver ADR-001).

AVALIAÇÃO DE HIPÓTESES

HIPÓTESE	COESÃO	ACOPLAMENTO	FREQ. MUDANÇA	DECISÃO
A · Cálculo + Pagamento como um único contexto financeiro	Baixa	Médio	Divergente	Rejeitado
B · Auditoria embutida em cada contexto	Alta (se isolada)	Alto (se embutida)	Estável	Rejeitado
C · Cadastro único para Beneficiário + Dependente + Programa	Alta	Baixo (externo)	Coerente	Aceito

Conclusões: separar Cálculo e Pagamento (juntá-los criaria um módulo com duas razões de mudança). Auditoria vira contexto próprio com interface de escrita publicada. Cadastro é o aggregate raiz de identidade.

BOUNDED CONTEXTS FINAIS

CONTEXTO	RESPONSABILIDADE	DADOS PRÓPRIOS
1 · Cadastro	Ciclo de vida de beneficiários, dependentes e programas; validação de CPF e elegibilidade.	`BENEFICIARIO (150) · DEPENDENTE · PROGRAMA_SOCIAL (151)`
2 · Cálculo	Valor base, fatores (regional, etário, família, renda, Fator-K), descontos (cap 30% exceto judicial), correção IPCA.	`tabelas de config versionadas · resultados efêmeros`
3 · Pagamento	Ciclo mensal, remessa CNAB 240, conciliação por match triplo, status G/P/E/R/D.	`PAGAMENTO (152, ~180M reg, particionado)`
4 · Auditoria	Trilha append-only de todas as ações; relatórios incluindo exclusões (corrige MYS-010).	`AUDITORIA (153, append-only, particionada por ano)`

COMUNICAÇÃO ENTRE CONTEXTOS

DE	PARA	MECANISMO	DADOS
Pagamento	Cadastro	Chamada síncrona	Beneficiários elegíveis por competência
Pagamento	Cálculo	Chamada síncrona	(beneficiariold, competencia) → valor calculado
Cadastro	Auditoria	Domain event	Inclusão/alteração/exclusão de beneficiário
Cálculo	Auditoria	Domain event	Evento batch (BT) de cálculo executado
Pagamento	Auditoria	Domain event	Geração de ciclo + conciliação

Regra: comunicação cross-context só via interface pública ou domain event, nunca acesso direto a repositório de outro módulo (ver ADR-001, anti-corruption layer).

DEMO 3 · O QUE ISSO REVELA

Sem source_legacy é chute. Com source_legacy é engenharia.

ANTES

Requisito solto, sem origem

→

DEPOIS

REQ-PAY-001 → BATCHPGT.NSN#L120

Cada requisito do sistema novo aponta pra linha exata do legado que o originou.
A rastreabilidade fica visível e auditável.

@builder
Traduz pra Java · semântico, não literal

O @builder traduz Natural para Java idiomático.

O QUE OBSERVAR

01

Não é port linha por linha. É tradução semântica.

02

COMPUTE vira BigDecimal com scale. MOVE vira Optional. FIND vira repository query.

03

Cada commit cita Implements REQ-XXX. O teste JUnit é gerado 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 · O QUE ISSO REVELA

O critério de aceite não é compilar. É o teste de equivalência passar.

ANTES

Compila ✓

DEPOIS

Equivalência semântica ✓ · 7 testes

Linha por linha você tem código que compila. Equivalência semântica contra o legado você tem sistema.

@evolution

Despacha pra nuvem · GitHub Copilot Coding Agent

O @evolution despacha a issue. O GitHub Copilot Coding Agent executa na nuvem.

O QUE OBSERVAR

01

Aqui saímos do VS Code e entramos no GitHub web.

02

Issue bem estruturada: contexto, critério de aceite, arquivos a tocar, restrições explícitas.

03

Hand off to GitHub Copilot. O agente executa em background e abre um PR draft.

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

@evolution /write-github-issue

ESTÁGIO 4 · EVOLUÇÃO · PREENCHIDO PELA EQUIPE

Relatório de Experiência com GitHub Copilot Agent

Preenchido pela equipe ao final do Estágio 4. Seja honesto: queremos aprender o que funciona e o que não funciona. Avaliação positiva forçada não ajuda ninguém.

1 · ISSUES CRIADAS E PRS GERADOS

ISSUE	PR	ARQUIVOS	TESTES	AJUSTE MANUAL	MERGE
#142 Cap descontos 30%	#312	4	Sim	Não	Após revisão
#143 Auditoria inclui exclusões EX	#318	6	Sim	Pequeno	Após revisão

2 · RETROSPECTIVA DA EQUIPE (NAS PALAVRAS DA EQUIPE)

Q1 Agente mais útil: o @archaeologist, leu 29 anos de Natural sem inventar regra. Economizou semanas de leitura manual.

Q2 Falha mais surpreendente: um teste gerado que passava sem testar nada de verdade. Pego na revisão.

Q3 O que mudaria: issues mais específicas desde o início. Issue vaga gera PR vago.

Q4 Confiança em produção: 7 de 10, sobe com mais cobertura de teste de equivalência.

Q5 Levar de volta: spec antes de código, sempre. O gate humano não é opcional.

3 · TIPOS DE FALHA ENCONTRADOS

✗	Imports incorretos ou circulares (import alucinado)
✗	Teste que passava sem exercitar a regra
✗	Scope creep, tocou arquivo fora do pedido
✓	Todos pegos na revisão humana antes do merge

4 · QUALIDADE DOS PRS (NOTA 1-5)

CRITÉRIO	NOTA
Corretude do código	4
Aderência à arquitetura	4
Qualidade dos testes	3
Documentação gerada	4
Média geral	3.8

ROLE PARA VER A RETROSPECTIVA ↓

DEMO 5 · O QUE ISSO REVELA

Nenhum merge sem revisão humana e security check.

ANTES

PR aberto pelo agente

DEPOIS

Merge só após revisão humana + security

O agente na nuvem não substitui o desenvolvedor. Tira dele a obrigação de escrever a primeira versão.

PARTE

V

O legado, refeito.

A mesma Maria, as mesmas 14 regras, agora em Spring Boot e Next.js. Nada foi descartado: o legado foi entendido, registrado e reconstruído.

DEMO 6 · O DESTINO

A mesma Maria, agora em Spring Boot e Next.js.

01 Mesmo CPF, mesmos R$ 1.412,00, mesmas regras de elegibilidade.

02 CBENF020.NSP virou BeneficiaryController.findByCpf().

03 As 14 regras cobertas por BeneficiaryServiceTest. source_legacy em 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

O banco já modernizado, vivo no Azure, e o próprio GitHub Copilot conversa com ele.

O QUE OBSERVAR

01

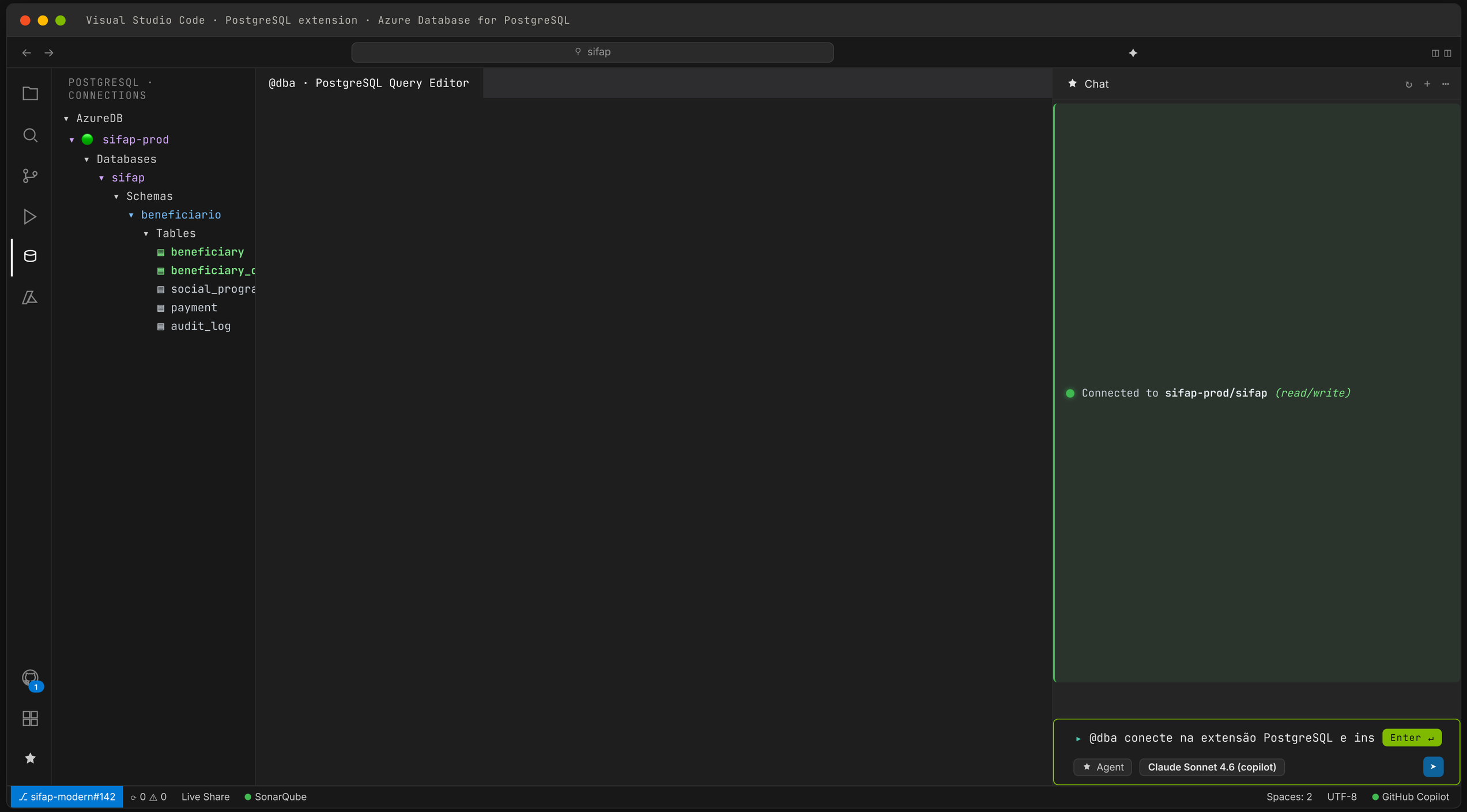
A extensão PostgreSQL conecta o VS Code ao Azure. O @dba opera o banco de produção em linguagem natural, sem sair do editor.

02

Schema relacional limpo: beneficiary e beneficiary_dependent ligados por FK, visível na árvore CONNECTIONS.

03

E o @dba renderiza o ERD inteiro pela extensão: cinco tabelas em verde, todas as FKs desenhadas, lido direto do catálogo do 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 · O QUE ISSO REVELA

O legado não foi descartado. Foi entendido, registrado e refeito.

ANTES

CBENF020.NSP · legado

DEPOIS

BeneficiaryController · com testes

É outro tipo de modernização. Tem assinatura humana em cada camada, do requisito ao teste.

PARTE

VI

Como começar.

Segunda de manhã, com GitHub Copilot. Quando crescer, integra Azure. Comece simples.

QUAL FATIA PRIMEIRO

Não comece pelo mais crítico. Comece pela fatia que prova o método sem te quebrar.

A primeira fatia é uma aposta de aprendizado, não de valor. Escolha algo pequeno o bastante para terminar em dias, real o bastante para convencer o time.

✓ ESCOLHA SE

Tem regra de negócio clara e testável (ex: cálculo de desconto), poucos integrantes downstream, e alguém vivo que conhece o domínio.

✗ EVITE SE

É o motor de pagamento inteiro, toca 180M de registros, ou ninguém sabe explicar por que funciona. Isso vem depois, com o método já provado.

→ NO SIFAP SERIA

CALCDSCT: o cálculo de descontos. Regra isolada, divergência conhecida (cap 30% vs judicial), e um golden master fácil de montar.

SEGUNDA DE MANHÃ, NA PRÁTICA

Três horizontes, comandos reais. GitHub Copilot é o ponto de partida; Azure entra conforme a complexidade.

01 Semana 1

Habilite o GitHub Copilot, suba os 4 agentes custom num repo de teste e rode o @archaeologist na primeira fatia.

```
# 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
```

Saída: **regras extraídas + mistérios** da fatia.

02 Mês 1

Instale o Spec-Kit, transforme os achados em spec assinada e ponha um CI mínimo no 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
```

Saída: **spec.md rastreável** + CI verde.

03 Quando crescer

Com o método provado, integre o Azure conforme a fatia exigir: runtime, segredos e observabilidade.

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

Saída: **fatia em produção**, observável.

O QUE MATA O PROJETO

Quatro armadilhas afundam modernização de legado. Todas têm o mesmo antídoto: entender antes de construir.

- ✕ **Big bang: reescrever tudo de uma vez, em paralelo ao legado que segue mudando.**
  → Fatie. Uma regra por vez, com golden master contra o legado. Strangler fig, não substituição total.
- ✕ **Confiar no código como spec: "o sistema é a documentação, vamos só traduzir."**
  → O código tem 44 mistérios e 9 divergências. Cada um vira [NEEDS CLARIFICATION], não uma suposição.
- ✕ **Vibe coding no legado: pedir pro agente "modernizar" sem spec nem teste de equivalência.**
  → Spec primeiro, validada por schema. O agente só implementa o que a EARS declara, e o teste prova paridade.
- ✕ **Corrigir bugs em silêncio: "isso tá errado no legado, vou só consertar no novo."**
  → Aquele "bug" pode ser regra fiscal de 20 anos. Escale a decisão; preserve ou corrija com respaldo, nunca por palpite.

COMO VOCÊ SABE QUE FUNCIONOU

Modernização sem métrica é fé. Estas quatro provam que a fatia está pronta, e seguem o método além dela.

100%

DAS REGRAS COM SOURCE_LEGACY RASTREÁVEL ATÉ ARQUIVO:LINHA

paridade

NO GOLDEN MASTER: NOVO E LEGADO DÃO O MESMO RESULTADO, CENTAVO A CENTAVO

0

MISTÉRIOS NÃO RESOLVIDOS VIRANDO CÓDIGO, TODOS VIRAM DECISÃO OU CLARIFICAÇÃO

↓

DÍVIDA DE INTENÇÃO CAINDO A CADA FATIA, MEDIDA E VISÍVEL NO PAINEL

Qual fatia do seu legado você levaria pro @archaeologist primeiro?

Essa é a fatia que começa segunda de manhã.

ENCERRAMENTO

O método é entender antes de construir.

CONTATO

Paula Silva
Software Global Black Belt · Microsoft
paulasilva@microsoft.com

PRÓXIMO PASSO

Workshop de uma fatia
Um dia, time real, legado real, primeira fatia rodando

.GITHUB/COPILOT · SUBA OS 4 AGENTES NA SEGUNDA

```
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 }
```