LEGACY MODERNIZATION

Legacy modernization with agents on GitHub.

How specialized agents on GitHub Copilot help understand Natural, Adabas, COBOL and DB2 estates, write auditable specs and build the new system. With the SIFAP case.

PRESENTER

Paula Silva

Software Global Black Belt · Microsoft

DURATION

~2 hours

DATE

11 jun 2026

AGENDA

Six parts, one arc.

I The problem, why traditional modernization stopped adding up

II The agentic shift, working with agents is different from using AI

III The invisible legacy, with SIFAP as a concrete example

IV The agents in action, from the archaeologist to the cloud agent

V The modern system, the legacy rebuilt and tested

VI How to start Monday morning

PART

I

The problem.

Why the traditional modernization math stopped adding up.

THE INVISIBLE COST

Every line of old code charges compound interest every quarter.

The system looks like it works. The cost hides in operations, compliance and speed. Let us quantify it.

75%
OF IT BUDGET SPENT KEEPING WHAT ALREADY EXISTS

10 to 15
YEARS, AVERAGE AGE OF CRITICAL CODE IN LEGACY ESTATES

5 to 7
YEARS PER TRADITIONAL MODERNIZATION PROGRAM

70%
OF MODERNIZATION PROJECTS NEVER REACH THE FINISH LINE

Market sources, Gartner and IDC. This is not doom, it is the baseline that makes the status quo expensive.

THE TWO CLASSIC PATHS

Full rewrite and lift-and-shift share one destination: stalling before arrival.

Big Bang rewrite

parallel team, new stack

Rebuild the system from scratch while the legacy keeps running and changing.

- When the new is done, the old has moved on. There is no parity.
- Tacit knowledge was never written down. It leaves with retirees.
- Years with no value delivered. The project dies before go-live.

Promises all new. Delivers delay and risk.

Lift-and-shift

same logic, new infra

Move the code as is to the cloud, without rewriting the logic.

- Technical debt travels along. The legacy now runs expensively in the cloud.
- No rule was understood. The black box stays black.
- You modernized the hosting, not the system. The problem waits.

Promises speed. Delivers the same problem, with a bigger bill.

YOU HAVE ALREADY TRIED AI

First generation generative AI broke three barriers. Three others still stand.

Chat, autocomplete and explain this code speed up the individual. But modernization is not an individual task.

WHAT TRADITIONAL AI SOLVED

Individual speed on familiar tasks.

- Explains old code in seconds.
- Generates boilerplate, simple tests, queries.
- A senior dev gets 30 to 40% faster at what they already know.

≠

WHAT IT DID NOT SOLVE

Context, coordination and continuity.

- It does not know your domain until you explain everything, every time.
- It does not coordinate several people around the same system.
- It has no memory across sessions. Every chat starts from zero.

PART

II

The agentic shift.

Working with agents is different from using AI. The unit of work changes.

THREE LEVELS OF ADOPTION

AI adoption in development moves through three distinct levels.

Most teams are stuck at Level 1. The jump that matters goes straight to Level 3.

LEVEL 1 · ASSISTED · ×1.3

Chat, autocomplete, inline generation. AI answers when asked, each dev alone with their copilot. Gain: individual speed. Limit: process and quality stay manual.

LEVEL 2 · AUGMENTED · ×2.5

Edits and Agent modes. AI makes multi-file changes, refactors, writes tests; the dev reviews and accepts. Gain: whole tasks become delegable. Limit: process and memory still missing.

LEVEL 3 · AGENTIC

Agents with role, tools and memory, driven by spec and gates. The team orchestrates, validates and signs. Gain: work scales without the dev becoming the bottleneck.

WHAT AN AGENT REALLY IS

Agent became a buzzword. Here it has a precise definition.

An LLM with a defined role, plus tools, plus session memory. Agents chain calls.

01 · ROLE

A focused system prompt. You are a legacy archaeologist, your job is to extract business rules from Natural with traceability, use this template.

02 · TOOLS

Actions it can perform. Read and edit files, run commands, query MCP servers: GitHub, database, documentation.

03 · MEMORY

State across turns. Specs saved to file, decisions recorded. The agent does not restart from zero each conversation.

SPECS AS THE UNIT OF WORK

The spec becomes the unit of work. Code is the consequence, not the starting point.

The agent executes what is written, not what you imagined. The more formal the request, the less ambiguity is left.

NO SPEC

Refactor this. Vague request: the agent guesses the architecture and the result does not match the legacy rule.

WITH SCOPE

Extract module X into its own class. Better, but still no acceptance criteria or test contract.

EARS SPEC

Formal requirements, acceptance criteria, test contracts, with the legacy origin recorded. A vague sentence does not pass.

EXECUTABLE SPEC

With Spec-Kit, the spec drives execution in stages, with gates. The agent only implements what the spec declares.

EFFORT ALLOCATION SHIFTS

Specs driving agents change who does what inside the team.

Traditional team

effort at the keyboard

Most of the time goes into writing code by hand.

- 70% writing code manually.
- 15% in meetings to align who does what.
- 10% reviewing PRs with poor context, 5% on docs nobody reads.

Bottleneck: the senior dev becomes the quality funnel.

Agentic team

effort on judgment

Most of the time goes into specifying, orchestrating and reviewing.

- Writing clear specs and acceptance criteria.
- Orchestrating agents and validating what they propose.
- Deciding the boundaries and signing what goes in.

Human judgment scales better than typing.

THE THESIS

Whoever modernizes the legacy decides where the agent comes in.

The team does not write from scratch. The agent reads, indexes, proposes. The human reviews, validates, decides, signs. The agent is not an oracle, and the human is not a typist. It is a pairing of distinct roles over the same artifact.

FOUR AGENTS IN GITHUB COPILOT CHAT

@archaeologist reads legacy · @architect writes spec · @builder constructs · @evolution operationalizes. Each one with tools scoped to its role.

PLUS THE GITHUB COPILOT CODING AGENT IN THE CLOUD

Receives well-written issues from @evolution, runs in background, opens a draft PR. Human reviews before any merge.

PART

III

The invisible estate.

There is mainframe code still running serious workloads in more systems than we like to admit. And the talent who understands it is retiring.

ONE SYSTEM, ONE EXAMPLE

SIFAP. Federal social benefits system. 29 years in production. Two Natural developers left.

Brazilian Federal Payment Administration System for social benefits. 3.2 million active beneficiaries, R$ 4.2 billion per year. 15 Natural programs, 4 Adabas DDMs. In production since 1996. The two Natural developers still active on the team retire this decade.

~3.2 M	R$ 4.2 B	29 years	2
ACTIVE BENEFICIARIES	ANNUAL THROUGHPUT	CODE AGE	ACTIVE NATURAL DEVS

> *It is an example. It could be your billing system, your banking core, your HR platform. The profile repeats across any estate of that generation.*

SIFAP · TN3270 · host PRDMVS01 · LU T0001 · COM-PLETE

```
HORA: 09:53:27            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 · WHAT THIS REVEALS

You cannot tell what CRPGM042 does just by looking at the screen.

BEFORE

3270 screen: CRPGM042 in the footer, opaque

AFTER

Rule documented and traceable

If that rule is rewritten wrong in Java, Maria gets paid wrong. That is exactly why archaeology comes before construction.

PART

IV

Four agents in action.

Four agents in GitHub Copilot Chat, plus the GitHub Copilot Coding Agent in the cloud. Each one with its own role.

@archaeologist

Reads the legacy · read-only

The @archaeologist reads the legacy and refuses to invent.

WHAT TO WATCH

01

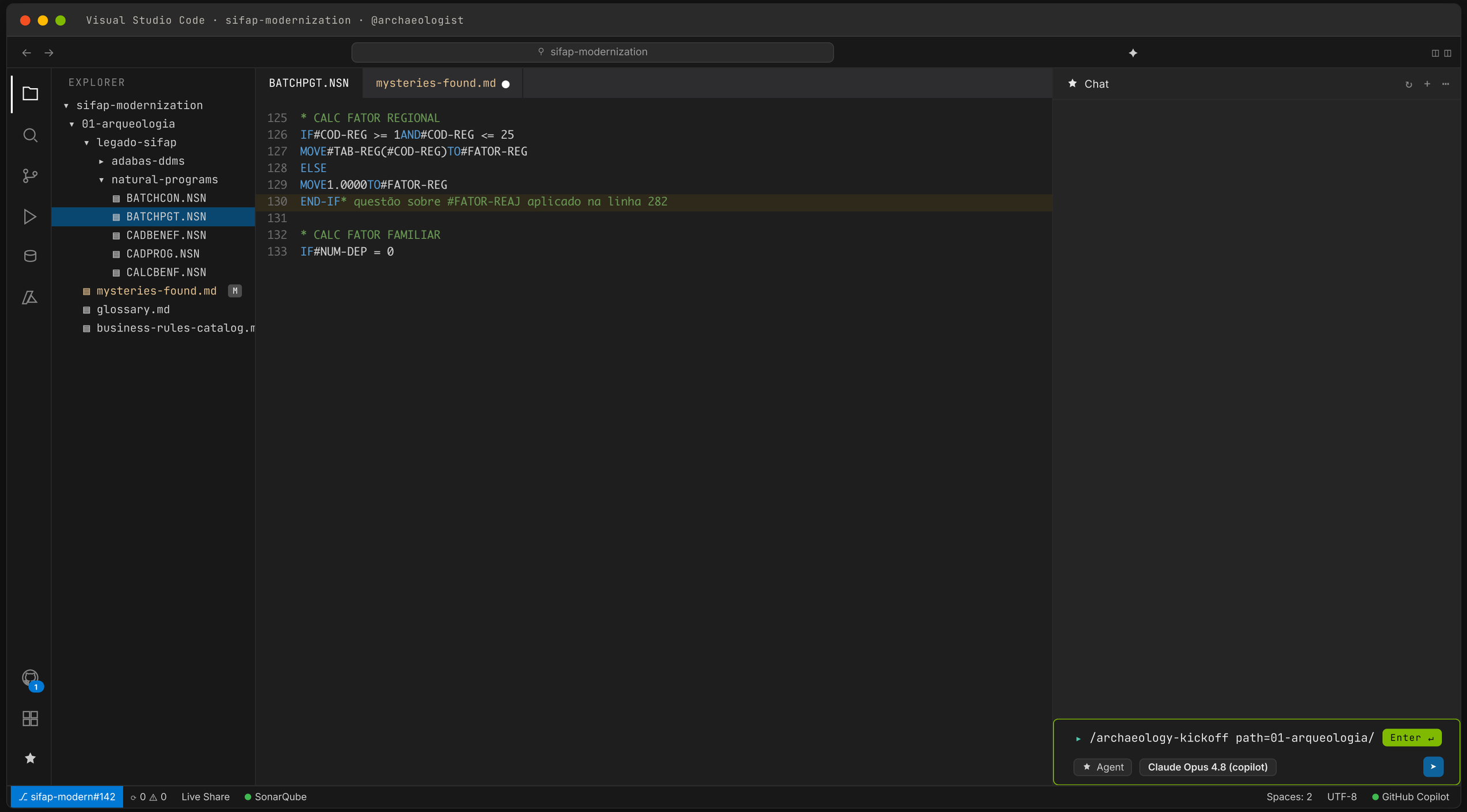
Read-only. It does not modify a single line of the legacy.

02

When it does not know, it records a mystery and moves on. Discovery above revelation.

03

Watch: it asks back before answering, and opens CADPROG.NSN at line 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

THE MYSTERIES THE AGENT SURFACES

Four hours on SIFAP. 187 business rules catalogued. 17 official mysteries.

187
BUSINESS RULES CATALOGUED

17
OFFICIAL MYSTERIES RECORDED

4 h
READ-ONLY, ZERO LEGACY CHANGES

MYS-003 · Fator-K

Magic constant 0.347215 with no legal origin in CADPROG.NSN. Likely Cruzeiro Real to Real conversion from 1994. VLR-BASE already arrives multiplied by K. Migrate wrong, pay twice.

MYS-007 · CPF 00000000000

Institutionalized in the DDM as a feature. A ghost-registration backdoor with no documented traceability.

MYS-008 · Region 99

COD-REGIAO=99 (INTERNATIONAL/DIPLOMATIC) skips every eligibility check in VALELEG.NSN. An entire class of beneficiaries that never passed the trail.

MYS-010 · RELAUDIT hides EX

Silently filters ACAO='EX' rows out of the audit print. Violates RN-011. Compliance red flag.

STAGE 1 · /EXTRACT-BUSINESS-RULES · GENERATED 2026-05-27

Business Rules Catalog · SIFAP Legacy

Consolidated index. The detailed tables (187 rules with source file.NSN:Lstart-Lend, EARS classification, and notes) live in the 5 fragments under _fragments/, one per pair. We do not duplicate the lines here, to keep a single source of traceability.

GENERAL SUMMARY

Natural programs read	15
DDMs cross-referenced	4
Rules extracted	187
Confirmed (cross-checked with docs)	43
Inferred (code only, no doc)	104
Mysteries	44
Critical doc × code divergences	9

Inferred/Confirmed ratio ≈ 2.4×: the historical documentation (1997/2008/2012) covers less than half of the real logic. Stage 2 treats Inferred rules as interview candidates with PO/SME, not as settled truth.

FRAGMENTS BY PAIR

PAIR	PROGRAMS	RULES	MYSTERIES
1 · Vision	`CADBENEF, CADDEPEND, CADPROG`	32	10
2 · Architecture	`BATCHCON, BATCHPGT, BATCHREL`	47	12
3 · Implementation	`CALCBENF, CALCCORR, CALCDSCT`	44	11
4 · Quality	`VALBENEF, VALDOCS, VALELEG`	30	6
5 · Operations	`CONSBENF, RELAUDIT, RELPGT`	34	5

CROSS-CUTTING DIVERGENCES (STAGE 2 BLOCKERS)

#	THEME	CONFLICT
1	Dependent limit	RN-004 says 3; CADDEPEND.NSN:L66-L69 accepts 5; CALCBENF accepts >3
2	Benefit formula (Family Factor)	REGRAS-NEGOCIO-2012 §6 = additive; CALCBENF.NSN = multiplicative

WHERE THE DATA LIVES

The @archaeologist read the code. But the data lives in a database that thinks differently from everything you learned.

Adabas has no flat tables with rows. It has an FDT, Field Definition Table, where one record carries entire groups inside it. Understanding this is understanding why the translation is hard.

THE MODEL YOU KNOW

Relational: tables, flat rows, foreign keys, B-tree indexes. A dependent is a row in another table.

- 1 row = 1 entity, always flat
- Relations via JOIN between tables
- Explicit indexes and PKs in the schema
- NUMERIC, VARCHAR, predictable types

≠

THE ADABAS MODEL

FDT: a beneficiary record carries up to 10 dependents INSIDE it. No separate table, no JOIN.

- Periodic group (PE): N occurrences in the same record
- Multi-value field (MU): many values in a single field
- Descriptor and superdescriptor instead of an index
- Packed decimal: N9.2, N5.4, never DOUBLE

THE TERRITORY TO MAP

Four DDMs describe the entire SIFAP. Each carries its own strangeness and its own mysteries.

BENEFICIARIO

FNR 150

Main base. The periodic group of dependents (max 10) and the sentinel CPF 00000000000 live here.

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

PROGRAMA-SOCIAL

FNR 151

Parametric table. The FATOR-K field (N5.4) is here, with ">>> NOT DOCUMENTED <<<" written into the FDT itself.

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

PAGAMENTO

FNR 152

Transactional, no purge policy. Every record since 1998. A 7-status state machine in a single field.

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

AUDITORIA

FNR 153

Immutable (INSERT ONLY), indefinite retention by law. Before/after pairs in multi-value of up to 20 fields.

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

ANATOMY OF A RECORD

A beneficiary record, from the inside. Click each part to see why it does not become a table row.

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)

FLAT FIELD

The easy case

NUM-CPF is a simple scalar field. It is the only kind that maps directly to a column in a relational table. Everything below it is what makes the migration hard.

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

Click the other parts of the record →

@dba

Maps the schema · DDM → JPA

The @dba reads the real Adabas FDT and translates each strange construct into the relational world.

WHAT TO WATCH

01

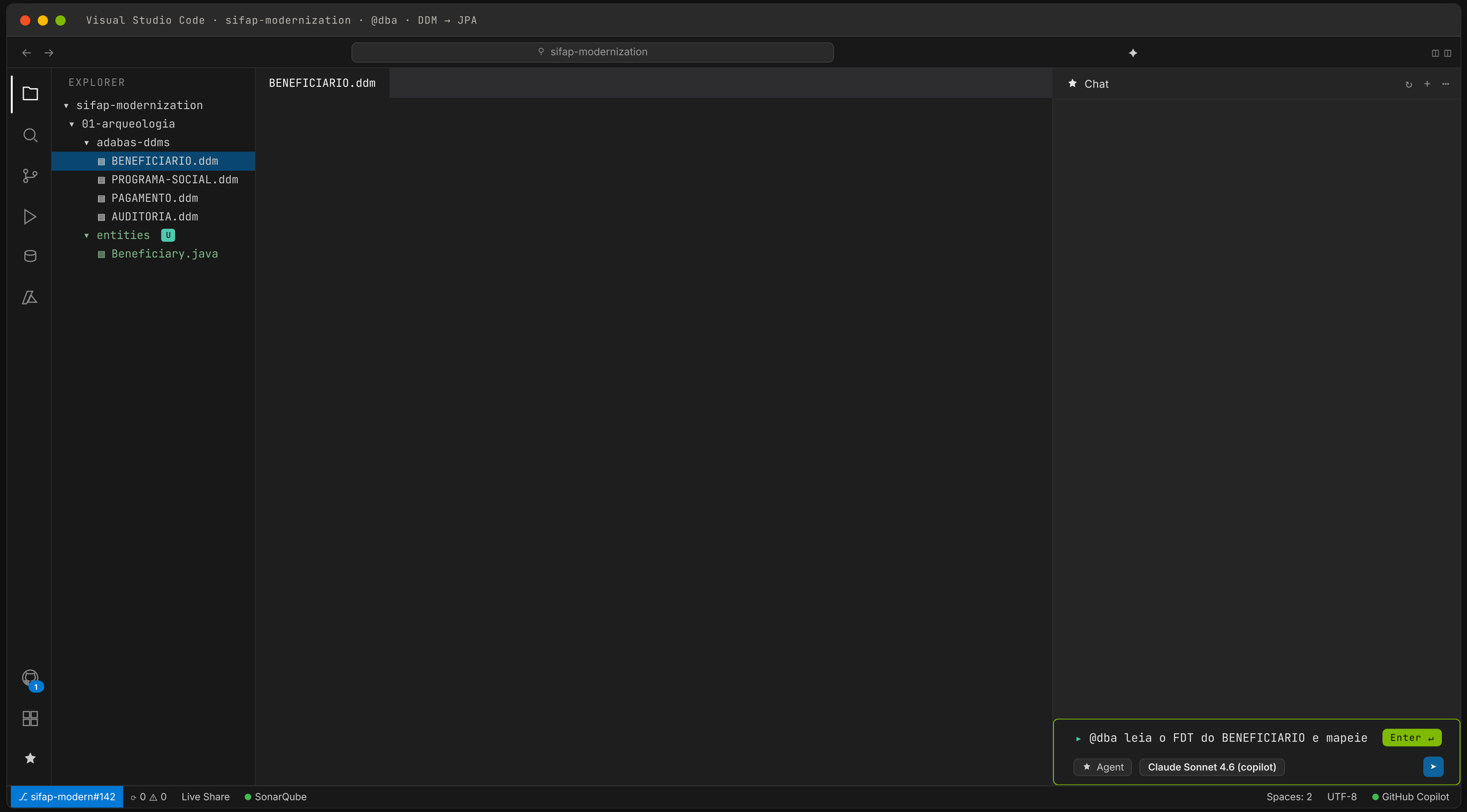
The periodic group of dependents is not a column, it becomes a separate table with an FK and a cardinality constraint.

02

The limit divergence (10/5/3) is born in the schema, not the code. The @dba exposes it, does not hide it.

03

FATOR-K shows up with ">>> NOT DOCUMENTED <<<" written into the FDT. It flags, it does not invent.


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-SECTION · WHAT THIS REVEALS

The database is the hardest artifact to translate, and the mysteries begin in the schema.

BEFORE

Adabas FDT: PE, MU, descriptors, packed decimals

AFTER

Relational schema + JPA, with every divergence traced

The @dba does not migrate the database blindly. It reads the FDT, recognizes what has no flat equivalent, and turns each strangeness into an explicit decision. FATOR-K and the dependent limit are not bugs to silently fix: they are mysteries that rise up for Spec-Driven to decide.

FROM DISCOVERY TO SPECIFICATION

187 rules discovered. Now, how do you turn them into something buildable without reintroducing ambiguity?

The answer is not to write code faster. It is to write the specification first, and let the agent work inside it.

VIBE CODING

"Build the benefit calculation." The AI generates code on the spot, guesses the architecture, skips the requirements. It works, but it does not match the legacy rule. 40% of the time turns into rework.

DETERMINISTIC

The CALCBENF.NSN rule becomes a testable EARS, with source_legacy. The agent only implements what the spec says. What comes out is verifiable against the legacy.

TWO PIECES, ONE LAYER

Spec-Kit is the method. Specky is the engine. They do not compete, they stack.

Spec-Kit

github/spec-kit · open source

The methodology. Defines the WHAT.

- EARS notation and the 6 requirement patterns
- Gated sequence: specify → clarify → plan → tasks → analyze → implement
- Constitution model and prompt templates the AI reads and follows
- Installs via the specify CLI; /speckit.* commands in GitHub Copilot

Lightweight. Ideal for learning SDD and adopting fast.

Specky

paulasilvatech/specky · CLI toolkit

The engine. Enforces the HOW.

- CLI toolkit: 13 agents, 22 prompts, 8 skills, 16 hooks, 57 MCP tools
- 10-phase state machine that blocks phase-skipping
- Schema-level EARS validator: a vague statement does not pass
- Cross-artifact analysis spec↔design↔tasks and compliance (HIPAA, SOC2, GDPR)

The AI is the operator. Specky is the engine.

Spec-Kit
The method · EARS + gates in GitHub Copilot

Spec-Kit turns the archaeology findings into traceable EARS.

WHAT TO WATCH

01

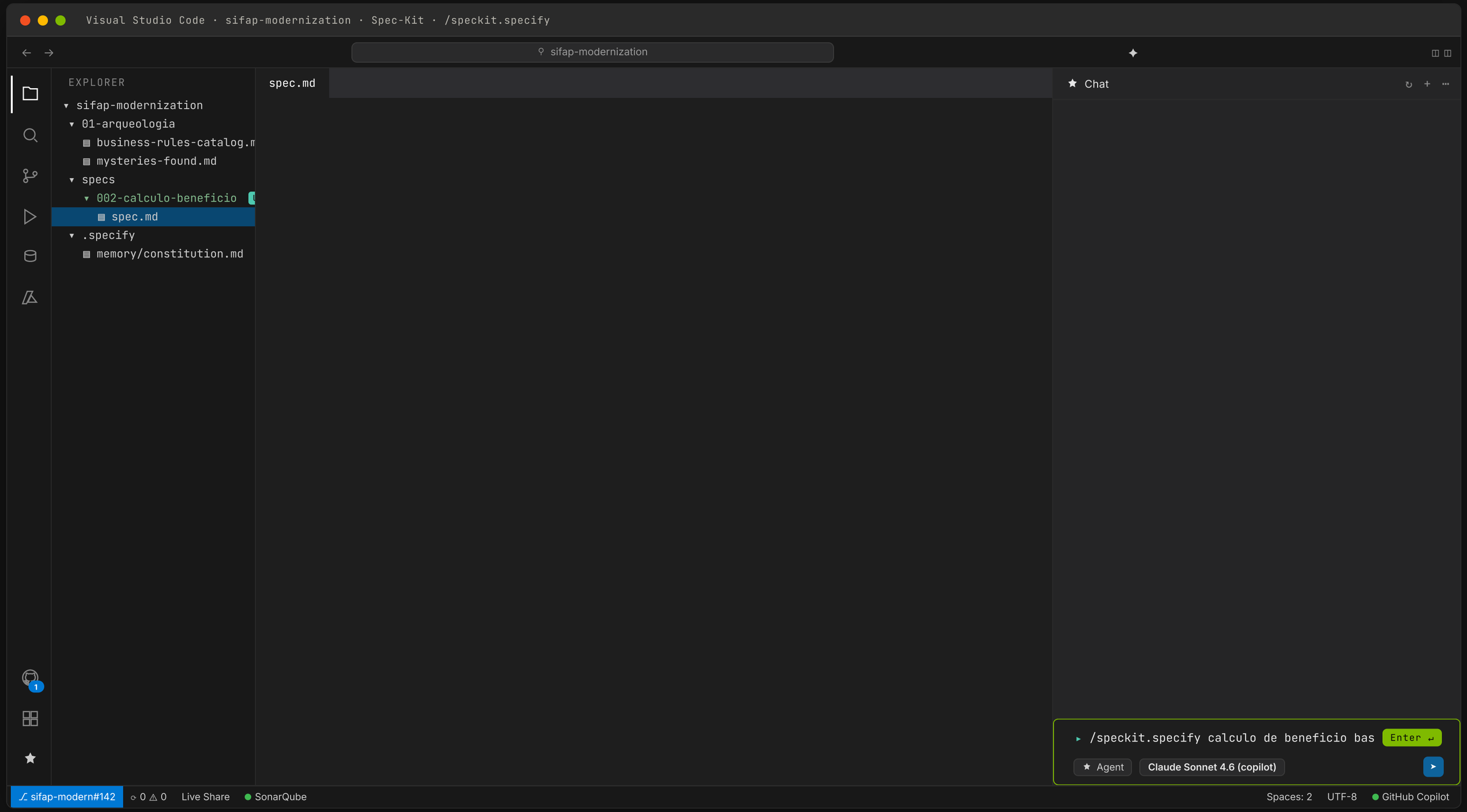
Spec-Kit is open source, from GitHub itself. The input is the @archaeologist artifacts.

02

The CALCBENF.NSN rule becomes REQ-CALC-001, with source_legacy pointing to the exact line.

03

What the AI does not know (the K-factor) it marks [NEEDS CLARIFICATION], it does not invent.


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

The engine · state machine + validation

Specky does not ask politely. It enforces the pipeline.

WHAT TO WATCH

01

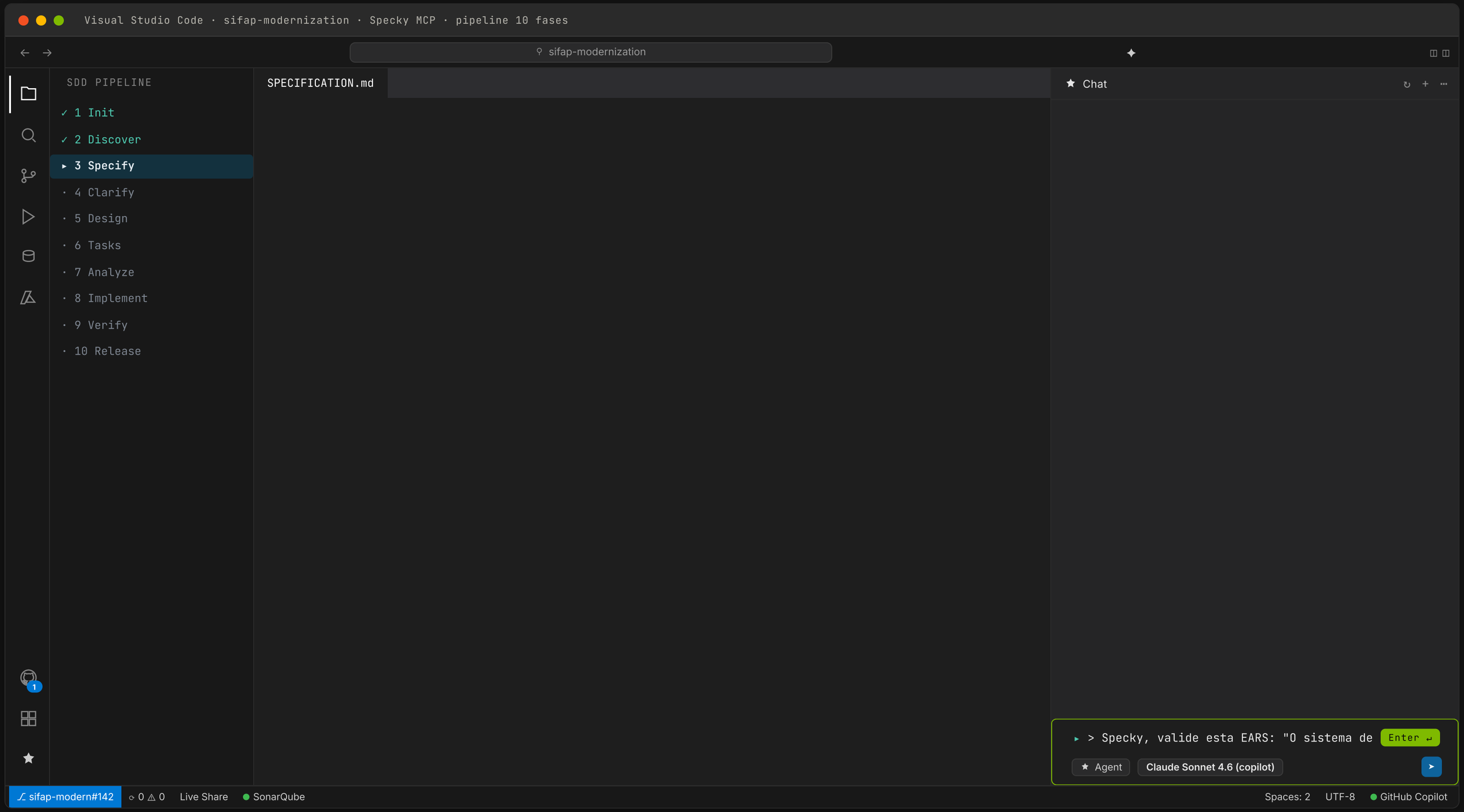
CLI toolkit: 13 agents, 22 prompts, 8 skills, 16 hooks, 57 MCP tools, installed via npm.

02

The EARS validator rejects a vague statement at the schema level, not an AI opinion.

03

The 10-phase state machine blocks phase-skipping. The shortcut that creates debt does not exist.


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-SECTION · WHAT THIS REVEALS

Method plus engine: the AI is the operator, Specky is the engine.

BEFORE

Spec suggested, gates optional

AFTER

Spec validated, phases mandatory

Spec-Kit provides the notation and the phases. Specky turns that into rules the machine enforces. The AI's creativity flows through a validated pipeline instead of guesswork.

A

@architect

Writes the spec · EARS + source_legacy

The @architect turns the findings into an auditable spec.

WHAT TO WATCH

01

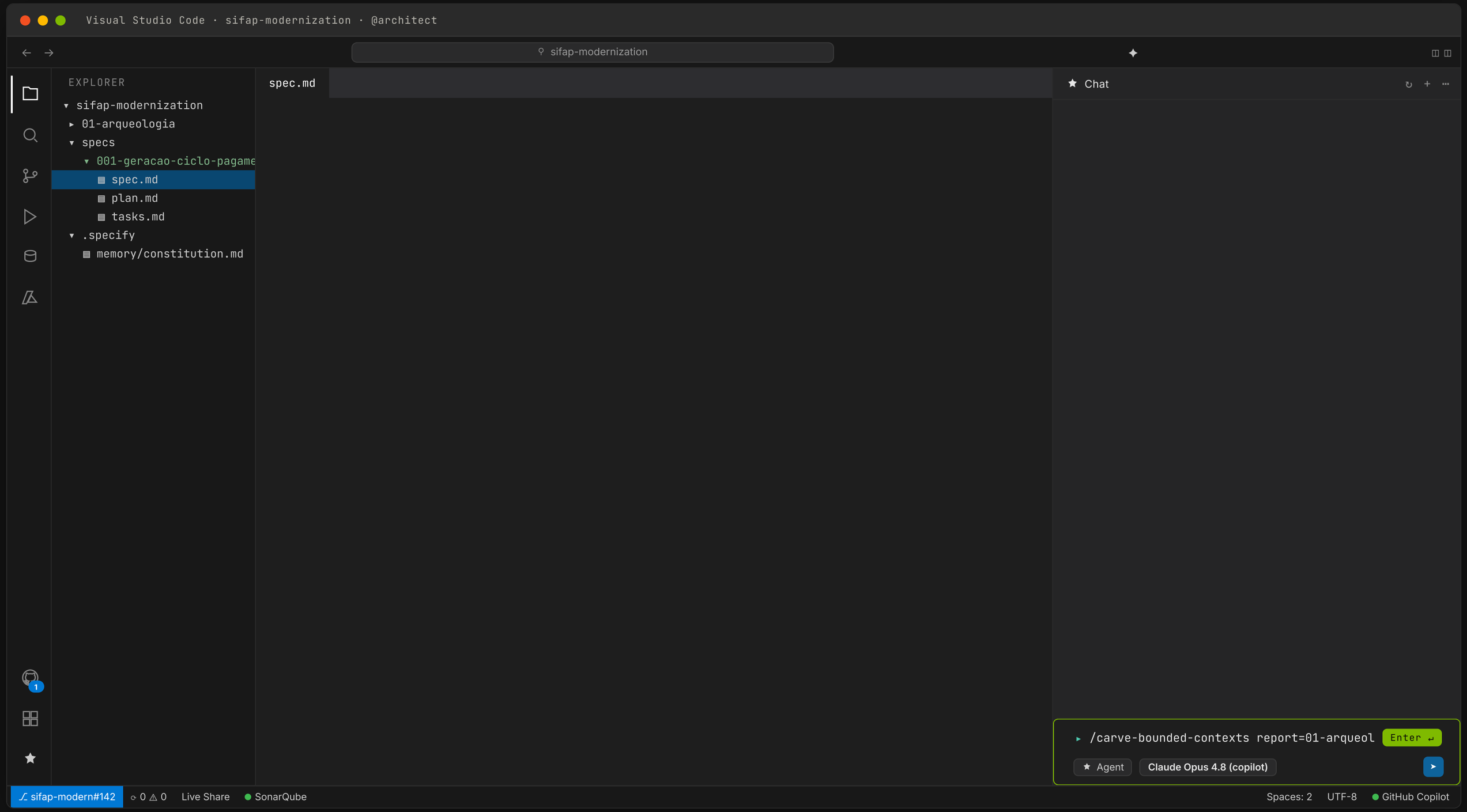
Spec-Kit (github/spec-kit) installs slash commands in GitHub Copilot. It is open source, not proprietary.

02

Input: the @archaeologist artifacts. Output: REQ-PAY-001 in EARS notation.

03

Watch the source_legacy pointing to BATCHPGT.NSN line 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

STAGE 2 · SPECIFICATION

Bounded Context Map · SIFAP 2.0

Produced by the @architect from discovery-report.md §7 and dependency-map.md. Target architecture: Modular Monolith (see ADR-001).

HYPOTHESIS EVALUATION

HYPOTHESIS	COHESION	COUPLING	CHANGE FREQ.	DECISION
A · Calculation + Payment as a single financial context	Low	Medium	Divergent	Rejected
B · Audit embedded in each context	High (if isolated)	High (if embedded)	Stable	Rejected
C · Single Registration for Beneficiary + Dependent + Program	High	Low (external)	Coherent	Accepted

Conclusions: separate Calculation and Payment (merging them would create a module with two reasons to change). Audit becomes its own context with a published write interface. Registration is the identity aggregate root.

FINAL BOUNDED CONTEXTS

CONTEXT	RESPONSIBILITY	OWNED DATA
1 · Registration	Lifecycle of beneficiaries, dependents, and programs; CPF and eligibility validation.	`BENEFICIARIO (150) · DEPENDENTE · PROGRAMA_SOCIAL (151)`
2 · Calculation	Base value, factors (regional, age, family, income, Fator-K), deductions (30% cap except judicial), IPCA correction.	`tabelas de config versionadas · resultados efêmeros`
3 · Payment	Monthly cycle, CNAB 240 remittance, triple-match reconciliation, status G/P/E/R/D.	`PAGAMENTO (152, ~180M reg, particionado)`
4 · Audit	Append-only trail of all actions; reports including exclusions (fixes MYS-010).	`AUDITORIA (153, append-only, particionada por ano)`

COMMUNICATION BETWEEN CONTEXTS

FROM	TO	MECHANISM	DATA
Pagamento	Cadastro	Synchronous call	Eligible beneficiaries per period
Pagamento	Cálculo	Synchronous call	(beneficiaryId, period) → calculated value
Cadastro	Auditoria	Domain event	Beneficiary insert/update/delete
Cálculo	Auditoria	Domain event	Batch (BT) calculation event
Pagamento	Auditoria	Domain event	Cycle generation + reconciliation

Rule: cross-context communication only via public interface or domain event, never direct access to another module's repository (see ADR-001, anti-corruption layer).

SCROLL TO SEE ALL 4 CONTEXTS ↓

DEMO 3 · WHAT THIS REVEALS

Without source_legacy it is a guess. With source_legacy it is engineering.

BEFORE

Requirement with no origin

→

AFTER

REQ-PAY-001 → BATCHPGT.NSN#L120

Every requirement of the new system points to the exact legacy line that originated it. Traceability becomes visible and auditable.

@builder
Translates to Java · semantic, not literal

The @builder translates Natural into idiomatic Java.

WHAT TO WATCH

01

Not a line-by-line port. It is semantic translation.

02

COMPUTE becomes BigDecimal with scale. MOVE becomes Optional. FIND becomes a repository query.

03

Every commit cites Implements REQ-XXX. The JUnit test is generated alongside.

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 · WHAT THIS REVEALS

The acceptance criterion is not that it compiles. It is that the equivalence test passes.

BEFORE

It compiles ✓

→

AFTER

Semantic equivalence ✓ · 7 tests

Line by line you get code that compiles. Semantic equivalence against the legacy gets you a system.

@evolution

Dispatches to the cloud · GitHub Copilot Coding Agent

The @evolution dispatches the issue. The GitHub Copilot Coding Agent runs in the cloud.

WHAT TO WATCH

01

Here we leave VS Code and enter GitHub web.

02

Well-structured issue: context, acceptance criteria, files to touch, explicit constraints.

03

Hand off to GitHub Copilot. The agent runs in the background and opens a draft PR.

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

@evolution /write-github-issue

STAGE 4 · EVOLUTION · FILLED BY THE TEAM

GitHub Copilot Agent Experience Report

Filled by the team at the end of Stage 4. Be honest: we want to learn what works and what does not. Forced positive reviews help no one.

1 · ISSUES CREATED AND PRS GENERATED

ISSUE	PR	FILES	TESTS	MANUAL FIX	MERGE
#142 Cap deductions at 30%	#312	4	Yes	No	After review
#143 Audit includes EX exclusions	#318	6	Yes	Minor	After review

2 · TEAM RETROSPECTIVE (IN THE TEAM WORDS)

Q1 Most useful agent: the @archaeologist, read 29 years of Natural without inventing a rule. Saved weeks of manual reading.

Q2 Most surprising failure: a generated test that passed without actually testing anything. Caught in review.

Q3 What I would change: more specific issues from the start. A vague issue produces a vague PR.

Q4 Production confidence: 7 of 10, rises with more equivalence-test coverage.

Q5 To take back: spec before code, always. The human gate is not optional.

3 · FAILURE MODES FOUND

✗	Incorrect or circular imports (hallucinated import)
✗	A test that passed without exercising the rule
✗	Scope creep, touched a file outside the request
✓	All caught in human review before merge

4 · PR QUALITY (SCORE 1-5)

CRITERION	SCORE
Code correctness	4
Architecture adherence	4
Test quality	3
Generated documentation	4
Overall average	3.8

SCROLL TO SEE THE RETROSPECTIVE ↓

DEMO 5 · WHAT THIS REVEALS

No merge without human review and a security check.

BEFORE

PR opened by the agent

AFTER

Merge only after human review + security

The cloud agent does not replace the developer. It removes the obligation to write the first version.

PART

V

The legacy, rebuilt.

The same Maria, the same 14 rules, now on Spring Boot and Next.js. Nothing was discarded: the legacy was understood, recorded and rebuilt.

DEMO 6 · THE DESTINATION

The same Maria, now on Spring Boot and Next.js.

01 Same CPF, same R$ 1,412.00, same eligibility rules.

02 CBENF020.NSP became BeneficiaryController.findByCpf().

03 The 14 rules covered by BeneficiaryServiceTest. source_legacy on every 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

Destination · Azure Database for PostgreSQL

The already-modernized database, live on Azure, and GitHub Copilot itself talks to it.

WHAT TO WATCH

01

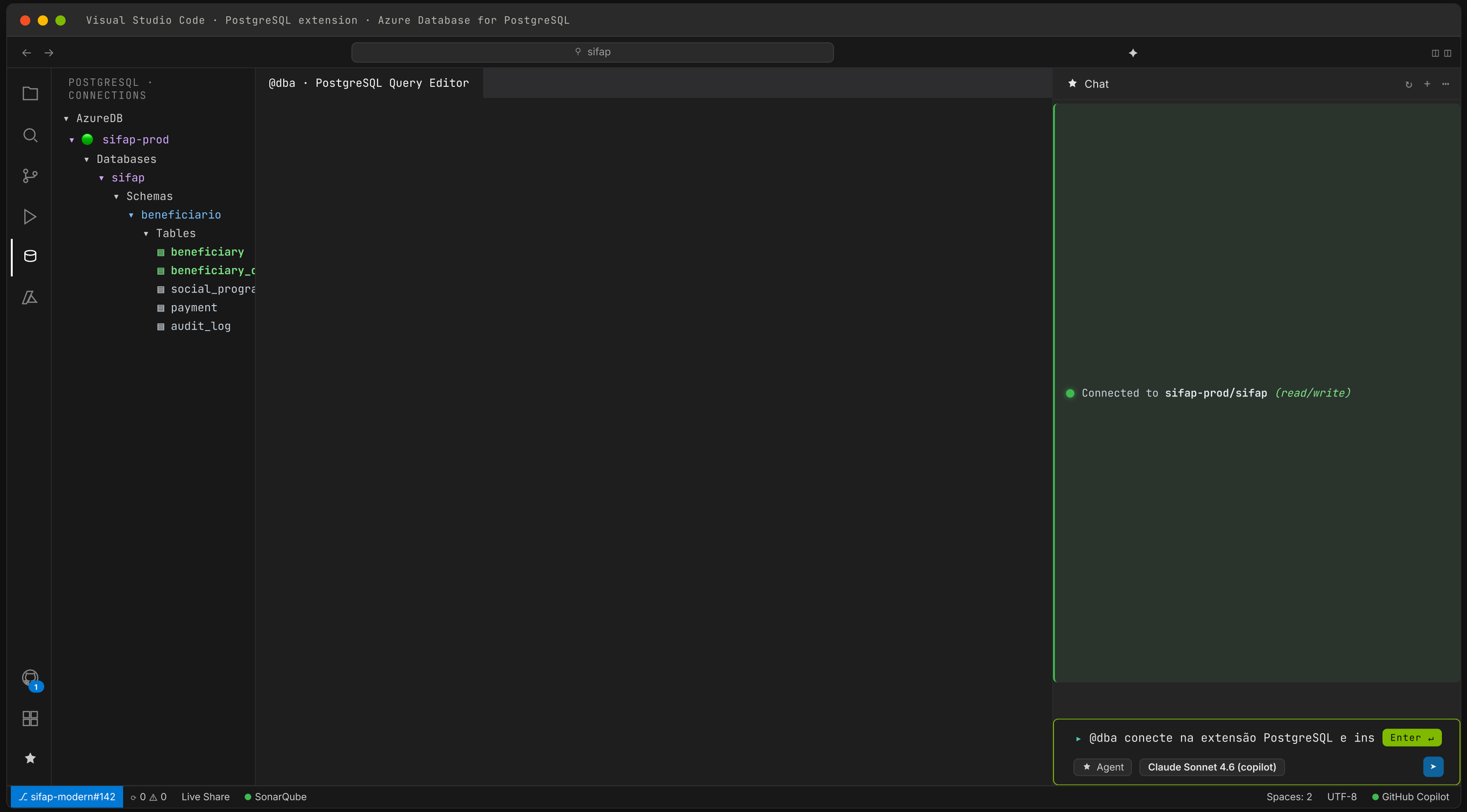
The PostgreSQL extension connects VS Code to Azure. The @dba operates the production database in natural language, without leaving the editor.

02

Clean relational schema: beneficiary and beneficiary_dependent linked by FK, visible in the CONNECTIONS tree.

03

And the @dba renders the full ERD through the extension: five tables in green, every FK drawn, read straight from the Postgres catalog.


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 · WHAT THIS REVEALS

The legacy was not thrown away. It was understood, recorded and rebuilt.

BEFORE

CBENF020.NSP · legacy

→

AFTER

BeneficiaryController · with tests

It is a different kind of modernization. There is a human signature on every layer, from requirement to test.

PART

VI

How to start.

Monday morning, with GitHub Copilot. Azure comes in as complexity grows. Start simple.

WHICH SLICE FIRST

Don't start with the most critical. Start with the slice that proves the method without breaking you.

The first slice is a bet on learning, not on value. Pick something small enough to finish in days, real enough to convince the team.

✓ CHOOSE IF

It has a clear, testable business rule (e.g. a discount calculation), few downstream consumers, and someone alive who knows the domain.

✗ AVOID IF

It's the whole payment engine, touches 180M records, or nobody can explain why it works. That comes later, once the method is proven.

→ IN SIFAP IT WOULD BE

CALCDSCT: the discount calculation. Isolated rule, a known divergence (30% cap vs judicial), and a golden master that's easy to build.

MONDAY MORNING, IN PRACTICE

Three horizons, real commands. GitHub Copilot is the starting point; Azure enters as complexity grows.

01 Week 1

Enable GitHub Copilot, bring up the 4 custom agents in a sandbox repo, and run @archaeologist on the first slice.

```
# 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
```

Output: **extracted rules + mysteries** for the slice.

02 Month 1

Install Spec-Kit, turn the findings into a signed spec, and put a minimum CI on 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
```

Output: **traceable spec.md** + green CI.

03 When it grows

With the method proven, integrate Azure as the slice demands: runtime, secrets, and observability.

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

Output: **slice in production**, observable.

WHAT KILLS THE PROJECT

Four traps sink legacy modernization. They all share one antidote: understand before you build.

- ✕ **Big bang: rewrite everything at once, alongside a legacy that keeps changing.**
  - → Slice it. One rule at a time, with a golden master against the legacy. Strangler fig, not full replacement.
- ✕ **Trusting code as the spec: "the system is the documentation, let's just translate it."**
  - → The code has 44 mysteries and 9 divergences. Each becomes a [NEEDS CLARIFICATION], not an assumption.
- ✕ **Vibe coding on legacy: asking the agent to "modernize" with no spec and no equivalence test.**
  - → Spec first, schema-validated. The agent only implements what the EARS declares, and the test proves parity.
- ✕ **Fixing bugs silently: "this is wrong in the legacy, I'll just fix it in the new one."**
  - → That "bug" may be a 20-year-old tax rule. Escalate the decision; preserve or fix with backing, never on a hunch.

HOW YOU KNOW IT WORKED

Modernization without metrics is faith. These four prove the slice is done, and carry the method beyond it.

100%

OF THE RULES WITH SOURCE_LEGACY TRACEABLE TO FILE:LINE

parity

ON THE GOLDEN MASTER: NEW AND LEGACY GIVE THE SAME RESULT, TO THE CENT

0

UNRESOLVED MYSTERIES TURNING INTO CODE, EACH BECOMES A DECISION OR A CLARIFICATION

↓

INTENT DEBT FALLING WITH EVERY SLICE, MEASURED AND VISIBLE ON THE PANEL

Which slice of your legacy would you take to @archaeologist first?

That is the slice that starts Monday morning.

CLOSING

The method is to understand before building.

CONTACT

Paula Silva

Software Global Black Belt · Microsoft

paulasilva@microsoft.com

NEXT STEP

One-slice workshop

One day, real team, real legacy, first slice running