TECHNICAL PORTFOLIO · AI-DIRECTED PROJECTS

Héctor Sulbarán

Senior .NET & AI-Augmented Developer

> Human-directed software engineering, executed with high-capability AI assistants. Six projects in real production — from humanitarian crisis response in under 24 hours to AI governance over a 482-project enterprise .NET monorepo.

6	<24h	99.7%	3×
projects in production	minimum time to production (Faro)	latency reduction (HorasBot)	commit velocity (NetSuite)

Email hectorsul26@gmail.com LinkedIn linkedin.com/in/hsulbaran Web hsulbaran.dev

WhatsApp +58 412-2004053

METHODOLOGY NOTE

Human-led engineering.
AI-augmented execution.

All six projects share a single axis: software engineering **directed by a human**, executed with high-capability AI assistants. None of these was undisciplined "vibe coding" — the repeated pattern is context, constraints, and verification before execution.

01

Living context document before code

Brief, spec, constitution, plan, brand voice, CLAUDE.md — never a loose prompt. The document is the source of truth that survives any working session.

02

Mandatory Plan Mode before implementing

File tree and phases proposed by the agent, approved by the human before touching a single line of code. Catches gaps before they become costly.

03

Phase-by-phase execution with human verification

Never full generation in one shot. Explicit acceptance criteria per step, verified in production or localhost — never just reviewing code in the abstract.

04

Explicit boundary of what is never delegated

Architecture and stack, security, product decisions, third-party relationships, phase approval. The agent executes — the human decides.

05

External dependencies never block critical functions

If something outside the team's control can take down the most critical function, the design is wrong. Detected and corrected in active production (Faro).

06

The already-installed channel wins

Telegram, WhatsApp, or Slack reduce friction when they already belong to the user's workflow. Architecture adapts to real behavior instead of forcing a new interface.

07

Governance that prevents errors

Scoped permissions, traceability, guardrails, and explicit approvals turn technical judgment into operating rules. The system does not depend on the model remembering to behave correctly.

→

What was never delegated to AI

Architecture · stack · security · product decisions · phase approval · risk management · client and team relationships · final visual and functional validation.

HUMANITARIAN PLATFORM · 2 DEPLOYMENTS · 2 COUNTRIES

Faro

**Same architecture. Two earthquakes.
Under 24 hours each.**

<24h	750+	89%	<3s
each deployment — 2 real emergencies	people reached (Venezuela + Colombia)	emergencies field-verified	bot response time

CONTEXT AND PROBLEM

Venezuela (June 2026): Two consecutive earthquakes (7.2 and 7.5 Mw, Yaracuy) left 920+ dead. An open-source humanitarian index (RedAyuda) existed but was active-query only. There was no low-friction input channel for citizens or automatic output toward rescuers.

Colombia (August 2026): 7.4 Mw earthquake in Chocó. The same architecture was adapted in under 24 hours incorporating lessons from the first emergency — eliminating by design the single point of failure detected in production.

> **Key product decision:** Telegram over a new web app. A bot inside an already-installed app eliminates half the friction steps. Measured: /search responds in <3 seconds vs 2–3 minutes on a traditional website under the same unstable connectivity.

FARO VENEZUELA V1

- Conversational layer over RedAyuda API
- 3 functions: search, report, alert channel
- 6 report types with manual state machine
- Rate limiting: 3 reports/hour per user_id
- Hotfix: decoupled alert channel from external dependency

FARO COLOMBIA V2 — IMPROVEMENTS

- **Zero external data dependencies** — autonomous architecture
- Gate: 5 official emergency numbers before rescue report
- Automatic masking of sensitive data
- Explicit public privacy policy
- 10 E2E test cases before launch

STACK

`Node.js` `TypeScript` `grammy` `Redis Upstash` `Vercel` `Telegram Bot API` `Claude Code`

INTERNAL ENTERPRISE AUTOMATION · NETACTICA

HorasBot

15 minutes of manual work turned into 3 automatic seconds.

99.7%	15min→3s	98%	100%
operational latency reduction	reporting time per session	data accuracy in Azure DevOps	team adoption from day 1

PROBLEM

The team logged hours daily in Azure DevOps — a 15-minute manual process per person with systematic Work Item association errors and enough friction to cause recurring registration delays.

SOLUTION — AGENT ARCHITECTURE

Multi-step pipeline: **Slack → Claude Code → MCP Protocol → Azure DevOps**. The developer writes in natural language in Slack. HorasBot interprets, validates available Work Items, registers in Azure DevOps, and confirms in the same thread.

> **Key architecture decision — hybrid, not all-AI:** 80% of daily interactions are resolved without invoking a language model — using direct Azure DevOps API calls. AI reasoning is reserved for interpreting ambiguous natural language.

MAIN FLOW

- Natural language command in Slack
- Active sprint Work Item validation
- Interactive confirmation before registering
- Registration in Azure DevOps via API
- Confirmation in the same Slack thread

WHAT IT DEMONSTRATED

- Operational AI in a real enterprise workflow
- Frictionless adoption: already-installed channel (Slack)
- Hybrid architecture: deterministic where it applies
- MCP Protocol as a stable integration layer

STACK

`Node.js` `TypeScript` `Slack Bolt SDK` `Claude Code` `MCP Protocol` `Azure DevOps API` `Vercel`

OWN PRODUCT · PERSONAL FINANCE · PROTO-SAAS

Ratio

**Eliminated spreadsheets and manual calculations.
Zero errors since launch. 3 companies want to adopt it.**

0	3	120+	6
errors since launch — Excel eliminated	companies in B2B pipeline	automated tests	sub-agents with scoped permissions

CONTEXT

Personal financial control system built with **Spec-Driven Development (SDD)**: formal specification before code. Has fully replaced manual spreadsheet review in daily production use since launch. WOI 2469 and 2 additional companies have expressed interest in adopting it as a scalable B2B platform.

MULTI-AGENT ARCHITECTURE — 6 SPECIALIZED SUB-AGENTS

- **Ingestion agent:** processes transactions from Telegram
- **Categorization agent:** classifies expenses with rules + AI
- **Validation agent:** verifies data integrity
- **Calculation agent:** executes financial operations
- **Reporting agent:** generates views and summaries
- **Spec-guardian (read-only):** prevents architectural drift

> **Critical design decision — read-only spec-guardian:** an agent with read-only access to constitution.md and spec.md. Verifies every proposed change does not violate the formal specification. Makes architectural drift structurally impossible without depending on the model "remembering" the constraints.

CURRENT STATUS AND ROADMAP

In daily production use. **Next phase:** metrics, statistics, advanced filters, and data visualization. Architecture designed to scale to multi-tenant B2B without refactoring.

STACK

`Next.js 15` `TypeScript` `Prisma` `PostgreSQL` `Supabase` `Telegram Bot API` `Claude Code` `MCP Protocol`

FREELANCE CLIENT · B2B/B2C CONVERSION

WOI 2469

4 B2B distributors in 60 days.
Delivered in 3 days. Lighthouse ≥95. $0/month.

4	10×	3 days	≥95
B2B distributors contacted in 60 days	faster than a typical agency timeline	≈12 hours of total engineering	Lighthouse · $0/month infrastructure

B2B/B2C landing page for a Venezuelan gourmet brand. Dual architecture in a single URL — wholesalers and end consumers via WhatsApp. No backend, no database. A structured asset intake process eliminated the revision cycles that delay 80% of freelance projects.

`Next.js 15` `TypeScript` `Tailwind CSS v4` `Framer Motion` `Vercel` `Claude Code`

PERSONAL BRAND · CONVERSION ASSET

hsulbaran.dev

Bilingual senior portfolio with documented operational AI — Lighthouse ≥90 — $0/month.

≥90	ES/EN	<3w	$0
Lighthouse across all categories	native bilingual — Context API	full development F1→F6	monthly infrastructure cost

`Next.js 15` `TypeScript strict` `Tailwind v4` `Framer Motion` `EmailJS` `Vercel` `Claude Code` `Claude in Chrome`

ENTERPRISE INITIATIVE · NETACTICA (CURRENT ROLE) · CASE 1 OF 2

NetSuite

AI governance over a modern 482-project .NET monorepo in active production.

1–8→18–23	482	125	3
commits/month after adopting CLAUDE.md	.csproj projects governed	active external integrations	.NET generations coexisting

CONTEXT

Travel booking platform with microservices architecture: **482 .csproj projects**, 125 external integrations (airlines, hotels, car rentals, GDS, payment gateways) and three technology generations (.NET 8, Standard 2.0, Framework 4.8.1) coexisting in active production. Git version control over Azure DevOps with CI/CD.

> **CLAUDE.md as institutional memory:** each of the 8 functional solutions has its own CLAUDE.md documenting local architecture, critical constraints, resolved incidents, and security guardrails. System knowledge stops living in one person's memory and becomes a versioned team asset.

GOVERNANCE RULES

- Mandatory commit ↔ Azure DevOps Work Item traceability
- No direct push to develop branch
- Explicit authorization before each commit
- Branch naming convention by change type

SENSITIVE LOGIC FIXES IN PRODUCTION

- Loyalty point consistency in multi-pax Extras
- Optimizer mode detection in mixed Hotel + Car payments
- Hardcoded templateType breaking notifications
- Centralization of voucher delivery

STACK

`.NET 8` `.NET Standard 2.0` `.NET Framework 4.8.1` `C#` `Azure DevOps` `SQL Server` `RabbitMQ` `Git` `Claude Code`

ENTERPRISE INITIATIVE · NETACTICA (CURRENT ROLE) · CASE 2 OF 2

NetOffice

The same governance over a legacy ERP most would consider untouchable.

18	0	5	37
.sln solutions governed	production regressions after adoption	countries with fiscal logic in production	projects in Netoffice.sln alone

CONTEXT — THE HARD SCENARIO

Desktop ERP for travel agencies covering the full business cycle: GDS booking (Amadeus, Sabre, Worldspan) → client file → electronic invoicing with fiscal authorities (AFIP Argentina, DIAN Colombia) → accounting → credit card payment reconciliation → cloud archiving. Operates across **Argentina, Mexico, Uruguay, Colombia, and Brazil**, each with its own fiscal integrations and configuration profiles.

> **Why this case matters more:** governing AI agents over a modern Git monorepo is the comfortable scenario. Doing it over WinForms + .NET Framework 4.5 + WCF + Crystal Reports, with TFVC version control lacking Git integration and no CLI build, is where most developers conclude AI doesn't apply — and where most real enterprise software in the world actually lives.

TECHNICAL CONSTRAINTS OF THE ENVIRONMENT

- 18 independent .sln solutions
- WinForms on .NET Framework 4.5
- All business logic flows through WCF
- Crystal Reports 13 for PDF generation
- TFVC — no native Git integration
- No CLI build — requires Visual Studio

ADOPTION RESULTS

- **Reduced onboarding time** — context lives in CLAUDE.md
- **Previously untouchable legacy code, now modifiable with confidence**
- More commits in less time
- No production regressions reported
- Domain-specialized skills for the ERP

WORKFLOW ADAPTED TO TFVC

Development, research, and analysis operate under CLAUDE.md and skills with Claude Code. Version control is executed manually against TFVC — an environment constraint resolved by separating the assisted generation phase from the versioning phase, without compromising traceability.

STACK

`.NET Framework 4.5` `C#` `WinForms` `WCF` `Crystal Reports 13` `SQL Server` `AWS S3` `TFVC` `Claude Code` `Skills`

AI GOVERNANCE · SYNTHESIS OF BOTH CASES

A framework validated at both ends of the technical spectrum.

Most case studies on AI agents in engineering are done on modern stacks and clean repositories. The real value of a governance framework is proven when it works equally well in the scenario nobody chooses.

DIMENSION	NETSUITE	NETOFFICE
System type	Microservices monorepo	`Modular monolithic desktop ERP`
Scale	482 .csproj projects	`18 .sln solutions`
Stack	.NET 8 / Standard 2.0 / 4.8.1	`.NET Framework 4.5 + WinForms + WCF`
Version control	Git + Azure DevOps + CI/CD	`TFVC without Git integration`
Build	CLI available	`Requires Visual Studio`
Geographic scope	125 GDS and provider integrations	`5 countries with distinct fiscal logic`
Impact metric	1–8 → 18–23 commits/month	`0 regressions · reduced onboarding`

What doing both proves: the governance framework does not depend on the stack, the modernity of the repository, or the availability of CLI tooling. It depends on documenting context, defining explicit constraints, and requiring human verification at every step. That is transferable to any system — including the one nobody wants to touch.

FRAMEWORK COMPONENTS

- **CLAUDE.md per solution** — architecture, constraints, incidents, guardrails
- **Specialized skills** — packaged, reusable domain knowledge
- **Explicit prohibitions** — what is never done, documented before generating code
- **Human authorization per commit** — no exceptions

THE DIFFERENCE FROM TYPICAL USAGE

- Most use Claude Code to **write code faster**
- This work uses it so that **system knowledge** becomes a versioned team asset
- Knowledge stops living and dying with one person or a single working session

CROSS-PROJECT SUMMARY

One method.
Six metrics that prove it.

PROJECT	TYPE	DELIVERY PRESSURE	HEADLINE METRIC
Faro	Humanitarian response	Extreme · <24h to production	750+ · 2 countries · <24h · 0 deps (v2)
HorasBot	Enterprise automation	Medium · daily team adoption	15 min → 3 s · 99.7% · 98%
Ratio	Own product · proto-SaaS	None · full SDD	0 errors · 120+ tests · 3 B2B
WOI 2469	B2B/B2C freelance landing	Medium · client delivery cycle	4 B2B · 3 days · 10× · ≥95
hsulbaran.dev	Personal brand	Low · fine visual iteration	Lighthouse ≥90 · $0/mo · bilingual
AI Governance	AI governance · 2 platforms	Sustained · active production	482 + 18 · 3× commits · 0 regressions

What was never delegated to AI in any project: architecture and stack selection · product and security decisions · explicit approval of each phase or commit · risk management and third-party relationships · final visual and functional validation · definition of non-negotiable constraints.

Héctor Sulbarán

Senior .NET & AI-Augmented Developer

Portfolio 2026

hectorsul26@gmail.com · linkedin.com/in/hsulbaran · hsulbaran.dev Architecture first. AI in the loop.