

Envoy

The AI-driven personal website behind luism.pro — a system built from the ground up to demonstrate agentic orchestration, RAG, and production AI-engineering practice, not just describe it

Prepared September 2026 · Live system, actively deployed

1. Executive Summary

What it's for. A CV or portfolio website is normally a static page: the same text, in the same order, for every visitor, whether they're skimming for keywords or looking for architecture details. Envoy exists to close that gap — instead of asking every visitor to dig through the same fixed layout, it lets them simply ask what they want to know, in their own words, and get an answer built from the actual question rather than a one-size-fits-all page. Beyond that, Envoy is deliberately built to be seen, not just used: the same engineering — retrieval, orchestration, guardrails, monitoring, graceful failure handling — that answers a visitor's question is also exposed for inspection, so a visitor evaluating Luis's engineering ability can watch it work rather than take a description of it on faith.

What it is. Envoy is Luis's personal website at luism.pro, built as two layers rather than one page. Underneath is a conventional static site — bio, CV download, portfolio — hosted independently on its own cloud infrastructure so its uptime doesn't depend on anything else. On top of it, when everything is healthy, sits a conversational AI layer that becomes the default way visitors explore the site: instead of a menu, a chat box invites questions, and the site's content reorganizes itself around the conversation. Any visitor can turn on a "show technical detail" toggle to see the mechanics behind each answer — that toggle is the only distinction the platform makes; it doesn't detect or route visitors by type. Unlike Luis's other portfolio pieces, which are tools and integrations each built to solve one problem, Envoy itself is the demonstration: a system built specifically to show engineering practice, with Data Express and Data Sentinel embedded directly inside it as working proof.

What it does. A visitor lands on the site, is greeted, and can ask about Luis's career history, working style, or any specific project in his portfolio; the assistant answers using only vetted facts about him. A visitor can also paste a job description and get a structured match analysis against Luis's actual profile, open and try the Data Express and Data Sentinel demos directly inside Envoy's own interface, request the CV as a download, or send a message that reaches Luis by email — all while the site quietly records anonymized engagement data so Luis can see what visitors actually cared about.

2. Technical Summary

What it does, technically. Envoy is a retrieval-augmented generation (RAG) system layered over a standard web frontend. Every visitor message triggers a similarity search over a curated, hand-written knowledge base about Luis (career, skills, projects, the system itself), and only the facts that search surfaces are handed to a language model along with the visitor's question — the model is never asked to answer from general knowledge or memory, which keeps its answers grounded and reduces the chance of it inventing something about Luis that isn't true.

How it does it. The dynamic layer runs as a React/Next.js application that proxies every dynamic request (chat, job-match, contact) to an n8n automation backend, with the whole dynamic stack currently running in Docker on Luis's own infrastructure during active development, reachable only through a Cloudflare Tunnel — no ports opened to the public internet. The static site is deliberately not served from that same infrastructure: it's hosted independently on its own cloud host, so a home-infrastructure outage degrades only the AI layer, never takes the whole site down. On the dynamic side, n8n first triages each message to the right specialist workflow (about Luis, about the system itself, about the portfolio, job matching, or contact), and that specialist embeds the query with a Gemini embedding model, searches a Postgres/pgvector index for matching knowledge-base chunks, and generates the reply with a Gemini language model. Redis handles chat rate-limiting and remembers a visitor's language; MinIO stores the RAG knowledge-base documents themselves; Uptime Kuma watches every service.

How it was built. The retrieval logic went through several rounds of tuning after early testing showed it under- or over-retrieving: the similarity threshold was recalibrated, a priority/keyword-weighted re-ranking step was added on top of raw vector similarity, and a related-chunk expansion step was introduced so an answer can pull in adjacent context the raw search missed. All of the orchestration — routing, retrieval, generation, guardrails — lives in visually-built n8n workflows rather than hand-written backend services. Every one of these design decisions was made to be inspectable as well as functional: the site's own "about this system" specialist and technical-detail toggle exist specifically so a visitor doesn't have to trust a claim about how it works — they can ask it, or watch it, directly.

3. Business-Scope Documentation

The problem it addresses

A static CV page asks every visitor to do the work of finding what's relevant to them. Recruiters skim; technical reviewers dig; a hiring manager evaluating fit for a specific role wants a direct answer, not a document to interpret. A conventional site cannot adapt to any of that — it shows the same content regardless of who's reading or what they came for.

More than adaptive content — an engineering showcase

Envoy is intentionally different from the rest of this portfolio. Data Express and Data Sentinel are tools and integrations, each built to solve one problem well. Envoy is a system, built from the ground up specifically so a visitor — especially one evaluating Luis for an AI Automation Engineer or Architect role — doesn't just read about his engineering practice, but can watch it operate directly: agentic orchestration, retrieval-augmented generation, security guardrails, safety and latency monitoring, information traceability, system integration,

self-reference, and graceful degradation. Each of these is a deliberate demonstration, and each maps to a value Luis designs toward on every project:

ENGINEERING ELEMENT	VALUES IT DEMONSTRATES
Agentic orchestration — guardrail, triage, and specialist routing in n8n	Efficiency, Relevance, Optimization of Results
Retrieval-augmented generation, grounded in a curated knowledge base	Relevance, Auditability, Traceability
Security guardrails — no exposed ports, Cloudflare Tunnel only, rate limiting	Security
Safety and latency monitoring — every service watched, every turn timed	Reliability, Transparency
Information traceability — a live per-turn trace of route, guardrail verdict, latency, and models used	Traceability, Transparency, Auditability
System integration — Data Express and Data Sentinel embedded directly inside Envoy, not just linked to	Integration, Experience
Self-reference — Envoy can show its own real workflows and RAG documents, not a generated summary	Transparency, Auditability, Experience
Graceful degradation — the static site keeps working if anything dynamic fails	Reliability, Experience
Cost-awareness — zero API spend on bots/crawlers, rate-limited chat, tiered model use	Cost-Awareness, Efficiency

A single distinction, chosen by the visitor

Every visitor talks to the same assistant, in their own words — the platform does not detect or route visitors by type (recruiter vs. technical, for instance). The one adjustable setting is a "show technical detail" toggle any visitor can turn on themselves, which adds a live trace to each answer — the route taken, the guardrail's verdict, latency, and which models were used — without changing anything else about the conversation. It's an opt-in window into the machinery, not a separate experience.

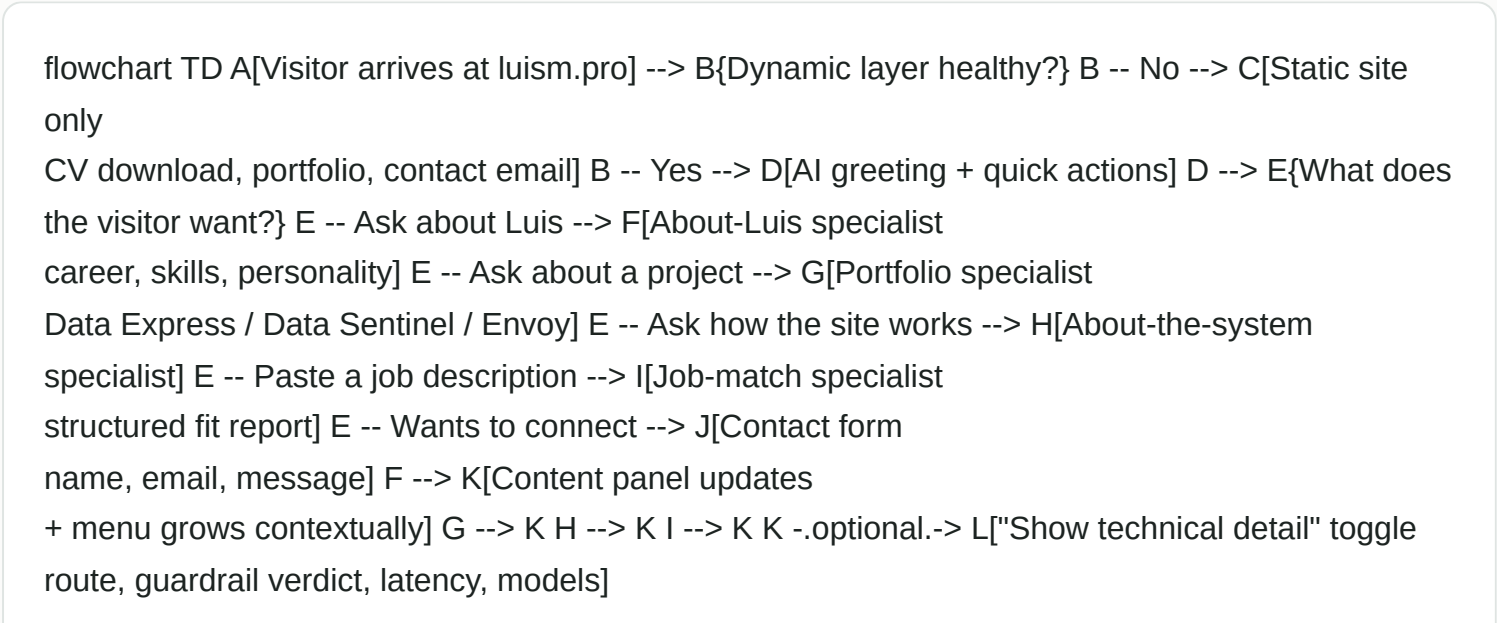


Fig. 1 — Visitor journey: static fallback on the left branch, AI-driven routing on the right, one optional detail toggle throughout.

What the site covers

SECTION	CONTENT
Me	Profile, career timeline, education, certifications, working style
Portfolio	Data Express, Data Sentinel, Envoy itself — each with its demo embedded directly in the interface
About the System	How Envoy's own AI layer is built, including its real workflow diagrams and RAG source documents
Job Match	Paste a job description, get a fit analysis against Luis's real profile
Analytics	A visitor's own session, summarized in plain language on request
Contact	Direct contact form (name, email, message), emailed to Luis with a confirmation copy to the visitor

Reliability as a business decision

The static layer is not a fallback bolted on as an afterthought — it is the deliberate baseline, and the reliability decision goes deeper than frontend logic. The system is not hosted from a single point: the static site runs on its own cloud infrastructure, independent of the Docker stack that runs the dynamic AI layer, so an outage on the AI side never takes the whole site down. If the AI layer, the database, or any backend service is unavailable, the site keeps functioning: the CV downloads, the portfolio is readable, contact information is visible. Visitors can also choose the static version voluntarily via a header link, even when the AI layer is healthy. A personal site going fully down during exactly the moment a recruiter checks it is the failure mode this design exists to prevent.

4. Technical-Scope Documentation

Hosting and deployment

The dynamic AI layer — n8n and everything it depends on (Postgres/pgvector, Redis, MinIO, the monitoring stack) — runs from one Docker Compose stack, started with one command and exposed publicly only through a Cloudflare Tunnel; n8n itself has no direct public route and no ports are opened on the host. This stack currently runs on Luis's own infrastructure while the system is under active development.

The static site is deliberately not hosted from that same single point — a system with one host for everything would mean one outage takes down the whole site, which is a failure mode this architecture is built to avoid. The static site is served independently from its own cloud hosting, so its availability doesn't depend on the home infrastructure running the dynamic layer: if that Docker stack is ever unreachable, the static site stays up on its own, and only the AI overlay degrades to the fallback behavior described below.

Services

SERVICE	ROLE
React / Next.js frontend (dynamic layer)	Serves the AI overlay, proxies dynamic calls to n8n, and embeds both other portfolio projects — currently Docker-hosted on Luis's own infrastructure, separate from the static site's own cloud hosting
n8n	Orchestrates every dynamic behavior — triage, retrieval, generation, contact, job-match, analytics
Postgres + pgvector	Vector store for the knowledge base; analytics storage (bronze/silver/gold schema)
Redis	Chat rate-limiting and per-visitor language-hint caching
MinIO	Object storage for the RAG knowledge-base documents themselves
Uptime Kuma	Visual monitoring dashboard across all services
cloudflared	The only service reachable from the public internet

Retrieval and generation pipeline

The knowledge base is a set of markdown documents about Luis, chunked at heading boundaries and tagged per-chunk with category, priority, keywords, and related-chunk references. On a query, the flow is: embed the query (Gemini) → pgvector similarity search across the full base → re-rank candidates using each chunk's own priority/keyword/category metadata as a soft boost on top of raw similarity → accept the result only if the top (unboosted) score clears 0.60 and at least two chunks qualify → optionally expand into a capped set of related chunks by cross-reference → generate the answer (Gemini) grounded in exactly that retrieved context.

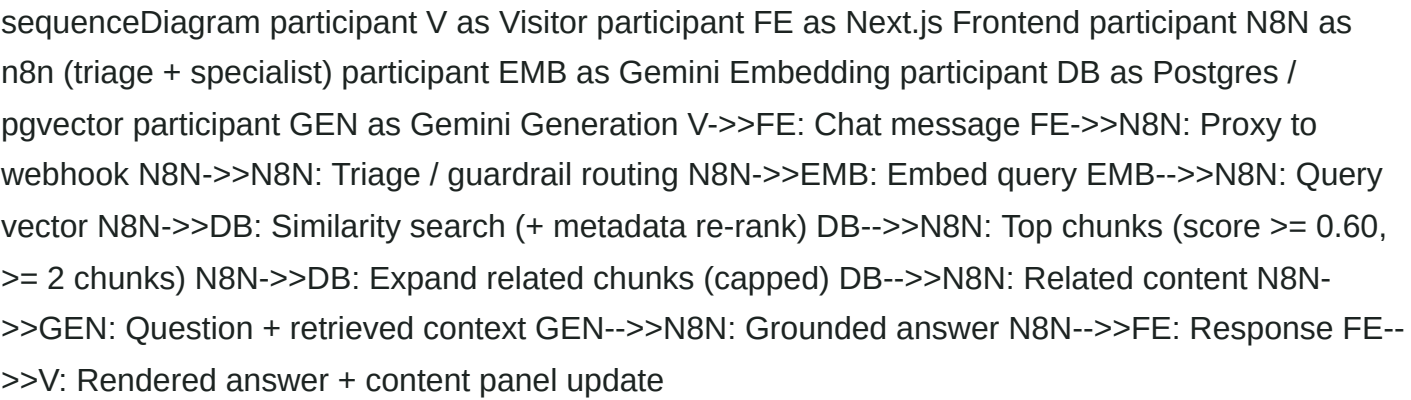


Fig. 2 — Chat query data flow, from message to grounded answer.

Cross-project integration

Both other portfolio projects are reachable from inside Envoy's own interface, not linked out to separately — this is the "system integration" element shown in Section 3, not just claimed. Data Express is embedded as a native port of its own demo widget, built directly into Envoy's React frontend, with two modes: a manual mode that calls the underlying tool functions directly with no model involved, and an agent mode using the

same Gemini function-calling flow as its standalone demo — both operating on the same in-memory session dataset, reached through this app's own Next.js server proxying to the Data Express backend container at `/api/demo/chat`, `/api/demo/tool`, and `/api/demo/health`. Data Sentinel runs as a fully separate application — its own container, its own Next.js build — but is reverse-proxied by this same frontend at `/data-sentinel/*` and embedded via an iframe pointed at that same-origin path. Neither demo needs its own public hostname or a separate Cloudflare Tunnel route.

Self-reference

Envoy's own portfolio entry doesn't describe the system in prose alone — it shows the real artifacts. An image carousel displays the actual screenshots of the live n8n workflows (guardrail, triage, each specialist, ingestion, admin flows), and a document list shows the real RAG source files, grouped exactly the way the specialists' own retrieval scoping groups them — not a generated summary standing in for the real thing. Combined with the per-turn technical trace available via the detail toggle (route, guardrail verdict, latency, models), a technical visitor can verify how the site works using the site itself as the evidence.

Other data flows

- **Analytics:** frontend events (section viewed, time spent, CV downloaded) go to n8n, which resolves visitor country from the Cloudflare-supplied header (no separate geolocation call), stores to Postgres, and can generate a plain-language session summary on request via Gemini.
- **Contact:** visitor fills a plain form (name, email, message) → n8n validates the submission, stores it, and sends two emails in parallel: a notification to Luis and a confirmation copy to the visitor. No AI drafting is involved in this flow.
- **Job match:** a pasted description or URL (n8n fetches the URL, with a paste-text fallback since many job sites block automated fetches) goes to Gemini together with Luis's profile from the RAG store; the result is framed to acknowledge gaps rather than oversell.

Degradation behavior

On page load, the frontend runs a same-origin health check with a 2-second timeout. If it fails, the site falls into graceful fallback: static content stays fully visible, and the chat box shows an "AI assistant offline" message with a direct email link — no broken interface, no dead ends.