



Vladyslav Vagin

FULLSTACK WEB DEVELOPER & AI AUTOMATOR

I build modern web platforms in Next.js, NestJS and FastAPI, and ship agentic AI systems, RAG pipelines and end-to-end automations to production. 10 highlighted projects inside — fullstack platforms, AI integrations, voice interfaces, and automation pipelines.

EMAIL

vladyslav1991devjs@yahoo.com

LINKEDIN

[/in/vladyslav-fullstack](https://www.linkedin.com/in/vladyslav-fullstack)

BASED IN

Oviedo, Asturias • CET

What's inside

Selected projects shipped to production — fullstack platforms, AI integrations, RAG pipelines and end-to-end automations. One page per project: image, what I built, stack, highlights.

01	Enterprise AI Platform	3
02	Any.ai	4
03	End-to-End Lead Generation & Marketing Automation	5
04	Slack ↔ Notion Automation	6
05	AI Sales Deck Generator	7
06	Bot Post Creator in Telegram	8
07	AI Calendar Assistant	9
08	Llama Energy Promocode	10
09	Vocali	11
10	AMAIA	12

21+

projects shipped

6+ yrs

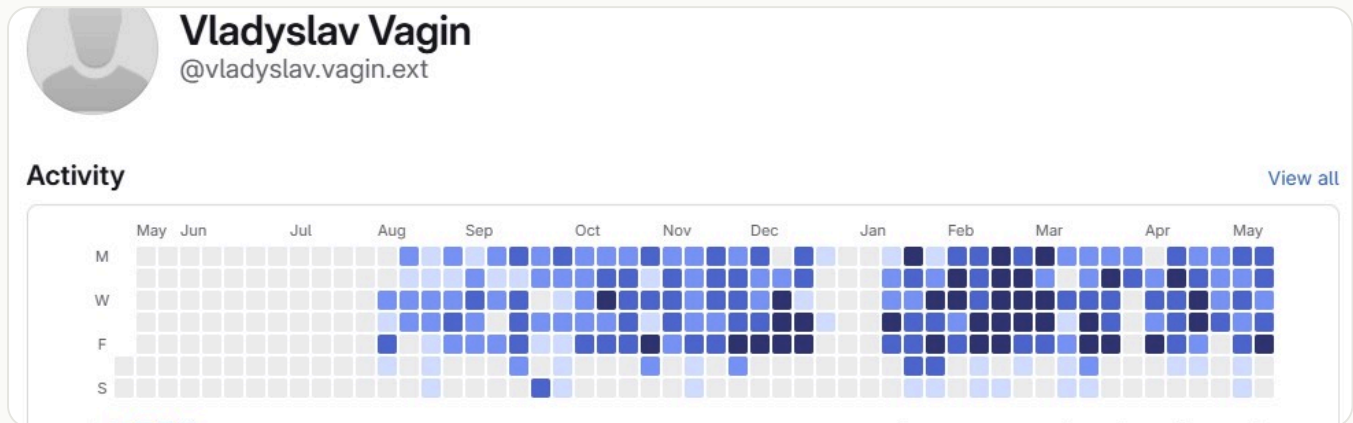
fullstack experience

EU + LatAm

clients across regions

Enterprise AI Platform — Fullstack Web Developer (Germany, NDA)

Major European enterprise (Germany) • NDA — name withheld • Fullstack web developer • AI integrations, interfaces, backend



Currently working as an external fullstack web developer for a large European enterprise based in Germany — building AI-powered user interfaces and backend services on Next.js and FastAPI. Under NDA: company name and code can't be shown; the activity heatmap stands in as evidence of continuous engagement since August 2025.

WHAT I BUILT

- Frontend on Next.js + TypeScript: production user interfaces against a typed API surface — routing, state, auth integration, design-system components, and the AI-feature UX (streaming responses, partial states, error handling).
- Backend on FastAPI: typed Python services sitting between the UIs and the AI layer — request validation via Pydantic, auth, structured outputs, observability hooks, and clean contracts for the frontend.
- AI integrations across the stack: wiring LLM-powered features into both surfaces — orchestration, prompt management, response handling, and graceful fallbacks for partial failures or rate-limit conditions.

STACK

Next.js React TypeScript FastAPI Python

AI / LLM integrations REST APIs

Git • MR workflow

HIGHLIGHTS

Aug 2025 → Present

ongoing engagement, continuous activity

Next.js + FastAPI

fullstack across UI and Python backend

AI integrations

LLM-powered features wired through both surfaces

Any.ai — AI Platform (Admin UI, Web App, WordPress Plugin)

Any.ai · AI startup · Fullstack developer · admin UI, web app, WordPress plugin



Ten-month fullstack build at an AI startup: a React admin dashboard, the user-facing web app, and a WordPress plugin that embeds Any.ai into third-party sites — all wired against a NestJS microservices core on AWS.

WHAT I BUILT

- anyai-admin-ui — React + TypeScript + Vite + Tailwind dashboard for managing the platform: tenants, services, model usage, and operations. Dockerized for reproducible deploys into staging and production.
- anyai-ui — the main user-facing React app: Redux for client state, react-i18next for multi-locale copy, Tailwind for a responsive design system shared with the admin surface, Vite for fast HMR in dev and a small production bundle.
- anyai-wordpress-plugin — WordPress plugin that embeds Any.ai functionality into third-party sites: widget support for the front-end, an admin panel for site owners to configure the connection, and separate production / staging deployment packages so installs map cleanly to the right backend.

STACK

React TypeScript Vite Tailwind CSS Redux
react-i18next NestJS Microservices MongoDB
Docker AWS WordPress

HIGHLIGHTS

10 months

production work at an AI startup

3 surfaces

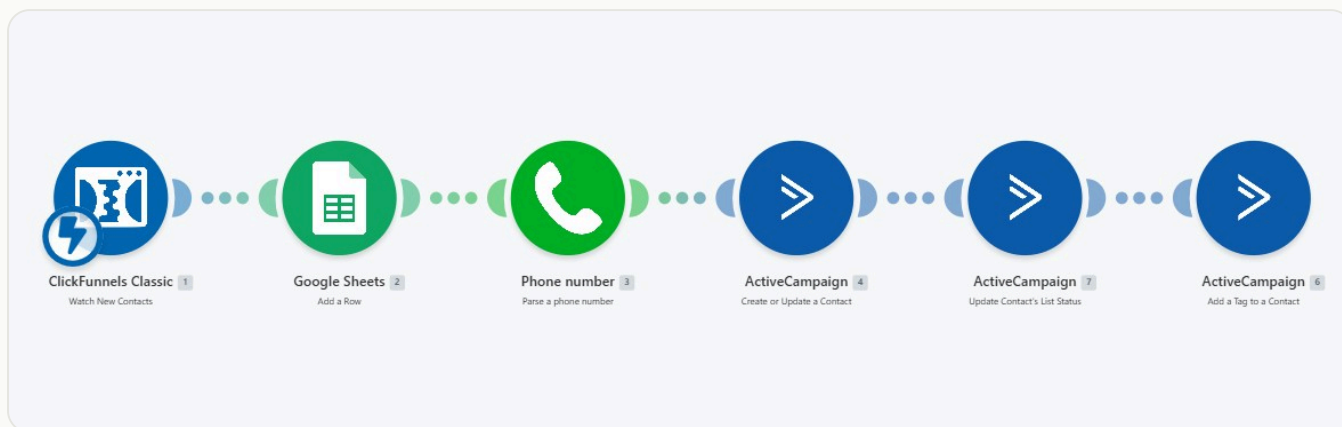
admin dashboard · web app · WordPress plugin

Microservices-wired

React + WP plugin against a NestJS gateway on AWS

End-to-End Lead Generation & Marketing Automation

Marketing department · Internal R&D · Solo build · funnels, qualification, routing, CRM segmentation, nurture



An end-to-end lead generation and marketing automation system: ClickFunnels captures traffic, Typeform qualifies on budget / project type / intent, Make.com routes and scores, and ActiveCampaign segments contacts into personalized email nurture sequences — built for scalable customer acquisition.

WHAT I BUILT

- Capture on ClickFunnels: high-converting funnels are the front door — every form submission fires a webhook into Make.com, so the lead is in the pipeline the moment the user hits submit.
- Qualification on Typeform: a conversational form scores leads on budget, project type and intent. The structured responses are what the rest of the system reads from — no free-text parsing later in the flow.
- Routing on Make.com: a central scenario reads the funnel + Typeform payload, applies a lead-scoring rubric (budget band × project type × intent signal), and routes each lead down the right branch — sales-ready, nurture, or disqualified — without a human in the middle.

STACK

ClickFunnels

Typeform

Make.com

ActiveCampaign

Webhooks

Lead scoring

CRM automation

Email nurture

HIGHLIGHTS

4 systems

ClickFunnels · Typeform · Make.com · ActiveCampaign — one pipeline

Auto-qualified

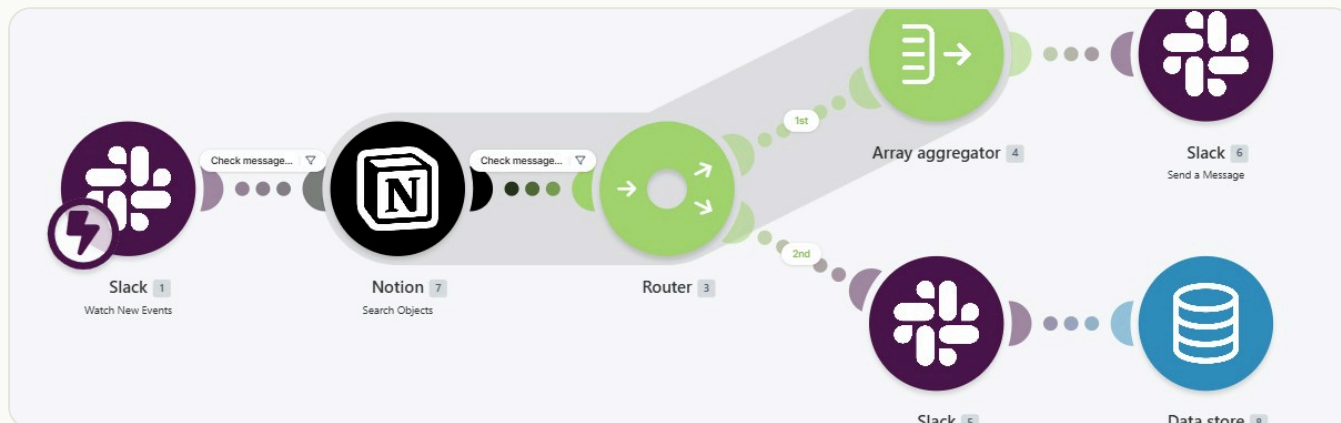
budget × project type × intent scored without human review

Segment-on-capture

contacts land in the right ActiveCampaign list the moment they submit

Slack ↔ Notion Automation — Manage Tasks Without Leaving Slack

Internal R&D • Solo build • Make.com orchestration + Slack/Notion APIs



A Make.com automation that bridges Slack and Notion: type "overdue" in Slack to get every delayed task with full context, and mark any task done with a single emoji reaction — no tab switching, no manual updates.

WHAT I BUILT

- Scenario 1 — list overdue: a Slack 'Watch New Events' trigger fires on the keyword "overdue". Make queries the Notion task database for items past their due date, a Router splits the flow, an Array Aggregator builds a single message with title + priority + status + link, and Slack posts the digest back into the channel.
- Scenario 2 — close on reaction: a separate Slack trigger watches for a designated emoji reaction. The Make Data Store maps the reacted-to Slack message back to its Notion page, Notion updates Status → Done, and Slack drops a confirmation in-thread — one click, zero forms.
- Make Data Store as the join table: every overdue list-out writes (slack_ts, channel_id) → notion_page_id, which is exactly what Scenario 2 reads at reaction time. No fragile message parsing, no LLM in the hot path.

STACK

Make.com

Slack Events API

Notion API

Make Data Store

Webhooks

OAuth 2.0

HIGHLIGHTS

1 keyword

type "overdue" → full list with context

1 emoji

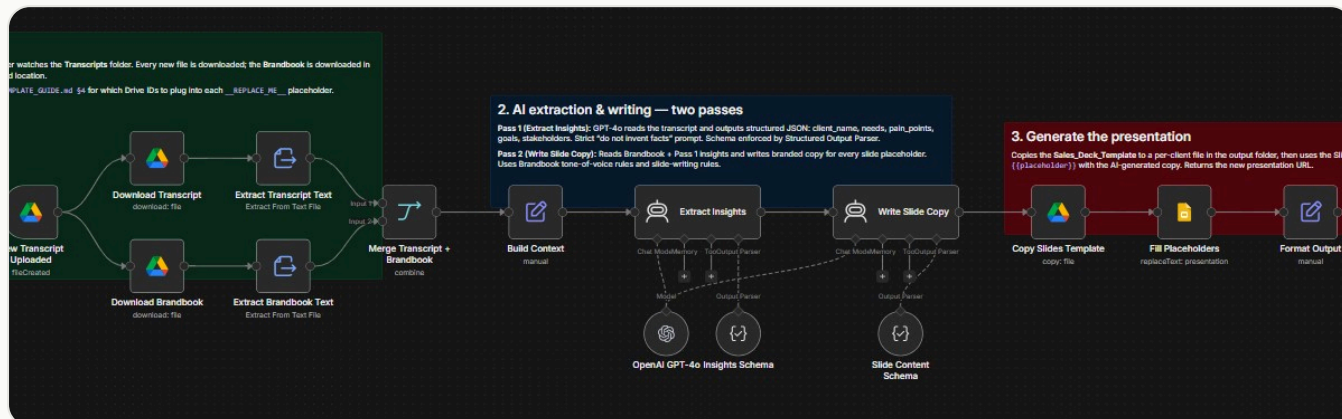
react to close a task — no tab switching

Single source of truth

Notion stays authoritative, Slack stays the interface

AI Sales Deck Generator — Transcript to Branded Presentation

Internal R&D • Solo build • architecture + n8n + prompts



An end-to-end AI workflow that turns a raw sales-meeting transcript into a client-ready, on-brand Google Slides deck in under 60 seconds.

WHAT I BUILT

- Two-pass agent architecture: pass 1 extracts structured client insights (needs, quantified pain points, goals with metric → target → deadline, stakeholders) under a strict no-fact-invention constraint; pass 2 writes branded slide copy from those insights + the Brandbook (tone, messaging pillars, banned-words list, slide-writing rules).
- LangChain Structured Output Parsers enforce valid JSON between passes so downstream nodes never receive malformed payloads.
- Drive API copies a pre-designed Google Slides template; Slides API batchUpdate swaps eleven placeholders with the generated copy — brand styling stays locked, content adapts per client.

STACK

n8n

OpenAI GPT-4o

LangChain Structured Output Parsers

Google Drive API

Google Slides API

OAuth 2.0

HIGHLIGHTS

~60s

transcript in, deck out

\$0.05-\$0.30

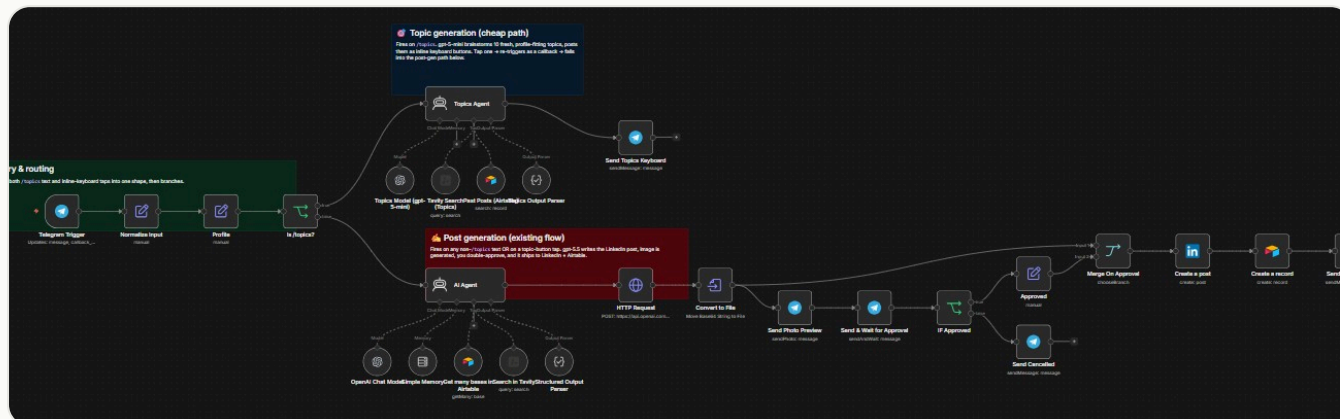
OpenAI cost per deck

Light review

only minor edits before client delivery

Bot Post Creator in Telegram — Automated LinkedIn Content Pipeline

Internal R&D · Solo build · n8n + prompts + integrations



An automated LinkedIn content pipeline orchestrated in n8n and driven from a Telegram bot — idea generation, content + image creation, human approval, then publish to LinkedIn and log in Airtable.

WHAT I BUILT

- Topic generation: the bot reads my LinkedIn profile + recent posts and proposes 10 relevant topics, sent to Telegram as tap-to-select buttons.
- Content generation: on selection, an LLM writes the full post and an image-generation model produces a matching visual. Structured Output Parsers enforce a strict JSON shape so downstream nodes never receive malformed payloads.
- Preview & approval in Telegram: the draft + image come back as a single preview with Approve / Reject buttons — no editorial work outside the chat thread.

STACK

n8n

Telegram Bot API

LinkedIn API

OpenAI (text + image)

Airtable

Structured Output Parsers

OAuth 2.0

HIGHLIGHTS

10 topics

AI-generated per session

1 tap

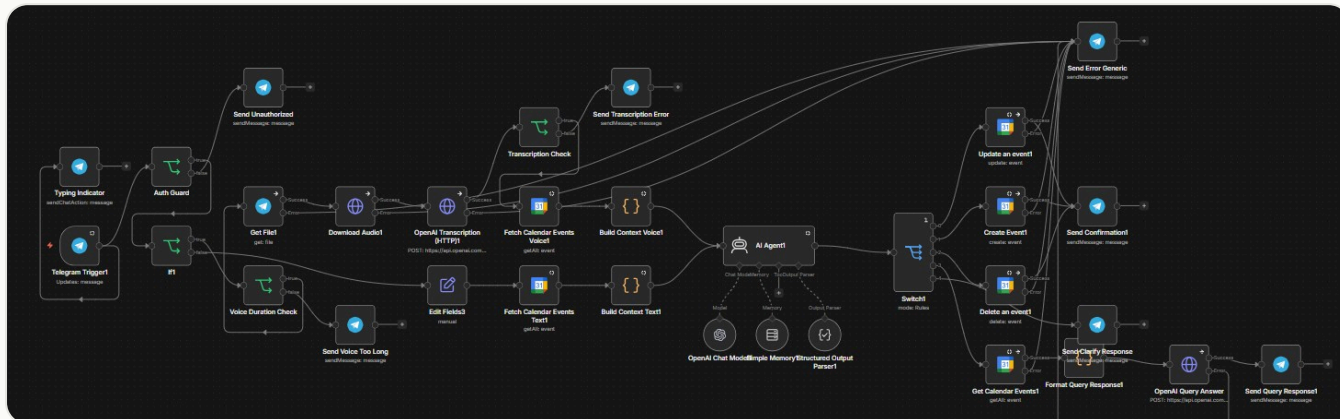
approve or reject in Telegram

4 systems

Telegram · LinkedIn · Airtable · OpenAI

AI Calendar Assistant — Manage Google Calendar from Telegram

Independent SaaS · [calendar-assistant.click](#) · Solo build · product, web app, agent + automations



A SaaS Telegram bot that manages your Google Calendar by voice or text — create, update, delete, and query events in natural language, no app switching.

WHAT I BUILT

- Telegram-first UX: users send a voice note or a text message to the bot and get back a confirmation, a clarifying question, or a calendar read-out — no app switching, no manual entry.
- Multilingual agent: OpenAI Whisper transcribes voice; a GPT-powered agent with a Structured Output Parser maps free-form intent to a strict action schema (create / update / delete / query) so downstream Google Calendar nodes never receive malformed payloads.
- n8n orchestration graph: Telegram trigger → auth guard → voice-duration check → transcription → context build (already-existing calendar events fetched for the relevant window) → AI agent → Switch on action → Google Calendar create/update/delete/query → Telegram confirmation. Branches for unauthorized users, voice-too-long, and transcription/agent errors send specific Telegram replies instead of failing silently.

STACK

Next.js TypeScript NextAuth Firebase
OpenAI (GPT + Whisper) n8n Telegram Bot API
Google Calendar API Docker

HIGHLIGHTS

Voice or text

natural-language scheduling in Telegram

4 actions

create · update · delete · query — one agent

2 languages

EN / UK web app and bot replies

Llama Energy Promocode — Multilingual Referral Landing Page

Llama Energy • referral campaign (Spain) • Solo build • Next.js 16 + Tailwind, SSG, i18n, analytics



High-performance, multilingual landing page for the Llama Energy referral campaign — built to convert paid and organic traffic into promo-code activations for an electricity-savings service in Spain.

WHAT I BUILT

- Next.js 16 with full Static Site Generation: every locale pre-rendered at build time, so first paint is near-instant on mobile and there's no server in the hot path between an ad click and the hero.
- next-intl for dynamic content localization: routes are locale-prefixed (/es as the live entry), copy lives in JSON dictionaries, and adding another language is a translation file — not a code change.
- Mobile-first Tailwind + TypeScript layout: hero, mascot lineup, benefit cards, FAQ and footer CTA all built from typed components with utility classes — small bundle, predictable styling, no CSS drift.

STACK

Next.js 16 TypeScript Tailwind CSS next-intl
Static Site Generation Next.js Animation API
Google Analytics Vercel

HIGHLIGHTS

SSG

every locale pre-rendered for instant first paint

i18n-ready

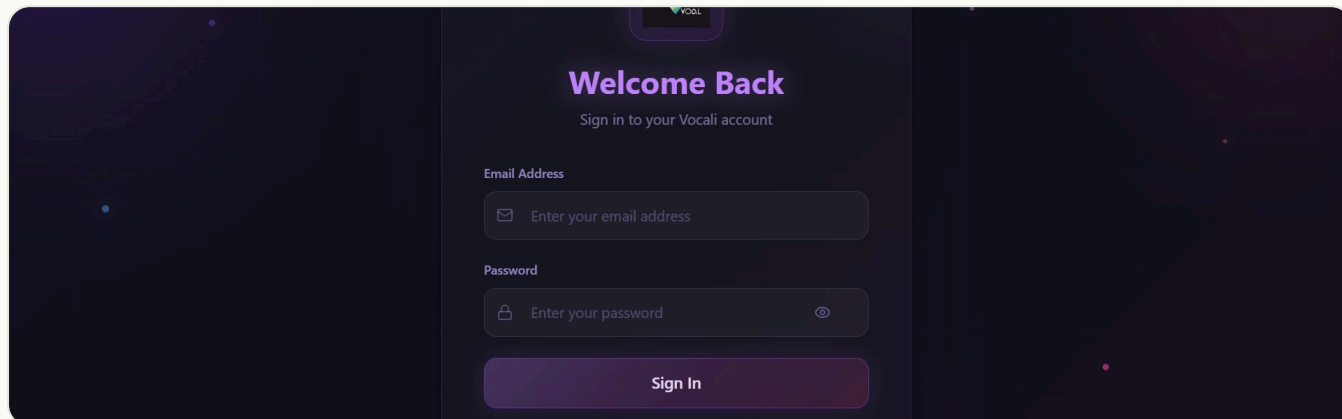
Spanish live (/es); new locales are a JSON file away

UTM + GA

traffic source attribution wired to every CTA

Vocali — Voice-Interface App with Live Transcription on AWS

Inbox Medical (Barcelona) · test task · Solo fullstack · React + NestJS + AWS + Terraform

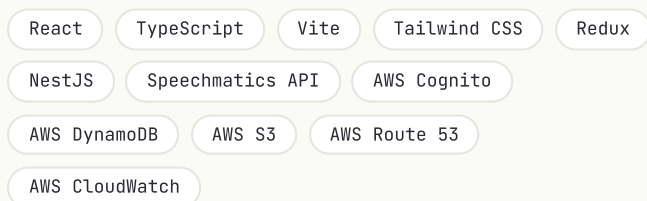


A 4-day fullstack test task for Inbox Medical (Barcelona): a voice-interface web app that uploads audio or records live in the browser, transcribes via Speechmatics in real-time or server-side, and lets the user download the transcript — backed by a NestJS API on AWS with Terraform-managed infra.

WHAT I BUILT

- React + Vite + Tailwind frontend with two capture modes: upload an audio file, or record live in the browser. Real-time transcription streams partial results into the UI as the user speaks; server-side mode shows job progress and serves the final transcript for download.
- NestJS + TypeScript backend owns auth, audio handling, and transcription orchestration. Speechmatics API powers both real-time streaming (WebSocket bridge) and server-side batch jobs — same provider, two flows, one API surface.
- AWS-native architecture: Cognito for sign-up / sign-in / JWT-protected routes, S3 for audio files and transcripts, DynamoDB for user-job metadata, Route 53 for the custom domain, CloudWatch for structured logs and job-lifecycle events.

STACK



HIGHLIGHTS

4 days

delivered against a 1-week brief

2 modes

live in-browser recording · server-side upload

AWS + Terraform

Cognito · DynamoDB · S3 · Route 53 · CloudWatch — all IaC

AMAIA — Aeromexico AI Chatbot: Document-Grounded RAG over SharePoint

Aeromexico • NDA, internal access only • Backend developer • semantic cache, vector DB, chat-history storage



Backend contributions to AMAIA — Aeromexico's internal AI chatbot that answers questions grounded in a SharePoint document corpus updated daily. Refactored the semantic cache and vector-DB layers, migrated chat history from S3 to DynamoDB, and helped optimize the API hot path.

WHAT I BUILT

- Refactored the Semantic Cache layer so repeated and near-duplicate questions short-circuit to a cached answer instead of re-embedding and re-querying the vector store — directly reduces LLM cost and tail latency on the hottest queries.
- Refactored the Vector Database integration to keep similarity queries fast as the SharePoint corpus is re-ingested daily — clean separation between ingestion writes and read-path queries so refresh runs don't degrade live chat performance.
- Migrated chat history from S3 (one object per chat/turn) to AWS DynamoDB with proper key design — constant-time access by user/session, no more listObjects pagination, and a model that actually supports queries for analytics and compliance review.

STACK

RAG architecture Vector database Semantic cache

AWS DynamoDB AWS S3 AWS EKS (Kubernetes)

AWS ECR AWS CodePipeline Docker

SharePoint API

HIGHLIGHTS

S3 → DynamoDB

chat history migrated, with replay + analytics access patterns

Cache + vector DB

refactored for lower latency and lower LLM cost

EKS pipeline

dev auto-deploy, qa/prod behind manual approval, rollback documented

// FIN

~24H REPLY

Let's build **something** great together.

If anything in here matched what you're trying to ship — drop me a line. I reply within a day, Mon–Fri.

EMAIL

vladyslav1991devjs@yahoo.com

LINKEDIN

[/in/vladyslav-fullstack](#)