Javier Norberto Zader

Córdoba, Argentina · jnzader@gmail.com · www.linkedin.com/in/jnzader/ · github.com/JNZader

Professional Summary

I work across backend (Java, Go, Rust), frontend (React, Next.js), ML pipelines, and developer tooling. I take the time to understand a problem before I write code, and I document the tradeoffs behind every technical decision. My path into tech started over 20 years ago in hardware support and equipment maintenance, with a stretch as a farmer in between, before I moved into software full time.

Professional Experience

Backend Developer

2024 – Present

Personal & freelance projects · Córdoba, Argentina

- Designed and built 6 end-to-end systems: APiGen (a code-generation platform), APiGen Studio (a visual microservices editor), Biogas Platform (industrial monitoring with real-time sensors and predictive models), ghagga (multi-agent AI code review), Consorcio Canalero (water-network management, freelance for a real client), and mcp-llm-bridge (an LLM gateway).
- Alongside those, tooling for my own workflow: starter templates for new projects, automated setup and configuration for development machines, and AI CLI configurations.
- I work with automated tests, conventional commits, and specs written before the code (OpenSpec) on the projects where it pays off.
- AI assistants (Claude Code, Codex, opencode) are a core part of my workflow, not an occasional helper.

- **Biogas Platform — Industrial monitoring platform**

  Biogas plants run on spreadsheets, with no real-time visibility into equipment health. An edge gateway reads the sensors with local ML inference that keeps working offline, and in the cloud the system flags anomalies and predicts failures hours before they happen — moving from reactive maintenance to predictive.
  - Status: my own side project, with the domain validated by an environmental engineer who knows the day-to-day of running biogas plants.
  - Rust edge gateway as a standalone industrial node: reads PLCs over Modbus TCP/RTU, stores locally in SQLite, runs ONNX inference (<50ms), works offline.
  - Task-specific models: edge (Isolation Forest + Autoencoder for anomalies), cloud (LSTM + Prophet for biogas/energy forecasting, Random Forest + XGBoost to predict failures 4-24h ahead). SHAP for explainability on every prediction.
  - Production-grade ML pipeline: training in Python with scikit-learn, export to ONNX, deployment in Rust with bit-for-bit parity tests. Continuous learning with automatic retraining and drift detection (PSI) that watches for model degradation in production.
  - Monorepo with 5 apps: Go backend, Rust edge gateway, Python ML service, React frontend, mobile app (Ionic).
  - Schedules maintenance by each machine's actual condition rather than a fixed calendar, cutting unplanned downtime.
  - Stack: Go + GORM + Gin, PostgreSQL, Redis, Mosquitto MQTT, Rust + Tokio + Axum, ONNX Runtime, Python, scikit-learn, React 19.
- **Consorcio Canalero — Management system**

  The Consorcio Canalero 10 de Mayo (Bell Ville) ran its water network, citizen reports, and administration across separate tools. The platform brings them together into a single application, adding satellite monitoring of the canals.
  - Status: freelance work for a real client (Consorcio Canalero 10 de Mayo, Bell Ville).
  - Google Earth Engine integration for satellite monitoring + MapLibre with Mapbox GL Draw for geospatial visualization and editing on the frontend.
  - A Martin tile server (Rust) serves vector data from PostGIS to the frontend in real time, with PMTiles tile caching.
  - Python ETL pipelines process spatial data (canals, schools, rural properties) and normalize it before loading into PostGIS.
  - Python backend with FastAPI and SQLAlchemy 2.0 (async via asyncpg) on PostgreSQL + PostGIS; migrations with Alembic and auth with fastapi-users.
  - Stack: React, TypeScript, Mantine, MapLibre, Mapbox GL Draw, Martin, Python, FastAPI, SQLAlchemy, PostgreSQL + PostGIS, Alembic, Google Earth Engine, Docker.
- **ghagga — Multi-agent AI code review**

  Manual code reviews don't scale: they hinge on the reviewer's time, focus, and experience. ghagga combines specialized AI agents with static-analysis tools to produce a single report per pull request, available as a SaaS, GitHub App, GitHub Action, and CLI.
  - Status: product in development. SaaS, GitHub App, GitHub Action, and CLI all shipped from the same monorepo.
  - Specialized agents (security, quality, performance) review in parallel and get synthesized into a single report.
  - 12+ static-analysis tools integrated (Semgrep, Trivy, Gitleaks, PMD, Biome, Ruff, and more) with SARIF export to plug into existing SecOps/SAST pipelines.
  - A per-PR health score as a unified metric that combines findings from the agents and the static analyzers.
  - One core serves all 3 channels — GitHub App, CLI, and SaaS dashboard — inside a Turborepo monorepo with 4 apps and 3 packages.
  - Stack: TypeScript, Turborepo + pnpm workspaces, Hono, BullMQ, Drizzle ORM, Vercel AI SDK, Octokit, React 19 + React Router 7 (dashboard), Commander (CLI), GitHub Action.
- **APiGen — Code-generation platform**

  Every new backend repeats the same structures (entity, repo, service, controller, DTOs, security, observability) with inconsistent variations across teams. APiGen — driven from a CLI, HTTP, an IDE plugin, or MCP — takes a SQL schema or an OpenAPI contract and generates a production-ready service with enterprise features on by default.
  - Status: my own side project. The public repo holds the original idea (Dec 2024); the current version is a private multi-language platform.
  - Decoupled pipeline: the parser and the templates don't know about each other, so new languages can be added without breaking what already works.
  - 22 Gradle modules organized into libraries, generators, and optional feature packs. The same engine runs from a CLI, an HTTP server,

an IDE plugin, or an MCP server.
  - Multi-protocol: the same model exposes REST + GraphQL + gRPC without rewriting business logic.
  - Enterprise features included with no extra code: soft delete, multi-tenancy, auditing with Hibernate Envers, multi-level cache (Caffeine + Redis), optimistic locking.
  - Stack: Java 25, Spring Boot 4, Gradle, Caffeine, Redis, Docker (with Dockerfile.native for GraalVM), OpenAPI, GraphQL, gRPC, Prometheus, JMH benchmarks, Spring Cloud Contract.
- **APiGen Studio — Visual microservices editor**

  Microservice architectures usually get designed in loose diagrams that never translate directly into code. Studio lets you model entities, service-to-service connections, gateway routes, and event flows visually in the browser, and export the multi-service project to feed into APiGen.
  - Status: my own side project. Public demo at apigen-web.vercel.app.
  - It's not just entity modeling: it designs full microservice architectures (services, connections, gateway routes, event flows).
  - When nodes are related to each other, the ELK library lays them out automatically to minimize line crossings; when they're standalone, they fall into a regular grid.
  - Multi-format export: a ready-to-run Spring Boot project ZIP, the model as JSON, SQL DDL, and a PNG/SVG diagram of the canvas.
  - Autosave to IndexedDB with snapshot history and safety snapshots before destructive imports. WCAG 2.1 AA with keyboard shortcuts — the graph editor works without a mouse.
  - Stack: React 19, TypeScript 5.9, Vite 7, Mantine 8, Zustand, Zod, React Flow, ELK, Playwright, Vitest.
- **mcp-llm-bridge — LLM gateway and MCP server**

  Every AI tool needs its credentials configured separately, with no central cost control or smart routing. mcp-llm-bridge plays two roles: an OpenAI-compatible HTTP gateway for external tools, and an MCP server for clients like Claude Code or Cursor. It centralizes 11 LLM providers (API keys and CLI subscriptions) behind an encrypted vault.
  - Status: my own side project. Gateway live at gateway.javierzader.com.
  - 11 provider adapters: 5 over direct API (Anthropic, OpenAI, Google, and more) and 6 backed by CLIs (Claude Code, Gemini CLI, Codex, Copilot, Qwen, OpenCode).
  - Encrypted vault for credentials, scoped per project with fallback to `_global`.
  - Task-based routing: it picks the best provider for each kind of work (generation, refactor, summary, code), with per-project overrides.
  - MCP tools that go beyond generation: vault operations, semantic code search, shared state via CRDT, usage inspection.
  - End-to-end observability: traces with OpenTelemetry, metrics with Prometheus, and structured logging with pino.
  - Stack: TypeScript, Hono, @modelcontextprotocol/sdk, better-sqlite3, OpenTelemetry, Prometheus, pino, Zod 4.

Technical Skills

Languages:	Java, Go, Rust, Python, TypeScript, JavaScript, SQL
Backend:	Spring Boot 4, Gin (Go), FastAPI, Axum (Rust), Hono, Tokio, Node.js
Frontend:	React 19, Next.js, Vite, Mantine, Tailwind CSS, Zustand, Zod, React Flow, ELK, Ionic + Capacitor
Databases:	PostgreSQL, MySQL, Redis, SQLite, IndexedDB
ML / AI:	scikit-learn, ONNX Runtime, llama.cpp, SHAP, LiteLLM, MCP (Model Context Protocol), Pandas, NumPy
Industrial / IoT / Edge:	Modbus TCP/RTU, MQTT (Mosquitto), Edge computing, OTA with ed25519 signing, Prometheus, NATS
GIS / Geo:	Google Earth Engine, MapLibre, Mapbox GL Draw
DevOps:	Docker, GitLab CI, GitHub Actions, Vercel, Cloudflare Pages, Caddy
Testing:	Playwright, Vitest, JUnit, Mockito, Pytest, JMH, SonarQube, Biome
APIs / Spec:	OpenAPI, GraphQL, gRPC, OpenSpec, Conventional Commits, Semantic Release
Tools:	Git, Linux, Prisma, IntelliJ, VS Code, Android Studio

Education & Certifications

Software Development Technician *2023-12 – 2025-07*
Universidad Gastón Dachary

ONE Tech Foundation G8 — Data Science, ETL & ML *2024 – 2025*
Alura Latam
- Statistics and Machine Learning (68h)
- ETL with Pandas/NumPy and visualization (61h)
- Data modeling with Python (40h)

Java & Spring Boot G6 — ONE *2024 – 2024*
Alura Latam
- Java and Spring Boot G6 (104h): REST API with Spring Boot 3, security, JPA/Hibernate, Streams/Lambdas.
- Object-Oriented Java (45h) and Java Web with Spring (34h).
- SQL with MySQL (DML, Procedures) (36h).

Intermediate Java Developer *2022-10 – 2023-11*
Argentina Programa

Cisco Certified Network Associate (CCNA) *2009 – 2009*
Fundación Proydesa

Languages

Spanish:	Native
English:	Intermediate (fluent technical reading)