Javier Norberto Zader

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

Resumen Profesional

Trabajo en backend (Java, Go, Rust), frontend (React, Next.js), pipelines de ML y herramientas para developers. Me tomo el tiempo de entender el problema antes de codear, y documento los tradeoffs de cada decisión técnica. Mi camino en tecnología empezó hace más de 20 años en soporte técnico y mantenimiento de equipos, con un paso intermedio como productor agropecuario antes de volcarme de lleno al software.

Experiencia Profesional

Backend Developer

2024 – Presente

Proyectos personales y freelance · Córdoba, Argentina

- Diseñé y construí 6 sistemas end-to-end: APiGen (plataforma de generación de código), APiGen Studio (editor visual de microservicios), Biogas Platform (monitoreo industrial con sensores en tiempo real y modelos predictivos), ghagga (code review con IA multi-agente), Consorcio Canalero (gestión hídrica, freelance para un cliente real) y mcp-llm-bridge (gateway de LLMs).
- Además, herramientas para mi propio flujo de trabajo: plantillas para empezar proyectos nuevos, instalación y configuración automática de máquinas de desarrollo, y configuraciones de CLIs de IA.
- Trabajo con tests automatizados, conventional commits y especificaciones escritas antes del código (OpenSpec) en los proyectos donde corresponde.
- Uso asistentes de IA (Claude Code, Codex, opencode) como parte central del workflow, no como ayuda esporádica.

- **Biogas Platform — Plataforma industrial de monitoreo**

  Las plantas de biogás dependen de planillas Excel, sin visibilidad en tiempo real del estado de los equipos. Un gateway en el edge lee los sensores con inferencia de ML local que funciona offline, y en la nube el sistema detecta anomalías y predice fallos horas antes de que ocurran — pasando de un mantenimiento reactivo a uno predictivo.
  - Estado: side project propio con dominio validado por un ingeniero ambiental que conoce el día a día de las plantas de biogás.
  - Edge gateway en Rust como nodo industrial autónomo: lee PLCs por Modbus TCP/RTU, almacena local en SQLite, hace inferencia ONNX (<50ms), funciona offline.
  - Modelos específicos por tarea: edge (Isolation Forest + Autoencoder para anomalías), cloud (LSTM + Prophet para forecasting de biogás/energía, Random Forest + XGBoost para predicción de fallos 4-24h adelante). SHAP para explicabilidad de cada predicción.
  - Pipeline de ML de nivel producción: entrenamiento en Python con scikit-learn, exportación a ONNX, despliegue en Rust con tests de paridad bit a bit. Aprendizaje continuo con reentrenamiento automático y drift detection (PSI) que monitorea degradación de modelos en producción.
  - Monorepo con 5 apps: backend Go, edge gateway Rust, servicio de ML en Python, frontend React, app mobile (Ionic).
  - Prioriza el mantenimiento por la condición real de cada equipo, no por calendario fijo, reduciendo las paradas no planificadas.
  - Stack: Go + GORM + Gin, PostgreSQL, Redis, Mosquitto MQTT, Rust + Tokio + Axum, ONNX Runtime, Python, scikit-learn, React 19.
- **Consorcio Canalero — Sistema de gestión**

  El Consorcio Canalero 10 de Mayo (Bell Ville) gestionaba la red hídrica, los reportes ciudadanos y la administración en herramientas separadas. La plataforma los unifica en una sola aplicación, sumando monitoreo satelital de los canales.
  - Estado: trabajo freelance para un cliente real (Consorcio Canalero 10 de Mayo, Bell Ville).
  - Integración con Google Earth Engine para monitoreo satelital + MapLibre con Mapbox GL Draw para visualización y edición geoespacial en el frontend.
  - Tile server Martin (Rust) sirve datos vectoriales desde PostGIS al frontend en tiempo real, con caché de tiles PMTiles.
  - Pipelines ETL en Python procesan datos espaciales (canales, escuelas, propiedades rurales) y los normalizan antes de cargarlos en PostGIS.
  - Backend Python con FastAPI y SQLAlchemy 2.0 (async con asyncpg) sobre PostgreSQL + PostGIS; migrations con Alembic y auth con fastapi-users.
  - Stack: React, TypeScript, Mantine, MapLibre, Mapbox GL Draw, Martin, Python, FastAPI, SQLAlchemy, PostgreSQL + PostGIS, Alembic, Google Earth Engine, Docker.
- **ghagga — Code review con IA multi-agente**

  Los code reviews manuales no escalan: dependen del tiempo, foco y experiencia del reviewer. ghagga combina agentes especializados de IA con herramientas de análisis estático para producir un informe único por pull request, disponible como SaaS, GitHub App, GitHub Action y CLI.
  - Estado: producto en desarrollo. SaaS, GitHub App, GitHub Action y CLI distribuidos desde el mismo monorepo.
  - Distintos agentes especializados (security, quality, performance) revisan en paralelo y se sintetizan en un informe único.
  - 12+ herramientas de análisis estático integradas (Semgrep, Trivy, Gitleaks, PMD, Biome, Ruff, etc.) con exportación SARIF para integrar con pipelines SecOps/SAST existentes.
  - Health score por PR como métrica unificada que combina los hallazgos de los agentes y de los analizadores estáticos.
  - Un solo core atiende los 3 canales: GitHub App, CLI y dashboard SaaS, dentro de un monorepo Turborepo con 4 apps y 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 — Plataforma de generación de código**

  Cada nuevo backend repite las mismas estructuras (entidad, repo, service, controller, DTOs, security, observability) con variaciones inconsistentes entre equipos. APiGen — que se maneja desde CLI, HTTP, plugin de IDE o MCP — toma un schema SQL o un contrato OpenAPI y genera un servicio listo para producción con features enterprise por default.

- Estado: side project propio. Repo público con la idea original (dic. 2024); la versión actual es una plataforma privada multi-lenguaje.
- Pipeline desacoplado: el parser y los templates no se conocen entre sí, lo que permite agregar lenguajes nuevos sin romper lo que ya funciona.
- 22 módulos Gradle organizados en librerías, generadores y feature packs opcionales. El mismo engine se usa desde CLI, servidor HTTP, plugin de IDE o servidor MCP.
- Multi-protocolo: el mismo modelo expone REST + GraphQL + gRPC sin reescribir lógica de negocio.
- Features enterprise incluidas sin código adicional: soft delete, multi-tenancy, auditoría con Hibernate Envers, caché multinivel (Caffeine + Redis), optimistic locking.
- Stack: Java 25, Spring Boot 4, Gradle, Caffeine, Redis, Docker (con Dockerfile.native para GraalVM), OpenAPI, GraphQL, gRPC, Prometheus, JMH benchmarks, Spring Cloud Contract.

- **APiGen Studio — Editor visual de microservicios**

  Las arquitecturas de microservicios suelen diseñarse en diagramas sueltos que después no se traducen directamente en código. Studio permite modelar visualmente entidades, conexiones entre servicios, gateway routes y flujos de eventos en el browser, y exportar el proyecto multi-servicio para alimentar APiGen.
  - Estado: side project propio. Demo pública en apigen-web.vercel.app.
  - No es solo modelado de entidades: diseña arquitecturas de microservicios completas (services, conexiones, gateway routes, flujos de eventos).
  - Cuando los nodos tienen relaciones entre sí, la librería ELK los acomoda automáticamente para minimizar cruces de líneas; cuando son nodos sueltos, caen en una grilla regular.
  - Exportación multiformato: ZIP de proyecto Spring Boot listo para correr, JSON del modelo, SQL DDL, y diagrama PNG/SVG del canvas.
  - Autoguardado en IndexedDB con historial de snapshots y safety snapshots antes de importaciones destructivas. WCAG 2.1 AA con atajos de teclado — el editor de grafos funciona sin mouse.
  - Stack: React 19, TypeScript 5.9, Vite 7, Mantine 8, Zustand, Zod, React Flow, ELK, Playwright, Vitest.
- **mcp-llm-bridge — LLM gateway y servidor MCP**

  Cada herramienta de IA requiere configurar sus credenciales por separado, sin control centralizado de costos ni routing inteligente. mcp-llm-bridge cumple dos roles: gateway HTTP compatible con la API de OpenAI para herramientas externas, y servidor MCP para clientes como Claude Code o Cursor. Centraliza 11 proveedores de LLM (API keys y suscripciones CLI) detrás de un vault cifrado.
  - Estado: side project propio. Gateway live en gateway.javierzader.com.
  - 11 adaptadores de proveedor: 5 por API directa (Anthropic, OpenAI, Google, etc.) y 6 respaldados por CLI (Claude Code, Gemini CLI, Codex, Copilot, Qwen, OpenCode).
  - Vault cifrado para credenciales, con scope por proyecto y fallback a `_global`.
  - Routing según la tarea: elige el proveedor óptimo para cada tipo (generation, refactor, summary, code), con override por proyecto.
  - Herramientas MCP que van más allá de la generación: operaciones de vault, búsqueda semántica de código, estado compartido vía CRDT, inspección de uso.
  - Observabilidad end-to-end: traces con OpenTelemetry, métricas con Prometheus y logging estructurado con pino.
  - Stack: TypeScript, Hono, @modelcontextprotocol/sdk, better-sqlite3, OpenTelemetry, Prometheus, pino, Zod 4.

Habilidades Técnicas

Lenguajes:	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
Bases de Datos:	PostgreSQL, MySQL, Redis, SQLite, IndexedDB
ML / IA:	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 con firma ed25519, 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
Herramientas:	Git, Linux, Prisma, IntelliJ, VS Code, Android Studio

Educación y Certificaciones

Técnico en Desarrollo de Software *2023-12 – 2025-07*
Universidad Gastón Dachary

ONE Tech Foundation G8 — Data Science, ETL y ML *2024 – 2025*
Alura Latam
- Estadísticas y Machine Learning (68h)
- ETL con Pandas/NumPy y visualización (61h)
- Modelado de datos con Python (40h)

Java y Spring Boot G6 — ONE *2024 – 2024*
Alura Latam
- Java y Spring Boot G6 (104h): API REST con Spring Boot 3, seguridad, JPA/Hibernate, Streams/Lambdas.
- Java Orientado a Objetos (45h) y Java Web con Spring (34h).
- SQL con MySQL (DML, Procedures) (36h).

Desarrollador Java Intermedio *2022-10 – 2023-11*
Argentina Programa

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

Idiomas

Español: Nativo
Inglés: Intermedio (lectura técnica fluida)