CAIO VILQUER CARVALHO

BACKEND DEVELOPMENT | JAVA | SPRING BOOT | NODE.JS

São Paulo, SP | +55 (31) 98815-6118 | caio@vilquer.dev

LinkedIn: linkedin.com/in/caio-vilquer | GitHub: github.com/caiovilquer | Portfolio: vilquer.dev

SUMMARY

Computer Engineering student at the Polytechnic School of the University of São Paulo (Poli-USP), with hands-on experience developing REST APIs and web systems. Focused on Java, Kotlin, Spring Boot, and PostgreSQL, complemented by Node.js, TypeScript, and NestJS. Experience with RAG using PostgreSQL/pgvector, LLM integrations, authentication and authorization, automated testing, Docker, CI, and observability. Seeking software development internships or junior backend developer roles.

EDUCATION

B.Sc. in Computer Engineering | Polytechnic School, University of São Paulo (Poli-USP)
02/2025 - 12/2029 (expected graduation)

SKILLS

Backend & AI: Java, Kotlin, Spring Boot, Spring Security, Spring Data JPA, Hibernate, Node.js, TypeScript, NestJS, REST APIs, RAG, OpenAI Responses API, Structured Outputs, and embeddings
Data: PostgreSQL, pgvector, MySQL, JPA/Hibernate, Prisma, Flyway, full-text search, and relational modeling
Testing & infrastructure: JUnit 5, Mockito, MockMvc, Testcontainers, LLM evals, Docker, Docker Compose, GitHub Actions, Git, and Linux
Architecture & security: modular and hexagonal architecture, Ports & Adapters, transactional outbox, idempotency, and prompt injection testing
Complementary frontend: React, Angular, TypeScript, Tailwind CSS, Vite, and TanStack Query
Languages: Intermediate English - technical documentation reading and written communication

SELECTED PROJECTS

VIAZIO | Explainable travel recommendation engine

03/2026 - 07/2026 | Java 21 | Spring Boot | PostgreSQL | Demo: viazio.vilquer.dev | Code: github.com/caiovilquer/viazio

- Designed and implemented a versioned REST API in Java 21 and Spring Boot to recommend short-trip destinations based on calendar, weather, estimated cost, distance, and local context, detailing each score factor.
- Organized the scoring domain into rules, filters, persistence, cache, and integration modules, isolating recommendation decisions from infrastructure components.
- Persisted data in PostgreSQL, versioned the schema with Flyway, and defined per-source refresh and caching policies; integrated Nager.Date, Open-Meteo, World Bank, AwesomeAPI, and Wikipedia.
- Applied retry, circuit breaker, and bulkhead patterns with Resilience4j, rate limiting with Bucket4j, validation, traceId, and Micrometer/Prometheus metrics; documented the API with OpenAPI/Swagger.
- Containerized the application with Docker Compose, configured CI in GitHub Actions, and maintained 250+ backend tests with no network dependency.

POLIATLETAS | Poli-USP track and field team management

08/2025 - Present (MVP in 03/2026) | NestJS | Fastify | PostgreSQL | Prisma | Demo: poliatletas.com.br

- Developed a NestJS and Fastify backend to manage athletes, competitions, events, results, rankings, records, and team administrative operations.
- Organized the application with Ports & Adapters, separating domain, use cases, and persistence, and modeled sports entities in PostgreSQL with Prisma.
- Implemented ranking calculations and record validation by event, category, gender, mark, and competition, plus batch Excel/CSV imports with validation and consistent history persistence.
- Secured administrative operations with JWT, RBAC, and custom guards, and integrated the API with dashboards built in React, Tailwind CSS, and Recharts.

ROTINA PET | Pet care management with AI assistant and RAG

05/2025 - Present | Kotlin | Spring Boot | Angular | PostgreSQL + pgvector | OpenAI | Demo: rotinapet.vilquer.dev | Backend: github.com/caiovilquer/rotina-pet | Frontend: github.com/caiovilquer/rotina-pet-front

- Evolved the Kotlin/Spring Boot API into a 7-module Gradle hexagonal architecture, with JWT, family/role-based authorization, and JPA/Hibernate persistence using Value Objects.
- Implemented hybrid RAG in PostgreSQL (PT-BR full-text + pgvector) with Reciprocal Rank Fusion and authorization before ranking; responses cite retrieved evidence and abstain when unsupported.
- Built asynchronous indexing with a transactional outbox, FOR UPDATE SKIP LOCKED, retry/dead letter, chunking, batch embeddings, and atomic swap; embedding version changes trigger reindexing.
- Integrated OpenAI Responses API with Structured Outputs/JSON Schema, human confirmation, and idempotency; created LLM evals in CI (Recall@5 ≥95%, citations ≥98%, abstention ≥95%), prompt injection tests, and Testcontainers/pgvector.