

Diego Demétrio

Senior Full-Stack Engineer

Belo Horizonte, Brazil (UTC-3) · diegodemetrio79@gmail.com · +55 31 99478-1420 · linkedin.com/in/diegodemetrio79 · github.com/Quorthon13 · diegodemetrio.dev

Remote, UTC-3, full overlap with US business hours. PJ/CNPJ preferred; open to EOR and CLT.

SUMMARY

I build mission-critical systems in Java and Angular. My backend work is Spring Boot, Quarkus and Jakarta EE; my frontend work is Angular and TypeScript. I am tech lead and architect for a real-time monitoring platform at a nuclear research center. Before that I moved a federal university's learning platform and institutional repository onto supported versions with no downtime. English at C2, EF SET certified.

SKILLS

Core: Java, Spring Boot, Quarkus, Jakarta EE, Angular, TypeScript, Node.js, Vue, Ionic, PostgreSQL, Oracle, MQTT, REST, Docker, Kubernetes, OpenShift, Jenkins, CI/CD, Git, Linux (RHEL/Rocky), Agile

Working: Next.js, React, C#/.NET, PHP (Moodle)

Learning: AWS — studying for the Solutions Architect Associate certification; Kotlin

EXPERIENCE

PD Case Informática Ltda

Belo Horizonte, Brazil (remote) · Feb 2020 — present

Senior Full-Stack Engineer, Tech Lead & Architect

May 2025 — present

Real-time monitoring platform for radioactive material storage at a nuclear research center. Sensors publish over MQTT into Quarkus and Jakarta EE services, running in containers on Kubernetes and OpenShift. The Angular console carries operational dashboards and a 3D view that puts each reading where it physically is.

- I chose Angular with Three.js over react-three-fiber for the console's 3D view. The team had years of Angular between them and no React at all. Three.js is plain JavaScript, so adapters had to be written before it was typed and idiomatic in Angular.
- I set the system architecture and the service boundaries alone, in the months before other engineers joined, and defined the data path from sensor ingestion through to what the console renders.
- I negotiated one MQTT topic hierarchy across teams, fixed per sensor and per zone, agreed as an RFC before the first message was published. A sensor nobody had specified yet still has one place its data can go.
- I set the code-review standards the team holds to, and mentor engineers on the patterns underneath the architecture.

Mid-Level Full-Stack Engineer

Oct 2022 — May 2025

Modernization of university-wide academic systems for a federal university. Alongside it, a legacy ASP.NET personnel and radiation monitoring records system at a nuclear research center.

- I moved a university learning platform from 3.1.x to 4.5 LTS, and the operating system under it from CentOS 7 to Rocky 9. Rocky 9 ships a PHP the live version could not run, so the application had to step forward first, through intermediate versions, on the old OS. I rehearsed every step on a production clone and cut over with no downtime.
- I rearchitected an institutional repository's frontend onto Angular (Dspace 5 to 7). The plugin that had issued historical course certificates no longer existed, so I stood up a replacement, rebuilt the templates against it, and carried the certificate history across by hand.
- I replaced hand-run deploys to on-premise servers with a shared Jenkins pipeline template, reused across the client portfolio.

Junior Full-Stack Engineer

Aug 2020 — Oct 2022

Web and mobile applications for a municipal civil-defense program monitoring landslide and flood risk areas. Java, Angular, Ionic, Node.js, Vue and PostgreSQL.

- I built the field inspector app in Ionic and Angular. One codebase covers iOS and Android, which a team of web developers can maintain.
- I implemented offline capture for sites with poor mobile connectivity. Observations queue on the device and sync when the connection returns. The server takes the last write, which holds because one inspector owns a site for the length of a visit.

Software Engineering Intern

Feb 2020 — Aug 2020

- I shipped full-stack features on a Java and Angular application over Oracle: CRUD interfaces and operational reports. I followed the conventions already in the codebase rather than the patterns I had just learned in coursework.

OPEN SOURCE PROJECT

OLED Sleeper — github.com/Quorthon13/OLED-Sleeper

138 stars · 15 forks

C#/WPF · DDC/CI · no external dependencies

- I wrote and maintain an open-source Windows utility that dims or blacks out individual OLED displays, each on its own idle timer. Dimming goes over DDC/CI to the monitor. Blacking out draws a full-screen black overlay, and the display returns when it stops being idle.
- Detection is separated from action by a command, so a new way of detecting idle is one emitter and a new action is one command. Most releases since the first one fix hardware quirks reported by users, and the split is what kept those fixes to one side or the other.
- I wrote the Win32 and DDC/CI calls directly with no external dependencies, so the binary stays portable and the whole codebase is auditable. Compiled portable releases are distributed on SourceForge.

EDUCATION

B.Sc. Computer Science — Universidade Federal de Ouro Preto (UFOP)

2020 — Aug 2026

Completed while working full-time.

Exchange semester — Universidad Nacional de La Plata, Argentina

Aug — Dec 2024

Technical Diploma in Information Technology — Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

2017 — 2019

Three-year federal technical program in information technology, completed concurrently with secondary education. Admission by competitive examination.

LANGUAGES AND CERTIFICATIONS

Portuguese: native

English: C2 — EF SET certified (Sep 2026), cert.efset.org/wMgQnp

Spanish: conversational