Pedro Revez da Silva

Platform Engineer

Email: pedro.revez.silva@gmail.com

GitHub: https://github.com/Pedro-Revez-Silva

Portfolio: https://pedrorevezsilva.me/

Notes & Documents: https://pedrorevezsilva.wiki

About Me

Platform engineer with 9 years of experience building monitoring infrastructure, observability services, developer tools and AI integrations. Additionally, I built custom testing platforms and other tools and apps that make my life easier. Occasionally I make my project notes public.

Professional Experience

Founder & Platform Engineer @ Listas Lendárias Consulting

November 2016 - Present

Platform engineering consultancy serving a broad range of clients in fintech, e-commerce and SaaS.

Platform Infrastructure & Automation:

- Architected and implemented scalable testing platforms for multiple enterprise clients, supporting hundreds of automated tests with parallel execution and intelligent test data management.

- Built deployment monitoring systems preventing production incidents through automated version drift detection across microservice clusters.
- Developed an intelligent incident response service through PagerDuty and DataDog integrations and traffic pattern analysis, reducing on-call escalations by a significant amount, from daily incidents to monthly, with no noticeable service degradation.
- Created cross-functional admin tools enabling self-service workflows for development, QA and product teams.

AI & Real-time Systems:

- Implemented real-time financial data monitoring systems to detect price anomalies across third party data sources with automated alerting and reports.
- Built custom AI integration platforms with context management and performance optimization.
- Built client-side parallel processing system for large audio file transcription, handling files up to 500MB entirely in-browser through optimized chunking and concurrent API orchestration.

Developer Experience & Tooling:

- Created comprehensive CI/CD pipelines and automations using GitHub Actions, Azure DevOps, Jenkins and Docker.
- Built internal tools to reduce manual processes.
- Implemented monitoring and observability solutions with custom alerting, dashboards and reporting.

Tech Stack:
TypeScript & JavaScript, Python, Go • React, Vue.js, Svelte • PostgreSQL, MongoDB, Redis • REST, GraphQL, Postman, OpenAPI, Swagger • AWS, Azure, Docker, Kubernetes, Jenkins, GitHub Actions • Playwright, Cypress • Agile,

Kanban, Git Flow, Trunk-Based Development, CI/CD • Confluence, Notion, Linear, Jira, Figma

---

Junior Developer @ Siemens

July 2015 - September 2016

As my first experience working in development after studying, it was where I had more opportunities to learn. In this project I was the only software developer in the team. Our objective was to build an interface between old, sometimes mechanical, traffic controllers and a state of the art traffic management central system. As expected some functions and data-collection would be unavailable or limited.

We achieved a production ready solution, comprised of a microcontroller programmed in C, connected via APN to a server application developed in Java and running in a Linux environment. The server application managed the incoming data and relaid it to the central system using a SOAP based protocol. Besides developing the server application from scratch, I was also responsible for the production of the devices, large scale testing and deployment of the field test units. During this time I also contributed to another project which was focused on providing a platform for internal day-to-day bureaucratic processes and document management.

Tech Stack:
Java • C • SOAP • XML • APN/VPN • Load/Performance Testing

Technical Projects

BedtimeBard - Bed time story generator

Built a bedtime story generator for my kids. The second version generates both text and images, allows sharing stories and supports translation. To avoid huge bills on the image generation side, it periodically generates stories and images

and then serves them randomly. The main challenge here is to keep images consistent without having the user waiting for a long period of time.

The first version generated stories in real time and generated all the images in parallel. The issue with this was the inconsistency of the images. Since they didn't have context of each other, the characters were different in each. To solve this I decided to move to async story generation. It takes longer, but gives me more freedom to do things with the story. The UX is improved because the user now gets stories on ms loading times instead of 30 sec. The more stories that exist, the less likely a user is to be sent the same story and it keeps track of the stories the user has seen to not send those. The user can always save a story via url and go back to it.

The next step is to support dynamic story telling, where the kids can choose the path of the story using branching. Then I want to build a different event driven story telling experience.

The app is offline at the moment because the managed DB was too expensive.

Tech Stack:

- Go on the backend
- Svelte kit on the frontend
- S3 bucket to save the images related to the story with uuid pathing (storyId/imageId)
- PostgreSQL

Audio Transcriber

Web App that runs fully on client side to transcribe audio. Uses ffmpeg.wasm and OpenAI API to process files. It was interesting to deal with file format and size limitations on OpenAI's side while making the UX as smooth as possible. The file splitting and post-processing merging of the transcribed text needs some work. In the future I will try to add support for eleven labs and its features directly.

Links:

- More details
- Live App

Personal Trainer Bot

Telegram Bot that acts as a personal trainer with adaptable workout plans and long memory. There are multiple challenges with this project, such as filtering the messages send to the AI to prevent abuse, correctly processing the messages to also receive admin commands or reset requests by the user via natural text. Another aspect would be the seamless "infinite" memory.

Since it uses telegram's UI as a chat interface with the bot, it can't start a new conversation in the same way AI chat apps do. Instead it keeps track of the context passed on to AI with metadata about the user, plan, etc. and once this context becomes too big, it automatically summarises the conversation history and some other parts of the metadata. This way it loses details but keeps the broad context at all times. I would love to see this features as a toggle in AI chat apps.

Links:

- More Details

Sheriff of Nottingham Board Game Score Counter

Web application to help keep track of the score in the board game Sheriff of Nottingham. All client side.

Links:

- Source Code
- Live App

Postman API Monitoring Status

Small web app to show the status of Postman Monitors.

- Source Code

Reporter Tool

Small app in Rust to compress and email automated test reports.

- Source Code

Technologies

- TypeScript & JavaScript
- Golang
- Python
- PostgreSQL
- MongoDB
- GitHub Actions, AWS, Azure DevOps, Digital Ocean, Render, Vercel
- AWS Lambda
- Docker, Kubernetes
- React, NextJS, Vue.js
- Playwright, Jest
- LLMs, Agentic tasks and workflows
- AI integrations and applications

Technical Skills

Programming Languages: Go, TypeScript, JavaScript, Python, Java, C, Rust

Frontend: React, NextJS, Vue.js, SvelteKit, HTML5, CSS3

Backend & Databases: PostgreSQL, MongoDB, Redis, REST APIs, GraphQL

Cloud & Infrastructure: AWS, Azure, Docker, Kubernetes, Digital Ocean, Vercel

DevOps & CI/CD: GitHub Actions, Jenkins, Git, GitOps

Testing & Monitoring: Playwright, Cypress, Jest, Postman, OpenAPI, Swagger

AI & ML: AI API integrations, LLM applications, Agentic workflows

Operating Systems: Linux, Windows, macOS

Tools: Slack, Jira, Figma, Confluence, Notion, Linear