HENRIQUE DERVICHE KASPRZAK

DESENVOLVEDOR FULL-STACK · GO & REACT

LOCAL
Curitiba, PR — Brasil

TELEFONE / WHATSAPP
(42) 99940-5868

E-MAIL
ihiquedk@gmail.com

PORTFÓLIO
ikki.dev.br

LINKEDIN
linkedin.com/in/henrique-kasprzak

GITHUB
github.com/IkkiDK

PERFIL

Desenvolvedor full-stack e engenheiro da computação (bacharel em 2026), com dois anos em uma plataforma de cidades inteligentes em produção. Escrevo os serviços em Go que reúnem as câmeras e os sensores de uma cidade em um só lugar — ingestão de reconhecimento de placas e faces, alertas de listas de interesse, controle PTZ, telemetria por MQTT — e as telas em React com que os operadores trabalham.

Fora do trabalho, entrego sistemas completos sozinho: um sistema de inspeção e abastecimento de frota com cadeia de evidências assinada e à prova de adulteração (perto de 300 commits, 10 ADRs), um diário de manutenção agrícola que roda como um único binário Go e um TCC de IoT do sensor ao navegador. Valorizo soluções simples e diretas — um handler HTTP enxuto a um framework, entender o problema a adicionar uma dependência — e uso IA como ferramenta séria sem abrir mão de entender o que construo.

DIFERENCIAIS

- Serviços Go em produção no Kubernetes: ingestão híbrida push e polling, webhooks, MQTT, Prometheus, concorrência limitada por memória.
- Sistemas entregues de ponta a ponta sozinho: backend, PWA offline-first, banco, deploys, testes, runbooks e ADRs.
- Dados com valor de evidência: tabelas append-only, cadeias de hash SHA-256, âncoras assinadas com Ed25519 em armazenamento WORM, verificação sem banco.
- Integração software/hardware: ESP32 e sensores, câmeras LPR/ANPR e faciais, controle PTZ.

COMPETÊNCIAS TÉCNICAS

Linguagens
Go, TypeScript, JavaScript, C, C++, Java, SQL, Bash, HTML e CSS.

Front-end
React 19, Next.js 15/16, Vite, Tailwind CSS, Recharts, Radix UI, Framer Motion, Leaflet e htmx.

DevOps e cloud
Docker, Kubernetes, GitHub Actions, Caddy, systemd, Vercel, Render e MagaLU Cloud.

IoT, hardware e domínios
ESP32, Raspberry Pi, sensores ADC, visão computacional, LPR/ANPR, reconhecimento facial, câmeras PTZ e telemetria urbana.

Back-end e tempo real
REST, WebSocket, SSE, MQTT (Paho/Mosquitto), webhooks, ingestão híbrida push/polling, Chi router, microsserviços e multi-tenancy.

Dados e infraestrutura
PostgreSQL/PostGIS, SQLite, Redis, Supabase, S3/MinIO com Object Lock (WORM), migrations (Goose) e pgx.

Observabilidade e práticas
Prometheus, OpenTelemetry, logging estruturado (Zerolog), testes, ADRs, runbooks, retry/backoff, cadeias de hash e assinatura Ed25519.

Ferramentas e IA
Git/GitHub, CLI, uso avançado de LLMs para desenvolvimento e revisão de código, modelos locais (LM Studio), GIMP e Inkscape.

EXPERIÊNCIA PROFISSIONAL

Metropolys — Desenvolvedor de Sistemas

Plataforma de cidades inteligentes | 14/10/2024 – Atual | Curitiba, PR

- Autor principal de dois serviços em Go de ingestão de reconhecimento de placas (LPR/ANPR): um serviço por webhook, com download e upload de imagens, extração de metadados do veículo, verificação contra listas de restrição com alertas priorizados, deduplicação por hash SHA-256 e métricas Prometheus; e, em 2026, um ingestor híbrido de push e polling que também recebe reconhecimentos faciais — os dois caminhos derivam uma única chave de deduplicação, então nada é gravado em dobro — e que virou o serviço facial da plataforma quando os dois foram fundidos.
- Construí os alertas de listas de interesse para placas e faces, do alarme da plataforma de câmeras até a tela do operador, incluindo anexar o snapshot da passagem, que só chega à plataforma 14 a 22 segundos depois do alarme.
- Entreguei as telas de placas e faces do produto em React e TypeScript: pesquisa e rastreamento de placas, mapa de rastro com visualizador de passagens, visão de proximidade, placas monitoradas agrupadas por prioridade de alerta, alertas faciais e cadastro com categorias na mesma escala de prioridade.
- Autor do serviço de controle de câmeras PTZ (pan-tilt-zoom) em Go e do overlay em React 19 — dial direcional, zoom/foco/íris e hooks de comando próprios renderizados sobre o vídeo ao vivo.
- Criei simuladores de telemetria urbana em Go (semáforos com modos de falha, iluminação pública, rastreamento GPS por rotas GeoJSON, detecção facial) para testar o pipeline inteiro de ponta a ponta sem hardware físico.
- Desenvolvi o serviço de agregação de indicadores (KPIs) que transforma dados de tráfego, clima e infraestrutura em métricas padronizadas multi-tenant, incluindo totais móveis de chuva a partir do histórico das estações; ajustei a produção no Kubernetes (concorrência limitada por memória após um OOM, retomada de cursor no reinício, sondas de readiness).
- Contribuí em transporte público em tempo real, telemetria de sensores IoT e iluminação pública via MQTT e nas fontes de câmera do gateway de vídeo, e construí o mosaico de câmeras do painel de operações (blocos arrastáveis, presets, busca, mapa Leaflet).

Go **TypeScript** **React 19** **PostgreSQL/PostGIS** **Redis** **MQTT** **WebSocket** **SSE** **Docker** **Kubernetes** **Prometheus**

PROJETOS EM DESTAQUE

Xanadu Fleet — Inspeção, abastecimento e manutenção de frota

Autor único, 2024 – atual · perto de 300 commits · 10 Architecture Decision Records

- Checklists digitais versionados (58 itens para caminhões, 19 para máquinas, carretas vinculadas) executados na entrega e na devolução do veículo; cada inspeção fixa a versão exata do checklist contra a qual rodou.
- PWA offline-first com fila em IndexedDB e API REST de sincronização neutra em relação ao framework; a ingestão tolera sincronização fora de ordem, coloca em quarentena registros com relógio defasado e recebe fotos por um canal separado de upload pré-assinado com conferência do hash declarado, de modo que um registro nunca fica preso às suas imagens.
- Correções são novos registros que substituem o original — nunca edições —, com o vínculo de substituição selado em uma cadeia de hashes SHA-256 por tenant sobre tabelas append-only.
- Ancoragem de evidências: um worker em segundo plano assina cada hash de registro com Ed25519 em um bucket com Object Lock (WORM); o atraso de ancoragem serve de alarme de backup, um resumo assinado é gravado diariamente e a ferramenta `evidence-verify` confere o log sem banco de dados.
- Módulo de abastecimento (diesel e ARLA, foto do canhoto selada na captura), painel do gestor com relatórios de combustível e por divisão, e conciliação com o ERP da empresa — resultado por filial cruzando documentos de frete.
- CI roda `go vet`, `go build` e `go test` contra um Postgres novo a cada push; mais de 50 migrations (Goose), workflow de deploy sob demanda, CSP de base, exportações de dados pessoais com limite de taxa e auditoria, runbooks de deploy, recuperação de desastre e ancoragem de evidências, e três manuais de usuário.

Go **PostgreSQL** **htmx** **PWA offline-first** **S3 Object Lock** **Ed25519** **Goose** **GitHub Actions** **MagaLU Cloud**

Manutenção Lavoura — Diário de manutenção de máquinas agrícolas

Autor único, 2026 · um binário Go e um arquivo SQLite, em produção

- Registra o que foi feito, quando e com qual leitura de horímetro ou hodômetro para trator, colheitadeira, plantadeira, semeadora, pulverizador e caminhão; cada visita lista serviços com preço e "repetir após N horas/km", e a tela de gastos totaliza custo por máquina e por serviço, agrupando nomes sem distinguir maiúsculas, acentos ou espaçamento.
- O formulário de novo registro funciona offline com cache no service worker e fila no aparelho; as fotos são redimensionadas no celular, enfileiradas à parte e enviadas logo depois do seu registro, que nunca fica bloqueado por elas.
- Apps instalados se atualizam sozinhos: o cache do service worker recebe o nome de um hash dos arquivos que guarda, então qualquer build que os altere dispara a atualização e o reload, sem versão para incrementar à mão.
- Implantado como binário estático (`CGO_ENABLED=0`, SQLite em Go puro) sob systemd atrás do Caddy em um VPS; as migrations rodam em transações na inicialização e recusam um schema pela metade; o workflow de release reexecuta os testes, compila, tira snapshot do banco e instala por SSH.

Go **SQLite** **htmx** **PWA** **Caddy** **systemd** **GitHub Actions**

Mini Estufa — TCC · IoT full-stack em tempo real

Monitoramento de estufa integrando hardware, sensores e aplicação web

- API em Go (autor único) recebendo telemetria do hardware (ESP32) via HTTP e distribuindo em tempo real por WebSocket, com handlers `/api/sensor/push`, `/api/sensor/latest`, `/ws` e `/health`.
- Dashboard em React 19 + Vite + Tailwind com gráficos em Recharts (temperatura, umidade do ar, luminosidade, umidade do solo e status de bomba/luz) e histórico persistido no Supabase, com filtros por período e tratamento de fuso horário.
- Montagem física, leitura e calibração de sensores ADC, automação da bomba de irrigação e pipeline MQTT/Mosquitto; API no Render, dashboard na Vercel.

Go **gorilla/websocket** **React 19** **Vite** **Tailwind** **Recharts** **Supabase** **MQTT** **Render** **Vercel**

Xanadu Transportes — Site institucional

Autor único · reconstruído em Next.js 15 em 2026 para pré-renderização e SEO

- Site totalmente pré-renderizado no App Router, com server components por padrão, `next/image`, seções em Framer Motion e vídeo de abertura no Cloudinary com um corte mais leve para celulares.
- Chamadas para ação só por WhatsApp, registradas como eventos; Google Analytics 4 carregado apenas após o consentimento de cookies (LGPD), com página de política de privacidade.
- Redirecionamento permanente do domínio de preview da Vercel para o domínio canônico, mais metadados canônicos e Open Graph, para que os buscadores indexem um único endereço.

Next.js 15 **React 19** **Framer Motion** **Cloudinary** **Google Analytics 4** **Vercel**

FORMAÇÃO, CURSOS E IDIOMAS

Universidade Positivo

Bacharel em Engenharia da Computação | 2021 – 2026 | concluído

Português: nativo.

Inglês: leitura, escrita e conversação.

Go (Golang): Udemy, 60 horas

React com APIs REST: Udemy, 30 horas

GitHub: Udemy, 20 horas

Dale Carnegie: comunicação, liderança e relações interpessoais