Justin Hofmann

Platform & Backend Engineer — Java/Quarkus · OpenResty · Docker · Vue/TypeScript

Brandenburg (remote-first) · hofmannjustinbenedikt@gmail.com · justinhofmann.dev · github.com/Nexinox

PROFIL

Sechs Jahre Aufbau und Betrieb einer On-Prem-Microservice-Plattform, die Poller in Städten smart macht — Zufahrt per RFID und Telefonanruf, zeitlich begrenzte Berechtigungen, lückenlose Historie — im Einsatz in rund zehn Städten. Während der Migration von einem Legacy-Monolithen habe ich das API-Gateway, die Authentifizierung und die Deployment-Schicht der Plattform gebaut; zuletzt habe ich ihr Backup/Restore- und ihr Update-System im Alleingang entworfen und ausgeliefert und reviewe querschnittliche Architekturänderungen über ihre 15+ Services. Meine Stärke ist Orchestrierung und Integration an Systemen mit hohem Schadenspotenzial: koordinierte Abläufe über zustandsbehaftete Systeme ohne gemeinsame Transaktionen, Rollback-Design und Release Engineering.

BERUFSERFAHRUNG

GEMTEC GmbH — Fachinformatiker für Anwendungsentwicklung

seit 08/2021

Festanstellung; Ausbildung im selben Unternehmen 09/2018 – 08/2021

On-Prem-Smart-City-Plattform: 15+ Quarkus-Microservices, 10 Vue-Frontends, Postgres, MQTT, Graylog/OpenSearch, Proxmox. 1.193 Commits in 45 Repositories. Die Verantwortung wuchs von Feature-Arbeit zu Plattform-Ownership — im Folgenden nach verantworteten Systemen gegliedert, nicht nach Jahren.

Plattform-Infrastruktur & Migration (2021–2023)

- **JWT-Authentifizierung** und **Consul Service Discovery** der Plattform gebaut, während der Migration von einer modularen OSGi/Vaadin-Plattform zu containerisierten Microservices — begonnen noch als Auszubildender.
- **Hauptautor des OpenResty-API-Gateways** vom ersten Commit bis heute (49 % aller Commits über 5 Jahre). Die Wahl von OpenResty gegenüber HAProxy war meine Entscheidung; JWT-Verifikation in Lua, Routing pro Service, später vollständige CRUD-Services im Gateway sowie Betriebsmodi (Restore-Modus, Status-Seite ohne Login).
- **Alleinautor (108/108 Commits) der Nomad/Consul-Deployment-Schicht**: Job-Specs pro Service, Abhängigkeits-Reihenfolge der Dienste, Memory Oversubscription, Log-Verwaltung, Installer-Skripte. Zwei Jahre später habe ich dafür argumentiert, **meine eigene Schicht durch Docker Compose zu ersetzen** — ich war ihr einziger Maintainer, und ein Stack, den nur ich betreiben konnte, war ein Risiko — und die Migration umgesetzt; das Compose-Deployment läuft bis heute in Produktion.
- Services anderer Teams containerisiert und Integrationstest-Fehlschläge als CI-Gate verdrahtet.

Backup & Restore mit versionsübergreifender Migration (2025–2026, solo)

- Plattformweites Backup/Restore-System entworfen und gebaut: ein **Saga-Orchestrator** mit 14 sequenzierten Restore-Schritten und Rollback in umgekehrter Reihenfolge, der sechs zustandsbehaftete Systeme koordiniert (Postgres, Graylog/MongoDB, OpenSearch, Node-RED, Gateway-Konfiguration), die keine gemeinsamen Transaktionen haben. Restore auf andere Maschinen und **neuere Produktversionen**, inklusive Datenmigration zwischen geänderten Backend-Schemata. Seit 11/2025 bei Kunden im Einsatz.
- Sicherheitsdesign: Restore in temporäre Wegwerf-Datenbanken mit anschließender Promotion; formatversionierte Archive mit automatisch aktualisierendem Descriptor-Migrator; ein Availability-Gate, das das Feature vollständig sperrt, wenn Vorbedingungen fehlen.
- Die funktionierende, bereits ausgelieferte v2 auf die Saga-Architektur umgeschrieben (~13.000 Zeilen entfernt, ~16.000 hinzugefügt in drei Wochen), bevor ihre Komplexität unwartbar wurde.
- Durchgängig **Streaming-I/O statt Read-then-Write** eingeführt, um Speicherüberläufe bei großen Restores zu verhindern — nebenbei wurden die Läufe **50–100× schneller** und deutlich zuverlässiger.

In-Place-Produkt-Updater (2026, solo)

- Das System 2024 auf einem Architektur-Board entworfen (Deployment-Topologien, Update-Protokoll, Risiko-Pre-Mortem); 2026 in zwei Monaten von Grund auf gebaut.
- **Blue-Green-Deployment auf Proxmox** (in Arbeit): zwei VMs auf gemeinsamem NFS-Speicher, von denen immer nur eine läuft — die Installation behält eine IP und eine MAC und braucht keinen vorgelagerten Load Balancer. Basiert auf Proxmox-Snapshot/Restore und dem obigen Backup/Restore-System; der schwierige Teil ist der Umzug zwischen Proxmox-Hosts mit abweichenden Namen und fehlenden ISOs.
- Ein **GraalVM-nativer Quarkus-Service**, der den gesamten On-Prem-Stack aktualisiert, inklusive Proxmox-VMs — und sich selbst, per versioniertem „Staffelstab"-Handoff an seinen Nachfolger. Resumable Uploads (tus) und Online-Paketbezug, SSE-Fortschritt, Readiness-Gates vor dem Commit, Rollback-Semantik, Speicherplatz-Guards. Ergänzt um ein eigenständiges Status-Frontend ohne Login, das genau dann dient, wenn Haupt-UI und Authentifizierung bewusst offline sind.
- Zusätzlich das letzte JVM-only-Backend der Plattform (Asterisk-Telefonie-Treiber) als **Native Image** lauffähig gemacht: diagnostiziert, dass eine Bibliothek ihre Protokollversion per Laufzeit-Reflexion über ~6.000 generierte Klassen aushandelt; die Version konfigurierbar gemacht und 1.216 Reflection-Registrierungen generiert. Boot ~50 ms; Speicher ~100 MB statt ~350 MB auf der JVM.

Plattformweites Code-Review & Release Engineering (2026)

- **Reviewer für querschnittliche Änderungen: 63 Merge Requests in 18 Repositories** — z. B. ein Architekturmuster an einem einzigen Tag über 8 Backends hinweg reviewt. Funde u. a.: release-blockierende API-Vertragslücken zu den Frontends, doppelte DB-Migrationsversionen, ein globaler Mutex über Netzwerkaufrufe hinweg.
- Den zuvor **undokumentierten manuellen Release-Prozess über 18 Repositories reverse-engineert und dokumentiert** (Abhängigkeits-Reihenfolge, Versionierung, Registry-Promotion) und die Automatisierungspotenziale aufgezeigt.

iLEM — erstes Cloud-Deployment des Unternehmens (AWS, ~2022, 6 Monate)

- Deployment eines Greenfield-Produkts für intelligentes Lade- und Energiemanagement (Solar, Ladeinfrastruktur, Speicher) komplett verantwortet: containerisierte Services auf **AWS ECS mit ECR und RDS**, IAM-Rollen, Netzwerkisolierung — aufgebaut ohne vorherige Cloud-Erfahrung im Haus; parallel beratend am Anwendungscode beteiligt. Showcase-Umgebung für Kunden aufgesetzt, die bis heute läuft. Das Produkt wurde mangels Nachfrage eingestellt.

Produkt- & Geräteintegration (2022–2026)

- **Estate-weite Vue-2 → Vue-3-Migration geleitet und umgesetzt** (Webpack → Vite, State-Management auf Pinia) über die Anwendungs-Frontends und die gemeinsame Bibliothekskette — inklusive Neuaufbau des Frontend-Build-Stacks (2025). Gemeinsam mit dem Frontend-Team anschließend entschieden, die Micro-Frontend-Architektur abzulösen; die modulbasierte Konsolidierung ist zu ~60 % umgesetzt.
- Full-Stack-Features über 10 Vue-Frontends und ~10 Backends; Owner der gemeinsamen Frontend-Bibliothekskette, von der jede Anwendung abhängt.
- Geräteintegration für physische Infrastruktur: MQTT-Messaging, IO-State-Handling, Heartbeats und Treiber-Abstraktionen für Poller; angrenzende Systeme: Zutrittskontrolle, Brandmeldeanlagen, Schwesternruf, VoIP/Telefonie, SPS.
- Architektur- und Interaktionsdesign in Figma: eine vollständige Architektur der nächsten Plattform-Generation (Service-Katalog, MQTT-Protokolldesign mit Payloads, UML-Abläufe — fünf der sechs vorgeschlagenen Services wurden anschließend gebaut) sowie ein 33-Screen-UI-Redesign auf Basis von Material Design 3.
- **Auszubildende und Werkstudenten kontinuierlich betreut (seit 2021, selbst initiiert):** Einarbeitung ins System, gemeinsame Code-Durchsprachen mit Begründung, was gut oder schlecht ist und warum, sowie gemeinsam geleitete Architektur-Design-Sessions.

KENNTNISSE

Backend: Java (6 J.; Quarkus — Jakarta EE/MicroProfile, direkt auf Spring Boot übertragbar; portiere derzeit privat einen Service auf Spring Boot), REST-API-Design & Versionierung, Saga-/Orchestrierungsmuster, GraalVM Native Images, SSE, Resumable Uploads (tus), JWT-Authentifizierung

Infrastruktur: Docker & Compose, NGINX/OpenResty + Lua, GitLab CI, HashiCorp Nomad & Consul (Scheduling-/Service-Discovery-Konzepte auf Kubernetes übertragbar), AWS (ECS, ECR, RDS, IAM — ein vollständiges Produkt-Deployment), Release Engineering, Proxmox, Postgres-Betrieb, Graylog/OpenSearch

Frontend: Vue 2/3 + Vuetify, TypeScript, Pinia (5 J., 10 Anwendungen)

Protokolle & Geräte: MQTT (Design und Implementierung), binäre Geräteprotokolle, Node-RED, Asterisk/ARI

Außerdem: Go (zwei vollständige Service-Reimplementierungen, nicht produktiv), Python, Bash, SQL/Flyway, Vaadin 7/8 (Java-UIs aus der Ausbildungszeit; dazu ein FIDO2/WebAuthn-Login-Projekt auf GitHub)

Sprachen: Deutsch (Muttersprache), Englisch (fließend — täglicher schriftlicher Gebrauch)

PRIVATE PROJEKTE

- **centurio-mqtt (Go):** Brücke von einem Batteriespeicher-/Wechselrichtersystem zu MQTT mit Home-Assistant-Auto-Discovery — selbst entwickelte binäre Protokollschicht mit Unit-Tests, Inselbetrieb-Steuerung, Erkennung von Speicher-Engpässen.
- **Hausautomatisierung von der Firmware aufwärts:** eigene Python-Firmware für schaltbare Steckdosen (MQTT, Relaissteuerung, Web-UI), Solardaten-Erfassung, lokale LLM/RAG-Experimente.
- **Lernen durch Reimplementierung:** einen produktiven User-Management-Service komplett in Go nachgebaut (JWT-Access/Refresh-Auth, Rollen/Berechtigungen, Migrationen, dokumentierte API); aktueller Neuaufbau in Spring Boot.

AUSBILDUNG

- **Ausbildung zum Fachinformatiker für Anwendungsentwicklung (IHK)**, GEMTEC GmbH, 09/2018 – 08/2021
- Schulabschluss nach der 10. Klasse, 2018