WORK SAMPLES

Over several years of platform and product development, I built and continuously improved a complete system landscape around influencer marketing, e-commerce, and B2B portals.

The examples below show key projects with a focus on:

- **scalable architecture** (microservices, GraphQL backbone, clustered databases)
- **data-driven features** (search, analytics, OLAP, conversion tracking)
- **reliable operations** (monitoring, backups, SSO, infrastructure automation)

1. PLATFORM & ARCHITECTURE

Platform overview

- **System landscape:** around 38 applications and services (frontends, backends, microservices, infrastructure).
- **Backbone:** central GraphQL APIs (GraphQL/GQL) as the data hub for portals, dashboards, and tools.
- **Core components:** SSO/logins, CDN/upload, Solr search, model server (ML/OCR), tracking services, cron orchestration.
- **Frontends:** Svelte/SvelteKit and Next.js/React; all consume GraphQL APIs.
- **Operations:** Nagios monitoring, cron utilities for jobs (backups, crawling, feeds), blue-green deployments, Hetzner Cloud infrastructure.

Microservice architecture & system landscape

- Built a microservice architecture in which functional areas (frontends, backends, ML, CDN, crawler, SSO, upload, tracking) run independently.
- Services communicate via APIs behind load balancers and can be scaled separately.
- Around 25 Hetzner instances run in a private network that isolates internal traffic from the public internet.
- **Data layer:** Percona XtraDB Cluster, MySQL/MariaDB, ClickHouse, Solr.
- **Infrastructure:** Nginx, PHP-FPM, Node.js services, Python APIs, GlusterFS, CDN storage.

Backup strategy & maintenance

- Introduced a backup strategy that automatically and time-scheduled backs up application data and databases.
- Backups are created locally, encrypted, and synchronized to separate backup servers.
- Performed regular restore tests to verify data integrity.
- Monitoring (Nagios) uses status files and custom checks to detect issues early.
- The microservice setup enables parallel development and fast rollouts for new features and bug fixes.

Infrastructure (short overview)

- **Server setup:** module installer for PHP/Node/Solr/ClickHouse/GlusterFS.

- **Server provisioning:** Hetzner IaC with hcloud CLI and cloud-init (networks, load balancer, IP scheme).
- **Solr:** multi-core search system for portal and typeahead.
- **Roehrig Nagios:** monitoring for 19 hosts (app, GraphQL, SSO, upload, DB).
- **Nagios plugins:** custom checks for web, HAProxy, RAID, DNS, SSL.
- **Cron utilities:** orchestration of backups, feeds, crawling, campaign jobs, sitemaps.

2. BELLISSY COMMERCE & PORTAL

Rebuild: beauty portal & content monetization platform

Goal: Modernize the existing portal, increase performance, and monetize content.

- Rebuilt bellissy.de and “Shop the Look” (Next.js/React, GraphQL).
- Migrated infrastructure to Hetzner Cloud (Nginx, Percona XtraDB Cluster, Solr, load balancer).
- Set up CI/CD pipelines with GitLab and automated releases.
- **Result:** according to Sistrix, visibility increased by ~250% after the relaunch.

Technology stack: Hetzner Cloud, Nginx, React, Next.js, GraphQL, Percona XtraDB Cluster, Solr, load balancing, GitLab CI/CD

Automated import of product feeds & availability

Goal: Centrally manage product data from various shops and keep it up to date.

- Built an import pipeline for shop feeds (CSV/JSON/XML) that ingests data into MySQL.
- Implemented validation logic (format, stock, prices) to filter faulty products early.
- Modeled product status so that search, frontend, and campaigns can access it directly.

Technology stack: PHP, MySQL

Solr search & advanced search features

Goal: Rebuild search and optimize it for large product volumes.

- Integrated Solr to enable full-text search and faceted filters (price, brand, category).
- Developed the search results page with filters, sorting, and facets.
- Implemented a zero-results page that guides users to alternative categories, brands, and products.

Technology stack: Solr, typeahead, React

Editorial system & monetization (CraftCMS, AdSense)

Goal: Simplify content maintenance and increase ad revenue.

- Introduced an editorial system with CraftCMS (guides, editorials, landing pages).
- Optimized media delivery and image management via CMS and CDN.
- Integrated Google AdSense ads and placed them to keep visibility high without degrading UX.

Technology stack: CraftCMS, GraphQL, Google AdSense, React

Price alerts, availability notifications & newsletter

Goal: Retain users and increase portal traffic through notifications.

- Implemented wishlists and alerts for price and availability changes.
- Built a double opt-in flow for newsletters and notifications.
- Connected email delivery via GraphQL-based services and Mailjet.

Technology stack: GraphQL, Mailjet API, React

Cart pixel & conversion tracking

Goal: Build in-house conversion measurement and reduce external affiliate costs.

- Developed a tracking system that captures clicks and purchases in partner shops.
- Provided a JavaScript snippet for merchants to embed on their order confirmation pages.
- Stored conversions and cart details centrally in the database and analyzed them.

Technology stack: PHP, Nginx, JavaScript

B2B portal: laserschneiden24.de

Goal: Digitize quoting and ordering for sheet■metal cut parts.

- Built a HubSpot integration to automatically import customer data into the portal.
- Implemented upload and processing of STP and DXF files, including automated “unfolding” of technical details.
- Implemented quotation creation and order management with status tracking in the portal.

Technology stack: PHP, Nginx, HubSpot, Mailjet API, CDN, WiCam API, STP, DXF, Hetzner Cloud

3. CAMPAIGN MANAGEMENT & ANALYTICS

Campaign Self Service (CSS) & scheduling

Goal: Enable customers to plan and manage campaigns themselves.

- Developed a self-service tool for scheduling, creating briefings, and monitoring campaigns.
- Built a workflow that suggests suitable influencers and measures campaign success via shortcodes and cart pixel.
- Integrated a calendar view with FullCalendar.

Technology stack: Svelte, FullCalendar

Performance tracking for customer campaigns

Goal: Make campaign performance visible per customer and influencer.

- Built a shortlink system that redirects to arbitrary target URLs and records page views as conversions.
- Generated individual shortlinks for influencers and campaigns.
- Designed the data model to enable attribution and reporting per customer, campaign, link, and influencer.

Campaign analytics UI (ClickHouse reporting)

Goal: Evaluate conversion data quickly and flexibly.

- Migrated conversion data from MySQL to ClickHouse to significantly speed up queries.
- Developed the reporting interface in Svelte with separate views for customers and the customer success team.
- Integrated CSV exports so teams can continue working with the data in BI tools.

Technology stack: ClickHouse, Svelte, GraphQL, CSV

Building a beauty community

Goal: Introduce community features for members and influencers.

- Developed a token-based login system for members and influencers.
- Expanded wishlists to automatically trigger price and availability alerts.
- Implemented social features: users can like posts and follow influencers.

Technology stack: PHP, GraphQL, MySQL, Mailjet API, React

4. IDENTITY, MESSAGING & CRM

Single Sign-On (SSO) for the platform ecosystem

Goal: Central login for all applications.

- Developed a passwordless SSO system: login via one-time code by email instead of a password.
- Managed sessions via cookies including CSRF protection and a per-application role model.
- Synchronized customer data with HubSpot to automate user management.

Technology stack: PHP, Percona XtraDB Cluster, Nginx, Mailjet API, HubSpot, Hetzner Cloud

Mailgun integration as an email microservice

Goal: Centralize and automate email sending.

- Integrated Mailgun as the central email service for newsletters and transactional emails.
- Rendered templates server-side; implemented double opt-in and unsubscribe flows.
- Created custom API endpoints to connect business logic (alerts, campaigns, onboarding) with the mailing system.

Technology stack: Mailgun API, PHP, Nginx, cron, Hetzner Cloud

WhatsApp chat gateway

Goal: Integrate WhatsApp communication into CRM processes.

- Implemented a Node.js/Express server as a bridge to the WhatsApp Business Cloud API.
- Processed incoming and outgoing messages (text and media) and logged delivery status in real time.
- Implemented webhook verification and signature checks for secure integration.

Technology stack: Node.js, Express, WhatsApp API

5. DATA, ML SERVICES & CRAWLING

Model server for ML APIs

Goal: Centrally provide machine-learning models.

- Built a Python-based model server that exposes multiple ML models as APIs within the private network.
- Implemented a Gunicorn setup with custom middleware; kept models as read-only structures in RAM to keep response times low.
- Integrated OCR and analysis endpoints e.g., for Instagram story insights.

Technology stack: Python, Gunicorn, OpenCV, Tesseract, Hetzner Cloud

Crawler & content recording

Goal: Automatically collect social media data for analytics and campaigns.

- Developed browser-based crawlers with remotely controlled Chrome (CDP).
- Built mobile-app-based crawlers via Appium to automatically navigate profiles in the Instagram app.
- Ingested data into proprietary databases and CRM systems (HubSpot).

Technology stack: Docker, Node.js, Chrome Remote Interface, Appium, HubSpot

COVID test-center price comparison portal

Goal: Create an overview of COVID test centers and prices in Germany.

- Set up a cloud crawling instance that regularly reads data from 12+ provider websites.
- Implemented postprocessing: detect new cities, geocode addresses, clean data.
- Optimized tracking via Google Tag Manager and provided a self-service API for test center providers.

Technology stack: Node.js, Chrome Remote Interface, Nginx, PHP, Solr, Google Tag Manager, Cookiebot, Hetzner Cloud

6. ADDITIONAL PROJECTS & SPECIAL SOLUTIONS

Content Delivery Network (CDN) & upload service

Goal: Deliver media with high performance and controlled access.

- Developed an in-house CDN with upload and download service.
- Implemented permissions per application and asset type (avatars, campaign images, briefings, etc.).
- Implemented an image pipeline with automatic resizing, compression, and content-addressed storage (SHA■256).
- Used Nginx X-Accel-Redirect to deliver files efficiently and securely.

Technology stack: PHP, Nginx, SSO, GlusterFS, Hetzner Cloud, ImageMagick

Villagari – vacation home portal

Goal: Implement a luxury vacation home web presence with a fast static frontend and secure CMS.

- Set up a WordPress-based Bedrock installation as the CMS backend.

- Implemented static HTML generation with a separate frontend (fast, CDN-friendly hosting).
- Modeled multilingual content seasonal pricing logic, and structured data.
- Developed Git-based deployment and static build processes with rollback options.

Technology stack: PHP, WordPress/Bedrock, MySQL, Nginx/PHP-FPM, wp2static, Git

7. SYSTEM OVERVIEW BY DOMAIN (EXCERPT)

Frontend applications (selection)

- **Campaign Self Service:** self-service tool for campaigns (SvelteKit, HubSpot, Mailgun, JSON storage).
- **Portal-Next:** e-commerce portal with SSR for bellissy.de (Next.js, GraphQL).
- **Campaign GUI:** real-time analytics for campaigns (SvelteKit, ClickHouse data).
- **Client Campaign GUI:** customer dashboards with CSV export.
- **Campaign Control:** admin tool for campaigns, shortlinks, influencer assignment.
- **GetInspired (STL):** creator portal with roles, CSRF protection, and IBAN handling.
- **Roehrig Frontend:** B2B portal with 3D preview and SSO.
- **Insights:** upload UI for Instagram insights with OCR evaluation.
- **Price-Check Frontend:** portal for test-center search (Solr, Mapbox).
- **Villagari:** WordPress-based rental CMS with static delivery.

Backend & APIs (selection)

- **GraphQL:** main API for products, campaigns, influencers, conversions (MySQL, Solr, ClickHouse).
- **GQL:** campaign-specific GraphQL instance with RBAC.
- **Old Backend:** Symfony-based legacy for tracking and catalog.
- **Roehrig Backend:** GraphQL API for quotes, orders, customers.
- **Bellissy Framework:** custom PHP MVC framework (DB abstraction, validator, CSRF).

Microservices (selection)

- **Model Server:** ML/OCR service for categorization, relevance, and story evaluation.
- **Bellissy CDN:** upload and image pipeline with SSO/RBAC and X-Accel-Redirect.
- **Roehrig Upload:** upload service with queue, roles, and checksums.
- **SSO / Login / Roehrig-Login:** passwordless login with token, CSRF, rate limiting, HubSpot sync.
- **HubSpot:** CRM sync plus social media crawler.
- **WhatsApp:** Cloud API bridge with status tracking and 24h-window logic.
- **Feed Tools:** 52 supplier feeds → DB + Solr.
- **COVID Crawler:** test-center crawling and geocoding.
- **Image Tools:** image pipeline for product photos.
- **Mailgun:** newsletters and transactional emails as a microservice.

- **Shop Redirect / Shopin / Tracking Pixel:** shortlinks, attribution, server-side conversion tracking.

Infrastructure

- **Server setup:** module installer for PHP/Node/Solr/ClickHouse/GlusterFS.
- **Server provisioning:** Hetzner IaC with hcloud CLI and cloud-init.
- **Solr:** multi-core search system (portal + typeahead).
- **Roehrig Nagios:** monitoring for 19 hosts.
- **Nagios plugins:** custom checks for web, HAProxy, RAID, DNS, SSL.
- **Cron utilities:** orchestration of backups, feeds, crawling, campaign jobs, sitemaps.