Andreas K. Weber
Fractional CTO & Principal Architect
hi@akw.dev · akw.dev
Provided via apiland GmbH

PRINCIPAL PROFILE & TRACK RECORD

I own the delivery outcome. We will ship.

Most engineering initiatives fail not from a lack of technical skill, but from the gap between strategic intent and production reality. Rewrites stall, architectures drift, and teams lose momentum when nobody owns the full path from decision to delivery. I step into complex, scaling, or distressed environments with a clear mandate: align strategy, architecture, team execution, and operations until the work ships.

FRACTIONAL CTO

I take ownership of technical direction, delivery cadence, and engineering decision-making when the organization needs senior leadership without waiting for a permanent CTO hire.

PRINCIPAL ARCHITECT

I design systems that stay aligned with the business model, using Domain-Driven Design to make boundaries, dependencies, and modernization paths explicit.

VERTICAL EXECUTION

I bridge strategy, architecture, implementation, deployment, and operational feedback. The role is not advisory only: I stay close enough to production to make decisions real.

The Antidote to Engineering Pain

VACUUM ENGINEERING KILLS

Ship to Learn. Technical decisions have to meet market and operational feedback quickly. I drive short delivery loops so assumptions are tested before they become expensive commitments.

HYPE BURNS CASH

Boring Technology. Innovation tokens are limited. I favor proven stacks and simple operational models unless a new technology creates a clear business advantage.

MEETINGS DESTROY FOCUS

Default to Async. Delivery improves when decisions are written down and context is reusable. I use documentation and asynchronous alignment to protect deep work.

Engagement Snapshot

MANDATES	Fractional CTO, Principal Architect, Turnaround Lead	TARGET ENVIRONMENTS	Legacy Modernization, Scale-ups (Series A+), Distressed Architectures
EXPERTISE	Domain-Driven Design, Cloud Architecture, Systems Engineering	WORKING MODEL	Remote-first, async-default. Strategic on-site presence.
AVAILABILITY	Limited capacity. Currently booking for Q3 2026.	CONTRACT SETUP	B2B Retainer or Interim Contract (apiland GmbH)

GREENFIELD PLATFORM OWNERSHIP

PadelCity Platform Development & Operations

Client: PadelCity · Role: Fractional CTO & Principal Architect · Period: 2023–2024

End-to-end technical ownership for PadelCity's own booking and operations platform.

DOMAIN	COMPLEXITY	PRIMARY RISK	TIME-TO-IMPACT
Sports Venue Booking	Greenfield Platform + Operations	Competitor dependency	Production launch and scale

THE SITUATION

PadelCity needed to move from dependence on an external competitor-operated system to its own digital platform for renting out sports courts, managing venues, and supporting daily operations across app and back-office workflows.

THE CHALLENGE

The mandate required more than application development. The company needed full systems architecture, a domain model, delivery infrastructure, production operations, identity management, API design, and developer experience that could take a greenfield system into reliable commercial use.

KEY STRATEGIC DECISIONS

Built the platform around a clear venue operations and booking domain, with a Symfony/API Platform backend exposing REST APIs for the customer app and back-office portal. Established GitLab-based delivery pipelines, test-driven backend development, and cloud operations from the beginning instead of treating infrastructure as an afterthought.

EXECUTION & SCOPE

- Acted as fractional CTO and principal architect with ownership of systems architecture, domain modeling, delivery, and operations.
- Owned the GitLab development infrastructure, CI pipelines, and test-driven backend delivery model.
- Operated the AWS production infrastructure, including Kubernetes, PostgreSQL, and Keycloak.
- Built the API backend for app, booking, venue, and back-office workflows.
- Improved developer experience through local tooling, repeatable environments, and deployment automation.

TECHNICAL ENVIRONMENT

PHP, Symfony, API Platform, REST APIs, GitLab CI/CD, AWS, Kubernetes, PostgreSQL, Keycloak, Docker.

PHP · Symfony · API Platform · REST API · GitLab CI/CD · AWS · Kubernetes · PostgreSQL · Keycloak · Docker

Outcome & Handover: Took the platform from greenfield build to production operation, supported rollout across venues, and scaled it to roughly 50,000 users. Once the platform was established and no major strategic changes were planned, ownership was handed over cleanly and the engagement ended.

LEGACY RESCUE & TRANSITION

Legacy PHP Platform Sunset & Replacement

Client: Employer Benefits Provider · Role: Technical Lead & Migration Architect · Period: 2025

Migration of a telecom benefits platform from a legacy PHP/Nuxt system to a modern Next.js and TypeScript implementation.

DOMAIN	COMPLEXITY	PRIMARY RISK	TIME-TO-IMPACT
Employer Benefits & Telecom Sales	Storefront + Back Office + BI	Vodafone API and compliance change	Zero-downtime transition

THE SITUATION

The company provided employer benefits to employees through a telecom sales platform connected to Vodafone. The existing system combined an old PHP 5.4 backend with two Nuxt.js frontends: a customer-facing storefront and an internal back office for call-center, sales, and fulfillment workflows.

THE CHALLENGE

The legacy system could no longer support the requirements of the evolving Vodafone API and compliance landscape. Replacing it required migration of product data, lead flows, sales workflows, business intelligence capabilities, and operational processes without interrupting the business.

KEY STRATEGIC DECISIONS

Rebuilt the platform on a clean Next.js and TypeScript architecture with test-driven development and GitLab CI pipelines. Integrated the complete Vodafone product portfolio while preserving the storefront, back-office, call-center, lead-management, and reporting workflows needed for daily operations.

EXECUTION & SCOPE

- Led the transition from a PHP 5.4 backend and two Nuxt.js frontends to a modern Next.js/TypeScript platform.
- Mapped and migrated storefront, back-office, call-center, lead-management, and fulfillment workflows.
- Integrated the Vodafone API and full product portfolio into the replacement system.
- Established a test-driven setup with GitLab CI pipelines and cleaner architectural boundaries.
- Prepared the system for internal team ownership after completion.

TECHNICAL ENVIRONMENT

Legacy PHP 5.4, Nuxt.js, Next.js, TypeScript, GitLab CI, Vodafone API, Ruby lead management, BI workflows.

PHP 5.4 · Nuxt.js · Next.js · TypeScript · GitLab CI · Vodafone API · Ruby · BI

Outcome & Handover: Replaced the legacy PHP/Nuxt platform with a maintainable Next.js/TypeScript implementation, migrated the required data and workflows, and completed the switch without downtime. The platform was handed over to the internal team for ongoing maintenance.

DATA-LED LEGACY MODERNIZATION

Seven Heaven Food Delivery Platform Rewrite

Client: Seven Heaven · Role: Principal Architect & Technical Lead · Period: 2026

Modernization of a multi-location food delivery platform from legacy PHP and trapped operational data to a canonical domain model and tested TypeScript application.

DOMAIN	COMPLEXITY	PRIMARY RISK	TIME-TO-IMPACT
Food Delivery Operations	Legacy Data + Domain Rewrite	Data quality and workflow continuity	BI first, rewrite second

THE SITUATION

Seven Heaven operated a food delivery business across five locations, primarily serving company lunch orders through a small storefront. The business had outgrown a legacy PHP 5.6 system where workflows were held together through ad hoc descriptions, manual conventions, and operational workarounds.

THE CHALLENGE

The system was bleeding money because important customizations were hard or impossible to implement. Menus were duplicated across locations, operational data was trapped in a poorly designed and outgrown database schema, and the business could not reliably analyze what was happening.

KEY STRATEGIC DECISIONS

Started with a dbt-based data migration and cleanup pipeline, then placed Evidence on top as a business intelligence layer. The resulting insight informed a canonical Domain-Driven Design model that replaced duplicated menu structures with explicit order, menu, and location contexts, which became the schema and architectural basis for the TypeScript rewrite.

EXECUTION & SCOPE

- Built a dbt migration pipeline to restructure, clean, and migrate legacy operational data.
- Implemented Evidence reports to make business patterns visible before the rewrite.
- Designed the canonical domain model for orders, menus, locations, and store operations.
- Removed duplicated location-menu handling by making menu and store concepts explicit.
- Delivered a modern TypeScript application with storefront and extensive back-office workflows.

TECHNICAL ENVIRONMENT

Legacy PHP 5.6, outgrown relational schema, dbt, Evidence BI, Domain-Driven Design, TanStack, TypeScript, Playwright.

PHP 5.6 · dbt · Evidence · DDD · TanStack · TypeScript · Playwright · TDD

Outcome & Handover: Migrated and cleaned the legacy data, used BI insight to derive the new domain model, and brought a modern storefront and back-office platform to production. The system supports menu, order, location, and store management workflows across the delivery operation.

METHODOLOGY

The Architecture of Execution

I do not rely on generic process language or hope as a strategy. I use rigorous engineering methods to keep business intent connected to domain models, implementation choices, and production diagnostics.

GOVERNED DOMAIN-DRIVEN DESIGN

Teams often treat code as truth while the domain model goes stale. I establish a working closed loop from model to implementation to diagnostic feedback, so architectural drift becomes visible early.

BEHAVIORAL CONSTRAINT SURFACES

Functional and non-functional requirements often fracture across tools. I model requirements as behavioral constraints, creating a clear path from business intent to diagnostics, delivery checks, and CI gates.

PREREQUISITES

Rules of Engagement

I only take on a few major engagements per year to ensure deep focus. Delivery ownership only works when the mandate, authority, and operating model are clear. Before accepting an engagement, I require alignment on these baselines.

01

EXECUTIVE AUTHORITY

You have decision-making authority or direct, unfiltered access to it. I need a mandate that allows technical and organizational blockers to be resolved quickly.

02

ARCHITECTURAL INVESTMENT

You are ready to invest in resilient architecture, not only short-term patches. The goal is to reduce execution risk while preserving delivery momentum.

03

ASYNC-FIRST ALIGNMENT

You value deep work. I build operating rhythms where written decisions and asynchronous communication reduce unnecessary synchronization meetings.

ECOSYSTEM & R&D

Beyond Consultancy

I am actively researching and building the future of software delivery and AI collaboration. Clients benefit from this continuous R&D, ensuring their architectures are future-proofed by default.

API.LAND

Curated infrastructure and building blocks for modern engineering teams. Providing the modules and operational patterns needed to build faster and safer.

INTELLIGENCE.SPACE

A research lab re-imagining human-AI collaboration. Building an "Intelligent Space" for the entire lifecycle of systems engineering and architecture.