HASSAN AKKARI

SOFTWARE ENGINEER · FULL-STACK
.NET / ASP.NET CORE

KONTAKT

+39 351 787 2307
hassan.akkari01@gmail.com
itshassan.it
in/hassan-akkari
github.com/hassan-akkari
Italienischer Staatsbürger (EU)

STACK

Backend & Daten
ASP.NET Core 10 · C# · MediatR · Dapper · SQL Server

Frontend
React 19 · TypeScript · Redux Toolkit · RTK Query · Next.js · Zod

KI-gestützte Entwicklung
Claude Code · Custom Subagents · Tiered CLAUDE.md · agentisches adversariales Review

Testing & Tooling
Vitest · Playwright (E2E + A11y) · Azure DevOps · CI/CD (YAML) · Figma

SPRACHEN

Englisch — C1 / fliessend
Italienisch — Muttersprache
Arabisch — Konversationsniveau

AUSBILDUNG

Level 3 Diploma · Computing
HCUC, Vereinigtes Königreich · 2021

PROFIL

Full-Stack Software Engineer mit ~2,5 Jahren Erfahrung im Aufbau von Produktionssystemen auf ASP.NET Core und React für ein Hospitality-Softwareunternehmen. Ich habe einen Exactly-once-Nexi-Zahlungsfluss über ein nicht-idempotentes Downstream-System entworfen (MediatR-State-Machine + Idempotenz-Lock), leite die Migration einer mandantenfähigen SaaS von Legacy-ASP.NET MVC/Razor zu React 19 hinter einem BFF und stelle die mandantenfähige Datenisolation über die gesamte API sicher. Zudem habe ich den KI-gestützten Review-Workflow des Teams auf Claude Code aufgebaut. Kernstack: ASP.NET Core 10, C#, MediatR, SQL Server, React 19, TypeScript.

BERUFSERFAHRUNG

Software Engineer
SEP. 2023 — HEUTE
Sibylla S.r.l. — Rom, Italien

Network — Hospitality-E-Commerce
MÄRZ 2025 — HEUTE
ASP.NET Core 10 · C# · MediatR · Dapper · SQL Server · React 19

- Zahlungsbestätigung für Online-Angebote *exactly-once* über ein nicht-idempotentes Downstream-Buchungssystem umgesetzt: ein lokaler Payment-Lock mit eindeutigem Transaktionscode als Schlüssel und eine über MediatR abgewickelte State Machine (`pending → nexi_paid → confirmed | failed`), die erneut übertragene Gateway-Callbacks ignoriert — keine Doppelbuchungen, keine verlorenen Bestätigungen, mit einem Schutz gegen stille „HTTP 200, aber fehlgeschlagen"-Antworten, sodass keine Zahlung ohne reale Buchung bestätigt wird.
- Hauptcontributor und Top-Autor nach Commit-Volumen auf der Hospitality-E-Commerce-Plattform, die intern neu aufgebaut wurde, nachdem eine vorherige externe Entwicklung die Anforderungen verfehlt hatte. Alleiniger Autor des Online-Angebots-Zahlungsflusses, Begründer des React-Self-Service-Check-in-Kiosks und dominierender Autor der Marketplace-SPA — Buchung, Warenkorb, Checkout, Wallet und Konto — von null bis zur Produktion.
- Das React-19-Frontend als drei unabhängige SPAs (Marketplace, Kiosk, Angebots-Landingpage) auf Redux Toolkit + RTK Query entworfen: ein *Single-Flight*-401-Refresh-Auth-Layer, der parallele Token-Refreshes zusammenführt und XSRF transparent wiederholt, eine Ports-and-Adapters-Zahlungsarchitektur (Nexi / Wallet / Rabatt hinter einem einzigen Provider-Interface) sowie Formulare auf react-hook-form + Zod über inkonsistenten Legacy-Payloads. Getestet mit Vitest und Playwright (E2E + A11y); Koordination des Release-Trains Testing → Staging → Produktion auf Azure DevOps.

Portal — mandantenfähige Hospitality-SaaS
SEP. 2023 — HEUTE
ASP.NET Core 10 · BFF · multi-tenant · React 19

- Leitung der *seitenweisen* Migration einer mandantenfähigen Hospitality-Management-SaaS von Legacy-ASP.NET MVC / Razor / jQuery zu React 19, eingebunden in die bestehende Anwendung hinter einem Backend-for-Frontend; ein Routen-Manifest entscheidet pro Seite, ob SPA-Navigation oder Full-Reload erfolgt, sodass Seiten inkrementell migrieren — ohne Big-Bang-Rewrite. Mehrere ursprünglich in jQuery ausgelieferte Features in React neu gebaut und die Deploy-Pipeline gegen veraltete Frontend-Bundles abgesichert.
- Mandantenfähige Datenisolation auditiert und gehärtet (Tenant-Claim auf Zeilenebene gefiltert, nicht über einen Parent-Join); ein Security- und Tenancy-Audit geleitet, das u. a. mandantenübergreifende Zugriffe (IDOR) und ein ausgestelltes, aber nie server-seitig validiertes CSRF-Token aufdeckte.

KI-GESTÜTZTES ENGINEERING

- Das Framework entworfen, das die KI-gestützte Migration des Teams zuverlässig macht: Instruktionsdateien pro Layer (jeder Agent lädt nur die Konventionen seiner Schicht), Spezialisten-Subagents mit eng begrenztem Tool-Zugriff zur Isolation des Blast-Radius, ein Orchestrierungs-Skill, der Aufgaben klassifiziert und die passenden Invarianten injiziert, sowie ein *Plan-first*-Prozess, in dem der erste Schritt immer Analyse ohne Code ist und ein Mensch freigibt, bevor etwas verändert wird — eine Seitenmigration wird zu einer festen, wiederholbaren Pipeline.
- Read-only, projektspezifisch abgestimmte Review-Subagents gebaut (adversariales Architektur-Review, SOLID-Audit, TS↔C#-Contract-Drift, Payment-Flow-Tracing) über der ASP.NET-Core-/React-Codebasis, gestützt auf eine mehrstufige CLAUDE.md-Wissensbasis mit Domänen-Glossar, Payment-State-Machine und nicht offensichtlichen Fallstricken — 3 latente Payment-Flow-Bugs vor dem Release aufgedeckt.

Frontend Developer · Praktikum
OKT. 2022 — MAI 2023
BetterTogether S.r.l. — Rom, Italien

- Frontend für automatisierte Rechnungsstellung (B2C / B2B2C) mit HTML, CSS, JS und C# / ASP.NET MVC; Aruba-APIs für elektronische Rechnungsstellung integriert; UI standardisiert und Legacy-Code refaktoriert.

PROJEKTE

- **laboratoire** — persönliches React-19-/TypeScript-Monorepo (pnpm + Turbo). Server-seitigen Kern eines Next.js-App-Router-Buchungsflusses gebaut: idempotente Auftragsverarbeitung und ein Checkout, der Preise server-seitig revalidiert, statt dem Client zu vertrauen, dazu ein Cal.com-Webhook mit timing-sicherer HMAC-Verifikation — ein produktionsbrechender *Bare-Hex*-Signatur-Bug wurde anhand des Quellcodes des Senders identifiziert und behoben. Dazu: iron-session-Admin-Auth, CORS-/Origin-Allowlists, ein Neon-/Drizzle-Postgres-Schema und ein secretless, cache-aware CI-Gate.