openDEVS

WHITEPAPER

AI, EMPATHIE UND SYSTEMS THINKING

DAS NEUE KOMPETENZPROFIL FÜR SENIOR DEVS

EINLEITUNG

Das neue Kompetenz-profil für Senior Devs

AI verändert gerade nicht nur, wie Software entsteht. Sie verändert, was Seniorität in der Softwareentwicklung bedeutet.

Code zu schreiben war lange die zentrale Engpasskompetenz. Wer komplexe Anforderungen in stabile Systeme übersetzen konnte, war für Unternehmen schwer ersetzbar. Doch diese Grenze verschiebt sich. Produktmanager erstellen Prototypen, Designer bauen funktionale Interfaces, Sales Teams testen Ideen mit generiertem Code. Large Language Models liefern in Sekunden Lösungen, für die früher ganze Entwicklungszyklen nötig waren.

Damit wird Implementierung nicht wertlos. Sie wird aber weniger einzigartig. Für Senior Devs entsteht dadurch eine neue Kernfrage: Wenn AI schneller Code erzeugt als je zuvor, worin liegt dann dein eigentlicher Wert?

Die Antwort liegt nicht in noch mehr Output. Sie liegt in „besserem Denken".

AI kann Code generieren, Muster erkennen und Vorschläge liefern. Sie versteht jedoch keine Verantwortung. Sie kennt nicht automatisch den wirtschaftlichen Kontext eines Produkts, die Dynamiken in einem Team, regulatorische Risiken oder die langfristigen Folgen technischer Entscheidungen. Sie optimiert häufig lokal, während erfahrene Entwickler:innen systemisch denken müssen. Seniorität definiert sich künftig weniger über reine Implementierung und stärker über Fähigkeiten wie:

- AI sinnvoll orchestrieren
- Qualität und Review-Prozesse absichern
- technische Entscheidungen mit Business-Zielen verbinden
- Risiken früh erkennen
- komplexe Systeme ganzheitlich denken
- Teams Orientierung geben
- die Kundschaft und ihre eigentlichen Probleme verstehen

Wer künftig relevant bleiben will, positioniert sich nicht mehr allein über Frameworks, Programmiersprachen oder Tooling. Entscheidend wird, ob du technische Entscheidungen mit Business Wirkung, Systemverständnis und Empathie verbinden kannst. Gerade für Freelancer ist diese klare Positionierung ein zentraler Erfolgsfaktor, weil Unternehmen gezielt nach Expertise suchen. **Es braucht Devs, die konkrete Probleme lösen und nicht nur Aufgaben abarbeiten.**

Dieses Whitepaper zeigt dir, warum AI Orchestrierung, Systems Thinking und Empathie zu den wichtigsten Fähigkeiten moderner Senior Developer:innen werden. Es geht darum, wie du vom Feature Builder zum System Builder wirst, wie du AI produktiv einsetzt, ohne dein eigenes Denken auszulagern, und wie du ein Profil entwickelst, das in einem zunehmend automatisierten Markt langfristig relevant bleibt.

CHAPTER 01

Produktivität ist nicht mehr der Engpass

LLMs verändern die Softwareentwicklung grundlegend. Sie schreiben Code, erklären Frameworks, generieren Tests, schlagen Architekturen vor und beschleunigen Routineaufgaben. Was früher Stunden dauerte, entsteht heute in Minuten. Dadurch verschiebt sich der eigentliche Engpass in Softwareprojekten. Nicht mehr die reine Umsetzung entscheidet über Erfolg, sondern die Fähigkeit, Output richtig einzuordnen, Risiken zu bewerten und Verantwortung für das Gesamtsystem zu übernehmen.

AI erzeugt Geschwindigkeit. Sie erzeugt aber nicht automatisch Qualität.

Gerade in komplexen Projekten kann mehr Output sogar zum Problem werden. Wenn schneller mehr Code entsteht, können auch mehr Fehler, mehr technische Schulden und mehr Sicherheitsrisiken entstehen. Ein Feature wirkt im ersten Moment produktiv, kann aber langfristig Wartungskosten erhöhen, Architekturentscheidungen erschweren oder regulatorische Risiken auslösen. Deshalb reicht es nicht, AI einfach als Beschleuniger einzusetzen. Senior Devs müssen AI in kontrollierte Engineering Prozesse einbetten.

AI ORCHESTRIERUNG BEGINNT VOR DEM ERSTEN PROMPT

AI Orchestrierung bedeutet nicht, möglichst viele Tools zu nutzen. Es bedeutet, bewusst zu entscheiden, welche Aufgaben AI übernehmen darf, welche Aufgaben menschliche Bewertung brauchen und welche Qualitätskriterien erfüllt sein müssen. Ein:e Senior Dev lässt AI nicht einfach Code schreiben. Er oder sie definiert zuerst Kontext, Constraints und Akzeptanzkriterien. Dazu gehören fachliche Anforderungen, technische Rahmenbedingungen, Architekturprinzipien, Security Vorgaben, Datenschutzanforderungen und Erwartungen an Wartbarkeit. Erst wenn diese Leitplanken klar sind, kann AI sinnvoll eingesetzt werden.

Der Unterschied ist entscheidend. Wer AI ohne Kontext nutzt, bekommt schnellen Output. Wer AI mit klarem Rahmen nutzt, bekommt verwertbare Vorschläge. Die eigentliche Senior Leistung liegt deshalb nicht im Prompt allein, sondern im Denken vor dem Prompt:

- Was soll AI lösen?
- Welche Informationen braucht sie dafür?
- Welche Entscheidungen darf sie nicht treffen?
- Welche Risiken müssen anschließend geprüft werden?

Es geht um technische Führung: Senior Devs gestalten den Prozess, in dem AI produktiv wird, statt sich vom Output treiben zu lassen.

QUALITÄTSGATES MACHEN AI OUTPUT BELASTBAR

Von AI generierter Code darf nicht als fertige Lösung betrachtet werden. Er ist ein Vorschlag, der geprüft, eingeordnet und häufig angepasst werden muss.

Deshalb brauchen Teams klare Qualitätsgates. Ein Qualitätsgate beantwortet die Frage: Wann ist AI Output gut genug, um in das System übernommen zu werden? Dabei geht es nicht nur darum, ob der Code kompiliert oder Tests besteht. Entscheidend ist, ob die Lösung zum bestehenden System passt.

Gerade Senior Devs müssen diese Prüfkriterien definieren und konsequent anwenden. Code Reviews werden dadurch nicht weniger wichtig, sondern wichtiger. Denn AI kann in kurzer Zeit sehr überzeugend wirkende Lösungen erzeugen, die bei genauer Betrachtung falsche Annahmen treffen, Randfälle ignorieren oder unnötige Komplexität einführen. Ein guter Review Prozess prüft daher nicht nur Syntax und Funktionalität. Er bewertet auch Architektur Fit, Datenflüsse, Fehlertoleranz, Observability, Testbarkeit und langfristige Änderbarkeit. Erst wenn diese Kriterien erfüllt sind, wird aus generiertem Code ein belastbarer Beitrag zum System.

AGENTS BRAUCHEN STEUERUNG, NICHT NUR AUFGABEN

Mit AI Agents verändert sich die Rolle von Senior Devs weiter. Agents können Teilaufgaben übernehmen, etwa Tests generieren, Dokumentation vorbereiten, Refactoring Vorschläge machen, Migrationspfade analysieren oder Pull Requests vorstrukturieren. Das kann enorme Produktivitätsgewinne bringen, wenn die Steuerung stimmt.

Doch Agents sind keine autonomen Verantwortlichen. Sie arbeiten innerhalb eines Rahmens, der von Menschen gestaltet werden muss. Senior Devs definieren, welche Rolle ein Agent übernimmt, welche Inputs er bekommt, welche Ergebnisse erwartet werden und wo menschliche Kontrolle zwingend bleibt.

Ein Agent, der Tests generiert, braucht andere Bewertungskriterien als ein Agent, der Security Risiken analysiert. Ein Agent für Dokumentation muss fachliche Verständlichkeit liefern. Ein Agent für Refactoring muss zeigen, dass bestehendes Verhalten erhalten bleibt. Ohne klare Rollen, Grenzen und Review Mechanismen entsteht nur scheinbare Effizienz.

AI Orchestrierung ist deshalb kein reines Toolthema. Sie ist ein Engineering Prozess. Es geht darum, Geschwindigkeit mit Kontrolle zu verbinden. Unternehmen brauchen dafür Expert:innen, die AI nicht nur bedienen, sondern in robuste Entwicklungsprozesse übersetzen können. Für Senior Devs entsteht daraus eine starke Positionierung. Ihr Wert liegt nicht darin, mit AI einfach mehr Code zu produzieren. Ihr Wert liegt darin, AI so einzusetzen, dass bessere Entscheidungen, stabilere Systeme und messbarer Business Nutzen entstehen. Weniger reine Tastaturarbeit, mehr Steuerung, mehr Qualitätssicherung, mehr Verantwortung. Produktivität ist damit nicht verschwunden. Sie ist nur nicht mehr der entscheidende Engpass.

Qualitätsgate Checkliste für AI-generierten Code

- ☐ *Passt der Code zur Architektur?*
- ☐ *Sind Security-Anforderungen erfüllt?*
- ☐ *Ist die Lösung wartbar?*
- ☐ *Gibt es neue problematische Abhängigkeiten?*
- ☐ *Sind Edge Cases berücksichtigt?*
- ☐ *Ist die Lösung testbar?*
- ☐ *Sind Datenschutz und Compliance geprüft?*

CHAPTER 02

Vom Feature Builder zum System Builder

In der Softwareentwicklung verschiebt sich der Wertbeitrag erfahrener Developer deutlich. Lange galt: Wer Anforderungen schnell, sauber und zuverlässig in Features übersetzen kann, ist besonders wertvoll. Diese Fähigkeit bleibt wichtig, reicht aber in modernen Unternehmensumgebungen nicht mehr aus. Gerade Senior Devs müssen heute verstehen, welche Wirkung ihre technischen Entscheidungen auf Geschäftsmodell, Kundennutzen und Wirtschaftlichkeit haben.

Der Unterschied zwischen einem Feature Builder und einem System Builder liegt nicht in der Programmiersprache, sondern in der Perspektive. Ein Feature Builder fragt: „Wie setze ich diese Anforderung technisch um?“ Ein System Builder fragt zusätzlich: „Warum ist diese Anforderung wichtig, welchen geschäftlichen Zweck erfüllt sie und welche Folgen hat meine Lösung für das gesamte System?“ Diese zweite Perspektive macht Senior Developer zu strategischen Partnern für Unternehmen.

GESCHÄFTSMODELLE VERSTEHEN

Technische Arbeit findet nie im luftleeren Raum statt. Jede Software unterstützt ein Geschäftsmodell, ob direkt oder indirekt. Sie ermöglicht Umsatz, senkt Kosten, verbessert Prozesse, reduziert Risiken oder schafft bessere Kundenerlebnisse. Senior Devs müssen deshalb verstehen, wie das Unternehmen Geld verdient und welche Rolle das digitale Produkt dabei spielt. Ein Payment Service in einem E-Commerce Unternehmen ist nicht nur ein technisches Modul. Er beeinflusst Conversion, Abbruchraten, Kundenzufriedenheit und Umsatz. Eine interne Plattform ist nicht nur Infrastruktur. Sie kann Entwicklungszeiten verkürzen, Betriebskosten senken und Teams schneller handlungsfähig machen. Ein Datenprodukt ist nicht nur eine Pipeline. Es kann bessere Entscheidungen ermöglichen und Wettbewerbsvorteile schaffen. Wer diese Zusammenhänge erkennt, priorisiert anders. **Dann geht es nicht mehr nur darum, Tickets abzuarbeiten oder technische Eleganz zu erreichen.** Es geht darum, technische Entscheidungen an geschäftlichen Zielen auszurichten.

KUNDENNUTZEN ERKENNEN

Business Verständnis bedeutet auch, den Nutzen für Kund:innen zu verstehen. Viele technische Diskussionen bleiben auf der Ebene von Architektur, Frameworks oder Deployment Prozessen stehen. Diese Themen sind wichtig, aber sie beantworten nicht automatisch die zentrale Frage: Welches Problem lösen wir für die Menschen, die das Produkt nutzen? Senior Devs sollten Anforderungen deshalb nicht nur technisch prüfen, sondern auch aus Nutzerperspektive hinterfragen:

- Verbessert dieses Feature wirklich den Alltag der Kundschaft?
- Reduziert es Reibung?
- Erhöht es Vertrauen?
- Spart es Zeit?
- Macht es einen Prozess verständlicher, schneller oder sicherer?

Diese Denkweise schützt Teams vor Scheinproduktivität. Ein Feature kann technisch korrekt umgesetzt sein und trotzdem wenig Nutzen stiften. Umgekehrt kann eine kleine technische Anpassung enorme Wirkung entfalten, wenn sie ein zentrales Kundenproblem löst. Senior Devs mit Systemblick erkennen solche Hebel früher und bringen sie aktiv in die Produktentwicklung ein.

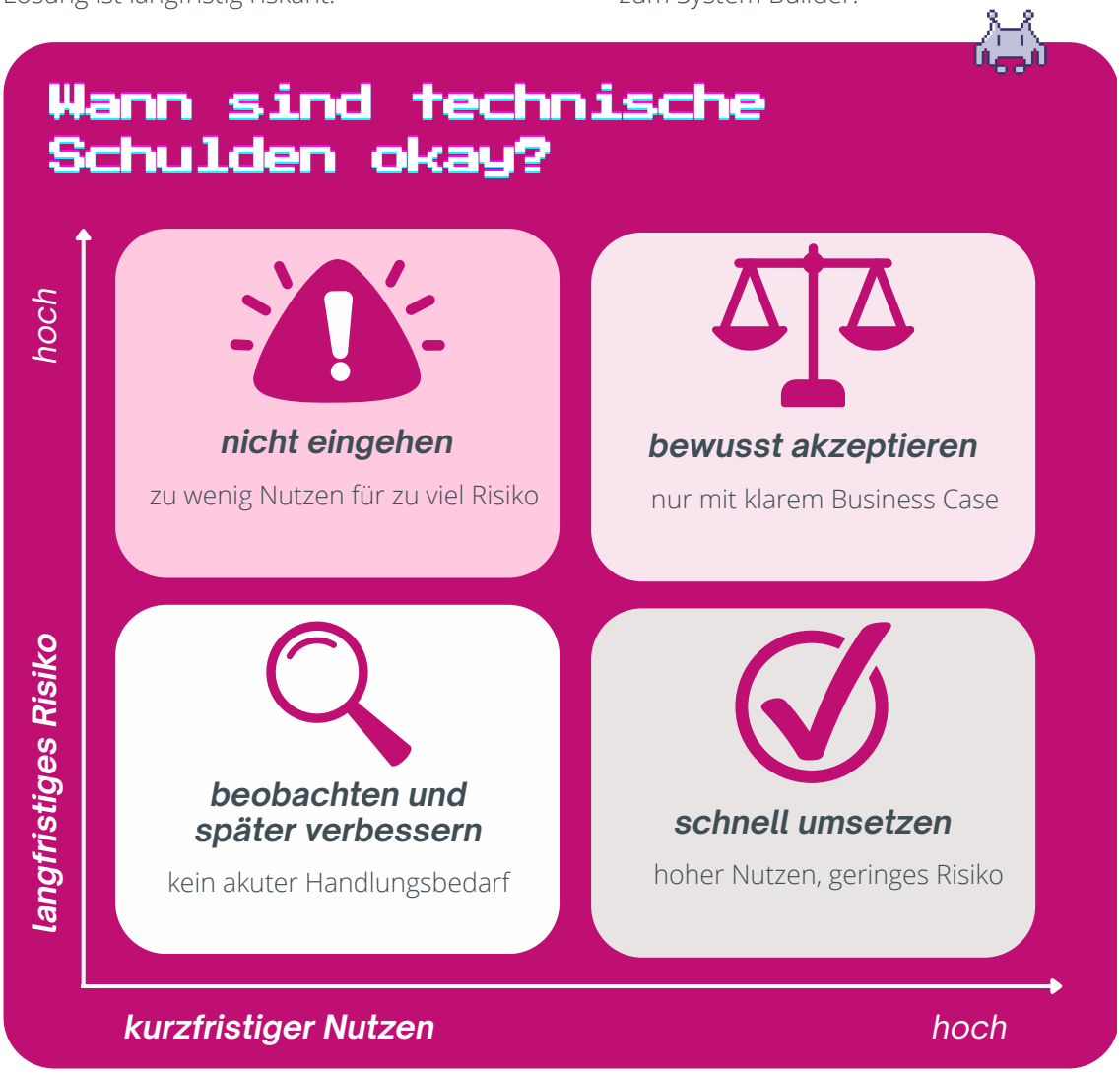
TECHNISCHE ENTSCHEIDUNGEN BETRIEBSWIRTSCHAFTLICH EINORDNEN

Jede technische Entscheidung hat Kosten. Dazu zählen Entwicklungszeit, Wartung, Betrieb, Security, Know how Aufbau, Abhängigkeiten und spätere Änderbarkeit. Senior Devs müssen diese Faktoren sichtbar machen und verständlich einordnen. Nicht jede moderne Technologie ist automatisch sinnvoll. Nicht jedes Refactoring ist wirtschaftlich gerechtfertigt. Nicht jede kurzfristige Lösung ist langfristig riskant.

Die eigentliche Stärke liegt im Abwägen. Wann lohnt sich eine robuste Architektur? Wann ist Geschwindigkeit wichtiger? Wann erzeugt technische Schuld ein vertretbares Risiko? Wann wird sie zum echten Geschäftsproblem? Diese Fragen verlangen mehr als Coding Skills. Sie verlangen Urteilsvermögen, Erfahrung und ein Verständnis für wirtschaftliche Konsequenzen.

Für Unternehmen werden solche Senior Devs immer wertvoller. Sie liefern nicht nur Code, sondern Orientierung.

Sie helfen Entscheider:innen dabei, technische Investitionen besser zu bewerten. Sie übersetzen zwischen Engineering, Produkt, Management und Business.
Die entscheidende Frage lautet deshalb: Verstehst du, wie das Unternehmen Geld verdient, oder nur, wie dein Service deployed wird? Wer diese Frage klar beantworten kann, entwickelt sich vom reinen Feature Builder zum System Builder.



CHAPTER 03

Systems Thinking: Der Blick fürs große Ganze

In komplexen Softwareprojekten entscheidet nicht nur die Qualität einzelner Codezeilen über den Erfolg. Entscheidend ist, ob du erkennst, wie sich technische Entscheidungen auf das gesamte System auswirken. Für Senior Devs wird diese Fähigkeit zur eigentlichen Kernkompetenz, weil moderne Software selten isoliert funktioniert. Sie ist Teil von Geschäftsprozessen, Datenflüssen, User Erwartungen, Compliance Vorgaben und langfristigen Unternehmenszielen.

WAS SYSTEMS THINKING WIRKLICH BEDEUTET

Systems Thinking heißt, technische Aufgaben nicht als Einzelproblem zu betrachten, sondern als Teil eines größeren Zusammenhangs. **Es geht darum, Muster, Wechselwirkungen und langfristige Folgen zu erkennen.** Eine API ist dann nicht nur ein Endpunkt, der Daten liefert. Sie ist ein Verbindungselement zwischen Teams, Produkten, Kundschaft, Infrastruktur und Business Logik. Senior Devs mit Systemblick fragen deshalb nicht nur: „Wie implementiere ich diese API?“ Sie fragen auch:

- Welche Prozesse hängen daran?
- Welche Datenqualität brauchen wir?
- Welche Teams nutzen diese Schnittstelle?
- Welche Risiken entstehen, wenn sie ausfällt oder falsch verwendet wird?

Diese Perspektive verändert die Qualität technischer Entscheidungen grundlegend.

EDGE CASES FRÜH ERKENNEN

Ein wesentliches Merkmal erfahrener Devs ist, dass sie Grenzfälle früh sehen. Während Junior Devs oft zuerst den Idealweg bauen, denken Senior Devs weiter: Sie rechnen mit Ausnahmen, Lastspitzen, fehlerhaften Eingaben, unerwartetem Nutzerverhalten und externen Abhängigkeiten.

Edge Cases sind daher mehr als technische Sonderfälle. Sie zeigen, ob ein System auch dann stabil bleibt, wenn etwas nicht nach Plan läuft. Ein Drittanbieter fällt aus. Daten kommen unvollständig an. Eine Verarbeitung verzögert sich. Solche Situationen wirken sich schnell auf die Kundschaft oder interne Abläufe aus. Wer sie früh mitdenkt, verhindert teure Fehler in Produktion.

STRUKTURELLE SCHWÄCHEN UND ABHÄNGIGKEITEN ANALYSIEREN

Systems Thinking bedeutet auch, strukturelle Schwächen sichtbar zu machen. Dazu gehören eng gekoppelte Services, unklare Verantwortlichkeiten, fehlende Observability, manuelle Workarounds oder technische Schulden, die bei Wachstum zum Problem werden.

Abhängigkeiten sind dabei besonders kritisch. Ein System kann auf den ersten Blick stabil wirken, aber in der Tiefe von Datenquellen, Legacy Komponenten, Deployment Prozessen oder einzelnen Expert:innen abhängen. Senior Devs erkennen diese Risiken und machen sie besprechbar. Sie schaffen Transparenz, bevor aus versteckten Abhängigkeiten echte Blocker entstehen.

ÜBER DAS OFFENSICHTLICHE HINAUSDENKEN

Die stärkste Form von Systems Thinking zeigt sich in Zukunftsfragen:

- Was passiert, wenn das System zehnfach wächst?
- Was verändert sich, wenn das Business Modell angepasst wird?
- Welche Architektur hält stand, wenn neue Märkte, neue Produkte oder regulatorische Anforderungen hinzukommen?

Es geht nicht darum, jedes mögliche Szenario perfekt vorherzusagen. Es geht darum, technische Lösungen so zu gestalten, dass sie anpassungsfähig bleiben. Senior Devs bauen nicht nur für den aktuellen Sprint. Sie denken an Skalierung, Wartbarkeit, Sicherheit und Veränderbarkeit.

TEAM UND KOMMUNIKATION

Systems Thinking zeigt sich außerdem darin, wie Senior Devs mit anderen sprechen. Sie erklären technische Folgen so, dass Product, Management oder Fachbereiche sie verstehen. Dadurch entscheiden Teams besser, weil Risiken, Kompromisse und mögliche Auswirkungen nicht im Code versteckt bleiben. Eine technische Lösung funktioniert nämlich erst dann wirklich gut, wenn alle Beteiligten verstehen, warum sie sinnvoll ist.

Senior Devs wägen außerdem ab, wann ein schneller Weg reicht und wann ein System mehr Stabilität braucht. Manchmal muss ein Team schnell reagieren, weil ein Release ansteht oder ein Problem dringend gelöst werden muss. Erfahrene Developer erkennen solche Situationen.

Sie nutzen Workarounds aber nicht blind, sondern machen klar, was später nachgebessert werden sollte.

Sie halten technische Schulden fest, sprechen Folgeaufgaben an und sorgen dafür, dass das Team die Entscheidung später wieder sauber einordnet.

CHAPTER 04

Empathie: unterschätzter High Impact Skill

In technischen Rollen wird Empathie oft unterschätzt. Viele verbinden sie mit Freundlichkeit, Kommunikation oder Teamkultur. Für Senior Devs ist Empathie jedoch weit mehr als ein Soft Skill. Sie ist eine strategische Fähigkeit, die dabei hilft, bessere Entscheidungen zu treffen, echte Probleme zu verstehen und tragfähige Lösungen zu entwickeln.

Gerade in komplexen Software Projekten entsteht Wirkung nicht allein durch sauberen Code. Wirkung entsteht, wenn technische Lösungen zu den tatsächlichen Bedürfnissen der Kundschaft, den Zielen des Unternehmens und der Realität im Team passen. Dafür müssen Senior Devs zuhören, einordnen und zwischen den Zeilen lesen können.

PROBLEME DER KUNDSCHAFT WIRKLICH VERSTEHEN

Empathie beginnt mit der Fähigkeit, Kundenprobleme nicht vorschnell technisch zu übersetzen. Oft kommt eine Anforderung als Feature Wunsch ins Team: ein neues Dashboard, eine zusätzliche Schnittstelle, eine Automatisierung oder eine Anpassung im Backend. Senior Devs mit Empathie fragen zuerst, welches Problem dahintersteht.

Geht es wirklich um ein neues Feature? Oder geht es um fehlende Transparenz, manuelle Arbeit, Unsicherheit in einem Prozess oder steigenden Druck durch die Kundschaft? Diese Unterscheidung ist entscheidend. Wer nur die sichtbare Anforderung umsetzt, behandelt manchmal nur ein Symptom. Wer das eigentliche Problem versteht, kann bessere technische Optionen entwickeln. Empathie bedeutet deshalb nicht, allem zuzustimmen. Sie bedeutet, die Perspektive der Kundschaft ernst zu nehmen und gleichzeitig professionell zu hinterfragen. So entstehen Lösungen, die nicht nur technisch funktionieren, sondern im Alltag Nutzen stiften.

PERSÖNLICHE ZIELE HINTER BUSINESS PROBLEMEN ERKENNEN

Business Probleme sind selten rein sachlich. Hinter ihnen stehen Menschen mit Zielen, Erwartungen, Druck und Verantwortung. Eine Entscheiderin möchte Risiken reduzieren. Ein Product Owner möchte schneller liefern.

Ein Team Lead möchte Überlastung vermeiden. Eine Fachabteilung möchte endlich einen Prozess vereinfachen, der seit Jahren Zeit kostet.

Senior Devs, die diese persönlichen Motive erkennen, kommunizieren anders. Sie erklären technische Entscheidungen nicht nur auf Architektur Ebene, sondern verbinden sie mit dem, was für die Beteiligten relevant ist.

Diese Fähigkeit macht Zusammenarbeit wirksamer. Sie schafft Vertrauen, weil Auftraggeber:innen merken: Diese Person versteht nicht nur den Code, sondern auch den Kontext.

PSYCHOLOGISCHE SICHERHEIT IM TEAM SCHAFFEN

Empathie wirkt nicht nur nach außen, sondern auch ins Team. Gute Software entsteht dort, wo Menschen Fragen stellen, Zweifel äußern und Fehler früh sichtbar machen können. Dafür braucht es psychologische Sicherheit.

Senior Devs haben hier eine besondere Verantwortung. Sie prägen durch ihr Verhalten, ob ein Team offen denkt oder Probleme versteckt. Wer Reviews respektvoll führt, Unsicherheiten ernst nimmt und Kritik konstruktiv formuliert, verbessert nicht nur die Stimmung. Er oder sie erhöht direkt die Qualität der Arbeit.

Das Forward Deployed Engineer Modell zeigt diesen Zusammenhang besonders deutlich. Wenn Engineers Kundschaft direkt erleben, verstehen sie Anforderungen tiefer, sehen Reibungspunkte früher und entwickeln ein besseres Gefühl für Wirkung. Nähe zur Kundschaft macht technische Entscheidungen nicht weniger professionell. Sie macht sie relevanter.

Empathie ist deshalb kein nettes Extra. Sie ist ein High Impact Skill für Senior Devs, die Verantwortung übernehmen, Systeme verbessern und Lösungen bauen wollen, die wirklich gebraucht werden.

VOM MARKTVERSTÄNDNIS ZUR EIGENEN RICHTUNG

Du hast nun einen Überblick darüber, wie sich der Markt verändert und welche Kompetenzen für Senior Devs immer wichtiger werden.

AI beschleunigt die Umsetzung, Unternehmen suchen stärker nach systemischem Denken, Business Verständnis und Menschen, die technische Entscheidungen in einen größeren Zusammenhang stellen können. Der Markt belohnt nicht nur reine Produktivität, sondern auch Verantwortung und Orientierungshilfe.

Als Senior Freelance Dev bist du nicht nur Nutzer:in neuer Technologien. Du bist Architekt:in deines Profils. Du entscheidest, welche Probleme du lösen willst, welche Auftraggeber:innen zu dir passen und welche Rolle du im Projekt einnehmen möchtest.

Deine Positionierung entsteht dort, wo deine Fähigkeiten, deine Erfahrung und dein innerer Antrieb zusammenkommen.

Vielleicht treibt dich technische Exzellenz an. Vielleicht möchtest du komplexe Systeme verständlicher machen. Vielleicht suchst du Projekte, die gesellschaftlich relevant sind oder einen klaren Beitrag für Kund:innen leisten.

Je besser du verstehst, was dich motiviert, desto leichter findest du Projekte, die nicht nur wirtschaftlich sinnvoll sind, sondern auch zu dir passen. Betrachte dieses Whitepaper daher gerne als Startpunkt. Nutze die Impulse, um dein Profil zu schärfen, deine Rolle bewusster zu wählen und deine nächsten Projekte gezielter auszuwählen.

Mehr dazu findest du im weiterführenden Whitepaper „Projekte mit Purpose“

AUF DER SUCHE NACH DEINEM NÄCHSTEN SOFTWAREPROJEKT?

MARIA LAUSCH

COMMUNITY MANAGEMENT

maria.lausch@opendevs.net

SCAN ME

BARBORA LICHNEROVA

COMMUNITY MANAGEMENT

barbora.lichnerova@opendevs.net

SCAN ME

HIER FINDEST DU UNS

INSTAGRAM
@open.devs

E-MAIL
ahoi@opendevs.net

WEBSITE
www.opendevs.net

ADRESSE
Mandlgasse 20/3-5, 1120 Wien