KI in der Software-Entwicklung

Die neue Realität für Entwickler

Warum dieses Thema?

KI dominiert aktuell die Software-Branche

Unsicherheit bei vielen Entwicklern und Studierenden

Agenda

Was ist KI?

Terminologie

Praxis Beispiel

Impact auf die Software Entwicklung

Die Rolle des Entwicklers in der Zukunft

Was ist KI?

Grundlagen verstehen

KI ist kein neues Phänomen

Artificial Intelligence existiert seit den 1950er Jahren

Aber: Durchbruch in den letzten Jahren durch:

- Massive Rechenleistung
- Große Datenmengen
- Neue Algorithmen (Deep Learning, Transformer)

Typen von KI

Narrow AI

Spezialisiert auf **eine Aufgabe**

Beispiele:

- Bilderkennung
- Sprachübersetzung
- Schachspielen

Das haben wir heute

General AI

Kann **jede Aufgabe** wie ein Mensch

Noch **nicht erreicht**

Gegenstand von Science Fiction

Terminologie

Wichtige Begriffe

Large Language Models (LLMs)

Was ist das?

Transformer Modelle (Neural Networks), trainiert auf riesigen Textmengen

Fähigkeiten:

- Text generieren auf Basis von Anweisungen (Prompts)
- Code generieren
- Bilder, Audio und Videos erstellen

Beispiele: GPT-4o, Claude Sonnet 4.5, Gemini 2.5 Flash

AI Agents?

Was ist das?

Autonome Systeme, die autonom Aufgaben erledigen

Eigenschaften:

- Versteht Ziele und handelt danach
- Führt autonom Aktionen aus
- Nutzt Werkzeuge
- Lernt eigenständig

Agents vs. Chatbots

Chatbot

💬 Reaktiv

Antwortet auf Anfragen

Braucht klare Anweisungen

Ein Austausch

Agent

🤖 Proaktiv

Arbeitet selbstständig

Plant mehrere Schritte

Nutzt Tools

Iteriert bis zum Ziel

Agent-Beispiel: Code-Review

Aufgabe: "Review meinen Pull Request"

Agent arbeitet autonom:

1. Liest Code-Änderungen
2. Führt statische Analyse durch
3. Sucht nach Sicherheitslücken
4. Prüft Tests
5. Erstellt strukturiertes Feedback
6. Schlägt Verbesserungen vor

Prompting

Was ist das?

- Instruktion, die an die KI gegeben wird

Schlechter Prompt:

Schreib mir eine Funktion

Gutes Prompt:

Schreibe eine TypeScript-Funktion, die ein Array von Zahlen nimmt und die Summe zurückgibt. Nutze funktionale Programmierung. Füge TypeScript-Typen und JSDoc-Kommentar hinzu.

Prompt Engineering Prinzipien

Kontext geben

- Rolle, Situation, Ziel

Spezifisch sein

- Format, Struktur, Constraints

Beispiele zeigen

- Few-shot learning

Iterativ verfeinern

- Feedback-Loop, Chain of Thought

Context Windows

Was ist das?

Maximale "Gedächtnisspanne" eines LLM -> Short Term Memory

Früher: ca. 4.000 Tokens (ca. 3.000 Wörter)
Heute: 200.000+ Tokens

Bedeutung:

- KI kann jetzt gesamte Projekte verstehen
- Längere Sessions mit mehreren Schritten möglich

Model Context Protocol

Was ist das?

- Protocol um LLMs mit externen Daten zu verbinden
- Zugriff auf externe Quellen wie APIs, Dateien, Tools
- Entwickelt von Anthropic

Beispiele

- Github MCP Server
- Context7 MCP Server
- Playwright MCP Server

Der KI Eisberg für Entwickler

Chat

z.b. ChatGPT

- Einfache Anfragen
- Prompting wichtig
- Kontext wichtig

CoPilot Systeme

z.b. Github CoPilot

- Oft integriert in die IDE
- Kontext wird von CoPilot organisiert
- Inline Suggestions
- Guidelines wichtig

Multi Agenten

z.b. opencode

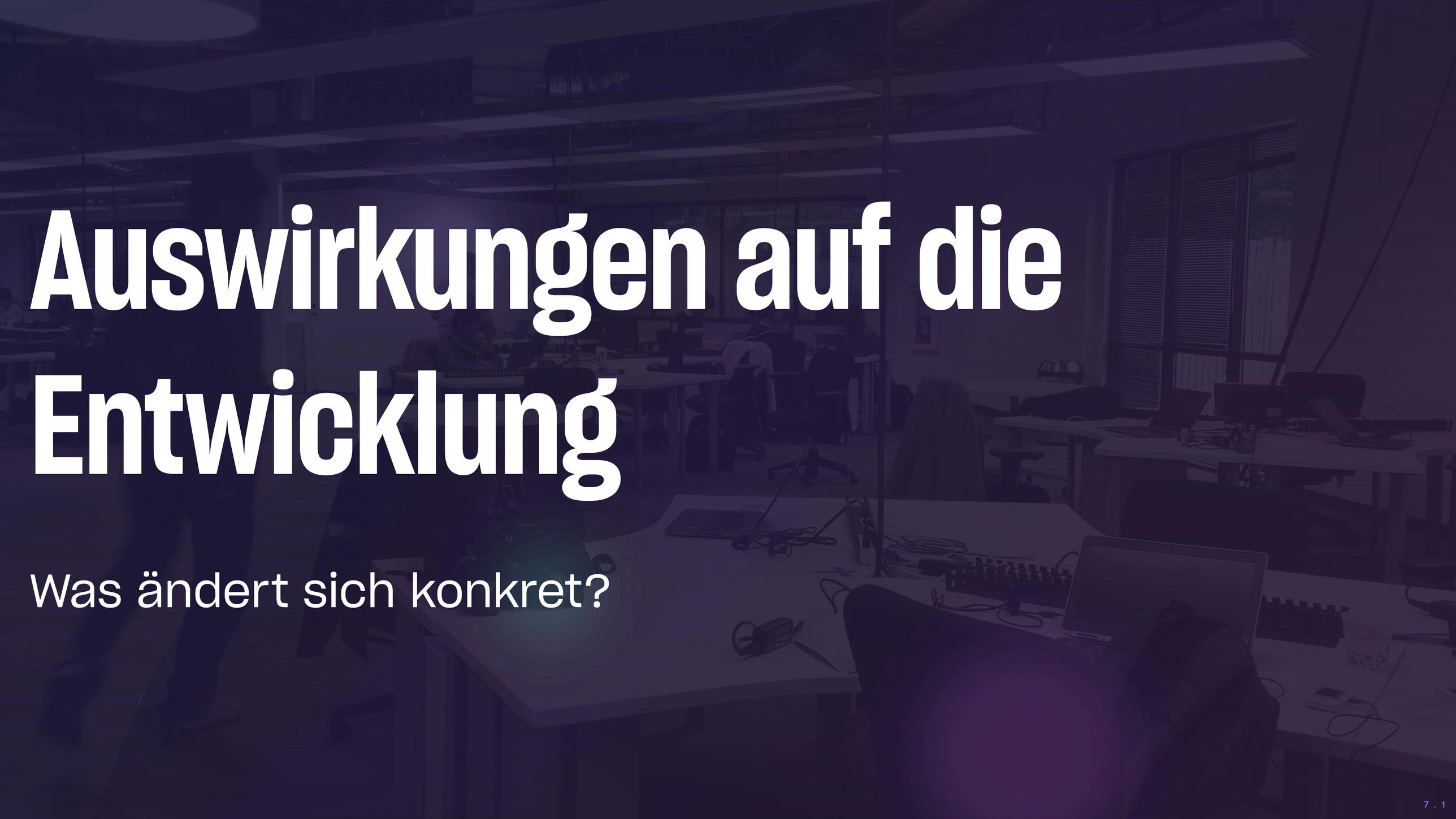
- Oft integriert ins Terminal
- Kontext Recompacting
- Spezialisierte Subagents
- Integriertes Permission System

Autonome Agenten

z.b. Claude Code

- Subagents mit dedizierten Context Windows
- PRDs oder Spec KIT hilfreich
- Oft deployed auf separaten Servern

Praxisbeispiel



Auswirkungen auf die Entwicklung

Was ändert sich konkret?

Was KI bereits kann

- ✅ Boilerplate-Code generieren
- ✅ Tests schreiben
- ✅ Code refactoring
- ✅ Bugs finden und fixen
- ✅ Dokumentation erstellen
- ✅ Code-Reviews
- ✅ Komplexe Features implementieren

Was KI (noch) nicht kann

- ❌ Komplexe Architektur-Entscheidungen
- ❌ Stakeholder-Kommunikation
- ❌ Geschäftslogik verstehen ohne Kontext
- ❌ Kreative Problemlösung bei unklaren Requirements
- ❌ Langfristige strategische Planung

Aber: Die Grenze verschiebt sich **schnell**

Der veränderte Workflow

Früher

Idee → Design → Code schreiben → Testen → Review → Deploy

Jeder Schritt manuell

Langsam, aber kontrolliert

Mit KI

Idee → KI generiert → Review & Anpassen → Testen (automatisch) → Deploy

Schneller, aber:

Neue Skills erforderlich

Qualitätssicherung wichtiger

Produktivitätssteigerung

Studien zeigen:

- 40-50% schnellere Entwicklung
- 60% mehr Aufgaben abgeschlossen
- Besonders für Junior-Entwickler

Quelle: GitHub Copilot Studie 2023, McKinsey Developer Survey 2024

Aber:

Code-Qualität muss überwacht werden

Neue Risiken

Sicherheit:

- Generierter Code kann Vulnerabilities enthalten

Lizenz-Probleme:

- Training auf öffentlichem Code (GPL, etc.)

Halluzinationen:

- KI erfindet manchmal APIs/Funktionen

Over-Reliance:

- Verlernen von Grundlagen

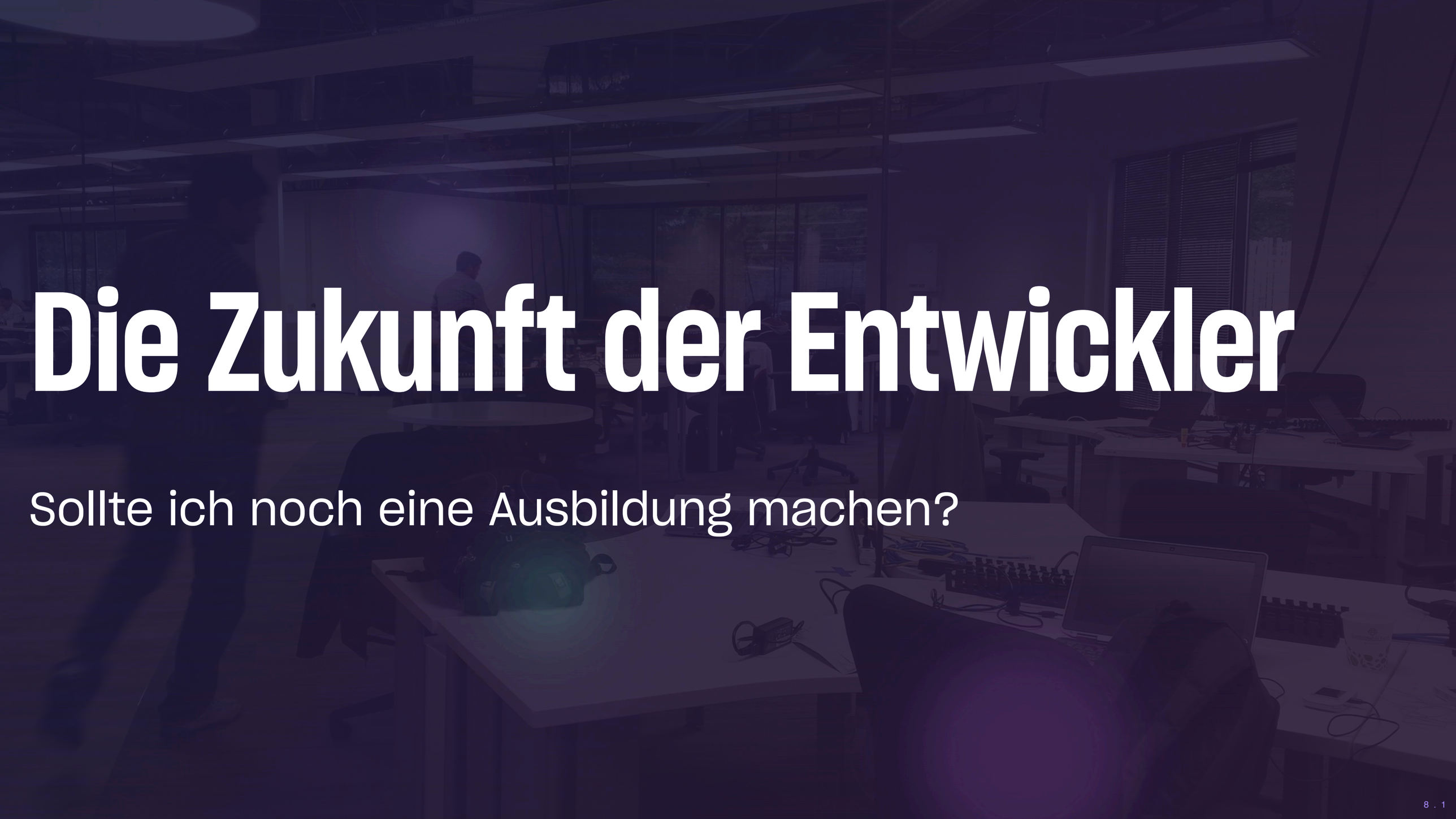
Best Practices

✅ DO:

- KI als Co-Pilot, nicht als Autopilot
- Generierten Code immer reviewen und verstehen
- Tests schreiben (auch KI-generiert)
- Kontext gut beschreiben

❌ DON'T:

- Blind Copy-Paste
- Credentials in Prompts
- Vertrauliche Daten teilen
- Entwickler Identität aufgeben



Die Zukunft der Entwickler

Sollte ich noch eine Ausbildung machen?

Die zentrale Frage

Wird KI Software-Entwickler ersetzen?

Kurze Antwort: Nein (Opinionated)

Lange Antwort: Die Rolle verändert sich fundamental

Was bleibt wichtig?

Problemlösung & kritisches Denken

KI gibt Lösungen → Du entscheidest, ob sie gut sind

Architektur & Design

Systemisches Denken bleibt menschlich

Domänenwissen

Business-Kontext verstehen

Kommunikation

Mit Menschen, Stakeholdern, Teams

Die neue Entwickler-Rolle

Traditionell

Code Writer

Syntaxkenntnisse

Frameworks auswendig

Debugging

Implementation

Mit KI

Solution Architect

Prompt Engineering

Systemdenken

Code Review

Orchestrierung

Quality Assurance

Wird es weniger Jobs geben?

Realistisches Szenario:

Jobs ändern sich, nicht verschwinden

Warum:

- Mehr Features = mehr Nachfrage
- Neue Anwendungsfälle möglich
- KI selbst muss entwickelt werden
- Qualitätssicherung wird wichtiger

Aber: Anforderungen steigen

Sollte ich noch Entwickler werden?

JA!

Aber mit anderen Schwerpunkten:

- Grundlagen intensiver als je zuvor
- Verstehen, warum etwas funktioniert
- Architektur und Design wichtiger
- Kommunikation und Kontext essentiell
- KI als Werkzeug beherrschen

Skills für die KI-Ära

Must-have:

1. **Solide Grundlagen** (Algorithmen, Datenstrukturen)
2. **System Design** (Architektur-Denken)
3. **Prompt Engineering** (Mit KI kommunizieren)
4. **Code Review** (Qualität beurteilen)
5. **Testing** (Absicherung)
6. **Domänenwissen** (Kontext verstehen)

Wie bereite ich mich vor?

Jetzt:

- Nutze KI-Tools regelmäßig
- Lerne Grundlagen **tief**
- Verstehe, was KI gut / schlecht kann
- Experimentiere mit Agents
- Bleib neugierig

Langfristig:

- Spezialisiere dich (AI, Security, DevOps)
- Entwickle Soft Skills
- Verstehe Business-Seite

Der KI-Entwickler 2026+

Ist gleichzeitig:

- Architekt (Design-Entscheidungen)
- Orchestrator (KI-Tools koordinieren)
- Qualitätswächter (Review & Testing)
- Problemlöser (Kreative Lösungen)
- Kommunikator (Menschen & Maschinen)

Nicht mehr: Reiner Code-Schreiber

Open questions

1. Wie blickt ihr in die Zukunft in Bezug auf eure Rolle / Branche?
2. Was sind eure Erfahrung mit KI-Tools bisher?
3. Wo seht ihr Chancen in dem Wandel? Und wie bereitet ihr euch heute schon darauf vor?