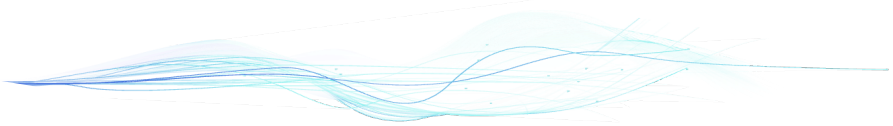
CONTECO

Workshop



Gen AI: LLM-Tools und -Integration für Developer

Gen AI: LLM-Tools und -Integration für Developer

Kursdauer
5 Tage

Zielgruppe
Softwareentwickler:innen, die **keine eigenen LLMs trainieren**, sondern bestehende **APIs, Open-Source-Modelle und Bibliotheken** effizient in ihre Anwendungen integrieren möchten. Der Fokus liegt auf einer **pragmatischen Nutzung** von KI-Tools, strukturiertem **Prompting**, der technischen **Integration** von LLMs in eigene Applikationen sowie auf dem sicheren und datenschutzkonformen Einsatz.

Schulungsziel
Nach Abschluss dieser Schulung sind die Teilnehmenden in der Lage, **bestehende LLMs und KI-Tools gezielt für Softwareprojekte zu nutzen**. Sie verstehen die **Limitierungen und Risiken** von LLMs, können **effektive Prompts** für verschiedene Coding-Szenarien schreiben und lernen **relevante Frameworks** wie LangChain oder Haystack für Retrieval-Augmented Generation (RAG) kennen. Praktische Hands-on-Sessions helfen dabei, LLMs effizient in Anwendungen zu integrieren, **Daten sicher zu verarbeiten** und typische Herausforderungen wie **Kontextgrenzen oder Sicherheitslücken** zu meistern.

Schulungsbeschreibung
In dieser **fünf**-tägigen praxisorientierten Schulung lernen Entwickler:innen, wie sie **bestehende LLMs sinnvoll in eigene Softwareprojekte einbinden**. Statt selbst Modelle zu trainieren, geht es darum, **bestehende APIs und Open-Source-Modelle** geschickt zu nutzen.

Am ersten Tag werden die **Grundlagen von LLMs** sowie ihre **Herausforderungen** behandelt: Datenschutz, Kontextlimits, Bias, Unzuverlässigkeit und typische Fehlermuster. Anschließend erlernen die Teilnehmenden **strukturierte Prompting-Techniken**, um LLMs gezielt für Programmieraufgaben einzusetzen, etwa für **Code-Generierung, Optimierung oder Review**.

Die folgenden Tage fokussieren sich auf die **technische Integration** von LLMs in eigene Anwendungen mit **LangChain, Haystack und RAG-Techniken**. Zudem wird gezeigt, wie sich mit **Gradio und Streamlit** schnell Prototypen von LLM-gestützten Anwendungen entwickeln lassen. Im weiteren Verlauf beschäftigen sich die Teilnehmenden mit **fortgeschrittenem Prompting, Sicherheitskonzepten (Jailbreaking, Prompt Injection) sowie Backup-Strategien für LLMs**.

Den Abschluss bildet ein **praxisnahes Abschlussprojekt**, in dem die Teilnehmenden eine eigene **KI-gestützte Developer-Assistant-App** bauen. Dabei werden erlernte Konzepte wie **RAG, LLM-APIs und Prompting-Frameworks** in einer realistischen Anwendung verknüpft.

Wer teilnehmen sollte
Dieser Kurs richtet sich an **Softwareentwickler:innen**, die **bestehende LLMs und KI-Tools** nutzen möchten, um ihre Anwendungen zu erweitern, effizienter zu arbeiten oder KI-basierte Features in bestehende Systeme zu integrieren. **Voraussetzung sind grundlegende Kenntnisse in Python** sowie Interesse an **APIs, Automatisierung und Machine Learning-Technologien**.

Tag 1: LLM-Grundlagen & Herausforderungen

1. **KI in der Softwareentwicklung**
   - Unzuverlässige LLMs
   - Beschränkter Kontext
   - Nicht auf dem neuesten Stand
   - Kein Zugriff auf Closed-Source-Code
   - Eingeschränkte Sicht/Tests
   - Datenschutz
2. **Wie funktionieren LLMs grundsätzlich?**
   - Kurzer Überblick: Transformer-Modelle, GPT-Architektur (Training vs. Fine-Tuning)
   - Wichtige Limitierungen, Bias und Fehlerquellen
3. **Setup**: Python, kurze Vorstellung von Tools (z. B. Make.com)
4. **Hands-on**: Kurzes Experiment mit OpenAI API oder einer lokalen GPT-Variante (bspw. Ollama/LM Studio) – Installation, erster Prompt

Tag 2: Prompting-Framework & Prompting-Grundlagen

1. **Strukturierter Ansatz zum Prompting**
   - Aufbau eines Prompt-Katalogs (System, Instruction, Output)
   - Zero-/One-/Few-Shot Prompting, Chain-of-Thought
2. **Prompting-Grundlagen**
   - Systemnachrichten, Rollen/Personas, Stilvorgaben
   - Delimiter und Format-Steuerung (z. B. Tabellen, Flowcharts)
3. **Praxis: Entwickler-Prompts**
   - Einbindung von Dokumentation, Code-Beispielen, Verzeichnisstrukturen
4. **Hands-on**:
   - Konkrete Prompts für eine Coding-Aufgabe (z. B. Generiere Unit Tests, Code Review, etc.)
   - Nutzung von Playground-Parametern (Temperature, Top-P)

Tag 3: Tools & Frameworks für LLM-Integration

1. **LangChain, Haystack & Co.**
   - Was ist RAG (Retrieval-Augmented Generation)?
   - Vektordatenbanken (Chroma, FAISS, Pinecone)
   - Embeddings einbinden (OpenAI, HF)
2. **Streamlit & Gradio**
   - Schnelles Prototyping von Chat-Interfaces
3. **Multimodale Fähigkeiten**
   - Kurzüberblick: CLIP, Stable Diffusion, In-/Outpainting, wann relevant für Developer?
   - „Sehen“ und „Hören“ (Bild- und Audio-Funktion) – meist nur über Add-ons
4. **Hands-on**:
   - Baue einen kleinen Chatbot mit RAG (z. B. Dokumente in Chroma indexieren), verbinde ihn mit GPT-API, und setze ein Gradio-Demo-Interface auf

Tag 4: Typische Problemfälle & fortgeschrittenes Prompting

1. **Jailbreaking & Prompt Injection**
   - Risiken, Grenzen, Sicherheitsaspekte
2. **Backup-LLM-Systeme**
   - Wann man mehrere Modelle parallel nutzen sollte
   - Fallback-Strategien (GPT-3.5 → GPT-4 oder Claude o. Ä.)
3. **Hyperparameter & Feineinstellungen**
   - Temperature, Top P, Frequency/Presence Penalty, Stop Sequences
   - Wie man ungewünschte Ausgaben minimiert oder mehr Kreativität erzwingt
4. **Datenschutz & Unternehmensrichtlinien**
   - Code & Daten nur in On-Prem-LLMs?
   - Verschlüsselung, Pseudonymisierung
5. **Hands-on**:
   - Spiele im Playground mit verschiedenen Hyperparametern
   - Demonstriere Prompt Injection / Jailbreaking und wie man sich schützt

Tag 5: Abschlussprojekt – KI in die Anwendung einbetten

1. **KI-Agenten & Integration**
   - Kurzvorstellung: BrowserUse, DeepResearch, Operator
   - Make.com / Zapier / n8n: Automatisierte Prozesse mit KI
2. **Auswahl & Einrichtung von LLMs**
   - GPT, Claude, Llama, Gemini, Deepseek
   - Kosten & Hardware (wenn Open-Source lokal)
3. **Abschlussprojekt**
   - Baue eine kleine **Developer Assistant**-App:
     - LLM + RAG + Streamlit/Gradio-Frontend
     - Prompt-Katalog (bestimmte Use-Cases)
   - Stelle den Prototyp in kleinem Team oder im Kurs vor
4. **Ausblick**
   - Weitere Möglichkeiten: Embeddings, Transcription (Whisper), MLOps-Skalierung