Big Brother is watching you

Lokales GenAI in Anwendungen

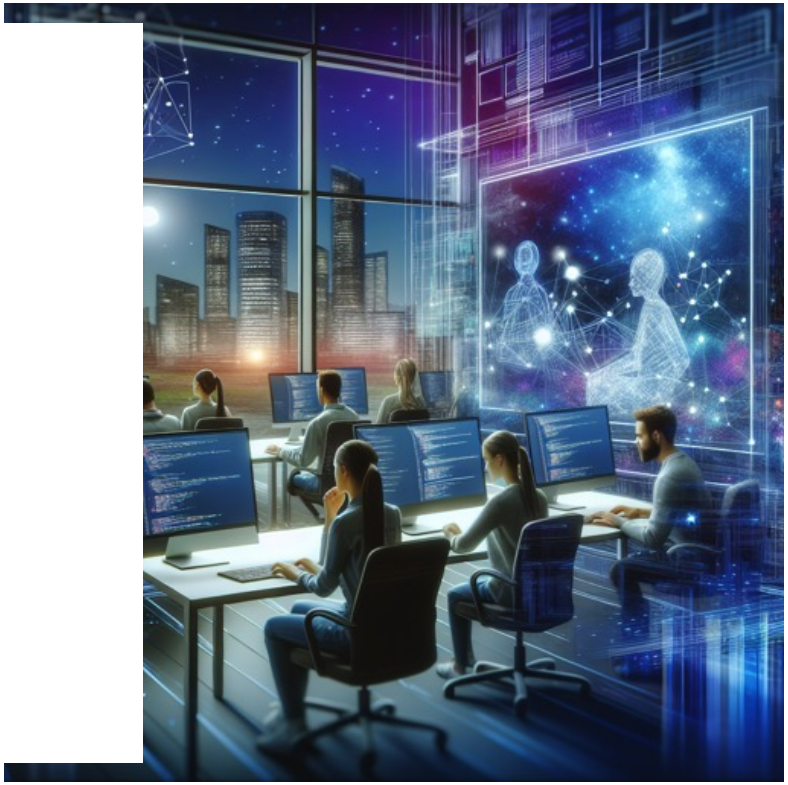

Einführung

Was ist GenAI ?

```
using System;

0 references
class Program
{
    0 references
    static void Main()
    {
        string[] poem = {
            "In digitaler Welt, so bunt und weit, sind .NET Entwickler stets bereit.",
            "Mit C# und ASP.NET zur Hand, erschaffen sie, was jeder kennt im Land.",
            "",
            "Vom Backend bis zum Frontend-Glanz, sie tanzen durch den Code, als wär's ein Tanz.",
            "Mit Azure und mit Blazor stets im Blick, lösen sie Probleme, Stück für Stück.",
            "",
            "Ob Datenbank, ob API, stets klar, kein Problem ist ihnen je zu rar.",
            "Mit Leidenschaft und einem klaren Ziel, erschaffen sie Software, die kann wirklich viel.",
            "",
            "So feiern wir die .NET Helden heut', denn ihre Arbeit, die uns stets erfreut.",
            "In der Welt der Codes sind sie die Besten, .NET Entwickler, ihr seid die Coolsten, ohne zu testen!"
        };

        foreach (string line in poem)
        {
            Console.WriteLine(line);
        }
    }
}
```



Cloud vs. Lokale LMs

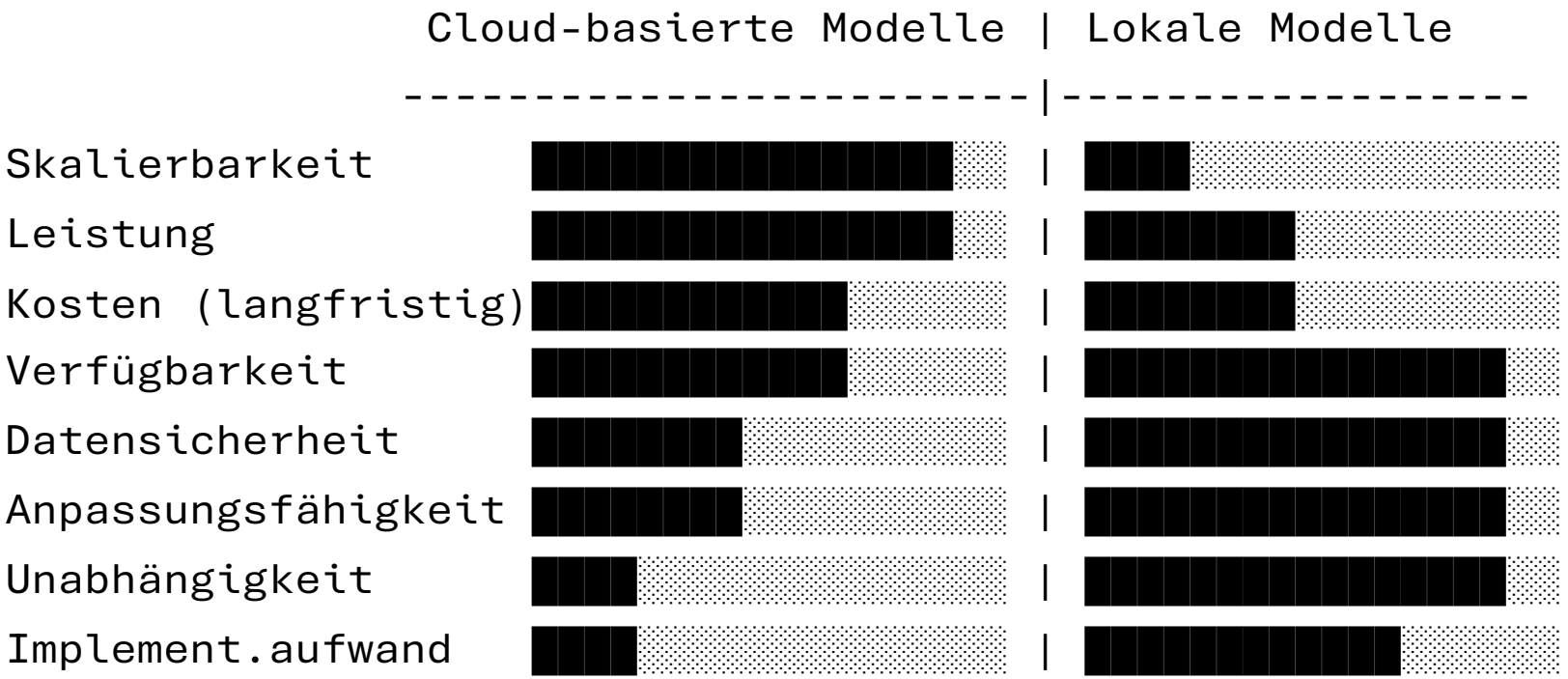
Cloud-basierte Modelle:

- Viele Unternehmen setzen auf Cloud-Lösungen für ihre KI-Anwendungen, da sie einfach zu skalieren und zu integrieren sind. Allerdings gibt es erhebliche Bedenken hinsichtlich Datenschutz und Datensicherheit.

Lokale Modelle:

- Lokale Modelle bieten eine Alternative, bei der sensible Daten im Unternehmensnetzwerk bleiben und nicht in die Cloud übertragen werden müssen. Dies bietet Vorteile in Bezug auf Datenschutz, Unabhängigkeit und Leistung.

Gegenüberstellung (PaaS/SaaS)



Relevanz von Small Language Models

Definition und Abgrenzung

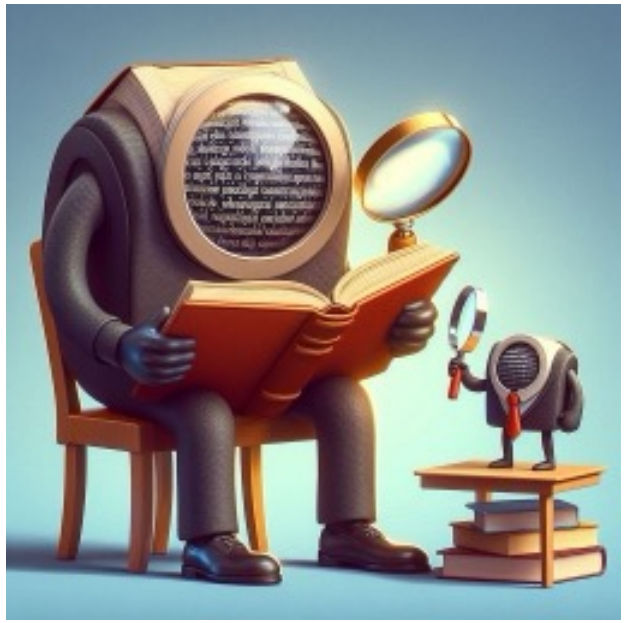
"Small Language Models (SLMs) sind eine **spezialisierte Form** von Large Language Models (LLMs), die für den **Einsatz auf lokalen Geräten optimiert** sind. Sie bieten viele der **Vorteile** von LLMs, **jedoch** mit einem **geringeren Ressourcenbedarf** (was sie ideal für lokale Implementierungen macht)."

Modell- und Trainingsmerkmale

Kriterium	Small Language Models (SLMs)	Large Language Models (LLMs)
Modellgröße	Typischerweise < 30 Milliarden Parameter	Hunderte Milliarden bis Billionen Parameter
Trainingsdaten	Kleinere, fokussierte Datensätze	Umfangreiche, vielfältige Datensätze
Trainingszeit	Wochen	Monate
Rechenleistung	Geringer, kann auf lokalen Geräten laufen	Hoch, erfordert oft Cloud-Ressourcen

Leistungsfähigkeiten von SLMs

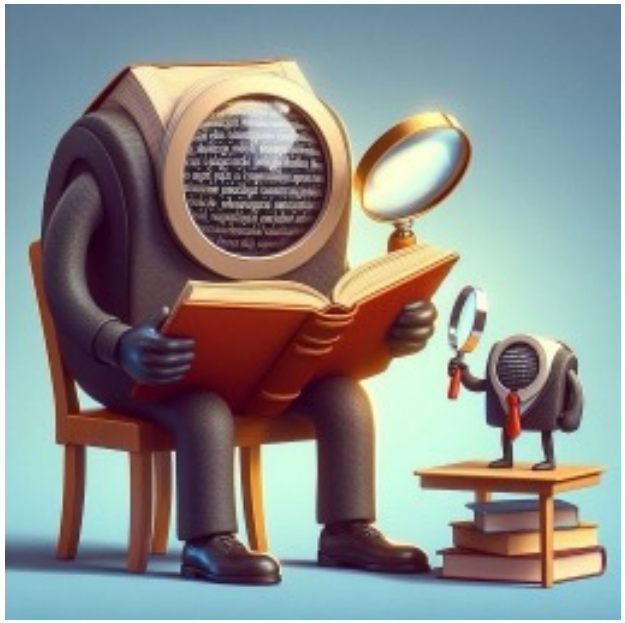
Category	Phi-3.5-MoE-instruct	Mistral-Nemo-12B-instruct-2407	Llama-3.1-8B-instruct	Gemma-2-9b-It	Gemini-1.5-Flash	GPT-4o-mini-2024-07-18 (Chat)
Popular aggregated benchmark	62.6	51.9	50.3	56.7	64.5	73.9
Reasoning	78.7	72.2	70.5	75.4	77.7	80.0
Language understanding	71.8	67.0	62.9	72.8	66.6	76.8
Robustness	75.6	65.2	59.8	64.7	68.9	77.5
Long context	25.5	24.5	25.5	0.0	27.0	25.4
Math	74.1	57.7	65.0	67.9	60.2	80.8
Code generation	68.3	56.9	65.8	58.3	66.8	69.9
Multi-lingual	**65.8**	**55.3**	**47.5**	**59.6**	**64.3**	**76.6**



https://www.trend.at/tech/small-language-

Leistungsfähigkeiten von SLMs

Category	Phi-3.5-MoE-instruct	GPT-4o-mini-2024-07-18 (Chat)
Popular aggregated benchmark	62.6	73.9
Reasoning	78.7	80.0
Language understanding	71.8	76.8
Robustness	75.6	77.5
Long context	25.5	25.4
Math	74.1	80.8
Code generation	68.3	69.9
Multi-lingual	**65.8**	**76.6**



16 Experten / je 3,8B Parameter → theoretische 42B Parameter
2 aktive Experten → 6,6B

https://techcommunity.microsoft.com/blog/azure-ai-services-blog/discover-the-new-multi-lingual-hig

Update: Phi4-Mini

Category	Similar size					2x size					
	Phi-4 mini-Ins	Phi-3.5-mini-Ins	Llama-3.2-3B-Ins	Mistral-3B	Qwen2.5-3B-Ins	Qwen2.5-7B-Ins	Mistral-8B-2410	Llama-3.1-8B-Ins	Llama-3.1-Tulu-3-8B	Gemma2-9B-Ins	GPT-4o-mini-2024-07-18
Popular aggregated benchmark	55,8	52,6	43,4	43,6	49,5	64,2	47,6	50,3	51,0	57,7	68,5
Reasoning	58,5	58,6	56,2	58,6	55,7	62,5	59,2	59,7	59,9	75,4	78,0
Multilingual	56,6	50,7	46,4	45,5	54,7	64,5	55,2	56,5	56,6	69,5	77,3
Math	76,3	63,4	61,2	61,0	71,2	74,6	61,8	65,0	65,2	68,1	80,8
Overall	**63,5**	**60,5**	**56,2**	**56,9**	**60,1**	**67,9**	**60,2**	**62,3**	**60,9**	**65,0**	**75,5**

https://huggingface.co/microsoft/Phi-4-mini-instruct (modifiziert)

Machine Learning-Model Typen

- **Computer Vision**
  - Modelle, die ausschließlich visuelle Informationen verarbeiten (z. B. Bildklassifikation, Objekterkennung, Segmentierung)
  - Beispiele: CNNs, Vision Transformer (ViT) bei rein bildbasierten Aufgaben
- **Natural Language Processing (NLP)**
  - Modelle, die primär für textbasierte Aufgaben entwickelt wurden (z. B. Textgenerierung, Übersetzung, Sentiment-Analyse)
  - Beispiele: LLMs (Large Language Models) und SLMs (Small Language Models)
- **Multimodale Modelle**
  - Modelle, die gezielt mehrere Datenmodalitäten kombinieren (z. B. Bild und Text oder Bild und Audio)
  - Beispiele: Vision-Language-Modelle, die sowohl visuelle als auch sprachliche Eingaben berücksichtigen, sowie multimodale Embedding-Modelle
- **Audio/Speech**
  - Modelle, die sich auf die Verarbeitung und Interpretation von Audio- bzw. Sprachdaten spezialisieren
  - Beispiele: Modelle zur Spracherkennung, Sprachsynthese und Klangklassifikation

Möglichkeiten zur lokalen Implementierung von LMs

Verschiedene Ansätze

TensorFlow

PyTorch

Hugging Face Transformers

ONNX

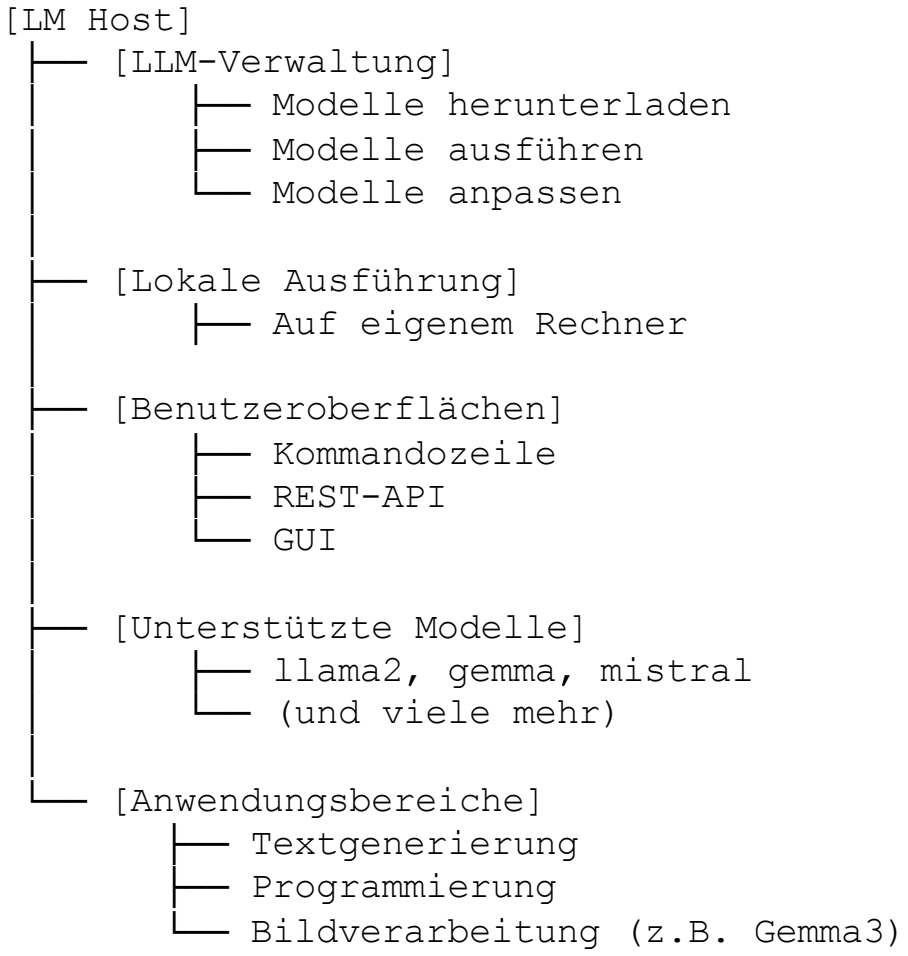
Apache MXNet

LM server

- Ollama
- LM Studio
- GPT4ALL
- LLaMa.cpp
- Llamafile
- CodeProject.AI

.
.
.

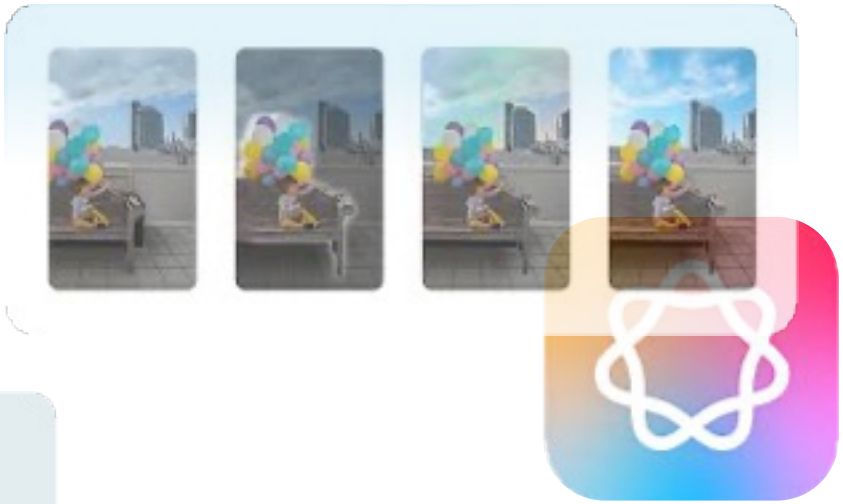
>



Implementierungen mit lokalen GenAI-Modellen

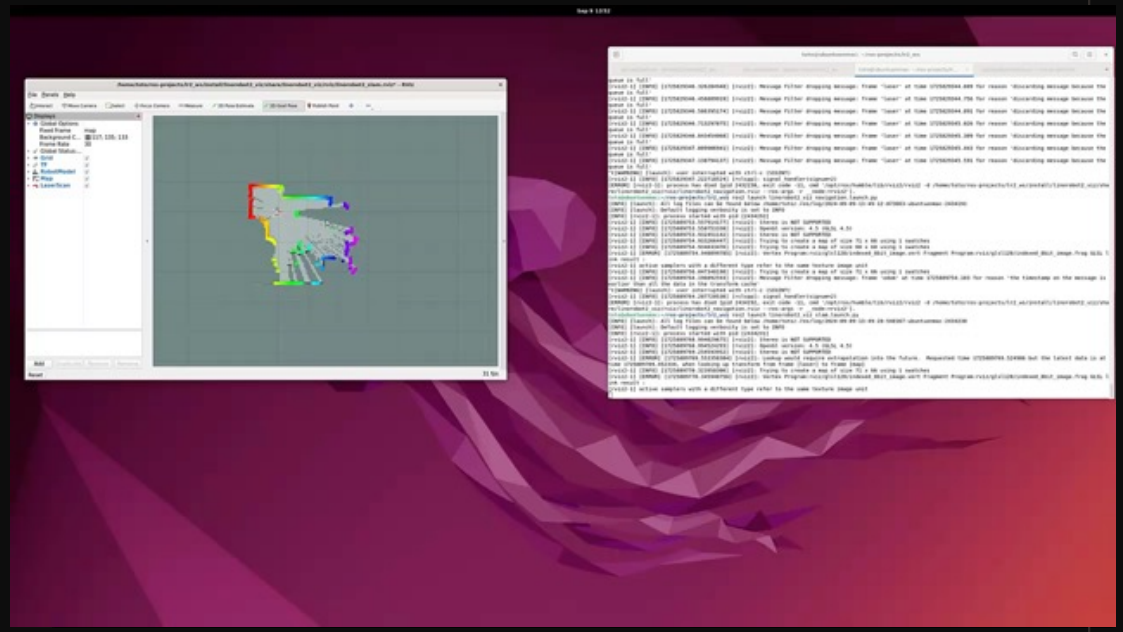
Beispiele aus dem täglichen oder zukünftigen Leben

- Mobile Phone Assistants
  - IPhone
  - Google Pixel



- Recall Features
  - Microsoft Recall
  - OpenRecall
  - Rewind.ai

Anwendungsbereich: Robotics



Herausforderungen und Grenzen von lokalen LLMs

Einschränkungen und Strategien

- Physische Limits

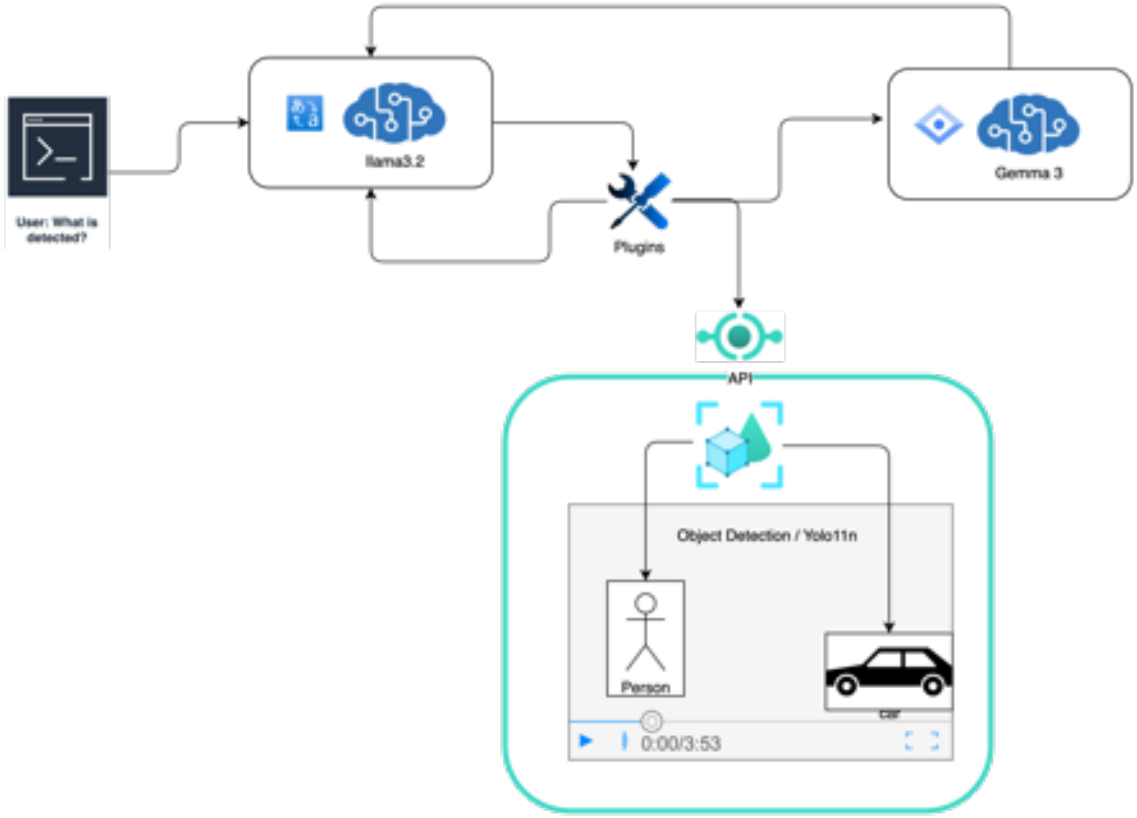
  - Ressourcen
  - Modellgrößen
  - Infrastruktur
- Soft Limits

  - Genauigkeit & Zuverlässigkeit
  - Anpassungsfähigkeit
  - Komplexität der Lösung
  - Wartung & Updates

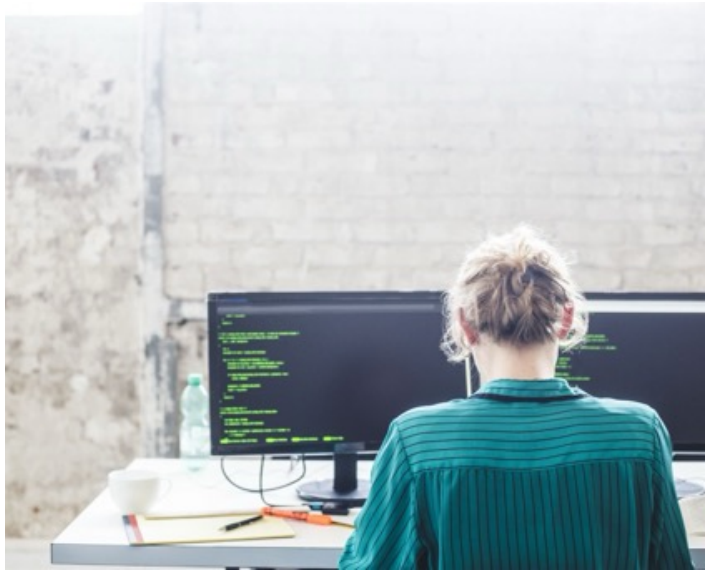

Demo-Time
Show me CODE

Demo-Case: Verstehen und Sehen

- Video Stream mit Object Detection
- SLM versteht Users Intent
- Fragt greift über Tools auf die VideoApp zu
- Für genauere visuelle Bewertung weiteres SVLM



Getting Started



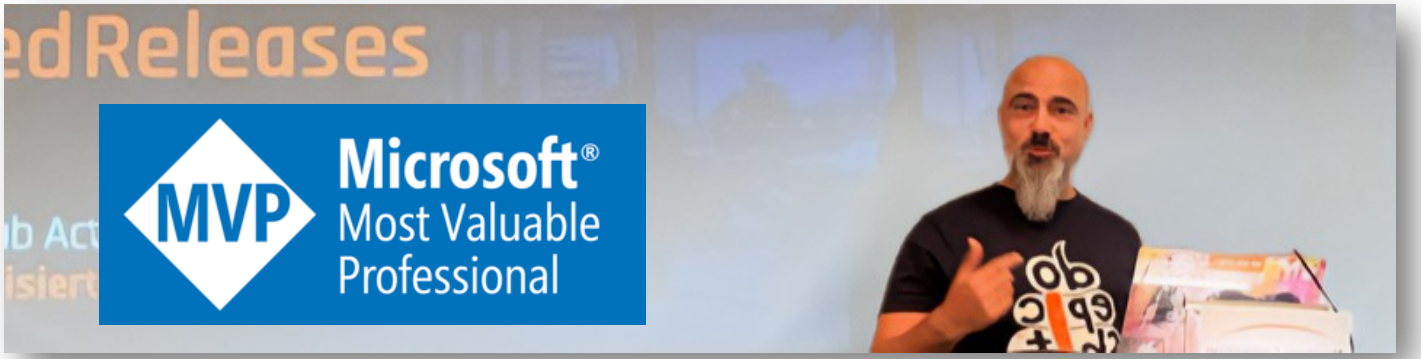
- **Installiere LLM-Host**:
  - Wähle und installiere einen LLM-Host wie Ollama oder LM Studio…
- **Setze Entwicklungsumgebung auf**:
  - Installiere Visual Studio Code.
  - Installiere das .NET SDK für C#-Entwicklung.
  - Installiere die C#-Erweiterung in VSCode.
- **Wähle das Modell aus**:
  - Konfiguriere den LLM-Host und wähle das gewünschte Modell aus.
- ***(Starte den LLM-Server:***
  - *Starte den LLM-Host, um den Server mit dem ausgewählten Modell zu betreiben.)*
- **Verwende passendes SDK**:
  - Nutze ein SDK, das mit OpenAI API-kompatiblen Servern funktioniert, um die Integration zu erleichtern.
- **Schreibe deine Anwendung**:
  - Erstelle ein neues C#-Projekt mit dotnet new console -n MeinCSharpProjekt.
  - Implementiere die Logik deiner Anwendung in C#.
- **Binde den Server ein**:
  - Integriere den LLM-Server in deine Anwendung, indem du die API des Servers aufrufst.

Zusammenfassung

- Ist Lokal wirklich etwas für mich?
- Kenne dein Modell!
- Sorge für eine passende Infrastruktur (Menschen & Tools)
- Verstehe die Effekte von GenAI in deiner Anwend
- Sei motiviert zu experimentieren!

Demo - Repo

Thomas Tomow



CTO & COO @

Xebia | Microsoft Services

GitHub MEETUP STUTTGART

Azure Meetup KONSTANZ

Azure Meetup STUTTGART

Zeit für Fragen