
ERFOLG MIT KI-AGENTEN
LangGraph vs. Camunda 8
viadee
IT-Unternehmensberatung

# Über mich

viadee

Mario Micudaj

## Senior Consultant

- Bei der viadee seit 2018
- Verheiratet, zwei Töchter
- Prozessautomatisierung, Data & AI, Cloud, Architetur

# Agenda

01 Anwendungsfall beim Kunden

02 LangGraph

03 Camunda 8

04 Vergleich

05 Q & A

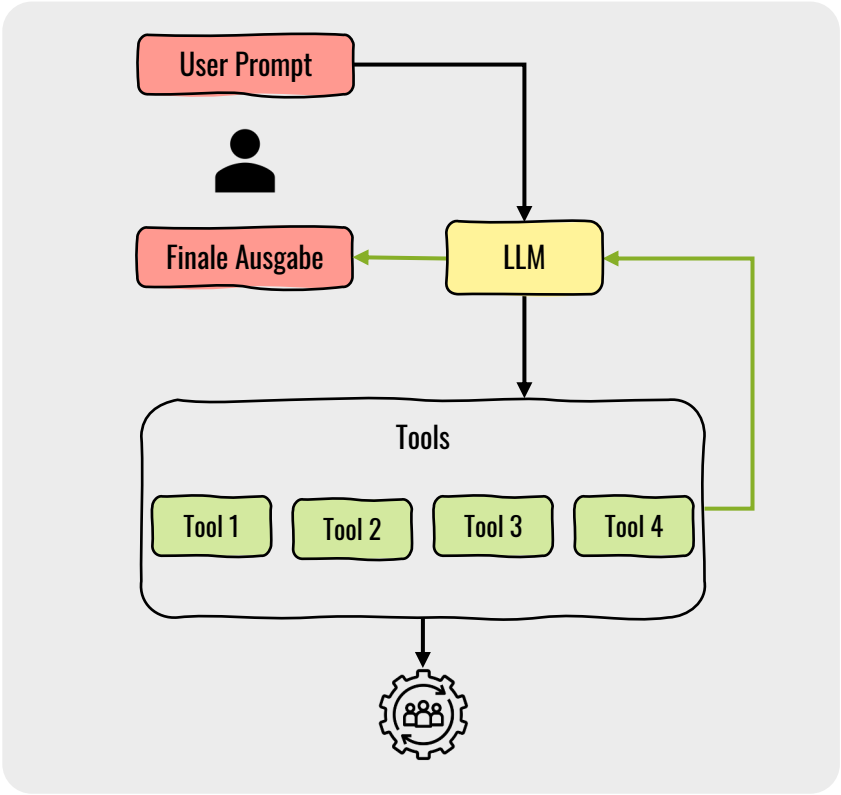
# KI-Agent

nimmt seine Umgebung wahr

trifft selbstständig Entscheidungen

handelt, um gesetzte Ziele zu erreichen

# KI-Agent




nimmt seine Umgebung wahr

trifft selbstständig Entscheidungen

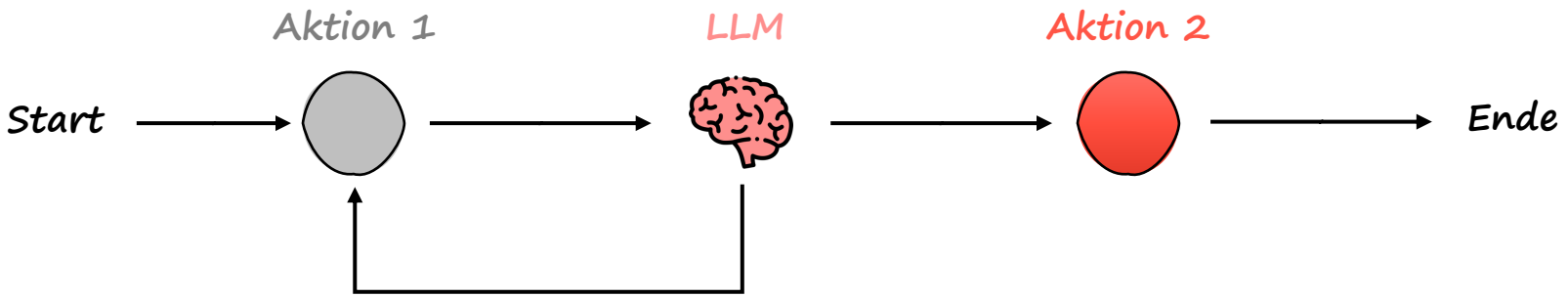
Memory

handelt, um gesetzte Ziele zu erreichen

System prompt

# Kontrollfluss

> *Steuerung des Kontrollflusses*
> *erfolgt durch ein Large Language Modell (LLM)*



# 1 Anwendungsfall beim Kunden

*Service Desk Agent*

# Kundenfall aus dem Handel

**20,000** Mitarbeitende benötigen **rund um die Uhr** Unterstützung durch den Service-Desk

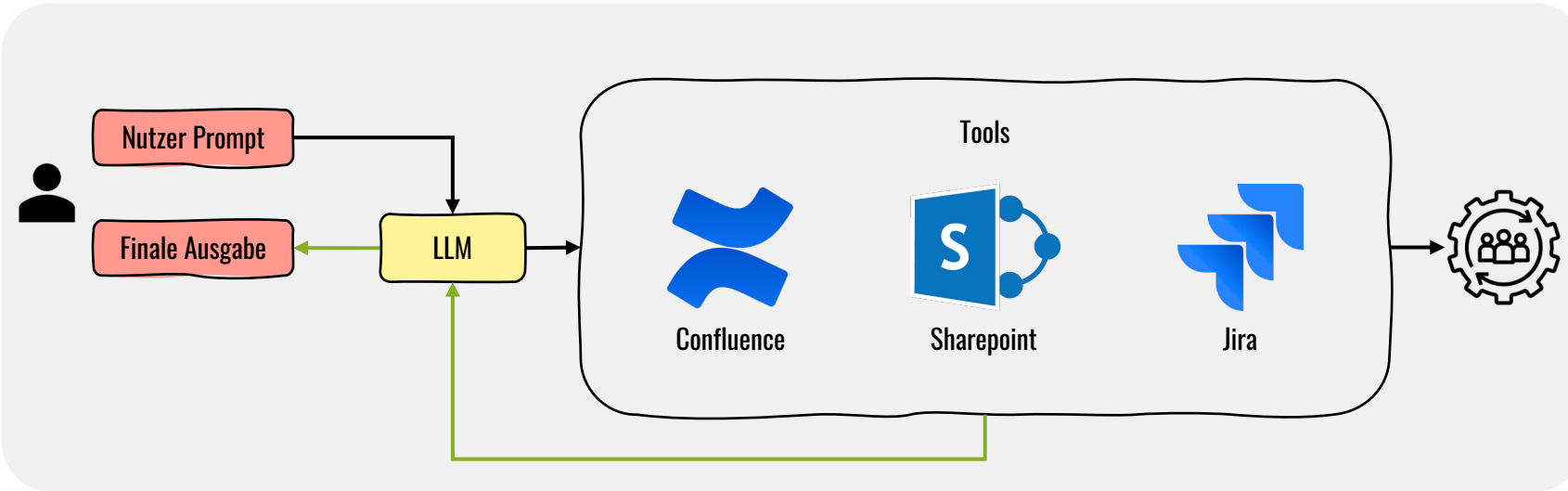
**> 2,000** Tickets müssen **jeden Monat manuell** von Hotline-Mitarbeitenden bearbeitet werden.

> *“Die Tastatur meines Laptops reagiert nicht auf meine Eingaben.”*

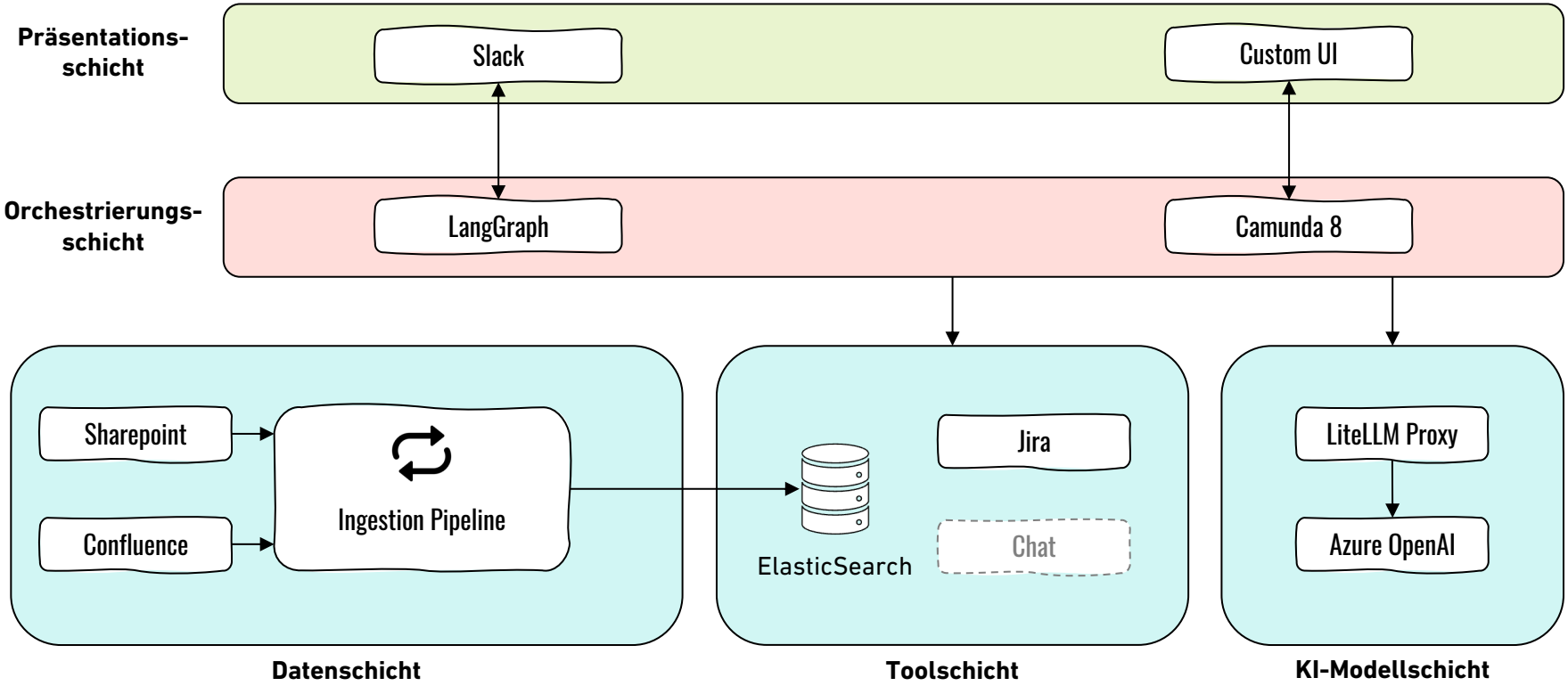
> *“Wie kann ich gelöschte E-Mails in Outlook wiederherstellen?”*

# Die Idee

Einen intelligenten Chatbot entwickeln, der die Anzahl der Support-Tickets reduziert, indem er Mitarbeitende befähigt Probleme selbständig lösen zu können und somit den Service-Desk entlastet



# Architektur

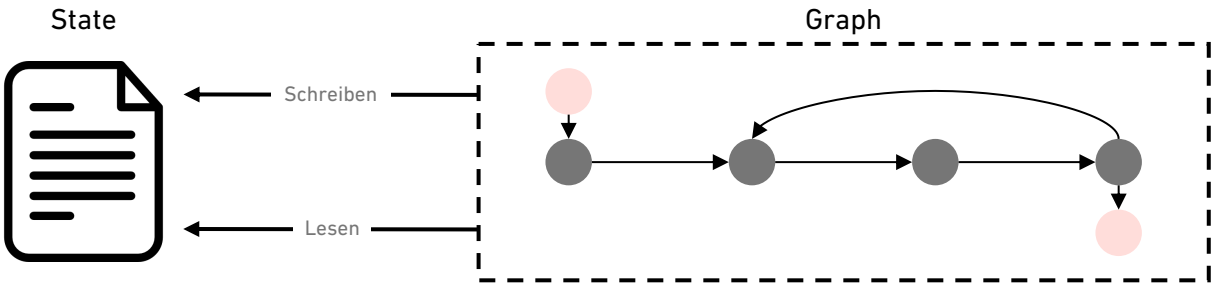


# 2 LangGraph

*Service Desk Agent*

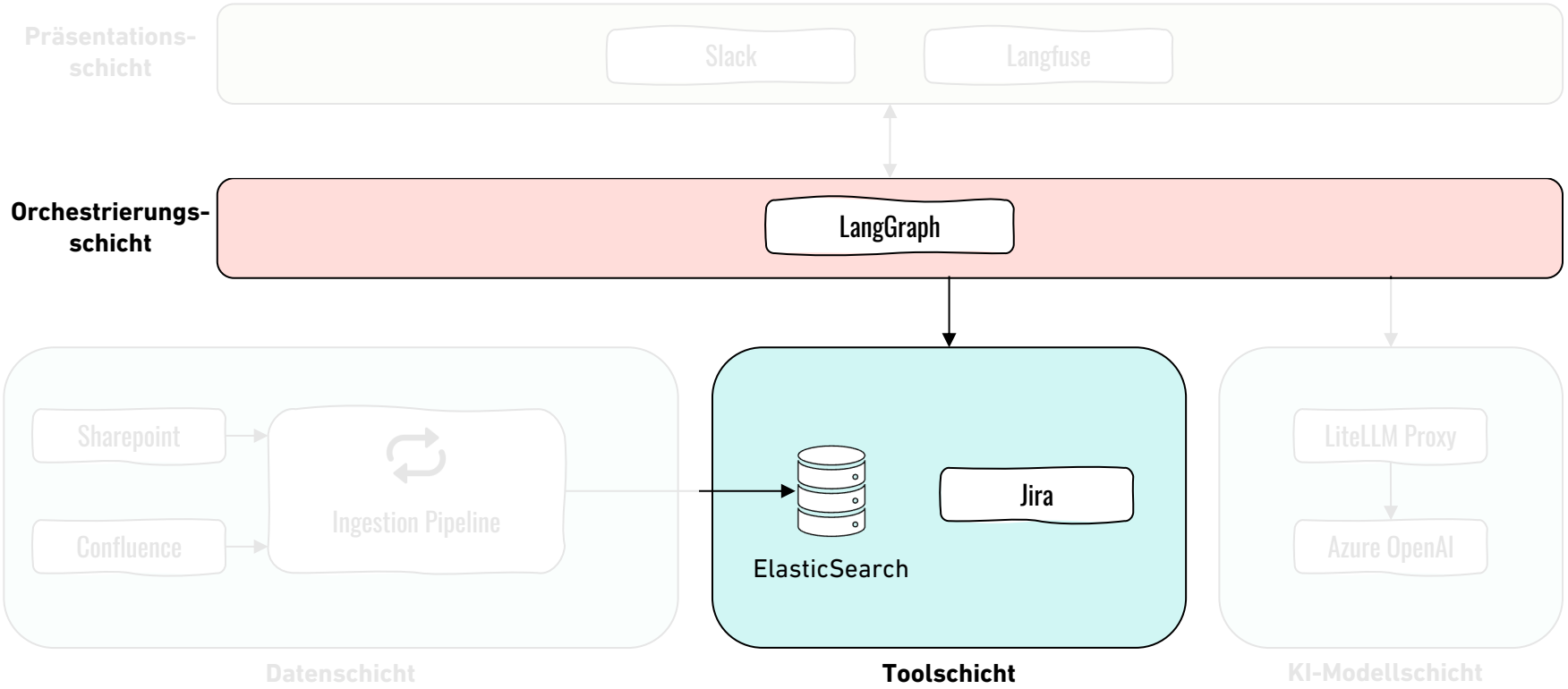
# LangGraph

*A low-level orchestration framework for building, managing, and deploying long-running, stateful agents.*




https://www.langchain.com/langgraph

# Architektur



# Umsetzung des Service-Desk-Agenten

... in drei Schritten:

1. Definition der verfügbaren Tools
2. Definition der Knoten und Kanten
3. Definition der Graphen und Agenten

# Umsetzung des Service-Desk-Agenten

... in drei Schritten:

1 Definition der verfügbaren Tools

Decorator

Beschreibung

```python
@tool
def ask_user_for_ticket_creation(request_text):
    """
    This tool can be used to create a Jira ticket for the user at the service desk so that someone can handle the request afterwards.
    """
    summary = get_llm().invoke(f"Generate a short, but descriptive title (do NOT exceed 250 characters) for a jira issue based on the following request: {request_text}").strip('"')
    description = get_llm().invoke(f"Generate a compact description for a jira issue with all details available from the following request: {request_text}")
    ticket = create_jira_ticket(summary, description)
    return f"Successfully created a jira ticket with the following details: {ticket}."

@tool
def query_knowledge_base(request_text):
    """
    Use this tool to search the knowledge database for the AI Agents Inc context.
    """
    query = get_llm().invoke(f"Generate a search query to search for relevant information in the knowledge database based on the following request: {request_text}")
    search_result = answer_query_against_rag(query)
    return f"Searching the knoweldge base has resulted in: {search_result}."
```

# Umsetzung des Service-Desk-Agenten

… in drei Schritten:

## 2 Definition der Knoten und Kanten

System prompt

```python
def llm_node(state: MessagesState):
    """LLM decides whether to call a tool or not."""

    prompt = """
        You are a helpful assistant named Hotline Agent for AI Agents Inc.

        Your main task is to handle internal requests from colleagues who need support with various topics.
        You have access to general information from the Confluence knowledge base and are eligible to create new Jira tickets for a user and view existing ones.
        .
        .
        .
    """

    response = llm_with_tools.invoke(
        [SystemMessage(content=prompt)]
        + state["messages"]
    )
    return {"messages": response}
```

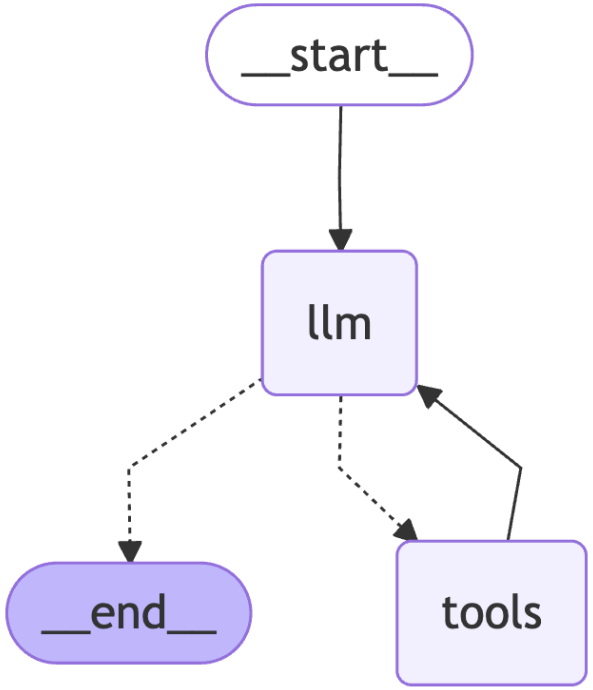
```python
def should_use_tool(state: MessagesState) -> str:
    """
    If the LLM has made a tool call, we are at the tools node.
    If the llm has not made a tool call, we want to escape the loop.
    """
    messages = state["messages"]
    last_message = messages[-1]
    return TOOLS_NODE if last_message.tool_calls else END
```

# Umsetzung des Service-Desk-Agenten

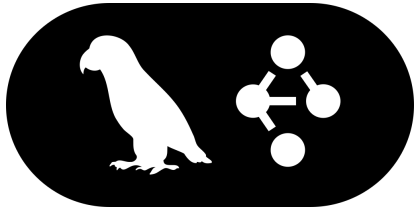
... in drei Schritten:

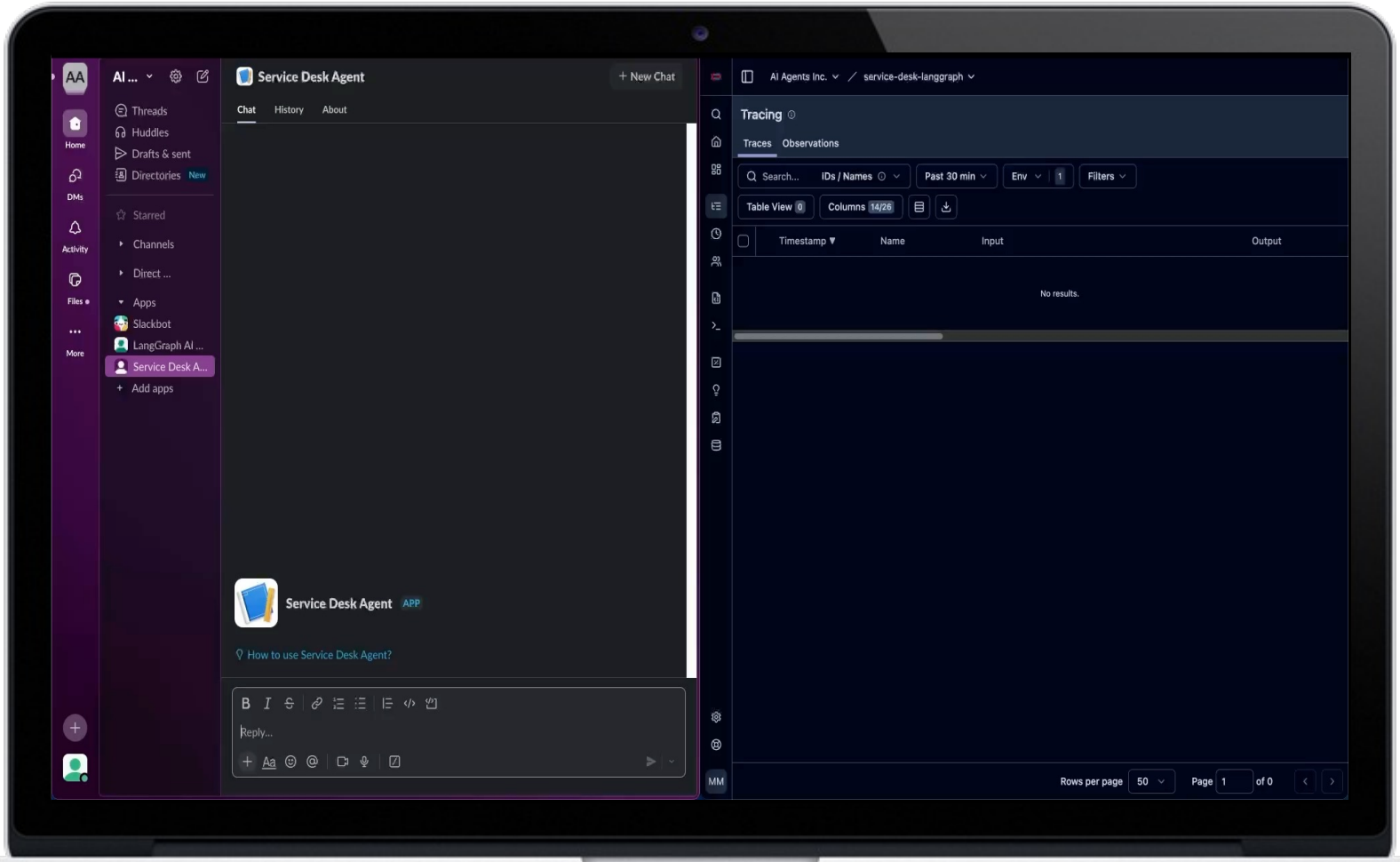
3 Definition der Graphen und Agenten

```python
agent_builder = StateGraph(MessagesState)
agent_builder.add_node(LLM_NODE, llm_node)
agent_builder.add_node(TOOLS_NODE, tool_node)

# Connect the nodes
agent_builder.add_edge(START, LLM_NODE)
agent_builder.add_conditional_edges(
    LLM_NODE,
    should_use_tool,
    {TOOLS_NODE: TOOLS_NODE, END: END}
)
agent_builder.add_edge(TOOLS_NODE, LLM_NODE)

# Compile the agent
agent = agent_builder.compile().with_config({"callbacks": [langfuse_handler]})
```

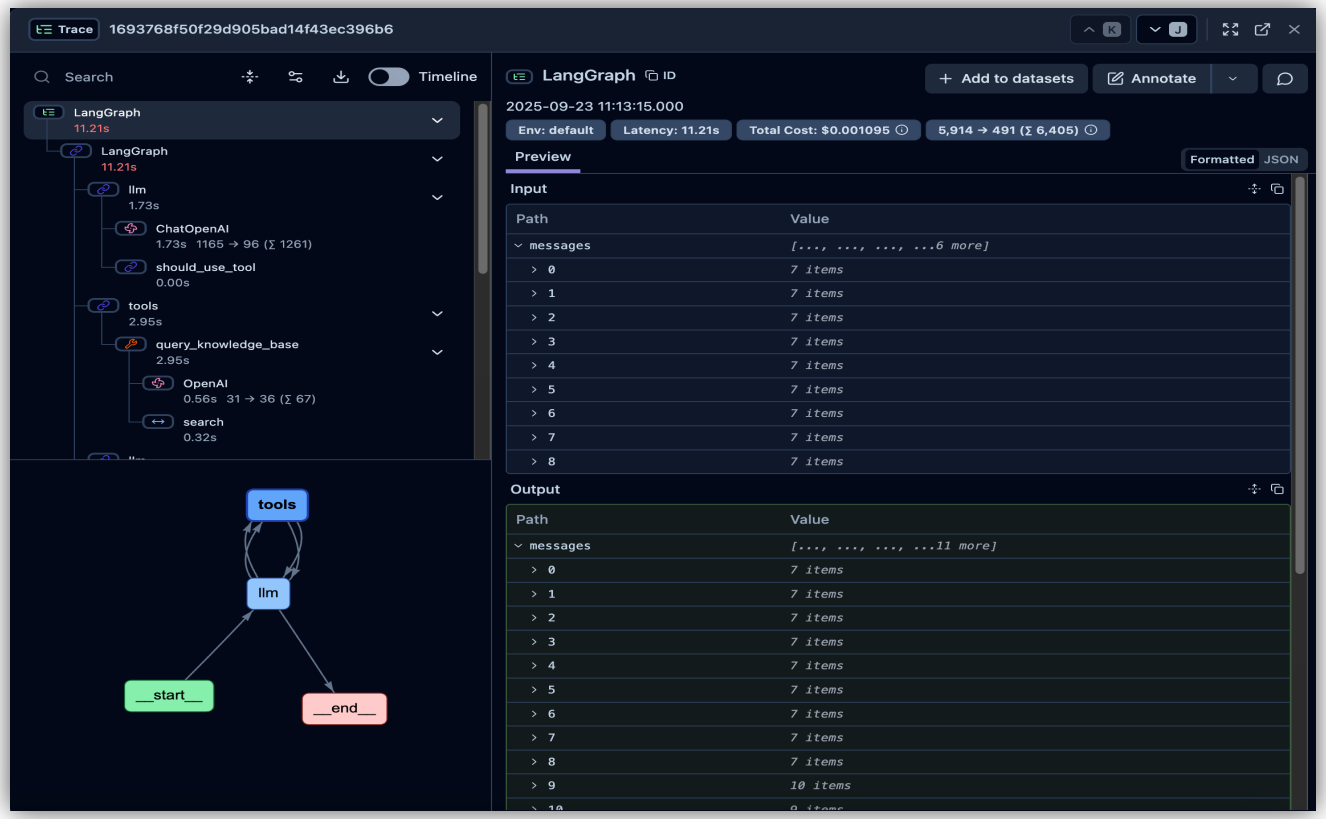


# Demo




AA
Home
DMs
Activity
Files
More
AI ...
Threads
Huddles
Drafts & sent
Directories New
Starred
Channels
Direct ...
Apps
Slackbot
LangGraph AI ...
Service Desk A...
Add apps
Service Desk Agent
+ New Chat
Chat
History
About
Service Desk Agent APP
How to use Service Desk Agent?
Reply...
AI Agents Inc.
service-desk-langgraph
Tracing
Traces
Observations
Search...
IDs / Names
Past 30 min
Env
1
Filters
Table View 0
Columns 14/26
Timestamp
Name
Input
Output
No results.
Rows per page
50
Page
1
of 0
MM

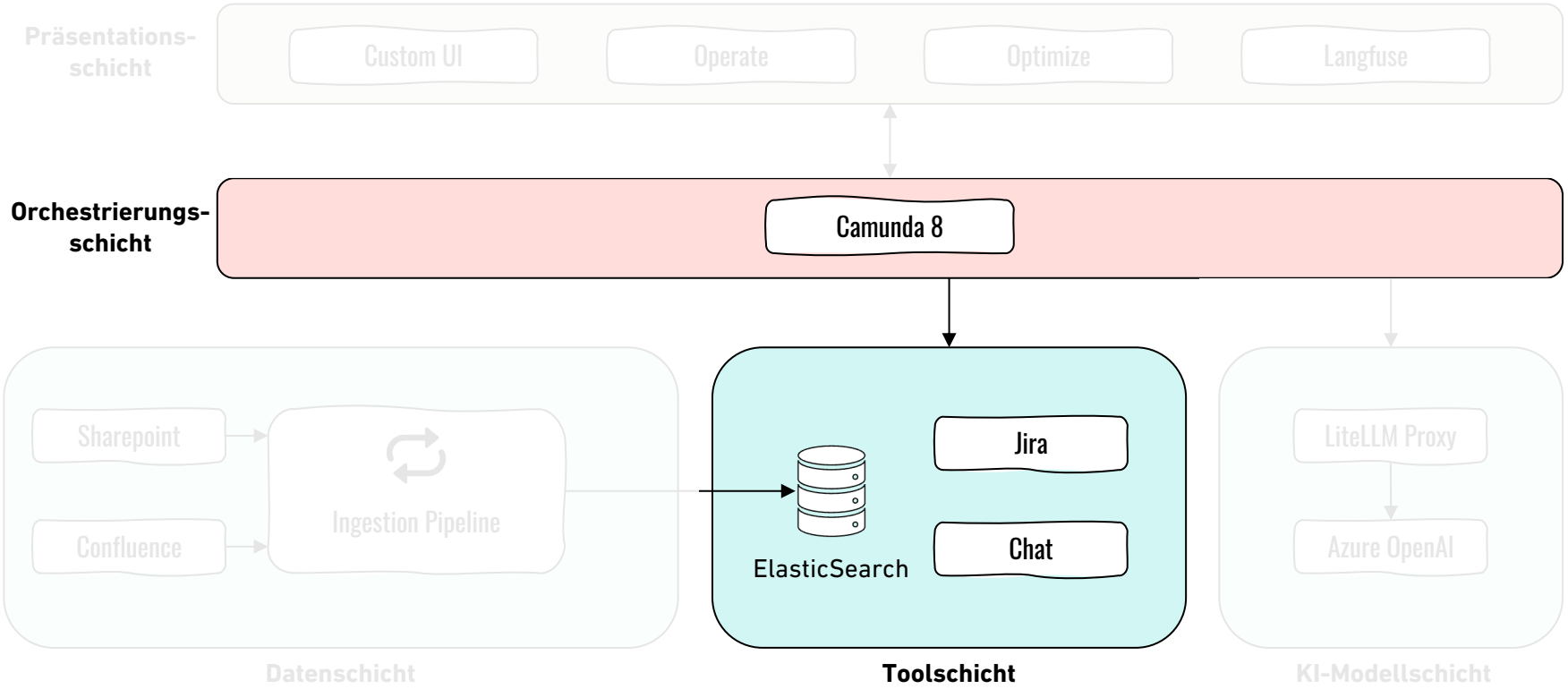
# Monitoring mit Langfuse



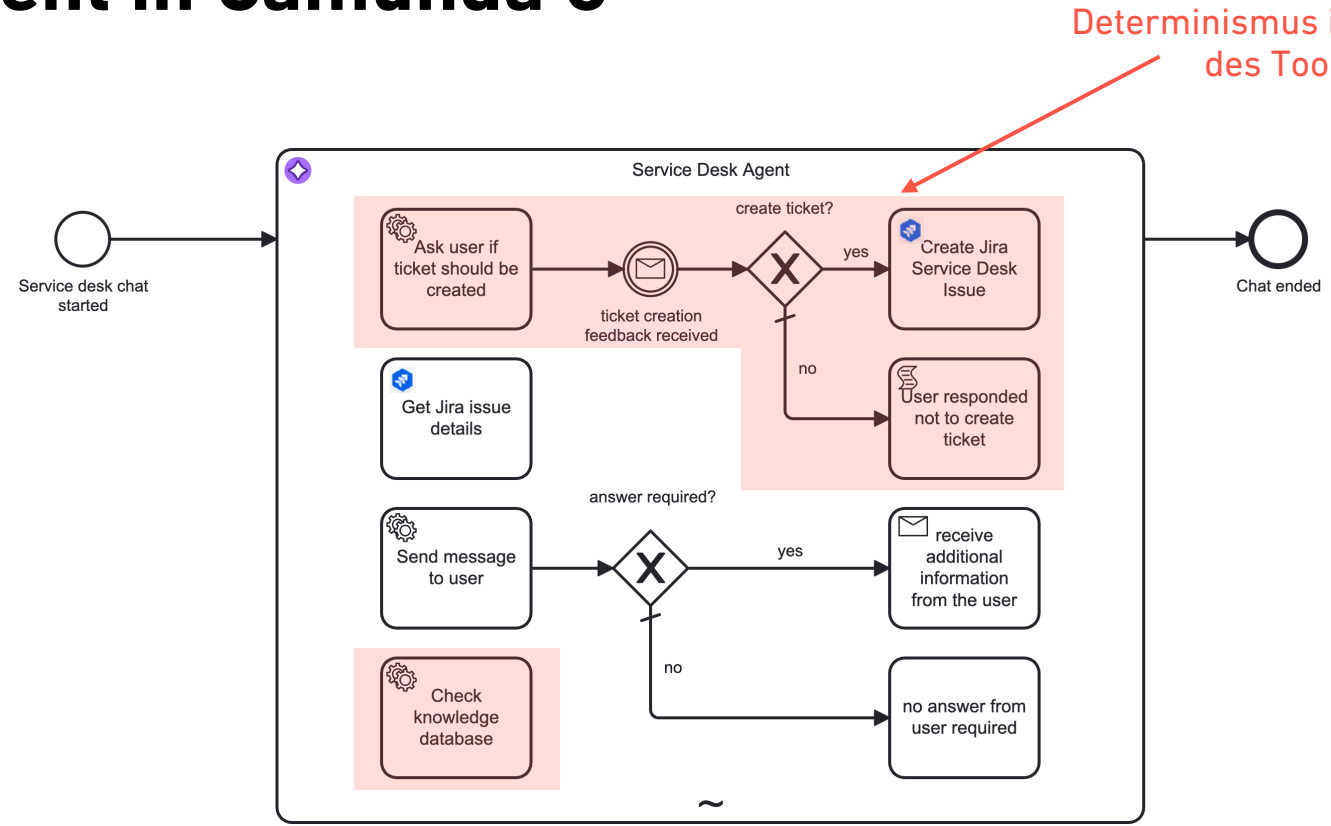
# 3 *Camunda 8*

*Service Desk Agent*

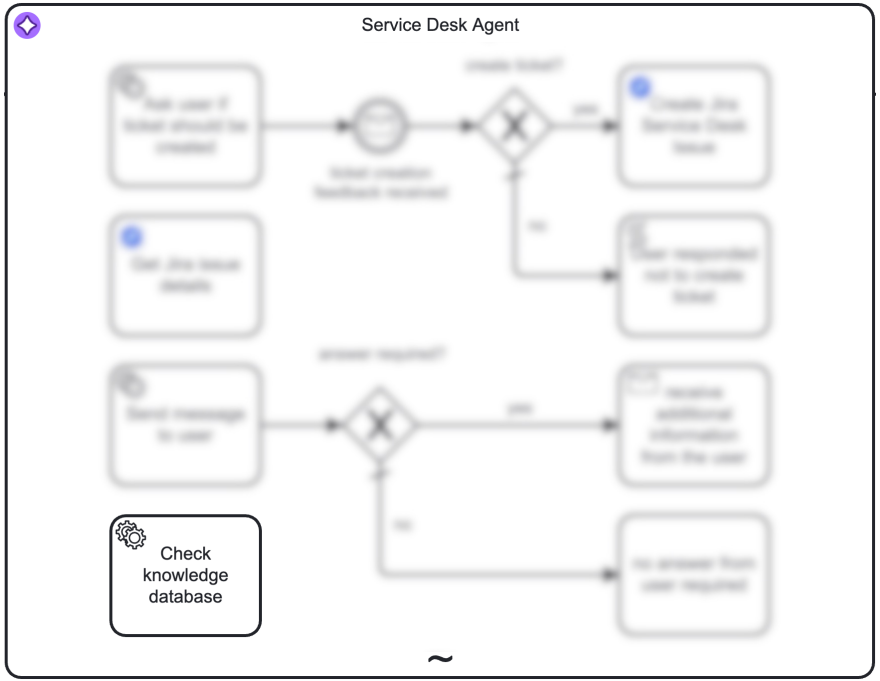
# Architektur



# KI-Agent in Camunda 8



# KI-Agent in Camunda 8




AI AGENT SUB-PROCESS
Service Desk Agent

General
Name: Service Desk Agent
ID: service_desk_agent

Documentation
Template: Applied
Model provider
Model
System prompt
User prompt
Memory
Limits
Event handling
Response
Output mapping
Error handling
Retries
Execution listeners

SERVICE TASK
Ask user if ticket should be created

General
Name: Ask user if ticket should be created
ID: ask_user_for_ticket_creation

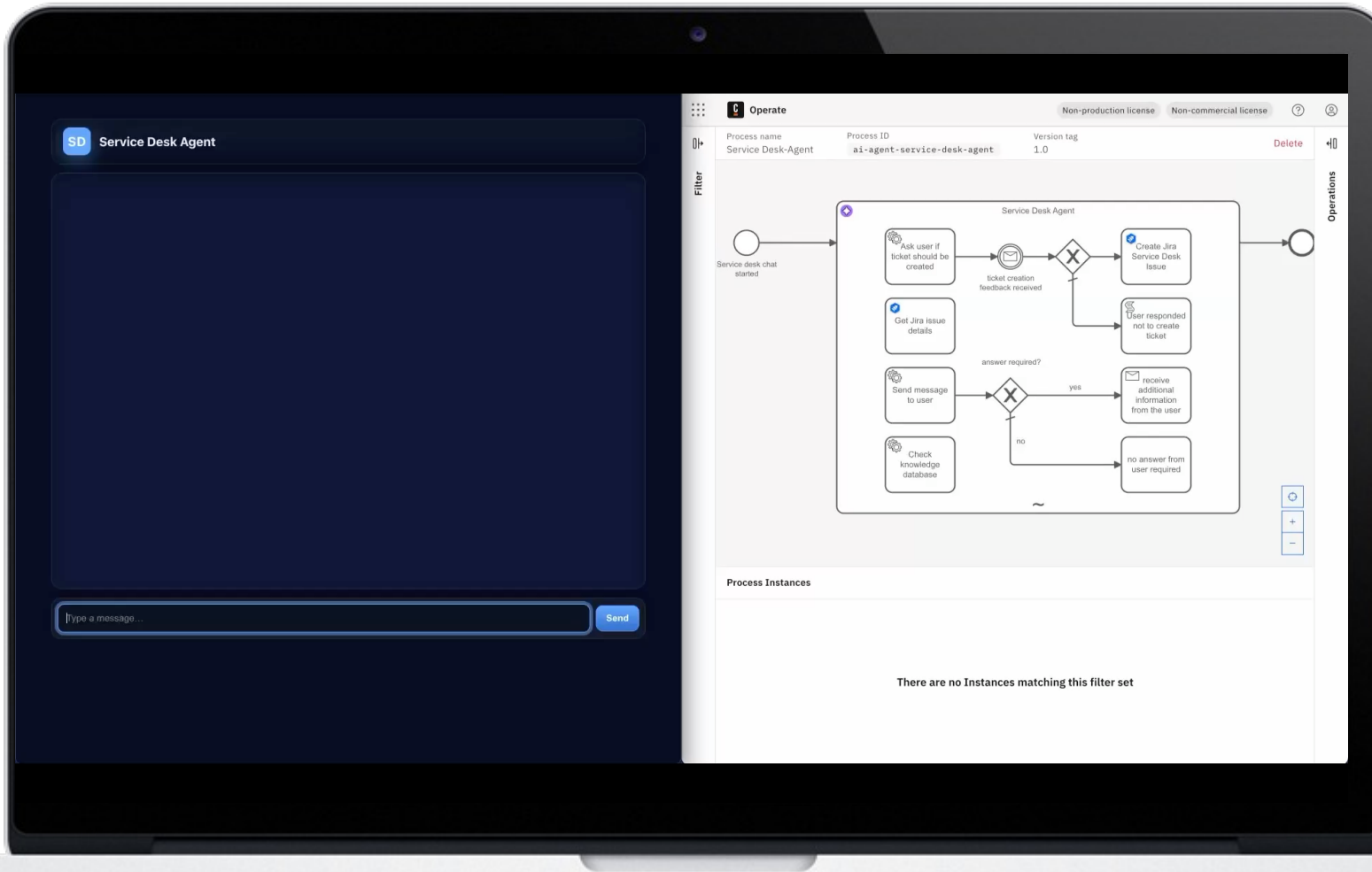
Documentation
Element documentation
This tool can be used to create a Jira ticket for the user at the service desk, so that someone can handle the request afterwards.

Template: + Select

Task definition
Job type: ask_user_for_ticket_creation
Retries

Inputs 2
summary
Local variable name: summary
Variable assignment value
= fromAi(toolCall.summary, "A consive title for the request of the user", "string")

# Demo

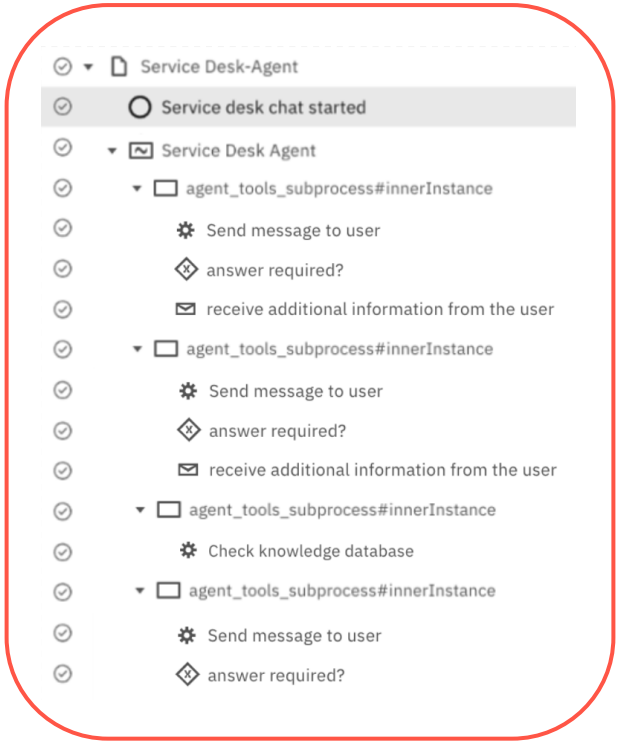

SD
Service Desk Agent
Type a message...
Send
Operate
Non-production license
Non-commercial license
Process name
Service Desk-Agent
Process ID
ai-agent-service-desk-agent
Version tag
1.0
Delete
Filter
Operations
Service Desk Agent
Service desk chat started
Ask user if ticket should be created
ticket creation feedback received
Create Jira Service Desk Issue
Get Jira issue details
User responded not to create ticket
answer required?
Send message to user
yes
receive additional information from the user
no
Check knowledge database
no answer from user required
Process Instances
There are no Instances matching this filter set

# Run-Time Monitoring

## Operate

### Auf Prozessinstanzebene

- Kontrollfluss der Toolnutzung
- Agenten-State



# Analyse & Optimierung

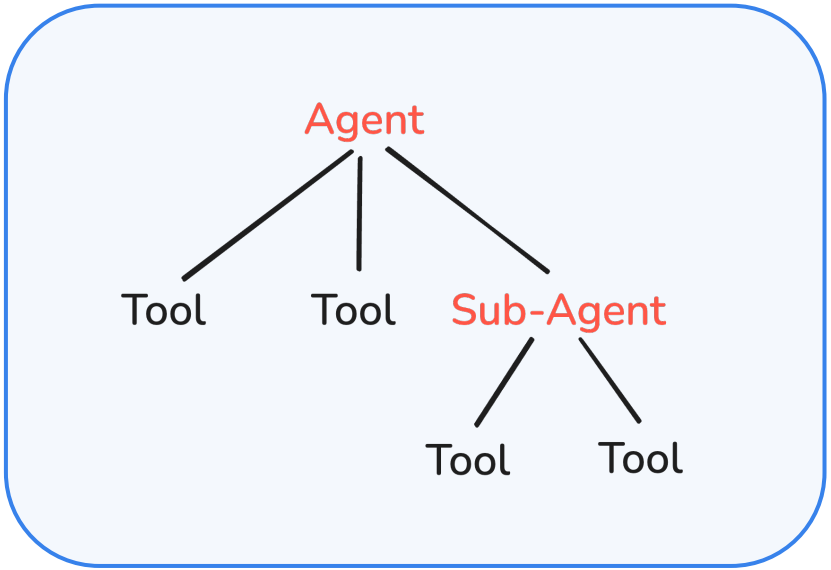
## Optimize

- Heatmap der Toolnutzung
- Prozessmetriken

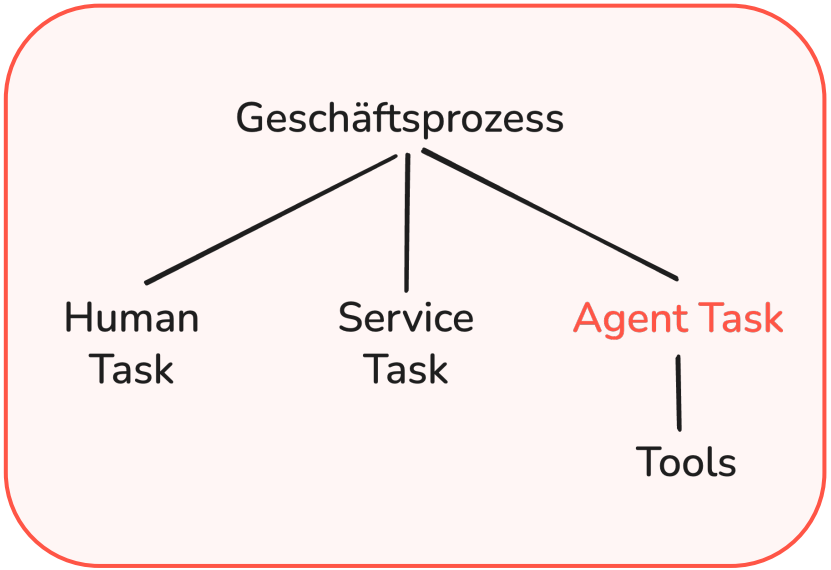
# 4 Vergleich

*LangGraph vs. Camunda 8*

# Architektur & Kontrollfluss

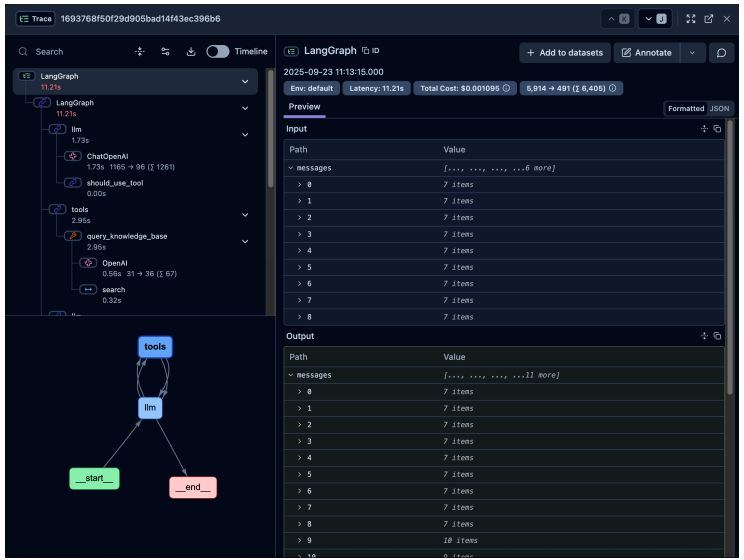


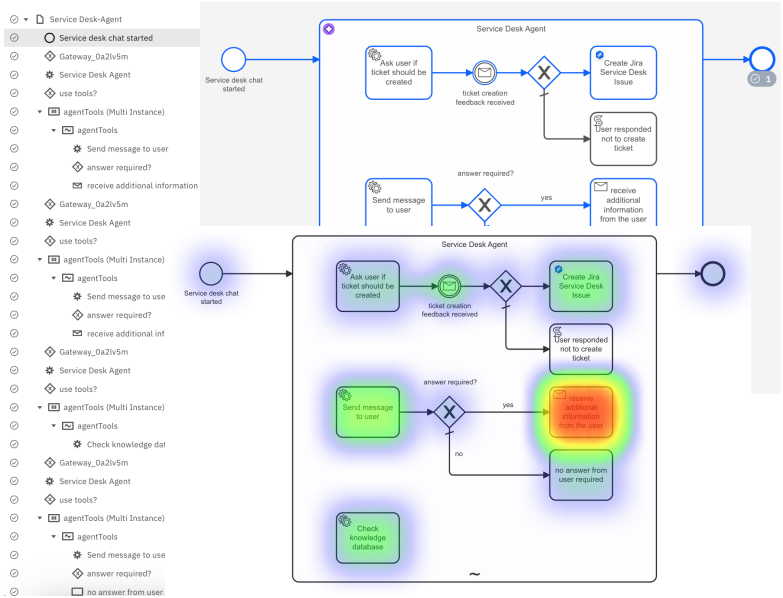

LangGraph: Agent orchestriert Denken




Camunda: Prozess orchestriert Arbeit
- auch mit Agenten

# Monitoring & Observability

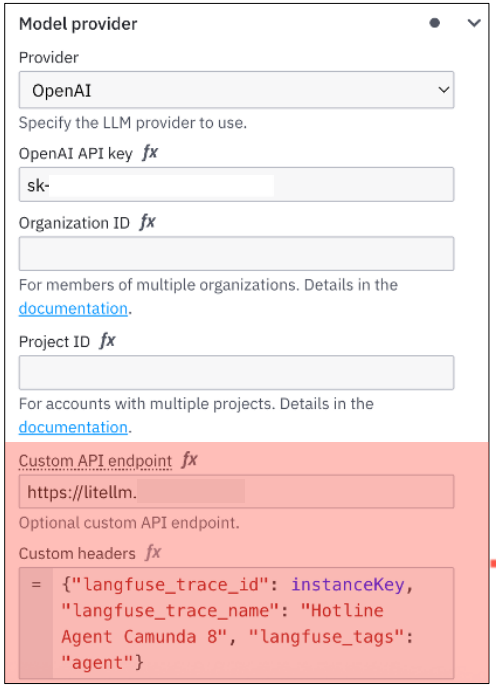


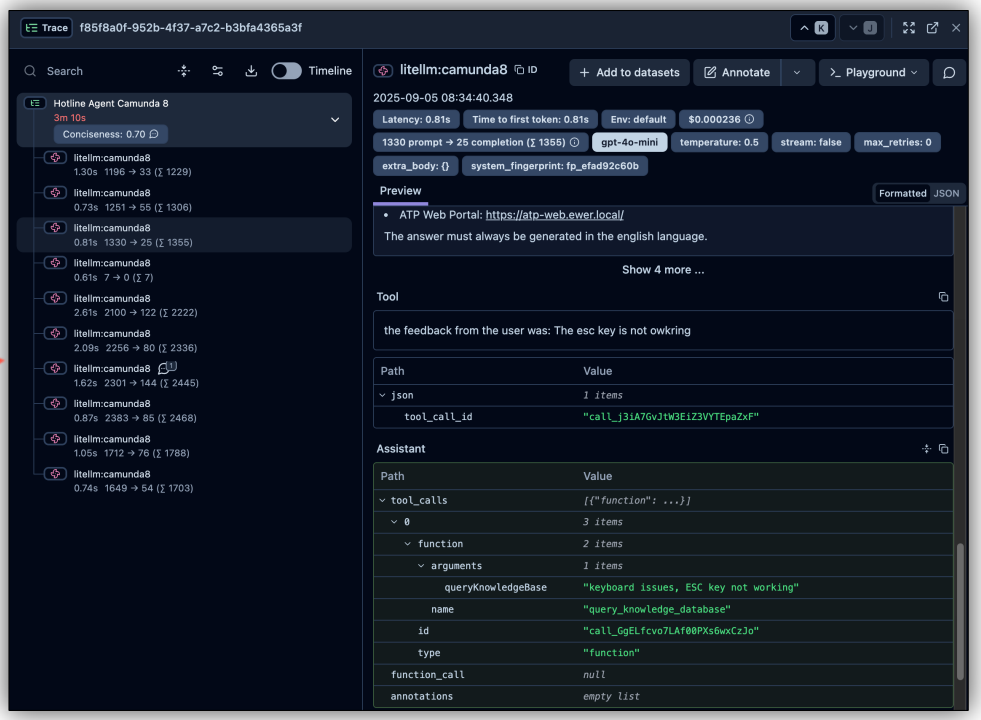




Fokus: Agentenebene

Fokus: Prozessebene

# LLM observability in Camunda 8



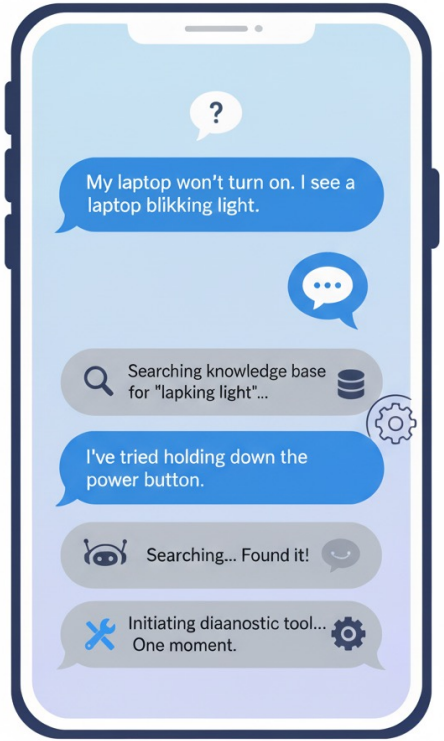
# Kontrollmechanismen

- Guardrails über Schemas, Tool-Restriktionen und Moderation
- Dynamisch zur Laufzeit
- Governance muss selbst definiert werden

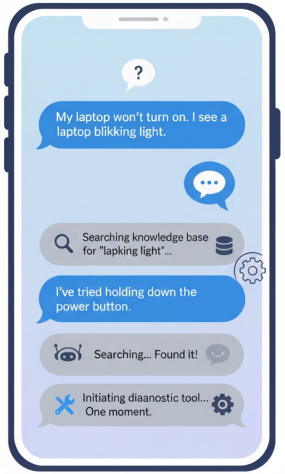
- Explizite Kontrolle durch BPMN-Prozesse
- Policies, Rollen und Freigaben eingebaut
- Governance und Compliance by-design

auf unterschiedlichen Ebenen

# Dialogbasierte Agenten



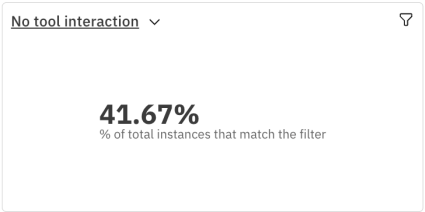
# Dialogbasierte Agenten




- Interaktivität durch low-level Framework gegeben
- Dialog ist nativer Bestandteil des Agentensystems

- BPMN nicht nativ dialog-oriented
- Interaktion mit Agenten muss modelliert werden (→ Tool)

Prompting wichtig, aber
gleichzeitig nicht zuverlässig



# Key Takeaways

- **LangGraph**: maximale Flexibilität bei der Agentenentwicklung.
- **Camunda 8**: strukturierte Orchestrierung und Governance für KI-Agenten in produktiven Unternehmensprozessen.

➔ Observability, Monitoring und Betrieb entscheidend für zuverlässige Integration von KI-Agenten in Geschäftsprozessen!

Q&A

# Danke fürs Teilnehmen!
# Fragen?

Mario Micudaj
Mario.Micudaj@viadee.de

**viadee** Unternehmensberatung AG
Anton-Bruchausen-Str. 8
48147 Münster
**Tel: +49 251 77777 138**
**www.viadee.de**

viadee

entdecken entwickeln entfalten

viadee Unternehmensberatung AG
viadee.de