

PBUGG · OKTOBER 2026

Lokale LLMs für PowerBuilder

Von der Theorie zur Praxis

Ludwin Feiten · Power People IT Services GmbH

EINSTIEG

Warum eine lokale KI für unseren Code?

Die Ausgangsfrage: Kann eine KI auf eigener Hardware unseren PowerBuilder-Code verstehen?

Der Code bleibt im Haus

Quellcode und Kommentare verlassen das eigene Netz nicht.
Keine Cloud, keine Weitergabe an Dritte.

Die KI kennt unsere Anwendung

Sie beantwortet Fragen zu SozialOffice mit echten Objekten, Events und SQL statt mit Allgemeinwissen.

Eigene Hardware, offene Software

Ein vorhandener Server mit einer Rechenkarte aus dem Rechenzentrum und ausschließlich Open-Source-Komponenten.

EINSTIEG

Dieselbe Frage, zwei Antworten

„Welches DataWindow zeigt die Stammdaten einer Station mit Name, Straße, Telefon, Fax und E-Mail, und mit welchen Argumenten wird es abgerufen?“

Am Anfang

Erfindet eine Tabelle stations und ein DataWindow dw_stations und schreibt ungefragt Code in C-Syntax mit geschweiften Klammern.

Heute

Nennt d_so_station aus stammdaten.pbl.src, abgerufen mit ras_mannr (string) und ran_stid (number), dazu das echte Retrieve-SQL.

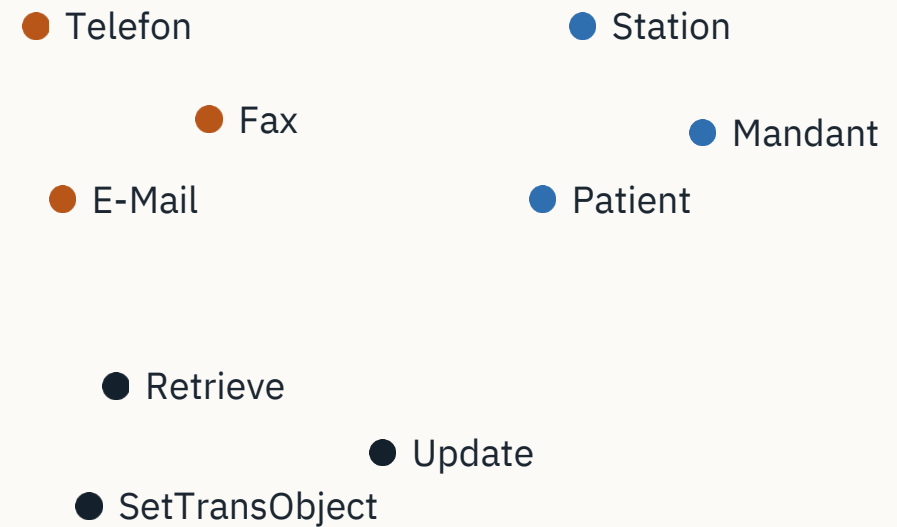
TEIL 2 · WIE FUNKTIONIERT EIN LLM?

Bedeutung wird zu einem Punkt im Raum

Ein Embedding-Modell übersetzt einen ganzen Text in eine Liste von Zahlen, einen Punkt im Bedeutungsraum. Bei bge-m3 sind es 1.024 Zahlen.

Texte mit ähnlicher Bedeutung liegen nah beieinander, auch wenn sie andere Wörter verwenden.

$$\begin{aligned}\text{Ähnlichkeit} &= \cos \varphi \\ &= (a \cdot b) / (|a| \cdot |b|)\end{aligned}$$



Vereinfacht auf 2 von 1.024 Dimensionen

TEIL 2 · WIE FUNKTIONIERT EIN LLM?

Ein LLM erzeugt plausiblen Text, keine geprüften Fakten.

ERFUNDEN

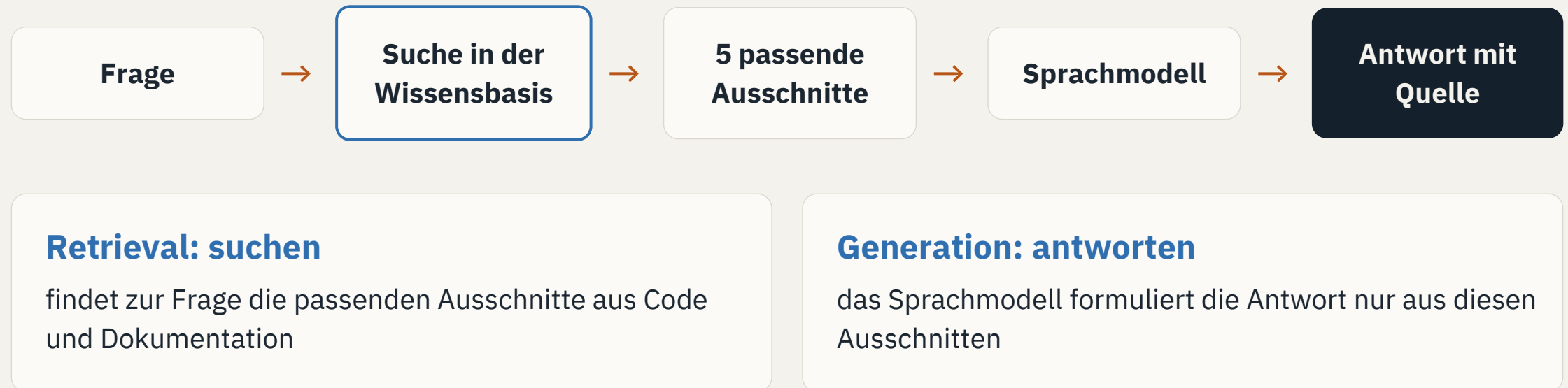
Eine Tabelle stations, dazu Code in C-Syntax

ERFUNDEN

Eine SAP-Anbindung über den Menü-Manager

Das Modell kennt unseren Code nicht. Also müssen wir ihm den richtigen Code zeigen.

RAG: erst nachschlagen, dann antworten



Die KI kennt unseren Code nicht. Sie liest ihn bei jeder Frage nach.

Deshalb entscheidet die Suche über die Qualität der Antwort.

Ein vorhandener Server, eine VM, eine Rechenkarte



Server

SuperMicro X10DAI mit Xeon-Prozessoren, Proxmox VE als Virtualisierung



VM für die KI

Ubuntu, 4 vCPUs, 15 GB RAM, 59 GB Platte, CPU-Typ host



Rechenkarte

NVIDIA Tesla P40 mit 24 GB Grafikspeicher, per PCI-Passthrough an die VM durchgereicht

Die P40 ist eine Rechenzentrumskarte ohne eigenen Lüfter. Das wird im Praxisbericht noch wichtig.

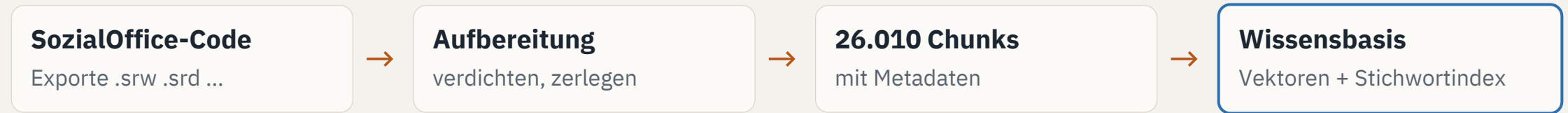
Ausschließlich Open-Source-Komponenten

Komponente	Aufgabe
Ollama (basiert auf llama.cpp)	lädt die Modelle und rechnet sie auf der GPU
qwen2.5-coder:14b	Sprachmodell für die Antworten
bge-m3	Embedding-Modell für die Vektor-Suche
ChromaDB	Vektordatenbank, eingebettet in die Anwendung
rank_bm25	Stichwortsuche
LangChain	verbindet Suche und Sprachmodell
Streamlit	Weboberfläche mit Chat, Code-Analyse und Wissensbasis
Python 3.14, NumPy 2.5.4	Laufzeitumgebung

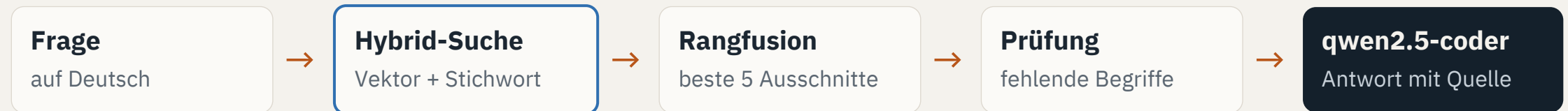
TEIL 4 · AUFBAU

So beantwortet die KI eine Frage

EINMALIG: INDIZIERUNG



BEI JEDER FRAGE



Die Wissensbasis: SozialOffice PB2022

2.384

Quelldateien

1.076

davon DataWindows

26.010

Chunks heute, anfangs
90.376

4

Architekturnotizen

Quelle sind die PowerBuilder-Exporte (.srw, .sru, .srd ...) aus dem Workspace, eingebunden über einen Netzwerk-Mount auf den Entwicklungsserver.

13 Schritte in fünf Kapiteln

1

Hardware

Die GPU fällt vom Bus

2

Daten aufbereiten

DataWindows
verdichten, Scripts
nach Events zerlegen

3

Suche

Embedding-Modell,
Hybrid-Suche,
Wortlücke

4

Modell und Prompt

Eigener Prompt,
Modellvergleich

5

Wissen aufschreiben

Architekturnotizen

Dazwischen: ein stiller Rechenfehler, der die Stichwortsuche lahmlegte, ohne eine einzige Fehlermeldung.

Die GPU fällt vom Bus

89 °C

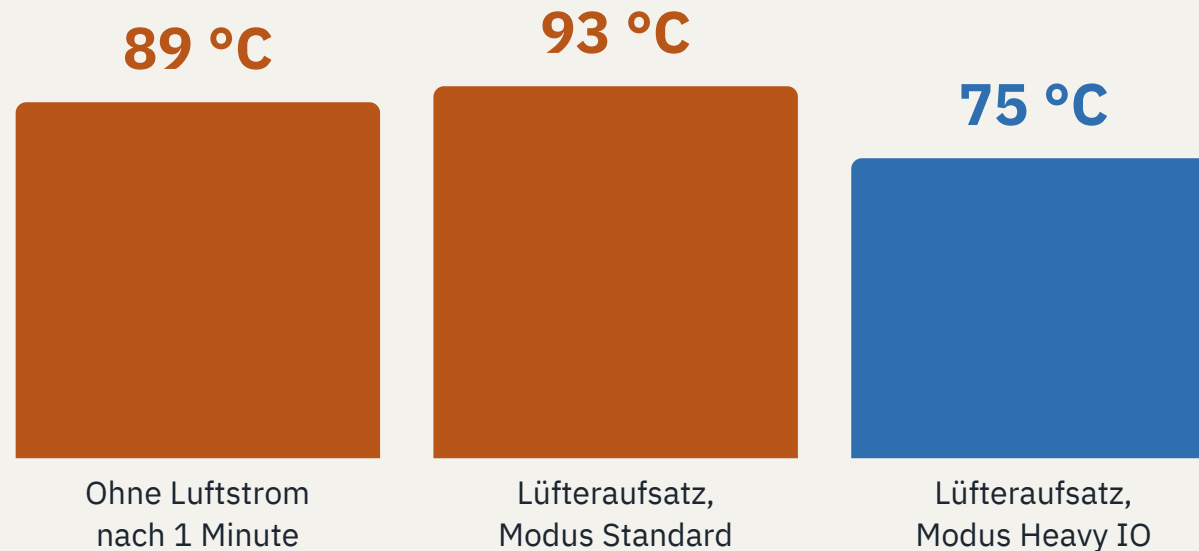
nach einer Minute Volllast ohne Luftstrom. Danach meldet der Treiber Xid 79: „GPU has fallen off the bus“, und die VM hängt.

Gegenmaßnahmen

- Provisorisch: ein Luftmatratzen-Gebläse auf die Karte
- Lüfteraufsatz für die P40 (nächste Folie)
- Power-Limit 125 W statt 250 W
- Temperaturwächter: Pause ab 80 °C, weiter ab 65 °C

Ergebnis: keine Abstürze mehr. Lange Läufe pausieren automatisch.

Der Lüfter war da, aber er folgte der falschen Temperatur



Standard: Takt auf 544 MHz gedrosselt, nur 95 W Leistung.

Heavy IO: 73 bis 75 °C über 10 Minuten, keine Hitze-Drosselung, Takt um 1.050 MHz.

Vermutete Ursache: Die Lüfteranschlüsse des Mainboards folgen der CPU- und Chipsatz-Temperatur, nicht der GPU.

Folge: Vektorisieren mit 9,3 Chunks pro Sekunde, stabil. Der Neuaufbau der Wissensbasis dauert hochgerechnet rund 47 Minuten statt 4,5 Stunden.

60 % der Wissensbasis waren Schriftarten und Farben

Objekt	vorher → nachher
d_so_station	20.424 → 1.341 Zeichen
d_dienstplan	1,64 MB → 45 KB
alle 1.076 DataWindows	54.506 → 3.898 Chunks

Ein .srd-Export besteht zu 90 bis 97 % aus x, y, Fonts, Farben und Tooltips. Diese Chunks verdrängten bei jeder Suche den eigentlichen Code.

Was der Verdichter behält

- Retrieve-SQL und Argumente
- Spalten mit Datentyp
- Update-Tabelle und Schlüssel
- Beschriftung → Spalte
- berechnete Felder
- DropDownDataWindows

Für Änderungen am Layout lädt man die Originaldatei nach; der verdichtete Text dient nur der Suche.

Scripts an Event-Grenzen zerlegen

```
PowerBuilder Fenster: w_so2_patientenstamm (erbt von w_pp_mmd)  
Kommentar: window zu Pflege der Patientenstammdaten ...  
Event: ue_printstammbblatt im Fenster (Teil 1 von 2)  
Beschreibung: öffnet die Druckvorschau für das Patientenstammbblatt
```

Vorher

Alle 1.000 Zeichen geschnitten. Ein Chunk mitten aus ue_daten_sichern wusste nicht, zu welchem Fenster und Event er gehört.

Nachher

Jeder Chunk trägt Objekt, Vorfahr, Control, Event und die Beschreibung aus dem Kopfkomentar.

Das Embedding-Modell muss Deutsch verstehen

Testfrage: „Stammdaten der Station: Name, Strasse, Telefon, Fax, Email“

Embedding-Modell	Rang d_so_station	Ähnlichkeit d_so_station	besten Fehltreffer
nomic-embed-text	6 von 6	0,610	0,854
nomic mit Präfixen	6 von 6	0,609	0,846
bge-m3	1 von 6	0,597	0,482

Der Wechsel bedeutete einen Neuaufbau aller 39.768 Vektoren: 4,5 Stunden, überwiegend Abkühlpausen der GPU.

Zwei Bibliothekare: Vektor- und Stichwortsuche



Vektor-Suche

Versteht Inhalte und findet Umschreibungen. Lässt sich von Nebenwörtern ablenken.



Stichwortsuche (BM25)

Findet exakte Begriffe wie Fax oder d_so_station. Versteht keine Umschreibungen.



Hybrid-Suche

Fragt beide und kombiniert die Ranglisten. Die Stichwortsuche zählt doppelt.

Bei der langen Stationsfrage lenkte das Wort „Argumente“ die Vektor-Suche zu d_so_stationsparameter. Die Hybrid-Suche bringt d_so_station unter die ersten fünf.

„Abstand 0,728“: die Vektor-Suche

$$d = 2 - 2 \cdot \cos \varphi \Rightarrow \cos \varphi = 1 - d / 2$$

0 bis 2

0 = gleiche Bedeutung, 2 = Gegenteil.
Kleiner ist besser.

0,728 → $\cos \varphi \approx 0,64$

entspricht einem Winkel von etwa
50°

Probe: 0,807 → 0,597

d_so_station: genau die Ähnlichkeit
aus dem Modelltest

„BM25 38,0“: die Stichwortsuche

$$\text{Score} = \frac{\sum \text{IDF}(w) \cdot f \cdot (k_1 + 1)}{(f + k_1 \cdot (1 - b + b \cdot |D| / \emptyset|D|))}$$

f – wie oft das Wort im Abschnitt steht, mit Sättigung

IDF – wie selten das Wort im gesamten Code ist

|D| – Länge des Abschnitts: kurze, treffende gewinnen

Wort	IDF
email	6,08
station	5,30
fax	5,05
telefon	4,93
sqlca	2,95
name	1,67

Echte Werte aus SozialOffice

$k_1 = 1,5$ (Sättigung) · $b = 0,75$ (Längenausgleich) · der Score hat keine Obergrenze

„Vektor: Platz 13 · Stichwort: Platz 8“

$$\text{Punkte} = \sum \text{Gewicht} / (60 + \text{Platz})$$

Fall	Rechnung	Punkte
Vektor Platz 13 + Stichwort Platz 8	$1/73 + 2/68$	0,0431
nur Vektor-Suche, Platz 1	$1/61$	0,0164
nur Stichwortsuche, Platz 1	$2/61$	0,0328
beide auf Platz 1	$1/61 + 2/61$	0,0492

Wer in beiden Listen solide abschneidet, schlägt den, der nur in einer glänzt.

Die Wortlücke zwischen Frage und Code

Die Frage sagt	Der Code sagt	Lösung
speichern	sichern, Update, save	Synonymliste
Stammblatt drucken	ue_printstammblatt, Druckvorschau	Wortteile und Synonyme
Patientenstamm	w_so2_patientenstamm	Objekt-Fokus auf diese Datei
Wo wird ... geöffnet?	mf_Opensheet("w_so2_patientenstamm") im Menü	Objekt selbst ausschließen

Die Synonymliste ist eine kleine Datei, die man von Hand erweitert, auch um Fachbereiche: Tourenplanung → sotp, Begutachtung → nba.

ZWISCHENFALL · DER STILLE FEHLER

40 Treffer außerhalb der Funktion, 0 innerhalb

```
def test():  
    a = np.arange(100_000.0)  
    np.argsort(-a) # darf a nicht ändern  
    return a[:5]
```

NumPy 1.26.4

`[-0. -1. -2. -3. -4.]`

NumPy 2.5.4

`[ 0. 1. 2. 3. 4.]`

Eine Speicheroptimierung von NumPy 1.26 passt nicht zu Python 3.14 und überschreibt das Array. Keine Fehlermeldung.
Lösung: CPU-Typ host in Proxmox, danach NumPy 2.5.4.

Nicht jedes Modell hält sich an Anweisungen

Testfrage: „Wie ist die Anbindung an SAP umgesetzt?“ – SAP steht im ganzen Code nur in einem Link in einem Kommentar.

Modell	Hinweis im System-Prompt	Hinweis in der Frage	Stations-Frage
deepseek-coder-v2:16b	erfindet eine Anbindung	ehrlich	richtig, ausschweifend
qwen2.5-coder:14b	ehrlich	ehrlich	richtig, kompakt

Unser Chat-Prompt

Nur aus dem Kontext antworten · Objekt und Datei nennen · kein ungefragter Code · gültiges PowerScript · Fehlendes ehrlich sagen

Vier Textdateien lösen, woran die Technik scheiterte

Fenster öffnen

Menü → mf_opensheet →
pfc_open in w_pp_frame →
OpenSheet

Hilfsfenster

Suchen über
w_pp_hilfsfenster, Rückgabe
in Message.StringParm

Drucken

Druckvorschau mit
of_setprintpreview und
Druckoptionen

Namenskonventionen

Präfixe und Module: so2, pp,
sopv, sotp, nba ...

Aufbau jeder Notiz: Titel · Ablauf · beteiligte Objekte · typische Fragen. Eine eigene Dokumentations-Spur in der Suche sorgt dafür, dass passende Notizen nicht im Code untergehen.

Von 0 auf 6 richtige Antworten

Frage zum Patientenstamm	Start	Scripts zerlegt	Suche erweitert	Korrekturen	Notizen + Prompt
Vererbung	falsch	richtig	richtig	richtig	richtig
Speichern	falsch	falsch	richtig	richtig	richtig
Pflichtfelder	teilweise	richtig	falsch	teilweise	richtig
Arzt suchen	falsch	teilweise	richtig	richtig	richtig
Stammblatt drucken	falsch	falsch	falsch	richtig	richtig
Fenster öffnen	teilweise	teilweise	falsch	teilweise	richtig

Sechs Fragen mit bekannter Musterlösung. „Falsch“ hieß hier immer: ehrlich „nichts gefunden“, nie erfunden.

TEIL 6

Live-Demo

1. Suchvergleich „alle drei“ mit der langen Stationsfrage
2. „Was passiert beim Speichern im Patientenstamm?“
3. SAP-Frage: deepseek-coder-v2 gegen qwen2.5-coder
4. „Was bedeutet Invo_parm in SozialOffice?“

Weboberfläche auf der eigenen VM · keine Verbindung ins Internet nötig

LIVE-DEMO · RESERVE 1

Suchvergleich: Vektor, Stichwort, Hybrid

Systemstatus

Status prüfen

Wissensbasis

Gespeicherte Chunks
26010

Dateien indizieren

Modell:
`qwen2.5-coder:14b`

Embeddings:
`bge-m3`

Code-Analyse

Code-Generierung

Chat-Assistent

Wissensbasis

Wissensbasis verwalten

Füge PowerBuilder-Dokumentation, Code-Beispiele und Best Practices hinzu

Quellenübersicht

Hinzufügen

Durchsuchen

Wissensbasis durchsuchen ↻

Testet, ob relevante Inhalte gefunden werden, und vergleicht Vektor-, Stichwort- und Hybrid-Suche.

Suchbegriff

„Welches DataWindow zeigt die Stammdaten einer Station mit Name, Straße, Telefon, Fax und E-Mail, und mit welchen Argumenten wird es abgerufen?“

Suchverfahren

Anzahl Ergebnisse

☐ Hybrid☒ Nur Vektor☐ Nur Stichwort☒ Vergleich aller drei

Suchen

Vektor-Suche

Stichwort-Suche

Hybrid-Suche

Abstand: kleiner = inhaltlich ähnlicher

BM25-Score: höher = mehr und seltenere passende Begriffe

Kombinierte Rangliste, Stichwortsuche doppelt gewichtet

> 1_d_so_stationsparameter.srd (Abstand 0.707)

> 2_d_so_user_parameter.srd (Abstand 0.726)

> 3_d_ddd_station.srd (Abstand 0.731)

> 4_d_patientendaten_serienbrief.srd (Abstand 0.738)

> 5_d_patientendaten_serienbrief.srd (Abstand 0.744)

> 6_d_ddd_station_erlaubt.srd (Abstand 0.744)

> 7_d_dddstationen_auswahl.srd (Abstand 0.746)

> 8_d_sopp_auslaufende_pflegeplaene.srd (Abstand 0.749)

> 1_d_so_station.srd (BM25 96.0)

> 2_d_so_stationsliste_mandant.srd (BM25 89.7)

> 3_d_so_stationsliste.srd (BM25 88.9)

> 4_so2_mandanten_stammdaten.srd (BM25 88.5)

> 5_d_ddd_station.srd (BM25 85.8)

> 6_d_ddd_station_mandant.srd (BM25 85.0)

> 7_d_ddd_station_erlaubt.srd (BM25 83.1)

> 8_d_dddstationen_auswahl.srd (BM25 76.0)

> 1_d_ddd_station.srd (Vektor: Platz 3 · Stichwort: Platz 5)

> 2_d_ddd_station_erlaubt.srd (Vektor: Platz 6 · Stichwort: Platz 7)

> 3_d_so_station.srd (Vektor: Platz 23 · Stichwort: Platz 1)

> 4_d_dddstationen_auswahl.srd (Vektor: Platz 7 · Stichwort: Platz 8)

> 5_d_ddd_station_mandant.srd (Vektor: Platz 17 · Stichwort: Platz 6)

> 6_so2_datev2018_debitoren_patienten.srd (Vektor: Platz 22 · Stichwort: Platz 10)

> 7_so2_patienten_stammdatenpflegegrad.srd (Vektor: Platz 9 · Stichwort: Platz 19)

> 8_so2_datev2018_debitoren_datan.srd (Vektor: Platz 25 · Stichwort: Platz 12)

d_so_station: Vektor Platz 23 · Stichwort Platz 1 (BM25 96,0) · Hybrid Platz 3

„Was passiert beim Speichern im Patientenstamm?“

Code-Analyse Code-Generierung **Chat-Assistent** Wissensbasis

PowerBuilder Chat-Assistent

Stelle Fragen zu PowerBuilder, lass Code erklären oder erstellen

Was passiert beim Speichern im Patientenstamm?

Beim Speichern im Patientenstamm werden die Daten in der Reihenfolge gesichert, um die Parent-Child-Beziehungen zu respektieren. Wenn eine neue Zeile geändert wurde, werden die Daten in der richtigen Reihenfolge gespeichert. Nach dem Speichern des Hauptfensters werden die Kinder mit der neuen Primärschlüssel versorgt.

Kernpunkte:

- Die Daten werden in der richtigen Reihenfolge gesichert.
- Bei einer neuen Zeile werden die Daten in der richtigen Reihenfolge gespeichert.
- Nach dem Speichern des Hauptfensters werden die Kinder mit der neuen Primärschlüssel versorgt.

Objekte:

- `w_so2_patientenstamm` (Fenster) - Verantwortlich für das Speichern der Patientenstammdaten.
- `dw_arzt` , `dw_seelsorger` , `dw_kostentraeger` , `dw_abwesenheit` (DataWindows) - Diese DataWindows werden gespeichert, wenn sie geändert wurden.

Gefunden über die Synonymliste: „speichern“ heißt im Code „sichern“ (Event `ue_daten_sichern`)

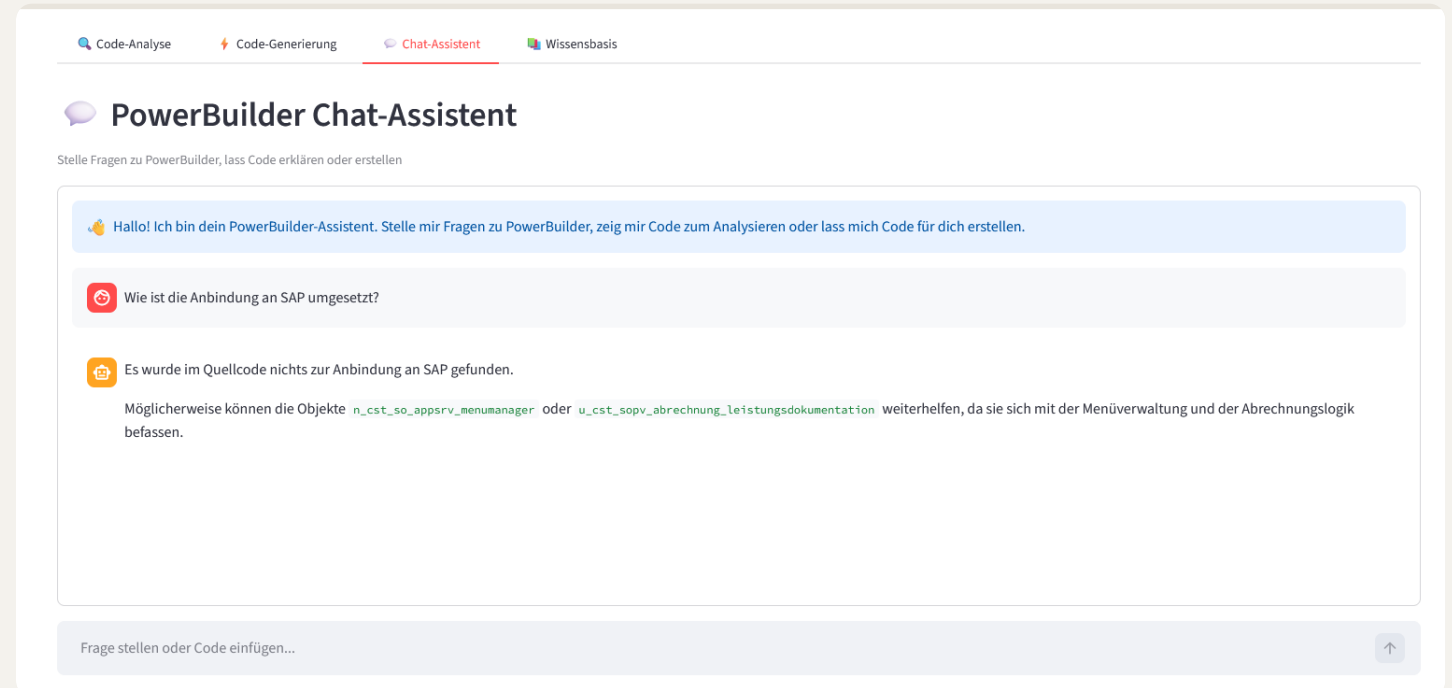
„Wie ist die Anbindung an SAP umgesetzt?“

DEEPSEEK-CODER-V2: ERFUNDEN

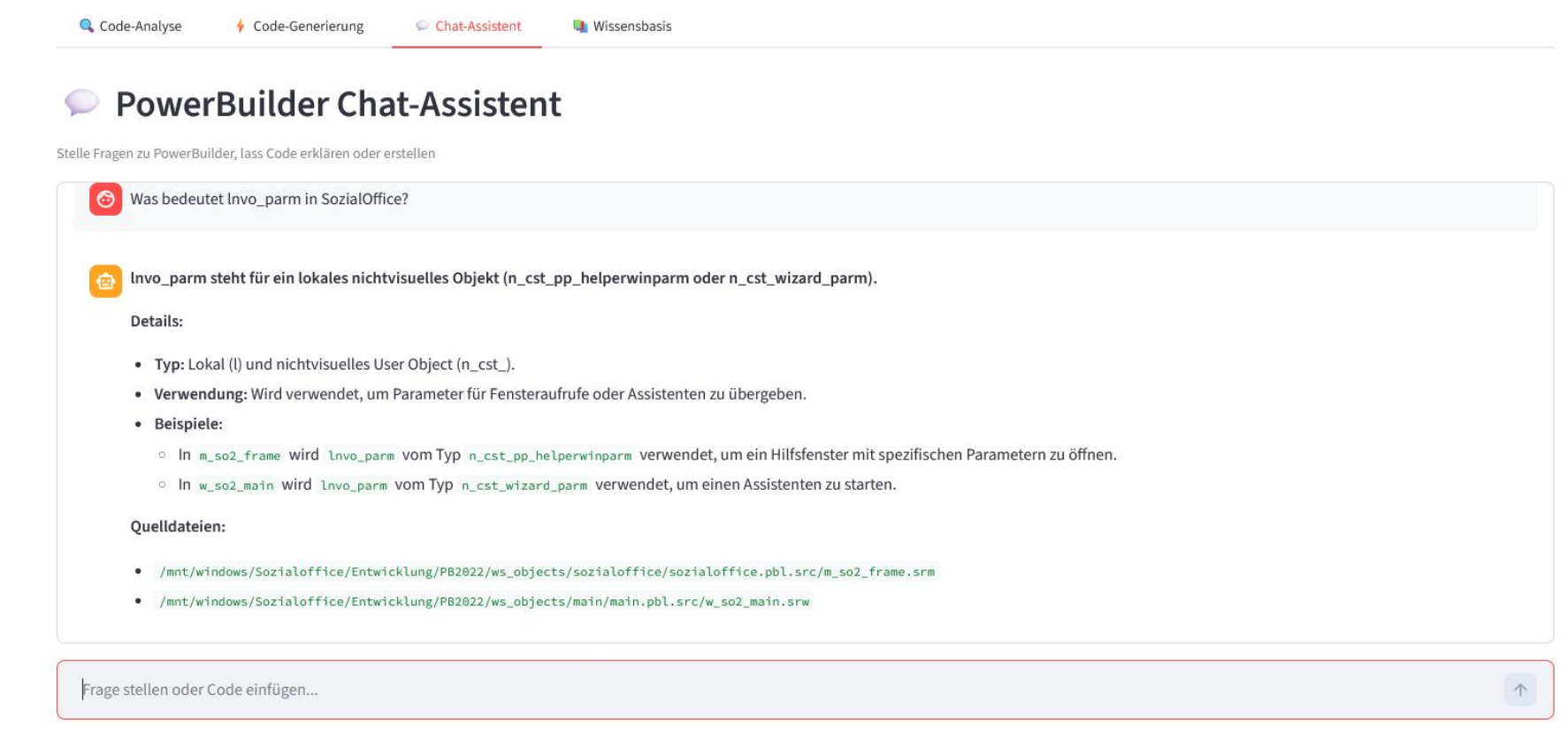
„Die Anbindung an SAP wird über das Userobject `n_cst_so_appsrv_menumanager` verwaltet, wobei die Methode `of_getmenucalls` für die Factory-Funktion zur Bereitstellung von Menucall-Parametern zuständig ist.“

Den Menü-Manager gibt es. Eine SAP-Anbindung nicht.

QWEN2.5-CODER: EHRlich



„Was bedeutet Invo_parm in SozialOffice?“



The screenshot shows the 'PowerBuilder Chat-Assistent' interface. At the top, there are four tabs: 'Code-Analyse', 'Code-Generierung', 'Chat-Assistent' (which is active), and 'Wissensbasis'. Below the tabs, the title 'PowerBuilder Chat-Assistent' is displayed with a speech bubble icon. A subtitle reads 'Stelle Fragen zu PowerBuilder, lass Code erklären oder erstellen'. The main chat area shows a user question: 'Was bedeutet Invo_parm in SozialOffice?'. The assistant's response explains that 'Invo_parm' is a local non-visual object (n_cst_pp_helperwinparm or n_cst_wizard_parm). It provides details on its type, usage, and examples. The examples show 'Invo_parm' being used in 'm_so2_frame.srm' and 'w_so2_main.srw' to open a help window and start an assistant, respectively. Source files are listed at the bottom of the response. A text input field at the bottom is labeled 'Frage stellen oder Code einfügen...' with an upward arrow button.

Code-Analyse Code-Generierung Chat-Assistent Wissensbasis

PowerBuilder Chat-Assistent

Stelle Fragen zu PowerBuilder, lass Code erklären oder erstellen

Was bedeutet Invo_parm in SozialOffice?

Invo_parm steht für ein lokales nichtvisuelles Objekt (n_cst_pp_helperwinparm oder n_cst_wizard_parm).

Details:

- **Typ:** Lokal (l) und nichtvisuelles User Object (n_cst_).
- **Verwendung:** Wird verwendet, um Parameter für Fensteraufrufe oder Assistenten zu übergeben.
- **Beispiele:**
 - In `m_so2_frame` wird `Invo_parm` vom Typ `n_cst_pp_helperwinparm` verwendet, um ein Hilfsfenster mit spezifischen Parametern zu öffnen.
 - In `w_so2_main` wird `Invo_parm` vom Typ `n_cst_wizard_parm` verwendet, um einen Assistenten zu starten.

Quelldateien:

- `/mnt/windows/Sozialoffice/Entwicklung/PB2022/ws_objects/sozialoffice/sozialoffice.pbl.src/m_so2_frame.srm`
- `/mnt/windows/Sozialoffice/Entwicklung/PB2022/ws_objects/main/main.pbl.src/w_so2_main.srw`

Frage stellen oder Code einfügen...

Beantwortet aus der Architekturnotiz zu den Namenskonventionen, ergänzt um Beispiele aus dem Code

Was wir gelernt haben

Erst messen, dann debuggen

Temperatur unter Last prüfen, bevor man Software verdächtigt.

Daten vor Modell

Aufbereitung bringt mehr als ein größeres Sprachmodell.

Deutsch braucht das passende Embedding

bge-m3 statt eines englisch trainierten Modells.

Code braucht exakte Begriffe

Hybrid-Suche statt reiner Vektor-Suche.

Mit festen Testfragen arbeiten

Jede Verbesserung kann eine andere Antwort verschlechtern.

Wissen aufschreiben

Was nur in Köpfen steckt, kann keine KI finden.

Was man zum Nachbauen braucht

Hardware

Ein Server mit genug RAM und eine GPU mit ausreichend Grafikspeicher (bei uns 24 GB). Und Luftstrom!

Software

Ollama, ChromaDB, rank_bm25, LangChain und Streamlit: alles Open Source.

Daten

Die PowerBuilder-Exporte (.srw, .sru, .srd ...), aufbereitet, plus einige Architekturnotizen.

Methode

Testfragen mit Musterlösung von Anfang an, und jede Änderung damit nachprüfen.

TEIL 8

Fragen und Diskussion

Ludwin Feiten · Power People IT Services GmbH · Vorstand PBUGG e.V.