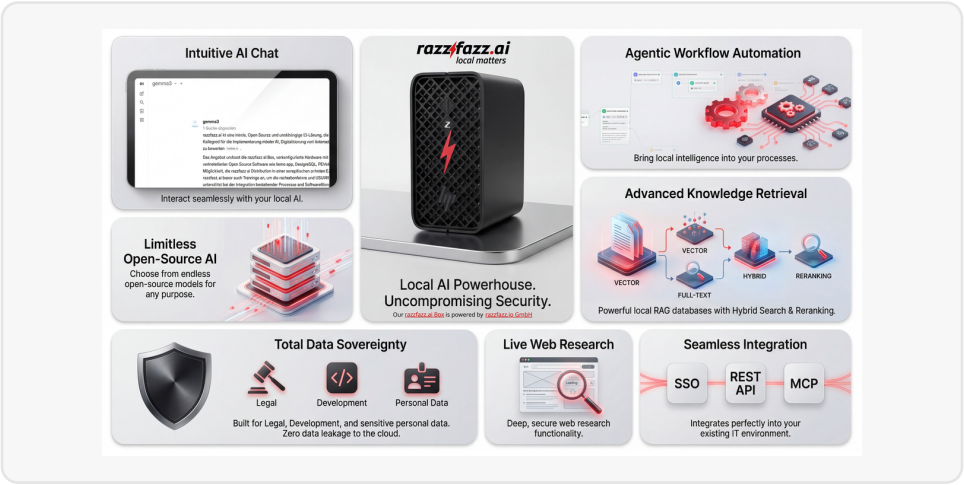
Die Box im Ueberblick



AGENDA

10 Things -- Die Roadmap

1. Die razzfazz.ai Box & Architektur
2. LLM-Grundlagen & GPUStack
3. Chat-Oberflaeche -- Open WebUI
4. Wissen & RAG
5. Skills, Tools & Webrecherche
6. Agentic Workflows mit Dify
7. Agentic Coding -- Planen & Entwickeln
8. TDD & Code Review
9. Test Automation mit Playwright
10. Integration

Die razzfazz.ai Box & Architektur

Was steckt drin -- und warum lokal?

Was ist die razzfazz.ai Box?

AMD Strix Halo SOC

Kompakter Mini-PC mit integrierter GPU

Keine separate Grafikkarte noetig

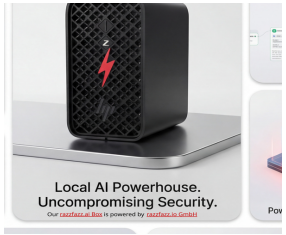
Lautlos & energieeffizient

128 GB Unified Memory

96 GB fuer LLMs als GPU Memory

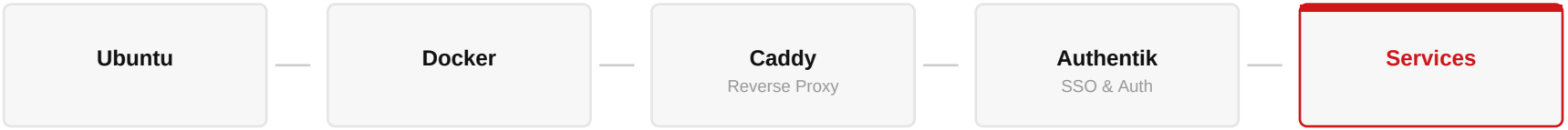
32 GB reserviert fuer OS + Container

Modelle bis 80B Parameter lokal moeglich



Die razzfazz.ai Distribution

Fully containerized: Ubuntu -> Docker/Compose -> alle Services



Beispiel: Gitea als internes Git-Repository hinter Caddy + Authentik

SSO wo moeglich -- nicht durchgaengig, da jede Open-Source-App eigene Einschraenkungen hat

Warum lokal? -- Abhaengigkeit & Geopolitik

Risiken der Cloud-Abhaengigkeit

API-Pricing kann sich jederzeit aendern

Daten verlassen Europa

AIs als maechtige Werkzeuge

Sanktionen - Regulierung - Abschaltung

Lokale AI macht unabhaengig

Technologisch souveraen

Wirtschaftlich planbar

Geopolitisch resilient

Open Source von Grund auf

Skalierbar & flexibel

Einzelplatz
1 Box

Cluster
Master + Worker

Hybrid
Cloud + Lokal

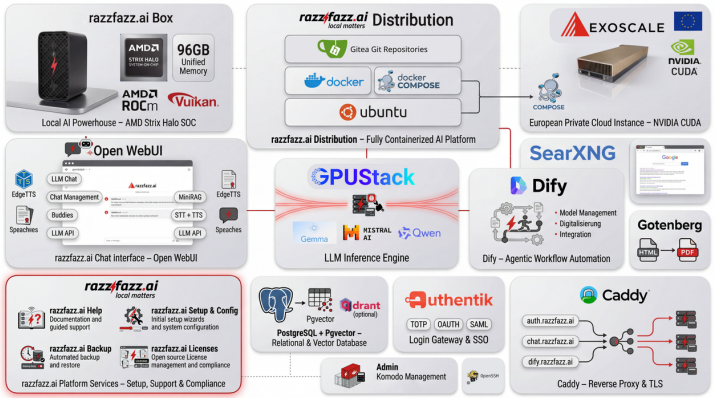
GPUStack ermoeglicht Zusammenschluss mehrerer Boxen (Master/Worker)
Cloud-Instanzen als Worker - Distribution auch auf normalen Servern

Auch in der Cloud

Exoscale European Private Cloud

Gleiche Distribution, anderer Host -- Daten bleiben in Europa

Schweizer Cloud-Anbieter
DSGVO-konform - ISO 27001
GPU-Instanzen verfuegbar



1

Volle Kontrolle, keine Abhaengigkeit

Kompakte Hardware mit Enterprise-Distribution

Unabhaengig von US/CN-Anbietern -- technologisch, wirtschaftlich, geopolitisch

Skalierbar vom Einzelplatz bis zum Cluster (Master/Worker via GPUStack)

Laeuft lokal, auf Servern oder in der European Cloud

Open Source von Grund auf

LLM-Grundlagen & GPUStack

Das Herzstueck der razzfazz.ai Box

Was ist ein LLM?

Transformer-Architektur vereinfacht:
Text zerlegen -> Zusammenhaenge erkennen -> Naechstes Wort vorhersagen

"Das Modell hat kein Wissen --
es hat Wahrscheinlichkeiten."

LLM -- Large Language Model

Ein KI-Modell, das natuerliche Sprache versteht und generiert.

Token

Kleinste Texteinheit fuer ein LLM -- ein Wort, Wortteil oder Satzzeichen.

Was ist Quantisierung?

128 GB total, davon 96 GB GPU Memory fuer LLMs (32 GB fuer OS/Container reserviert)
Trade-off: Praezision vs. Geschwindigkeit vs. Speicher

Quantisierung
Komprimierung eines Modells durch reduzierte Zahlenpraezision (z.B. 32-Bit -> 4-Bit) -- macht grosse Modelle auf kleiner Hardware moeglich.

Kontextfenster erklaert

256k Tokens

ca. 500 Seiten Text

"Ein ganzes Buch passt in eine Anfrage"

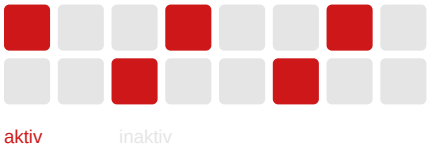
Ganze Codebasen - Komplette Dokumentensammlungen - Lange Konversationen

Warum Mixture of Experts (MoE)?

Viele Experten, nur wenige aktiv pro Token.

Perfekt fuer lokale Hardware:

- Grosses Modell, kleiner Rechenaufwand
- Schnellere Inferenz
- Weniger GPU-Auslastung pro Token



MoE -- Mixture of Experts

Architektur mit vielen Spezialisten, von denen nur wenige pro Anfrage aktiv werden.

Cloud-LLMs vs. Lokale LLMs

Cloud-LLMs

ChatGPT - Claude - Gemini

Fuer allgemeine Aufgaben

State-of-the-Art Reasoning

Lokale LLMs

Businesskritisch & spezialisiert

Sensible Daten bleiben intern

Volle Kontrolle & Planbarkeit

Spezialisierte Modelle: qwen3-coder-next (Coding, 80B MoE) - medgemma (Medizin) - SaulLM (Recht) - finance-LLM

Conclusio: Lokale LLMs ergaenzen Cloud-LLMs fuer businesskritische, spezialisierte Vorgaenge

Unsere Protagonisten

gemma-4-26b-it-a4b

Google - Chat & Allrounder

25,2 Mrd. Parameter total ("26B")

~3,8 Mrd. aktiv pro Token ("A4B")

128 Experten + 1 Shared, 8 aktiv/Token

Q8-Quantisierung - 27,15 GB

Qwen3-Coder-Next

Alibaba - Coding-Agent

80 Mrd. Parameter total

3 Mrd. aktiv pro Token

512 Experten + 1 Shared, 10 aktiv/Token

Q4-Quantisierung - 45,09 GB (4 Teile)

gemma4 bei Q8 = 27 GB - Qwen3-Coder bei Q4 = 45 GB -> zeigt den Unterschied 26B vs. 80B Parameter

Embedding & Reranking erklaert

Embedding-Modell

Text -> Zahlenvektor

Aehnliche Texte = nahe Vektoren

nomic-embed-text

Reranker

Ergebnisse nach Relevanz umsortieren

Wie ein Experte: Beste Treffer oben

qwen3-reranker

Beides zusammen macht RAG erst richtig gut

RAG -- Retrieval Augmented Generation

Das LLM sucht zuerst in einer Wissensdatenbank, bevor es antwortet.

Embedding / Reranking

Text wird in Zahlenvektoren umgewandelt; Suchergebnisse werden nach Relevanz neu sortiert.

Screen-Video: GPUStack Playground

Embedding live - Reranking live - Backend-Parameter gemma4 & Qwen3-Coder - Chat-Inferenz

~3 Min

GPUStack + llama.cpp

gemma-4-26b

ctx-size = 262144 (256k)
parallel = 2
flash-attn = on
cache-type-k/v = q8_0
temp=1.0 - top-p=0.95 - top-k=64

Qwen3-Coder-Next

ctx-size = 262144 (256k)
parallel = 1
flash-attn = on
cache-type-k/v = q8_0
temp=1.0 - top-p=0.95 - top-k=40

Tipp: llama.cpp teilt das Kontextfenster auf alle parallelen Slots auf -- parallel=2 -> jeder Slot bekommt nur 128k statt 256k.

GPUStack + llama.cpp

GPUStack -- Management

Modelle laden & verwalten

OpenAI-kompatible REST API

Monitoring & Multi-Worker-Support

llama.cpp -- Inference Engine

Direkt in GPUStack integriert

Optimiert fuer lokale Hardware (CPU+GPU)

GGUF-Format - Flash Attention - KV-Cache

Inferenz

Der Vorgang, bei dem ein trainiertes KI-Modell aus einer Eingabe eine Antwort erzeugt.

2

Die richtigen Modelle, richtig optimiert

Quantisierung macht grosse Modelle lokal moeglich

MoE-Architektur ist der Game-Changer fuer lokale AI

Embedding + Reranking machen RAG erst richtig gut

GPUStack + llama.cpp = zuverlaessige Inference-Engine

Chat-Oberflaeche -- Open WebUI

Mehr als nur ein Chat-Interface

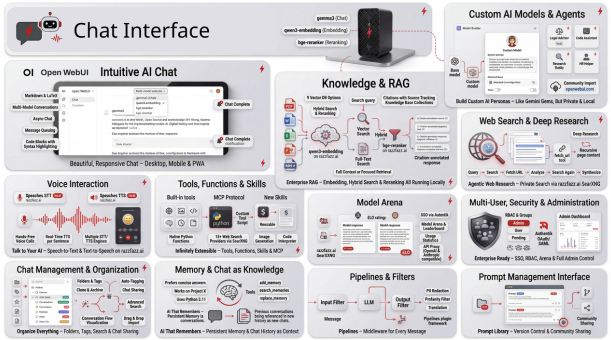
Open WebUI -- Mehr als Chat

Multi-Model
Verschiedene LLMs in einem Interface

Markdown & Code
Rich-Text & Code Blocks

Voice (STT/TTS)
Speaches -- lokal, Hands-Free

Async Chat
Parallele Konversationen



Custom Models / "Buddies"

Base Model
gemma4

+

Systemprompt
Rolle & Regeln

+

Wissen
Knowledge Base

+

Tools
Skills & APIs

=

Buddy
Spezialisiert

Wie ChatGPT GPTs und Gemini Gems -- aber privat & lokal

Screen-Video: Angebots-Buddy erstellen

Chat -> Systemprompt generieren lassen -> Model Builder -> gemma4 als Base -> Name & Avatar

~2,5 Min

3

Ein privater AI-Assistent, eigene Regeln

Open WebUI als intuitive Chat-Oberflaeche

Custom Models/Buddies fuer jeden Use Case

Lokal, privat, ohne Datenverlust

Wissen & RAG

Enterprise-grade, komplett lokal

Knowledge & RAG

PGVector
Vector-DB vorkonfiguriert & ready-to-use
Kein Setup noetig -- Teil der Distribution

Hybrid Search
Vector + Full-Text + Reranking
nomic-embed-text + qwen3-reranker

100% Lokal
Embedding + Reranking + Suche
Alles auf der Box

RAG -- Retrieval Augmented Generation
Das LLM sucht zuerst in einer Wissensdatenbank, bevor es antwortet.

Volltextsuche & Foldermanagement

Folder-Struktur mit Tags & Kategorien
Drag & Drop Upload
Advanced Search -- semantisch + Volltext
Quellennachweis in jeder Antwort
Conversation Flow Visualization

Screen-Video: Wissen fuer den Angebots-Buddy

Knowledge Base anlegen -> 7 Demo-PDFs hochladen -> Ordner organisieren
-> Buddy verknuepfen -> Testfragen -> Quellennachweis

~2,5 Min

4

Enterprise RAG -- komplett lokal

Hybrid Search mit Vector + Full-Text + Reranking

Quellennachweis in jeder Antwort

Alles auf der Box, keine Cloud noetig

Skills, Tools & Webrecherche

Den Buddy zum Spezialisten machen

Tools, Functions & Skills

Built-in Tools

Python Interpreter
Web Search - Image Gen
Code Interpreter

MCP Protocol

Offener Standard
Externe Tools & Datenquellen
Bidirektional

Custom Skills

Eigene Workflows definieren
Input/Output strukturiert
Wiederverwendbar

MCP -- Model Context Protocol

Offener Standard, ueber den AI-Modelle externe Tools und Datenquellen nutzen koennen.

Webrecherche via SearXNG

Private Metasuchmaschine

15+ Suchprovider - Deep Research mit fetch_url - Kein Tracking

Lokal gehostet auf der razzfazz.ai Box

Interne Daten bleiben intern -- nur die Websuche geht raus

Screen-Video: Skill & Webrecherche

Custom Skill "Angebots-Entwurf Generator" - SearXNG aktivieren
Kombinierter Test: internes Wissen + Marktrecherche

~2,5 Min

5

Offen fuer alles

Custom Skills machen den Buddy zum Spezialisten

Private Webrecherche ohne Tracking via SearXNG

Internes Wissen + aktuelle Marktdaten kombiniert

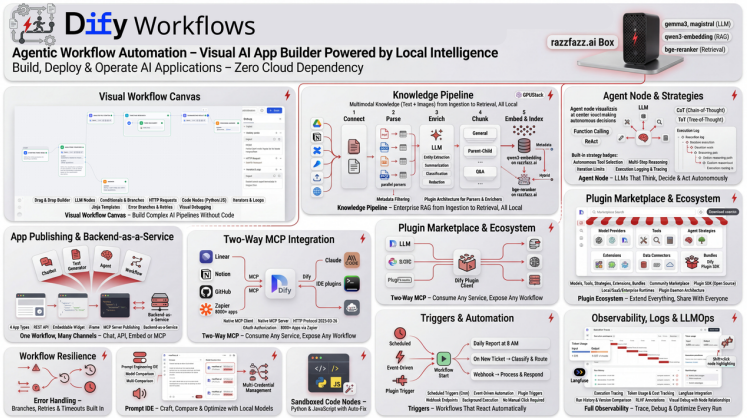
Privacy by Design: Sensible Daten verlassen nie die Box

Agentic Workflows mit Dify

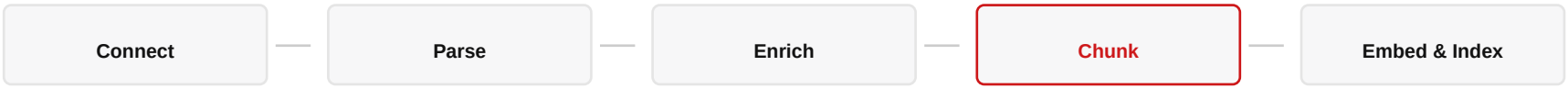
Visual AI App Builder -- lokal betrieben

Dify -- Visual AI App Builder

Drag & Drop Workflow Canvas - LLM Nodes - Conditionals - Code Nodes



Knowledge Pipeline & Parent/Child Chunking



Parent/Child Chunking: Grosse Abschnitte als Parent, Details als Child.

Wenn ein Detail-Chunk gefunden wird, wird der uebergeordnete Parent-Chunk mit ausgeliefert.

Wichtig bei regulatorischen Texten -- der Kontext ist alles.

Screen-Video: TARIC Buddy in Dify

Workflow Canvas - Parent/Child Chunking
Konkrete Abfrage: Zolltarifnummer ermitteln

~2,5 Min

MCP & Integration

Dify als MCP Client

Linear - Notion - GitHub

Externe Systeme einbinden

Dify als MCP Server

IDE Plugins - Claude Code

Andere Systeme nutzen Dify

6

KI-Workflows ohne Code

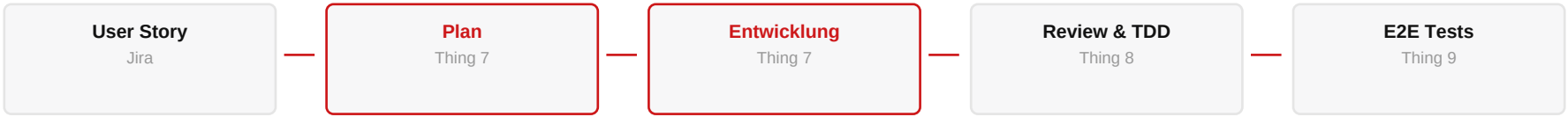
Visueller Workflow-Builder powered by lokale Models

Enterprise Knowledge Pipeline mit Parent/Child Chunking

Reales Beispiel: TARIC-EU-Datenbank als intelligenter Chatbot

Agentic Coding -- Planen & Entwickeln

Vom Jira-Ticket zur fertigen Applikation



Use Case "Angebots-Rechner" -- durchgaengig ueber Things 7, 8 & 9

Kilo Code -- Der AI Coding Agent

VSCode Plugin
Open Source

500+ Models
Inkl. lokale

Agent Modes
Plan - Code - Debug - Review

Parallel Subagents
Mehrere Tasks gleichzeitig

opencode -- Agentic Coding im Terminal

Go-basierte CLI mit Bubble Tea TUI
75+ Provider -- inklusive lokale Modelle
Multi-Session - LSP Integration
Gleiche Power wie Kilo Code -- anderes Interface

Unser Setup

qwen3-coder-next

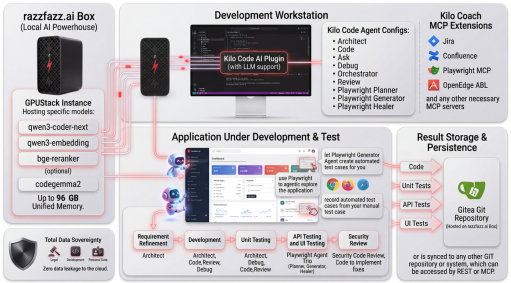
80B MoE - 3B aktiv

REST API

GPUStack

opencode

Kilo Code



Screen-Video: Von der User Story zum Plan

Jira-Import via MCP -> Plan-Modus: Rueckfragen, Analyse, strukturierter Plan

~2 Min (Teil 1 von 2)

Screen-Video: Umsetzung

Angebots-Rechner als HTML/JS/CSS - ~45 Tokens/Sek - Ergebnis im Browser
ggf. beschleunigt/geschnitten

~2 Min (Teil 2 von 2)

7

Vom Ticket zur App in Minuten

Jira-Integration via MCP: User Story direkt im Editor

Plan-Modus: AI fragt nach, plant strukturiert, dann setzt um

qwen3-coder-next als lokaler Coding-Spezialist -- kein Code verlaesst das Netzwerk

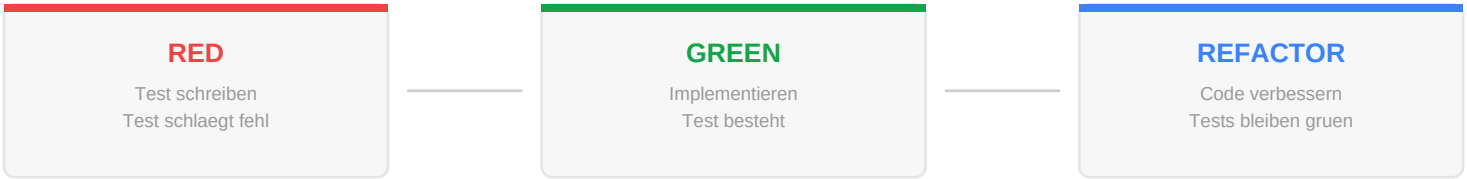
TDD & Code Review

Qualitaet eingebaut, nicht nachgetestet

Fortsetzung Use Case: Angebots-Rechner -- jetzt absichern & erweitern

TDD mit AI: Test First Development

AI schreibt zuerst den Test, dann die Implementierung.
Lokale AI + TDD = schnell iterieren ohne Datenabfluss



TDD -- Test Driven Development

Zuerst den Test schreiben, dann den Code. Sicherstellen dass alles funktioniert, bevor es fertig ist.

Screen-Video: Code Review

Kilo Code Review-Modus -> Angebots-Rechner analysieren
-> Issues finden (Rabatt >100%, negative Werte)

~1 Min

Screen-Video: TDD fuer Rabatt-Validierung

Unit Test schreiben -> RED (Fehlschlag) -> Validierung implementieren -> GREEN (bestanden)

~1,5 Min

Screen-Video: Refactoring mit opencode

"Extrahiere die Rabatt-Logik in eine eigene Funktion"
Refactoring im Terminal mit qwen3-coder-next

~30 Sek

8

Qualitaet eingebaut, nicht nachgetestet

Code Review durch AI findet Issues bevor sie in Produktion gehen

TDD mit AI: Tests first, Implementierung second

IDE oder Terminal -- beide Wege moeglich

Alles lokal mit qwen3-coder-next

Test Automation mit Playwright

Testen lassen statt Testen muessen

Fortsetzung Use Case: Angebots-Rechner -- jetzt End-to-End absichern

Agentic Testing

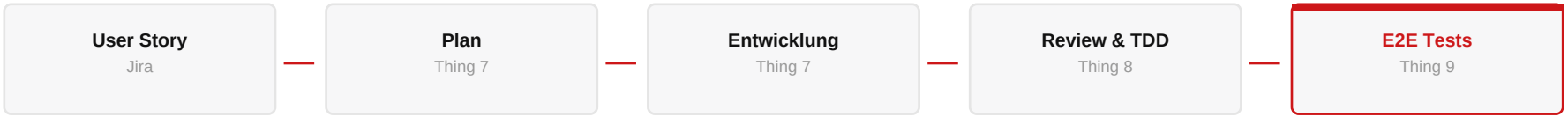
Planner
Plant die Teststrategie

Generator
Erstellt die Testfaelle

Healer
Repariert fehlschlagende Tests

E2E -- End-to-End Testing
Manuelle oder automatisierte Tests, die eine Anwendung so bedienen wie ein echter Benutzer.

Der vollstaendige Workflow



Jeder Schritt mit passendem Agent-Modus -- ein durchgaengiger, lokaler Workflow

Screen-Video: Playwright E2E Tests

App explorieren -> Testfaelle generieren -> Tests ausfuehren -> alle gruen

~2 Min

9

Testen lassen statt Testen muessen

Playwright-Agents generieren, heilen und pflegen E2E-Tests

Vom Jira-Ticket ueber Entwicklung bis zum getesteten Feature

qwen3-coder-next versteht Code UND Testlogik

Kein Token hat die Box verlassen

Integration

Maximale Anschlussfaehigkeit

API & Protokolle

REST API

OpenAI-kompatibel

Jede App, die OpenAI kann,
kann auch die razzfazz.ai Box

MCP Protocol

Offener Standard

AI-Modelle nutzen externe Tools

Bidirektional

Bi-direktionale Integration

Dify Workflows ueber Webhooks ansprechbar

Dify Workflows als MCP Tools einbindbar (IDE, Claude Code, andere Systeme)

Dify bindet externe Systeme ein (Linear, Notion, GitHub, Zapier)

Alle Supporting Container (Gotenberg, SearXNG, Speaches...) bieten eigene APIs

Externe Systeme koennen die LLMs auf der Box ueber die REST API ansprechen

Ergebnis: Massive bi-direktionale Integrationsfaehigkeit

Individuelle Integrationsloesungen

Das razzfazz.ai Team entwickelt gerne
individuelle Integrationsloesungen

Von der Anbindung an bestehende Systeme bis zu massgeschneiderten Workflows

10

Maximale Integrationsfaehigkeit

OpenAI-kompatible REST API + MCP Protocol

Bi-direktionale Integration: Webhooks, MCP, REST APIs aller Container

Das razzfazz.ai Team unterstuetzt bei individuellen Integrationsloesungen

ZUSAMMENFASSUNG

Alle 10 Things auf einen Blick

1. Box & Architektur -- Volle Kontrolle
2. LLM & GPUStack -- Richtig optimiert
3. Open WebUI -- Privater AI-Assistent
4. Wissen & RAG -- Enterprise-grade lokal
5. Skills & Webrecherche -- Erweiterbar
6. Dify -- KI-Workflows ohne Code
7. Agentic Coding -- Vom Ticket zur App
8. TDD & Review -- Qualitaet eingebaut
9. Playwright -- Testen lassen
10. Integration -- Maximal anschlussfaehig

AUSBLICK

Version 2026.05

Paperless-ngx + Tika + Docling

Intelligentes Dokumenten-Management -- Rechnungen, Vertraege, Belege automatisch erkennen & klassifizieren

Onyx - Enterprise Search

KI-gestuetzte Suche ueber alle internen Datenquellen -- Docs, Wikis, Tickets, Code

Vaultwarden + Infisical

Passwort-Management & Secrets-Management -- sensible Zugangsdaten sicher verwalten, lokal gehostet

OpenHands - Agentic Coding

Autonome Coding-Agents im Browser -- komplette Engineering-Tasks planen, umsetzen & testen

Paperclip - Hermes - Moltis

Multi-Agent-Orchestrierung -- AI-Agents die zusammenarbeiten, Ziele verfolgen & voneinander lernen

Matrix/Synapse + Element Web

Sichere Team-Kommunikation -- Ende-zu-Ende verschluesselt, selbst gehostet, AI-Agents als Chat-Teilnehmer

Alexander Vukovic

Fragen? Jetzt oder jederzeit per Mail.