Executive Summary

Gamechanger: Wir haben eine **eigene, lokal laufende TranceLate-Engine** aufgebaut (LLM + MT) und sie mit einem **Website-Vorscan** sowie einem **Gateway mit RAG** (Retrieval-Augmented Generation) verbunden. Ziel: **Enterprise-taugliche** Antworten schon beim **ersten Prompt**, ohne Prompt-Gymnastik beim Kunden.

Kernideen: - **Self-Hosting** → Datenhoheit, Konfigurationsmacht, keine Providerkosten. - **Zentrale API-Frontdoor** (OpenAI-kompatibel) → Auth, Policies, Quoten, Billing, Routing, Observability. - **Website-Vorscan** → OrgCard + Glossar + Wissensindex (Embeddings) pro Kunde/Tenant. - **Gateway-RAG** injiziert **serverseitig** Kontext/Policies → einfache Client-Prompts, konsistente, markentreue Antworten. - **Skalierbar**: Heute Einzelnode; morgen Multi-Region/GPU-Cluster.

Komponenten & Ports (Ist-Stand)

- **LLM-Service (llama.cpp)** — Mistral-7B-Instruct-v0.2 (GGUF, Q4_K_M)
- Endpoint: `http://127.0.0.1:8000/v1/chat/completions`
- Startflags: `--chat_format mistral-instruct`, `--n_ctx 8192`, `--n_threads 6` (Mac, Metal via `LLAMA_METAL=1`).
- **MT-Service (Transformers, stabil)** — OPUS-MT de↔en (Marian Pipeline)
- Endpoint: `http://127.0.0.1:8090/translate`
- CT2-Variante vorhanden (Experiment), aktuell **deaktiviert** bis korrekte Konvertierung (siehe TODO).
- **Gateway-RAG (FastAPI)** — Zentrale Frontdoor
- Endpoint: `http://127.0.0.1:8088/…`
- Funktion: System-Policies + OrgCard + RAG-Kontext („Stuffing“) + Postprocessing (z. B. Klammer-Übersetzungen entfernen, UTF-8).
- **Website-Scanner (CLI)** — erzeugt pro Domain/Tenant:
- `orgcard.json` (Brand-Steckbrief), `glossary.json` („never_translate“), `kb.sqlite` (Pages & Chunks).
- **Steuer-Skript** `tl.sh` — Start/Stop/Restart/Status/Scan/Test in einem Terminal.

Referenz-Endpunkte (OpenAI-kompatibel)

- Chat: `POST /v1/chat/completions` (Gateway; OpenAI-Schema).
- MT: `POST /translate` (separater Dienst; später Adapter nach `/v1/chat/completions` als Tool-Call möglich).

Auth (lokal dev): aktuell offen; Produktiv: `Authorization: Bearer <API_KEY>` + optional HMAC/mTLS.

Runbooks (Dev-Pfad)

1) LLM (llama.cpp) starten

```
export LLAMA_METAL=1
python -m llama_cpp.server
  --host 127.0.0.1 --port 8000
  --model ~/trancelate-onprem/llm/models/mistral-7b-instruct-
v0.2.Q4_K_M.gguf
  --chat_format mistral-instruct --n_ctx 8192 --n_threads 6
```

Check: `curl -s http://127.0.0.1:8000/docs` → 200 OK.

2) MT (stabil, Transformers) starten

```
uvicorn mt_server:app --host 127.0.0.1 --port 8090
```

Test: `POST /translate {"source":"de","target":"en","text":"Guten Morgen"}` → `Good morning`.

3) Website scannen (Tenant anlegen)

```
python scan_site.py https://domain.tld --out tenants/<tenant> --max-pages 40
```

Artefakte im Ordner: `orgcard.json`, `glossary.json`, `kb.sqlite`.

4) Gateway-RAG starten (nutzt LLM + Tenant-Artefakte)

```
UPSTREAM=http://127.0.0.1:8000 TENANT_DIR=tenants/<tenant>
EMB_MODEL=intfloat/multilingual-e5-base TOP_K=16
uvicorn gateway_rag:app --host 127.0.0.1 --port 8088
```

Test: `POST /v1/chat/completions` mit kurzem Prompt; gateway injiziert Policies + Kontext.

5) Ein-Terminal-Bedienung via `tl.sh`

```
./tl.sh start   # LLM + Gateway starten (nohup, Logs)
./tl.sh status  # Ports + letzte Logs
./tl.sh scan https://domain.tld 80  # (Re)Scan
./tl.sh test    # Ping + Mini-Chat
```

```
./tl.sh restart # sauber neu
./tl.sh stop    # alles stoppen
```

Logs: `logs/llm.log`, `logs/gateway.log`.

Gateway-Policies (Serverseitig)

- **System-Default** (neutral, universell): kurze, sachliche Antworten in **Sprache des Website-Kontexts**; keinerlei Klammer-Übersetzungen/EN-Beisätze.
- **OrgCard-Injection**: aus `orgcard.json` (Brand, Summary).
- **RAG-Kontext**: `TOP_K` relevante Snippets aus `kb.sqlite`, **längere Snippets** (bis 800 Zeichen) möglich.
- **„Stuffing"**: Kontext wird in der **User-Message** platziert → LLM folgt Kontext strikter.
- **Defaults**: `temperature=0`, `top_p=1`, `max_tokens` hoch; `stop`-Sequenzen gegen EN-Anhänge.
- **Postprocessing**: Entfernt `(Translation: …)`, hält UTF-8/Umlaute via `json.dumps(..., ensure_ascii=False)`.

  Ergebnis: selbst **einfache Prompts** liefern markenkonforme, kontexttreue Antworten.

Website-Scanner (MVP)

- Crawl **gleiche Domain** (Follow-Redirects; HTML only im MVP).
- **Extraktion**: `trafilatura` (Fallback BeautifulSoup), Entfernen von Scripts/Styles; Heuristik für Titel + Text.
- **Chunking**: ~900 Zeichen/Chunk (hard cap 64/Seite; konfigurierbar).
- **Embeddings**: On-the-fly im Gateway; DB speichert Texte (SQLite).
- **OrgCard**: host-basierte Kurzbeschreibung; **Glossar**: naive Brand-Terms (MVP).

Geplante Upgrades: Sitemap-Pull, robots.txt-Respekt, Canonicals/Noindex, deduplizierte Templates, Headless-Rendering (Playwright) für JS-Seiten, PDF-Parsing, strukturierte Produktdaten, Heuristik für Seitensprache.

RAG-Details

- **Embeddings**: `intfloat/multilingual-e5-base` (mehrsprachig).
- **Similarity**: Cosine via normalisierte Dot-Products; `TOP_K` dynamisch (MVP 16–24).
- **Kontextgröße**: Snippetlänge 320–800 Zeichen (konfigurierbar).
- **Antwortdisziplin**: „nur Kontext" (Systemregel) + Kontext in User-Message.

Multi-Tenant & Datenlayout

Pro Kunde/Tenant ein Verzeichnis `tenants/<tenant>/` mit: - `orgcard.json` — Brand/Steckbrief (≤600 Token). - `glossary.json` — `never_translate` (Marken/Produkte). - `kb.sqlite` — Tabellen `pages(url,title)` und `chunks(page_id,idx,text)`.

Tenant-Auswahl: über API-Key → Gateway lädt richtige Artefakte (MVP: ENV `TENANT_DIR`).

Billing/Abos/Referrals – Anknüpfpunkte

- **Usage-Events** (Gateway):
- `request_id`, `tenant_id`, `route` (chat/translate), `model_alias`, `tokens_prompt`, `tokens_completion` **oder** `words`, `latency_ms`, `status`, Timestamp.
- **Ledger**: Aggregation in `token_ledger` (Monat/Plan), Anbindung an CRM/Shop/Affiliate.
- **Paket-Schätzung (Lead-Magnet)**:
- Wortinventar `W_total` (Scan); Änderungsrate `r`; Zielsprachen `L`.
- Monatsvolumen: `W_mon = W_total * r * L`.
- Mapping auf Tiers (Starter/Growth/Pro/Enterprise) + Begründung.

Security, Privacy & Compliance

- **Zentrale Frontdoor** (Gateway) – Engines nicht öffentlich.
- **Auth**: API-Keys/JWT, optional HMAC/mTLS/IP-Allowlist.
- **Privacy**: **keine Prompt-Bodies** loggen; nur Metriken/IDs.
- **Data Residency**: EU-Region; Mandantentrennung (Tenant-Ordner/DB-Schemas).
- **Retention**: Logs/Metriken nur kurzzeitig; Backups verschlüsselt (KMS).
- **AI-Act**: Dokumentation von Modellen, Versionen, Policies (Registry).

Reliability & SLOs

- **Healthz**: Gateway `/openapi.json`; LLM `/docs`; MT `/healthz`.
- **Backpressure**: Rate-Limit/429 pro Tenant; Queue-Limit; Timeouts (Gateway 180s).
- **Circuit Breaker**: externe Fallbacks nur mit Opt-in (Enterprise).
- **Rollouts**: Alias→Version, Canary (z. B. 5%), schnelle Rollbacks.

Observability

- **Metriken**: p50/p95/p99, Durchsatz, Fehlerquote, Queue-Tiefe, Token/Words pro Tenant, Kosten.
- **Tracing**: `request_id` End-to-End.
- **Logs**: `logs/llm.log`, `logs/gateway.log` (ohne Bodies).

CT2 (CTranslate2) – Stand & Plan

Ist: erste CT2-Server-Variante zeigte Wiederholungen (EOS/Tokenizer-Thema) → zurück auf **Transformers (stabil)**. **Fix-Plan**: 1) Konvertierung **korrekt** mit `python -m ctranslate2.bin.opus_mt_converter --model Helsinki-NLP/opus-mt-de-en --output_dir ... --quantization int8` (statt Transformers-Converter). 2) `SentencePiece` korrekt: `decode_pieces()` für Detokenisierung. 3) Decoding-Einstellungen: `beam_size=4`, `length_penalty≈1.0`, `no_repeat_ngram_size=3`, `end_token="</s>"`. 4) Threads/Compute: `Translator(..., inter_threads=2, intra_threads=6)` (CPU), später GPU.

Architektur-Leitplanken (Enterprise)

- **Frontdoor-Gateway** pro Region (EU zuerst).
- **OpenAI-kompatibler Vertrag** fix (Chat/Embeddings/Translate).
- **Routing-Matrix**: Tenant→Model-Pool; Sprachpaar→MT-Pool (Hot-Pairs OPUS-MT; Universal-Fallback M2M100); optional Pivot via EN.
- **Server-Policies**: System-Prompt/Decoding/Grammar/Terminologie **erzwingen**.
- **Eval/Guardrails**: Style/Terminologie/JSON-Validität; „keine Annahmen“-Policy.
- **Data Governance**: keine Bodies; Audit Trails; DPA, Residency.

Backlog (geordnete Nächste Schritte)

1) **CT2 korrekt** (de↔en) + weitere Hot-Pairs; **M2M100** als Fallback. 2) **Tenant-Auswahl per API-Key** (Header→Tenant-Dir Mapping). 3) **Sitemap-Crawler** + Headless (Playwright) + PDF-Support; Sprachdetektion je Seite. 4) **Glossar-Enforcer**: nie-übersetzen/erzwingen im MT; Regex/Grammar-Guard im LLM. 5) **Embeddings-Service** (TEI) + persistente Vektoren (pgvector). 6) **Admin UI**: OrgCard/Glossar/Scan-Status/Tier-Empfehlung/ABOs. 7) **Observability**: Prometheus/Grafana-Dashboards, SLO-Alarme. 8) **Auth**: API-Keys, HMAC, mTLS, IP-Allowlists. 9) **Model-Registry**: Aliase, Versionen, Rollbacks, Lizenzen, Eval-Scores. 10) **RAG-Qualität**: bessere Chunking/Ranking (BM25+E5 Hybrid, Reranking).

Troubleshooting (Quick Map)

- `zsh: parse error near ')'` → `setopt interactivecomments` oder Einzeiler ohne `#` in zsh.
- `Invalid chat handler: openai` → `--chat_format mistral-instruct`.
- `ModuleNotFoundError: uvicorn / sse_starlette` → `pip install -U "uvicorn[standard]" fastapi sse-starlette`.
- `port ... already in use` → `lsof -ti:<port> | xargs kill`.
- `Internal Server Error` **(CT2)** → prüfen auf `model.bin`, `source.spm`, `target.spm`; korrekte Konvertierung; `decode_pieces`.
- **Umlaute falsch/escaped** → `json.dumps(..., ensure_ascii=False)`.

Cheat-Sheet (Copy-Ready)

Alles neu & los

```
cd ~/trancelate-onprem && source .venv/bin/activate && ./tl.sh start
```

Scan (80 Seiten)

```
./tl.sh scan https://kunde.tld 80
```

Test

```
./tl.sh test
```

Stop/Restart

```
./tl.sh stop
./tl.sh restart
```

Anhang A — Dateiübersicht (Kern)

- `scan_site.py` — Crawler/Extractor → OrgCard/Glossar/KB.
- `gateway_rag.py` — Frontdoor-Gateway mit Policies, RAG, Postprocessing.
- `mt_server.py` — Stabile Übersetzung via Transformers (OPUS-MT).
- `mt_ct2_server.py` — CT2-Variante (optimiert, pending Fix-Konvertierung).
- `tl.sh` — Supervisor/Runner (LLM, Gateway, Scan, Logs).
- `tenants/<name>/` — Artefakte pro Kunde.
- `logs/` — Laufzeitlogs (ohne Bodies).

Anhang B — Empfohlene Defaults (Server)

- **LLM**: `temperature=0.2 (oder 0)`, `top_p=0.9 (oder 1)`, `max_tokens` je Anwendungsfall, Stop-Sequenzen für Klammer-EN.
- **RAG**: `EMB_MODEL=intfloat/multilingual-e5-base`, `TOP_K=16..24`, Snippets 600–800 Zeichen.
- **MT (Transformers)**: OPUS-Modelle; später CT2-Upgrade.
- **Timeouts**: Gateway→LLM 180s (langes Kontext-Stuffing).

Schluss

Mit dieser Architektur und den Runbooks ist TranceLate **enterprise-ready**: - **Zentrale Kontrolle** (Policies, Quoten, Billing, Audits). - **Markentreue Antworten** durch OrgCard + RAG – **ohne** Prompt-Gymnastik. - **Skalierbar** von Einzelnode bis Multi-Region. - **Erweiterbar** (CT2, Fallback-Modelle, Eval/Guardrails, Admin-UI, TEI/pgvector).

> Nächster Meilenstein (empfohlen, 1–2 Tage): **CT2 korrekt migrieren**, Embeddings als Service, Sitemap-Crawler. Danach: Auth/Quoten, Admin-UI, Observability-Dashboards.