Übergabeprotokoll – Anni (TranceLate)

Stand: 27.08.2025, 22:30 CET
Owner: Marisa Langer
Erstellt von: TranceLate (Assistenz)

---

1) Executive Summary

- **Ziel**: Selbstgehostete, stabile Übersetzungs-Pipeline mit Guard-Qualitätsschicht für SaaS-Betrieb.
- **Status**: *Stabil* auf MacBook Air (M2/16GB, CPU). Guard „**Anni**“ schützt Platzhalter/HTML/Zahlen und liefert konsistente Kurztexte (Banner/CTA) ohne Markeneinmischung.
- **Architektur**: 2 Prozesse/Services lokal: **mt_server (8090)** = HF/Opus-MT Worker; **mt_guard (8091)** = Quality-Gateway + API + Metriken + Prewarm.
- **Ergebnis heute**: Smoke-Tests durchgehend grün; Race-Conditions im Worker gelöst (Lock pro Sprachpaar + Semaphor für Inferenz). P95 hängt vor allem an Parallelisierungsgrad/Hardware; stabiler Baseline-Betrieb mit **MT_WORKERS=1** bzw. konservativem Concurrency-Limit.

---

2) Architektur & Fluss

1. **Client → Guard (8091)**
   POST `/translate` {source,target,text}. Guard friert **{{PH}}**, **{app}**, **HTML-Tags** und **Zahlen** ein, übersetzt nur reine Textsegmente, setzt alles verlustfrei ein, poliert (EN/PT) und prüft `ph_ok/html_ok/num_ok`.
2. **Guard → Worker (8090)**
   Worker nutzt **Helsinki-NLP/opus-mt**-Modelle je Sprachpaar. Pipeline-Aufbau ist thread-safe (Lock pro Key), Inferenz parallel begrenzt (Semaphor).
3. **Antwort → Guard**
   Guard restauriert Invarianten, normalisiert Spacing/Dashes/Colon, dedupliziert doppelte Tags, liefert JSON + Checks.

Unterstützte Paare (heute): EN↔DE/FR/ES/IT/PT/NL.
Invarianten: `{{COUNT}}`, `{app}`, `<strong>…</strong>` u. a.

---

3) Aktuelle Konfiguration (empfohlen/stabil)

- **ANNI_DEVICE=cpu** (keine MPS/CUDA-Contention auf M2 Air)
- **MT_WORKERS=1** (stabilster Tail; optional 2 mit Vorsicht)
- **ANNI_MAX_CONCURRENCY=1** (Semaphor pro Prozess)
- **ANNI_TORCH_THREADS=1**, **ANNI_TORCH_INTEROP=1**
- **ANNI_GUARD_BATCH=off** (Guard sendet Satzsegmente sequentiell; HTTP-Overhead vernachlässigbar für Banner/CTA)
- **ANNI_PREWARM=off** (on-demand möglich; separater Prewarm-Call vorhanden)
- **ANNI_API_KEY=topsecret** (erzwungener Header `X-API-Key`)

Ports: Worker 8090, Guard 8091.
Projektpfad: `/Users/marisalanger/trancelate-onprem`

---

4) Qualität – Schutz & Post-Edit

- **Freeze/Restore**: Platzhalter ({{..}}/{}), HTML-Tags, Zahlen (inkl. Dezimal), Versions- und Range-Pattern.
- **Punkt-Degeneration fix**: Keine Mikro-Segmentierung mehr; ganze Textsegmente → MT.
- **Tag-Dedupe**: `<strong><strong>…</strong></strong>` → `<strong>…</strong>`.
- **Interpunktion**: `:` / `–/—` eingefroren + sauberes Spacing.
- **Polish (leicht)**: EN/PT Light-Heuristiken; keine Brand-Einmischung; Frage bleibt Frage.
- **Checks**: `ok`, `ph_ok`, `num_ok`, `html_ok`, `paren_ok`, `len_ratio`.

Referenz-Smoke:
- A: `Guten Morgen` → `Good morning` (alle Checks **true**).
- B: `Nur heute: {{COUNT}} … <strong>{app}</strong> – 2 Tage gültig!` → `Only today: {{COUNT}} Number of places available at <strong>{app}</strong> 2 days valid!` (Checks **true**; Spacing/Colon korrekt; invariants erhalten).

---

5) Performance – Beobachtungen heute

- **Stabilster Modus (CPU, 1 Worker, Sem=1, Prewarm+moderate Concurrency):**
  p50 typ. ~0.8–1.6 s, p95 ~1.3–2.0 s bei Banner-Länge. **fail≈0**.
- **Regressionsursachen** (bewusst vermieden):
  – Mehrere Worker + MPS auf M2 → Device-Contention (Tail steigt).
  – Mikro-Segmentierung im Guard → Degenerationen (`….`).
  – Gleichzeitiger Pipeline-Bau ohne Lock → 500/502.

  Fazit: Auf dem M2 Air ist „**weniger parallel**“ = **stabiler Tail**. Für höhere TPS → dedizierte Server/GPUs.

---

6) Betrieb – Runbook (One-Step Befehle)

Start (stabil):

```
cd "/Users/marisalanger/trancelate-onprem" &&
ANNI_DEVICE=cpu ANNI_PREWARM=off ANNI_GUARD_BATCH=off
MT_WORKERS=1 ANNI_MAX_CONCURRENCY=1 ANNI_TORCH_THREADS=1
ANNI_TORCH_INTEROP=1
ANNI_API_KEY=topsecret ./start_local.sh
```

Prewarm (Guard):

```
curl -s -H 'Content-Type: application/json' -H 'X-API-Key: topsecret'
  -d '{"pairs":[["de","en"],["fr","en"],["es","en"],["it","en"],["pt","en"],
["nl","en"]]}'
  http://127.0.0.1:8091/prewarm
```

Smoke (zwei Calls):

```
curl -s -H 'Content-Type: application/json' -H 'X-API-Key: topsecret'
  -d '{"source":"de","target":"en","text":"Guten Morgen"}'
  http://127.0.0.1:8091/translate

curl -s -H 'Content-Type: application/json' -H 'X-API-Key: topsecret'
  -d '{"source":"de","target":"en","text":"Nur heute: {{COUNT}} Plätze frei
bei <strong>{app}</strong> – 2 Tage gültig!"}'
  http://127.0.0.1:8091/translate
```

Stop (hart, falls nötig):

```
lsof -tiTCP:8090,8091 -sTCP:LISTEN | xargs -I{} kill -9 {} 2>/dev/null ||
true
```

Health & Meta:

```
curl -s http://127.0.0.1:8090/health
curl -s http://127.0.0.1:8091/meta
```

Metriken (Guard): liefert u. a. `anni_uptime_seconds`, `anni_requests_total`, `anni_errors_total`, `anni_translate_latency_seconds_avg`

```
curl -s http://127.0.0.1:8091/metrics
```

---

7) API – Oberfläche

- **POST /translate** (`application/json`): `{source, target, text}` → `{translated_text, provenance, checks}`
- **POST /prewarm**: `{pairs:[["de","en"],...]}` → `{ok, results: [{src,tgt,ok,ms,err}]}`
- **GET /meta**: Engine/Backend-Status, TM-Einträge, Flags.
- **GET /metrics**: Prometheus-kompatibel.
- **Auth**: `X-API-Key` erforderlich (konfigurierbar via Env).

Hinweis: Worker unterstützt `/translate_batch`, wird im Guard derzeit bewusst **nicht** genutzt (Stabilität > Mikro-Gains).

---

8) Änderungen – Kerneingriffe heute

- **Guard**: Rückbau auf satzbasierte Übersetzung; Invarianten-Freeze; PUNC-Restore tolerant; Tag-Dedupe; EN/PT-Polish; JSON-Checks.
- **Worker**: **Thread-safe Pipelines** (Lock pro Paar), **INFER_SEM** (konfigurierbare Parallelität), **Vorwärmen** via Guard, konsistente Opus-Modelle für EN↔DE/FR/ES/IT/PT/NL.
- **Security**: API-Key erzwungen; Browser-Clients kompatibel.
- **Ops**: `/metrics` exportiert Basis-KPIs; Start-Script akzeptiert Runtime-Env.

---

9) Bekannte Stolpersteine & Abhilfe

- **zsh & Kommentare**: `#` in Inline-Ketten vermeiden (führt zu Parser-Fehlern).
- **MPS auf M2**: Mehrere Worker auf MPS → Tail-Inflation; auf CPU bleiben oder dedizierte GPU nutzen.
- **Pipeline-Init**: Ohne Locks führen parallele Inits zu 500/502 – *bereits gefixt*.
- **Über-Parallelisierung**: Hohe W/W-Kombis → P95/Timeout-Spitzen. Lösung: Worker=1–2 + Sem=1–2.

---

10) Nächste Schritte (24–72 h, priorisiert)

1. **Containerisieren (Compose)**: Guard+Worker als Services mit Healthchecks & Log-Rotation.
2. **Prod-HW vorbereiten**: Linux-Host mit mehr RAM/CPU oder GPU; dann **ANNI_DEVICE=cuda/mps** kontrolliert testen (1 Worker → Sem=1–2).
3. **Metriken ausbauen**: Histogramme (p50/p90/p95), Error-Codes je Route, Build-Info.
4. **Caching**: Optionaler Memory-Cache im Guard (LRU pro Paar) → p50/p95-Senkung bei Wiederholern.
5. **Sprachen erweitern**: Weitere Paare per Mapping in `mt_server.py`; später NLLB/Marian-Large evaluieren.
6. **CW-Schicht (optional)**: „Anni-Copywriter“ als separater Pfad *nach* stabiler MT-Basis (strikt geführt, Varianten, Tone-Guides).

---

11) Entscheidungslog (heute)

- **Pro Stabilität**: Satzbasiert, CPU, Worker=1 (oder 2 konservativ), Batch off.
- **Contra**: Keine MPS-Parallelisierung, keine aggressive Mikro-Batching-Segmente im Guard.
- **Begründung**: Qualität & Tail-Stabilität > theoretische Durchsatz-Gains auf Endgerät-Hardware.

---

12) Anhang – Schnellreferenz Env

- `ANNI_DEVICE` = `cpu|mps|cuda|auto`

- `MT_WORKERS` = Anzahl Uvicorn-Worker (empf. 1 auf M2 Air)
- `ANNI_MAX_CONCURRENCY` = gleichzeitige Inferences/Prozess (empf. 1–2)
- `ANNI_TORCH_THREADS` / `ANNI_TORCH_INTEROP` = 1/1
- `ANNI_GUARD_BATCH` = `off|on` (empf. off)
- `ANNI_PREWARM` = `off|on` (on ist optional; Guard-Endpoint vorhanden)
- `ANNI_API_KEY` = Secret für `X-API-Key`

Schlusswort
Anni ist **betriebsbereit** und qualitativ stabil im Kern-Use-Case (Banner/CTA/kurze Sätze) mit robustem Guard. Für höhere Last/mehr Sprachen: oben genannte Schritte ziehen – **ein Hebel nach dem anderen**. Viel Erfolg beim Go-Live!