Content Guardian Tower

Architettura Operativa & Decisioni MVP

1. Scopo del documento

Questo documento raccoglie **tutte le decisioni architetturali operative** prese per il MVP di Content Guardian Tower.

È pensato come **fonte di verità unica** per sviluppo backend, worker, DevOps e come riferimento per l'evoluzione futura della piattaforma.

---

2. User Segments & Core Workflow

User segments

- **Admin**: configurazione sistema, rules, sources, utenti, settings
- **Analyst**: triage ticket, analisi contenuti, commenti, chiusura
- **Viewer**: consultazione dashboard, ticket, export

Core user job

1. Ingestion periodica o manuale dei contenuti
2. Change detection e versioning
3. Analisi automatica (LLM + rules)
4. Creazione / aggiornamento ticket
5. Triage umano (assign, comment, close)
6. Escalation automatica
7. Export e audit

---

3. Principi architetturali

- **Asincronia by design**: ingestion, analysis, export, escalation e retention sono job async
- **State machine esplicita**: ogni processo lungo è modellato come run con step
- **Idempotenza**: retry sicuri senza duplicazioni
- **Soft cancel**: ogni run può essere fermata in modo consistente
- **Single-tenant on-prem**: semplicità e controllo
- **No microservizi nel MVP**: meno overhead, più velocità

---

4. Architettura ad alto livello

Componenti principali

4.1 API Server

Responsabilità: - Autenticazione (login/password su DB locale) - RBAC (Admin / Analyst / Viewer) - CRUD: tickets, rules, sources, users, settings - Monitor ingestion runs - Export sincrono CSV - Audit log query

4.2 Worker Service

Responsabilità: - Consumo job da queue - Orchestrazione pipeline di ingestion - Analisi LLM - Scheduler interno (escalation, retention, ingestion periodica) - Retry, timeout e cancel

4.3 Datastore

- **Postgres**: system-of-record
- utenti, ruoli
- configurazioni (sources, rules, settings)
- stato ticket
- run/job state
- audit log
- **Elasticsearch**:
- revisions
- analysis results
- proiezioni ticket per search/filtri
- **Redis**:
- queue job
- lock distribuiti (scheduler)

4.4 Storage file

- MVP: filesystem locale (volume Docker)
- Usato per:
- attachments ticket
- export CSV

---

5. Orchestrazione delle attività (job system)

5.1 Modello a state machine

Ogni ingestion è una **run** composta da step idempotenti:

1. RUN_START
2. FETCH_ITEMS (per canale)
3. NORMALIZE + HASH
4. STORE_REVISION
5. DIFF
6. ANALYZE_LLM

7. UPSERT_TICKET
8. RUN_FINISH

Ogni step: - ha stato (`PENDING / RUNNING / SUCCEEDED / FAILED / SKIPPED`) - registra `attempts`, `last_error` - controlla `cancel_requested`

5.2 Idempotenza

• Ogni contenuto genera una chiave:
• `content_key = sha256(normalized_text + canonical_url)`
• Vincoli unici su:
• `(source_id, external_item_id, revision_hash)`
• Retry e crash **non generano duplicati**

5.3 Retry & timeout

Per ogni job type: - `max_retries` (es. 3) - `timeout_seconds` per step - backoff esponenziale con jitter - errori non-retryable (es. credenziali invalide)

5.4 Cancel / Stop run

• API: `POST /ingestion-runs/{id}/cancel`
• Implementazione: **soft cancel**
• Il worker:
• controlla flag `cancel_requested` tra step
• non accoda nuovi step
• marca run come `CANCELED`

---

6. Scheduler interno

Funzioni schedulate

• Escalation automatica ticket
• Retention 6 mesi
• Ingestion periodica source

Implementazione

• Scheduler **interno al worker**
• Loop periodico (30–60s)
• Lock distribuito (Postgres advisory lock o Redis)
• Un solo worker attivo come leader

---

7. Escalation automatica

- Job periodico `ESCALATION_SCAN`
- Regola:
- ticket OPEN / IN_REVIEW
- nessuna attività > 48h
- Azioni:
- `escalated = true`
- audit event `ESCALATED_AUTO`

---

8. Export CSV (MVP)

- Modalità: **sincrona**
- Streaming response (no carico RAM)
- Guardrail:
- limite righe
- timeout request
- Se limite superato → errore 422 (future async export)

---

9. Retention

- Job `RETENTION_PURGE`
- Cadenza: periodica (scheduler)
- Azioni:
- cancellazione dati DB
- delete documenti ES
- delete file storage
- audit event

(Politica dettagliata da definire in documento dedicato)

---

10. Observability & audit

Logging

- JSON logs su stdout
- Campi obbligatori:
- request_id
- job_id
- run_id
- ticket_id (se applicabile)

Audit log

- Azioni utente (API)
- Azioni automatiche (job, escalation, retention)
- Append-only

11. Sicurezza (MVP)

- Login/password su DB locale
- Password hashing (bcrypt consigliato)
- Sessione 24h
- RBAC su endpoint
- Secrets via env / docker secrets
- Nessun secret nei log

---

12. MVP Scope Summary

Incluso: - API + Worker separati - Postgres + Redis + Elasticsearch - State machine job - Retry, timeout, cancel - Scheduler interno - Export sincrono

Escluso (per ora): - Microservizi - Export async - MinIO - Stack observability avanzato

---

13. Prossimi passi

1. Definizione schema DB (run, job, locks, audit)
2. Definizione job types e payload
3. docker-compose MVP
4. Vertical slice end-to-end

Status documento: APPROVATO PER SVILUPPO MVP