

Progetto GAINS

(Generative AI for Network Sustainability)

Deliverable D3.3

Technical Report su Progettazione e Implementazione Knowledge Manager

Giordano, A. - Basile, L. - De Giacomo A. - Haider, B.- Fruncillo, D.
2026-01

Work Package: WP3 Progettazione, Implementazione Prototipale e Valutazione degli Agenti

Stato: Definitivo | Versione: 1.0

Classificazione

Abstract

Questo lavoro descrive la progettazione e l'implementazione prototipale del Knowledge Manager, componente di knowledge-plane orientato all'operatività per ambienti telco e cloud, con l'obiettivo di trasformare documentazione, policy e artefatti tecnici in un sistema interrogabile, applicabile e verificabile. Il Knowledge Manager è concepito come servizio containerizzato, riproducibile in Docker, capace di operare con un endpoint esterno OpenAI-compatibile per embeddings e generazione, mantenendo disaccoppiamento dal provider e controllo sul perimetro informativo effettivamente trasmesso all'esterno. Il design è centrato su tre requisiti non negoziabili: applicabilità contestuale, grounding con citazioni verificabili e sicurezza contro contaminazione ed esfiltrazione. La soluzione adotta un'architettura a piani: pipeline di ingest deterministica, knowledge store con ricerca ibrida e filtri strutturati, e runtime agentico per retrieval e sintesi. L'ingest è implementato in Langflow mediante flow versionati, esportabili e importabili, che realizzano parsing, normalizzazione, chunking deterministico, hashing canonicalizzato, redaction obbligatoria per snapshot e controlli anti-leakage con rifiuto fail-closed. L'indicizzazione utilizza Weaviate con configurazione BYOV, in cui i vettori sono calcolati dall'applicazione e associati agli oggetti persistiti, così da preservare portabilità e governance del ciclo embeddings. Il retrieval combina keyword search e vector search tramite hybrid search, con fusion algorithms selezionabili e metadati di scoring ispezionabili, e applica filtri vincolanti basati su proprietà flattened di applicabilità come vendor, ruolo del device, servizio e vincoli di versione, riducendo la probabilità di drift semantico in contesti operativi. Il comportamento agentico è orchestrato tramite Elysia, adottando un decision tree che rende esplicite le transizioni tra classificazione della richiesta, gating, retrieval multi-sorgente, re-ranking deterministico, costruzione dell'evidence pack, sintesi grounded e validazione finale. La modalità operativa impone vincoli stringenti: nessun claim prescrittivo è emesso senza almeno una reference valida, e i conflitti tra fonti sono risolti con policy-first, degradando la risposta a non-operativa quando le evidenze sono insufficienti. A supporto dell'auditabilità, il sistema produce un Explainability Pack persistibile e esportabile, che include mapping claim-refs, parametri di retrieval, hash degli artefatti, e indicatori di qualità come citation coverage e unsupported claim rate, abilitando verifiche ripetibili e diagnosi

di regressioni. Sono definiti contratti API coerenti per ingest, query cited o structured, blueprint di servizio, snippet discovery, e explain pack, con envelope comune, model di errori, RBAC e controlli di export. L'implementazione prototipale include una reference repository, docker-compose completo, bootstrap automatico di schema e flow, gestione segreti via Docker secrets e smoke test riproducibile. La valutazione è strutturata su metriche di retrieval e grounding, includendo precision e stabilità del top-k, tasso di violazione dell'applicabilità, citation coverage e assenza di claim non supportati in modalità operativa, con una pipeline di regressione su dataset bloccato. La sicurezza è trattata come requisito end-to-end: minimizzazione dei payload verso l'LLM, difese contro prompt injection diretta e indiretta tramite evidence-pack isolation, policy di export control, rate limiting e fail-closed, in coerenza con le principali tassonomie di rischio per applicazioni LLM-integrate. Nel complesso, il Knowledge Manager fornisce una base implementativa solida per integrare conoscenza tecnica e governance operativa in un contesto agentic, con riproducibilità industriale, tracciabilità e controlli di qualità verificabili.

Indice

1. Scopo, perimetro e contesto operativo

1.1 Obiettivi del Knowledge Manager

1.2 Confini di responsabilità e interazioni con gli altri agenti

1.3 Assunzioni, vincoli e definizioni operative

1.4 Modalità di funzionamento: PoC e profilo esteso

1.5 Requisiti di replicabilità industriale e criteri PNRR

2. Requisiti e criteri di progetto

2.1 Requisiti funzionali (ingest, retrieval applicabile, blueprint, snippet, citazioni)

2.2 Requisiti non funzionali (sicurezza, auditabilità, portabilità, osservabilità)

2.3 Criteri di accettazione e quality gates

2.4 Scelte architetturali e trade-off

3. Architettura logica del Knowledge Manager

3.1 Vista d'insieme e componenti (Elysia, Weaviate, Langflow, LLM OpenAI-compatibile)

3.2 Flussi principali: ingest, query cited, blueprint, explain pack

3.3 Confini di sicurezza e governance (RBAC, audit, segregazione ruoli)

3.4 Error model e degradazione controllata

3.5 Osservabilità: trace_id, metriche, log strutturati

4. Modello dati canonico e schemi

4.1 Principi del modello dati (applicabilità, tracciabilità, deduplica)

4.2 Entità cross-domain (KnowledgeRef, TaxonomyApplicability, ResponseMeta)

4.3 Oggetti knowledge (DocSource, DocChunk, Policy, ServiceDefinition, ConfigSnippet, Blueprint, ConfigSnapshotSanitized)

4.4 Regole di hashing, idempotenza e versioning

4.5 Esempi JSON completi e campi obbligatori

5. Knowledge Store con Weaviate

5.1 Strategia di persistenza e indicizzazione ibrida

5.2 Schema classi Weaviate e proprietà indicizzate

5.3 Filtri strutturati per applicabilità (vendor, device_role, os_version, service_id)

5.4 Hybrid search e parametri di ranking (keyword + semantico)

5.5 Deduplica, aggiornamenti e rebuild dell'indice

5.6 Politiche di indicizzazione conservativa (snapshot sanitizzati)

6. Pipeline di ingest con Langflow

6.1 Profili di ingest e policy di trattamento

6.2 Parsing, normalizzazione e chunking deterministico

6.3 Redaction e controlli anti-leakage

6.4 Enrichment metadati e validazione applicabilità

6.5 Calcolo embeddings via endpoint OpenAI-compatibile

6.6 Upsert in Weaviate e gestione idempotenza

6.7 Receipt, reason codes e audit events

6.8 Esempi di flow Langflow (export) e configurazione runtime

7. Retrieval applicabile e RAG agentic con Elysia

7.1 Policy di gating e classificazione query (operativa vs informativa)

7.2 Costruzione candidate set e filtri vincolanti

- 7.3 Ranking deterministico e boosting (policy-first)
- 7.4 Grounded synthesis con citazioni e validazione claim-refs
- 7.5 Calcolo confidence, warnings e gestione conflitti tra fonti
- 7.6 Modalità structured output per integrazione con altri agenti
- 8. Contratti API del Knowledge Manager
 - 8.1 Convenzioni (versioning, envelope, trace_id, idempotency)
 - 8.2 Endpoint: `/query`, `/blueprint`, `/snippets`, `/ingest`, `/explain`, `/health`
 - 8.3 Schemi request/response e codici errore canonici
 - 8.4 RBAC per endpoint e policy di export
 - 8.5 OpenAPI (estratto) e regole di compatibilità
- 9. Explainability pack e document assembly
 - 9.1 Struttura canonica ExplainPack
 - 9.2 Mapping claim-refs e criteri di granularità
 - 9.3 Export in json e markdown_bundle
 - 9.4 Persistenza, bundle_uri e correlazione con audit trail
 - 9.5 Controlli di sicurezza sull'assembly
- 10. Implementazione prototipale (codice e componenti)
 - 10.1 Struttura repository e moduli
 - 10.2 Client OpenAI-compatibile (embeddings e chat)
 - 10.3 Integrazione Weaviate (schema init, query, filtri)
 - 10.4 Langflow (flows, secrets, variabili, ambienti)
 - 10.5 Elysia (decision tree, tool bindings, output contracts)
 - 10.6 Logging, tracing e metriche
 - 10.7 Strategie di test (unit, contract, e2e)
- 11. Deployment in Docker
 - 11.1 docker-compose: Weaviate, Langflow, Elysia, Knowledge API, dipendenze opzionali
 - 11.2 Dockerfile e build strategy
 - 11.3 Secrets management e configurazioni (env + docker secrets)

- 11.4 Bootstrap schema Weaviate e import flow Langflow
- 11.5 Smoke test end-to-end ripetibile
- 11.6 Hardening minimo e profilo produzione
- 12. Valutazione, qualità e regressione
 - 12.1 Dataset di test e coverage (documenti, policy, snippet)
 - 12.2 Metriche retrieval (precision@k, nDCG, hit-rate per applicabilità)
 - 12.3 Metriche grounding (citation coverage, unsupported claim rate)
 - 12.4 Test di regressione e “knowledge drift”
 - 12.5 Criteri di accettazione PoC e soglie
- 13. Sicurezza, compliance e privacy
 - 13.1 Minimizzazione dati e trattamento snapshot sanitizzati
 - 13.2 Threat model sintetico (abuso retrieval, leakage, prompt injection)
 - 13.3 Contromisure (redaction, allowlist, policy-first, export control)
 - 13.4 Auditabilità e tracciabilità per revisione
- 14. Limitazioni e roadmap evolutiva
 - 14.1 Limiti del PoC e rischi residui
 - 14.2 Evoluzioni: approval workflow, stato “approved” per snippet/policy, governance avanzata
 - 14.3 Estensione multi-vendor e gestione versioni OS avanzata
 - 14.4 Ottimizzazioni prestazionali e costi
- 15. Conclusioni

Bibliografia

Appendici