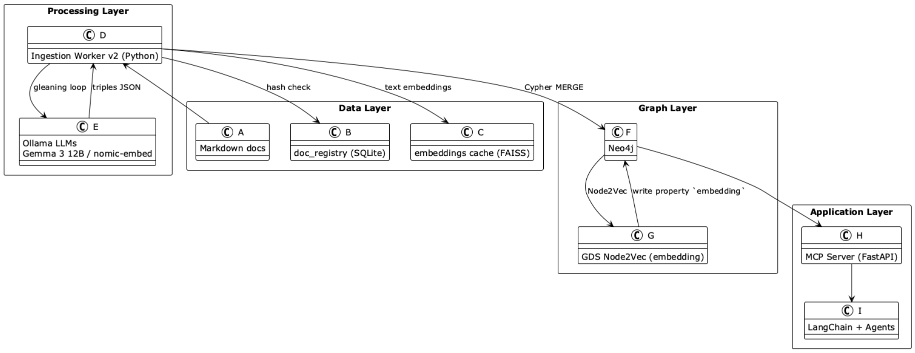
Skills-Graph Architecture

Bernd Prager

1 Logical View (high-level)



2 Ingestion Worker v2 (detailed steps)

1. **SHA-256 change detection** – skip unchanged docs (SQLite `doc_registry`).
2. **Chunk & embed** – 1 500-word chunks / 200-word overlap → `nomic-embed-text` vectors → optional FAISS.
3. **Gleaning loop extraction** – up to **3 LLM passes** (Gemma 3 12 B) per chunk; each pass only requests *new* triples.
4. **Cypher MERGE insert** – deterministic `MERGE` for nodes/relations; alias map normalisation.
5. **Registry update** – store new hash & timestamp (UTC).
6. **Node2Vec batch job** – after all files processed: GDS `node2vec.write()` (128-dim, 10×20 walks) → node property `embedding`.
7. **(optional)** Add graph embeddings to FAISS for hybrid doc + structural search.

   Performance note – With gleaning + Node2Vec the first full build takes ~3× the v1 time, but incremental runs only pay the Node2Vec cost if *any* doc changed.

3 Updated Infrastructure Topology

Host	Stack	Ports
odin	Neo4j 5.15 + GDS 2.x	7474 / 7687
odin	Ollama 0.6.8 (local models & `/api/embed`)	11434
odin	Ingestion Worker v2 (systemd)	–
odin	FastAPI MCP server	8000

4 Maintenance Jobs

Job	Schedule	Notes
`nightly_dedupe`	03:00	APOC `refactor.mergeNodes`
`node2vec_refresh`	After *any* ingest	Triggered automatically by worker
`refresh_embeddings`	Weekly	Re-runs text embeddings if model upgraded

5 Future Enhancements (next, ordered)

1. **Edge weighting & centrality pre-compute** for richer MCP ranking.
2. **Auto-summary blurb** (store summary on Entity)
3. **Embedding-aware LLM cache** to avoid redundant Gemma calls.
4. **Incremental Node2Vec** once graph size or runtime makes full runs painful
5. **Async ingestion + two-pass RAG** when we start serving high-QPS MCP queries