**Architettura (contenitori/servizi)**

* **UI (Streamlit)** – front-end locale per l’utente. Chiama il router.
* **Central (Router/Hub)** – decide a quale agente inoltrare i messaggi, gestisce lo stato di sessione e fa da proxy per test del retriever.
* **Triage (Classifier)** – LLM “rigido” che classifica la richiesta (classe 1 o 2) e fornisce il messaggio di benvenuto.
* **Agent Base (RAG)** – l’agente principale: recupera documenti da **Qdrant** (embeddings via Ollama) e chiede la risposta al modello LLM su **Ollama**. Espone diagnostica (health/debug/qdrant).
* **Qdrant** – vector DB per i documenti (collezione kb\_legale\_it). È invocato da Agent Base.
* **Ollama** – runtime modelli LLM ed embeddings; usato da Triage (LLM chat), Agent Base (LLM base) e per mxbai-embed-large.

**Endpoint/Chiamate tra componenti**

**1) UI → Central**

* **POST /chat**: invia {session\_id, message}. Se message è vuoto, vuole un preface.
* **GET /healthz**: per mostrare lo stato nella sidebar.
* **GET /test\_retriever**: la UI lo chiama per vedere gli snippet dei documenti (Central fa proxy verso Agent Base).

**2) Central → Triage / Agent Base**

* **Preface (messaggio vuoto)**: Central chiama **Triage** → **POST /chat** per ottenere un saluto iniziale.
* **Classificazione**: alla prima richiesta con testo, Central chiama **Triage** → **POST /classify** e memorizza klass in sessione (in-memory).
* **Conversazione**: Central inoltra all’agente scelto (base o special) → **POST /{agent}/chat**; gestisce hopping/hand-off con MAX\_HOPS e done/handoff nel payload di risposta. (Per default done=True così si evita il loop.)
* **Proxy retriever**: **GET /test\_retriever?q=…** su Central → inoltro a **Agent Base** /test\_retriever.

**3) Triage (Classifier)**

* **POST /classify**: ritorna {klass: 1|2} usando ChatOllama (temperature 0) con prompt rigido e parsing <K>…</K>.
* **POST /chat**: se il messaggio è vuoto, risponde con il preface; se non è vuoto, risponde minimal (il router userà /classify).
* **GET /healthz**: diagnostica.

**4) Agent Base (RAG)**

* **POST /chat**:
  1. inizializza componenti (LLM Ollama + retriever Qdrant con OllamaEmbeddings)
  2. retriever.get\_relevant\_documents(...) (top-K=8)
  3. costruisce SYSTEM\_PROMPT + context
  4. llm.invoke(user\_text)
  5. ritorna {"reply": ...}.
* **GET /test\_retriever**: prova di ricerca (torna snippet documenti).
* **GET /qdrantz**: info collezione (punti, status, vettori).
* **GET /debugz / /healthz**: stato inizializzazione e trace errori.

Note tecniche di init Agent Base: controlla che la **dimensione embedding** del modello (via embed\_query("probe")) coincida con la dimensione dei vettori in collezione; se mismatch → status=error e blocca l’init.

**Flusso tipico di una richiesta**

1. **UI** manda POST /chat a **Central** con {sid, message}.
2. **Central**:
   * Se message == "" → chiama **Triage /chat** per il preface e lo ritorna alla UI.
   * Se è il primo messaggio non vuoto → chiama **Triage /classify** e salva klass nello stato di sessione.
   * Sceglie l’agente (di default **base**) e inoltra a **Agent Base /chat**.
3. **Agent Base**:
   * Recupera documenti da **Qdrant** via retriever (embeddings **mxbai-embed-large** su Ollama).
   * Costruisce il prompt con **SYSTEM\_PROMPT** + contesto e invoca **Ollama** (Ollama LLM).
   * Ritorna {"reply": ...} al Router.
4. **Central** ritorna reply alla **UI**. In parallelo, la UI può chiamare **Central /test\_retriever** (che proxya **Agent Base** /test\_retriever) per mostrare i documenti usati.

**Dove sono configurati i target**

* **URL servizi** che il Router usa: TRIAGE\_URL, BASE\_AGENT\_URL, SPECIAL\_AGENT\_URL, REQUEST\_TIMEOUT\_S, MAX\_HOPS. Tutto via env in central\_agent.py.
* **Modello Ollama** e **Qdrant** in Agent Base: OLLAMA\_MODEL, OLLAMA\_BASE\_URL, QDRANT\_URL, QDRANT\_COLLECTION, EMBED\_MODEL.
* **Modello Triage**: OLLAMA\_MODEL (chat) e OLLAMA\_BASE\_URL.

**Endpoints utili per debug rapido**

* **UI**: controlla stato con **GET /healthz** (dal base URL impostato nella sidebar).
* **Central**: **GET /healthz**, **GET /session/{sid}**, **POST /reset\_session?session\_id=…**.
* **Agent Base**: **GET /healthz**, **GET /debugz**, **GET /qdrantz**, **GET /test\_retriever?q=…**.
* **Triage**: **GET /healthz**, **POST /classify**, **POST /chat** (preface).

Se vuoi, ti preparo un **diagramma ASCII** del flusso o uno **schema “cheat-sheet”** con cURL pronti per ogni endpoint (UI/Central/Agent/Triage).