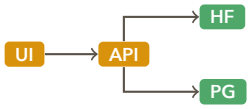
Chat MVP

Retrieval-Augmented Generation over a civic-tech corpus

Python 3.13 · FastAPI · Streamlit · PostgreSQL + pgvector · LangChain · Hugging Face

What it is A minimal RAG chat app grounded in the civictechdc/cib-mango-tree GitHub repository and related web pages. Sources are crawled, chunked, and embedded into PostgreSQL with `pgvector`; FastAPI retrieves context and calls the LLM; Streamlit provides the chat UI.

Architecture



Frontend	Streamlit UI (`frontend/app.py`) → `POST /query`
Backend	FastAPI + Uvicorn – health, query, session history
Store	PostgreSQL + pgvector – docs, embeddings, conversations
Models	Hugging Face – chat + BGE embeddings (768-d)
Ingest	Offline crawl/chunk + embed (`corpus/`)

How a query works

1. Client sends question (+ optional `session_id`, `top_k`)
2. Query embedded; pgvector returns top-k nearest chunks
3. LLM answers from retrieved context; history for follow-ups
4. Returns `answer`, `sources[]`, and `session_id`

Repository layout

`backend/`	FastAPI: `/health`, `/query`
`frontend/`	Streamlit chat UI
`corpus/`	Schema, ingest, embed, chat history
`evaluations/`	RAGAS, custom metrics, A/B, cost
`docs/`	Architecture & pipeline notes

API snapshot

POST /query

Body: `query`, optional `top_k`, `session_id`

Returns: `answer`, `sources[]`, `session_id`

GET /health → `{"status":"ok"}`

Swagger UI: `/docs` · ReDoc: `/redoc`

Stack

LangChain · Pydantic · Uvicorn

Trafilatura / BeautifulSoup (HTML extract)

Chat history trimmed by `CHAT_HISTORY_MAX_TOKENS`

Quick start

1	2	3	4	5
Env	**Install**	**Configure**	**Ingest**	**Run**
venv	`pip` + editable	.env + schema	crawl + embed	API + UI

`pip install -e .` → `python -m corpus.ingest.crawl && python -m corpus.embed` → `uvicorn` + `streamlit`

see README for full commands

MVP scope

No auth, no rate limiting, single-process deployment. Conversation history is used to interpret follow-ups–not as a factual source. Evaluation suite under `evaluations/`: RAGAS, hallucination, completeness, source retrieval, latency.