Architecture

Three layers:

1. **PHP CRM** (Worker Consumer), calls orchestrator via HTTP
2. **Python orchestrator** (FastAPI, LangChain)
3. **LLM / MCP servers / Chroma**

Relevant Worker Consumer methods: `llmChat()` (chat relay), `elaborateRag()` (document upload + vector build).

RAG

Indexing

Triggered on agent save with the Documents feature enabled. The Worker Consumer:

1. Reads selected CRM Documents
2. `POST /rag/keep` — prunes stale docs from orchestrator
3. `POST /rag/upload` — uploads new/changed files (multipart), stored as `docs/shared/<md5>.<ext>`, symlinked into `docs/<agent_id>/`
4. `POST /rag/build` — indexes all docs for the agent into Chroma at `vectors/<agent_id>/`.

Querying

With `rag: true` in `/agent/run`, Python injects a `query_documents` tool. The LLM decides when to call it. The orchestrator decomposes the question into ≤3 sub-questions (`needs_retrieval` flag), queries Chroma per sub-question, deduplicates by `doc_id`, re-ranks with FlashRank, and returns context. The LLM answer is grounded strictly in retrieved context.

Endpoints

The Python service exposes the following REST endpoints. All are mounted on the FastAPI app at port 8120.

Endpoint	Description

`POST /agent/run`	Agent loop: LLM + MCP tools + guardrails + optional RAG. SSE or JSON.
`POST /tools/inspect`	Introspect MCP server tools.
`POST /rag/build`	Index documents for an `agent_id` into Chroma.
`POST /rag/run`	Query vector store with question decomposition.
`POST /rag/keep`	Prune agent's doc symlinks to match `{filename: md5}`.
`POST /rag/upload?agent_id=`	Upload file to shared pool + symlink into agent's dir.

File Reference

```
plugins/agent/
├── compose.yaml
├── app.Dockerfile
├── requirements.txt
├── config.yaml        # auth token
├── src/vte_agent/
│   ├── __init__.py
│   ├── __main__.py
│   ├── agent.py             # /agent/run, /tools/inspect, calculator + rag tools
│   ├── config.py
│   ├── docs.py              # doc loaders
│   ├── logs.py
│   ├── models.py            # GGUFEmbeddings
│   ├── rag.py               # /rag/* endpoints
│   ├── schemas.py
│   ├── user_manual.py       # builtin vtenext user manual search tool
│   └── utils.py
├── docs/
│   ├── shared/        # <md5>.<ext> — deduplicated by content hash
│   └── <agent_id>/    # symlinks → ../shared/<md5>.<ext>
└── vectors/
    └── <agent_id>/    # chroma.sqlite3, parent_docs.json, description.txt

cache_local/
└── huggingface/       # local models cache (embedding, rerank)
```

Revision #2
Created 2026-09-17 15:25:06 UTC by Diego
Updated 2026-09-17 15:36:40 UTC by Diego