# **Documentation Technique Détaillée**

# **l** Contexte du Projet

Ce projet a pour but de créer un système conversationnel avancé reposant sur des LLMs (Claude Sonnet, OpenAI, ou locaux via LM Studio) et enrichi par un mécanisme RAG (Retrieval-Augmented Generation). Il intègre également un module de gestion de notes personnelles vectorisées et activables.

L'application doit être utilisable **en local**, avec des options hybrides vers des API payantes si configurées.

# 1. Base de données (PostgreSQL)

### 1.1 Schéma ERD (Entity Relationship Diagram)

- Entités principales: conversations, messages, rag\_corpus, documents, document chunks, notes, note chunks, llm configs, conversation context
- Relations:
  - o conversation 1 < -> n messages
  - o rag corpus 1 < -> n documents
  - o documents 1 < -> n document chunks
  - o notes 1 < -> n note chunks
  - o llm configs 1 < -> n conversations

#### 1.2 Détails des tables

#### Table: conversations

Champ	Type	Contraintes
id	SERIAL	PK
title	TEXT	NOT NULL
created_at	TIMESTAMP	DEFAULT now()
llm_config_id	INTEGER	FK -> llm_configs(id)

#### Table: messages

Champ	Type	Contraintes
id	SERIAL	PK
conversation_	_id INTEGER	FK -> conversations(id), ON DELETE CASCADE
role	TEXT	CHECK ('user', 'assistant')
content	TEXT	NOT NULL

```
Champ Type Contraintes
```

created\_at TIMESTAMP DEFAULT now()

#### Table: document\_chunks

```
| chunk_text | TEXT | NOT NULL |
| embedding | VECTOR | Index vectoriel (CHROMADB) |
```

#### 1.3 Index

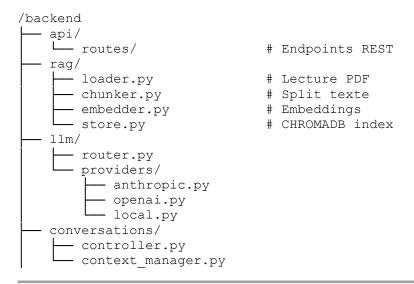
- GIN/IVFFlat sur embedding
- Index temporels sur created at
- Index FK pour optimisations de jointures

# 2. ii Architecture logicielle

#### 2.1 Vue d'ensemble

```
graph TD;
UI[Interface utilisateur] --> API[API Backend - FastAPI];
API --> DB[PostgreSQL];
API --> VS[Vector Store (CHROMADB/Chroma)];
API --> LLM[LLM (LM Studio ou API Claude)];
```

#### 2.2 Structure des dossiers Backend



### 3. RAG & Notes Vectorielles

### 3.1 Pipeline RAG

- 1. Upload PDF
- 2. Extraction du texte
- 3. Chunking (par paragraphe, ou 500 tokens avec overlap)
- 4. Embedding local (e5-base, Instructor, all-MiniLM-L6-v2)
- 5. Indexation CHROMADB
- 6. Utilisés dans les conversations activées par l'utilisateur

#### 3.2 Notes personnelles

- CRUD sur notes (titre, contenu)
- Même pipeline : chunking + vectorisation
- Stockées dans un index CHROMADB à part ou mêlé
- Utilisées dans les conversations activées par l'utilisateur

### 4. S Appels LLM

#### 4.1 Abstraction dynamique

- Configuration dans 11m configs
- Choix: local (LM Studio) ou distant (Claude, GPT-4)
- Construction du prompt :
  - Prompt utilisateur
    - o Ecrire...
    - Top-k chunks RAG + notes activées
    - o Ecrire...
    - Instructions système

### 4.2 Exemple de requête API

```
{
  "model": "claude-3-sonnet",
  "prompt": "Voici le contexte ...\nQuestion: ...",
  "temperature": 0.7,
  "max_tokens": 1024
}
```

### 5. Format des données JSON

#### Requête utilisateur

```
"conversation_id": 42,
  "content": "Que dit ce document ?",
  "llm_config_id": 1,
  "active_rags": [2],
  "active_notes": [5, 8]
}
```

#### Réponse du LLM

```
{
  "role": "assistant",
  "content": "Voici un résumé...",
  "sources": [
      {"type": "rag", "id": 2, "score": 0.89},
      {"type": "note", "id": 8, "score": 0.72}
]
}
```

### 6. Section Tests recommandés

#### **Tests unitaires**

- Vectorisation correcte (mock embedding)
- Routes FastAPI
- Chunking / parsing PDF

#### **Tests fonctionnels**

- Upload de document + RAG
- Activation de notes + recherche contextuelle
- Commutation LLM local/API

## 7. <u>A</u> Technologies / Coûts

Composant	Outil	<b>Gratuit?</b>
LLM local	LM Studio + Mistral	Oui
Embedding	Sentence Transformers	Oui
Vector Store	CHROMADB	Oui
API payante	Claude / OpenAI	Optionnel
DB	PostgreSQL	Oui
Backend	FastAPI	Oui
UI	Next.js / React	Oui

# 8. Instructions d'installation (local)

```
# Installer les dépendances Python
pip install -r requirements.txt
# Démarrer CHROMADB + PostgreSQL (Docker possible)
# Lancer LM Studio en local : http://localhost:1234
```

# 9. 🎄 Bonnes pratiques

- Toute configuration LLM modifiable dans <code>llm\_configs</code>
- Structuration stricte des prompts via system + context + user
- Modularité des providers pour en ajouter facilement
- Logs et erreurs centralisés dans /logs