Document de Conception
Système Chatbot Snappy avec Chiffrement E2EE

Équipe de Développement

5 juillet 2025

Table des matières

1 Introduction

Ce document présente la conception technique du système Chatbot Snappy, une solution de chatbot intelligent intégrant un module de chiffrement de bout en bout (E2EE). Le système permet de créer des chatbots personnalisés basés sur des documents PDF avec des capacités de traitement du langage naturel et de sécurité avancée.

1.1 Objectifs du Système

— Créer et gérer des chatbots personnalisés
— Traiter et indexer des documents PDF pour la base de connaissances
— Fournir des réponses contextuelles intelligentes
— Assurer la sécurité des communications avec le chiffrement E2EE
— Maintenir l'historique des conversations

1.2 Technologies Utilisées

— **Backend :** FastAPI, Python
— **Base de données :** PostgreSQL avec SQLAlchemy
— **IA :** Google Gemini, spaCy, sentence-transformers
— **Traitement PDF :** PyPDF2
— **Chiffrement :** Module E2EE (@Nameless0l/e2ee)

2 Diagramme de Classes

Le système est composé des classes principales suivantes :

Classe	Attributs	Méthodes
ChatbotDB	— id : UUID — accesskeys : String — label : String — prompt : String — description : String — projectId : String — languageModel : String — chatbotAttachements : JSONB	— create_chatbot() — get_chatbot_by_access_key()
ConversationDB	— idInstanceChat : String — accesskeys : String	— create_conversation() — get_conversation()
MessageDB	— id : String — idInstanceChat : String — message : Text — response : Text — timestamp : DateTime	— save_message() — get_messages_by_conversation()
EmbeddingDB	— id : String — accesskeys : String — embedding : JSON	— save_embedding() — get_embeddings_by_chatbot()
FastAPIApp	— app : FastAPI — model : GenerativeModel	— init_chatbot() — create_chatbot_instance() — infer_chatbot_response() — get_conversation_history()
E2EEModule	— public_key : String — private_key : String	— encrypt_message() — decrypt_message() — generate_keys() — exchange_keys()

2.1 Relations entre Classes

— **ChatbotDB** a une relation 1 :* avec **ConversationDB**
— **ConversationDB** a une relation 1 :* avec **MessageDB**
— **ChatbotDB** a une relation 1 :* avec **EmbeddingDB**
— **FastAPIApp** utilise **PDFProcessor**, **E2EEModule** et les modèles Pydantic

3 Diagramme de Cas d'Utilisation

3.1 Acteurs du Système

- **Utilisateur :** Personne qui interagit avec le chatbot
- **Administrateur :** Personne qui configure et initialise les chatbots
- **Système IA :** Composant intelligent pour le traitement du langage naturel

3.2 Cas d'Utilisation

- **Initialiser Chatbot** (Administrateur)
- **Créer Conversation** (Utilisateur)
- **Envoyer Message** (Utilisateur, Système IA)
- **Consulter Historique** (Utilisateur)
- **Chiffrer Communication** (Utilisateur, Administrateur)
- **Traiter Document PDF** (Administrateur, Système IA)

3.3 Relations

- **Include :** "Envoyer Message" inclut "Chiffrer Communication"
- **Include :** "Initialiser Chatbot" inclut "Traiter Document PDF"

4 Description des Cas d'Utilisation

4.1 Cas d'utilisation 1 : Initialiser Chatbot

- **Acteur principal :** Administrateur
- **Description :** Créer un nouveau chatbot avec des documents PDF
- **Préconditions :** L'administrateur a accès au système
- **Flux principal :**
  1. L'administrateur envoie une requête POST /init avec les paramètres du chatbot
  2. Le système vérifie si le chatbot existe déjà
  3. Le système traite les documents PDF joints
  4. Le système génère des embeddings à partir du contenu
  5. Le système sauvegarde le chatbot et les embeddings en base
  6. Le système retourne une confirmation de succès

4.2 Cas d'utilisation 2 : Créer Conversation

- **Acteur principal :** Utilisateur
- **Description :** Initier une nouvelle conversation avec un chatbot
- **Préconditions :** Le chatbot existe et l'utilisateur a les clés d'accès
- **Flux principal :**
  1. L'utilisateur envoie une requête POST /create avec les clés d'accès
  2. Le système vérifie l'existence du chatbot
  3. Le système génère un identifiant unique de conversation

4. Le système initialise le chiffrement E2EE
5. Le système retourne l'ID de la conversation

4.3 Cas d'utilisation 3 : Envoyer Message

— **Acteur principal :** Utilisateur
— **Description :** Envoyer un message et recevoir une réponse du chatbot
— **Préconditions :** Une conversation active existe
— **Flux principal :**

1. L'utilisateur envoie un message chiffré
2. Le système déchiffre le message
3. Le système génère un embedding du message
4. Le système recherche le contexte pertinent par similarité
5. Le système génère une réponse via Google Gemini
6. Le système sauvegarde l'échange en base
7. Le système chiffre et retourne la réponse

4.4 Cas d'utilisation 4 : Consulter Historique

— **Acteur principal :** Utilisateur
— **Description :** Consulter l'historique d'une conversation
— **Préconditions :** Une conversation existe avec des messages
— **Flux principal :**

1. L'utilisateur demande l'historique d'une conversation
2. Le système vérifie l'existence de la conversation
3. Le système récupère tous les messages de la conversation
4. Le système chiffre l'historique
5. Le système retourne l'historique chiffré

5 Diagrammes de Séquence

5.1 Séquence : Initialiser Chatbot

Étape	Administrateur	API FastAPI	Système/Base
1	Envoie POST /init avec PDF		
2		Vérifie chatbot existant	Base de données
3		Extrait texte PDF	PDFProcessor
4		Génère embeddings	Système IA
5		Sauvegarde chatbot et embeddings	Base de données
6	Reçoit confirmation	Retourne succès	

TABLE 2 – Séquence d'initialisation du chatbot

5.2 Séquence : Envoyer Message

Étape	Utilisateur	API	E2EE/IA	Base
1	Envoie message chiffré			
2		Déchiffre message	Module E2EE	
3		Vérifie conversation		Base de données
4		Génère embedding	Système IA	
5		Recherche similarité	Système IA	
6		Génère réponse	Google Gemini	
7		Sauvegarde échange		Base de données
8		Chiffre réponse	Module E2EE	
9	Reçoit réponse chiffrée	Retourne réponse		

TABLE 3 – Séquence d'envoi de message

6 Architecture du Système

6.1 Architecture Générale

Le système Chatbot Snappy suit une architecture modulaire en couches :
— **Couche Présentation :** API REST FastAPI
— **Couche Logique Métier :** Traitement des requêtes, IA, chiffrement
— **Couche Données :** Base de données PostgreSQL
— **Couche Sécurité :** Module E2EE pour le chiffrement

6.2 Flux de Données

1. Les documents PDF sont traités et convertis en embeddings
2. Les messages utilisateur sont chiffrés côté client
3. Le système recherche le contexte pertinent via similarité cosinus
4. Google Gemini génère une réponse contextualisée
5. La réponse est chiffrée avant envoi à l'utilisateur
6. Tous les échanges sont sauvegardés de manière sécurisée

6.3 Sécurité et Chiffrement

Le module E2EE (@Nameless0l/e2ee) assure :
— Chiffrement de bout en bout des communications
— Génération et échange sécurisé des clés
— Protection des données sensibles en base
— Authentification des utilisateurs

7 Modèle de Données

7.1 Base de Données PostgreSQL

Table	Clé Primaire	Clés Étrangères
chatbots	id (UUID)	-
conversations	idInstanceChat	accesskeys → chatbots.accesskeys
messages	id	idInstanceChat → conversations.idInstanceChat
embeddings	id	accesskeys → chatbots.accesskeys

TABLE 4 – Structure de la base de données

7.2 APIs REST

Endpoint	Méthode	Description
/init	POST	Initialiser un nouveau chatbot
/create	POST	Créer une nouvelle conversation
/infer	POST	Envoyer un message et recevoir une réponse
/history	POST	Récupérer l'historique d'une conversation

Table 5 – Endpoints de l'API

8 Conclusion

Ce document présente la conception complète du système Chatbot Snappy avec son module de chiffrement intégré. L'architecture modulaire permet une maintenance aisée et une évolutivité future. L'intégration du chiffrement E2EE garantit la confidentialité des échanges, répondant aux exigences de sécurité modernes.

Le système offre une solution complète pour créer des chatbots intelligents et sécurisés, capables de traiter des documents métier et de fournir des réponses contextuelles pertinentes.

8.1 Points Clés

— Architecture en couches pour la séparation des responsabilités
— Chiffrement E2EE pour la sécurité des communications
— Utilisation d'embeddings pour la recherche sémantique
— Intégration avec Google Gemini pour la génération de réponses
— Base de données PostgreSQL pour la persistance
— API REST FastAPI pour l'interface