PROJET D'INGÉNIERIE

Développement d'une solution logicielle d'« Audit Flash » pour la structuration de données énergétiques métiers

Projet mené en partenariat avec la société EQUANS

Étudiants :
Mathys Viau (Chef de projet)
Matthis Marini
Joachim Kahan
Kogan Kouyate
Ilian OUMEZZAOUCHE

Encadrants :
M. Anthony Mary (Equans)
M. Benoit Delinchant

Année universitaire 2025 – 2026

Table des matières

1 Introduction, cadre du projet et état de l'art **3**

1.1 Remerciements 3

1.2 Contexte industriel et présentation de l'équipe 3

1.3 Expression du besoin et cadre d'extraction strict 3

1.4 État de l'art et étude bibliographique 4

1.4.1 Les limites de l'OCR traditionnel par gabarit 4

1.4.2 L'apport des modèles de documents multimodaux 4

1.4.3 Évolution des systèmes de recommandation en efficacité énergétique . . . 5

2 Analyse fonctionnelle et architecture technique **5**

2.1 Parcours utilisateur et démarche « Human-in-the-Loop » 5

2.2 Architecture logicielle et flux de communication 5

2.3 Modélisation du graphe d'agents et persistance de l'état 6

2.4 Optimisation de l'interface utilisateur et internationalisation 7

2.4.1 La gestion centralisée de l'état de session 7

2.4.2 L'optimisation des performances par la mise en cache 8

2.4.3 L'architecture d'internationalisation (i18n) 8

3 Conception technique et implémentation logicielle **8**

3.1 Pipeline multi-documents et gestion du fallback visuel 8

3.2 Algorithme de consolidation et traitement des factures 9

3.3 Structure du moteur de recommandations à trois niveaux 9

3.4 Calcul de complétude et purification chirurgicale des fichiers Excel 10

3.5 Ingénierie des requêtes et contrôle structurel des sorties 11

3.5.1 La séparation des rôles : Instructions Système et Instructions Utilisateur . 11

3.5.2 Typage strict et suppression des hallucinations par schéma Pydantic . . . 11

3.6 Mécanique de vectorisation et mémoire historique (RAG) 12

4 Protocole de test, validation et résultats **14**

4.1 Protocole de test et métriques d'évaluation 14

4.2 Résultats quantitatifs par typologie de document 15

4.3 Évaluation de la pertinence des recommandations 15

4.4 Étude de cas concrète : Traitement de bout en bout 15
4.5 Analyse comparative inter-modèles et défis de la migration 16

5 Limites du système, modèle économique et perspectives 17
5.1 Sécurité industrielle et choix stratégique de l'infrastructure Cloud 17
5.2 Limites intrinsèques et points d'alerte techniques 17
5.3 Analyse financière et coûts de fonctionnement par document 18
5.4 Sobriété numérique et éco-conception de l'architecture 18
5.5 Perspectives d'amélioration et hébergement local 19

6 Documentation et guide de prise en main 19
6.1 Manuel de déploiement 19
6.1.1 Configuration initiale et variables d'environnement 19
6.1.2 Lancement automatisé de l'application 20
6.2 Guide d'utilisation de l'interface (Orienté Utilisateur) 20

Conclusion 21

7 Annexe 22

1 Introduction, cadre du projet et état de l'art

1.1 Remerciements

La réalisation de ce projet d'ingénierie a bénéficié de l'appui et de la collaboration de plusieurs acteurs du monde industriel et académique.

Nous tenons à remercier Monsieur Anthony Mary, porteur de ce projet au sein de la société Equans, pour sa disponibilité, la pertinence du sujet proposé et le partage des données historiques d'audits nécessaires au développement de notre application.

Nos remerciements s'adressent également à Monsieur Benoit Delinchant, professeur encadrant, pour son suivi régulier et ses orientations tout au long du semestre.

Enfin nous souhaitons exprimer notre reconnaissance à Messieurs Muhammad-Salman Shahid et Vincent Imard, chercheurs au sein du laboratoire G2ELab. Leurs compétences scientifiques nous ont permis de définir un cadre d'étude structuré dès la phase d'amorçage du projet.

1.2 Contexte industriel et présentation de l'équipe

La transition énergétique impose aux gestionnaires de parcs immobiliers et aux industriels une réduction de leur consommation de fluides et de leur empreinte carbone. Dans ce cadre, la société Equans réalise des audits énergétiques pour guider ses clients vers des solutions de décarbonation.

Le processus classique d'un audit commence par une phase d'ingestion documentaire. L'ingénieur thermicien doit collecter et lire manuellement des dizaines de fichiers hétérogènes : factures d'électricité ou de gaz, rapports de Diagnostic de Performance Énergétique (DPE) et fiches techniques d'équipements. Cette tâche répétitive limite le temps que l'expert peut consacrer à l'analyse technique fine et au conseil sur site. L'objectif de ce projet est donc de concevoir un logiciel d'« Audit Flash » pour automatiser la lecture et la structuration de ces documents bruts.

Afin de mener à bien ce projet, l'équipe s'est organisée selon la répartition suivante :

- **Mathys Viau (Chef de projet et développeur principal) :** Responsable de la conception globale et de l'écriture de la quasi-totalité de l'infrastructure logicielle. Cela comprend le développement de l'API avec FastAPI, la modélisation du graphe d'agents avec LangGraph, le codage du moteur de règles déterministes, la gestion des appels aux modèles Azure OpenAI, l'algorithme de tri des factures et les modules d'exportation vers Excel, Word et PDF.
- **Matthis Marini (Responsable frontend et supports visuels) :** En charge du développement des couches d'optimisation de l'interface utilisateur sous Streamlit, ainsi que de la création du poster scientifique et de la vidéo de présentation.
- **Joachim Khan (Support backend) :** Intervenu en soutien sur la partie backend pour assister le développeur principal lors de l'intégration de certaines routes de l'API.
- **Kogan Kouyate (Responsable bibliographie) :** Chargée des recherches documentaires, de la veille technologique sur le traitement de documents et de la structuration des bases théoriques.

1.3 Expression du besoin et cadre d'extraction strict

Le logiciel développé ne doit pas effectuer une simple lecture globale, mais extraire des données ciblées selon un cadre métier précis défini avec Equans. Les informations recherchées se divisent en quatre catégories techniques :

— **Morphologie et identification du site :** Le nom du client ou du site, l'adresse, l'année de construction du bâtiment, la surface totale, la surface chauffée et le secteur d'activité (tertiaire, industrie, logistique ou santé).
— **Caractéristiques de l'enveloppe (Le Bâti) :** La composition des murs, l'état de l'isolation des toitures, le type de menuiseries (PVC, bois, aluminium) et la nature du vitrage (simple, double ou triple).
— **Systèmes techniques (CVC / ECS / Éclairage) :** Pour le chauffage et le froid, la marque, le modèle, la puissance nominale en kW, le rendement et le fluide frigorigène des générateurs. Les températures de la loi d'eau (départ réseau et bascule extérieure) sont également recherchées. Pour la ventilation, les débits nominaux en m^3/h et le type de régulation sont requis. Enfin, le type et la puissance des éclairages complètent ce volet.
— **Historique de consommation :** L'extraction ligne par ligne de chaque facture d'énergie (électricité, gaz, réseau de chaleur) et d'eau, en isolant les périodes de consommation, les volumes consommés (kWh ou m^3) et les montants financiers hors TVA.

1.4 État de l'art et étude bibliographique

Pour orienter les choix techniques du projet, une analyse des technologies existantes dans le domaine du traitement automatique des documents (*Document AI*) et des systèmes de recommandation a été menée.

1.4.1 Les limites de l'OCR traditionnel par gabarit

L'extraction de données textuelles dans des documents reposait historiquement sur des moteurs de Reconnaissance Optique de Caractères (OCR) associés à des règles de lecture rigides ou des gabarits géométriques fixes (coordonnées en pixels). Si cette méthode fonctionne pour des documents standardisés, elle montre ses limites face aux documents de terrain d'Equans. Une simple modification de mise en page d'un fournisseur, un document numérisé de travers ou une photo d'étiquette technique prise avec un angle incliné suffisent à mettre le système en défaut. Ces outils traditionnels manquent de flexibilité et n'analysent pas le sens du texte.

1.4.2 L'apport des modèles de documents multimodaux

L'apparition des grands modèles de langage (LLM) et des architectures de transformateurs a modifié cette approche. Les modèles actuels, comme GPT-4o utilisé dans notre application, intègrent une approche multimodale qui combine deux capacités :
— **L'analyse sémantique (Textuelle) :** Le modèle comprend le contexte d'une phrase. Il peut identifier qu'un chiffre associé aux termes « kW » ou « puissance nominale » situé à côté d'une marque de fabricant correspond à la puissance d'une chaudière, indépendamment de sa place dans le document.
— **L'analyse topologique (Visuelle) :** Le modèle appréhende la structure visuelle des pages, la disposition des tableaux et l'alignement des colonnes. Cela lui permet de lire des documents scannés ou des factures complexes où les données sont imbriquées.
Cette technologie permet d'utiliser un seul moteur de traitement pour des fichiers de natures différentes (mémos audio retranscrits, photographies ou rapports PDF de plusieurs pages).

1.4.3 Évolution des systèmes de recommandation en efficacité énergétique

Traditionnellement, l'élaboration des recommandations d'actions de performance énergétique (APE) se fait manuellement par l'auditeur, qui s'appuie sur son expérience ou sur des grilles d'évaluation statiques au format Excel. Cette méthode garantit une approche logique mais dépend fortement de l'historique personnel de l'ingénieur et limite la capitalisation des solutions au sein de l'entreprise.

La recherche actuelle montre que l'utilisation combinée de bases de connaissances dynamiques et de modèles de langage permet de systématiser la recherche de gains énergétiques. C'est sur ce principe qu'est construit notre moteur de recommandations. Il associe des règles logiques strictes pour les évidences physiques à un système de recherche de documents similaires (RAG) basé sur l'historique global d'Equans, tout en laissant le modèle suggérer des pistes d'exploration transversales.

2 Analyse fonctionnelle et architecture technique

2.1 Parcours utilisateur et démarche « Human-in-the-Loop »

L'application est construite autour du principe de l'humain dans la boucle (*Human-in-the-Loop* ou HITL). Ce choix méthodologique consiste à ne pas déléguer la validation finale à un algorithme autonome, mais à utiliser l'intelligence artificielle pour exécuter les tâches répétitives de lecture et de mise en forme, tout en laissant le contrôle de la décision à l'auditeur énergétique.

Le parcours de l'utilisateur au sein de l'interface se déroule en six étapes :

1. **Le dépôt des documents :** L'auditeur dépose ses fichiers bruts (factures, rapports techniques, mémos audio) sur la plateforme Streamlit par simple glisser-déposer.
2. **L'analyse et le signalement des ambiguïtés :** Le système traite les documents et affiche deux résultats immédiats : le tableau des données extraites et une liste de points à éclaircir (par exemple, des incohérences détectées entre deux puissances de chaudières).
3. **La révision par l'auditeur :** L'utilisateur accède à un tableau modifiable. Il vérifie les valeurs extraites par la machine, corrige les éventuels oublis ou erreurs de transcription, et valide l'ensemble.
4. **L'évaluation de la complétude :** Une fois les données validées, un algorithme spécifique analyse les manques du dossier. Il attribue un score mathématique de complétude et liste les pièces ou les informations manquantes nécessaires pour finaliser l'audit réglementaire.
5. **L'ajustement des préconisations :** L'interface présente les fiches de recommandations énergétiques générées par le système. L'auditeur peut modifier les intitulés, réévaluer les investissements (CAPEX), ajuster les temps de retour sur investissement ou supprimer des actions s'il les juge non pertinentes.
6. **L'exportation des livrables :** L'auditeur télécharge les résultats sous trois formats : la matrice Excel officielle d'Equans remplie, ou un rapport de synthèse au format Word ou PDF, avec l'option d'intégrer ou de masquer le tableau des données brutes.

2.2 Architecture logicielle et flux de communication

Pour garantir la modularité du code et faciliter la maintenance, l'application est séparée en deux couches principales qui communiquent par le réseau :

— **L'interface utilisateur (Frontend) :** Développée avec Streamlit, cette couche gère l'affichage des onglets, les formulaires interactifs de modification des données et le téléchargement des rapports finaux.
— **Le serveur d'application (Backend) :** Développé avec FastAPI, il centralise la logique métier, la gestion du graphe d'agents et la communication avec les modèles d'intelligence artificielle.

La communication entre ces deux blocs s'appuie sur le protocole HTTP standard à travers des routes d'API dédiées (comme `/upload/` pour la réception des pièces ou `/validate/{thread_id}` pour l'enregistrement des corrections). Les fichiers physiques sont transmis sous forme de paquets de données composites (*Multipart/Form-Data*), tandis que les variables textuelles et les tableaux de données transitent via des messages JSON standardisés et vérifiés par des modèles de données Pydantic.

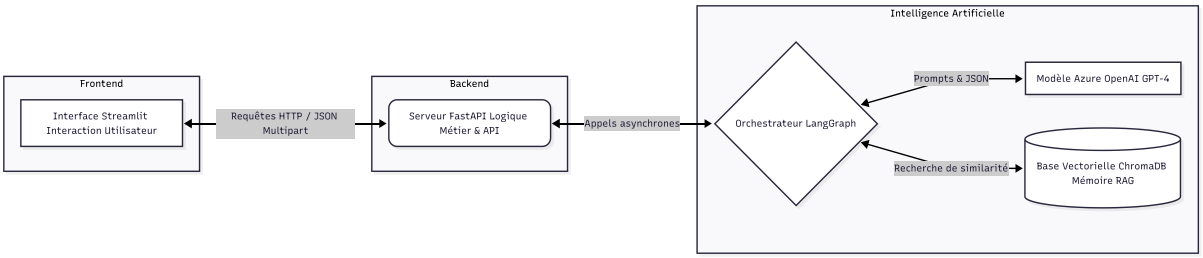


FIGURE 1 – Architecture logicielle globale et flux de données

2.3 Modélisation du graphe d'agents et persistance de l'état

La logique de traitement des documents est orchestrée sous la forme d'un graphe orienté grâce au framework LangGraph. Ce choix permet de définir une suite d'étapes logiques précises pour transformer un document brut en données structurées. Le graphe est structuré autour de quatre nœuds séquentiels :

1. **Le nœud de classification (`classify`) :** Ce point d'entrée analyse brièvement la nature du fichier reçu afin de lui attribuer une catégorie stricte (facture, rapport DPE, document d'éclairage ou mémo audio). Cette étape permet d'aiguiller le fichier vers le traitement le plus adapté et d'éviter de soumettre des volumes de texte inutiles aux modèles de langage.
2. **Le nœud d'extraction (`extract`) :** Ce module applique des consignes de lecture spécifiques à la catégorie du document validée à l'étape précédente. Si le fichier est un rapport DPE volumineux, le backend active une lecture parallèle : deux requêtes distinctes sont envoyées simultanément (un agent dédié aux structures du bâtiment et un ingénieur dédié aux systèmes thermiques) avant de fusionner leurs réponses, ce qui permet de gagner du temps et de ne pas surcharger la mémoire du modèle.
3. **Le nœud de revue humaine (`human_review`) :** Ce nœud constitue un point d'arrêt obligatoire dans le code. Le graphe interrompt son exécution automatique et attend un signal extérieur provenant de l'action de validation de l'auditeur sur l'interface Streamlit.
4. **Le nœud de contrôle final (`cross_check`) :** Ce dernier point vérifie la cohérence globale des modifications apportées par l'humain avant de clore le processus d'analyse du document.

L'interruption du graphe à l'étape de relecture humaine pose un problème technique : il faut sauvegarder l'état des variables sans bloquer les ressources du serveur web pendant que l'utilisateur effectue ses vérifications. Pour résoudre cela, un mécanisme de sauvegarde d'état (*State Checkpointing*) a été mis en place.

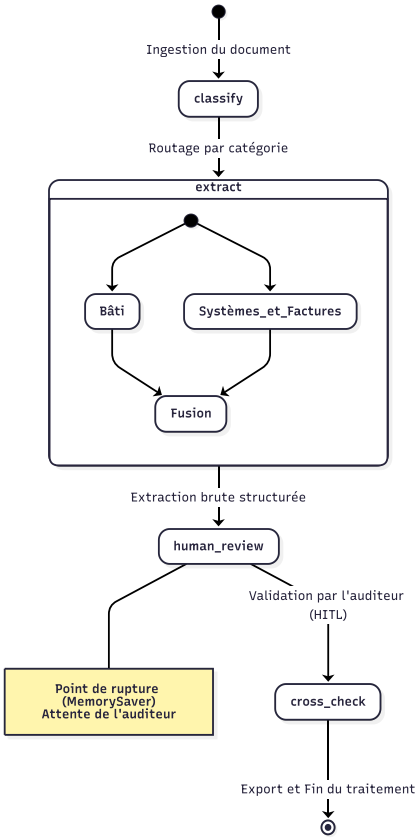


FIGURE 2 – Architecture logicielle globale et flux de données

À chaque session d'audit, le serveur FastAPI attribue un identifiant unique (`thread_id`). Lors de la compilation du graphe, la classe `MemorySaver` de LangGraph est configurée pour agir comme une mémoire tampon. Arrivé au point de rupture de la relecture humaine, le système sérialise l'état actuel des données extraites et le stocke en mémoire sous la clé de cet identifiant unique. Le processus d'exécution est alors interrompu, libérant immédiatement les ressources du serveur. Lorsque l'auditeur clique sur le bouton de validation dans son navigateur, la route d'API correspondante transmet les corrections avec le `thread_id` ; le système recharge instantanément l'état figé, applique les modifications humaines et relance le graphe vers les étapes suivantes sans perte de contexte.

2.4 Optimisation de l'interface utilisateur et internationalisation

Le développement de l'interface graphique sous Streamlit a fait l'objet d'une attention particulière afin de transformer un script de traitement de données en une application web robuste, fluide et adaptée aux contraintes des utilisateurs finaux. Trois choix d'architecture logicielle majeurs caractérisent cette couche de présentation :

2.4.1 La gestion centralisée de l'état de session

Par conception, le framework Streamlit réexécute l'intégralité du script Python à chaque interaction de l'utilisateur (clic sur un bouton, modification d'une case). Sans aménagement technique, ce comportement détruit le contexte et réinitialise les variables. Pour sécuriser le parcours de l'auditeur, un module de gestion des sessions (`session_manager`) a été développé. Il encapsule

l'état global de l'application, les identifiants uniques des documents en cours de traitement et les données de configuration. Cette encapsulation garantit la persistance des tableaux de révision, permettant à l'utilisateur de naviguer d'un onglet à un autre sans perdre ses modifications.

2.4.2 L'optimisation des performances par la mise en cache

Pour éviter de ralentir l'interface lors des appels réseau ou des lectures de fichiers sur le disque, l'application utilise des décorateurs de mise en cache (`@st.cache_data`). Par exemple, la fonction chargeant les statistiques de la base vectorielle locale est configurée avec une durée de vie limitée de 60 secondes. Cela permet d'afficher instantanément le nombre de vecteurs en mémoire dans la barre latérale sans saturer le serveur de requêtes répétitives à chaque rafraîchissement. De la même manière, le logo de l'entreprise est chargé en mémoire une seule fois pour toute la durée de la session.

2.4.3 L'architecture d'internationalisation (i18n)

Dans l'optique d'un déploiement au sein de filiales internationales ou pour répondre à des audits de sites multinationaux, l'application intègre un système d'internationalisation natif. Toutes les chaînes de caractères visibles à l'écran (titres, messages d'erreur, étiquettes de boutons) sont appelées via une matrice de traduction. Ce mécanisme permet de basculer instantanément l'intégralité de l'interface et des rapports exportés du français vers l'anglais, sans avoir à dupliquer le code source de l'application.

3 Conception technique et implémentation logicielle

3.1 Pipeline multi-documents et gestion du fallback visuel

La fiabilité du logiciel repose sur sa capacité à traiter des fichiers aux structures hétérogènes. Pour cela, un pipeline d'ingestion de bas niveau a été développé pour évaluer et conditionner la qualité des documents avant toute soumission aux modèles de langage.

Afin de prémunir le serveur contre la saturation de la fenêtre de contexte (limite de jetons ou *tokens*) et d'optimiser les temps de latence, un garde-fou volumétrique a été implémenté. Le code tronque l'analyse des documents dépassant un seuil critique défini par la constante globale `MAX_PDF_PAGES = 40`.

Lorsqu'un document valide est soumis, le système le parcourt page par page via la bibliothèque PyMuPDF. Il évalue le volume de texte brut extractible. Si une page contient moins de 50 caractères — ce qui caractérise généralement un document numérisé ou un schéma technique —, le pipeline bascule automatiquement sur un mode d'analyse visuelle adaptatif.

Listing 1 – Mecanisme de fallback visuel avec zoom matriciel

```
import fitz # PyMuPDF
import base64

def process_page_with_fallback(page):
    text = page.get_text("text").strip()

    # Si le texte est insuffisant (scan ou schema), on passe en OCR
       visuel
```

```
    if len(text) < 50:
        # Application d'un zoom 1.5x pour garantir la lisibilite
        zoom_matrix = fitz.Matrix(1.5, 1.5)
        pix = page.get_pixmap(matrix=zoom_matrix)

        # Encodage de l'image en Base64 pour le modele multimodal
        img_base64 = base64.b64encode(pix.tobytes("png")).decode("utf-8"
            )
        return {"type": "image", "content": img_base64}

    return {"type": "text", "content": text}
```

Cette approche hybride permet de conserver la rapidité du traitement textuel classique tout en assurant une continuité de lecture sur les pièces de mauvaise qualité issues du terrain.

3.2 Algorithme de consolidation et traitement des factures

Le module de traitement des factures énergétiques (électricité, gaz naturel, réseau de chaleur) fait l'objet d'un algorithme spécifique pour pallier le fait que les documents bruts sont souvent fournis en vrac. L'algorithme de génération de l'Excel, codé dans la classe d'exportation, opère un tri dynamique en deux étapes.

La première étape consiste en un partitionnement par site. Le modèle extrait de chaque ligne de facture le champ `nom_client_site`. Le backend utilise ensuite cette variable comme clé pour regrouper les données financières par entité géographique distincte, créant ainsi un espace isolé pour chaque bâtiment audité.

La seconde étape consiste en un redressement chronologique matriciel. Les dates textuelles extraites sont normalisées par une fonction de conversion interne. Les données mensuelles sont ensuite injectées séquentiellement dans des blocs de lignes fixes du classeur d'Equans, en respectant la séparation des années (N, N-1, N-2). Si deux factures partagent le même couple site/mois, l'algorithme cumule automatiquement leurs valeurs et y associe les Degrés Jours Unifiés (DJU).

3.3 Structure du moteur de recommandations à trois niveaux

Pour générer des préconisations d'efficacité énergétique fiables et éviter les hallucinations inhérentes aux modèles génératifs, le système implémente une architecture de décision structurée en trois niveaux.

Niveau 1 (Classe A) - Le moteur de règles déterministes : Ce niveau repose sur des règles logiques strictes codées en Python et ne fait appel à aucune probabilité. Par exemple, si l'extraction révèle la présence de circulateurs sans variateur de vitesse, le moteur déclenche l'insertion automatique d'une fiche d'action prioritaire. Ces préconisations assurent une conformité totale avec les standards métiers de l'entreprise.

Niveau 2 (Classe B) - Le RAG basé sur l'historique d'Equans : Ce niveau exploite la mémoire capitalisée de l'entreprise via une base de données vectorielle locale. Les données du bâtiment analysé sont comparées aux anciens audits. En cas de forte similarité, le système calque ses propositions et ses estimations d'investissement sur ces cas réels.

Niveau 3 - La génération directe (Pistes d'exploration) : En complément, le modèle est utilisé comme force de proposition. Pour forcer l'intelligence artificielle à formater ses idées

selon la norme attendue par l'interface, la requête est encapsulée via un schéma de données strict (bibliothèque Pydantic).

Listing 2 – Schéma de validation Pydantic pour le formatage

```
from pydantic import BaseModel, Field
from typing import List

class ActionPerformance(BaseModel):
    titre: str = Field(description="Titre de l'action de performance")
    description: str = Field(description="Principe technique detaille")
    cout_estime: str = Field(description="Estimation du CAPEX")

class SyntheseGlobale(BaseModel):
    recommandations_fondees: List[ActionPerformance] = Field(
        description="Recommandations justifiees par l'historique RAG"
    )
    pistes_exploration: List[ActionPerformance] = Field(
        description="Idees creatives et transversales suggerees par le
            LLM"
    )
```

Libéré des contraintes déterministes, le modèle analyse les signaux faibles du dossier pour proposer des stratégies d'optimisation plus larges, soumises in fine à la validation de l'auditeur.

3.4 Calcul de complétude et purification chirurgicale des fichiers Excel

L'évaluation de la complétude du dossier ne s'effectue pas via le modèle de langage — jugé trop incertain pour cette tâche — mais via un algorithme qui vérifie la présence effective des variables critiques au sein du tableau validé par l'humain.

Enfin, la sécurité et l'intégrité des exports sont assurées par un module de purification post-génération. L'injection de données dynamiques dans un tableur type avec la bibliothèque `openpyxl` laisse souvent des liaisons externes résiduelles (liens morts vers des répertoires Sharepoint d'origine) qui déclenchent des avertissements de sécurité chez le client final.

Le format `.xlsx` étant structurellement une archive ZIP contenant des fichiers XML, le script Python l'ouvre et parcourt son arborescence interne.

Listing 3 – Purification chirurgicale de l'archive Excel

```
import zipfile
import re

def _sanitize_post_generation(filepath: str):
    """Ouvre le fichier .xlsx comme une archive ZIP et purge les liens
        corrompus"""
    with zipfile.ZipFile(filepath, 'r') as zin:
        with zipfile.ZipFile(filepath + '.temp', 'w') as zout:
            for item in zin.infolist():
                content = zin.read(item.filename)

                # Cible les fichiers XML regissant la structure et les
                    liens
                if item.filename.endswith('.xml') or item.filename.
                    endswith('.rels'):
                    content_str = content.decode('utf-8')
```

```
                    # Suppression par Regex des liaisons externes
                        parasites
                    content_str = re.sub(r'<externalLink[^>]*>', '',
                        content_str)
                    content_str = re.sub(r'</externalLink>', '',
                        content_str)
                    content = content_str.encode('utf-8')

                zout.writestr(item, content)
```

Il supprime chirurgicalement les balises XML corrompues avant de ré-encode l'archive. Cette opération garantit la livraison d'un classeur Excel totalement sain et exploitable dès son ouverture.

3.5 Ingénierie des requêtes et contrôle structurel des sorties

L'implémentation de modèles de langage au sein d'un processus industriel d'audit énergétique nécessite de garantir la stabilité des réponses générées. Les modèles de type boîte noire ont naturellement tendance à faire varier la forme de leurs réponses (ajouts de commentaires, modifications de clés, omissions de champs), ce qui s'avère incompatible avec un système informatique automatisé qui doit ensuite injecter ces données dans des matrices d'exportation fixes. Pour résoudre ce problème, le logiciel s'appuie sur une ingénierie des requêtes (*Prompt Engineering*) rigoureuse associée à un contrôle structurel strict en amont et en aval de l'appel au modèle.

3.5.1 La séparation des rôles : Instructions Système et Instructions Utilisateur

La construction des requêtes envoyées à l'API Azure OpenAI repose sur un cloisonnement étanche des informations transmises au modèle. Chaque appel est encapsulé dans une structure de messages double :

- **Les instructions système (*System Messages*) :** Elles définissent le cadre de compétence, la posture métier et les règles de comportement absolues du modèle. Le prompt système configure l'IA comme un ingénieur thermicien expert d'Equans, spécialisé dans l'efficacité énergétique des bâtiments. C'est à ce niveau qu'on lui ordonne d'ignorer les formulations floues, de rester factuel et d'extraire exclusivement les éléments explicitement mentionnés dans le document, sans jamais extrapoler.
- **Les instructions utilisateur (*Human Messages*) :** Elles contiennent uniquement la matière brute à analyser (le texte nettoyé par le pipeline d'ingestion ou le texte transcrit de la pièce audio) ainsi que les données complémentaires éventuellement dictées par l'auditeur sur l'interface.

Cette méthode évite que le modèle ne confonde les consignes d'analyse (le processus) avec les données de l'audit (le contenu), limitant ainsi les risques d'interprétation erronée.

3.5.2 Typage strict et suppression des hallucinations par schéma Pydantic

Pour forcer le modèle à restituer ses analyses sous une forme directement exploitable par le code Python du backend, les appels n'utilisent pas de simples invites textuelles demandant un formatage libre. Le modèle GPT-4o est contraint par un mécanisme d'appel de fonctions (*Function Calling*) adossé à des classes Pydantic.

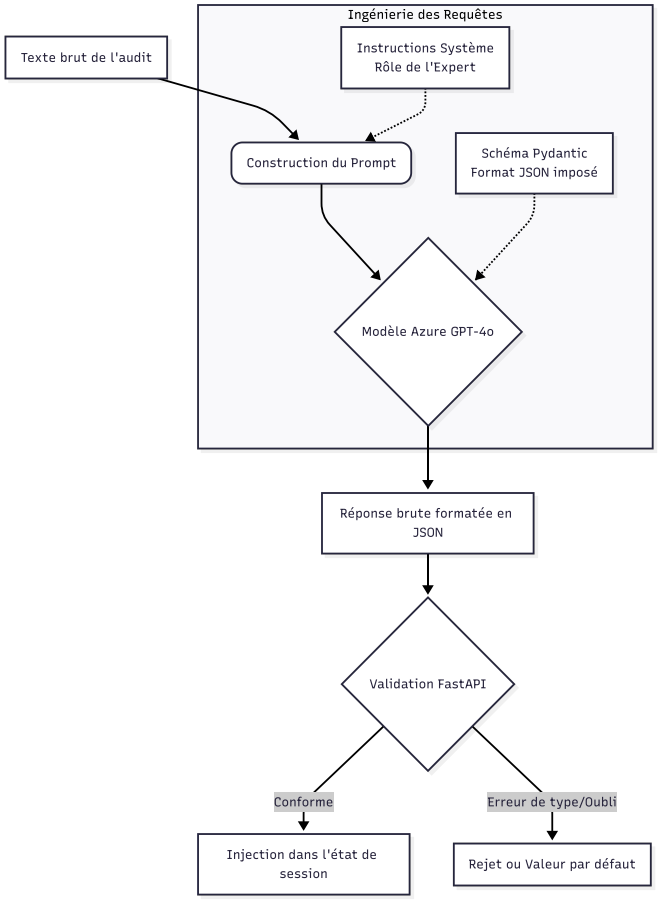


FIGURE 3 – Logigramme de validation structurelle des réponses de l'IA

Le fonctionnement de ce contrôle structurel se déroule ainsi :

1. Les variables attendues par Equans (puissances, marques, surfaces, isolation) sont déclarées dans le code Python sous forme d'attributs typés (chaînes de caractères, entiers, listes) au sein d'une classe héritant de `BaseModel`.
2. Chaque attribut est associé à une description textuelle explicite qui sert de consigne locale pour le modèle (par exemple : `puissance_nominale: float = Field(description="Puissance thermique utile du generateur en kW")`).
3. Lors de l'invocation de l'API, ce schéma de données est converti à la volée en un manifeste JSON Schema transmis au modèle Azure OpenAI. Le modèle ne génère plus un texte libre, mais remplit des cases pré-définies.
4. À la réception du flux, le backend FastAPI valide la conformité du format. Si un champ obligatoire est manquant ou si un type de donnée n'est pas respecté (par exemple une chaîne de caractères à la place d'un nombre pour une surface), le système intercepte l'anomalie.

Cette approche logicielle permet de supprimer la quasi-totalité des hallucinations structurelles : le modèle est mathématiquement contraint de respecter l'arborescence attendue, ce qui sécurise l'alimentation des scripts d'exportation Excel et Word en aval.

3.6 Mécanique de vectorisation et mémoire historique (RAG)

L'intelligence du moteur de recommandations de Classe B repose sur la technique du RAG (*Retrieval-Augmented Generation*). Ce système permet d'interroger l'historique des audits

d'Equans pour inspirer de nouvelles préconisations. Cependant, une base de données relationnelle classique ne peut pas comparer des concepts sémantiques ou des synonymes métiers. Le logiciel intègre donc un processus de transformation mathématique des documents hébergé sur une instance de base de données vectorielle ChromaDB.

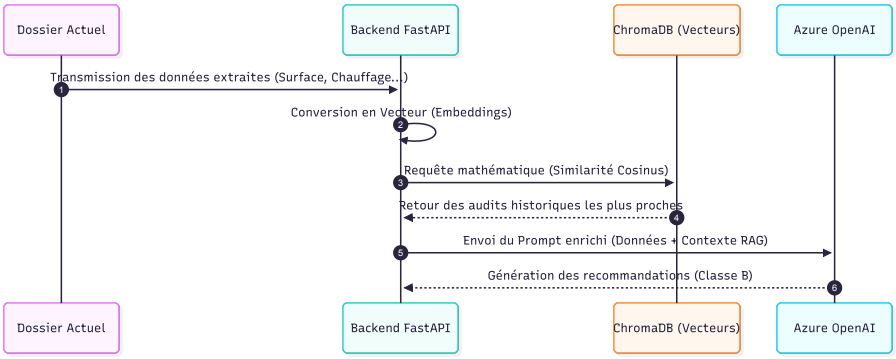


FIGURE 4 – Diagramme de séquence de la recherche vectorielle (RAG)

L'alimentation et l'exploitation de cette mémoire se déroulent en quatre étapes de calcul distinctes :

1. **La segmentation (*Chunking*) :** Lorsqu'un ancien audit validé est déposé pour enrichir le système, le texte brut est extrait puis découpé en blocs sémantiques de taille fixe (1000 caractères). Ce découpage empêche de perdre l'information pertinente dans un contexte trop vaste et optimise la pertinence des futures recherches.
2. **La vectorisation (*Embeddings*) :** Chaque bloc de texte est converti en un vecteur mathématique de haute dimension via un modèle d'intégration. Le texte devient ainsi une coordonnée dans un espace vectoriel virtuel. Dans cet espace, deux concepts thermiques similaires (comme « isolation par l'extérieur » et « ITE ») possèderont des coordonnées très proches, indépendamment des mots exacts employés.
3. **La recherche par similarité :** Lorsqu'un nouveau dossier est en cours de traitement, le backend synthétise les données du bâtiment (secteur, surface, type de chauffage) et les convertit à leur tour en un vecteur de requête. La base ChromaDB calcule alors la distance mathématique de cosinus entre ce nouveau vecteur et tous les vecteurs de l'historique d'Equans selon la formule :

$$\text{Sim}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

4. **L'augmentation du contexte :** Le système extrait les blocs historiques les plus proches du problème actuel (ceux dont la similarité tend vers 1) et les injecte de manière invisible dans le prompt envoyé au modèle de langage. Le modèle rédige alors ses recommandations en se basant sur des situations réelles et vécues par l'entreprise.

Cette architecture garantit que le logiciel ne se limite pas à des règles statiques, mais qu'il capitalise activement sur le savoir-faire des ingénieurs au fil des projets, transformant une archive passive en une mémoire active et interrogeable.

4 Protocole de test, validation et résultats

4.1 Protocole de test et métriques d'évaluation

Pour mesurer l'efficacité de la solution logicielle, une campagne de tests a été mise en place sur un échantillon de documents réels fournis par la société Equans. L'objectif est de quantifier la fiabilité de l'extraction automatique par rapport à une double relecture humaine, qui sert ici de référence absolue.

Le protocole appliqué pour chaque document se déroule ainsi :

1. Un opérateur humain liste l'ensemble des données critiques visibles dans le document selon les critères du cadre strict (ce volume constitue le nombre total d'informations attendues).
2. Le document est soumis à l'application sans modification préalable.
3. Les résultats générés en sortie sont comparés à la liste humaine pour comptabiliser précisément deux types d'anomalies : les oublis (données présentes mais non détectées) et les erreurs (données mal transcrites ou inventées).

Le calcul du score de précision final est obtenu par la formule suivante :

$$\text{Score d'extraction } (\%) = \frac{\text{Informations attendues} - (\text{Oublis} + \text{Erreurs})}{\text{Informations attendues}} \times 100$$

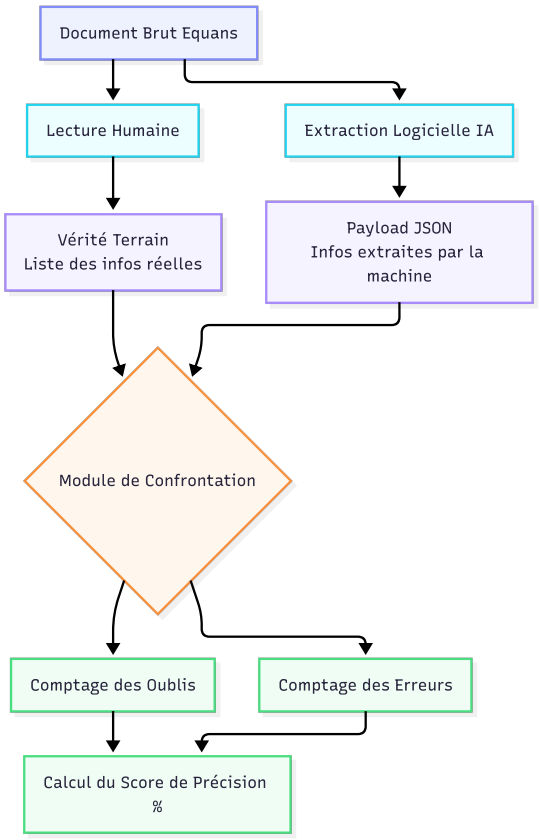


FIGURE 5 – Architecture logicielle globale et flux de données

4.2 Résultats quantitatifs par typologie de document

Les premiers tests font apparaître des variations de performance importantes selon le format et la qualité d'origine des fichiers d'entrée. Le tableau ci-dessous synthétise les taux de précision moyens observés lors des phases de validation :

Table 1 – Synthèse des taux de précision d'extraction par format

Type de document	Précision moyenne	Origine principale des écarts
Fichiers texte et PDF natifs	$\sim 95\%$	Aucun écart significatif constaté
PDF scannés et factures d'énergie	~ 85 %	Contrastes faibles, tampons superposés
Mémos audio (via Whisper)	~ 80 %	Bruits de fond techniques sur site
Photographies de terrain	~ 10 %	Reflets, étiquettes floues, manque de repères

Ces précisions ont été obtenu à partir d'un échantillon de 10 tests par type de documents. Le nombre de test étant assez faible, les résultats ne sont donc pas précis mais nous permet d'avoir un ordre d'idée des précision en fonction du type de document. Les documents numériques natifs obtiennent un résultat maximal. La séparation de la lecture en deux requêtes parallèles pour les rapports DPE lourds permet de contourner les phénomènes d'oubli en milieu de page. Pour les factures scannées, le taux de 85 % confirme l'utilité du traitement visuel par copie PNG, les erreurs restantes étant liées à des défauts de numérisation (plis du papier ou écritures manuscrites superposées). En revanche, le score très faible obtenu sur les photographies met en évidence l'incapacité actuelle du modèle à interpréter des images complexes de terrain ou à déduire l'état d'usure d'un équipement.

4.3 Évaluation de la pertinence des recommandations

La validation des recommandations générées par le système s'appuie sur une logique différente de celle de l'extraction brute. Les propositions doivent être cohérentes avec les règles de l'art du génie thermique.

- **Les actions de Classe A (Déterministes) :** Elles affichent une fiabilité totale. Reposant sur des vérifications logiques codées en dur, elles éliminent tout risque d'erreur. Si un réseau de distribution est détecté comme non calorifugé, l'action correspondante est systématiquement injectée avec les bonnes estimations de coûts unitaires.
- **Les actions de Classe B (RAG Historique) :** Leur pertinence dépend directement de la richesse de la base vectorielle locale. Lorsque le bâtiment étudié présente des caractéristiques proches d'un dossier historique d'Equans, le système parvient à répliquer des solutions adaptées, ce qui valorise l'expérience accumulée de l'entreprise.
- **Les suggestions directes de l'IA (Classe C) :** Elles demandent une vigilance accrue de la part de l'auditeur. Si elles apportent parfois des idées intéressantes pour ouvrir la discussion technique (comme des propositions de solarisation de parkings), elles manquent parfois de précision sur les coûts réels et nécessitent d'être ajustées ou complétées lors de la phase de révision humaine.

4.4 Étude de cas concrète : Traitement de bout en bout

Afin d'illustrer le fonctionnement opérationnel du logiciel, une étude de cas a été menée sur le dossier technique d'un bâtiment tertiaire de démonstration. Le jeu de documents comprenait un

rapport DPE au format PDF de 35 pages, un lot de 12 factures d'électricité étalées sur deux ans, et un mémorial audio d'une minute enregistré par l'auditeur lors de sa visite.

Le déroulement du traitement a mis en lumière les étapes suivantes :

1. **Ingestion et classification :** Le système a correctement identifié et aiguillé le rapport DPE et les factures vers leurs modules d'extraction respectifs. Le fichier audio a été transcrit textuellement avec succès, conservant les termes métiers dictés par l'ingénieur.
2. **Extraction et fusion :** L'analyse parallèle du DPE a permis d'isoler la surface du site ($2450m^2$) et les caractéristiques des murs sans omission. Les 12 factures ont été lues, et l'algorithme de regroupement a rassemblé l'ensemble des consommations sous un même identifiant de site, redressant les périodes pour les aligner chronologiquement.
3. **Phase de relecture (HITL) :** L'interface a signalé une incertitude sur la puissance nominale de la chaudière à gaz (le document mentionnant deux valeurs distinctes pour le chauffage et l'eau chaude). L'auditeur a pu corriger cette valeur directement dans le tableau Streamlit en quelques secondes avant de valider.
4. **Calcul des lacunes et export :** Le module de complétude a validé le dossier avec un score de 92 %, signalant l'absence de données sur le système de ventilation. Enfin, l'export post-purifié a généré une matrice Excel propre, exploitable immédiatement par les équipes d'Equans, ainsi qu'un compte-rendu au format Word regroupant les recommandations de Classe A et les pistes d'exploration de l'IA.

4.5 Analyse comparative inter-modèles et défis de la migration

La phase initiale de recherche et de développement du prototype a été menée en s'appuyant sur l'API du modèle Gemini 1.5 Flash. Ce choix initial s'expliquait par la rapidité de traitement du modèle, sa flexibilité face aux structures de données de test, ainsi que par la gratuité de son API sous un certain volume de requêtes, ce qui représentait un atout décisif pour maîtriser les coûts lors de l'itération du code. Cependant, conformément aux exigences de cybersécurité et de souveraineté des données d'Equans, une migration vers l'infrastructure cloud Microsoft Azure OpenAI (modèle GPT-4o) a été imposée pour la version de production.

Cette transition a mis en évidence des différences de comportement notables entre les deux architectures, constituant un retour d'expérience technique central pour le projet :

— **Le constat sur la complétude et l'enjeu multimodal :** Lors de la soumission de rapports identiques, le modèle Azure OpenAI parvenait à analyser l'intégralité des fichiers, mais affichait un taux de complétude inférieur. Il omettait plusieurs variables techniques que le modèle initial parvenait à isoler. Cet écart s'explique en grande partie par la supériorité de l'architecture de Gemini sur le traitement multimodal natif. Le logiciel s'appuyant sur un mécanisme de basculement en analyse visuelle (conversion matricielle PNG) pour les documents scannés, Gemini a démontré une capacité nettement plus robuste à extraire des données structurées directement depuis ces images complexes, là où GPT-4o s'est montré moins exhaustif.

— **La stratégie d'ajustement des requêtes :** Pour pallier cette baisse de performance, le code des prompts système a été restructuré afin de rendre les instructions plus directives. Les consignes d'extraction ont été segmentées et associées à des exemples de tolérance zéro face aux oublis pour forcer l'attention du modèle Azure.

— **Le bilan opérationnel :** Bien que le durcissement des requêtes ait permis de remonter le niveau de précision globale de l'application sur l'infrastructure Microsoft, une légère perte d'information persistait par rapport aux tests initiaux. Ce phénomène démontre que la portabilité d'une intelligence artificielle à une autre n'est pas automatique ; la supériorité

d'un modèle sur une modalité spécifique (ici, la vision) nécessite parfois d'accepter des compromis lors d'une migration dictée par des impératifs de sécurité industrielle.

5 Limites du système, modèle économique et perspectives

L'évaluation d'une solution logicielle en milieu professionnel ne peut se limiter à sa seule validation fonctionnelle. Elle doit intégrer une analyse de ses conditions de sécurité, de ses restrictions opérationnelles, d'une modélisation de ses coûts récurrents, ainsi qu'une feuille de route pour son évolution à plus grande échelle.

5.1 Sécurité industrielle et choix stratégique de l'infrastructure Cloud

L'intégration de modèles de langage dans un processus d'audit énergétique soulève des enjeux importants en matière de cybersécurité et de confidentialité. Les documents traités (factures, adresses de sites, plans architecturaux, puissances installées) contiennent des données sensibles, tant pour Equans que pour ses clients.

L'utilisation d'interfaces publiques (comme l'API standard d'OpenAI) a été écartée dès le début du projet. En effet, les conditions d'utilisation de ces services grand public autorisent l'exploitant à utiliser les données entrantes pour ré-entraîner ses modèles, ce qui pose un problème majeur de confidentialité.

Pour répondre à cette contrainte, l'architecture s'appuie exclusivement sur les services cloud d'entreprise Microsoft Azure (Azure OpenAI). Ce choix technologique garantit la protection des données selon plusieurs critères :

1. **Le cloisonnement des requêtes :** Les textes envoyés au modèle et les documents analysés restent confinés au sein de l'espace virtuel privé (le *tenant*) d'Equans.
2. **L'absence de ré-entraînement :** L'éditeur s'engage contractuellement à ne jamais utiliser les données soumises via l'API pour entraîner les modèles de la fondation OpenAI.
3. **La conformité réglementaire :** Les serveurs de traitement sont localisés géographiquement en Europe (sur la région France Central), ce qui assure le respect du Règlement Général sur la Protection des Données (RGPD) et des politiques de sécurité informatique d'Equans.

Cette sécurisation de bout en bout est indispensable pour envisager un déploiement de l'outil sur les postes de travail des ingénieurs.

5.2 Limites intrinsèques et points d'alerte techniques

Malgré la fiabilité globale du prototype pour un usage d'« Audit Flash », trois limitations techniques principales ont été identifiées lors des phases de tests :

— **La contrainte de taille des fichiers :** Afin de prémunir le serveur contre les surcharges de mémoire et de limiter les temps d'attente, un garde-fou strict a été codé via la variable `MAX_PDF_PAGES = 40`. Si un document dépasse cette limite, le pipeline applique une coupure automatique et n'analyse que la première partie. L'utilisateur doit donc être averti de cette perte potentielle d'informations.

— **Les faiblesses de l'analyse visuelle complexe :** Si le système réussit à déchiffrer les caractères dactylographiés ou scannés (grâce au zoom de 1.5x appliqué lors de la conversion en PNG), sa capacité à extraire du sens à partir de photographies de terrain ou de schémas reste très faible (taux de réussite inférieur à 20 %). Le modèle ne peut pas interpréter de manière fiable un schéma hydraulique ou déduire un diagnostic de vétusté à partir de la simple photo d'un équipement.
— **Le problème de confusion multi-bâtiments :** Lorsqu'un utilisateur dépose en même temps les pièces techniques de plusieurs bâtiments différents au sein d'une même session, l'intelligence artificielle a tendance à mélanger les informations. Ne disposant pas de barrières sémantiques pour segmenter les espaces physiques, le modèle peut attribuer les caractéristiques d'isolation ou les puissances de chauffage d'un bâtiment à un autre. Un traitement séparé, dossier par dossier, est donc obligatoire en l'état actuel du logiciel.

5.3 Analyse financière et coûts de fonctionnement par document

Pour évaluer la viabilité économique de l'outil, il est nécessaire d'estimer le coût de facturation des modèles Azure OpenAI. Ce coût dépend directement du nombre de jetons (*tokens*) consommés en entrée et générés en sortie. Sur la base des tarifs standards pour GPT-4o et Whisper, voici une estimation par type d'élément :

— **Un mémo audio de terrain (1 minute) :** La retranscription par Whisper coûte environ 0,006 euro par minute. La structuration textuelle par le modèle de langage étant très courte, le coût total est estimé à environ **0,01 euro par mémo**.
— **Un document numérique natif (10 pages) :** Une dizaine de pages représente un volume d'entrée d'environ 7 000 jetons. Avec le prompt métier et la réponse structurée en JSON, le coût total se situe entre **0,06 et 0,08 euro**. Ce coût peut doubler (environ 0,12 euro) pour les rapports DPE lourds qui activent la lecture en deux appels parallèles.
— **Un document scanné ou factures (10 pages) :** C'est le mode le plus lourd financièrement. La conversion en images haute résolution transforme le texte en jetons visuels. Le coût total est alors estimé entre **0,09 et 0,11 euro** pour 10 pages.

Au regard du temps humain économisé (plusieurs heures de tri et de saisie par dossier), ces coûts d'API, de l'ordre de quelques dizaines de centimes par audit global, offrent un retour sur investissement immédiat.

5.4 Sobriété numérique et éco-conception de l'architecture

Le cœur d'activité d'Equans étant orienté vers la décarbonation et l'efficacité énergétique des bâtiments, la conception d'un outil logiciel d'ingénierie se devait d'intégrer des problématiques de sobriété numérique. Les appels répétés à des modèles de langage lourds représentant un coût énergétique et calculatoire important dans les centres de données, trois choix d'architecture ont été mis en œuvre pour limiter l'empreinte environnementale de l'application :

1. **Le filtrage par pipeline hybride :** Au lieu de soumettre systématiquement l'intégralité des documents au modèle de vision multimodal (très consommateur en ressources), le système privilégie une extraction textuelle légère via PyMuPDF. Le basculement vers le rendu d'images PNG et l'analyse visuelle par l'intelligence artificielle ne s'active qu'en dernier recours, uniquement lorsque la page analysée comporte moins de 50 caractères.
2. **Le contrôle des volumes d'entrée :** La mise en place de la variable limite `MAX_PDF_PAGES = 40` agit comme un modérateur de charge. En bloquant l'analyse automatique des annexes

administratives ou réglementaires souvent présentes à la fin des longs rapports techniques, le code évite de solliciter inutilement les serveurs de calcul pour des données sans valeur ajoutée métier.

3. **La réduction des requêtes redondantes :** L'utilisation de mécanismes de mise en cache (`@st.cache_data`) au niveau de l'interface utilisateur Streamlit permet de figer les données d'affichage et les statistiques de la base vectorielle ChromaDB pendant une durée définie. Cela évite de relancer des cycles complets de calcul et de transfert de paquets réseau à chaque clic ou rafraîchissement de page provoqué par l'auditeur.

5.5 Perspectives d'amélioration et hébergement local

Pour faire évoluer le prototype vers une version logicielle définitive, deux axes de développement sont envisageables :

1. **L'assouplissement de la matrice Excel :** Actuellement, l'algorithme d'exportation est contraint par la rigidité du fichier Excel type d'Equans, qui impose des structures de tableaux et des nombres de lignes fixes. Une évolution importante consisterait à reconstruire cette matrice pour la rendre dynamique, permettant à l'application de créer ou de supprimer des blocs entiers selon les équipements réellement détectés (comme cela a été amorcé pour l'onglet des factures).
2. **La transition vers des modèles *Open Source* locaux :** Afin de garantir une indépendance totale vis-à-vis des fournisseurs cloud et d'annuler les coûts d'API, la solution la plus adaptée consiste à migrer l'application sur un serveur interne équipé de cartes graphiques (GPU). L'application pourrait alors s'appuyer sur des modèles ouverts performants :
   — **Mixtral 8x22B :** Son architecture compartimentée est bien adaptée pour l'analyse de tableaux et le respect des formats JSON.
   — **Llama-3 70B :** Ses capacités de raisonnement en font un bon candidat pour identifier les incohérences thermiques.
   — **La suite Qwen-2 :** Très précise pour l'extraction de données structurées, elle propose des versions audio natives qui permettraient de traiter les mémos vocaux en une seule étape, sans passer par un modèle intermédiaire.

6 Documentation et guide de prise en main

6.1 Manuel de déploiement

L'architecture de l'application étant scindée en deux services distincts (Backend et Frontend), son déploiement a été entièrement automatisé pour faciliter sa prise en main par les équipes techniques et métiers d'Equans. Il n'est pas nécessaire d'initialiser manuellement les environnements de développement ou d'installer les dépendances ligne par ligne.

6.1.1 Configuration initiale et variables d'environnement

Avant le tout premier lancement, il est impératif de s'assurer de la présence des variables d'environnement. Un fichier nommé `.env` doit être configuré à la racine du projet. Il contient notamment les clés d'accès sécurisées à l'API Azure OpenAI et les paramètres de la base de données vectorielle locale :

Listing 4 – Structure attendue du fichier .env

```
AZURE_OPENAI_API_KEY=votre_cle_securisee
AZURE_OPENAI_ENDPOINT=https://votre-espace.openai.azure.com/
DATABASE_URL=ChromaDB_local_path
```

6.1.2 Lancement automatisé de l'application

Le démarrage du système s'effectue simplement via l'exécution du script de lancement `lancer.bat` situé à la racine du dossier.

Ce script exécutable se charge de manière totalement transparente pour l'utilisateur de :

- Vérifier et activer l'environnement Python.
- Lancer le Backend (FastAPI) via le serveur ASGI Uvicorn en arrière-plan (exposant l'API sur le port 8000).
- Instancier le Frontend (Streamlit) sur le port 8501.

Une fois le script exécuté, l'interface graphique de l'Audit Flash s'ouvre automatiquement dans le navigateur web par défaut de l'utilisateur. En parallèle, pour les développeurs, la documentation interactive de l'API (Swagger UI) devient accessible à l'adresse `http://localhost:8000/docs`.

6.2 Guide d'utilisation de l'interface (Orienté Utilisateur)

Pour l'ingénieur thermicien, l'utilisation du logiciel a été pensée pour être directe, sans nécessiter de compétences en programmation. Le processus se divise en trois actions :

1. **Le dépôt de la matière brute :** Sur la page d'accueil, l'utilisateur glisse-dépose ses fichiers (PDF, factures, enregistrements vocaux) dans la zone de téléversement. Un indicateur de progression visuel l'informe en temps réel de l'état d'avancement de l'analyse.
2. **La révision interactive :** Une fois l'analyse terminée, l'interface affiche l'onglet de validation. L'auditeur se retrouve face à un tableau de données dynamique. Les champs où l'intelligence artificielle a détecté une ambiguïté sont mis en évidence. L'utilisateur peut modifier directement n'importe quelle cellule pour corriger une valeur ou valider la ligne.
3. **L'ajustement des recommandations et l'export :** Dans le dernier onglet, l'ingénieur consulte les fiches d'actions énergétiques suggérées par le triple moteur de recommandations. Il peut modifier le montant estimé des travaux (CAPEX) ou écarter les propositions hors de portée. Un clic sur le bouton « Générer les livrables » déclenche le téléchargement immédiat de la matrice Excel officielle remplie.

Conclusion

La réalisation de ce projet d'ingénierie logicielle apporte une réponse opérationnelle concrète à la problématique d'ingestion documentaire rencontrée par la société Equans lors des phases préliminaires d'audits énergétiques.

Sur le plan technique et méthodologique, ce travail marque une évolution par rapport aux outils d'extraction traditionnels. Le passage de systèmes de Reconnaissance Optique de Caractères (OCR) rigides, limités par des gabarits géométriques fixes, à une architecture multimodale adossée à un graphe d'agents (LangGraph) permet d'absorber la diversité des documents de terrain. La confrontation de la théorie académique aux exigences du monde industriel a imposé la mise en place d'une infrastructure robuste (FastAPI, schémas de contrôle Pydantic, base de données vectorielle ChromaDB) capable de garantir la sécurité des données grâce au cloud privé Azure OpenAI.

Au-delà de la performance algorithmique, la valeur de cette solution réside dans l'application de la philosophie de « l'humain dans la boucle » (*Human-in-the-Loop*). L'objectif du logiciel n'est pas de remplacer l'expertise de l'auditeur par un système totalement autonome, mais de le libérer des tâches chronophages de saisie, de tri et de mise en forme de données brutes. En automatisant cette phase sans valeur ajoutée, l'outil redonne de la disponibilité à l'ingénieur thermicien pour son cœur de métier : l'analyse technique sur site, la finesse du diagnostic et la conception de stratégies de décarbonation personnalisées pour les clients d'Equans.

Ce projet aura ainsi permis de lier les compétences en développement logiciel et en intelligence artificielle acquises au cours du cursus à une application industrielle concrète, posant les bases d'un outil évolutif et industrialisable au sein de l'entreprise.

7 Annexe

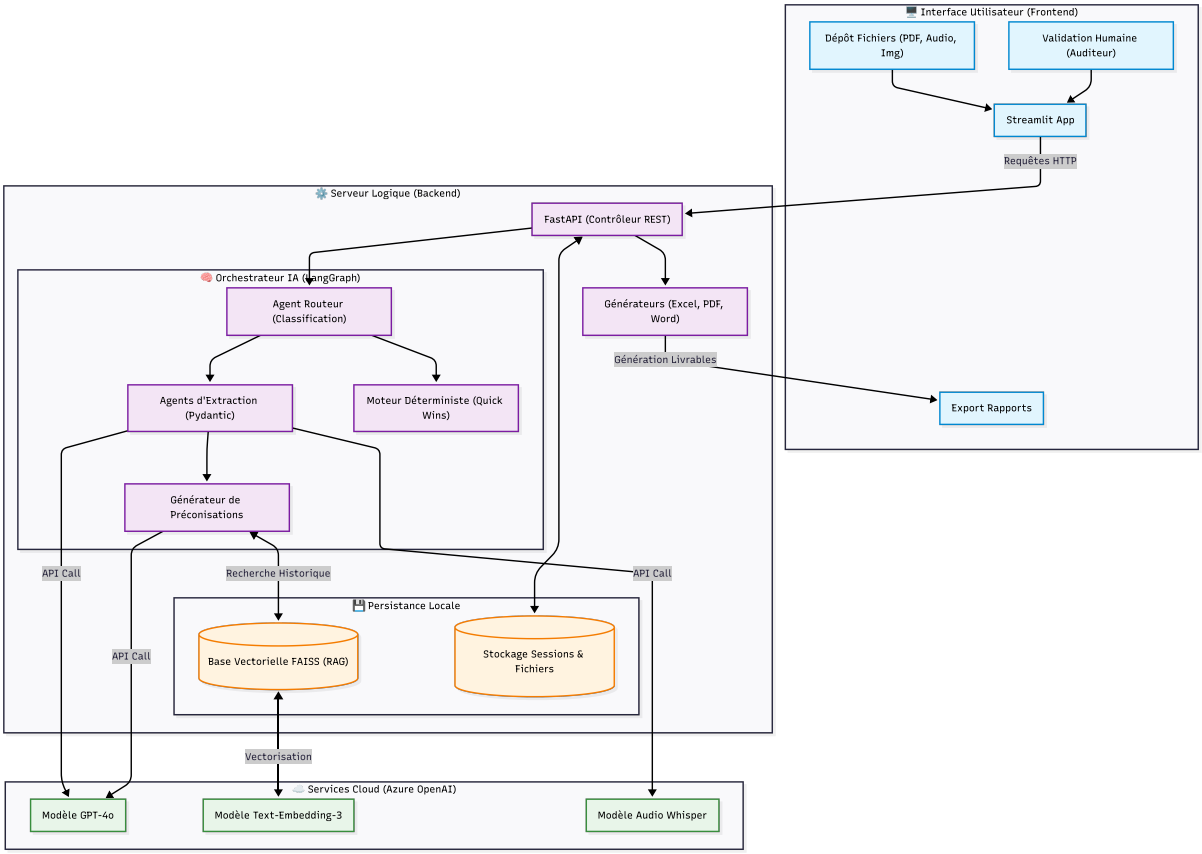


FIGURE 6 – Schema D'architecture