# **Document d'Architecture Global (GAD)**

## **1. Introduction**

Ce document présente l'architecture globale du système d'extraction et de traitement des documents. Le système a pour objectif de permettre la lecture, l'analyse et l'extraction d'informations (classification, résumé, modélisation thématique, NER, Q&A, extraction d'entités) à partir de documents financiers ou autres documents variés. Il s'intègre dans le SI et se présente sous deux formes d'interactions principales :

* **API programmables** : permettant l'invocation du lecteur de documents via des appels API.
* **Interface Utilisateur (UI)** : offrant aux utilisateurs finaux la possibilité d'uploader un document et de choisir le traitement à appliquer.

Le système est capable de traiter des documents de tailles, formats et niveaux de confidentialité variés, transmis par différents canaux, et supporte des traitements synchrones ou asynchrones.

## **2. Vue d'ensemble de l'architecture**

L'architecture se décompose en plusieurs couches :

* **Couche de Présentation**
  + *Interface Utilisateur (UI)* : Application web qui permet aux utilisateurs d'uploader des documents et de choisir l'opération souhaitée (classification, résumé, NER, etc.).
  + Command Line Interface (CLI) : Permet aux développeurs d’accéder à l’API directement
* **Couche API / Intégration**
  + *API REST (Document Reader API) avec FASTAPI* : Expose des endpoints pour recevoir les documents uploadés, spécifier le type de traitement et retourner le résultat.
  + *Gateway / Broker* : Optionnellement, une solution de gestion des files d'attente peut être mise en place pour supporter le traitement asynchrone de documents volumineux ou sensibles.
* **Couche Traitement (Backend)**
  + *Modules de Parsers* : Extraction du texte brut depuis différents formats (PDF, DOCX, TXT).
  + *Modules d'Extraction* :
    - **Extraction NER** : Pour les fichiers texte (chats) via un modèle NER (par exemple, spaCy personnalisé).
    - **Extraction Rule-based** : Pour les fichiers DOCX, avec détection des tableaux et extraction clé/valeur (mapping avec les entités recherchées).
    - **Pipeline RAG** : Pour les fichiers PDF, reposant sur un LLM avancé via Langchain et un modèle LLM robuste
      * Pour un POC on peut utiliser le modèle llama3-8b-8192 présent dans l’api de Groq (pour Q&A où résumé).car c’est simple à mettre en place. Nous pouvons préconiser d’utiliser OPENAI ou Bedrock pour aller plus loin.
* **Couche de Stockage et de Sécurité**
  + *Stockage Temporaire et Archive* : Les documents uploadés peuvent être stockés temporairement pour traitement, puis archivés ou supprimés selon les politiques de confidentialité. Pour le POC, On peut supprimer les données à la fin du traitement.
  + *Gestion des Accès et Sécurité* : Authentification, chiffrement et contrôle d'accès pour garantir la confidentialité des documents transmis.

## **3. Description des Composants et Interactions**

### **3.1. Interface Utilisateur (UI)**

* **Fonctionnalités** :
  + Upload de documents (formats PDF, DOCX, TXT, etc.).
  + Sélection du type de traitement souhaité (résumé, NER, extraction d'entités, Q&A).
  + Possibilité de fournir des paramètres additionnels, tels qu'une question pour Q&A ou une clé API pour interroger le modèle LLM (ici, Groq via Langchain).
* **Interfaçage avec l’API** :
  + L'UI envoie des requêtes HTTP POST au endpoint /process de l'API.
  + Les réponses structurées (au format JSON) sont affichées à l'utilisateur.
  + Un fichier texte peut être téléchargé

### **3.2. API REST du Document Reader**

* **Endpoints** :
  + /process : Endpoint principal qui accepte un document avec la possibilité de le split avec le multi-partitionning, des paramètres d'opération, une question éventuelle et une clé API pour l'accès au LLM Groq.
* **Traitement par Type de Document** :
  + **DOCX** : Le système détecte le contenu (tableaux et/ou paragraphes) et utilise un module rule-based pour extraire les paires clé/valeur (par exemple, Counterparty, Notional, etc.).
  + **TXT** : Le document sera traité par un module NER personnalisé (adapté aux textes de type chat) pour extraire les entités financières.
  + **PDF** : Le document, souvent non structuré, sera traité par un pipeline RAG. Selon l'opération spécifiée ("qa", "summarization" ou "entity\_extraction"), un LLM (Groq via Langchain) peut être appeler pour générer une réponse, un résumé ou extraire des entités.
* **Sécurité et Confidentialité** : (pas encore fait)
  + Le système doit supporter des transferts sécurisés (HTTPS) et une authentification pour limiter l'accès aux API sensibles.
  + Les documents, souvent sensibles,doivent être traités en respectant des politiques strictes de confidentialité et de gestion des accès.

### **3.3. Modules de Traitement (Backend)**

* **Parsers** :
  + Implémentation en Python pour extraire le texte brut à partir de fichiers PDF (via PyPDF2), DOCX (via python-docx) et TXT.
* **Extraction NER et Rule-based** :
  + Le module NER utilise un modèle open source (de la librairie spaCy) pour extraire des entités financières à partir de textes (le modèle peut être fine-tuné selon le besoin).
  + Le module rule-based est spécialisé pour les fichiers DOCX et comprend une logique spécifique pour détecter les tableaux et extraire les correspondances clés/valeurs.
* **Pipeline RAG avec Langchain et Groq** :
  + Le pipeline RAG est mis en œuvre via Langchain qui orchestre un LLMChain interrogeant un modèle LLM (pour l’exemple c’est LLama 3).
  + Selon l’opération demandée, le système génère une réponse Q&A, un résumé ou une extraction d’entités.
  + L'utilisateur doit fournir sa clé API (Openai, Groq, Azure..) pour permettre l'interrogation du modèle.

### **3.4. Communication et Orchestration**

* **Synchronous vs Asynchronous** :
  + Pour les documents de petite ou moyenne taille, le traitement peut être effectué de manière synchrone via l'API.
  + Pour des documents volumineux ou des traitements particulièrement longs, le système peut être étendu pour supporter une exécution asynchrone, via une file d'attente ou un broker de messages.
* **Intégration avec le SI**:
  + Le lecteur de documents est conçu pour être appelé par d'autres composants du SI via des API.
  + Le système fournira également des logs et des métriques de performance pour faciliter le monitoring et la maintenance.

## **4. Diagramme d'Architecture (Schéma Conceptuel)**

Voici une représentation schématique de l'architecture globale :



**5. Argumentation et Justification**

* **Modularité et Scalabilité** :  
  La séparation des modules (parsers, extraction rule-based, NER, RAG) permet d'ajouter ou de modifier des fonctionnalités sans impacter l'ensemble du système. Cela facilite également la scalabilité et l'évolution du système dans le temps.
* **Intégration API et UI** :  
  Le choix d'une API REST pour l'invocation programmée et d'une interface utilisateur interactive garantit que le système est accessible à la fois par des développeurs et par des utilisateurs finaux, répondant ainsi aux besoins du SI.
* **Flexibilité dans le Traitement** :  
  Le système est conçu pour gérer des documents de divers formats et niveaux de confidentialité. L'usage de techniques avancées (LLM via Langchain) pour les PDF verbeux assure une extraction d'entités et un traitement contextuel plus précis.
* **Sécurité et Confidentialité** :  
  Les documents étant sensibles, le système inclut des mécanismes de sécurisation des échanges (HTTPS, gestion des accès) et une gestion fine du stockage temporaire des fichiers.
* **Support Synchronous/Asynchronous** :  
  La possibilité de traiter des documents en mode synchrone ou asynchrone offre une grande flexibilité pour s'adapter aux différents volumes de données et exigences de performances.