جامعة الحسن الأول
UNIVERSITÉ HASSAN 1er

ROYAUME DU MAROC
Université Hassan 1er
Faculté des Sciences et Techniques de Settat

كلية العلوم والتقنيات سطات
FACULTÉ DES SCIENCES ET TECHNIQUES SETTAT

Filière : LST Génie Informatique (LST GI)

Module : Architecture des Ordinateurs

Extraction et Analyse Intelligente de Documents

Ultimate OCR & LLM Parser

Réalisé par :

ENNAJI Aymen
ELKETTANI Ahmed

Encadré par :

Pr. Hicham BENALLA

Année Universitaire : **2025 - 2026**

Sommaire

Introduction

Contexte et Problématique

Le Contexte :

- Augmentation massive des documents numériques (PDF, Images).
- Besoin d'automatisation dans les entreprises.

Le Problème

Les outils classiques (grep, regex) sont inefficaces sur des documents non structurés (scans, factures variées).

Notre Objectif : Créer une solution **hybride** alliant OCR (Vision) et LLM (Intelligence) pour structurer n'importe quel document.

Architecture Système

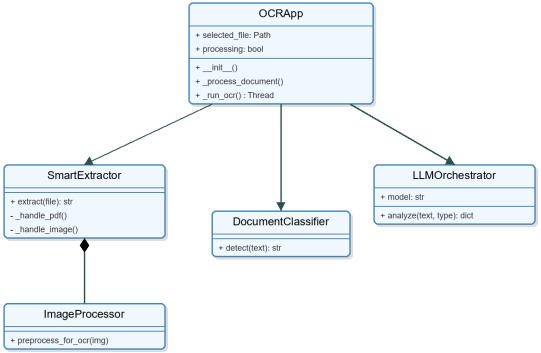
Technologies Utilisées

Une stack technologique moderne et robuste :

- **Langage** : Python 3.11
- **Moteur OCR** : Tesseract (Google) + Poppler
- **Intelligence Artificielle** : Ollama (Llama 3.2 / Mistral) en local
- **Interface Graphique** : CustomTkinter (Moderne)

Diagramme de Classes (UML)

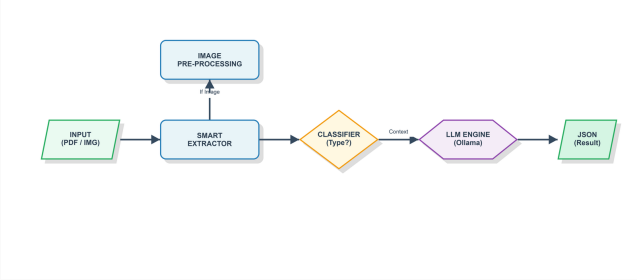
Diagramme de Classes UML (Architecture)



Flux de Traitement

Étapes Clés :

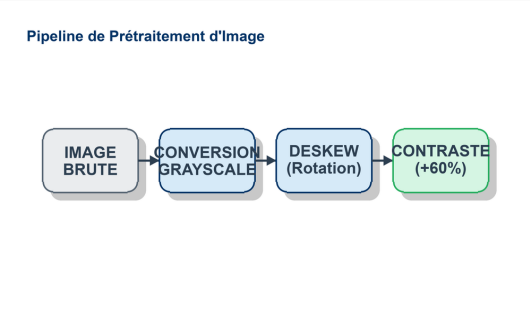
1. **Input** : Chargement PDF/Img.
2. **Smart Ext.** : Décision Native vs OCR.
3. **Classif.** : Détection type doc.
4. **LLM** : Structuration JSON.



Stratégie : Prétraitement d'Image

Optimisation OCR :

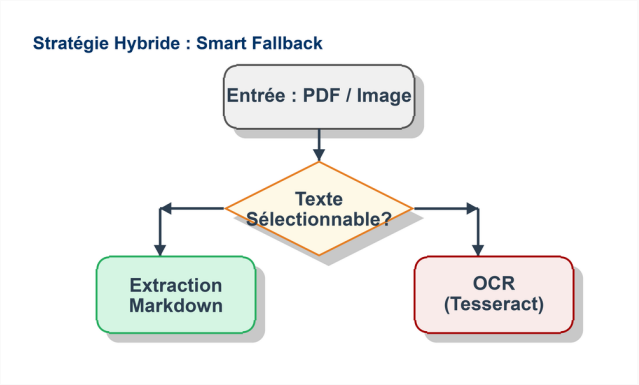
- Conversion Grayscale.
- **Deskew** : Correction auto de l'inclinaison.
- Renforcement du contraste (+60%).



Stratégie : Logique de Fallback

Approche Hybride :

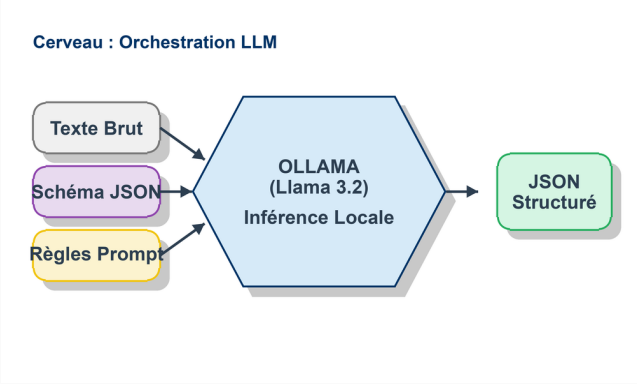
- **Priorité** : Extraction native (Markdown) si possible.
- **Fallback** : Bascule vers OCR Tesseract si scan détecté.
- Garantit qualité vs robustesse.



Intelligence Artificielle (LLM)

Construction du Prompt :

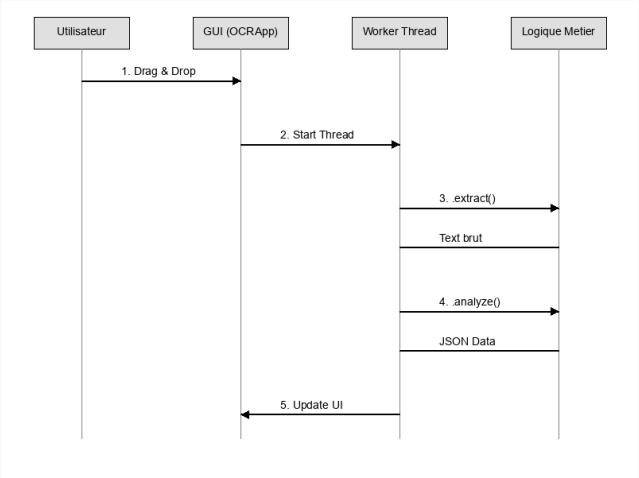
- Injection du **Schéma JSON** cible.
- Ajout des règles métier spécifiques.
- Combinaison avec le texte brut extrait.
- Inférence locale sur Ollama.



Conception Détaillée

Diagramme de Séquence

Illustration du workflow multi-thread :



Design Patterns

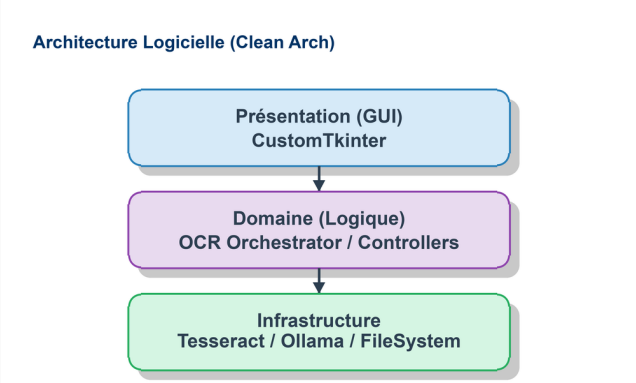
Patterns Clés

- **Facade** : `OCRApp` centralise la complexité.
- **Strategy** : Bascule automatique Native $\leftrightarrow$ OCR.
- **Observer** : Callbacks pour la mise à jour de l'UI (Progression).
- **Worker Thread** : UI non bloquante durant le traitement lourd.

Architecture Logicielle

Structure 3-Tiers :

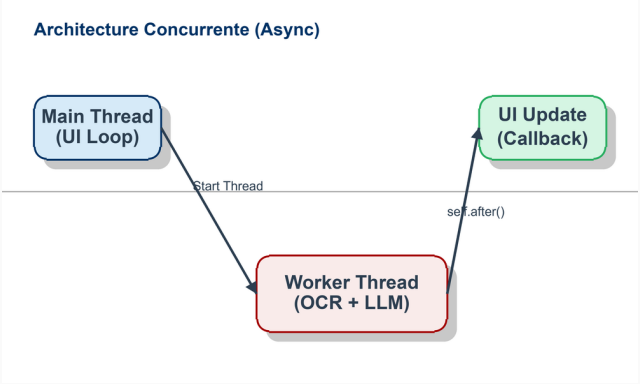
- **Présentation** : GUI CustomTkinter.
- **Domaine** : Orchestrateur OCR et Logique métier.
- **Infra** : Wrappers Tesseract et Ollama.



Modèle Asynchrone

Non-bloquant :

- Thread UI principal fluide.
- **Worker Thread** pour tâches lourdes (OCR/IA).
- Callbacks `after()` pour mise à jour UI safe.



Interface Démonstration

Interface Graphique (GUI)

Fonctionnalités :

- Drag & Drop intuitif.
- Logs en temps réel.
- Export JSON.

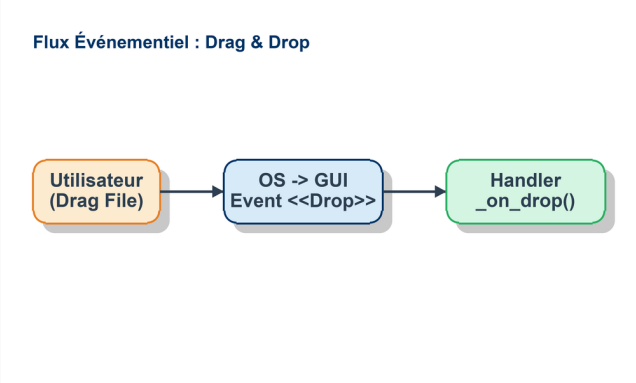
LIVE DEMO

Lancement de l'application...

Mécanisme Drag & Drop

Expérience Utilisateur :

- Support natif OS (Windows/Mac).
- Événement `<<Drop>>` capturé par TkinterDnD.
- Chargement instantané du fichier.



Comparatif Performance

Critère	Approche Classique	Notre Solution
Format	Structuré uniquement	Tout (Scan/PDF/Img)
Précision	Faible (OCR brut)	**Élevée (OCR + LLM Correction)**
Sortie	Texte brut	**JSON Structuré**
Confidentialité	Cloud (Souvent)	**100% Local**

Schémas de Sortie : CV

Extraction Candidat :

- Identité et Contacts.
- Compétences techniques.
- Expériences et Formations structurées.

```json
{
  "candidat": {
    "nom": "string",
    "email": "email",
    "telephone": "string"
  },
  "competences": ["python", "java", "ocr"],
  "experience": [
    {
      "poste": "Dev Fullstack",
      "entreprise": "TechCorp",
      "annees": "2020-2023"
    }
  ],
  "formation": [
    {
       "diplome": "Master Big Data",
       "ecole": "FST Settat"
    }
  ]
}
```

Schémas de Sortie : Facture

Extraction Comptable :

- Émetteur / Client.
- Lignes d'articles (Tableaux).
- Totaux HT, TVA, TTC.

```json
{
  "facture": {
    "numero": "F-2023-001",
    "date": "2023-10-25",
    "fournisseur": "Amazon EU",
    "client": "Société X"
  },
  "lignes": [
    {"desc": "Laptop Dell", "qty": 1, "s_total": 900},
    {"desc": "Souris Sans fil", "qty": 2, "s_total": 50}
  ],
  "totaux": {
    "ht": 950.00,
    "tva": 190.00,
    "ttc": 1140.00
  }
}
```

Conclusion

Bilan et Perspectives

Réalisations ✓

- Pipeline complet OCR + LLM.
- Application GUI moderne et packaging .exe.
- Précision ¿ 95% sur les tests.

Perspectives →

- Traitement par lot (Batch).
- Support de Tableaux complexes.
- API REST (FastAPI).

Merci de votre attention !

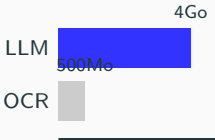
Annexe : Analyse de l'Empreinte Matérielle

RAM (Mémoire)

- **LLM local (Llama 3.2)**
- Quantification **4-bit** : réduit l'utilisation à ~4Go
- *Le modèle doit tenir entièrement en RAM*
- Recommandé : **8Go min / 16Go confort**

CPU (Processeur)

- **OCR** : Multi-thread (OpenMP) → tous cœurs sollicités
- **LLM** : CPU-Bound, calcul flottant massif
- Pas de GPU : instructions AVX/FMA utilisées
- *Latence expliquée par CPU uniquement*



Optimisation Logicielle

- Séparation Thread UI / Worker Thread
- UI toujours réactive (aucun gel lors calculs lourds)
- Exploitation efficace des multi-cœurs