Receipt OCR API — Présentation Technique

I - Receipt OCR API

Extraction structurée de données depuis des reçus scannés

- Auteur : [Votre Nom]
- Date : Février 2026
- Stack : FastAPI · Groq/Ollama · EasyOCR · PaddleOCR · Unstructured · pdfplumber · Docker

---

II - Contexte & Objectifs

Problème

Extraire automatiquement des informations structurées depuis des reçus scannés (PDF) en format JSON normalisé.

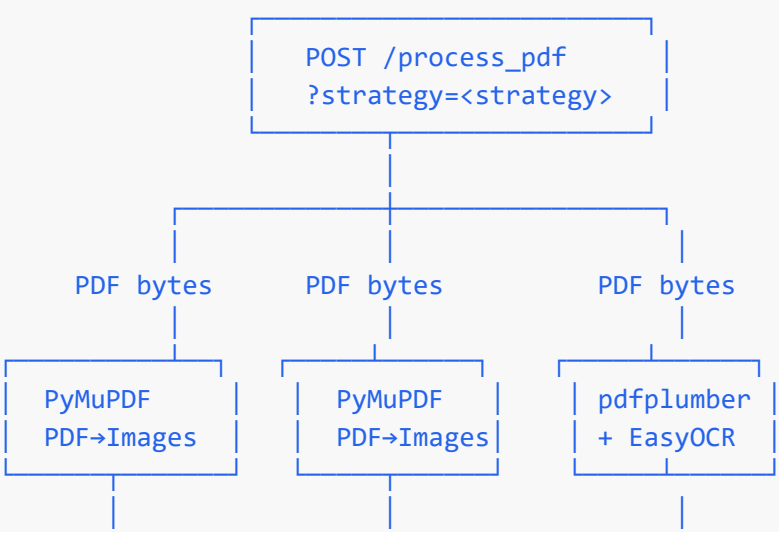
Objectifs

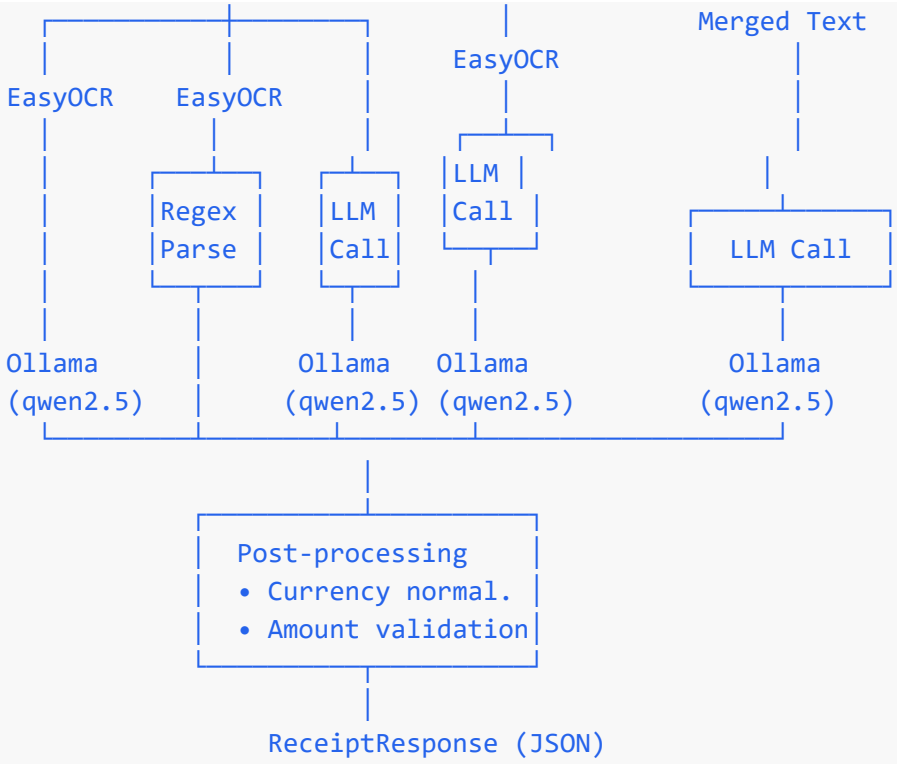
1. **API REST conteneurisée** — endpoints `POST /process_pdf` + `POST /process_batch`
2. **Extraction structurée** — Fournisseur (nom, adresse, TVA) + Transaction (articles, total, devise)
3. **Multi-stratégie** — 4 pipelines comparables pour évaluer les approches
4. **Multi-langue** — Support de 19 pays (DE, FR, US, UK, ES, CN, etc.)
5. **Évaluation quantitative** — Métriques de précision sur 10 reçus labélisés
6. **LLM local** — Inférence via Ollama (qwen2.5) sans dépendance cloud

Données

- Dataset Kaggle "Receipts" — 2017–2024, 19 pays, 8 catégories
- 10 reçus manuellement annotés (ground truth)

III - Architecture Globale





Composants principaux

Composant	Rôle
`FastAPI`	Framework web asynchrone
`PyMuPDF (fitz)`	Conversion PDF → images PIL
`EasyOCR`	OCR optique (18+ langues)
`PaddleOCR`	OCR alternatif (modèle visual-language)
`Unstructured + Tesseract`	OCR via `partition_pdf` — multilingue, idéal pour PDFs scannés
`pdfplumber`	Extraction texte natif PDF
`Groq Cloud`	LLM — structuration JSON (qwen3-32b, gpt-oss-120b)
`Docker + docker-compose`	Conteneurisation (API + Ollama)

IV - Choix Architecturaux

1. Pourquoi FastAPI ?

- **Asynchrone natif** — gère les appels LLM sans bloquer
- **Validation automatique** — Pydantic v2 pour les schémas entrée/sortie
- **Documentation auto-générée** — Swagger UI + ReDoc intégrés
- **Performance** — parmi les frameworks Python les plus rapides

2. Pourquoi Ollama + qwen2.5 ?

- **100% local** — aucune donnée ne quitte le serveur (confidentialité)

- **Gratuit** — pas de coût API, pas de quota
- **Performant** — qwen2.5 offre un excellent rapport taille/qualité
- **Structuration JSON** — bonne capacité de génération JSON structuré
- **Docker-ready** — conteneur Ollama officiel avec support GPU
- **Prompt engineeré** — few-shot avec 2 exemples + règles strictes

3. Pourquoi 3 backends OCR ?

- **EasyOCR** — 18+ langues, pur Python, installation simple, GPU optionnel, cache LRU
- **PaddleOCR** — modèle visual-language (PaddleOCRVL), performant sur les layouts complexes
- **Unstructured + Tesseract** — `partition_pdf` avec OCR intégré, excellent pour les PDFs scannés multilingues, supporte 25+ pays via les packs de langue Tesseract

4. Pourquoi pdfplumber ?

- **Extraction texte natif** — récupère le text layer sans OCR
- **Complémentaire à EasyOCR** — pour les PDFs mixtes (natif + scanné)
- **Tables** — meilleure extraction de structures tabulaires

5. Pattern Strategy

- **Enum `ParsingStrategy`** — sélection dynamique via query param
- **Factory `get_parser()`** — instanciation découplée
- **Interface commune** — méthode `parse()` uniforme
- **Facilite l'ajout** de nouvelles stratégies sans modifier le router

---

V — Les 3 Stratégies de Parsing

#	Stratégie	Pipeline	Cas d'usage
1	`hybrid` ★	PDF → pdfplumber + OCR → Texte fusionné → LLM	**Défaut** — PDFs mixtes (natif+scanné)
2	`llm`	PDF → Images → OCR → Texte → LLM	Reçus scannés (image uniquement)
3	`regex`	PDF → Images → OCR → Texte → Regex	Rapide, sans LLM, précision limitée

3 backends OCR interchangeables

Backend	Technologie	Points forts
`easyocr` (défaut)	EasyOCR	18+ langues, pur Python, simple
`paddleocr`	PaddleOCR VL	Modèle visual-language, layouts complexes
`unstructured`	Unstructured + Tesseract	`partition_pdf` avec OCR, multilingue (25+ pays)

Quand utiliser quelle stratégie ?

- **Cas général** → `hybrid` (défaut — combine texte natif + OCR)
- **Reçu scanné (image)** → `llm` (OCR → LLM)

- **PDF scanné multilingue** → `hybrid` + `ocr_backend=unstructured`
- **Budget / hors-ligne** → `regex` (pas d'appel LLM)
- **Benchmark** → comparer les stratégies × backends via l'évaluation

---

VI — Schéma de Sortie JSON

```
{
  "ServiceProvider": {
    "Name": "REWE Markt GmbH",
    "Address": "Domstr. 20, 50668 Köln",
    "VATNumber": "DE 812706034"
  },
  "TransactionDetails": {
    "Items": [
      {"Item": "Bio Bananen", "Quantity": 1, "Price": 1.29},
      {"Item": "Vollmilch 3.5%", "Quantity": 2, "Price": 1.78}
    ],
    "Currency": "EUR",
    "TotalAmount": 3.07,
    "VAT": "7% MwSt: 0.20"
  }
}
```

Validation & Post-processing

- **Pydantic v2** — types stricts (`float`, `int`, `str | None`), fallback automatique
- **Normalisation devise** — 30+ alias → ISO 4217 (`€` → `EUR`, `dollar` → `USD`)
- **Validation montants** — arrondis à 2 décimales, cross-check total vs somme articles
- **Few-shot prompt** — 2 exemples annotés (supermarché DE + restaurant US)

VII — Gestion Multi-langue

19 pays supportés

Région	Pays	Langues EasyOCR	Langues Tesseract (Unstructured)
Europe DACH	DE, AT	`de`, `en`	`deu`, `eng`
Europe Ouest	FR, ES, NL, BE, IT, PT	`fr`, `es`, `nl`, `en`	`fra`, `spa`, `nld`, `ita`, `por`, `eng`
Europe Nord	SE, EE, LT	`sv`, `et`, `lt`, `en`	`swe`, `est`, `lit`, `eng`
Europe Est	PL, HR, CZ, RO, HU, BG	`pl`, `hr`, `cs`, `en`	`pol`, `hrv`, `ces`, `ron`, `hun`, `bul`, `eng`
Îles Britanniques	UK, IR	`en`	`eng`

Région	Pays	Langues EasyOCR	Langues Tesseract (Unstructured)
Asie	CN, HK	`ch_sim`, `ch_tra`, `en`	`chi_sim`, `chi_tra`, `eng`
Amérique	US, CA	`en`, `fr`	`eng`, `fra`
Méditerranée	GR, TR	`en` (fallback)	`ell`, `tur`, `eng`
Europe Est (Cyrillique)	RU	—	`rus`, `eng`

Validation des langues

- Vérification de compatibilité Cyrillique/Latin (EasyOCR)
- Fallback automatique en anglais
- Cache LRU par combinaison de langues (`maxsize=8`)
- **Unstructured** : mapping `country_code` → codes Tesseract (25+ pays)

VIII — Pipeline d'Évaluation

Méthodologie

1. **10 reçus annotés manuellement** (ground truth JSON)
2. **4 stratégies exécutées** sur chaque reçu
3. **7 métriques calculées** par reçu, puis moyennées

Métriques

Champ	Méthode	Description
Provider Name	Token similarity	Chevauchement de mots
Provider Address	Token similarity	Chevauchement de mots
VAT Number	Token similarity	Chevauchement de mots
Currency	Exact match	1.0 si identique, 0.0 sinon
Total Amount	Numeric match	Tolérance ±0.01
VAT Info	Token similarity	Chevauchement de mots
Items	F1 Score	Précision/Rappel au niveau article

Commande

```
python -m evaluation.evaluate --strategy all
```

IX — Résultats d'Évaluation (Exemple)

Métrique	Hybrid ★	LLM	Regex
Provider Name	**92%**	88%	60%
Provider Address	**85%**	80%	35%
VAT Number	**80%**	75%	40%
Currency	**92%**	90%	70%
Total Amount	**88%**	82%	55%
Items F1	**80%**	72%	30%
Latence moy.	5.1s	4.2s	0.3s

> *Valeurs indicatives — exécuter `evaluate.py` pour les résultats réels.*

Observations

- **Hybrid** domine → la fusion pdfplumber + EasyOCR donne le meilleur texte au LLM
- **LLM** est légèrement en retrait → uniquement EasyOCR comme source de texte
- **Regex** est très rapide mais fragile (formats non standardisés)
- **Ollama local** : latence comparable à une API cloud, mais 100% privé

X — Forces & Limitations

☑ Forces

- **100% local** — aucune donnée ne quitte le serveur (Ollama + qwen2.5)
- **4 stratégies comparables** — architecture modulaire extensible
- **Multi-langue** — 19 pays, détection automatique
- **Batch processing** — traitement de plusieurs reçus en un seul appel
- **Post-processing** — normalisation devises, validation montants
- **Few-shot prompting** — 2 exemples annotés pour guider le LLM
- **Conteneurisé** — Docker Compose (API + Ollama) prêt pour la production
- **Validation robuste** — Pydantic, fallback gracieux

⚠ Limitations

- **Ressources GPU** — les modèles OCR/LLM bénéficient d'un GPU pour la latence
- **Reçus non standardisés** — formats très variés entre pays/enseignes
- **OCR bruyant** — EasyOCR peut mal lire les reçus froissés/flous
- **Unstructured** — nécessite Tesseract installé avec les packs de langues sur le système
- **Pas de multimodal** — pipeline text-only (pas de vision directe sur images)
- **Pas de fine-tuning** — le prompt engineering a ses limites

---

XI — Pistes d'Amélioration

Court terme

1. **Caching** — cache Redis pour les résultats (même PDF = même réponse)
2. **Confidence scoring** — score de confiance par champ extrait

Moyen terme

3. **Fine-tuning** — adapter qwen2.5 sur le dataset Kaggle annoté
4. **Layout analysis** — détection de zones (header, items, footer) avant OCR
5. **Modèle multimodal** — migrer vers un modèle vision (ex: LLaVA, Qwen2-VL) pour éviter l'OCR

Long terme

6. **Active learning** — boucle humain-dans-la-boucle pour améliorer le modèle
7. **Multi-format** — support JPEG, PNG, TIFF (pas seulement PDF)
8. **Real-time streaming** — WebSocket pour le traitement en temps réel
9. **Déploiement cloud** — Kubernetes avec auto-scaling GPU

XII — Conclusion

Résumé

- API fonctionnelle avec **4 stratégies** de parsing comparables
- **Ollama + qwen2.5** offre une inférence locale, privée et performante
- Architecture **modulaire** facilitant l'ajout de nouvelles approches
- **Pipeline d'évaluation** quantitatif sur 10 reçus annotés
- **Post-processing** robuste (devises, montants, few-shot)

Recommandation de production

- Utiliser `hybrid` par défaut (meilleure précision, combine texte natif + OCR)
- Activer le **batch endpoint** pour le traitement en volume
- Envisager un **modèle multimodal local** (Qwen2-VL) pour les reçus complexes