UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Sujet – Extraction et traitement de données du format PDF (analyse et conception)

Travail pratique (phase 1)

présenté à

Adam Sébastien

par

Yannick Turpin, Timothé Leduc, Camilien Provencher, Mohamed Lamine Messaoudene

2024-GEN1423 - Génie logiciel

Saint-Jérôme

25 mars 2025

Annexe

Objectif

Selon le devis-- L'objectif de ce travail pratique est de concevoir, d'implémenter et de valider une solution logicielle permettant l'extraction automatisée de données à partir de fichiers PDF de réquisition fournis par différents clients. Ces fichiers contiennent du texte, des images de profils de matrices, ainsi que des étiquettes associées à des paramètres industriels pour l'extrusion d'aluminium. Le but sera de générer automatiquement les structures de données attendues à partir des fichiers PDF fournis avec l'énoncé.

Voici les artéfacts qui seront produits par notre équipe :

- Exigences **fonctionnelles** et **non-fonctionnelles**.
- Identification et justification des **pilotes architecturaux** (attributs de qualité).
- Description des **patrons, styles et tactiques de conception** appliqués.

- Description des **modules logiciels** utilisés et des **structures de la solution** (modules, composants et connecteurs, affectation / allocation).
- **Diagrammes UML** (statiques et dynamiques) pour modéliser la solution
- Ce rapport d'analyse et de conception qui permettra de décrire les artéfacts et justifier nos décisions avec un rapport de qualité.

Exigences fonctionnelles et non-fonctionnelles

Exigences fonctionnelles

a. Extraction de données depuis les PDF

Description
L'application doit pouvoir extraire avec précision le texte, les images et les infos des fichiers PDF fournis.
Extraire les parties spécifiques du document en se basant sur des repères comme les titres et les zones de texte.

b. Transformation et structuration des données

Description
Organiser les informations extraites dans des modèles définis.
Convertir les données brutes en format utilisable (exemple, JSON).

c. Traitement des images et OCR

Description
Utiliser des techniques de reconnaissance de caractères (OCR) pour transformer le texte contenu dans les images en données exploitables.
Extraire et traiter les images pour identifier et classer les profils et autres éléments graphiques.

d. Gestion des erreurs et des données incomplètes

Description
Repérer et signaler les erreurs pendant l'extraction (par exemple, PDF mal formaté ou données manquantes).
Mettre en place des vérifications pour s'assurer que les données extraites sont cohérentes et précises.

Exigences non-fonctionnelles

Description
Performance 1. L'application doit pouvoir traiter et extraire les informations d'un PDF rapidement, même pour des fichiers volumineux ou complexes.
Précision / fiabilité 1. L'extraction doit être très précise pour minimiser les erreurs (avec un taux d'extraction proche de 100 %). 2. Elle doit aussi être robuste face aux différences de format et de contenu des PDF.
Extensibilité / modularité : 1. La solution doit être conçue de façon modulaire pour permettre l'ajout de nouvelles fonctionnalités ou la modification de fonctionnalités existantes.

Évolutivité / maintenance 1. Le code et la documentation doivent être clairs pour faciliter la maintenance et les évolutions futures. 2. Il faut respecter les bonnes pratiques du génie logiciel (tests unitaires, documentation, etc.)

Identification et justification des pilotes architecturaux

Dans ce projet, nous avons choisi plusieurs attributs de qualité qui vont guider nos choix architecturaux pour la solution d'extraction de données depuis des PDF. Ces attributs sont essentiels pour que le système réponde bien aux besoins. Voici les principaux :

Performance :
Le système doit être capable de traiter rapidement les PDF, même lorsqu'ils sont volumineux ou complexes. Nous utiliserons donc la bibliothèque **PyMuPDF** pour l'extraction de texte et d'images, **PyTesseract** pour la reconnaissance optique de texte, **OpenCV** pour le traitement d'image et la librairie **json** standard de Python pour la sérialisation en format JSON. Ces bibliothèques souvent utilisées dans ce contexte et seront adaptées pour ce projet. Elles sont compatibles avec le langage **Python**, notre langage principal pour le projet.

Fiabilité :
L'extraction des données doit avoir une très faible marge d'erreur, de sorte que les informations récupérées soient majoritairement correctes. Nous voulons après tout un logiciel précis qui peut faire sa tâche principale de manière fiable.

Modularité :
La solution sera organisée en plusieurs modules (par exemple, un module pour l'extraction du texte, un pour le traitement OCR, un pour la structuration des données et un pour la gestion des erreurs, voir **Modules logiciels utilisés**). Cette approche simplifie la maintenance, permet de tester chaque partie séparément et de distribuer facilement les tâches lors de la phase 2 du projet.

Extensibilité :
Le système doit pouvoir évoluer et intégrer de nouvelles fonctionnalités ou remplacer certains composants sans nécessiter une réécriture complète. Cela permet de s'adapter aux futurs besoins ou aux évolutions technologiques sans repartir de zéro.

Maintenance et évolutivité :
Un code bien structuré, clair et documenté est essentiel pour ajouter les mises à jour dans le futur, si on veut. En choisissant cette approche, on réduit le temps nécessaire à la maintenance et on garantit que le système pourra s'adapter aux changements d'exigences. C'est aussi un aspect qui contribuera à l'extensibilité.

Sécurité et confidentialité :

Vu que les PDF peuvent contenir des informations sensibles, il sera nécessaire de protéger les données. Un chiffrement avec un mot de passe est une simple mesure qui permettra de garantir la confidentialité des données. La librairie **PyCryptodome** implémente plusieurs algorithmes cryptographiques, assurant une implémentation sécuritaire, rapide et conforme aux standards. L'algorithme AES permet d'encrypter les données, et l'algorithme SHA-256 permet de hacher le mot de passe, le hash résultant pouvant être utilisé comme clé AES.

Patrons, styles et tactiques de conception

1. Style Architecture en Couches

Nous avons choisi ce style car il permet de diviser la solution en modules clairement séparés, qui ont une responsabilité clairement définie.

Extraction de texte : Récupérer le contenu brut des PDF.
Traitement OCR : Il faudra convertir le texte présent dans les images en données exploitables.
Structuration des données : Organiser les informations dans les modèles prédéfinis (Piece, Tool, etc.)
Gestion des erreurs : Surveiller et traiter les éventuels problèmes rencontrés lors de l'extraction et du traitement.

Le lien avec le projet est que cette approche facilite le développement en permettant aux membres de l'équipe de se concentrer sur des parties spécifiques. Elle rend également les tests unitaires plus simples, car chaque module peut être testé indépendamment, garantissant ainsi une meilleure qualité du code et une maintenance plus aisée.

2. Patron Stratégie

Nous avons choisi ce patron car il permet de faire une sélection de méthodes d'extraction OCR et de choisir celle qui est la mieux adaptée. Cela signifie que nous pouvons facilement passer d'un moteur OCR à un autre qui est plus adapté à nos besoins.

Le lien avec le projet est que cette flexibilité est nécessaire pour s'attaquer à différent types de documents (en restant dans le format PDF). Cela permet de tester plusieurs approches d'extraction.

3. Patron Façade

Nous avons choisi ce patron car il offre une interface simple pour accéder aux fonctionnalités complexes de la solution, comme l'extraction, le traitement et la structuration des données. La façade masque la complexité des différents modules internes, rendant le système plus facile à utiliser.

Le lien avec le projet est que la mise en place d'une façade simplifie l'interaction avec le système, autant pour les développeurs qui l'intègrent que pour les utilisateurs finaux (les clients) qui lancent les opérations d'extraction.

4. Tactique Chaîne des Responsabilités

Nous avons choisi cette tactique car elle permet d'attribuer une tâche spécifique à chaque module (extraction, transformation, validation, etc.), ce qui permet de réduire la complexité globale du système. Chaque module ne s'occupe que d'une partie spécifique du traitement, ce qui facilite le débogage et, encore une fois, la maintenance.

Le lien avec le projet est que cette approche permet d'isoler les problèmes et de tester chaque fonctionnalité de manière indépendante. Cela devrait également nous aider à avoir un code clair et à répartir les responsabilités au sein de l'équipe.

Modules logiciels utilisés, structures de la solution

Pour répondre aux exigences fonctionnelles et non-fonctionnelles de notre projet, nous avons choisi une **architecture modulaire en couches.** Cette approche nous permet de diviser la solution en plusieurs composants spécialisés, chacun dédié à une tâche spécifique. Cela facilite la compréhension, la maintenance et l'évolution du système tout en assurant une meilleure qualité du code.

Modules logiciels utilisés

1) **Module d'extraction de données:**
   Ce module utilise **PyMuPDF** pour localiser et extraire précisément les blocs de texte et les images dans les fichiers PDF.

2) **Module OCR:**
   Ce module utilise PyTesseract pour reconnaître du texte contenu dans des images, par exemple du texte écrit à la main.

3) **Module de transformation:**
   Ce module prend les données extraites comme du texte brut ou venant d'OCR et les transforme en structures Python correspondant aux modèles définis tel que le modèle Profile.

4) **Module de gestion des erreurs:**
   Ce module valide que toutes les données requises sont présentes et cohérentes. Il détecte aussi les anomalies comme les champs manquants ou des erreurs de l'utilisateur comme un fichier invalide.

5) **Module façade:**
   Ce module joue le rôle de coordonnateur entre tous les modules précédents. L'utilisateur interagit uniquement avec lui, et c'est lui qui orchestre l'appel à chaque module.

6) **Interface Utilisateur(CLI):**
   L'interaction avec la solution s'effectue par une interface en ligne de commande. C'est ce qui permet à l'utilisateur de sélectionner un ou plusieurs fichiers PDF à analyser, d'initier le processus d'extraction des données, d'afficher le résultat de la tâche (soit succès ou une liste d'erreurs rencontrées) directement dans le terminal, puis de sauvegarder les données en format sérialisé.

Structure en couches

Afin d'assurer une meilleure organisation, maintenabilité et évolutivité de notre solution, nous avons opté pour une architecture en couches.

1) **Couche présentation:**

Cette couche est constituée de l'interface CLI qui sert de point d'entrée pour les utilisateurs. Elle s'occupe de : récupérer les chemins de fichier PDF, afficher les erreurs ou messages et sauvegarder les fichiers JSON qui ont été produits.

2) **Couche application/Façade:**

C'est ici que se situe la façade de notre application. Celle-ci s'occupe de faire exécuter la tâche de chaque composant. Cela permettra de simplifier le système, son utilisation et la logique des traitements.

3) **Couche métier/Traitement:**

Cette couche contient le cœur de la logique :

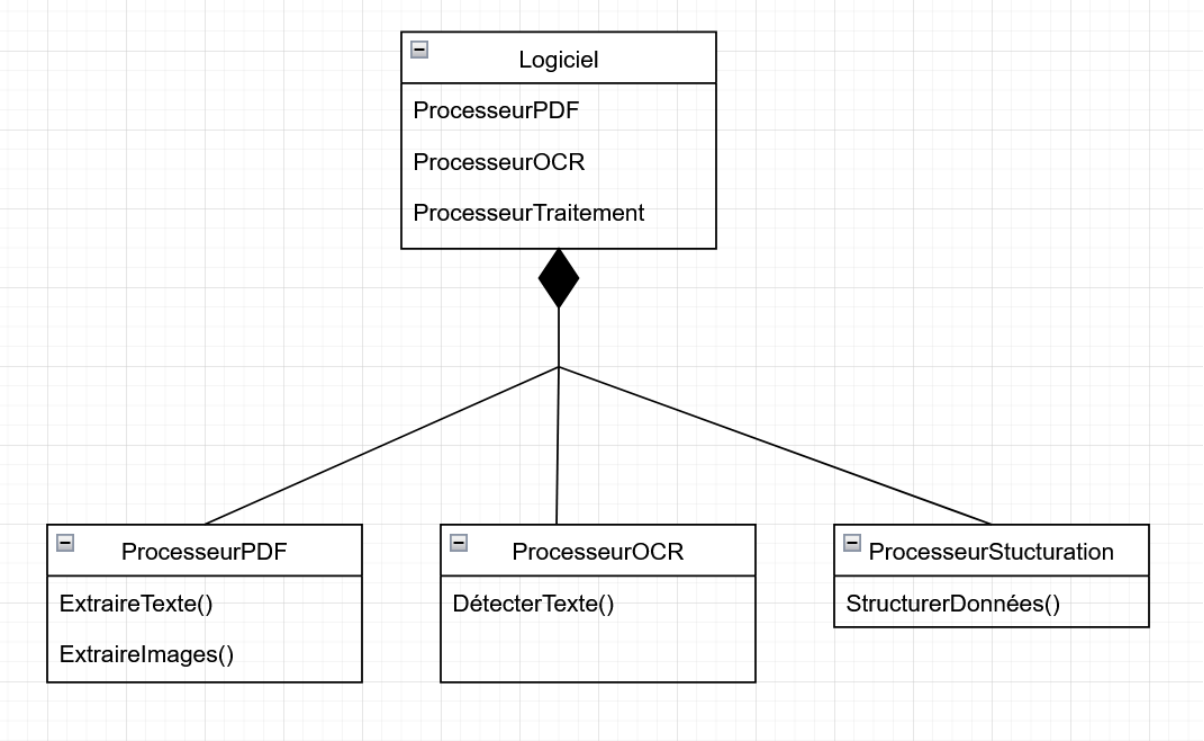
- Extraction de données du PDF
- Extraction de texte d'images
- Structuration des données

Chaque composant est indépendant et remplit une fonction spécifique.

Diagrammes UML (modélisation)

Diagramme statique

Diagramme de classe



Diagrammes dynamiques

Diagramme de cas d'utilisation

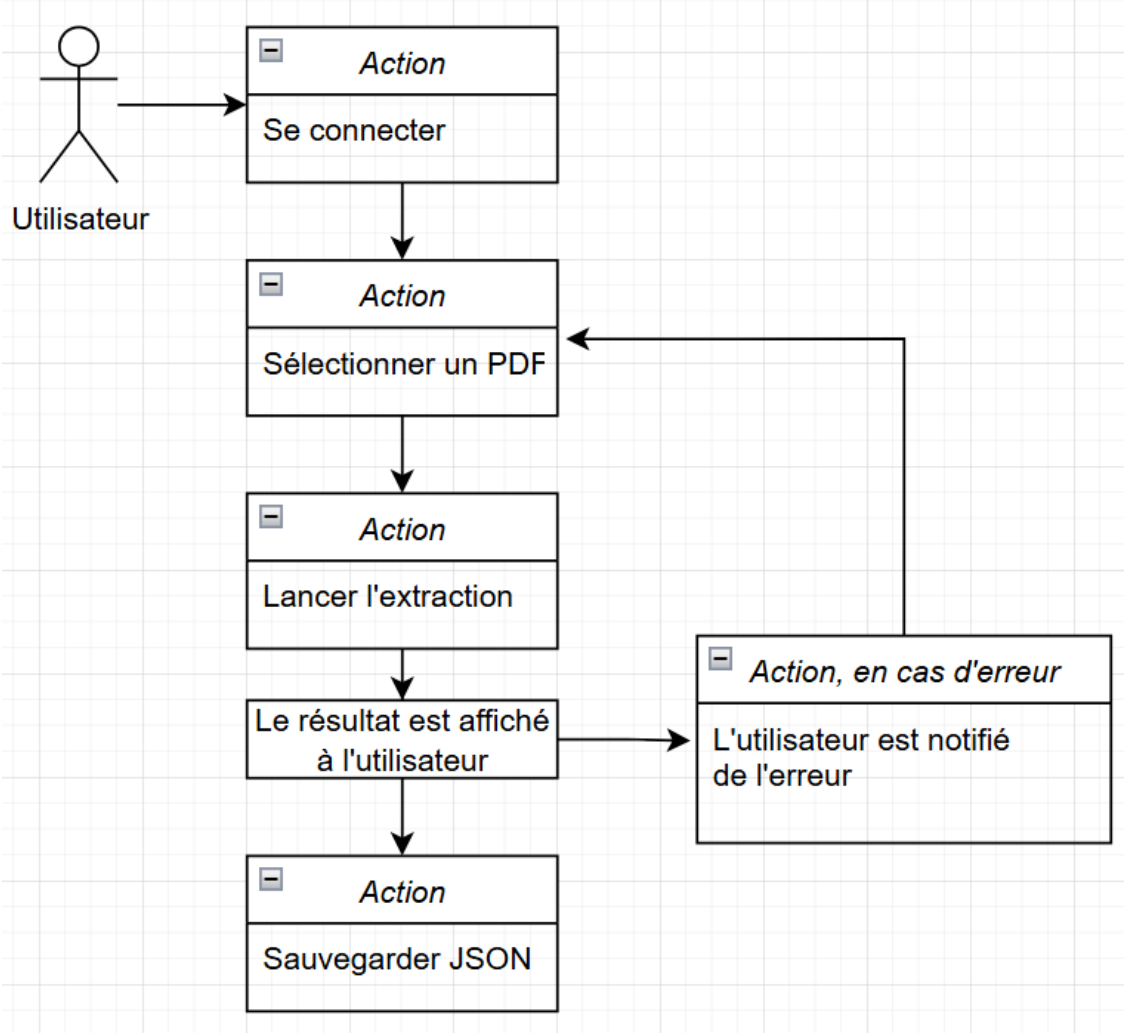


Diagramme de séquence