Document de spécification - Projet de GL52

Maxime Grolleau, Yannis Guyon, Camille Mougin et Célia Porcher

Printemps 2014

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte

Posséder un outil permettant l'automatisation du traitement des données de facturation est un besoin présent non seulement en entreprise mais également dans le cadre d'une utilisation personnelle.

L'élément clé permettant d'automatiser le traitement de factures est le concept de reconnaissance optique de caractères, ou OCR. Les premiers systèmes d'OCR ont été développés durant les années 1970 et longtemps maintenus sous secret professionnel. Des dispositifs d'OCR open-source ont finalement fait leur apparition en ligne début 2000.

Le nombre d'applications proposant l'informatisation de factures est aujourd'hui impressionnant ; cependant aucun de ces logiciels ne semble se démarquer totalement.

On peut facilement imaginer le potentiel d'une telle application si, en parallèle de sa fonction première, elle classe et interprète les données traitées en les conservant de manière anonyme.

C'est dans cet esprit que nous avons développé notre propre application de traitement de factures à partir d'images, en nous appuyant sur la librairie OCR open-source Tesseract.

1.2 Organisation du document

Ce document regroupe les différentes spécifications du projet, présentées principalement à travers le langage UML selon les méthodes apprises en cours de GL52 ce semestre. Le système est donc étudié dans un premier temps de façon globale : la granularité de l'analyse s'affine ensuite au cours du développement du projet.

Dans un premier temps, des use case globaux et leurs diagrammes de séquences associés présentent l'interface utilisateur et les traitements nécessaires du côté système. Par la suite, l'affinement des besoins apporte des précisions quant au découpage des différentes fonctions et classes en packages. Enfin, ces derniers sont définis précisément via un diagramme de classes détaillé et des diagrammes de séquence raffinés.

1.3 Définitions et abréviations

OCR : Optical Character Recognition
Traduction d'une image de texte imprimé ou manuscrit en fichier de texte.

Chapitre 2

Partie 1 - Analyse et spécification des besoins

2.1 Objectifs de l'application, environnement, utilisateurs

2.1.1 Objectif principal

Le principal objectif de l'application est la collecte et l'informatisation d'un maximum de données pouvant être contenues dans une facture ou un ticket de caisse.

2.1.2 Contexte

L'application est destinée à un ou plusieurs utilisateur. Les documents à analyser peuvent être introduits sous format image (fichiers PNG, JPG, ...), PDF, ou éventuellement HTML.

2.2 Contraintes de développement et performances

2.2.1 Limites en termes de terminaux et transactions simultanées

Le nombre maximum de terminaux n'est pas pertinent dans ce projet. L'intégralité des calculs est effectuée du côté client, et la base de données est stockée localement. Si cette dernière est distante, le nombre de requêtes étant très limité, une somme importante de clients n'est pas un souci.

Le nombre d'accès à la base de données (partie potentiellement commune aux clients) est très réduit. De ce fait, un nombre maximum de transactions simultanées à la base de données de 1 n'est pas un problème.

2.2.2 Nombre de fichiers et leurs tailles

Pour chaque facture / ticket de caisse considéré, il y a un fichier image (jusqu'à 10 Mo), un fichier texte contenant les données récupérées (200 Ko maximum) et un autre contenant les modifications apportées par l'utilisateur (taille inférieure au premier fichier texte). Un modèle de facture / ticket de caisse doit être aussi présent pour chaque type de document ; stocké sous forme textuelle, sa taille ne dépasse pas 500 Ko. Au total, en moyenne, pour une somme de 1000 factures, un espace de stockage de 5 Go doit être alloué.

2.2.3 Temps de réponse souhaité

Les accès à la base de données ne sont pas cruciaux en termes de durée. La lecture et l'écriture ne doivent pas être très lentes, mais un temps de réponse élevé (une seconde maxi-

mum) reste convenable. Cependant, la partie extraction et affichage des données ne doit pas être mal optimisée, car les technologies utilisées (OCR) peuvent être rapidement gourmandes en temps et capacité de calcul si elles sont poussives.

2.2.4 Fiabilité et tolérance aux fautes

La reconnaissance optique de caractères ne peut pas donner de résultats fiables pour n'importe quelle entrée. De ce fait, la partie vérification des erreurs est majoritairement réalisée par l'utilisateur, via la phase de correction et d'enregistrement dans la base de données. L'image dont les données sont extraites est aussi stockée, laissant la possibilité d'une vérification ultérieure.

Lorsqu'une urgence survient, le logiciel doit être en mesure de prévenir l'utilisateur et de ne pas transmettre de données erronées.

2.2.5 Sécurité

Si les données présentes dans les factures sont sensibles, la mise en place de protocoles de transmission sécurisés et de chiffrage des données stockées peut être réalisée. Certaines fonctions (notamment récupération et/ou édition d'informations présentes dans la base de données) doivent pouvoir être limitées à certains utilisateurs.

Comme écrit ci-dessus, certains utilisateurs peuvent n'avoir besoin que de l'affichage de certaines données. L'accès à d'autres et la modification doivent pouvoir être contrôlées, soit par des fonctions restreintes en fonction de l'utilisateur, soit par des enregistrements des activités et des vérifications de ces dernières.

Si les problèmes de sécurité discutés ci-dessus sont présents, un système de mot de passe ou de carte magnétique doit pouvoir être instauré afin de respecter les contraintes citées précédemment.

2.2.6 Utilisation de standards en ce qui concerne les méthodes, outils et langages de développement.

Le logiciel fourni doit pouvoir être portable et fonctionnel. Il doit être réutilisable et donc modulaire. Un environnement de travail à licence permissive doit être adopté, ainsi que des langages de développement libres d'utilisation. Développé de manière portable, le logiciel devra être exécutable depuis une machine Windows ou Unix (Linux ou Mac).

2.3 Analyse des besoins

Les principales fonctions du système sont le chargement, l'extraction et la modification des données présentes dans les factures et les tickets de caisses ainsi que l'enregistrement.

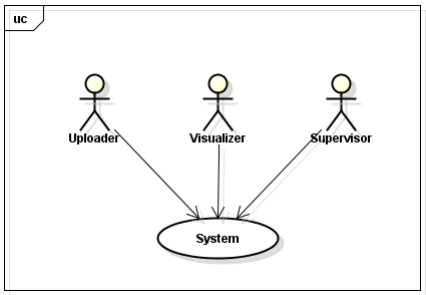

FIGURE 2.1 – *Diagramme de cas d'utilisation*

UID : UC01

Name : Upload

Description : Upload an image and apply the OCR extractor to fill the appropriate fields, modify the fields and save the file.

Actors : Uploader

Initiator : Uploader

Pre-conditions : /

Post-conditions : File, extracted data and modified data saved into database.

Flow : 1.a. The uploader clicks on the button « Load Image » and choose the file.
1.b. The uploader writes the path of the image
2. The System tries to load the file and tests if it exists and if it is an image.
2.a. [File exists and is an image]
2.a.1. The system displays the image in the left part of the application
2.a.2. The uploader selects the type of document.
2.a.2. The system tries to preselect the model
2.a.2.a [The model is preselected by the system]
2.a.2.a.1.a. [The model preselected is the user choice]
2.a.3. The uploader push the button « Extract »
2.a.4. The system extracts the date from the file following the model.
2.a.4. The uploader modifies the incorrect values
2.a.4.a [Correct Modifications]
2.a.5 The uploader types a new name for the entry
2.a.6 The uploader saves the file
2.a.7 The system saves the file with the name chosen by the uploader

Alternative
2.a.2.a.1.b. [The preselected model is not the user choice]
2.a.2.a.1.b.1. The uploader selects another model

Alternative
2.a.2.b [The Model is not preselected]
2.a.2.b.1 The system shows the model lists
2.a.2.b.2 The uploader selects a model

Alternative
2.b. [File doesn't exist]
2.b.1. The uploader receives an error message « The file chosen doesn't exist».
2.b.2. Return to 1.
2.c. [File isn't an image]
2.b.1. The uploader receives an error message « The file chosen isn't an image ».
2.b.2. Return to 1.

Alternative
2.a.4.b [Incorrect modifications]
2.a.4.b.1. The uploader pushes the « Cancel all modifications » button

Exception Flow
x. The uploader push the delete button
x.1 return to 1

UID : UC02

Name : Visualize

Description : Visualize a file from the database

Actors : Visualizer

Initiator : Visualizer

Pre-conditions : /

Post-conditions : /

Flow :
1. The visualizer clicks on the « Load from database » button
2. The system shows the appropriate interface
3. The visualizer uses available filters to organize the data stored
4. The system shows the data according to the filters chosen
5. The visualizer selects the data he wants to see
6. The system shows the data selected
7. The visualizer visualizes
8. The visualizer quits

UID : UC03

Name : Supervise

Description : Consult the database to obtain statistical data

Actors : Supervisor

Initiator : Supervisor

Pre-conditions : /

Post-conditions : /

Flow :
1. The supervisor clicks on the button « Display Statistics »
2. The system calculates and shows the current statistics
3. The supervisor visualizes
4. The supervisor quits

2.4 Description des scénarios

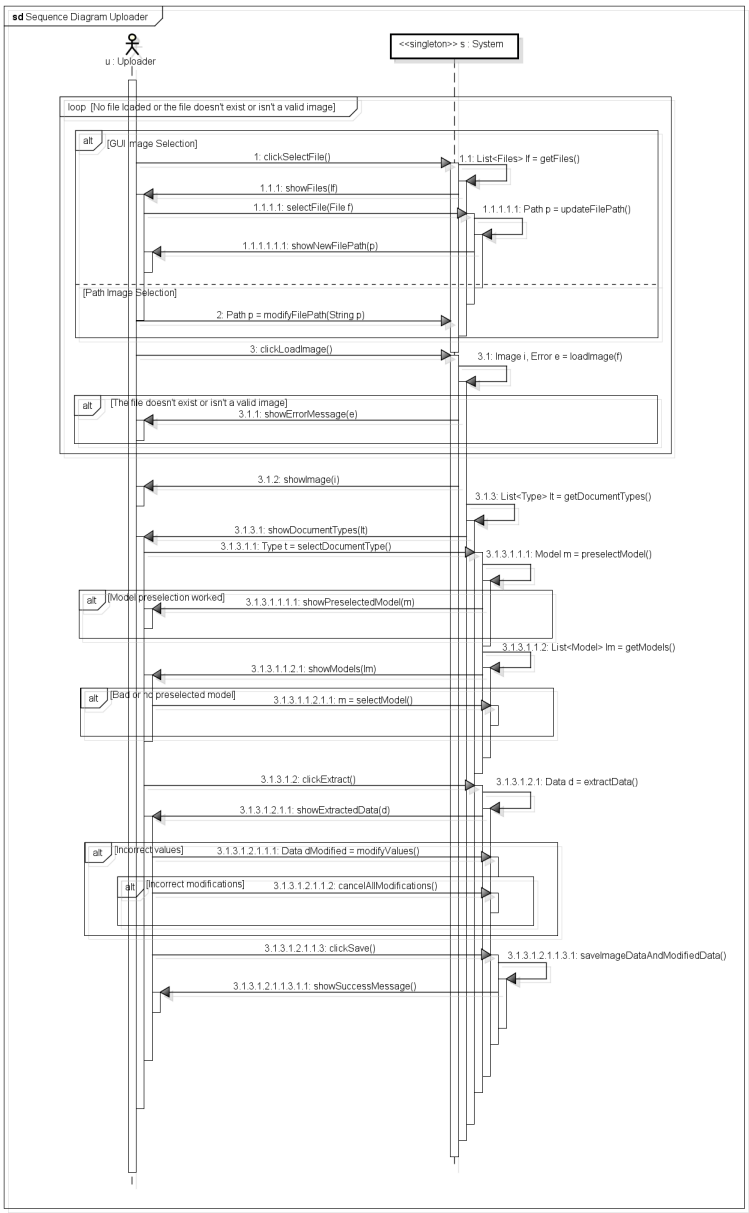

FIGURE 2.2 – *Diagramme de séquence - Chargeur*

UID : sd0
Nom : Diagramme de séquence Chargeur
Resumé : Charge une image et procède à l'extraction des données contenues pour remplir les champs adéquats, modifie les champs erronés et enregistre le document dans la base de données (image, données extraites et données modifiées).
Acteurs : Chargeur
Initiateur : Chargeur
Pré-conditions : /
Post-conditions : /

Description : Les fonctions propres au choix et vérifications du fichier getFiles(), showFiles() et selectFiles() sont prises en charge par l'interface elle-même. Elle est chargée d'afficher l'arborescence permettant la sélection du fichier à ouvrir. La fonction showNewFilePath() inscrit seulement le chemin du fichier sélectionné dans le champ correspondant. Le chemin peut également être entré manuellement.

Au chargement de l'image, une double vérification est effectuée : le fichier indiqué par le chemin existe-t-il , et le format de ce fichier est-il supporté par le programme. Si ces deux conditions sont vérifiées, l'image s'affiche ; sinon, un message d'erreur informe l'utilisateur du problème rencontré.

S'il le peut, le système pré-sélectionne un model via showPreselectedModel(). L'utilisateur peut choisir d'utiliser un autre modèle via selectModel(), et clique ensuite sur le bouton "Extract".

La fonction extractData() applique l'algorithme d'OCR grâce aux informations fournies par le modèle afin de remplir un maximum de champs. Les valeurs des champs sont affichées et peuvent potentiellement être modifiées par l'utilisateur.

L'utilisateur peut enregistrer les données extraites, annuler les modifications effectuées pour revenir aux données extraites brutes, ou modifié les valeurs présentent dans la base de donnée. Un message informe l'utilisateur du succès de l'opération.

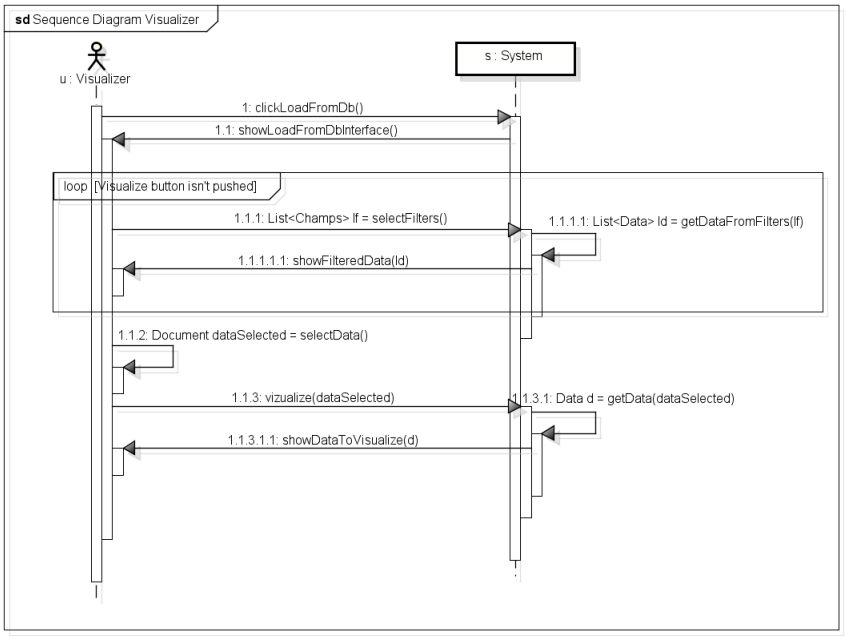

FIGURE 2.3 – *Diagramme de séquence - Visualiseur*

UID : sd1
Nom : Diagramme de séquence Visualiseur
Resumé : Permet de visualiser un fichier présent dans la base
Acteurs : Visualiseur
Initiateur : Visualiseur
Pré-conditions : /
Post-conditions : /

Description : Un clic sur le bouton "LoadFromDb" appelle la fonction showLoadFormDbInterface() qui ouvre une fenêtre permettant à l'utilisateur de choisir un fichier présent

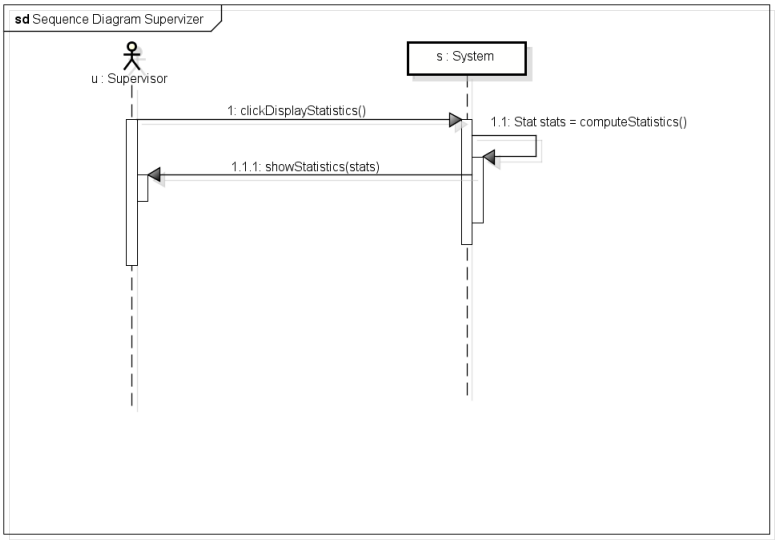

FIGURE 2.4 – *Diagramme de séquence - Superviseur*

dans la base de données. Un filtre permet d'affiner la recherche selon des critères particuliers appliqués aux champs.

UID : sd2

Nom : Diagramme de séquence Superviseur

Resumé : Permet d'afficher les statistiques obtenues à partir de l'ensemble des données récoltées.

Acteurs : Superviseur

Initiateur : Superviseur

Pré-conditions : /

Post-conditions : /

Description : Le bouton "Show Statistics" appelle la fonction clickDisplayStatistics() qui permet d'avoir des informations globales sur l'ensemble des documents analysés, voire le tracé de courbes ou de graphiques représentant l'évolution ou la répartition des données en fonction de leur type ou occurrences par exemple.

2.5 Concepts principaux et définitions

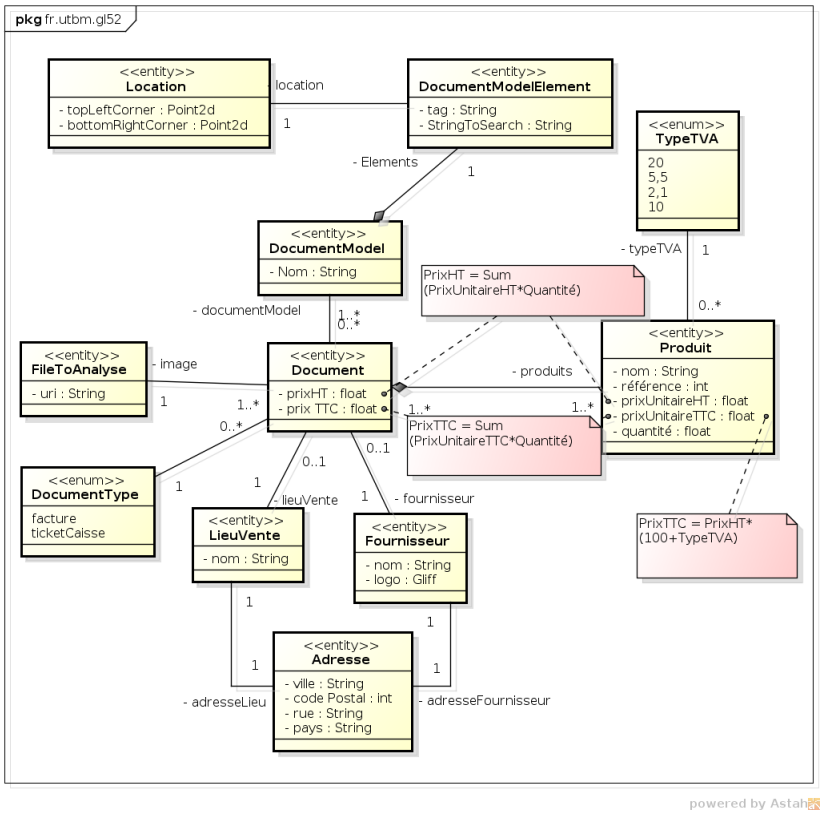

FIGURE 2.5 – *Domain Model*

Document : On appelle document l'entité enregistrée dans la base de données. Un document regroupe le fichier d'origine, l'ensemble des champs extraits par la procédure OCR, et l'ensemble des champs en tenant compte des modifications de l'utilisateur.

Modèle de facture : Un modèle de facture est composé de DocumentModelElement. Dans la majorité des cas, un modèle est propre à un prestataire.

DocumentType : Le type de document peut être soit facture, soit ticket de caisse. Il est représenté par une enum.

DocumentModelElement : Chaque documentModelElement correspond à une zone d'intérêt et possède un nom qui indique le type de données contenues dans la zone, le tag. La zone elle-même est définie par un rectangle Location et/ou un mot clé StringToSearch (la donnée recherchée sera située après le mot-clé).

FileToAnalyse : Cette entité est un fichier physiquement présent sur le disque. Il est sélectionné par l'utilisateur et est défini par son chemin (absolu ou relatif à l'application) uri.

LieuVente : Un document (tel qu'une facture ou un ticket de caisse) doit comprendre plusieurs informations, telles que l'endroit où la vente a eu lieu.

Fournisseur : Le fournisseur est défini par son nom, prénom mais son logo peut aussi être

stocké à des fins de reconnaissance automatique de modèle de document. L'émetteur de ce dernier a généralement une adresse ainsi qu'une raison sociale.

Client : Le client est le destinataire de la facture, défini par son nom, prénom et adresse.

Produit : En plus des informations uniques, un document possède obligatoirement une liste de produits achetés. Ces derniers ont chacun un nom, une référence, un prixUnitaireHT, un prixUnitaireTTC (calculé grâce au taux de TVA appliqué) ainsi qu'une quantité.

OCR : L'OCR, outil de reconnaissance de caractères, va permettre toute la gestion d'extraction de String à partir d'une image. Il est possible de limiter les zones de détection, et d'y associer une détection de mots clés.

Hibernate : Le wrapper pour Hibernate permet d'effectuer une liaison dynamique de persistance des données. Les classes dont les informations doivent être stockées dans la base de données seront converties en tables.

Controller : Partie du pattern Modèle Vue Contrôleur, le controller effectue la liaison entre l'interface utilisateur et les données.

GUI : L'interface graphique utilisateur met en place un champ d'action ergonomique, via des modules systèmes prédéfinis tels que des boutons et d'autres éléments intéractifs.

Chapitre 3

Partie 2 - Conception architecturale et détaillée

3.1 Architecture de l'application

3.1.1 Database

La persistance des données est assurée grâce à la création et à la maintenance d'une base de données. Cette dernière, structurée à l'aide du framework open-source Hibernate, stocke à la fois les modèles des factures, mais aussi le document résultant des chargements et extractions, qui sont composés comme expliqué précédemment de l'image elle-même, des données extraites par reconnaissance de caractères, et des éventuelles corrections de l'utilisateur.

3.1.2 Modèles

Ces structures de données permettent de stocker les différents modèles de factures et leurs caractéristiques. Les informations concernant le contenu et sa position dans l'image d'entrée sont définies grâce à ce package. Un modèle peut être de type Facture ou Ticket.

3.1.3 Optical Caracter Recognition (OCR)

Ce package regroupe les fonctions utilisant la librairie de reconnaissance de caractère. Un premier framework offre les fonctions premières d'extraction à partir d'une zone de l'image, en coordonnées relatives ou absolues, exprimées en pixels ou en unité. Des options permettent de rechercher sur une ou plusieurs lignes, de séparer des résultats en plusieurs champs et autres manipulations simples visant à faciliter la procédure globale.

Des méthodes plus spécifiques utilisant ce framework effectuent l'extraction de données en fonction du modèle de facture ou ticket de caisse utilisé.

3.1.4 Controller

Le controller fait le lien entre l'interface, et la partie modèle (au sens théorique MVC) de l'application, composée principalement du module de lecture OCR et de la base de données. En particulier, il écoute les actions réalisées par l'utilisateur sur l'interface et déclenche les actions nécessaires en conséquence.

3.1.5 GUI

L'interface utilisateur utilise la librairie Java Swing. Très simple d'aspect, elle est divisée verticalement en deux parties propres respectivement au document à analyser et aux données

extraites.

La partie gauche permet de rechercher un fichier, de paramétrer son type et le modèle adéquat, de visualiser le fichier une fois ouvert et enfin de lancer l'extraction des données à l'aide du bouton "Extract". La partie droite présente quant à elle l'ensemble des informations que l'on souhaite obtenir. Les données globales de la facture apparaissent sous forme de champs, et un tableau regroupe les données de chaque article. Enfin, trois boutons "Save", "Delete" et "Cancel all modifications" permettent de communiquer avec la base de données.

En haut de la fenêtre, on trouve également une barre de menu qui propose un prolongement éventuel dans le développement, avec un module d'aperçu des statistiques d'utilisation.

3.2 Conception

3.2.1 Diagramme de classe détaillé

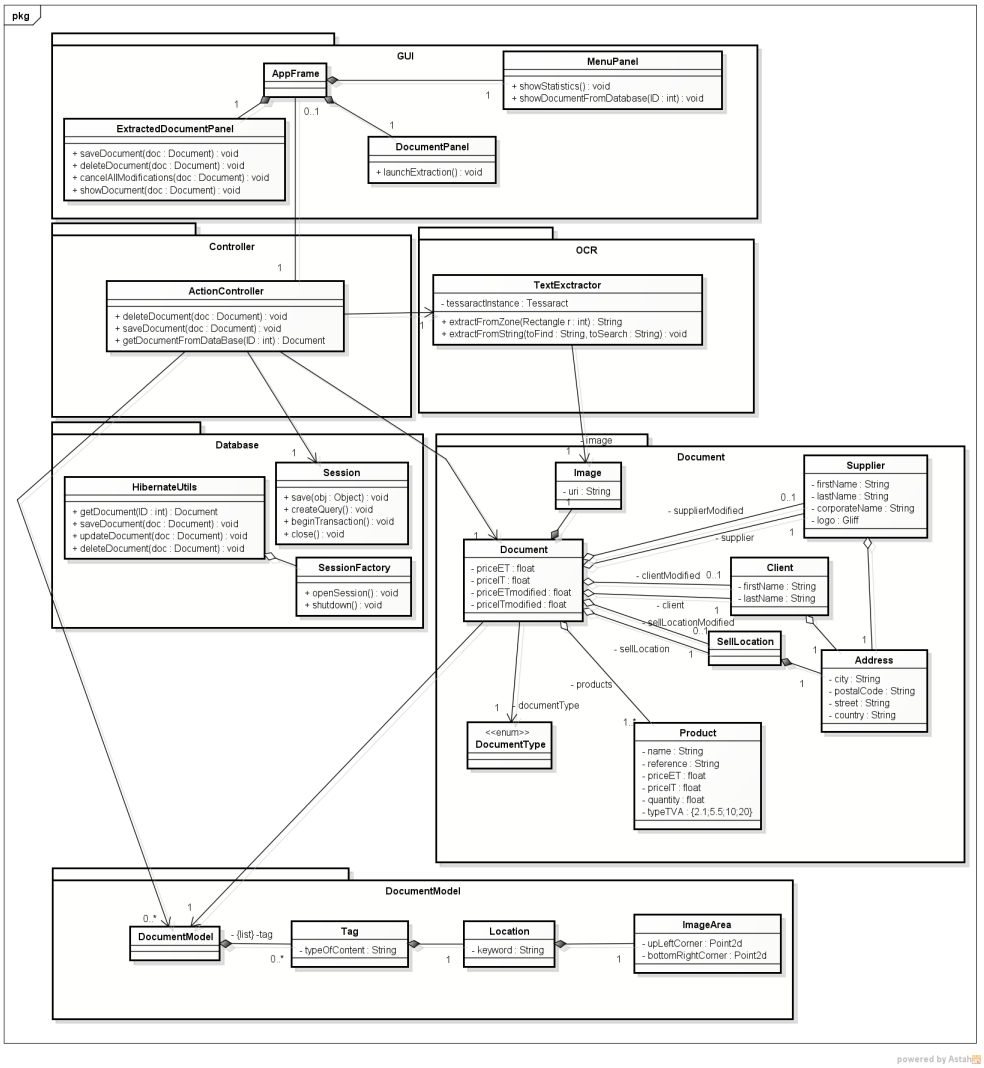

FIGURE 3.1 – *Diagramme de classe détaillé*

ActionController : L'entité ActionController a accès à tout. Elle utilise la classe TextExtractor situé dans le package OCR et une instance de DocumentModel situé dans le package DocumentModel afin d'extraire les données de l'image. Elle met à jour les valeurs du document courant et répond aux signaux envoyés par le GUI. Elle communique également avec la base de donnée du package Database pour le téléchargement et la sauvegarde d'informations.

TextExtractor : La classe TextExtractor va permettre tous les traitements à partir d'une image pour obtenir des String. Elle utilise une instance de Tesseract la librairie OCR et dispose de fonctions qui vont lui permettre de récupérer des données dans une zone de l'image, à partir d'un mot clé.

DocumentModel : La classe DocumentModel représente la manière dont on doit récupérer les données d'une image et leur attribuer un sens. Pour ce faire, il contient une liste de tags. Les tags vont correspondre à une location soit aussi une zone de l'image à traiter et un string correspondant à un mot-clé.

Document : C'est l'entité qui va permettre de regrouper tous les champs dont peut être composé une facture ou un ticket de caisse soit le DocumentType, le lieu de vente, le client, le fournisseur, les produits. Deux variables pour chacun de ces champs existeront dans le document, dans le but de stocker les données extraites mais également les données modifiées. Le document stockera aussi une image.

3.2.2 Diagramme de séquence détaillé

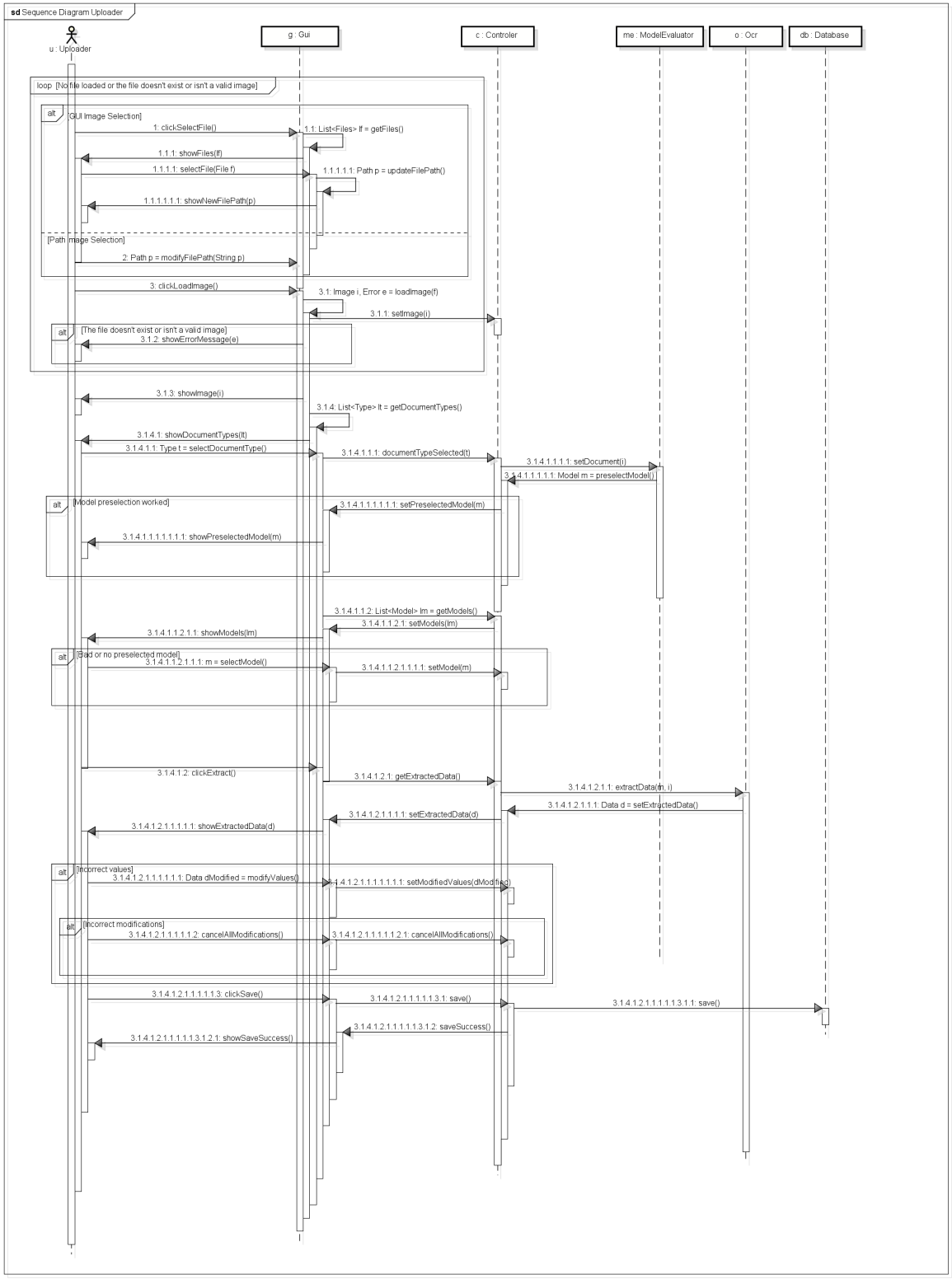

FIGURE 3.2 – *Diagramme de séquence détaillé - Uploader*

UID : sd3
Nom : Diagramme de séquence détaillé Chargeur
Resumé : Charge une image et applique l'extracteur OCR pour remplir les champs appropriés. Les champs peuvent être modifiés et enregistrés dans la base de données.
Acteurs : Chargeur
Initiateur : Chargeur
Pré-conditions : /
Post-conditions : /

Description : Les fonctions propres au choix et vérifications du fichier getFiles(), showFiles() et selectFiles() sont prises en charges par l'interface elle-même. Elle est chargée d'afficher l'arborescence permettant la sélection du fichier à ouvrir. La fonction showNewFilePath() inscrit seulement le chemin du fichier sélectionné dans le champs correspondant. Le chemin peut également être entré manuellement.

Au chargement de l'image, une double vérification est effectuée : le fichier indiqué par le chemin existe-t-il , et le format de ce fichier est-il supporté par le programme. Si ces deux conditions sont vérifiées, l'image s'affiche ; sinon, un message d'erreur informe l'utilisateur du problème rencontré.

Le système pré-sélectionne un modèle à travers le Controller via preselectModel() et la classe ModelEvaluator, et le présente à l'utilisateur grâce à la méthode showPreselectedModel() de l'interface. Celui confirme ou remplace le modèle puis clique ensuite sur le bouton "Extract".

La fonction extractData() applique l'algorithme d'OCR grâce aux informations fournies par le modèle afin de remplir un maximum de champs. Les valeurs des champs sont affichées et peuvent potentiellement être modifiées par l'utilisateur puis enregistrées. Une modification va mettre à jour les données du document correspondant contenu dans la base de données. L'utilisateur peut également enregistrer simplement le document, ou supprimer un document existant. Le package Controller envoie systématiquement les informations à enregistrer au DataBase package. Un message confirme à l'utilisateur que l'action qu'il vient d'effectuer s'est bien déroulée.

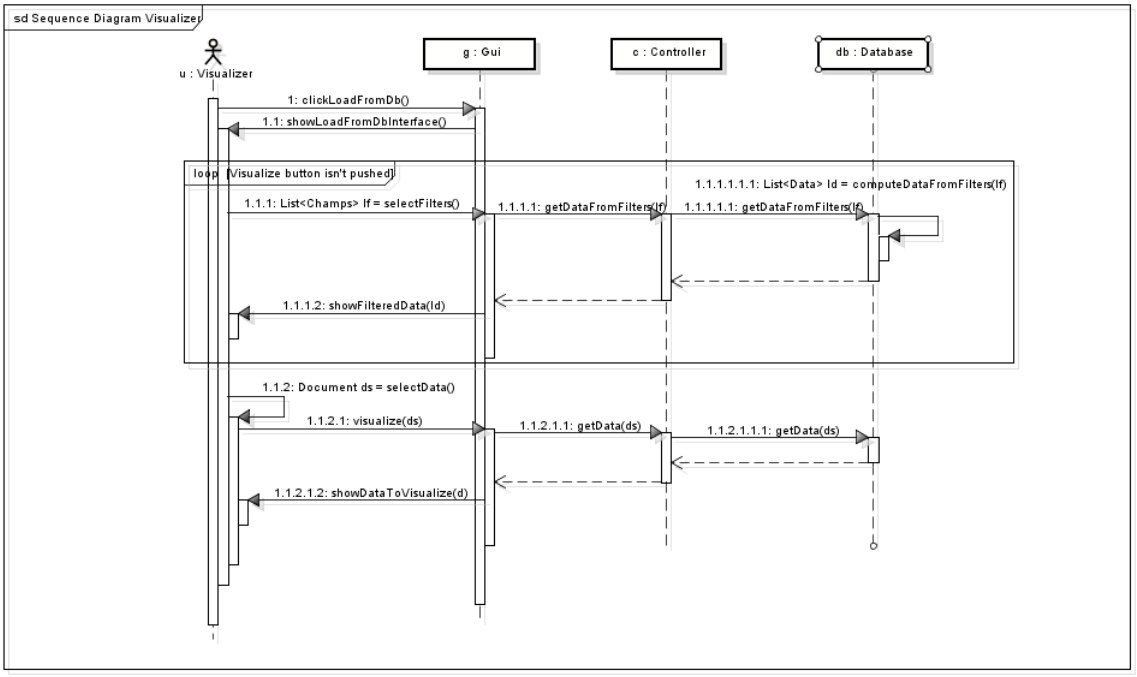

FIGURE 3.3 – *Diagramme de séquence détaillé - Visualiseur*

UID : sd4
Nom : Diagramme de séquence détaillé Visualiseur
Résumé : Affiche un document provenant de la base de données.
Acteurs : Visualiseur
Initiateur : Visualiseur
Pré-conditions : /
Post-conditions : /

Description : Lors du clic sur le bouton "Load From DB", la fonction showLoadFormFbInterface() permet à l'utilisateur de filtrer les données de la base de données afin de sélectionner un fichier spécifique à visualiser. L'accès à la base de donnée est réalisé à travers les packages GUI, Controller et Database dans un sens et dans l'autre.

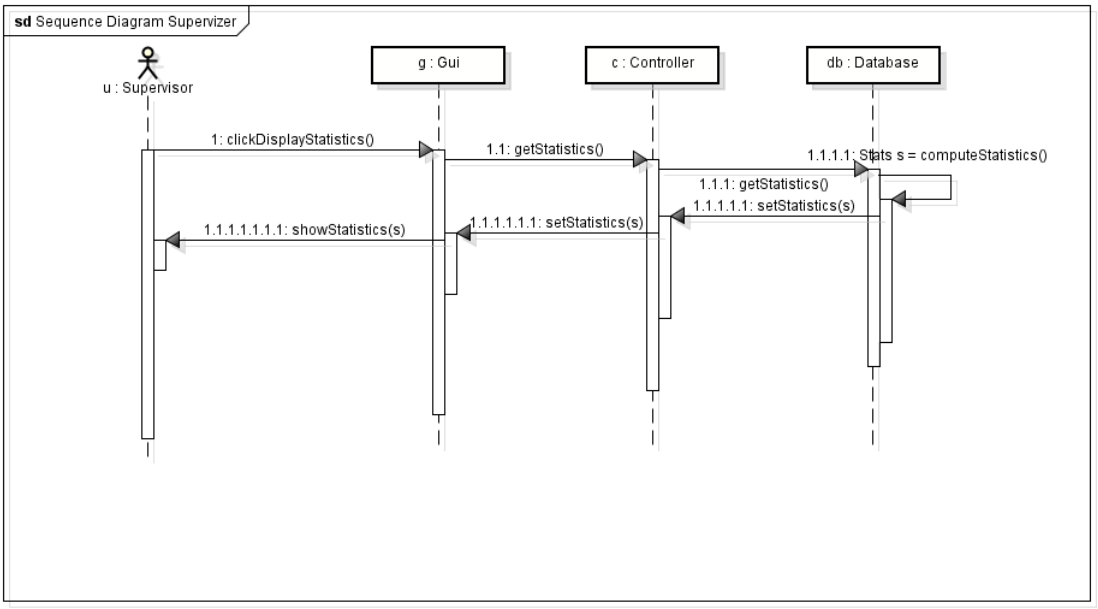

FIGURE 3.4 – *Diagramme de séquence détaillé - Superviseur*

UID : sd5
Nom : Diagramme de séquence détaillé Superviseur
Résumé : Charge une image et applique l'extracteur OCR pour remplir les champs appropriés. Les champs peuvent être modifiés et enregistrés dans la base de données.
Acteurs : Chargeur
Initiateur : Chargeur
Pré-conditions : /
Post-conditions : /
Description : La fonction clickDisplayStatistics() répond au clic de l'utilisateur cherchant à obtenir des informations sur les données stockées. La commande est transmises par le GUI package, puis le Controller package pour enfin arriver au DataBase package, après quoi le résultat parcours le chemin inverse pour permettre l'ouverture d'une fenêtre présentant les données statistiques à l'utilisateur.

Chapitre 4

Conclusion

4.1 Retour aux objectifs

L'application développée répond aux grands objectifs définis : chargement et analyse des différents formats de fichiers, visualisation et modification des informations extraites, et enfin gestion de l'enregistrement et de l'accès aux données relatives à chaque facture ou ticket de caisse. Ce document de spécification garantie la réflexion portée autour de sa conception et son développement.

Nous avons cependant dû faire face à un certain nombre de problèmes, à commencer par la conception de la structure représentative d'un document. Nous avons d'abord modélisé la structure avec un trop grand niveau de généricité ce qui rendait le processus beaucoup trop compliqué à gérer en particulier au niveau de la base de données. Etant donné que seuls deux types de documents (facture et tickets de caisse) sont traités, une grande généricité sur les champs n'est pas nécessaire.

Au niveau du processus d'OCR en lui-même, l'application est dépendante du temps d'exécution de Tesseract. En effet, environ vingt secondes sont nécessaires pour l'extraction de données d'une facture selon un modèle comprenant 4 zones.

Concernant les objectifs non atteints, nous n'avons pas réussi à développer de méthode détection automatique de modèle au chargement de la facture par manque de temps et d'idée. D'autre part, le nombre de modèles de tests créés est très réduit et doit correspondre parfaitement à la facture pour obtenir un résultat correct. L'élasticité des modèles est en effet très faible et une bonne extraction de données nécessite un modèle très précis.

4.2 Regard critique

Le projet a été développé dans le respect des consignes et objectifs donnés. Un soin particulier a été apporté aux techniques de développement : nous obtenons un projet Maven, qui utilise conjointement une base de données et l'ORM hibernate, une libraire OCR adaptée à nos besoins à travers un classe type framework, et qui fonctionne sous Linux et Windows.

La maintenance et l'évolution du projet sont donc grandement facilitées, et on peut imaginer mettre en place des solutions aux différents problèmes rencontrés : pour répondre aux difficutés de précision par exemple, un module pourrait être mis en place permettant la sélection manuelle de la zone d'articles par l'utilisateur. L'application gagnerait grandement en fiabilité avec en contrepartie une contrainte mineure pour l'utilisateur.
Si l'on veut épargner le maximum d'actions à l'utilisateur, on pourrait proposer une solution alternative avec un nombre de modèles beaucoup plus élevé et l'ajout à l'interface d'un bouton

d'enregistrement de l'image en tant que modèle avec définition des zones et reconnaissance du logo du prestataire.

Le module de statistique mériterait également d'être développé, un des grands intérêt de l'informatisation de données étant l'analyse et l'étude globales des informations récoltées.

Chapitre 5

Annexes

5.1 Annexe 1 - Modèle Conceptuel de Données

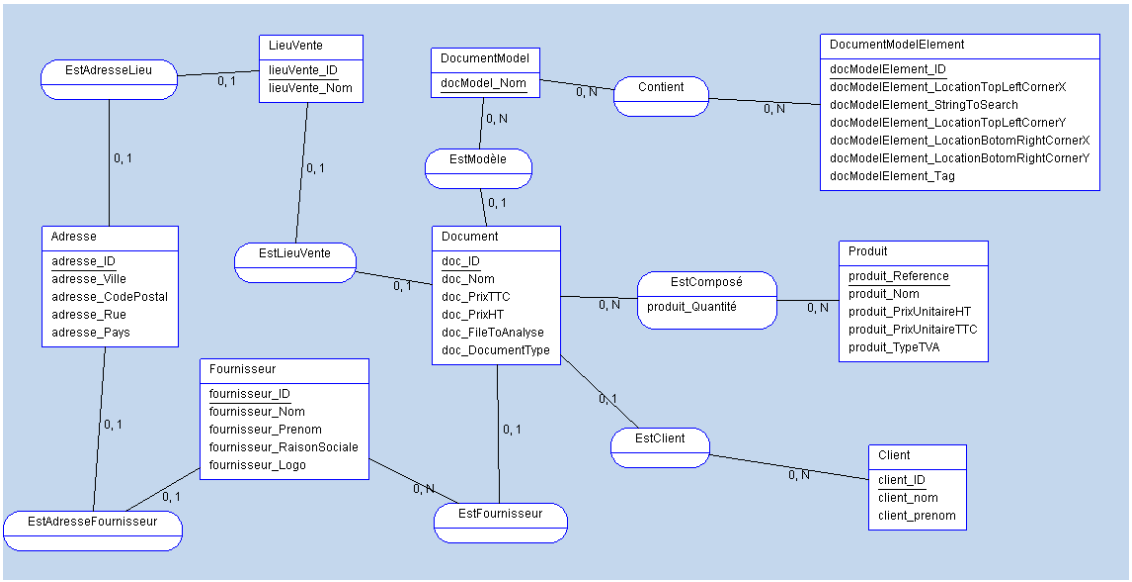

FIGURE 5.1 – *Annexe 1 - Modèle Conceptuel de Données*

5.2 Annexe 2 - Modèle Logique de Données

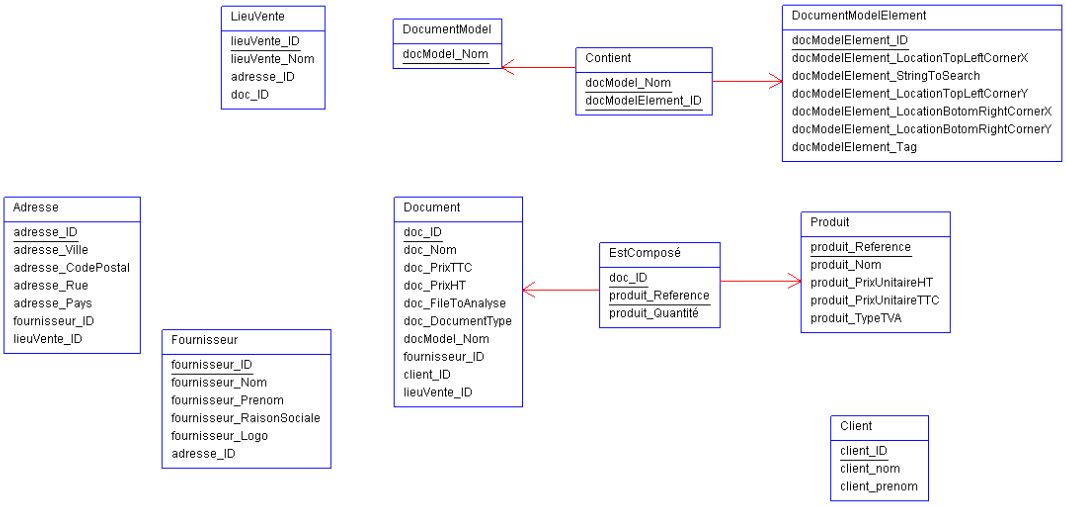

FIGURE 5.2 – *Annexe 2 - Modèle Logique de Données*