Document de spécification - Projet de GL52

Maxime Grolleau, Yannis Guyon, Camille Mougin et Célia Porcher

Printemps 2014

Table des matières

1 Introduction 3
- 1.1 Contexte . . . 3
- 1.2 Organisation du document . . . 3
- 1.3 Définitions et abréviations . . . 3

2 Partie 1 - Analyse et spécification des besoins 4
- 2.1 Objectifs de l'application, environnement, utilisateurs . . . 4
  - 2.1.1 Objectif principal . . . 4
  - 2.1.2 Contexte . . . 4
- 2.2 Contraintes de développement et performances . . . 4
  - 2.2.1 Limites en terme de terminaux et transactions simultanées . . . 4
  - 2.2.2 Nombre de fichiers et leurs tailles . . . 4
  - 2.2.3 Temps de réponse souhaité . . . 4
  - 2.2.4 Fiabilité et tolérance aux fautes . . . 5
  - 2.2.5 Sécurité . . . 5
  - 2.2.6 Utilisation de standards en ce qui concerne les méthodes, outils et langages de développement. . . . 5
- 2.3 Analyse des besoins . . . 5
- 2.4 Description des scénarios . . . 6
- 2.5 Concepts principaux et définitions . . . 8

3 Partie 2 - Conception architecturale et détaillée 10
- 3.1 Architecture de l'application . . . 10
  - 3.1.1 Database . . . 10
  - 3.1.2 Modèles . . . 10
  - 3.1.3 Optical Caracter Recognition (OCR) . . . 10
  - 3.1.4 Controller . . . 10
  - 3.1.5 GUI . . . 10
- 3.2 Conception . . . 11
  - 3.2.1 Diagramme de classe détaillé . . . 11
  - 3.2.2 Diagramme de séquence détaillé . . . 11
  - 3.2.3 MCD et MLD . . . 11

4 Annexes 12

5 Conclusion 13

Table des figures

2.1 Diagramme de cas d'utilisation 5
2.2 Diagramme de séquence - Chargeur 6
2.3 Diagramme de séquence - Visualiseur 7
2.4 Diagramme de séquence - Superviseur 8
2.5 Domain Model 9

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 Tessaract.

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 traitement 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 raffiné.

1.3 Définitions et abréviations

OCR : Optical Caracter 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 unique 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 terme 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 maximum) 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 (Linus 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.

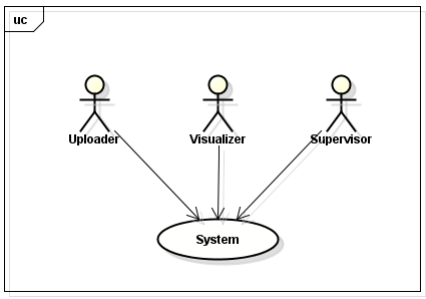

Figure 2.1 – Diagramme de cas d'utilisation

2.4 Description des scénarios

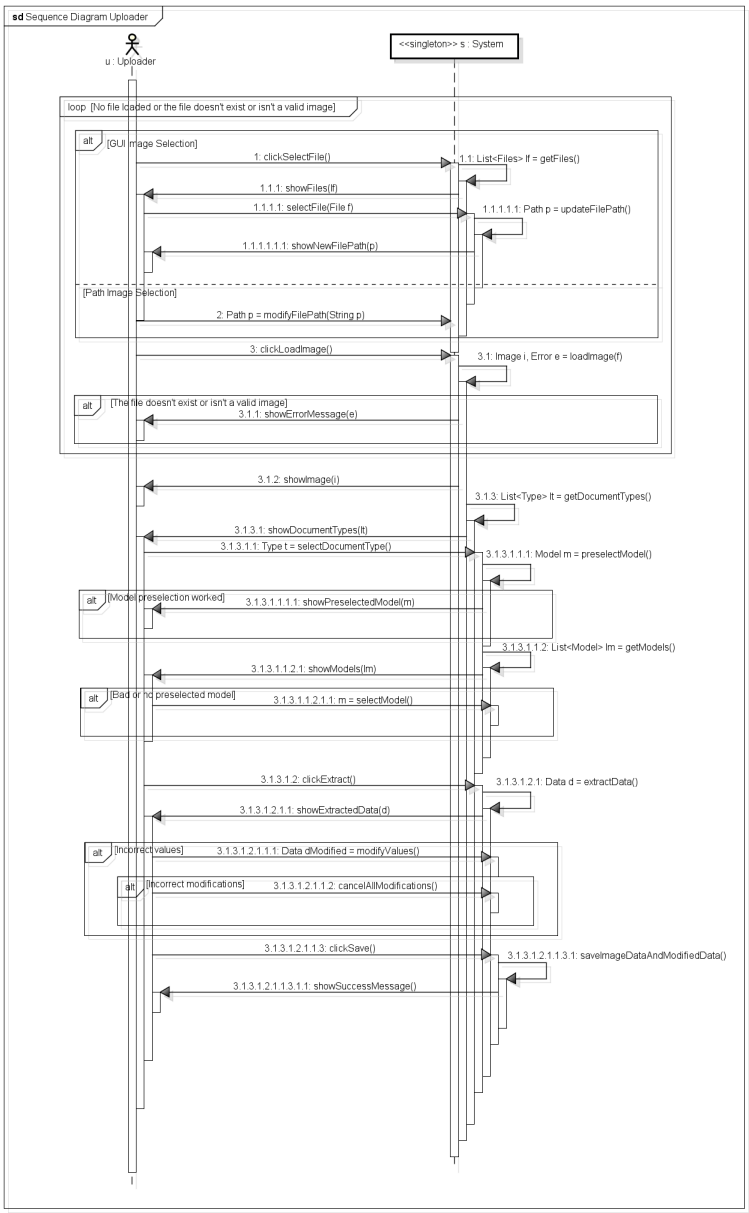

Figure 2.2 – Diagramme de séquence - Chargeur

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 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.
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.

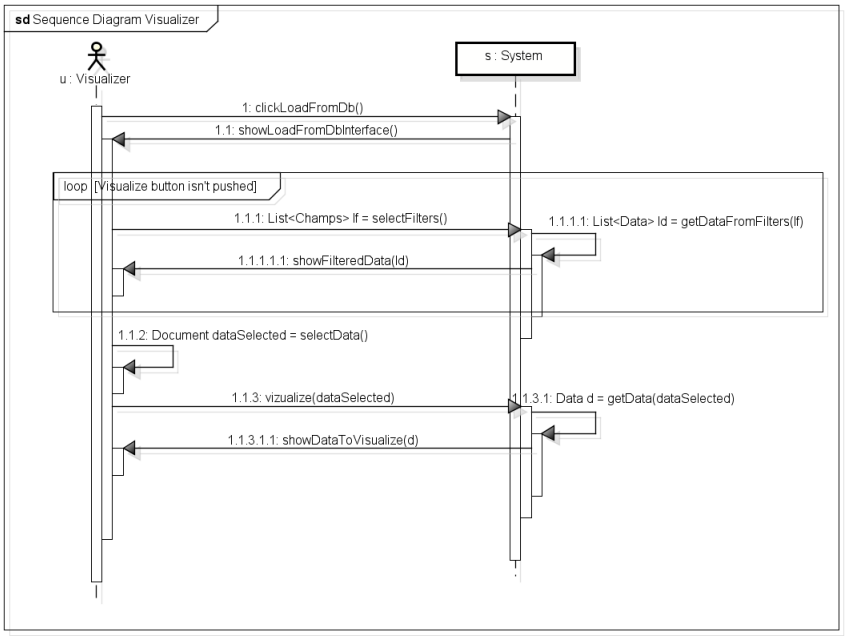

Figure 2.3 – Diagramme de séquence - Visualiseur

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 : /

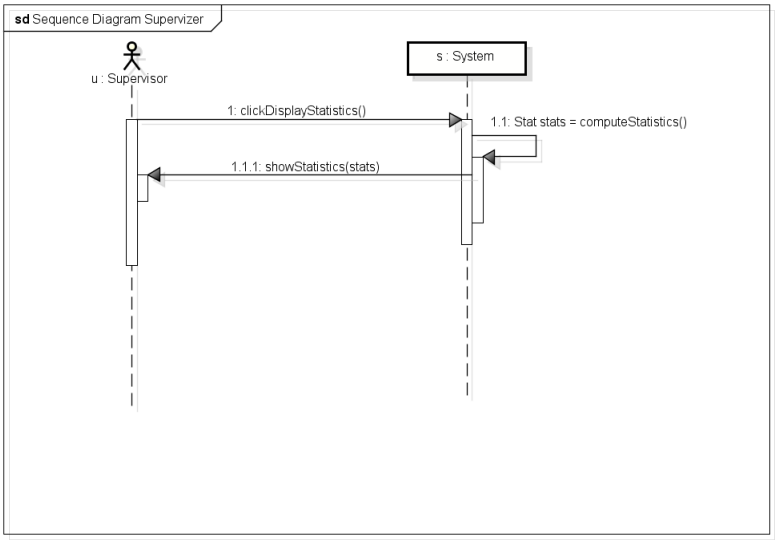

Figure 2.4 – Diagramme de séquence - Superviseur

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

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 tags. Chaque tag correspond à une zone d'intérêt et possède un nom qui indique le type de données contenues dans la zone. La zone elle-même est définie par un rectangle et/ou un mot clé (la donnée recherchée sera située après le mot-clé). Dans la majorité des cas, un modèle est propre à un prestataire.

Champ :

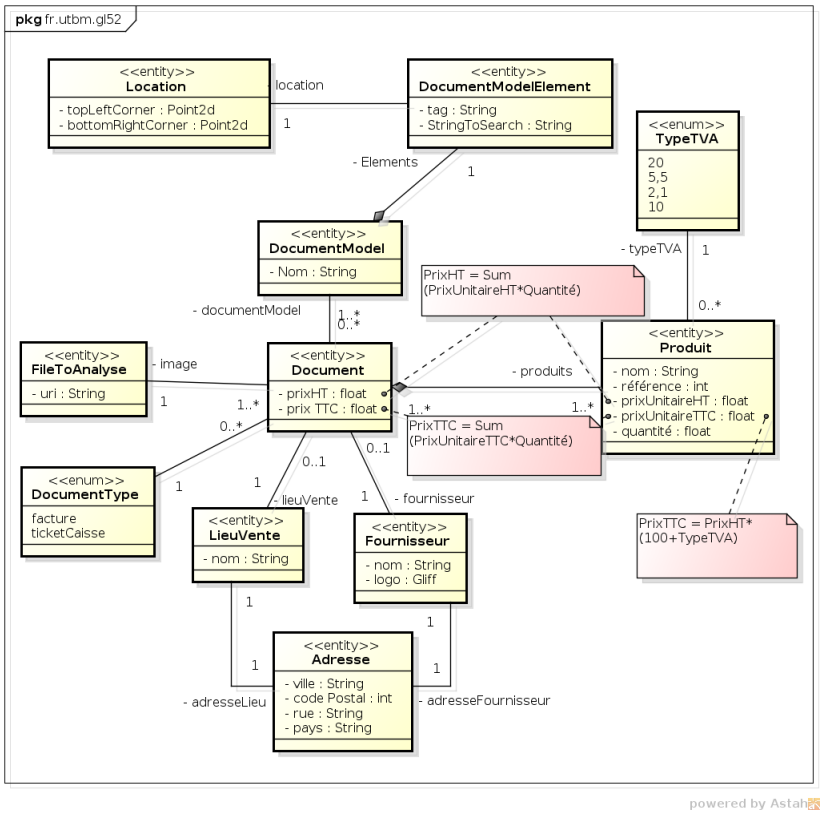

Figure 2.5 – Domain Model

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.

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

3.2.2 Diagramme de séquence détaillé

3.2.3 MCD et MLD

Chapitre 4

Annexes

Chapitre 5

Conclusion