Développement d'une application mobile en technologie Microsoft .NET et Android de gestion et d'utilisation de check-lists standardisées pour le secteur industriel

Ing. O. DEKNOP
ECAM – Bruxelles

Ce document présente l'étude et l'implémentation d'une application mobile en technologie .NET et Android destinée à faciliter le système de contrôle par check-lists des camions qui transitent sur le site de Total Petrochemicals à Feluy. L'objectif du projet est double. Il s'agit d'une part d'apporter la meilleure solution possible à une demande de Total et d'autre part d'étendre cette solution à toute autre situation qui nécessiterait un contrôle plus efficace par check-lists numériques standardisées sur équipement mobile.

Mots-clefs : applications mobiles, tablette, Android, Windows, WPF, Web Service

This paper presents the study and implementation of a mobile application .NET and Android whose objective is to facilitate and improve trucks security checks using checklists at Total Petrochemicals Feluy.
Initially, the main goal is to provide the best possible solution to a request from Total. Then the idea is to extend this solution to any situation that would require a more effective control system using standardized checklists on mobile equipment.

Keywords: mobile applications, tablet, Android, Windows, WPF, Web Service

1. Introduction

1.1. Contexte du projet

Le marché des applications mobiles en est plein boom. Le nombre d'applications mobiles téléchargées augmente de manière exponentielle chaque année. Et l'essor des tablettes informatiques n'a fait que renforcer ce secteur déjà florissant.

Dans ce contexte, de plus en plus d'entreprises du secteur industriel souhaitent investir dans le domaine et semblent intéressées par l'utilisation d'applications mobiles dans certains processus industriels.

Parmi elles, Total Petrochemical a émis le désir d'informatiser le système de contrôle des camions sur le site mais sans pour autant produire un cahier des charges précis. Cette requête de Total répond plus à un projet d'améliorations futures qu'à un besoin pressant.

Ce document présente d'une part une proposition de réponse concrète à la requête de Total et d'autre part une analyse des possibilités que pourrait offrir une application de ce type dans d'autres secteurs industriels.

1.2. L'entreprise à l'origine du projet

Pulsar est une société de services en informatique active dans le secteur depuis 1998. Sa mission est d'aider ses clients à améliorer la performance de leur entreprise avec des solutions informatiques de haut niveau. Son département consultance est chargé d'offrir les meilleurs services possibles à sa clientèle. Qualifié et expérimenté, le personnel suit la philosophie Pulsar qui privilégie la disponibilité, l'écoute active et le transfert de connaissance et de technologie.

La société est également très active dans le développement de projets sur mesures réalisés à prix fixes.

Les clients de Pulsar sont des entités de tailles variées pouvant aller de la petite PME aux grandes multinationales.

1.3. eLisa Solution logistique

C'est fort d'une expérience de plus de 15 ans dans le domaine de la logistique industrielle que Pulsar a développé eLisa, une solution informatique complète pour traiter de A à Z les réceptions et expéditions de marchandises.

De l'accueil des chauffeurs, à l'édition des documents de transport, en passant par le chargement, eLisa est l'outil indispensable de gestion opérationnelle des expéditions et réceptions, quel que soit le mode de transport. Elle permet :

- d'alléger les tâches administratives ;
- d'optimiser les chargements ;
- d'améliorer la sécurité sur site ;
- de mieux maîtriser les temps d'attente ;
- d'aider à respecter au mieux les demandes du client.

L'application a pour objectif la gestion complète des cycles de transport aussi bien de camions que de trains. Elle s'occupe entre autres :

- de l'accueil des chauffeurs ;
- de la gestion des rendez-vous ;
- des contrôles de sécurité (ADR[1]) ;
- du suivi des chargements ;
- de multiples pesées ;
- des contrôles en sortie ;
- de l'impression des documents.

Le software permet également un interfaçage très simple à un ERP[2], au contrôle d'accès, au chargement automatique, aux ponts de pesées, etc. Il assure la communication entre les différents intervenants dans l'usine : les gardes, le chargement, le bureau logistique, le service commercial. L'application convient aux petites implantations, aussi bien qu'aux installations complexes avec un important flux de véhicules.

---

[1] Accord européen relatif au transport international des marchandises dangereuses par route. Pour plus d'informations :
http://www.unece.org/trans/danger/publi/adr/adr2011/11contentse.html

[2] *Enterprise resource planning*

Voici une liste des modules de l'application. Chaque module s'intègre au noyau (*Kernel*) principal de l'application. Celui-ci est composé de la base de données centrale ainsi que des fonctionnalités standards de l'application.

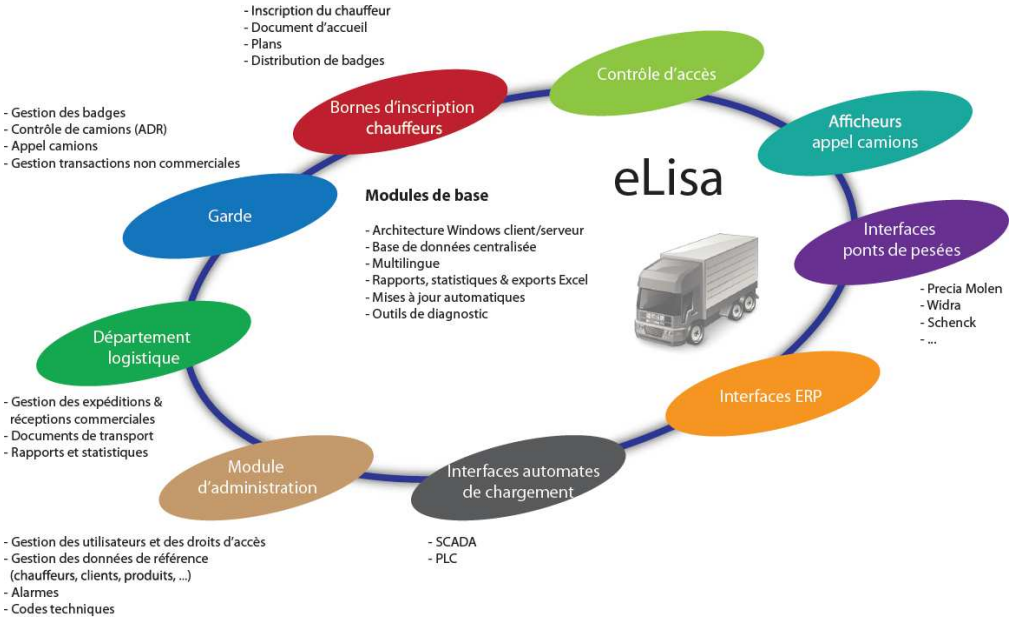


Figure 1 Modules eLisa

eLisa a déjà été implémentée chez une multitude de clients à travers l'Europe et pour différents types de transport : Aperam, Arcelor, Total Petrochemicals (Belgique) ; Total Petrochemicals (France) ; Yara (Pays-Bas) ; Yara (Allemagne) ; Yara (Italie).

2. Problématique

2.1. Processus de contrôle existant

Description générale

Total doit gérer quotidiennement la réception et l'expédition d'énormes quantités de marchandises. Plusieurs centaines de camions se présentent chaque jour à l'entrée de sites logistiques pour y être chargés ou déchargés.

Dans un premier temps, le chauffeur doit s'enregistrer via une borne interactive pour que le système puisse identifier les informations relatives

aux chauffeurs, tracteurs, remorques et commandes. Diverses interfaces entre les ERP et les équipements de terrain permettent d'automatiser au maximum l'acquisition des informations.

Avant le chargement (ou le déchargement) et pour assurer une sécurité maximale sur le site, le camion est soumis à un contrôle complet : des check-lists imprimées sont remplies afin de vérifier par une série de critères à la fois administratifs et techniques que le véhicule est prêt à être chargé (ou déchargé). Le point suivant détaillera le contenu de ces check-lists de contrôle.

Les check-lists

À l'arrivée du camion, le contrôleur imprime la check-list correspondant au type de produit transporté et à l'opération à effectuer : chargement ou déchargement. Il est important d'indiquer ici qu'aucun champ n'est pré-rempli automatiquement avant l'impression.

Le document est divisé en sections qui traitent chacune d'une opération, d'un sujet différent :

- Documents administratifs : Le chauffeur est-il en ordre de papiers ? Qui a envoyé le produit ? Où doit-il être envoyé ?
- Avant d'entrer sur site : Le camion (tracteur et remorque) est-il en ordre de contrôle technique ? Est-il assuré ? Le chauffeur possède-t-il l'équipement nécessaire ?
- Opération de chargement : Quantité à charger ? Volume disponible ? Le matériel nécessaire au chargement est-il en ordre ?
- Avant le départ : Y a-t-il eu des éclaboussures ? Les trappes sont-elles bien fermées ?

La liste des sujets/sections et questions cités ci-dessus n'est évidemment pas exhaustive. L'idée est de donner une idée du type de questions posées et de l'agencement de celles-ci.

Le document peut également inclure des photos représentant un équipement, une partie du camion ou une plaquette de code symbole ADR. Dans certains cas, une illustration peut en effet faciliter ou accélérer le contrôle.

Les réponses aux questions posées peuvent être encodées de différentes manières. On retrouvera le plus couramment des cases à cocher mais aussi des champs de texte.
En fin de document, un cachet et/ou une signature devront être apposés par le chauffeur, le contrôleur et éventuellement d'autres acteurs de la chaîne de contrôle.

Au verso de chaque check-list se trouve une liste de points numérotés correspondant chacun à une des questions du recto. Ils servent à en préciser la signification.

Notons enfin qu'il est fait usage de couleurs pour mettre en évidence différentes parties de la page et certaines questions importantes.

Le suivi du contrôle
Une fois complétées et signées, les check-lists de contrôle sont triées et rassemblées dans des classeurs. Chaque mois, un contrôleur se charge de vérifier si elles ont été bien remplies et prend note des contrevenants dans le cas contraire.

À la fin de chaque mois, les sociétés de transport prises en défaut sont prévenues et/ou sanctionnées. Aucun lien n'existe entre cette procédure de contrôle par check-lists et l'ERP de Total.

Avantages et inconvénients
Inconvénients
- aucun contrôle à l'encodage
- aucun lien avec l'ERP
- suivi peu efficace

Avantages
- facilité de l'encodage
- solution bon marché
- création facile de nouvelles check-lists

2.2. Spécifications du nouveau processus

Description générale

Les contrôles mis en place par Total ont pour but de s'assurer que chaque camion qui transite sur le site respecte les demandes du client (produit, lot, quantité) et toutes les règles de sécurité routière (PTAC[3], signalisation ADR, documents de transport, etc.)

Ces procédures sont relativement lourdes et rendent non négligeable le temps passé sur site par les chauffeurs. Or tout retard est souvent facturé par le transporteur et peut représenter un coût significatif pour l'usine. Il est donc dans l'intérêt de Total de rendre le système le plus efficace possible pour limiter ces frais au maximum.

Le souhait de Total est de remplacer l'actuelle procédure de contrôle par check-lists papier par une application mobile utilisable sur un périphérique mobile de type tablette. Ce périphérique doit posséder une connexion soit physique soit sans-fil à un serveur indépendant du système informatique général permettant la sauvegarde des réponses. Un lien entre le système « check-list » et la partie « business intelligence » de Total sera implémenté sur mesure ultérieurement.

Les check-lists doivent être claires et facilement compréhensibles par le personnel concerné. L'application mobile se doit d'être *user-friendly* et son utilisation ne doit nécessiter aucune formation ni connaissances informatiques. Un simple document d'explication doit suffire.

Une application tierce doit permettre de créer des check-lists personnalisées. L'accès à cette application ne sera cependant pas tout de suite accordé à Total. Dans un premier temps et pour des raisons surtout commerciales, Pulsar se chargera du design des check-lists. Un suivi technique de support ou de maintenance pour un client représente toujours un intérêt financier pour une société de services informatiques.

Type d'équipement

Le choix de l'équipement qui remplacera les actuelles check-lists papier chez Total est crucial. L'appareil devra répondre à une série de critères

[3] Poids total autorisé en charge.

destinés à faciliter et accélérer le contrôle des camions : dimensions, poids, OS, type de processeur, mémoire, affichage, stockage, moyens de communication, etc.

Lien avec le système informatique de l'entreprise

L'utilisation de check-lists imprimées et l'absence de lien avec le système informatique central de Total constituent un point négatif assez important du système de contrôle actuel. La vérification mensuelle des check-lists représente une perte de temps considérable et retarde largement le suivi des camions.

Le nouveau système se doit de corriger ce manque d'efficacité en proposant une solution qui permettra de gérer le contrôle des camions en temps réel. Si un véhicule n'est pas en ordre vis-à-vis d'un des points du formulaire de contrôle, le système doit en être averti instantanément.

Pour établir ce lien, l'application de gestion de check-lists sera en contact par l'intermédiaire d'un *Web Service* avec la solution logistique eLisa déjà mise en place chez Total. L'utilité de ce lien sera double.

La première utilité du lien est, comme expliqué ci-dessus, d'assurer un meilleur suivi des camions. Chaque véhicule qui circule sur site est identifié dans eLisa. L'idée est d'ajouter à ces informations d'identification, des données complémentaires à propos du contrôle de sécurité. Le système pourra alors générer des alertes en cas de problème, assurer un suivi des camions sur site, établir un historique des camions et des sociétés de transport qui les emploient ou encore faire une analyse statistique des manquements à la sécurité les plus fréquents.

Un deuxième avantage de ce lien est la possibilité de pré-remplir automatiquement certaines informations d'identification du camion au moment du contrôle et de faciliter le travail du contrôleur.

Avantages et inconvénients

Inconvénients :

- Changement radical par rapport aux procédures de contrôles actuelles
- besoin d'équiper et de former des contrôleurs

- risque de résistance au changement puisque la nouvelle application contrôle le travail des contrôleurs

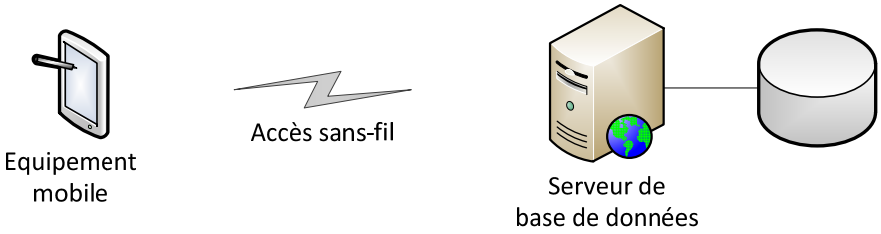
Avantages :

- Résultats des contrôles immédiatement disponibles par camion
- Fin de l'encodage manuel
- Alerte de sécurité directement communiquée aux différents services du site (logistique, garde, chargement)
- possibilité de contrôles plus ciblés, basés sur les non conformités précédentes
- statistiques et *reporting* facilités
- possibilité de croiser les données par société de transport, camion, chauffeur.

3. Solution proposée

3.1. Vue globale

L'architecture choisie en réponse aux exigences de Total repose sur trois éléments : le périphérique mobile, son mode de communication avec un serveur et le serveur lui-même qui gère le stockage des réponses aux check-lists dans une base de données.



Le choix du périphérique mobile s'est assez vite dirigé vers un appareil de type tablette pour plusieurs raisons. Celles-ci seront détaillées dans l'approche hardware.

L'application de gestion de check-lists sera installée sur cet équipement mobile et disposera d'un accès sans-fil vers l'extérieur grâce à une interface réseau. Le choix initial s'est orienté vers un mode connecté.

Nous avons également fait le choix de développer l'application sur deux plateformes : Windows et Android. La solution Windows était la plus

évidente étant donné que l'ensemble du système informatique de Total repose sur une architecture Microsoft. Nous avons cependant choisi de proposer une alternative sur Android. Parmi les raisons de ce choix, on retrouve : la grande variété de tablettes Android disponibles sur le marché, leur prix très abordable pour des performances élevées, un SDK gratuit et fourni par Google.

La connexion sans fil permettra l'accès à un serveur doté d'un gestionnaire de base de données servant de stockage pour les données des check-lists validées sur l'application. Le mode de communication se fera via un *Web Service* lui aussi hébergé sur le serveur. Les données seront échangées via des requêtes de type HTTP avec l'application mobile.
Enfin le lien avec la solution logistique eLisa s'effectuera via des requêtes SQL à travers le réseau de Total. Cette partie nécessite un accès au système informatique de l'entreprise. Elle ne rentrera donc pas dans les spécifications de l'application test fonctionnelle.

Nous obtenons une chaîne de communication qui inclut bien les trois éléments d'architecture cités en début de chapitre : une application mobile, un *Web Service* et une base de données.

3.2. Approche hardware

L'équipement mobile

Le marché technologique actuel propose une large gamme de produits mobiles : ordinateurs ultraportables, tablettes, Smartphones, etc. La tendance s'oriente clairement vers des appareils de taille et de poids réduits pour une taille d'écran maximale.
L'analyse des actuelles check-lists démontre que la plupart des contrôles permettant d'entrer les informations sont des cases à cocher. La quantité de données à entrer sous forme de caractères est donc restreinte. Pourquoi alors s'encombrer d'un clavier physique ? La solution de l'ordinateur même ultraportable n'est donc pas la plus adaptée.

Reste à faire un choix entre le Smartphone et la tablette. Ces deux types d'appareil sont finalement assez semblables. Dans la plupart des cas, leurs performances sont quasi identiques. On peut en fait considérer qu'une tablette est un grand Smartphone. Leur plus grande différence est donc la taille de l'écran, critère plutôt important dans le cas qui nous occupe.

Dans beaucoup de projets IT, lorsqu'il s'agit d'informatiser un processus, un des principaux enjeux est de rendre l'application développée la plus *user-friendly* possible afin d'adoucir un maximum la transition pour les employés qui devront apprendre à l'utiliser.
Ces différents aspects confirment bien notre choix d'utiliser périphérique de type tablette dans le cadre du projet.

Périphériques
L'équipement doit être muni d'une WebCam. D'autres périphériques peuvent être intéressants pour l'implémentation ultérieure de nouvelles fonctionnalités comme la lecture de QR tags ou de code-barres.

Le mode de communication
En terme de normes réseaux, nous avons gardé deux solutions possibles : la norme Wi-Fi ou la norme 3G.

Trois critères principaux interviennent dans la problématique de la transmission de données sans-fil :

- La couverture réseau
- La vitesse de transmission des données
- La fiabilité de la connexion

Total n'est actuellement équipé de bornes Wi-Fi que pour la partie bureau. L'entièreté du site n'est donc pas couverte par le signal, or sa superficie est importante. L'accès au réseau sans-fil n'est cependant nécessaire que dans les zones où est effectué le contrôle des camions. Une étude sur site pourrait facilement indiquer comment agencer les bornes pour une couverture optimale et un coût d'investissement réduit.

La solution 3G, si une antenne de l'opérateur se trouve à proximité du site, résout le problème de la couverture réseau. Mais contrairement au Wi-Fi, il ne s'agit plus d'un réseau privé. Total devrait passer par un opérateur et accepter que les données transmises transitent à l'extérieur du réseau local. Notons que dans ce cas, une solution VPN pourrait résoudre le problème de la sécurité des échanges.

Dans les deux cas, et vu la faible quantité de données échangées, la vitesse de connexion ne devrait poser aucun problème. La fiabilité de la connexion

quant à elle devrait être assurée si l'infrastructure réseau est bien dimensionnée.
Comme les tablettes sont équipées d'interfaces réseaux permettant l'utilisation des normes Wi-Fi et 3G, une bonne solution serait de permettre à l'application de switcher de l'un à l'autre pour assurer une réception continue de qualité.
Dans le cas où aucune des deux solutions n'est acceptable, l'application pourrait fonctionner en mode déconnecté et se synchroniser régulièrement avec le serveur via un terminal intermédiaire. Le contrôleur devrait alors se connecter physiquement avec ce terminal via un câble USB.

Le serveur
Étant donné l'infrastructure Microsoft du système informatique de Total, le choix d'un serveur du même type pour l'installation du *Web Service* et du serveur de base de données parait évident.

De plus, puisque ce serveur devra être en mesure de communiquer avec eLisa pour échanger des données de formulaire, nous avons tout intérêt à utiliser la même technologie pour garantir la compatibilité. L'architecture Microsoft sera donc privilégiée.

3.3. Approche software

L'application mobile
Une application mobile est un software écrit pour des équipements mobiles qui exécutent une tâche précise comme un jeu, un calendrier, un lecteur de musique, etc.

Deux grandes tendances existent dans le développement d'applications mobiles. Nous avons d'une part les applications web de type browser et d'autre part les applications natives. De plus en plus, les différences de fonctionnalités qu'offrent l'une et l'autre s'amenuisent. Voyons ci-après quelle sont les principales différences entre les deux :

- Une application native est conçue pour un système d'exploitation et un firmware spécifiques et nécessite une adaptation pour pouvoir s'exécuter sur d'autres appareils. Si elle a été développée pour iPad par exemple, elle devra pouvoir s'exécuter sur la plateforme propriétaire iOS spécifique à ce dernier et ne sera pas compatible

avec un appareil Nokia (Symbian[4]), Google (Android) ou Microsoft (Windows). Pour qu'elle le soit, elle devra être entièrement redéveloppée.

- Une application web de type browser est construite de manière à ce qu'une partie ou l'entièreté du software qui la constitue est téléchargée à partir du Web à chaque exécution. Cette caractéristique la rend accessible à n'importe quel appareil mobile muni d'un navigateur Web mobile. Typiquement, le langage utilisé pour ce genre d'applications est du HTML combiné à du JavaScript.

Un point intéressant à noter est que même si ces deux types d'applications restent très différents du point de vue de leur architecture, l'utilisation qui en est faite rend leur distinction parfois difficile. On retrouve ainsi sur le marché des applications natives qui utilisent constamment un accès Web et des applications de type browser qui proposent un mode offline permettant une utilisation sans connexion Internet.
Dans beaucoup de cas et avec les rapides avancées de l'HTML5, une application de type browser est souvent préférée puisqu'elle permet une utilisation multiplateforme. Les spécifications du projet qui nous occupe ne nous permettent cependant pas de nous orienter vers cette solution.

En effet, une des fonctionnalités que Total souhaite intégrer au processus de contrôle des camions est la possibilité de pouvoir prendre des photos de certaines parties du véhicule et de les joindre à la check-list. Or l'utilisation d'un périphérique de la tablette – l'appareil photo dans ce cas-ci – nécessite de passer par le système d'exploitation du périphérique ce qu'une application de type de browser n'est pas capable de faire.
Les applications Google et Microsoft seront donc toutes les deux développées en natif. Pour la première, nous utiliserons les APIs Android 3.2 fournies par Google et largement inspirées de Java. Pour la seconde, nous utiliserons le Framework .NET 4 qui utilise le langage Microsoft C#.

Le Web Service
L'application mobile doit être capable de transmettre vers le serveur les données entrées par l'utilisateur via l'interface. Puisque le serveur repose déjà sur une architecture Microsoft, nous avons choisi d'utiliser la

[4] Système d'exploitation pour téléphones portables conçu par Symbian LTD.

technologie *Windows Communication Foundation* (WCF) du Framework Microsoft .NET pour implémenter cette fonctionnalité. Il s'agit d'une architecture permettant de concevoir et déployer des applications orientées service.

WCF permet l'échange de messages asynchrones entre un service et différents points de terminaison. Les messages échangés peuvent être codés sous la forme d'un objet encapsulé dans une structure au format XML ou sous la forme d'un flux de données binaires. Un service de type WCF peut être hébergé sur un serveur de type IIS ou à l'intérieur d'une application.

Parmi les avantages mis en avant par WCF, on peut citer :

- Le respect des principes et normes en vigueur dans le domaine des SOA [5];
- Couplage faible entre le service et les applications client ;
- Une grande interopérabilité due au respect des standards établis ;
- Grande variété de protocoles et types d'encodages utilisables pour les messages ;
- Intégration au sein du Framework Microsoft .NET.

Parmi les architectures possibles pour la réalisation d'un *Web Service*, on retrouve deux classes importantes :

- Les *Web Service* de type REST[6] qui donnent accès à leurs fonctionnalités via une série de ressources (URI[7]) dont la syntaxe respecte une norme d'Internet mise en place par le W3C[8] et d'opérations indépendantes (*stateless*) respectant le protocole HTTP.
- Les *Web Service* de type WS-* dont les spécifications reposent sur les standards SOAP et WSDL[9].

[5] *Service Oriented Architecture*.
[6] REST pour *Representational state transfer*. Type d'architecture d'un Web Service.
[7] URI pour *Uniform Resource Identifier* en anglais, soit littéralement « identifiant uniforme de ressource ».
[8] W3C pour *World Wide Web Consortium*.
[9] WSDL pour *Web Services Description Language*. Il s'agit d'une interface publique d'accès à un Web Service.

Le SOAP est conçu pour une utilisation dans un contexte d'informatique distribuée. Il est complexe à mettre en place mais extensible et possède une gestion intégrée des erreurs.
Un *Web Service* REST est beaucoup plus simple à développer. Sa courbe d'apprentissage est courte et son utilisation très simple. Il est concis, plus proche du Web au niveau de la philosophie et du design et convient bien à une communication point à point.
Les contraintes de temps du projet et l'architecture du système proposé ont orienté le choix du *Web Service* vers une implémentation de type REST.

Serveur de base de données
Le serveur de base de données est hébergé sur la même machine que le *Web Service*. Son rôle est de gérer la base de données qui stocke les réponses apportées aux check-lists par les utilisateurs.
La gestion des check-lists
Lorsqu'un camion arrive sur le site de Total, le contrôleur doit charger sur sa tablette un modèle de check-list correspondant au produit transporté par ce camion. Ce modèle a été préalablement créé dans une application de design qui fera l'objet d'un point ultérieur. Il est ensuite stocké sous un format déterminé.

Un des enjeux du système est la standardisation de ces modèles de check-lists. L'application mobile doit être en mesure de gérer des formulaires ultra personnalisables et de les afficher en respectant au mieux le style des documents imprimés de l'ancien système.
Nous aurons donc une hiérarchie de pages, de chapitres, de sections et de questions qui permettront à Total de produire une check-list sur mesure.

Les modèles de check-lists doivent pouvoir être sauvegardés. La structure d'objets créée sur base de l'arborescence de classes doit pouvoir être exportée pour permettre un rechargement ultérieur.
La sérialisation est un outil du Framework .NET permettant de convertir une structure d'objets et leurs états en un format `XML` pouvant être stocké et rechargé (par une désérialisation) par la suite. Il correspond donc assez bien à ce qu'on cherche. Nous verrons d'ailleurs plus loin que la sérialisation est utilisée côté serveur par le *Web Service* REST lorsqu'il reçoit un message `XML`.
Lors du développement des premiers modèles de check-lists, il s'est avéré que le `XML` généré représentait un volume de données non négligeable.

L'envoi systématique, à chaque sauvegarde, du modèle complet de la check-list et de ses réponses n'est donc pas conseillé. Il est en fait même inutile puisque contrairement aux réponses, le modèle ne change pas à chaque fois. Une meilleure solution était donc de créer une structure d'objets parallèle réservée uniquement aux réponses entrées par l'utilisateur. Chaque réponse correspond à un contrôle de l'interface (`TextBox`, `CheckBox`, …) identifiable par un id unique. Ces réponses forment une liste liée à une sauvegarde unique et à un modèle de check-list lui aussi unique.

À la validation du formulaire, c'est uniquement cette liste de réponses qui est envoyée au serveur via le *Web Service*.

4. Développement

4.1. Étapes de mise en œuvre

Suite à l'analyse technique des différents éléments, l'ordre des étapes du projet spécifiées dans le cahier des charges a été quelque peu modifié. Nous avons commencé par la mise en place du *Web Service*, continué avec l'application mobile elle-même (version .NET) et terminé avec le serveur SQL. L'application Android n'a été réalisée qu'en fin de projet.

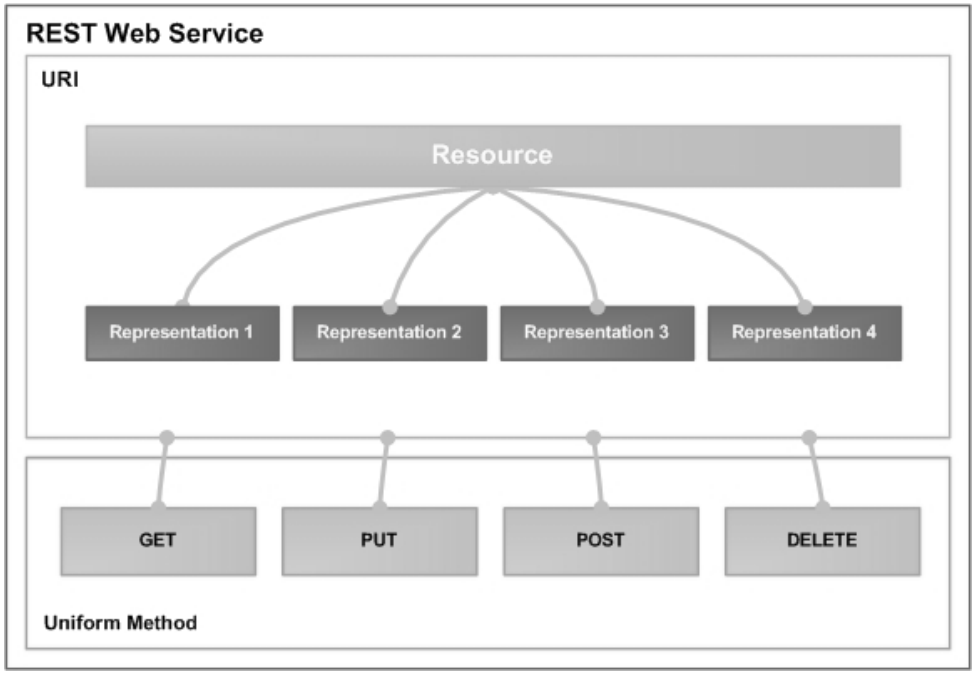
4.2. Outils de développement

Le développement du système nécessite une série d'outils de développement indispensables. Parmi eux, on retrouve : Microsoft Visual Studio 2010, Microsoft Expression Design, WPF Theme Editor, Eclipse et Microsoft SQL Server Management.

4.3. Web Service

La première chose à faire lorsqu'on met en place un *Web Service* REST est de déterminer quelles sont les ressources à exposer. Dans notre cas, puisque les données échangées sont des listes de réponses sauvegardées dans une base de données, voici une série de ressources qui nous seront utiles :

- Toutes les listes de réponses
- La liste de réponses d'une certaine date
- La liste de réponses possédant un certain ID
- Les listes de réponses liées à une plaque de véhicule



L'utilisation des URI est une contrainte du service REST. Un emploi intelligent du principe de ressources est indispensable. Retrouver une liste de réponses particulière peut donc être très simple une fois que les ressources et leurs adresses sont correctement définies.
Les principales méthodes de l'interface sont `GET`, `POST`, `PUT` et `DELETE`. `GET` est la méthode permettant de dire au service que l'utilisateur souhaite accéder à une ressource en lecture uniquement. `DELETE` correspond à une demande de suppression d'une ressource. `POST` indique que le client souhaite créer une nouvelle ressource. `PUT` est typiquement utilisé pour modifier une ressource existante. Le client peut cependant utiliser cette quatrième méthode pour ajouter une ressource s'il connaît l'URI à spécifier.

REST n'a pas de contraintes architecturales concernant la représentation physique des ressources. Il y a cependant un certain nombre de formats assez populaires et couramment utilisés : `XML`, `RSS/Atom`, `XHTML`, etc. Nous utiliserons dans ce cas-ci le format `XML`.

Dans le projet du *Web Service* dans Visual Studio, on retrouve une série de classes, une interface et un fichier de configuration.

4.4. Interface de l'application mobile

Chargement des check-lists
Sur base de la hiérarchie de classes correspondant à la structure d'une check-list, l'application doit être capable de charger dans son interface les contrôles correspondant aux questions, sections, pages et chapitres du modèle.
Les contrôles de l'interface peuvent être chargés soit au démarrage de l'application via du code `XAML` (ou `XML` en Android), soit de manière dynamique une fois l'application lancée.

Nous avons donc choisi de concevoir la mise en page générale de l'interface en `XAML/XML` et d'utiliser une fonction `LoadCheckList` qui s'occupe de charger dynamiquement les contrôles durant l'exécution du programme.
Notons qu'il s'agit d'une fonction récursive puisqu'elle doit être capable de gérer une arborescence d'objets potentiellement infinie.
Une fois les contrôles du modèle créés, un système de navigation s'occupe d'afficher les différentes pages du formulaire de contrôle.

Système de navigation
En WPF, pour naviguer entre les différentes pages d'une check-list, nous avons choisi d'utiliser la classe `NavigationService` du Framework .NET 4. Elle encapsule dans chaque page la possibilité de télécharger du contenu en se limitant au contexte d'une navigation de style navigateur.
Ce contenu peut être n'importe quel objet du *Framework* mais les objets de la classe Page sont préférablement utilisés. Voici un exemple d'implémentation d'une Page en `XAML` :

```
<Page
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presenta
tion">
    <Page.Content>
        <!-- Page Content -->
        Hello, Page!
    </Page.Content>
</Page>
```

Tout contenu est accessible via la navigation en appelant la méthode `Navigate` et spécifiant comme argument de cette fonction une instance de la Page vers laquelle on souhaite naviguer :

```
NavigationService.Navigate(Page page)
```

On peut également accéder au contenu désiré en spécifiant l'URI à l'une des surcharges de la méthode `Navigate` :

```
NavigationService.Navigate(Uri uri)
```

Lorsque le navigateur a atteint la page demandée, `NavigationService` retourne un objet avec son contenu.
Le système qu'utilise Android pour naviguer entre les différentes pages de l'application se base sur le concept d'`Activity`. Il ressemble assez bien au `NavigationService` de WPF. Chaque `Activity` correspond à un écran différent et la méthode `startActivity()` permet de passer de l'une à l'autre.

Style et thème
La création d'une interface utilisateur adaptée est primordiale pour permettre une utilisation facile et intuitive de l'application.
Pour le développement d'interfaces WPF, DevExpress propose une série de thèmes et de styles visuels conçus par les leaders du marché dans le domaine. Nous nous sommes donc orientés vers l'un de ces thèmes pour notre application.
Bien que ces thèmes soient parfaitement compatibles avec les contrôles que nous utilisons, il était nécessaire d'en adapter certains pour une utilisation tactile agréable. Pour ce faire, nous nous sommes servis de l'outil de personnalisation fourni lui aussi par DevExpress : WPF Theme Editor.

En Android par contre, une série de contrôles orientés mobiles sont fournis directement avec l'API de Google et aucune adaptation n'a été nécessaire.

Data Binding
Pour sauvegarder au fur et à mesure les réponses entrées par l'utilisateur dans les différents contrôles de l'interface, nous avons utilisé du *data binding*. Il s'agit d'une technique permettant de lier des objets entre eux pour les faire communiquer.

Dans notre application, les objets qui doivent communiquer entre eux sont d'une part, les contrôles de l'interface (`CheckBox`, `TextBox`, etc.) et d'autre part les éléments `CAnswer` contenus dans les `AnswersLists` envoyées au serveur via le *Web Service*.

C'est au moment de la création dynamique des contrôles que le binding est effectué. Voici un exemple d'implémentation du binding :

```
Binding b = new Binding()
{
Mode = BindingMode.TwoWay,
       UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged,
       Source = Answer,
       Path = new PropertyPath("AnswerValue")
};
BindingOperations.SetBinding(radio, RadioButton.IsCheckedProperty, b);
```

Plus d'informations sur le principe de Data Binding sont disponibles sur la plateforme MSDN[10] de Microsoft.

5. Généralisation de la solution

Les entreprises soucieuses de répondre aux besoins spécifiques de leurs clients tout en rentabilisant au mieux le travail de développement se doivent de se pencher sans cesse sur le défi de la standardisation et de la variabilité de leurs solutions. La question est simple : Comment rendre une application générique suffisamment flexible pour qu'elle puisse être proposée et adaptée à différents clients, dans différents environnements ?

Aujourd'hui par exemple, Pulsar équipe plusieurs autres usines de la solution eLisa. Toutes ces usines ont en effet un point commun : elles effectuent un contrôle à l'entrée, avant le chargement et avant la sortie des camions. L'application eLisa peut donc être ici proposée et facilement adaptée pour plusieurs clients.

Mais d'autres secteurs sont demandeurs de contrôles spécifiques pour la réception de marchandises. C'est le cas des entreprises travaillant dans le domaine de l'alimentaire en général et des produits frais en particulier.

---

[10] *Microsoft Developer Network Platforms*

Lors de leur réception dans un entrepôt, les marchandises devront être contrôlées sur le terrain. L'agent de contrôle peut dès lors, facilement, via une application mobile en mode connecté sur tablette, valider ou refuser ces marchandises. En cas de doute, l'application lui permettra de contacter immédiatement son supérieur via une communication VoIP[11] et lui montrer la marchandise via une caméra embarquée.

En fait, de nombreuses entreprises dans d'autres secteurs pourraient également investir dans une solution mobile de ce genre :

- Les sociétés de location ou de leasing qui doivent régulièrement vérifier l'état des véhicules qui entrent et sortent de parcs automobiles ;
- Les sociétés de logistique qui gèrent le transport de marchandises ;
- Les sociétés d'immobilier qui établissent quotidiennement des états des lieux de logements.

Le marché est donc large mais développer une application qui puisse être suffisamment standard pour ne pas devoir être redéveloppée pour chacune des applications citées ci-dessus n'est pas chose facile. Dans beaucoup de cas, la demande du client est trop spécifique.

L'idée est de penser les développements en amont pour arriver à concevoir une architecture standard qui puisse, par l'utilisation de certaines techniques, être adaptée selon les spécificités de la situation : modèles de checklists entièrement personnalisables, bibliothèques de contrôles prédéfinies, structure hiérarchique adaptable, etc.

Il existe par ailleurs aujourd'hui des initiatives en ce sens tel le projet Varibru[12], supporté par Innoviris/IRSIB[13] et la Région bruxelloise, qui vise à réunir des experts pour plancher sur des solutions industrielles pertinentes répondant au défi de la variabilité de l'informatique dans les entreprises.
Nul doute que la problématique est vaste et passionnante et justifierait, à elle seule, un travail de fin d'études entier.

---

[11] VoIP pour *Voice over IP*.
[12] Varibru pour *Variability in software-intensive product development.* Plus d'informations sur http://www.varibru.be/
[13] Innoviris est l'Institut Bruxellois pour la Recherche et l'Innovation.

6. Conclusion

Les objectifs de ce projet étaient ambitieux. Il s'agissait entre autres :

- de travailler sur un pré-projet en réponse à une demande concrète d'un client important (Total) ;
- de faire un tour d'horizon des technologies et outils de développement du monde des tablettes informatiques et de les mettre en pratique ;
- d'envisager l'intégration d'applications mobiles chez d'autres clients dans d'autres secteurs.

Au final, ce travail nous a permis d'avoir une vision plus claire sur les possibilités et les limites des différentes solutions et outils envisagés pour répondre aux besoins de Total. Il nous a également donné une vue globale sur les perspectives de standardisation de solutions à toutes les sociétés ayant besoin de systèmes de contrôle sur le terrain.

Une série d'outils et de technologies de pointe ont pu être étudiés, discutés, testés et implémentés au sein de maquettes de développement particulièrement intéressantes.

Une application test fonctionnelle a été présentée au terme du travail. Elle pourra servir de base si Total décide de poursuivre le projet.

7. Sources

[1] MASSÉ, MARK. *REST API.* Sebastopol : O'Reilly Media, Inc., 2012.
[2] FLANDERS, JON. *RESTful .NET.* Sebastopol : O'Reilly Media, Inc., 2009.
[3] NATHAN, ADAM. *WPF 4 Unleashed.* Indianapolis : Pearson Education, 2010.
[4] FARREL, JOYCE. *Microsoft Visual C# 2010 An introduction to object-oriented programming.* Boston : Course Technology, 2011.
[5] DELOITTE. *Marché des tablettes, phénomène de mode ou tendance de fond ?* 2010.
[6] GRABHAM, DAN. *Windows 8 tablets: what you need to know.* Techradar. [Online] April 20, 2012.

http://www.techradar.com/news/mobile-computing/tablets/windows-8-tablets-what-you-need-to-know-916134.

[7] COMMISSION EUROPEENNE. *Appareils et systèmes de protection pour les atmosphères explosibles* - ATEX. Commission européenne. [En ligne] 2 Février 2012. http://ec.europa.eu/enterprise/sectors/mechanical/atex/index_fr.htm.

[8] LUO, LIE. *Mobile applications: native v Web apps – what are the pros and cons?* mobiThinking. [Online] http://mobithinking.com/native-or-web-app.

[9] SMASHING MAGAZINE. *The Smashing Book – User Interface Design in Modern Web Applications*. Smashing Magazine. [En ligne] 28 September 2011. http://www.smashingmagazine.com/.