ZAC des Ancises, 5 Rue des Ancises, 03300 Creuzier-le-Neuf

Rapport d'élève ingénieur

Stage de 2e année

Filière : Génie Logiciel et Systèmes Informatiques

Conception et réalisation d'une Web App de gestion des stocks destinée à des responsables logistiques.

Présenté par : Rousseau Quentin

Responsable ISIMA : Yon Loïc
Responsable entreprise : Barrier Vincent

Date de la soutenance : 7 septembre 2022
Durée du stage : 4 avril 2022 - 29 juillet 2022

ISIMA
Campus Universitaire des Cézeaux · 1 rue de la Chebarde · TSA 600026
63178 Aubière cedex

Remerciements.

Je souhaite remercier l'ensemble des personnes qui ont contribuées au bon déroulement de ce stage. En premier lieu, mon maître de stage M. Vincent Barrier pour son investissement et sa disponibilité tout au long du stage. Je souhaite également remercier M. Christophe Chevalier pour ses nombreux conseils techniques qui m'ont permit d'améliorer grandement la qualité de mon travail. Je remercie également mon tuteur académique M. Loïc Yon pour son sérieux et son investissement dans l'encadrement du stage. Enfin je tiens à remercier l'ensemble du personnel de W3Plus pour leur accueil chaleureux et leur bienveillance qui ont fait de ces cinq mois un expérience plus qu'agréable.

Résumé

L'objectif de ce stage est de concevoir une **application web** destinée à des responsables logistiques. Ce nouveau projet, devrait, à terme remplacer une application existante, utilisée sur des **PDAs** vieillissants. Une application étant déjà existante, un long travail d'analyse et de retro-engineering a du être mené pour mieux cerner ce qui a du être réalisé par la suite.

L'architecture visée est une **architecture n-tier**, avec une couche de présentation des données en réalisée en **Angular**, un framework web maintenu par *Google*, une couche d'accès aux données réalisée en **C# .NET Core**, et la couche de données en **SQL Server**.

Mots-clés : Application Web, PDA, architecture n-tier, Angular, C# .NET Core, SQL Server.

Abstract

The objective of this intership is to develop a **web application** for logistics managers. Eventually this new project should replace an existing application used on agging **PDAs**. An existing application being already in place, a long work of analysis and retro-engineering had to be done afterwards.

The targeted architecture is an **n-tier architecture** with a frontend developed in **Angular**, a web framework maintened by Google, a backend realized in **C# .Net Core** and databse using **SQL Server**.

Keywords : Application Web, PDA, architecture n-tier, Angular, C# .NET Core, SQL Server.

Table des matières

Table des figures

Acronymes

CSS Cascade Style Sheet

DOM Document Object Model

DTO Data Transfert Object

EDI Environnement de Développement Intégré

ERP Entreprise Ressource Planning

ESN Entreprise de Services du Numérique

GED Gestion Électronique de Document

HTML HyperText Markup Language

HTTP HyperText Transfert Protocol

JSON JavaScript Object Notation

PDA Personal Digital Assistant

PWA Progressive Web App

REST Representational State Transfert

SGBD Système de Gestion de Base de Données

SPA Single Page Application

SQL Structured Query Language

Glossaire

API Une API (application programming interface ou « interface de programmation d'application ») est une interface logicielle qui permet de « connecter » un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités [1]

Emplacement Entité informatique qui permet de contenir des stocks.

Framework Cadriciel en français, ensemble de composant logiciels structurels pouvant servir de base à un logiciel.

Gencode code barres utilisé par les magasins pour la vente des articles.

MVC Modèle Vue Contrôleur : Patron de conception. Il met en avant la séparation entre la partie métier d'un logiciel et son affichage [2].

Paquet Un paquet peut contenir d'autres paquets ou des références.

Picking En logistique, c'est le fait d'aller chercher de la marchandise (carton, housse, article) afin de les regrouper en vue d'une commande.

Profiler Logiciel permettant l'analyse d'un programme.

Regular Expression Objet utilisé en JavaScript pour étudier les correspondances entre un texte et un motif [3].

Référence Détail d'un article à la taille/couleur.

SaaS Le software as a service (SaaS) ou logiciel en tant que service, est un modèle d'exploitation commerciale des logiciels dans lequel ceux-ci sont installés sur des serveurs distants plutôt que sur la machine de l'utilisateur. Les clients ne paient pas de licence d'utilisation pour une version, mais utilisent librement le service en ligne ou, plus généralement, payent un abonnement [4].

1 Introduction

Dans le carde de ma deuxième année à l'ISIMA (l'Institut Supérieur d'Informatique, de Modélisation et de leurs Applications), j'ai effectué un stage de quatre mois au sein de la société W3+ située dans l'Allier (03). W3+ est une Entreprise de Services du Numérique (ESN) qui concentre la plupart de ses activités autour de son client principal, *Textilot*.

La société Textilot, déclinée sous la marque de prêt-à-porter *Plus*, possède une forte activité liée à la logistique et à l'acheminement de ses articles dans les différents points de ventes partout en France et en Belgique.

Basée à Varennes-Vauzelles dans la Nièvre (58), sur le même site que le siège social, le centre logistique se sert d'une application de gestion de stock sur des Personal Digital Assistant (PDA) durcis qui a également été produite par W3+. Cette application étant basée sur des technologies vieillissantes, elle devient de plus en plus compliqué à maintenir, d'où la nécessité de la remplacer. La solution choisie à été la création d'une nouvelle application qui prendra la forme d'une Progressive Web App (PWA). Indépendantes du matériel, les PWAs peuvent en théorie fonctionner sur n'importe quelle machine dotée d'un navigateur internet.

L'objectif de ce stage était donc d'imaginer et de développer cette nouvelle application de gestion de stock, en utilisant des technologies plus récentes.

Pour expliquer l'ensemble du travail effectué au cours de ces quatre mois, nous débuterons par la présentation de l'entreprise, du contexte ainsi que du sujet. Nous regarderons ensuite le travail et l'analyse qui ont du être menés en amont du développement. Nous finirons donc par la phase de conception de l'application et d'organisation du travail en détaillant les différentes étapes qui ont été effectuées.

2 Présentation du contexte

2.1 La société W3+

2.1.1 Historique

Fondée en 1988, la société *CS3i* a été l'une des première à proposer de l'infogérence à ses clients dans la région Auvergne. Elle s'est principalement développée grâce à l'essor du Minitel dans les années 90, puis grâce à internet sur la fin de la même décennie, en orientant ses activités vers de l'hébergement. En 2002, elle lance le développement du logiciel Emed® à la demande d'un client. Il est destiné au monde hospitalier et permet notamment la gestion des dossiers médicaux des patients. En 2008, la société décide de se scinder en deux, pour d'un coté regrouper toutes les activités liées au médical, et de l'autre les autres projets. *CS3i* garde donc la partie médicale, *W3+* est créé pour gérer l'hébergement et l'infogérence. Les deux entreprises sont alors sous la même holding, *C4j*. En 2014, *CS3i* est vendu à un groupe allemand, puis en 2018 *C4j* décide de se tourner vers le mobile avec la création de *CalaoSoft*, dont les locaux sont dans le même bâtiment que ceux de *W3+*. En 2020, *Textilot*, le client principal de *W3+*, devient l'actionnaire majoritaire, et M. Sébastien DUMANGE en devient de PDG.

2.1.2 Activités et organigramme

En 2021, 85% du chiffre d'affaire global de *W3+* provient de l'activité lié à *Textilot*. La moitié de l'activité restante représente la revente de matériel et de services à d'autres clients. La revente de logiciels prend la forme de SaaS* (Software As A Service). *W3+* tend à augmenter cette activité, en proposant par exemple la solution *Axelor*, un ERP open source qui possède des modules par défaut qui sont ensuite adaptés en fonction des besoins de chaque client. Le reste des activités concernent l'hébergement et la formation.

W3+ compte une quinzaine d'employés qui travaillent ensemble dans un open-space, ce qui permet une bonne communication aussi bien entre les différentes équipes, qu'a l'intérieur même de ces équipes.

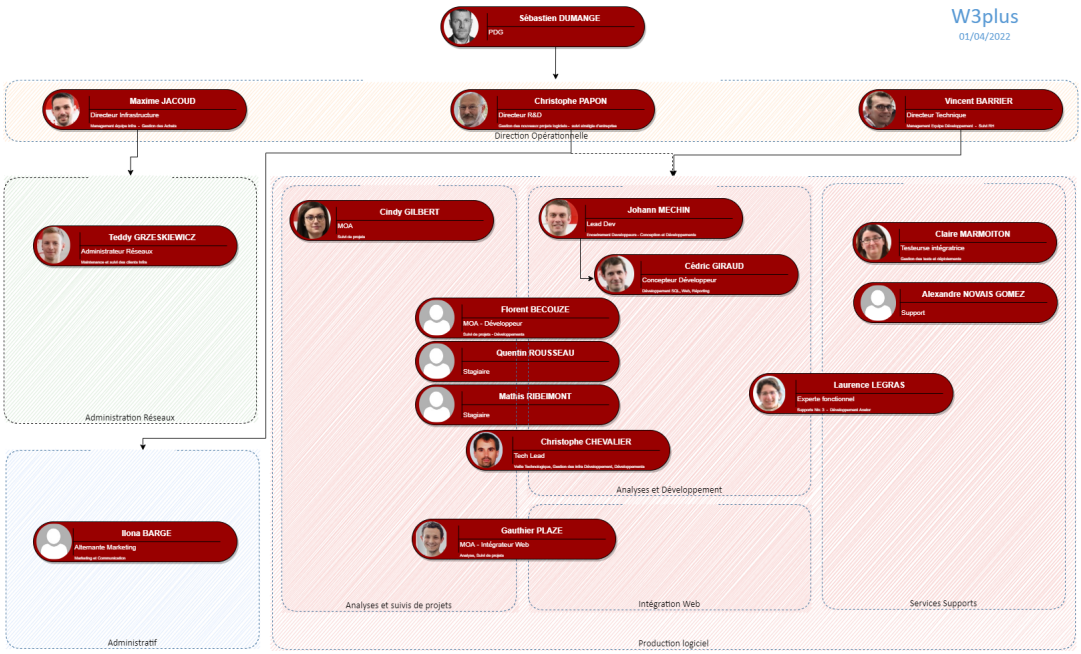


FIGURE 1 – Organigramme de W3+

Sur l'organigramme ci-dessus, on peut différencier différents types de métiers qui sont les suivants :

- Développeur
- Architecte logiciel
- UI/UX designer
- Consultant fonctionnel
- Testeur-intégrateur
- Administrateur système

2.2 La société Textilot

Le projet mené pendant mon stage est à destination de la société *Textilot*, il est donc important de la présenter. Via sa marque *Plus*, elle est l'une des leaders du marché du textile avec 3400 clients dans toute la France, la Belgique et le

Luxembourg. L'entreprise a su mettre en place un système innovant, elle propose aux grandes et moyennes surfaces la gestion d'un rayon textile. *Textilot* se charge de l'agencement du rayon ainsi que de son approvisionnement. Toutes les semaines, les articles sont mis à jour, de nouveaux sont mis en vente, et les invendus sont repris. Il existe également 9 magasins *Plus* qui servent de pilotes pour tester les collections et ainsi pouvoir adapter au mieux l'approvisionnement des clients.

Pour livrer ses 850.000 articles hebdomadaires, *Textilot* dispose d'une flotte de plus de 300 camions qui acheminent la marchandise du centre logistique, depuis Varennes-Vauzelles (périphérie de Nevers) jusqu'aux nombreux points de ventes. La charge logistique est donc très importante et nécessite un suivi informatique conséquent. Plus de 60.000m^2 d'espace de stockage sont nécessaires, dont une partie est complètement automatisée.

FIGURE 2 – Camions Plus

2.3 Objectifs du stage et du projet

2.3.1 Présentation de l'existant

L'objectif de mon stage a donc été de réaliser un application web de gestion de stock visant à remplacer une application existante et vieillissante. Dans les espaces de stockage non automatisés de *Textilot*, les préparateurs de commandes se servent d'une application (*MobileStock*) qui leur permet d'effectuer tous types de traitement sur les stocks de l'entreprise, comme par exemple des inventaires ou des opérations de picking*. Cette application est exécutée sur des CN50, des PDAs

durcis qui sont dotés d'un système d'exploitation Windows Mobile et d'un scanner intégré. Toutes les unités de stock à *Textilot* sont identifiées par un code-barre, d'où la nécessité d'un scanner pour lire ces codes.

FIGURE 3 – CN50 - InterMec

Pour mieux comprendre l'utilisation de Mobile Stock dans les entrepôts de *Textilot*, j'ai eu l'occasion de visiter le centre logistique, et de suivre des préparateurs de commandes dans leurs différentes tâches. J'ai donc pu voir comment les CN50 sont utilisés pendant des opérations de picking et de gestion de stock (déplacement de marchandise depuis un emplacement* de stockage vers un autre). L'application possède beaucoup d'autres fonctionnalités que je n'ai pas forcement eu l'occasion de voir pendant la visite, les chefs d'équipes ont néanmoins listé toutes les opérations qui sont effectuées depuis Mobile Stock ce qui m'a permis de mieux comprendre l'utilisation globale de l'application ainsi que le travail qui sera à réaliser :

- Commandes
- Réassorts
- Pickings Stock / Outlet
- Pré-Pickings

— Rangement de lignes
— Retours lingeries
— Inventaires
— Lectures nouveautés (Distribution)
— Implantations RAQ (reste à quai) / Non livrés
— Déplacements internes
— Retours spéciaux
— Lectures BA (Bon d'Arrivage) nouveautés
— Transferts

2.3.2 Intérêt du projet

Comme mentionné plus tôt, les CN50 commencent à être des appareils vieillissants, il ne sont plus produits et ne sont donc pas remplaçables en cas de panne. Les CN50 sont dotés d'un système d'exploitation Windows Mobile dont la dernière version stable remonte à février 2010. Après cette date, le système d'exploitation a été remplacé par Windows Phone, puis par Windows 10 Mobile. L'application Mobile Stock a été programmé en Visual Bassic .Net à l'aide de Environnement de Développement Intégré (EDI) Visual Studio 2008. Ainsi, les évolutions et l'ajout de nouvelles fonctionnalités deviennent de plus en plus compliqués. C'est pourquoi la refonte de l'application Mobile Stock est intéressante, avec l'utilisation de technologies plus récentes.

Le but est de développer une nouvelle application qui tournera sur des *CipherLab RS35*. Ces smartphones durcis, dotés d'un système Android, embarquent comme les CN50 un scanner, qui permet la lecture de code-barre. Mais ils possèdent de nombreux avantages, ils sont plus légers et possèdent un écran plus grand que le CN50, il paraît donc plus agréable à l'utilisation. L'écran tactile est également de meilleure qualité et on peut donc imaginer une utilisation plus précise de celui-ci, avec plus de possibilités que sur le CN50. À l'inverse de Mobile Stock, la nouvelle application ne sera pas développée uniquement pour un terminal précis, la PWA pourra techniquement être utilisable sur n'importe quel appareil possédant un navigateur internet. L'application sera donc totalement indépendante du matériel contrairement à l'application actuelle.

FIGURE 4 – CipherLab RS35

3 Travail préparatoire

3.1 Organisation générale

Ce stage de 4 mois s'est organisé en deux grandes phases principales. Tout d'abord un longue partie d'analyse et de retro-engineering puis une deuxième de conception de la nouvelle l'application. La première phase a été essentielle au bon déroulement de ce projet, se basant sur une application déjà existante, il a été important de bien comprendre ce qui existait déjà et de bien définir les éléments qui ont du être repensés ainsi que ceux qui ont pu être réutilisés dans la nouvelle application.

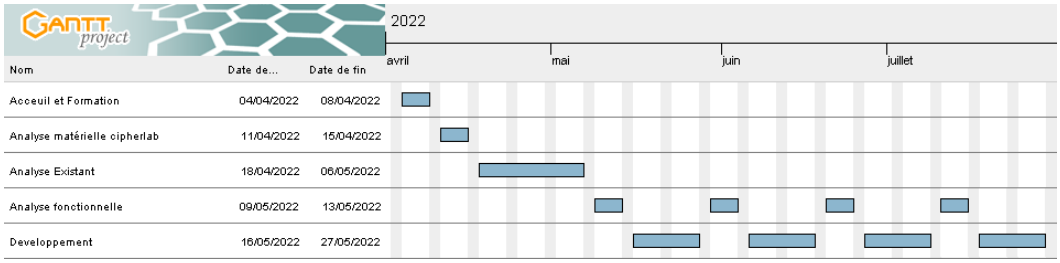


FIGURE 5 – Diagramme de Gantt prévisionnel

Sur le digramme de Gannt prévisionnel on remarque ces deux phases d'analyse et de conception. On voit aussi que la partie développement est divisée en plusieurs cycle, selon des *sprints* qui seront detaillés juste après. Au début du stage, les parties de l'application qui allaient être implémentées en premier n'étaient pas encore définies.

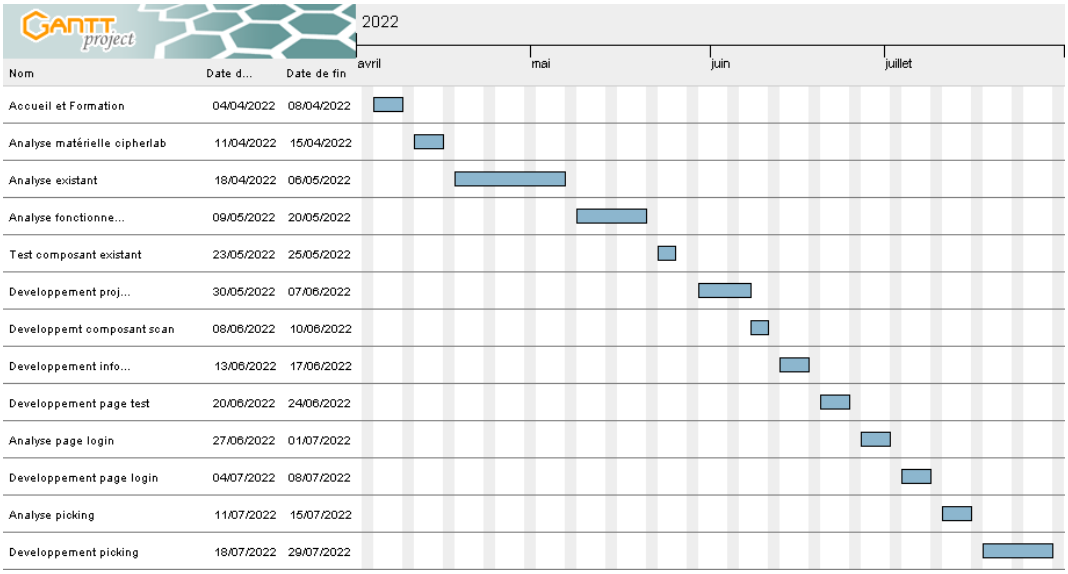


FIGURE 6 – Diagramme de Gantt réel

On peut noter quelques différences entre ce qui était prévu et ce qui a été réalisé, la partie d'analyse au début du stage a duré un peu plus longtemps que prévu. J'ai également pris part au développement d'un projet d'un autre stagiaire au début du mois de juin. Ensuite, les cycles d'analyse et de développement, sous forme de sprint, sur des parties spécifiques de l'application se sont déroulés comme prévu, à l'exception peut être du temps passé sur chacun de ces cycles.

3.2 Les sprints

Le développement des différentes parties prennent la forme de sprints. Deux solutions sont utilisés à W3Plus pour mettre en place ces sprints, Azure DevOps et Team Fundation Server. Pour ce projet je me suis principalement servi d'Azure.

En méthodologie agile, un sprint est une itération. C'est une période de deux ou trois semaines pendant laquelle une version terminée et utilisable d'une partie d'un projet doit être réalisée. Un sprint est découpé en backlogs, qui eux mêmes regroupent des tâches.

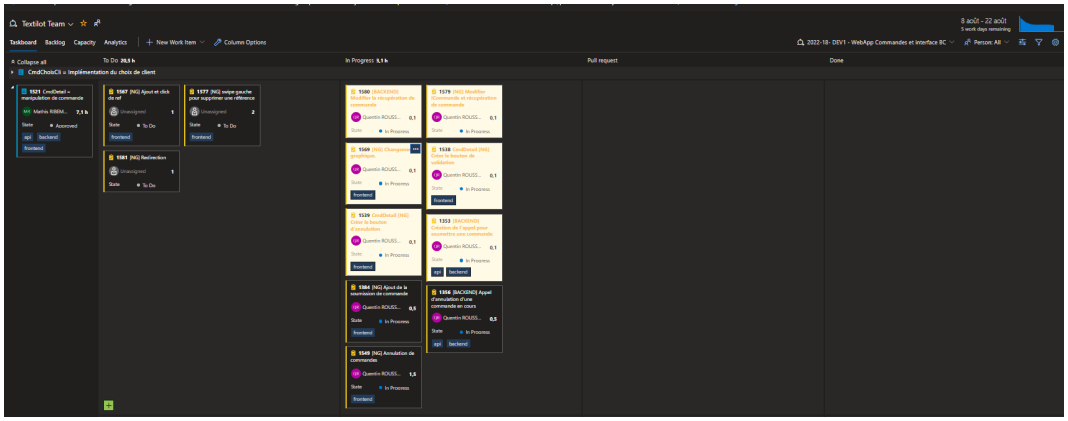


FIGURE 7 – Exemple d'un backlog

Sur cette page d'Azure, on peut voir les différents backlogs d'un sprint, les tâches à réaliser sont dans un tableau avec différentes colonnes en fonction de leur avancement, ce qui permet aussi de mettre en place une méthodologie *Kanban*

Azure Devops permet d'héberger des repositories Git. Chaque backlogs est développé sur un branche Git locale, une fois toutes les tâches terminées, on effectue une *pull request*. Après plusieurs échanges avec un relecteur, la *pull request* peut être complété, la branche de développement est mergée dans la branche master, et une pipeline s'occupe d'apporter l'ensemble des modifications sur une machine de « dev ». Ensuite les testeurs peuvent récupérer les nouveautés sur leur machine de « test ». Un fois les tests passés, les modifications peuvent être mises en production.

3.3 Analyse de l'existant

3.3.1 Les travaux et les lectures

Pour bien comprendre comment fonctionne l'application, il est important d'expliquer ce qui est appelé les *travaux* et les *lectures*. Les travaux représentent tout ce qui peut être en lien avec des pièces/articles présents dans le stock. Quand on doit effectuer une opération sur le stock, on a un *travail* à effectuer dessus. Il existe différents type de travaux, et ces travaux possèdent des lectures. Une lecture permet de détailler un travail, en précisant sur quelles pièces les travaux doivent être menés, où ces pièces se trouvent et en quelle quantité. Les lectures sont elles aussi subdivisées en *rames*, dans lesquelles on trouve le détail d'une lecture. En quelque sorte, les rames sont l'historique d'une lecture.

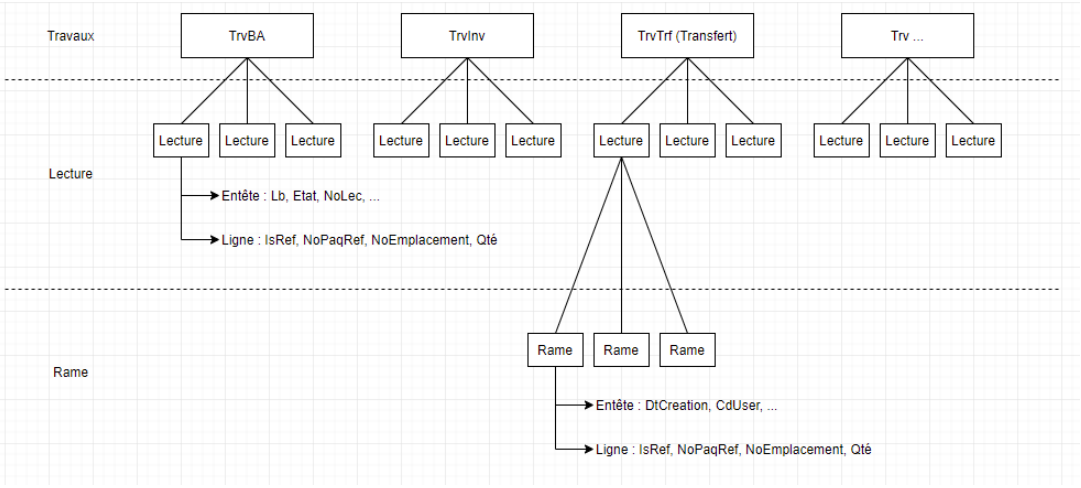


FIGURE 8 – Schéma récapitulatif des travaux/lectures

L'application Mobile Stock permet donc d'effectuer différents type de travaux

3.3.2 Les procédures stockées

Le corps du métier de l'application est effectué via des procédures stockées, en Transact-SQL. L'application effectue des requêtes sur une base de données, lance des procédures qui mettent à jour le stock. Un grande partie du processus est implémenté directement dans la base de données, l'analyse de ces procédures à donc été primordiale, afin de déterminer ce qui est fait au niveau de la base (et qui pourra être réutilisé) et ce qui est directement fait sur l'application.

3.3.3 Réglages des PDAs

Chaque lectures nécessite des réglages différents au niveau des PDAs, tous ces réglages sont stockés dans une base de données dont le schéma est donné en annexe A. On remarque qu'un grand nombre de paramètres est stocké dans cette base, l'analyse de leur utilisation à l'intérieur des procédures a permis de déterminer plus précisément quelle partie est implémentée au niveau de la base de données.

À l'aide d'un profiler*, et en simulant une lecture à partir de Mobile Stock on peut voir toutes les procédures stockées qui sont appelées. On remarque que deux procédures sont au coeur des lectures :

— ac_Lec_RameAdd : Quand un emplacement est scanné.

— ac_Lec_RameAddUpd : Quand un paquet* ou une référence* est scanné.

L'analyse de ces deux procédures a permis de déterminer quels paramètres présents dans la table des réglages PDAs sont utilisés dans ces procédures et par conséquent, lesquels sont gérés au niveau de l'application. Pour mieux comprendre le fonctionnement, plusieurs diagrammes de flux ont été réalisés.

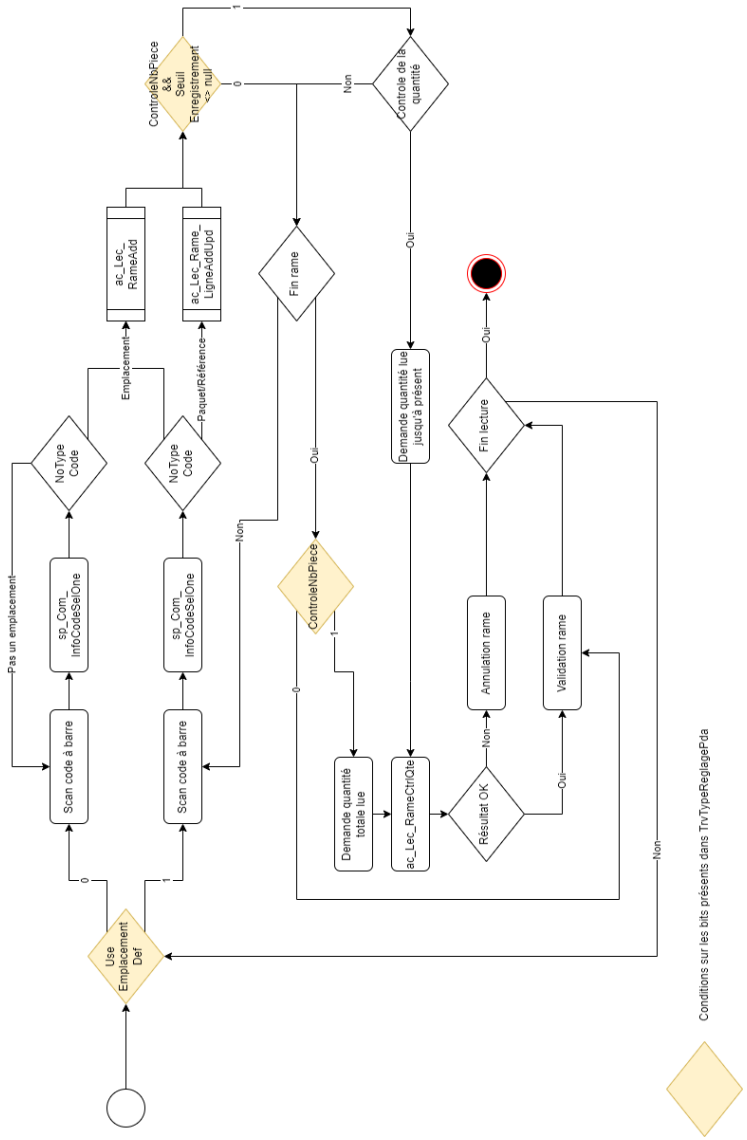


FIGURE 9 – Diagramme de flux d'une lecture

Finalement, on voit sur le diagramme que peu de paramètres sont gérés au niveau de l'application, et que la plupart sont gérés dans les procédures stockées. D'autres diagrammes, notamment ceux qui décrivent les deux procédures citées précédemment, ont été réalisés (Disponible en annexe B.1), ainsi que des diagrammes concernant d'autres fonctionnalités de l'application (Disponible en annexe B.2).

3.4 Architecture visée

La nouvelle application de gestion de stock s'intègrera dans un projet déjà existant, *TexPlusErp(TPE)*. TPE est l'Entreprise Ressource Planning (ERP) actuellement utilisé par *Textilot*, principalement pour la gestion des frais des employés. C'est un projet plutôt récent dans lequel s'ajoutent de nouvelles fonctionnalités, dont la nouvelle application de gestion des stocks.

L'architecture de ce projet se rapproche d'une architecture *3-tiers*. Ce type de projet se divise en trois « couches » bien distinctes :

— Couche de présentation : Correspond a l'affichage des données et d'interaction avec l'utilisateur. (frontend)

— Couche d'application : Correspond à la mise en œuvre des règles de gestion et de logique applicative. (backend)

— Couche de données : Correspond à la persistance des données. (base de données)

La seule différence se situe au niveau de la couche applicative, dans *TPE* la logique métier se situe dans la base de données via des procédures stockées. Il y a donc la partie frontend qui s'exécute sur la machine du client et qui contient les interfaces, et la partie backend qui tourne sur un serveur.

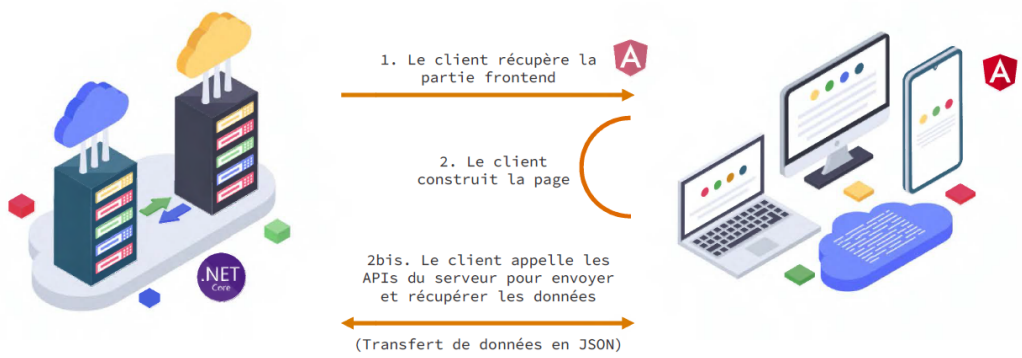


FIGURE 10 – Schéma architecture TPE

4 Présentation des outils et des technologies

4.1 Angular

Angular est un framework coté client (web) open-source. Il est développé et maintenu par *Google*. Angular est une réédition complète d'AngularJS, un framework qui a été développé par la même équipe, on le qualifie parfois Angular d'« Angular2+ » ou « Angular V2 et plus ». Il est basé sur une architecture MVC*, il sépare donc les parties gestion de données, affichage et métier, ce qui permet une bonne maintenabilité et ce qui facilite le travail collaboratif.

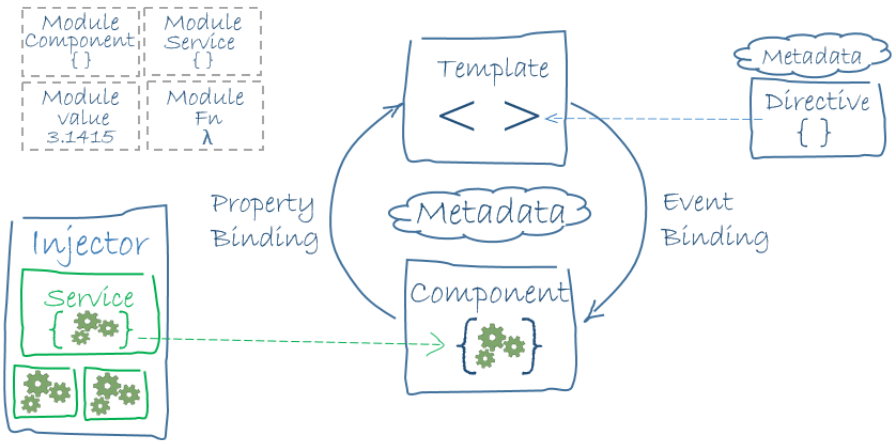


FIGURE 11 – Architecture d'une application Angular

Les composants sont les principaux éléments d'une application Angular, chaque application consiste en un « assemblage » de composants. Chacun de ses composants est constitué de :

— Un template HyperText Markup Language (HTML) qui déclare quoi afficher sur une page,
— Une classe TypeScript qui définit le comportement du composant,
— Un selecteur Cascade Style Sheet (CSS) qui définit comment le composant est utilisé dans le template,
— Le style CSS à appliquer au composant.

Angular permet la création de Single Page Application (SPA). Ce type d'application charge tout le contenu de l'application web dans une unique page HTML,

puis actualise seulement l'affichage pendant la navigation. Une fois l'application chargée, le changement entre les pages est donc très rapide, ce qui améliore grandement l'expérience de l'utilisateur. En Angular, ce qui est affiché à l'écran est géré par ce que l'on appelle des *routes*. Ces routes permettent de naviguer d'une partie à une autre dans l'application, chaque composant possède une URL et les routes permettent de spécifier quelle URL de quel composant on souhaite afficher.

Pour mieux structurer le projet, certains composants sont appelés des « composants page », d'un point de vue purement technique, les pages sont identiques à des composant classiques. Ce n'est donc juste qu'une notation. Mais elle permet de mieux séparer les différentes parties du projet. Les pages sont donc un ensemble de composants, avec une logique propre, et ce sont ces pages qui seront affichées via les *routes*.

PrimeNg, une bibliothèque de composant graphique pour Angular, a également été utilisée tout au long du projet.

4.2 TypeScript

TypeScript est un langage de programmation orienté-objet libre et open-source développé par *Microsoft*. Il s'agit d'un sur-ensemble syntaxique de JavaScript, il ajoute donc certaines fonctionnalités à ce dernier, principalement le typage des variables et des fonctions. Le Typescript est transcomplilé (compilé de source à source) en JavaScript, et peut donc être interprété par n'importe quel navigateur internet.

4.3 C# et .NET Core

Le *C#* est un langage orienté objet développé par *Microsoft*. Il est dérivé du *C* et du *C++* mais sa syntaxe se rapproche de celle du Java. Dans la cadre du projet ce langage est utilisé avec le framework .NET Core, qui regroupe un ensemble de bibliothèque de haut niveau. Ces technologies ont principalement été utilisées pour la mise en place des APIs* nécessaires à l'application.

4.4 SQL Server et Transact-SQL

Microsoft SQL Server est Système de Gestion de Base de Données (SGBD) en langage Structured Query Language (SQL). *Transact-SQL* est une extension du langage SQL permettant d'étendre ses fonctionnalités via des procédures stockées. Ces procédures permettent d'ajouter tout l'aspect de la programmation procédurale aux SGBDs avec les instructions basique (IF, ELSE, WHILE ...).

4.5 Outils

Au cours de ce stage j'ai pu utiliser un certain nombre de logiciels pour pouvoir utiliser les différentes technologies citées ci-dessus :

- Axure RP : logiciel de maquettage et de prototypage,
- Draw.IO : logiciel d'édition de graphe utilisé dans la partie d'analyse pour réaliser des diagrammes de flux.
- Visual Studio : EDI* utilisé pour le C#/.NET,
- Visual Studio Code : EDI* utilisé pour Angular,
- SQL Server Management Studio : EDI* dédié à SQL Server, utilisé pour tester les modifications de procédures stockées, pour faire des requêtes dans les différentes bases.

5 Conception de l'application

5.1 Composant de scan

5.1.1 Précision sur les inputs et les outputs d'Angular

Angular permet de partager des données entre les différents composants d'une application via ce que l'on appelle des inputs et des outputs. Dans le cas d'un composant placé dans une page, la page sera le composant « parent » et le composant en lui même sera « l'enfant ». Pour définir un input, il suffit de le préciser grâce à un décorateur sur une propriété de la classe typescript.

```
1 export class ChildComponent {
2   @Input() item = ''; // decorate the property with @Input()
3 }
```

Extrait de code 1 – Exemple d'input 1

Ensuite pour donner une valeur à cet input, il suffit de la *binder* dans le template du composant parent.

```
1 <app-child-component [item]="currentItem"></app-child-component>
```

Extrait de code 2 – Exemple d'input 2

Le composant enfant récupère le valeur de la propriété *currentItem* du composant parent.

Les inputs permettent donc de faire transiter des données d'un composant parent à un composant enfant. De façon similaire, les outputs permettent de le faire dans le sens inverse grâce à un *EventEmitter*. Le composant enfant peut émettre la valeur d'une de ses propriété et le composant parent peut la récupérer via la gestion d'un évènement.

5.1.2 Analyse du besoin et maquette

Le fait de scanner un code barres sera l'action principale que les utilisateurs de l'application pourront faire. La création d'un composant spécifique de scan est donc une idée intéressante. Un tel composant pourrait être utilisé à différents endroits dans l'application de stock voire même dans d'autres projets. Ce composant doit donc être modulable en fonction des besoins.

Quand un code est scanné, en fonction de ce que l'on veut faire, différentes APIs devront être appelées. Le composant ne peut donc pas faire directement les appels aux APIs. Le choix qui a été fait est de faire remonter le code lu par le composant à la page dans laquelle il est placé, pour que celle-ci puisse les effectuer.

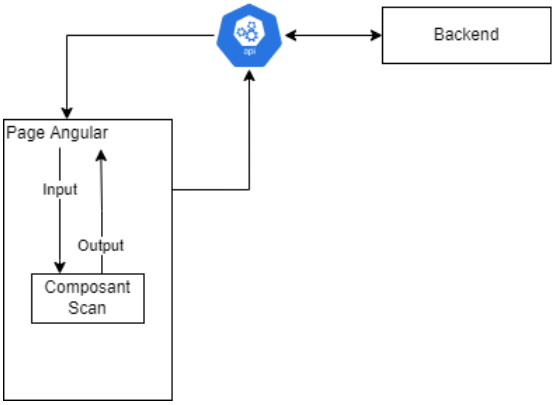


FIGURE 12 – Principe de fonctionnement du composant scan

Pour paramétrer le composant, on lui passera donc une liste d'input. Quand un code sera lu, il émettra la valeur du code lu via un output. Finalement le composant de scan ne comporte que peu « *d'intelligence* », en effet c'est dans les pages où il sera placé que le gros du travail sera à faire, ce sont ces pages qui géreront son comportement, son état ... La liste de tous les inputs qui permettent le paramétrage est disponible en annexe C.

FIGURE 13 – Maquette composant scan en attente et en erreur

5.1.3 Conception du composant

La conception du composant est assez simple, une liste d'input pour le paramétrage et la gestion de son état, deux outputs pour le code lu et un booléen indiquant si le code doit être ajouté ou retiré de la lecture en cours. En Angular il est possible de détecter les changements d'inputs dans un composant, ce qui est

très pratique pour le composant de scan. Par exemple quand les pages demandent au composant de passer en erreur, on peut déclencher toutes les méthodes pour afficher les messages ou pour déclencher une vibration du PDA.

L'amélioration majeure par rapport à l'existant est l'utilisation d'expression régulière. Avant de faire remonter le code à la page pour qu'elle fasse les appels d'APIs, on vérifie si le code répond à une RegExp d'erreur et une RegExp d'acceptation. Si le code ne passe pas ces tests, le composant se met lui même en état d'erreur, ne remonte pas le code lu et évite ainsi des appels d'APIs. Par exemple, si dans une situation l'utilisateur doit impérativement scanner un code d'emplacement, on paramètre le composant pour qu'il n'accepte que ce type de code. Tous les codes d'emplacement sont de la même forme, ils commencent par la séquence 100000000, suivi du numéro d'emplacement (4 chiffres) et d'un checksum. Dans ce cas ci, si le code ne répond pas à la RegExp d'acceptation *'100000000([0-9]{5}$)'*, le composant se met en erreur directement sans avoir à questionner la base de données.

La partie la plus complexe dans ce composant a été d'avoir le bon fonctionnement sur le cipherlab. Sur le smartphone, quand un code est scanné, il simule un clavier et écrit les caractères du code barres lu. Le composant est donc un champ d'input où il est aussi possible de taper un code manuellement. Le souci se pose au niveau du focus dans ce champ. Le cipherlab exécute son scan et simule le clavier, mais ne se préoccupe pas d'où les caractères tapés sont entrés. Pour les récupérer, il faut donc être dans un champ d'input quand la simulation du clavier est faite. Il était bien évidemment impossible de demander à l'utilisateur de se mettre manuellement dans le champ d'input à chaque fois qu'il veut scanner un code.

```
@ViewChild('pInput') private input: any;
 @HostListener('document:keydown', ['$event'])
 handleKeyboardEvent(event: any): void {
   if (event.key === 'Unidentified') {
     this.nCodebarre = undefined;
     this.isEtatScan = EEtatScan.Attente;
     if (!(event.target instanceof HTMLInputElement)) {
       this.setFocus();
     }
   }
 }
 setFocus(): void {
   setTimeout(() => {
     this.input.nativeElement.focus();
   });
 }
```

Extrait de code 3 – Auto-Focus

C'est ce code qui permet de faire l'auto-focus dans le champ d'input. Le décorateur *@ViewChild* permet de récupérer dans le Document Object Model (DOM) le champ ou le focus devra être fait. *@HostListener* permet d'écouter des évènements sur toute la page, dont les appuis clavier (la touche qui déclenche le scan sur le cipherlab est reconnu comme un appui clavier par les navigateurs). On peut donc simplement gérer l'évènement en faisant le focus dès que cette touche est pressé par l'utilisateur. On vérifie également que nous ne sommes pas déjà dans un champ d'input avant de faire le focus.

5.1.4 Page de test du composant scan

Contenant beaucoup de paramètres, le composant de scan est assez compliqué d'utilisation, c'est pourquoi il m'a été demandé de réaliser une page de test de ce composant. Dans la page, on retrouve l'ensemble des paramètres qui peuvent directement y être modifiés pour voir le comportement du composant. Par exemple l'activation ou non du mode de décompatge :

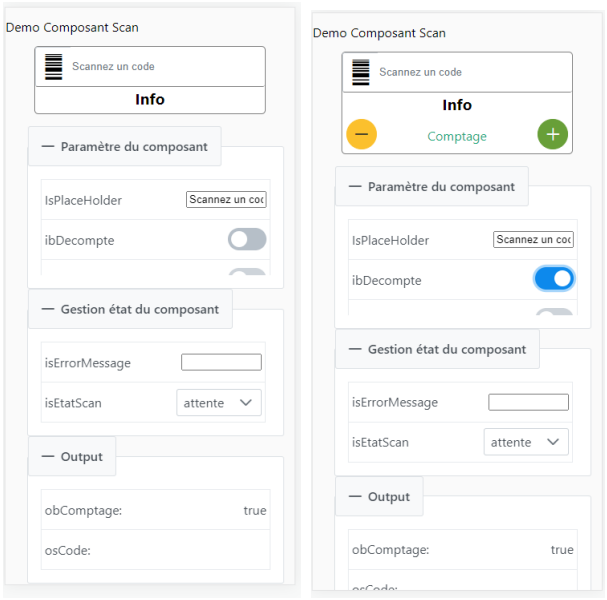


FIGURE 14 – Page de test du composant scan

De plus dans cette page, le composant est fonctionnel. Il y a la possibilité, en indiquant l'URL d'une API, de faire appel au backend et de voir le retour en fonction du code scanné. La réponse de l'API est mise sous forme de tableau avec les différents champs de retour, quand on change d'API, le tableau est remis à zéro.

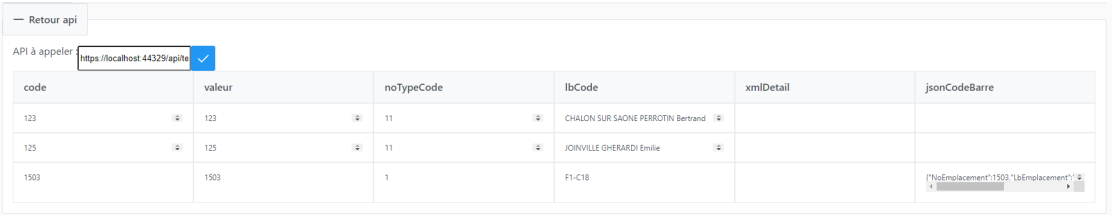


FIGURE 15 – Retour des APIs

5.2 Les APIs

Beaucoup d'APIs ont été réalisées pendant le stage, la création des APIs étant quasiment tout le temps identique, celle pour récupérer les informations sur les codes barre servira d'exemple. Les APIs créées sont des APIs Representational State Transfert (REST) et s'intègrent dans un projet existant contenant déjà des APIs utilisées dans TPE. Une API REST permet de mettre à disposition des données ou des services via des requêtes HyperText Transfert Protocol (HTTP). Ces requêtes peuvent être appelées depuis n'importe quel langage ou framework ayant la possibilité d'appeler des URLs (une application web dans notre cas).

5.2.1 Exemple pour récupérer les informations d'un code-barre

Tout d'abord, nous devons créer deux objets, le modèle qui représente ce que retourne la base de donnée (chaque champ retourné par la base correspond à un attribut de l'objet). Et un deuxième qui servira de Data Transfert Object (DTO), l'objet qui sera renvoyé par l'API au frontend.

La première couche est le *controller*, c'est lui qui va gérer les requêtes entrantes.

```
1     [HttpGet("api/texCommun/info")]
2     public InfoCodeBarreDTO GetInfoCodeBarre([FromQuery] string
          psCode, [FromQuery] bool pbDetailed)
3     {
4       return _mapper.Map<InfoCodeBarreDTO>(_codeBarreService.
            GetInfoCodeBarre(psCode, pbDetailed));
5     }
```

Extrait de code 4 – Méthode du controller

Cette méthode sera exécutée quand une requête arrivera sur le serveur avec l'adresse */api/texCommun/info*. Les paramètres de la méthode sont récupérés depuis l'URL, que l'on doit passer en *query string*. Pour demander les informations d'un code à l'API, l'URL sera donc de la forme *.../api/texCommun/info ?psCode=123456& ?pbDetailed=true*. La méthode retourne donc un DTO qui est obtenu via le mappage de ce que retourne la méthode d'un *service*.

Les services permettent d'ajouter une logique métier à la requête. Dans notre cas, on doit uniquement récupérer des informations depuis la base, et tous les

paramètres sont récupérés depuis l'URL de la requête, la méthode du service sert donc uniquement à appeler une méthode du repository.

```
1    public InfoCodeBarre GetInfoCodeBarre(string psCode, bool
        pbDetailed)
2    {
3      return _codeBarreRepository.GetInfoCodeBarre(psCode,
          pbDetailed);
4    }
```

Extrait de code 5 – Méthode du service

Ce sont les méthodes dans les repositories qui vont appeler la base et lancer les exécutions des procédures stockées.

```
1    public InfoCodeBarre GetInfoCodeBarre(string psCode, bool
        pbDetailed)
2    {
3      return CreateSqlQueryBuilder()
4          .UseSP("sp_Com_InfoCodeSelOne")
5          .AddParams(
6            new SqlQueryParam("Code", psCode),
7            new SqlQueryParam("Detailed", pbDetailed))
8          .SingleOrDefault<InfoCodeBarre>();
9    }
```

Extrait de code 6 – Méthode du repository

Cette méthode exécute et récupère ce que retourne la procédure stockée indiquée.

5.3 Page d'information sur un code barres

Dans Mobile Stock, une boîte à outils est disponible, elle propose plusieurs fonctionnalités comme la recherche d'un article dans le stock ou afficher le stock d'un emplacement. Un autre fonctionnalité permet de scanner n'importe quel type de code barres pour avoir des informations sur celui-ci. Une procédure stockée est utilisée pour retourner ces informations, l'application affiche simplement les champs que retourne cette procédure :

	Code	Valeur	NoTypeCode	LbCode	XmlDetail
1	6274080000265	627408000026	10	6274080 ENS LEGG TUN VICHY 8ANS COQUELICOT PET...	<Detail NoArticle="6274080" />

FIGURE 16 – Retour de la procédure sp_Com_InfoCodeSelOne

FIGURE 17 – Existant de l'info code-barre

5.3.1 Améliorations

Sur la nouvelle application, on peut facilement rendre les informations visuellement plus agréables, et ne pas juste afficher à la suite les information retournées par la procédure comme sur l'existant. On peut aussi afficher d'avantage d'informations. La procédure a donc été modifiée, et retourne une colonne en plus, avec toutes les informations que l'on souhaite afficher sous forme de JavaScript Object Notation (JSON). Le code étant un peu long, l'exemple pour la génération du JSON contenant les informations d'une référence* est disponible en annexe D.

Voici le JSON obtenu :


```
CdTypePrix: "PET"
ImageDocIdCouleur: "97E63A55-3566-4227-BAF6-D4A3C1756281"
LbArticle: "CARD STYLE CHANEL"
LbCouleur: "MARINE"
LbTaille: "T 5"
NoArticle: 2071501
NoReference: 907150100017
Prix: "22.99€"
```


FIGURE 18 – JSON généré

Une fois le JSON retourné dans le frontend, il n'y a plus qu'à afficher ses informations. Pour le code prix, une fonte d'icônes est disponible dans laquelle on peut trouver l'icône correspondant. Pour l'image de la couleur, c'est un docId qui est récupéré dans la base (un code unique qui correspond à un document). On récupère l'image correspondante avec un autre appel d'API qui permet de télécharger ce document dans la Gestion Électronique de Document (GED).
Voici à quoi ressemble la page quand le code d'une référence* est saisi :

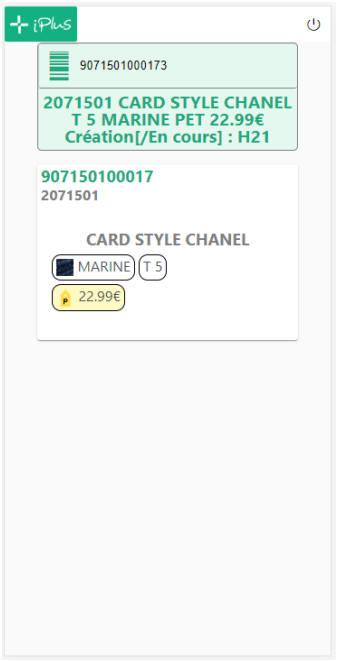


FIGURE 19 – Page d'information sur un code barres

Sur cette page, j'ai pu définir les différents backlogs ainsi que les tâches sur Azure. Mais je ne les ai pas toutes implémentées. Du coté frontend, j'ai créé la page et le composant « card » pour les informations liées à un emplacement Les autres cards ont été faites par un autre développeur. Ceci m'a permis d'échanger beaucoup avec lui, de mieux voir la complexité de la définition des tâches, et de voir que ce qui était attendu n'était pas forcément très bien expliqué dans mes tâches. Étant un peu plus à l'aise avec Angular au moment du développement, j'ai également pu expliquer certains aspect du framework.

5.4 Page de login

5.4.1 Analyse et recherche d'amélioration

Pour se connecter à l'application existante, les utilisateurs cherchent leur nom dans une liste et saisissent leur mot de passe.

FIGURE 20 – Page de login de l'existant

On remarque rapidement que pour les utilisateurs, la recherche de son nom n'est pas très pratique, ils doivent scroller jusqu'à trouver leur nom, en sachant que le tactile du CN50 n'est très réactif comme ce que nous pouvons avoir sur nos smartphones aujourd'hui. Deux moyens de recherche on donc été mis en place

pour améliorer l'expérience utilisateur : un champ de recherche classique, et une barre, contenant des lettres, qui permet en appuyant dessus de directement faire défiler la liste aux noms qui commencent par cette lettre.

5.4.2 Conception de la page

Les APIs pour gérer la connexion et la gestion des sessions existaient déjà sur TPE et ont donc pu être réutilisées. Pour récupérer la liste des utilisateurs, la procédure existante sur MobileStock a également été réutilisée, mais l'API pour l'appeler a du être faite. Pour l'affichage de la liste, un composant PrimeNG (*Accordion*) a été utilisé. Il a permis d'afficher le nom des utilisateurs, et au clic sur un nom d'afficher un champ pour entrer le mot de passe. Pour le composant avec les lettres permettant le scroll, un composant open-source répondant aux besoins a été trouvé, et après quelques tests a pu être intégré à la page [5]. Seul l'affichage de la lettre sélectionnée a été ajouté en plus du composant de base.

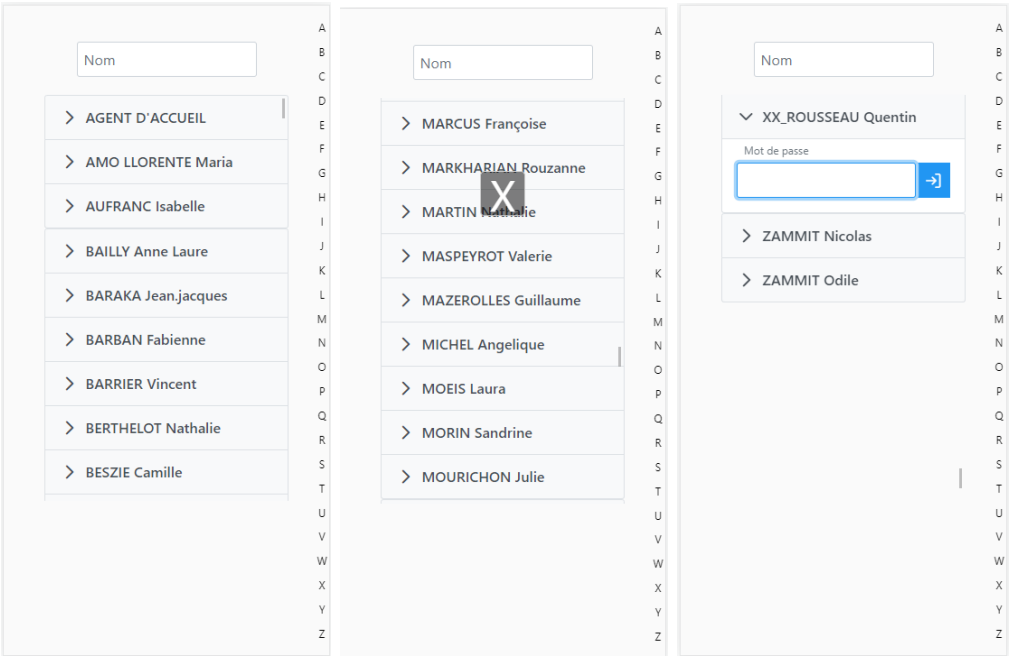


FIGURE 21 – Page de login de l'application

5.5 Menu de l'application

5.5.1 Menu principal

Peu de changement entre le menu de l'ancienne et de la nouvelle application, les menus et les sous-menus gardent le même nom. Uniquement le style évolue pour cette partie. Au niveau du code, le menu est un objet contenant soit un tableau de sous-menus, soit un lien de redirection vers une page qui liste les lectures en fonction du menu que l'on a sélectionné. On peut également se déconnecter via cette page de menu, on demande tout d'abord la confirmation à l'utilisateur grâce à un composant PrimeNG *ConfirmDialog*.

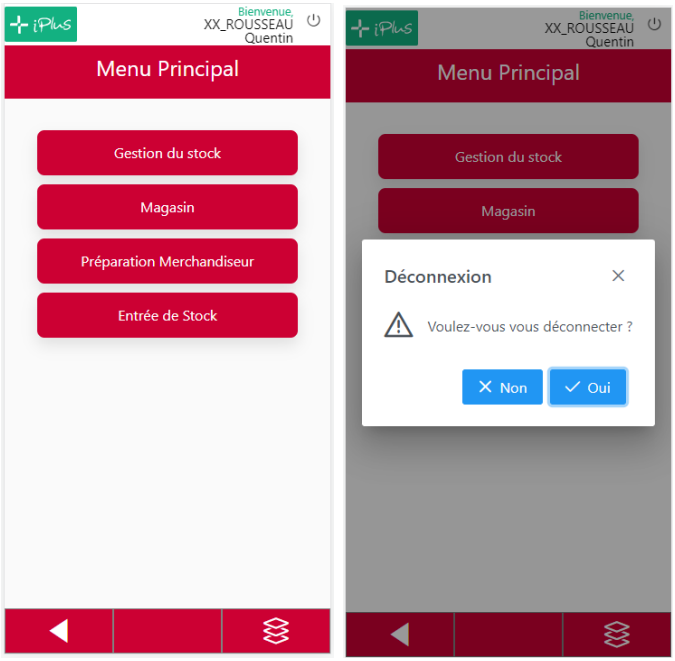


FIGURE 22 – Menu de l'application

5.5.2 Page de de lecture

Cette page liste toutes les lectures d'un type de travail. Cette page est générique pour tous les types de travaux. Au niveau du menu, quand un lien de redirection est choisi, on passe le type de travail correspondant en *queryParams*. An Angular, les *queryParams* permettent de faire circuler des données dans une URL, sur le même principe que les *query string* pour les APIs. En arrivant sur la page, on récupère ce paramètre dans l'url et on peut effectuer une requête au backend pour avoir la liste des lectures correspondantes. Cette page permet également de voir si un utilisateur n'a pas de rame en cours sur le type de travail sélectionné. Si des rames sont en cours, l'utilisateur doit les clôturer avant de pouvoir choisir une autre lecture, on cache donc les lectures et on affiche le travail qui doit être terminé.

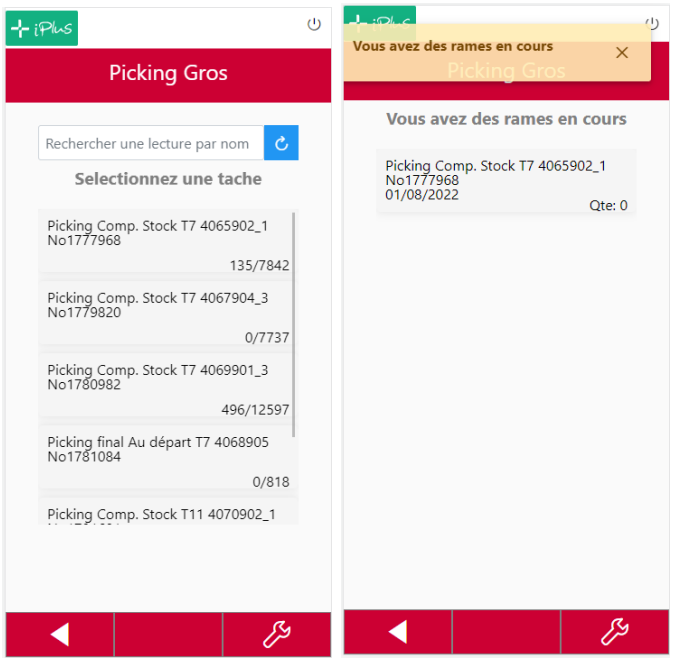


FIGURE 23 – Page de la liste de lectures

5.5.3 Barre de navigation

Il a également fallut créer une barre de navigation dans l'application. Dans Mobile Stock, un bouton permettant d'accéder à des outils était disponible à chaque instant. Ce bouton se retrouve donc dans la barre de navigation. Pour le retour en arrière, utiliser simplement le retour du navigateur n'a pas été une bonne solution. Dans une même page, le fait d'appuyer sur le bouton retour arrière doit avoir différents comportements et ne pas uniquement afficher la page précédemment visitée. Par exemple, sur la page du menu, si nous sommes dans un sous menu, le fait d'appuyer sur le bouton retour doit afficher le menu parent. Alors que si c'est le menu principal qui est affiché, un appui sur le bouton retour on doit afficher la pop-up de déconnexion. Finalement, le choix qui a été fait pour la barre de navigation est de laisser aux pages la gestion des évènements issus de la barre de navigation. Quand le bouton retour est sélectionné on remonte l'évènement via un output aux pages et ce sont elles qui définissent le comportement à adopter. Le bouton central fonctionne sur le même principe mais il est paramétrable, on peut passer en input du composant un icône à afficher et ainsi avoir un bouton d'action supplémentaire si il y en a la nécessité dans les pages.

5.6 Page de picking

5.6.1 Analyse de l'existant

Dans l'application Mobile Stock, trois programmes principaux permettent de réaliser les différents types de travaux.

	NoTypeInterface	LbTypeInterface
1	1	Lecture multi-vues : 10 derniers lus
2	2	Picking
3	3	Lecture multi-vues : articles
4	4	Lecture multi-vues : emplacements
5	5	Répartition

FIGURE 24 – Différentes interfaces présentes dans Mobile Stock

La première interface a être implémenté est celle des pickings. Sur l'application existante, cette interface possède 3 « vues » différentes :

— Emplacement : la liste des emplacements dans lesquels des articles sont à picker,
— Article : les articles à picker dans un emplacement,
— Référence : les références à picker d'un article dans un emplacement.

FIGURE 25 – Existant de l'interface picking

L'analyse de cette partie de l'application a principalement été de voir quelles procédures stockées sont appelées et à quel moment. J'ai donc simuler des pickings sur MobileStock et utiliser un profiler pour lister les procédures nécessaires. Pour cette page de picking, beaucoup d'APIs ont du être mises en place :

— Récupérer les information d'un lecture (avec les réglages PDA),
— Récupérer la liste des emplacements, des articles, et des références,
— Créer une rame,
— Récupérer les informations d'une rame,
— Ajouter des référence à une rame,
— Valider une rame,
— Supprimer une rame,
— Valider une lecture.

5.6.2 Conception de la page

Malgré trois « vues » différentes une seule page pour toute l'interface des pickings a été mise en place. Implémenter trois pages distinctes aurait permit de mieux séparer le code, mais il aurait été nécessaire de faire transiter un grand nombre de données entre ces pages. De plus, bien qu'en Angular, l'échange de données entre des composants parents/enfants est assez simple via les input/output, celui entre des composants qui ne sont pas liés par cette relation est plus complexe. Faire transiter ces données par un service aurait pu être la solution, mais avec les appels d'APIs plus ce partage de données, il aurait été un peu lourd. C'est pour cela que le choix de faire une unique page avec différents composants pour les différentes vues a été mis en place, et en fonction de l'avancement de la lecture, on affiche les bons composants. Il est simple d'indiquer si l'on veut ou non afficher un composant, dans le template d'une page, quand on place le sélecteur d'un composant, on indique en paramètre la propriété **ngIf* suivi d'une expression booléenne. Si cette expression est vrai, le composant est affiché, sinon, il n'est même pas présent dans le DOM, et sera construit quand la condition passera à vrai.

Les mêmes procédures stockées que sur Mobile Stock ont été réutilisées sans être modifiées pour ne pas faire régresser l'existant. Peu de changement au niveau de la logique pour effectuer une lecture, les utilisateurs ayant l'habitude de l'application existante, le but n'était pas de tout changer d'un seul coup. À l'avenir, des évolutions pourront néanmoins être apportées plus facilement que sur Mobile-Stock. Les évolutions portent donc principalement sur l'affichage, et sur les appels des APIs qui ne sont pas perceptibles par l'utilisateur.

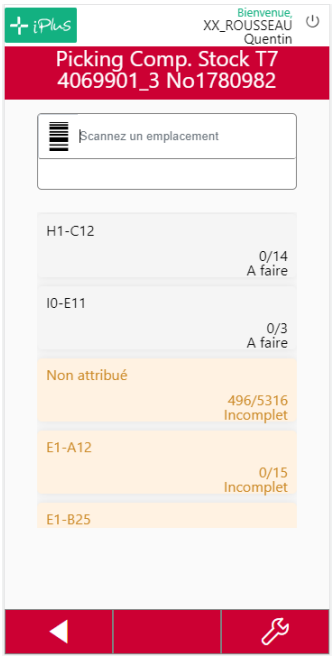


FIGURE 26 – Liste des emplacements

En arrivant sur la lecture, on affiche donc la liste des emplacements, pour chacun d'entre eux, on affiche son nom, les quantités prévues à picker, et son état (À faire, Incomplet, Terminé). Pour chaque état on change la couleur de l'emplacement. L'utilisateur peut sélectionner un emplacement sans restrictions pour voir quels articles sont à picker dans celui-ci.

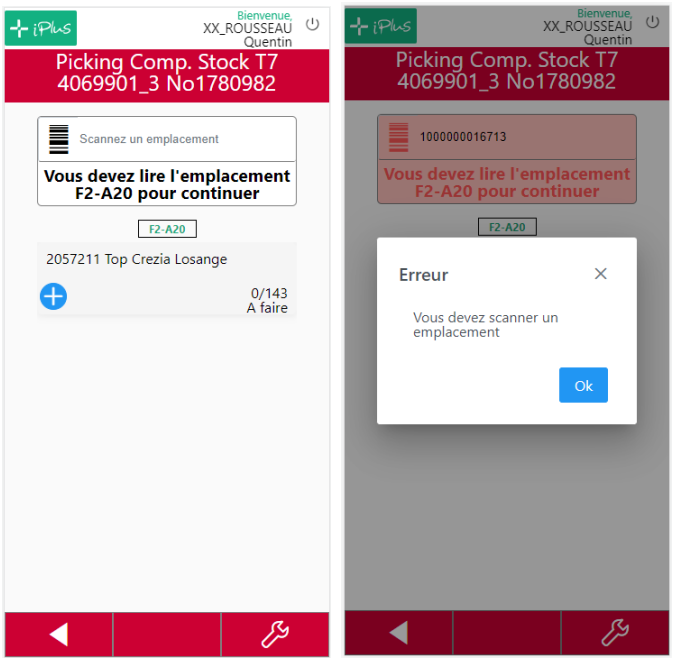


FIGURE 27 – Liste des articles

La liste des articles dans un emplacement est affiché de la même façon, toujours avec le code couleur en fonction de son état. Cependant, il est nécessaire de scanner l'emplacement pour pouvoir afficher la liste des articles. Encore une fois, grâce aux RegExp, on peut vérifier si le code lu est un code emplacement, on évite ici un appel d'API pour vérifier le type de code. Quand un emplacement est sélectionné, l'utilisateur est obligé de scanner l'emplacement qui est affiché, le fait de scanner un code emplacement créera une nouvelle rame. Pour certains emplacements, le scan n'est pas obligatoire, la rame est donc créée automatiquement à la sélection de l'emplacement. De plus quand la liste des emplacements est affichée, le scan d'un code emplacement est possible, ce qui créera également une rame et affichera l'emplacement scanné.

Une fois une rame créée, on peut scanner des références. Quand une référence est scannée on affiche la liste des références à picker du même articles dans l'emplacement en cours. Le composant scan indique si les références lu ont bien été ajoutées à la rame, en fonction de son état.

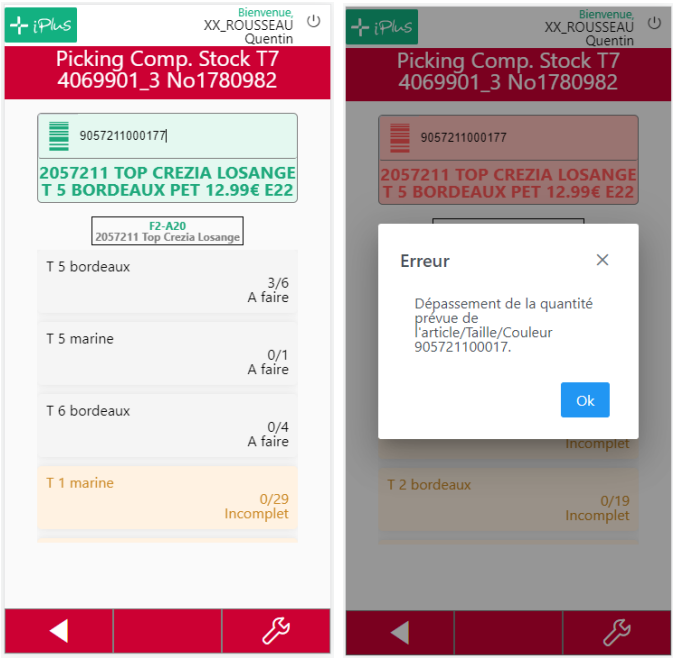


FIGURE 28 – Liste des références

Une fois la bonne quantité ajoutée à la lecture, si on re-scanne la même référence, on affiche une erreur pour indiquer que la quantité prévue est atteinte. Ici aussi les RegExp sont utilisées, dans cette configuration, on bloque le scan des emplacements et des gencode*. Les gencodes ne sont pas utilisés à *Textilot* mais sont présents sur les étiquettes pour les magasins où les articles sont vendus. On les bloque donc au moment du scan pour éviter un appel d'API ou le serveur nous retournera que le code n'est pas connu. Un exemple d'étiquette est disponible en annexe E.

Une fois les références pickées, en appuyant sur le bouton retour de la barre de navigation, on retourne sur la liste des articles, puis des emplacements, ce qui validera automatiquement la rame en cours.

Cette page représente le sprint le plus long que j'ai pu réalisé au cour du stage. À la définition des différents backlogs et des tâches, le temps prévu était d'environ

80 heures, et le développement a duré deux semaines complètes. Sur ce sprint j'ai réalisé l'entièreté du frontend mais un autre développeur à réaliser quelques APIs qui sont utilisées dans cette page.

6 Bilan

Ce stage m'a donc permis de prendre pleinement part à un projet, aussi bien au niveau de l'analyse que du développement. J'ai donc pu réaliser beaucoup de documentation sur *Mobile Stock*, l'application existante, pour mieux comprendre son fonctionnement. Bien qu'un peu longue, cette phase d'analyse et de retro-engenering a été primordiale. L'application existante a été une très bonne base pour la nouvelle application, les utilisateurs ayant l'habitude de celle-ci, l'objectif n'était pas de tout révolutionner. Les différents diagrammes de flux réalisés pourront servir au développement des autres parties qui n'ont pas pu être faites dans la cadre du stage.

Après plusieurs échanges avec un relecteur et après avoir passé la phase de test, la nouvelle application a pu être intégrée à TPE et déployée en production. Pour rappel, voici les différentes parties du projet qui ont pu être réalisées :

— Un composant permettant le scan de code-barres,
— Une page permettant d'afficher des informations sur un code-barres,
— Une page de login,
— Le menu de la nouvelle application,
— Une page listant les lectures selon un type de travail,
— Une page permettant de réaliser des lectures de type picking.

Des tests de performances sont en cours d'études, les temps de réponse depuis le cipherlab (qui est connecté en wi-fi) sont plus longs que ceux observés pendant le développement. Au cours d'une lecture, le temps entre la lecture du code-barres et le retour des APIs qui indiquent aux utilisateurs que la référence a bien été lue est de l'ordre d'une demi seconde. Le projet devra donc probablement passer par une phase d'optimisation avant de pouvoir être utilisé pleinement à *Textilot*. D'autres tests en conditions réelles dans les ateliers sont également prévus prochainement.

7 Conclusion

En conclusion de ce rapport, j'estime que l'objectif du stage est en partie atteint. Le but était de concevoir une application web destinée à la gestion de stock, au moment de la rédaction de ce rapport, l'application est fonctionnelle et déployée en production mais n'est pas encore utilisé par *Textilot*. Au cours de récent test, les temps de réponse ne sont pas aussi bas qu'espérés et sont en cours d'étude.

Ce stage m'a beaucoup apporté tant sur le plan technique que sur le plan humain. J'ai pu monter en compétence sur framework le .NET CORE et le SQL Server, mais principalement sur le framework Angular que j'ai pu réutiliser pour des projets personnels. J'ai également beaucoup appris sur la méthodologie de la gestion d'un projet, beaucoup mieux appréhender la méthodologie agile et tout ce qu'il en dépend. J'ai pu m'investir dans les différentes phase du projet, que ce soit sur l'analyse, le maquettage, le développement ou les tests.

Il reste encore du travail sur le projet. La « base » de la nouvelle application à pu être mise en place, avec une nouvelle interface de connexion et un menu pour accéder aux différents type de lecture. Néanmoins, seule l'interface des pickings à pu être implémenté, deux autres programmes permettant de faire des lectures dans l'application existante n'ont pas encore leur équivalent dans la nouvelle application web. Le composant scan, quand a lui, répond plutôt bien aux exigences et pourra être réutilisé dans différents projets, sa page de présentation permettra aux développeurs de manipuler simplement ses différents paramètres pour savoir comment l'intégrer en fonction des besoins.

Webographie

[1] CNIL, Consulté en Août 2022. URL https://www.cnil.fr/fr/definition/interface-de-programmation-dapplication-api.

[2] Developer Mozilla MVC, Consulté en juillet 2022. URL https://developer.mozilla.org/fr/docs/Glossary/MVC.

[3] Developer Mozilla RegExp, Consulté en juillet 2022. URL https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/RegExp.

[4] Wikipedia SaaS, Consulté en juillet 2022. URL https://fr.wikipedia.org/wiki/Software_as_a_service.

[5] Composant Alphabetical Scroll Bar, Consulté en Juin 2022. URL https://www.npmjs.com/package/alphabetical-scroll-bar.

[6] Formation Angular, Consulté en avril 2022. URL https://angular.io/.

[7] Aide diverses Angular (forum stackoverflow), Consulté de nombreuses fois. URL https://stackoverflow.com/.

[8] Aide diverses C# .net, Consulté de nombreuses fois. URL https://docs.microsoft.com/fr-fr/dotnet/.

[9] Turoriel AxureRP, Consulté en Avril 2022. URL https://docs.axure.com/axure-rp/reference/getting-started-video/.

[10] Documentation cipherlab, Consulté en Avril 2022. URL https://www.cipherlab.com/en/download-c2242/User-Manual.html.

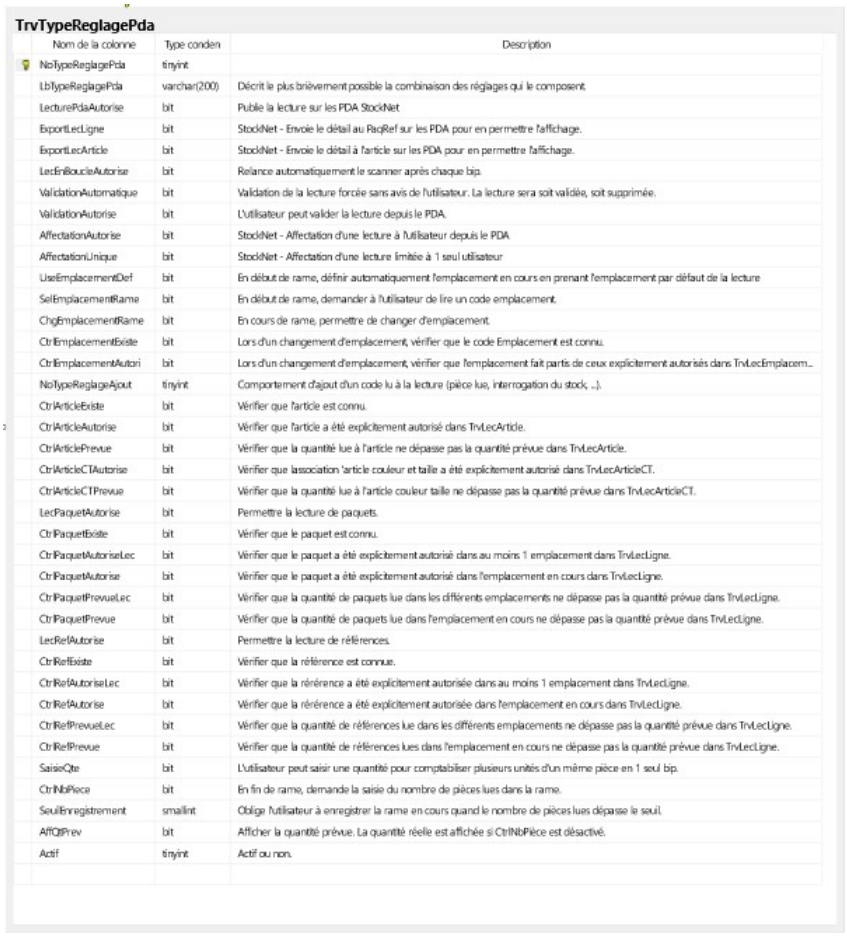
[11] Documentation module de scan cipherlab, Consulté en Mai 2022. URL https://www.cipherlab.com/en/download-c2229/ReaderConfig.html.

[12] Documentation bibliothèque PrimeNG, Consulté de nombreuses fois. URL https://www.primefaces.org/primeng/setup.

Annexes

Table des matières

Annexe A Réglage des PDA



TrvTypeReglagePda

Nom de la colonne	Type conden	Description
NoTypeReglagePda	tinyint	
LbTypeReglagePda	varchar(200)	Décrit le plus brièvement possible la combinaison des réglages qui le composent.
LecturePdaAutorise	bit	Publie la lecture sur les PDA StockNet
ExportLecLigne	bit	StockNet - Envoie le détail au PaqRef sur les PDA pour en permettre l'affichage.
ExportLecArticle	bit	StockNet - Envoie le détail à l'article sur les PDA pour en permettre l'affichage.
LecEnBoucleAutorise	bit	Relance automatiquement le scanner après chaque bip.
ValidationAutomatique	bit	Validation de la lecture forcée sans avis de l'utilisateur. La lecture sera soit validée, soit supprimée.
ValidationAutorise	bit	L'utilisateur peut valider la lecture depuis le PDA.
AffectationAutorise	bit	StockNet - Affectation d'une lecture à l'utilisateur depuis le PDA
AffectationUnique	bit	StockNet - Affectation d'une lecture limitée à 1 seul utilisateur
UseEmplacementDef	bit	En début de rame, définir automatiquement l'emplacement en cours en prenant l'emplacement par défaut de la lecture
SelEmplacementRame	bit	En début de rame, demander à l'utilisateur de lire un code emplacement.
ChgEmplacementRame	bit	En cours de rame, permettre de changer d'emplacement.
CtrlEmplacementExiste	bit	Lors d'un changement d'emplacement, vérifier que le code Emplacement est connu.
CtrlEmplacementAutori	bit	Lors d'un changement d'emplacement, vérifier que l'emplacement fait partis de ceux explicitement autorisés dans TrvLecEmplacem...
NoTypeReglageAjout	tinyint	Comportement d'ajout d'un code lu à la lecture (pièce lue, interrogation du stock, ...).
CtrlArticleExiste	bit	Vérifier que l'article est connu.
CtrlArticleAutorise	bit	Vérifier que l'article a été explicitement autorisé dans TrvLecArticle.
CtrlArticlePrevue	bit	Vérifier que la quantité lue à l'article ne dépasse pas la quantité prévue dans TrvLecArticle.
CtrlArticleCTAutorise	bit	Vérifier que lassociation 'article couleur et taille a été explicitement autorisé dans TrvLecArticleCT.
CtrlArticleCTPrevue	bit	Vérifier que la quantité lue à l'article couleur taille ne dépasse pas la quantité prévue dans TrvLecArticleCT.
LecPaquetAutorise	bit	Permettre la lecture de paquets.
CtrlPaquetExiste	bit	Vérifier que le paquet est connu.
CtrlPaquetAutoriseLec	bit	Vérifier que le paquet a été explicitement autorisé dans au moins 1 emplacement dans TrvLecLigne.
CtrlPaquetAutorise	bit	Vérifier que le paquet a été explicitement autorisé dans l'emplacement en cours dans TrvLecLigne.
CtrlPaquetPrevueLec	bit	Vérifier que la quantité de paquets lue dans les différents emplacements ne dépasse pas la quantité prévue dans TrvLecLigne.
CtrlPaquetPrevue	bit	Vérifier que la quantité de paquets lue dans l'emplacement en cours ne dépasse pas la quantité prévue dans TrvLecLigne.
LecRefAutorise	bit	Permettre la lecture de références.
CtrlRefExiste	bit	Vérifier que la référence est connue.
CtrlRefAutoriseLec	bit	Vérifier que la rérérence a été explicitement autorisée dans au moins 1 emplacement dans TrvLecLigne.
CtrlRefAutorise	bit	Vérifier que la rérérence a été explicitement autorisée dans l'emplacement en cours dans TrvLecLigne.
CtrlRefPrevueLec	bit	Vérifier que la quantité de références lue dans les différents emplacements ne dépasse pas la quantité prévue dans TrvLecLigne.
CtrlRefPrevue	bit	Vérifier que la quantité de références lues dans l'emplacement en cours ne dépasse pas la quantité prévue dans TrvLecLigne.
SaisieQte	bit	L'utilisateur peut saisir une quantité pour comptabiliser plusieurs unités d'un même pièce en 1 seul bip.
CtrlNbPiece	bit	En fin de rame, demande la saisie du nombre de pièces lues dans la rame.
SeuilEnregistrement	smallint	Oblige l'utilisateur à enregistrer la rame en cours quand le nombre de pièces lues dépasse le seuil.
AffQtPrev	bit	Afficher la quantité prévue. La quantité réelle est affichée si CtrlNbPièce est désactivé.
Actif	tinyint	Actif ou non.

FIGURE 29 – Table des réglages des PDA

Annexe B Diagramme de flux

B.1 Procédures stockées utilisées dans les lectures

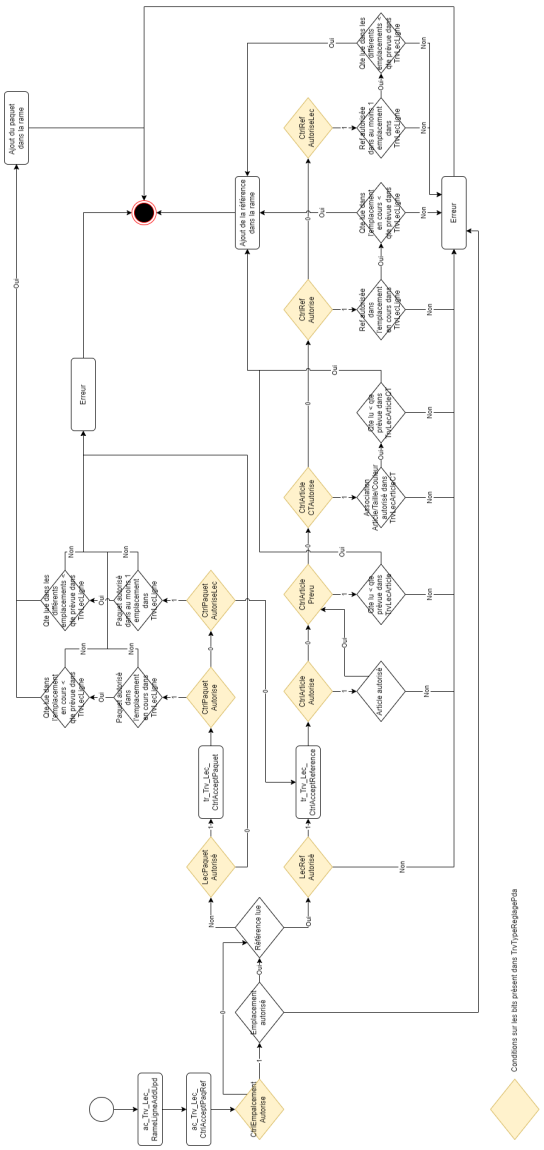


FIGURE 30 – Diagramme de flux : scan d'un paquet/référence

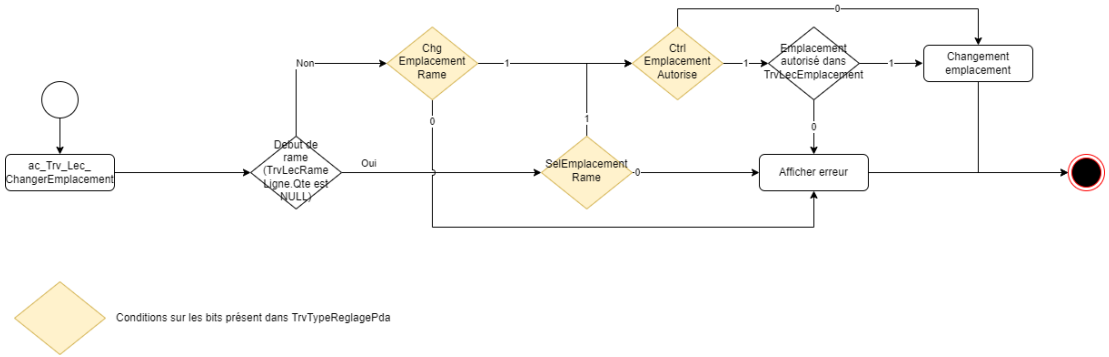


FIGURE 31 – Diagramme de flux : scan d'un emplacement

B.2 Autres digrammes de flux

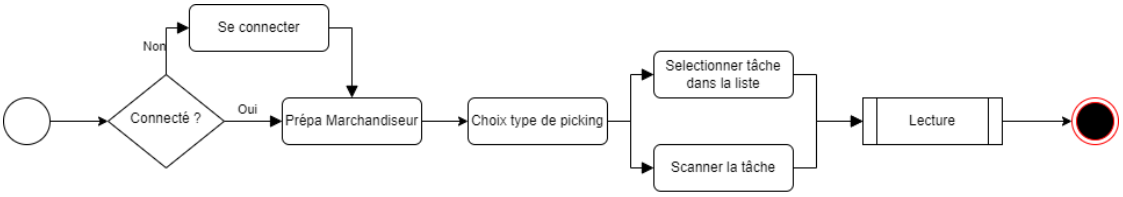


FIGURE 32 – Diagramme de flux : Picking

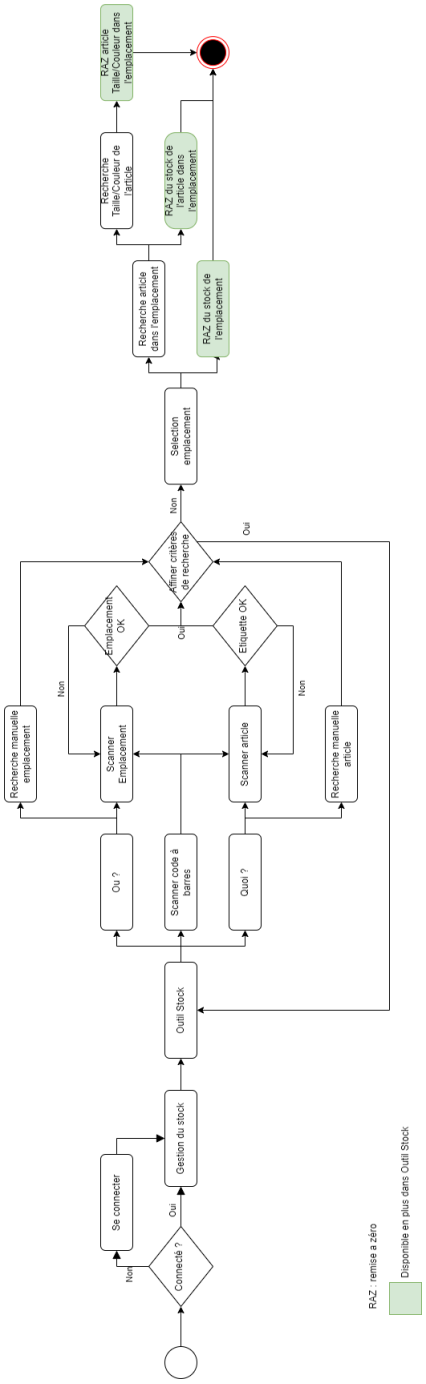


FIGURE 33 – Diagramme de flux : Outils Stock

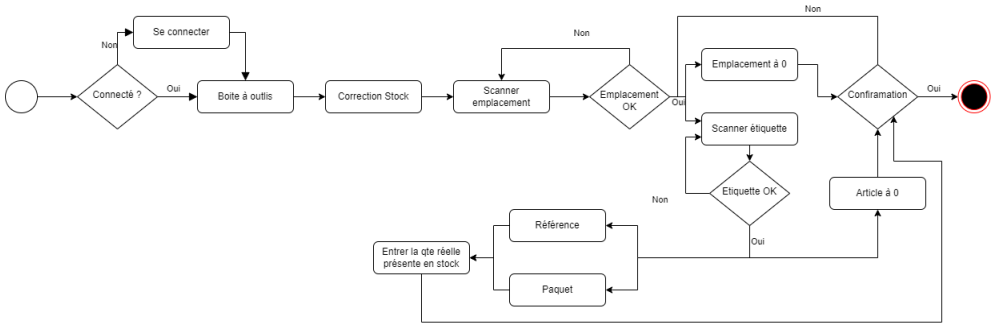


FIGURE 34 – Diagramme de flux : Correction de stock

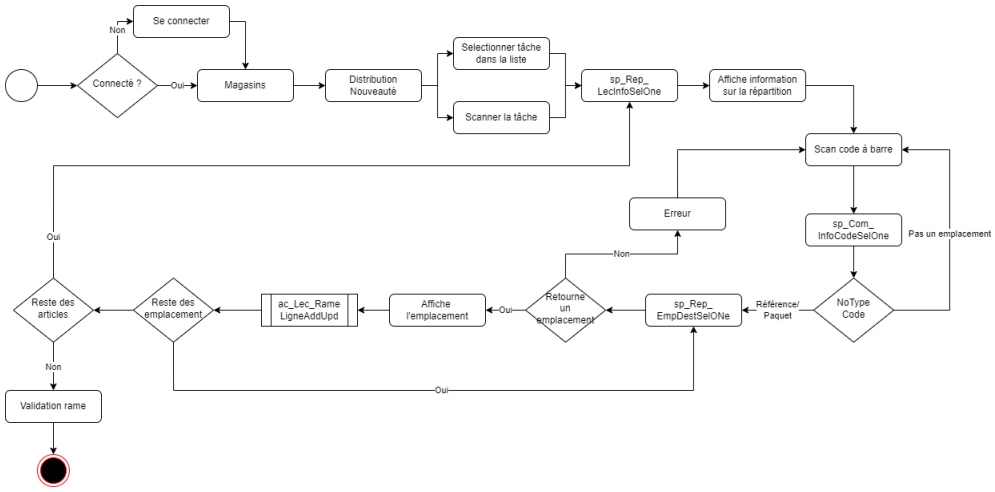


FIGURE 35 – Diagramme de flux : Répartition

Annexe C Composant scan

Nom	Type	Default	Descriprion
INPUT			
isPlaceholder	string	""	Placeholder du composant, peut indiquer quels types de code sont acceptés
ibDecompte	boolean	false	Active ou non l'option de décomptage, fait apparaitre les boutons pour changer de mode
ibPopUp	boolean	false	Permet d'afficher ou nom une pop-up en cas d'erreur
isErrorMessage	string	""	Message d'erreur à afficher dans la pop-up
isLbInfo	string	""	Texte à afficher dans le label info
iaVibrationOK	number[]	[]	Tableau de vibrations en cas de succès
iaVibrationErreur	number[]	[]	Tableau de vibrations en cas d'erreur
isEtatScan	EEtatScan	EEtatScan.Attente	Etat du scan, change la couleur du composant
isRegExRefuse	String	""	Si le code lu correspond à cette expression régulière, il sera refuser
isRegExAccept	String	""	Si le code lu ne correspond pas à cette expression régulière, il sera refuser
isMsgErreurRegEx Accept	String	""	Message à afficher en cas de code refuser par une regex
isMsgErreurRegEx Accept	String	""	Message à afficher en cas de code non accepter par une regex
OUTPUT			
obComptage	EventEmitter<boolean>	/	Emet un event lors du clic sur le bouton de comptage, permet de recupérer dans quel mode est le composant
osCode	EventEmitter<string>	/	Emet un event lors de la lecture d'un code, permet de récupérer ce code

FIGURE 36 – Paramétrage du composant scan

Annexe D Génération d'information sous forme de JSON

```
ELSE IF @IsReference = 1
    BEGIN
        --Code référence
        INSERT INTO @Result(Code, Valeur, NoTypeCode, LbCode, XmlDetail, JsonCodeBarre)
            SELECT
                @Code AS Code
                , ArtReference.NoReference AS Valeur
                , @NO_TYPE_CODE_REFERENCE AS NoTypeCode
                , CAST(ArtReference.NoArticle AS VARCHAR) + CHAR(10) + CHAR(13) +
                    ArtArticle.LbArticle + ' ' + ArtTaille.LbTaille + ' ' + ArtCouleur.LbCouleur +
                    CHAR(10) + CHAR(13) + ArtTarif.CdTypePrix + ' ' +
                    CAST(ROUND(ArtTarif.PxVteDev, 2) AS VARCHAR) + ComDevise.Symbole + ' ' +
                    CASE @Detailed WHEN 1 THEN CHAR(10) + CHAR(13) + 'Création[/En cours] : ' ELSE '' END +
                    SaisonCreate.RefSaison + SUBSTRING(CAST(ArtArticle.AnneeCreate AS VARCHAR(4)), 3, 2) +
                    CASE WHEN NoSaisonEnCours = NoSaisonCreate AND ArtArticle.AnneeEnCours = ArtArticle.AnneeCreate
                    THEN '' ELSE '/' + SaisonEnCours.RefSaison +
                    SUBSTRING(CAST(ArtArticle.AnneeEnCours AS VARCHAR(4)), 3, 2) END + CHAR(10) + CHAR(13) AS LbCode
                , XmlDetail = (SELECT ArtReference.NoArticle FOR XML RAW('Detail'))
                , JsonCodeBarre = (CASE WHEN @Detailed = 1 THEN
                                        (SELECT
                                            ArtReference.NoReference,
                                            ArtArticle.NoArticle,
                                            ArtArticle.LbArticle,
                                            ArtTaille.LbTaille,
                                            ArtCouleur.LbCouleur,
                                            CAST(ROUND(ArtTarif.PxVteDev, 2) AS VARCHAR) + ComDevise.Symbole AS Prix,
                                            ArtTarif.CdTypePrix,
                                            ArtArticleCouleurImage.ImageDocId AS ImageDocIdCouleur
                                        FOR JSON PATH, WITHOUT_ARRAY_WRAPPER)
                                    ELSE NULL END)
            FROM TexArticle..ArtReference
                INNER JOIN TexArticle..ArtArticle ON ArtArticle.NoArticle = ArtReference.NoArticle
                INNER JOIN TexArticle..ArtTaille ON ArtTaille.NoTaille = ArtReference.NoTaille
                INNER JOIN TexArticle..ArtCouleur ON ArtCouleur.NoCouleur = ArtReference.NoCouleur
                INNER JOIN TexArticle..ArtSaison AS SaisonEnCours ON SaisonEnCours.NoSaison  = ArtArticle.NoSaisonEnCours
                INNER JOIN TexArticle..ArtSaison AS SaisonCreate ON SaisonCreate.NoSaison  = ArtArticle.NoSaisonCreate
                INNER JOIN TexArticle..ArtTarif ON ArtTarif.NoTarif = ArtReference.NoTarif
                INNER JOIN Textilot..ComDevise ON ComDevise.CdDevise = ArtTarif.CdDevise
                LEFT JOIN TexArticle..ArtArticleCouleurImage ON
                    ArtArticleCouleurImage.NoArticle = ArtArticle.NoArticle AND
                    ArtArticleCouleurImage.NoCouleur = ArtCouleur.NoCouleur AND ArtArticleCouleurImage.NoTypeImage = 2
            WHERE ArtReference.NoReference = CAST(SUBSTRING(@Code, 1, 12) AS BIGINT)
END
```

FIGURE 37 – Procédure stockée générant le JSON

Annexe E Étiquette article

FIGURE 38 – Exemple d'étiquette