EPSI

430 rue des apothicaires

34000 Montpellier

EMINENCE

Route de gallargues

30470 Aimargues

Florian Roura

Mémoire professionnel

L'évolution d'une application B2B de l'AS400 vers Angular

Tuteur entreprise – Vincent Descreux

Tuteur école – Jean-Luc Bompard

2018-2019

Soutenance le 26/08/19

Remerciements

Je tiens, en premier lieu, à adresser mes remerciements à Monsieur **François Marchand**. En sa qualité de directeur du service informatique, il m'a permis d'intégrer la société Eminence, dans laquelle je suis depuis maintenant près de deux ans. Soucieux, responsable mais néanmoins proche de son équipe, son dévouement et son implication dans les activités du service font de lui un pilier de l'entreprise. J'ai pu, grâce à lui, découvrir l'entreprise et ses services au travers d'une mission intéressante, qui m'a permis de consolider mes connaissances en développement et de m'enrichir au niveau professionnel.

Je remercie également mon tuteur Monsieur **Vincent Descreux** et Monsieur **Sébastien Rousselle**. Sous leur aile j'ai pu m'épanouir dans un environnement favorable au développement personnel et professionnel. Leur expertise et leur expérience non seulement au sein de l'entreprise mais également en tant que développeurs m'ont permis de m'intégrer rapidement dans l'entreprise et d'en apprendre plus sur les ficelles du métier. J'adresse également des remerciements aux autres membres du service informatique, qui contribuent chaque jour à en faire un espace de travail sain et détendu. J'ai une petite pensée émue à Sébastien qui quitte l'entreprise en Juillet 2019, pour voguer vers d'autres horizons.

Pour finir, je remercie également Monsieur **Frédéric Beaumer** et l'ensemble du service Négoce. Les réunions auxquelles j'ai été convié ont été une mine d'informations permettant un développement plus poussé et exaltant. Leur disponibilité et leur retour sur leurs outils et sur leur demande ont été indispensables au développement qui a constitué ma mission.

Résumé du mémoire

EMINENCE, entreprise de fabrication et création de sous-vêtements créée en 1944, est l'une des dernières entreprises à fabriquer une partie de ses produits en France. Possédant plusieurs sites, elle a décliné son secteur d'activité qui était le slip pour homme pour pouvoir maintenant produire des boxers, des caleçons, des chaussettes, des T-shirts, des débardeurs, des pyjamas et des sous-vêtements chauds. Son architecture tourne autour de l'IBM System I ou AS400, interface de type console avec des zones modifiables. Cette interface était certes révolutionnaire à l'époque mais manque aujourd'hui de fraîcheur, d'accessibilité et de fonctionnalités.

En son sein existe un service nommé le service Négoce, qui fait appel à des fournisseurs externes afin de confectionner les produits d'Eminence. Les commandes qui lui sont transmises sont acheminées par le service de l'ordonnancement, qui vérifie à tout moment que l'usine d'Aimargues possède les produits nécessaires à l'envoi des commandes reçues. Après vérification, la commande est envoyée au fournisseur par mail, en compagnie de pièces jointes comme des documents techniques spécifiant les processus de création ou encore le packaging.

Le service Négoce n'a, pour vérifier les commandes, accès qu'à l'AS400 qui est la seule ressource ayant ce type d'information. L'interface est mal adaptée, et les employés doivent constamment jongler entre l'émulateur AS400, le logiciel de mails et le site WeTransfer, les pièces jointes n'étant transmissibles qu'en interne à cause de leur taille. Des problèmes de visibilité dus aux restrictions graphiques de l'AS400 et des problèmes sur certains champs de caractères compliquent le fonctionnement du service, qui peine à tenir la cadence avec la quantité de commandes reçue. Une demande a été effectuée auprès du service informatique pour pouvoir bénéficier d'une toute nouvelle interface et de pouvoir résoudre les problèmes rencontrés.

Des critères de réussite sont donc établis pour évaluer les performances de l'application et s'assurer, autant pendant le développement que pendant son utilisation ultérieure, qu'elle répond correctement à la problématique et correspond aux besoins utilisateurs. Ces critères sont le découlement direct de l'analyse des problèmes rencontrés ; La nouvelle interface se doit d'être axée sur ces principaux problèmes. On retiendra principalement la capacité d'intégration, l'ouverture d'accès, l'intuitivité, l'évolutivité ainsi que le dynamisme comment étant des points clés nécessaires pour obtenir une solution répondant aux besoins analysés.

Disposer de ces critères permet de dresser un inventaire des choix possibles auxquels l'entreprise sera confrontée au vu de ses outils : Elle a en effet le choix entre adapter l'interface de l'AS400 directement pour le Web avec des logiciels comme aXes ou Silverdev, reprogrammer les écrans en utilisant des langages comme le COBOL ou le RPG pour le web, ou prendre une approche différente en créant une interface web, soit sur le portail classique, en PHP procédural, ou bien sur le portail Symfony, en créant un module pour l'application. Symfony est ici le choix idéal, étant déjà mis en place et proposant une structure pour l'affichage des pages ; La création d'un site en PHP procédural serait en effet ici trop chronophage, et l'AS400 reste une technologie que l'on souhaite aujourd'hui éviter.

Construire l'interface de toutes pièces représente le même problème. Il existe pour cela plusieurs frameworks ou librairies permettant des développements rapides et uniformes, facilement intégrables au back-end Symfony. Il conviendra de

les comparer pour choisir la librairie idéale à implémenter et référencer tout au long du projet. La comparaison est ici faite entre trois grandes librairies existantes, axées sur le langage JavaScript, qui sont ReactJs, VueJs et AngularJs. Les deux premières technologies sont certes très adaptées dans le cas où l'on souhaite développer des composants applicatifs, mais sont absolument incapables d'être utilisés en tant que tels, du moins de manière simple ; Le but étant d'inclure le minimum nécessaire en maximisant les fonctionnalités. AngularJs propose tout un set de fonctionnalités et de composants qui en fait la solution idéale. De plus, il est facile de le mettre en place, et la formation dans ce nouveau framework ne m'a pris que peu de temps.

Le projet doit maintenant être mis en place, suivant un plan défini et personnalisé en fonction de la solution : Si l'on applique des étapes comme la définition du besoin, l'équipe de développement ou encore la planification des tâches, il faut ici se concentrer sur l'établissement d'une politique de structure et d'affichage, ou encore la planification des différents développements en fonction de cette politique. Un cycle itératif sera adopté pour le développement, afin de maximiser l'échange d'informations et de tenir le Négoce au courant des modifications et conceptions en cours. Par ailleurs, ce cycle de projet permettra dans des développements futurs, de maximiser l'évolutivité de la solution en favorisant un développement segmenté et modulaire de l'application. La politique de structure et d'affichage sera assurée par la librairie AngularJS Material, qui est une implémentation de la philosophie du Material Design mis au point par Google. Disposant de composants réutilisables et uniformes, on est donc certains d'avoir une interface logique et conforme à cette politique. Il ne faut cependant pas oublier de définir par quels procédés graphiques les fonctionnalités doivent apparaître, comme afficher une liste de commandes ou proposer un dialogue à l'utilisateur.

Chacune des entités clients disposant d'une interface semblable à l'autre, avec des fonctionnalités supplémentaires, on choisira de développer l'interface la plus complexe d'abord, pour ensuite l'adapter et défalquer des fonctionnalités pour l'autre entité client. On développera d'abord l'interface des commandes côté Négoce, pour l'adapter ensuite pour les fournisseurs, avant de faire l'opération inverse, consistant à concevoir la partie des expéditions pour les fournisseurs, avant d'en créer un ersatz limité pour le service Négoce.

L'entreprise se retrouve donc à choisir maintenant entre un prestataire externe et un développement interne réalisé au sein de la société. Une analyse des avantages et inconvénient montrera que si les compétences d'un développeur chevronné sont certes toujours bénéfiques, il n'en résulte pas moins une grosse prise de risques au niveau de la sécurité mais également du maintien de la solution, ainsi que de son support et de ses développements futurs. Compte tenu des besoins du service, il est nécessaire de la développer en interne.

L'interface a donc été développée en suivant les principes de la théorie de l'UX/UI. Etant le point phare de ce développement, il faut s'étendre sur ce qui caractérise une interface correcte, et ce qui constitue une expérience utilisateur à la fois agréable et répondant aux problèmes soulevés par le service. Il sera montré grâce à une modélisation des processus métier, post-développement, que l'application développée suivant le contenu de ce mémoire, correspond à ce qui était attendu, et permet une simplification conséquente du fonctionnement actuel. L'interface est précise, uniforme, et permet de jeter un œil nouveau sur les activités du service Négoce.

Abstract

EMINENCE, a company manufacturing and creating underwear created in 1944, is one of the last companies to still manufacture some of its products in France. Possessing several sites, it has declined its sector of activity which was man underwear to be able to now produce boxers, underpants, socks, T-shirts, tank tops, pajamas and warm underwear . Its architecture revolves around the IBM System I or AS400, console type interface with editable areas. This interface was revolutionary at the time but nowadays lacks freshness, accessibility and functionality.

There is a service called the Trade Department, which uses external suppliers to manufacture Eminence's products. The orders are forwarded by the Ordering Department, which checks that the factory in Aimargues owns all necessary products needed for upcoming orders at any time. After verification, orders are sent to the supplier by email, along with attachments, such as technical documents about the creation process or the packaging.

The trading department only has access to the AS400, which is the only resource containing such information. The interface is poorly adapted, and employees have to constantly juggle between the AS400 emulator, the mailer software and the WeTransfer site, attachments locked inside the servers because of their size. Visibility problems due to graphic restrictions of the AS400 and problems on certain character fields complexify the service's operations, which is struggling to keep pace with the amount of orders received. A request was made to the IT department to be able to benefit from a brand new interface and to be able to solve the problems encountered.

Success criteria are therefore established to evaluate the performance of the application and ensure, both during development and during its subsequent use, that it answers the problem accordingly and fits the user's needs. These criteria are the direct flow of the analysis of the problems encountered ; The new interface needs to focus on these key issues. We will mainly retain the integrability, open access, intuitiveness, scalability and dynamism as key points needed to obtain a solution meeting the needs analyzed.

Having these criteria allows us to draw up an inventory of the possible choices that the company will face in view of its tools : It has the choice between adapting the interface of the AS400 directly for the Web with software like aXes or Silverdev , reprogram the screens using languages like COBOL or RPG for the web, or take a different approach by creating a web interface, either on the classical portal, in procedural PHP, or on the Symfony portal, by creating a module for the application. Symfony is here the ideal choice, being already set up and offering a structure for the display of the pages; The creation of a site in procedural PHP would indeed be too time-consuming here, and the AS400 remains a technology that we wish to avoid today.

Building the interface from scratch represents the same problem. For this purpose, there are several frameworks or libraries allowing fast and uniform developments that can be easily integrated into the Symfony back-end. They should be compared to choose the ideal library to implement and reference throughout the project. The comparison is made here between three large existing libraries, focused on JavaScript, which are ReactJs , VueJs and AngularJs . The first two technologies are certainly very suitable in the case where it is desired to develop application components, but are absolutely incapable of being used as standalone, at least in a simple manner ; The goal is to include the minimum necessary by maximizing the features. AngularJs offers a whole set of features and components that make it the ideal solution. In addition, it is easy to set up, and training in this new framework took me only a short time.

The project must now be implemented, according to a defined and customized plan defined beforehand : If we apply steps such as the definition of the need, the development team or the task planning, it is necessary here to focus on establishing a structure and display policy, or planning different developments according to this policy. An iterative cycle will be adopted for development, in order to maximize the exchange of information and keep the Trade Department informed of current changes

and designs. In addition, this project cycle will allow in future developments, to maximize the scalability of the solution by promoting a segmented and modular development of the application. The structure and display policy will be ensured by the AngularJS Material library, which is an implementation of the Material Design philosophy developed by Google. Having reusable and uniform components, we are therefore sure to have a logical interface that complies with this policy. However, we must not forget to define by which graphical processes the functionalities must appear, such as displaying a list of commands or proposing a dialogue to the user.

Each of the client entities with a similar interface to the other, with additional features, we will choose to develop the most complex interface first, and then adapt and deduct features for the other client entity. We will first develop the order interface on the trading side, then adapt it for the suppliers, before doing the opposite operation, consisting of designing the part of the shipments for the suppliers, before creating a limited ersatz for the trading department.

The company finds itself now choosing between an external provider and an internal development within the company. An analysis of the advantages and disadvantages will show that while the skills of a seasoned developer are certainly always beneficial, the result is not only a big risk-taking in terms of security but also the maintenance of the solution, as well as its support and its future developments. Given the needs of the service, it is necessary to develop it internally.

The interface has been developed following the principles of the UX / UI theory. Being the highlight of this development, we must expand on what characterizes a correct interface, and what constitutes a user experience both pleasant and responding to the problems raised by the service. It will be shown through a modeling of business processes, post-development, that the application developed according to the content of this memory, corresponds to what was expected, and allows a significant simplification of the current operation. The interface is precise, uniform, and allows you to take a fresh look at the activities of the trading department.

Table des matières

Introduction

Les technologies de l'information ont, depuis quelques années, radicalement changé. Si les structures informatiques restent stables malgré leur fonctionnement ancien, beaucoup recherchent la nouveauté et l'efficacité que peuvent apporter les nouvelles technologies.

Cependant, beaucoup d'entreprises peinent à mettre à jour leurs processus internes ; En résultent des fonctionnements paléontologiques qui ont pu témoigner d'un semblant d'innovation lors de leur mise en place, mais qui aujourd'hui compliquent inutilement les processus métier. Par ailleurs, dans la plupart des grandes entreprises dont le cœur de métier n'est pas le développement informatique pur, la majorité du corpus des employés ne dispose que de connaissances limitées en matière d'utilisation de l'informatique. Beaucoup d'entre eux sont peu désireux d'apprendre de nouvelles méthodes et sont contraints à utiliser les interfaces et outils d'antan. Ceci pose un réel frein à la mise à jour des interfaces, un pourcentage conséquent de cette population se voyant noyé sous les innovations.

S'il semble trivial de mettre à jour un système graphique, il n'est en réalité pas uniquement la résultante d'une analyse du client cible. Il faut également proposer une interface qui allie efficacité et lisibilité. On peut aujourd'hui constater de nombreuses technologies et entreprises focalisées sur l'apparence des interfaces. Mais le risque engendré par toutes ces interfaces différentes les unes des autres est bien réel : Elles se dressent souvent entre l'utilisateur et le fonctionnement du service. Sensées offrir à l'utilisateur des opérations simples et précises, la recherche toujours plus poussée des fonctionnalités perd de vue l'efficacité qui est voulue et délaie de plus en plus les opérations que l'utilisateur veut effectuer.

Par ailleurs, un manque de distinction crucial est observé de manière générale entre l'interface et l'expérience utilisateur. Il est trop souvent supposé qu'une interface jugée « belle », reposant sur des critères purement esthétiques définis par la recherche graphique et artistique, puisse en réalité offrir une expérience utilisateur nouvelle. A l'inverse, lors de la recherche de l'efficacité extrême au sein d'une interface, le côté graphique est négligé, beaucoup de développeurs faisant fi de l'apparence au profit du fonctionnement.

De nombreuses technologies voient aujourd'hui le jour, se focalisant sur l'apparence de l'application tout en étant soucieuses du fonctionnement interne des composants. Les utilisateurs jugeant la qualité d'une application à son UI/UX, il est essentiel de polir cet aspect afin de garantir une utilisation efficace et agréable pour l'utilisateur. Il est donc courant de voir des entreprises ou des éditeurs de logiciels choisir les méthodes de conception graphiques les plus courantes et les outils plus communément utilisés. Inclure un script ou une libraire, ou axer son développement sur un langage ou framework particulier devient la solution facile pour régler ce problème. Mais uniformiser les interfaces ne suffit pas et il convient d'étudier les différents besoins que peuvent avoir les utilisateurs, sans oublier les fonctionnalités que l'application se doit d'offrir.

Surfant sur la vague épidémique de l'UI/UX design, nombre de « néo-web-gourous » ont fait de cette nouvelle philosophie leur expertise : Alliant le classique développement d'arrière-plan avec l'interface, ils s'étendent à offrir leur vision

unique des nouvelles interfaces d'aujourd'hui. De la charte graphique aux éléments qui composent l'application, ils sont à même d'être certains que l'application correspond bien au besoin qui est demandé. Il y a également une dimension affective à ce développement, une interface assez bien réalisée autant sur le fond que sur la forme fidélisant généralement les clients. Cependant, faire appel à un prestataire externe n'est pas sans risque, et il n'est jamais judicieux de jeter son dévolu sur cette option avant d'avoir analysé en profondeur les avantages et les inconvénients qu'implique une telle décision.

Tous ces aspects constituent un obstacle, sensiblement complexe à surmonter pour certaines entreprises. En particulier, les entreprises dont les applications voyant leur conception dater de quelques décennies se trouvent confrontés à des problèmes : Les besoins utilisateurs évoluent mais les interfaces restent toujours rudimentaires et peu regardantes des fonctionnalités qu'elles doivent offrir ; Toutes les actions sont plus insignifiantes les unes que les autres et le pouvoir que l'utilisateur devrait pouvoir posséder se retrouve effrité dans les méandres d'applications dont les interfaces regorgent de redirections, d'animations illogiques, de comportements aberrants ou encore de composants logiciels graphiques mal conçus.

Eminence, entreprise textile créée en 1944, est aujourd'hui confrontée à ce défi. Soucieuse de vouloir mettre à jour son fonctionnement, elle est limitée dans ses actions par le recours global aux technologies venant d'IBM, comme l'AS400 (appelé également IBM System I) ou encore la base de données DB2. Le service informatique ne représente qu'une petite partie des employés présents sur le site principal d'Aimargues. Le reste de la force salariale est réparti dans le siège principal ainsi que dans l'usine. Si les employés utilisent quotidiennement des ordinateurs pour mener à bien les opérations qui constituent leur métier, il est généralement constaté de la part du service informatique d'un manque général de connaissances vis-à-vis des outils informatiques. Les interfaces et outils internes sont donc développés en prenant en compte les utilisateurs.

Un des services d'Eminence est particulièrement affecté aujourd'hui par ce problème : Le service Négoce. Chargés de répartir les commandes de produits aux fournisseurs externes aux quatre coins du monde, les membres de ce service utilisent quotidiennement une interface de type console pour parvenir à leurs fins. Les limitations liées à l'affichage se font sentir de plus en plus au fil du temps, et le manque de fonctionnalités offertes les forcent à recourir à des méthodes peu conventionnelles pour transmettre les informations aux fournisseurs. Le service a donc fait la demande auprès de l'informatique afin de disposer d'un meilleur système.

Ce mémoire a pour vocation d'accompagner Eminence dans sa démarche de développement. Il fait état de plusieurs couches d'analyse, du système d'information en passant par le processus métier. Une dimension toute particulière est accordée à l'interface et l'expérience utilisateur, qui sont aujourd'hui l'amélioration recherchée par le service.

La difficulté principale rencontrée est due au fait que la solution doit faire appel aux mêmes ressources que les outils actuels ; Il n'est pas question de créer une nouvelle base de données, et il est impératif de garder la même cohérence de données.

Ce mémoire permet donc à l'entreprise Eminence de mettre en place une politique de développement adaptée aux besoins du service. Entreprenant une démarche de modernisation du service Négoce en utilisant les nouvelles technologies, ce mémoire relate du développement qui constitue :

L'évolution d'une application B2B de l'AS400 vers Angular.

Ce processus implique plusieurs analyses au préalable : Des problèmes actuels aux solutions que peuvent apporter les nouvelles technologies. Le choix d'un développement interne ou de l'appel à un prestataire externe est également une décision importante, impactant sur le long terme le fonctionnement et le support d'une telle solution.

Dans un premier temps, une analyse poussée sera effectuée du service actuel. Après une compréhension exhaustive des services concernés, un état des lieux sera réalisé, comprenant les outils dont ils disposent et les difficultés que ceux-ci posent au sein du processus métier.

Par la suite, l'étude du cahier des charges et du besoin utilisateur permettra de concevoir un portrait-robot de l'interface nécessaire à l'amélioration de la solution. L'établissement d'une politique de développement permettra à tout moment d'être certain de respecter les exigences du service et de leur proposer une solution qui réponde convenablement aux besoins actuels.

Contexte

Eminence

EMINENCE est une entreprise de fabrication et création de sous-vêtements, créée en 1944. Leader sur le marché français, elle propose d'année en année des produits modernes de grande qualité, soucieuse de la satisfaction des clients.

Elle possède plusieurs sites : celui d'Aimargues, et celui de SAUVE, dans les Cévennes. Ce dernier réalise toutes les opérations sauf la teinture, qui sera sous-traitée au sud de la France ou au nord de l'Espagne. La marque principale est Eminence, mais on pourra également retrouver la marque Athena pour les sous-vêtements féminins, et finalement Liabel, en italie.

Néanmoins, EMINENCE ne produit pas seulement des slips pour homme. En effet, elle a décliné son secteur d'activité pour pouvoir maintenant produire des boxers, des caleçons, des chaussettes, des T-shirts, des débardeurs, des pyjamas et des sous-vêtements chauds. Cette diversité de production et création, couplée à la minutie continuelle et appliquée de l'entreprise, lui permet d'assurer sa place de marque numéro 1 en France sur ce marché.

L'intégralité des produits d'EMINENCE sont conceptualisés et inventés à Aimargues. On pourra retrouver ses produits en grande surface ou sur son site Internet. Elle propose également des lots d'articles pour chaque produit conçu à prix réduit, ce qui permet aux clients d'obtenir leurs produits à un coût plus favorable.

Historique

En 1937, Georges Jonathan, représentant de commerce à Paris, et Gilbert Sivel, technicien du textile dans les Cévennes, s'associent pour créer l'Atelier Artisanal de Bonneterie de Nîmes. Cependant, la Seconde Guerre mondiale, ayant lieu deux ans après, empêche la société de fonctionner.

Ne s'avouant pas vaincus, les deux créateurs de l'entreprise décident de continuer la production dès 1944, et par la même occasion créant le nom de marque Eminence, en honneur au surnom du cardinal de Richelieu.

En 1946, les deux créateurs apportent de la nouveauté au milieu du sous-vêtement. En effet, ils parviennent à acquérir des métiers à tisser suisses « point tamisé », à ce jour inconnus en France. Grâce à ces outils, ils parviennent à confectionner un tissu à mailles plus aérées et donc plus confortable qu'ils nommeront le « petit point noué ».

En 1947, l'un des deux fondateurs de l'entreprise effectue un voyage en Argentine. Lors de ce voyage, il découvre que les « gauchos », peuple de gardiens de troupeaux, portent un slip fendu. S'inspirant de leurs vêtements, la société met au point un nouveau slip à poche, le modèle 100 en « petit point noué ». Par la suite, vers la fin des années 1950, le modèle 108 en côtes fines fera son apparition. Grâce à ce dernier, la société EMINENCE devient une société spécialiste en matière de sous-vêtement masculin.

La même année, l'entreprise tente une nouvelle technique commerciale couronnée de succès, la vente de ses produits sous plastique transparent. Elle est la première à utiliser ce procédé.

Aujourd'hui elle est l'une des dernières entreprises fabriquant des sous-vêtements en grande série. Ces quelques dernières années ont été difficiles pour l'entreprise, mais depuis 2016, sa situation s'est améliorée. S'investissant dans les marché publics, tels que des vêtements ignifugés et antimoustiques pour la police ou encore des chemises tactiques pour l'armée, et ayant recours à la fabrication en sous-traitance pour des marques telles que Celio, Monoprix, Carrefour ou le Slip Français, elle connaît une remontée phénoménale, prête d'atteindre son record précédent de 130 millions d'euros sur un marché en recul.

Aujourd'hui

Le site principal d'EMINENCE se situe aujourd'hui à Aimargues, en Camargue. Il est constitué d'une usine, du siège de l'entreprise ainsi que d'un magasin d'usine, dont les produits viennent directement de ladite usine. Les produits y sont proposés à des prix 30% inférieurs comparés au circuit classique. Ces produits peuvent être des invendus, des fins de collections ou des écoulements de surstocks des différentes marques de l'entreprise.

L'entreprise y conçoit ici la plupart de ses produits, qui sont redistribués en grande surface ou à des particuliers. Le siège social de l'entreprise est donc directement lié à l'usine, et l'on pourra constater de nombreux échanges entre celui-ci et les employés de cette dernière.

A noter que plusieurs des départements du siège social sont en relation directe avec l'usine, bien qu'ils ne travaillent pas au même endroit. Ainsi, un service nommé l'ordonnancement est chargé de vérifier que l'usine dispose, à tout moment des ressources nécessaires à la fabrication des produits.

Si l'entreprise conçoit les produits qu'elle vend, elle n'en réalise en réalité qu'une petite partie. Le reste des produits est réalisé à l'étranger, notamment en Asie (Chine, Cambodge, Viêt-Nam, Bangladesh), en Europe de l'Est, en Roumanie ou encore au Maghreb. Cette démarche que le site usinenouvelle.fr appelle une « *délocalisation synonyme de maintien de la production* » a été une des facteurs ayant permis au groupe Eminence de rester sur le marché.

Ordonnancement et Service Négoce

Au sein de l'entreprise EMINENCE existent plusieurs services, comme dans toute entreprise. Ressources humaines, accueil, service informatique, et autres services que l'on peut retrouver partout ailleurs.

Mais il existe également des services propres à EMINENCE, qui se rapprochent de son cœur de métier. De plus, étant donné la proximité immédiate de l'usine, connectée au siège social de l'entreprise, il existe plusieurs services dont la fonction principale est directement liée à l'usine : production, fabrication, et cætera. En particulier, il existe un service lié à la production, dont la fonction est de vérifier si, à tout moment, l'entreprise possède assez de produits pour fabriquer tel ou tel lot ou colis : l'ordonnancement (ou planification).

Son activité est donc dépendante d'une multitude de facteurs, les commandes effectuées sur internet étant diverses et imprévisibles. A noter que certains produits sont plus prisés lors de certaines saisons, comme par exemple des maillots de bains ou des sous-vêtements plus légers lors de l'été, qui laisseront la place aux vêtements plus chauds en hiver. Ainsi, ce service est constamment à l'écoute de l'équipe chargée de la production, et réalise de manière continue des commandes de produits afin de ne pas en manquer si ceux-ci sont nécessaires à la production de lots ou colis récemment commandés.

> *Le service de l'ordonnancement, contrairement à ce que l'on pourrait penser, ne se situe pas au niveau de l'usine. Des échanges et des vérifications sont faites avec l'équipe qui travaille à l'usine, mais les commandes, réunions et autres actions du service sont réalisées au sein de l'entreprise, ce qui facilite leur transmission au service suivant.*

Après passage d'une commande par le service de l'ordonnancement, celle-ci est acheminée dans les bureaux du premier étage. Ceux-ci sont le lieu de deux services aux fonctions similaires : Le service Sous-traitance et le service Négoce.

Le service Sous-traitance a pour fonction de relayer une partie des commandes à des entreprises distantes, souvent présentes dans d'autres pays. Ces dernières sont souvent tirées des commandes internet, qui sont alors produites à l'étranger et ensuite expédiées. On pourra également faire appel à ce service lorsqu'on aura besoin de composants simples nécessaires à la fabrication des produits, comme par exemple du tissu, des élastiques, et cætera.

En revanche, le service Négoce a pour fonction de relayer une partie des produits que produirait normalement EMINENCE à d'autres entreprises réparties dans le monde. Il s'agit surtout ici de faire appel à des fournisseurs extérieurs, en leur fournissant des documents spécifiques décrivant les étapes de confection des produits, afin d'obtenir par la suite suffisamment de produits pouvant ensuite être expédiés par éminence sous forme de lots ou colis, à destination de grandes surfaces ou de particuliers ayant commandé sur Internet.

Workflow

Une fois que la commande est passée par l'ordonnancement, elle prend le statut de « **nouvelle ».** Elle est, après réception, examinée par les membres du service Négoce. L'ordonnancement spécifiant le fournisseur désigné, un employé du service Négoce est en général assigné à un fournisseur spécifique. Il faut bien sûr noter qu'en cas de problème il est possible aux autres employés de prendre en charge les opérations concernant un fournisseur, et également qu'un employé est à même de gérer plusieurs fournisseurs externes (le nombre de fournisseurs dépassant les 3500, il est rigoureusement impossible d'avoir un employé assigné à chacun d'entre eux).

Après décisions internes propres au Négoce, regardant les prix, dates d'expéditions demandées et de réception, la commande est « **validée »** par le service Négoce.

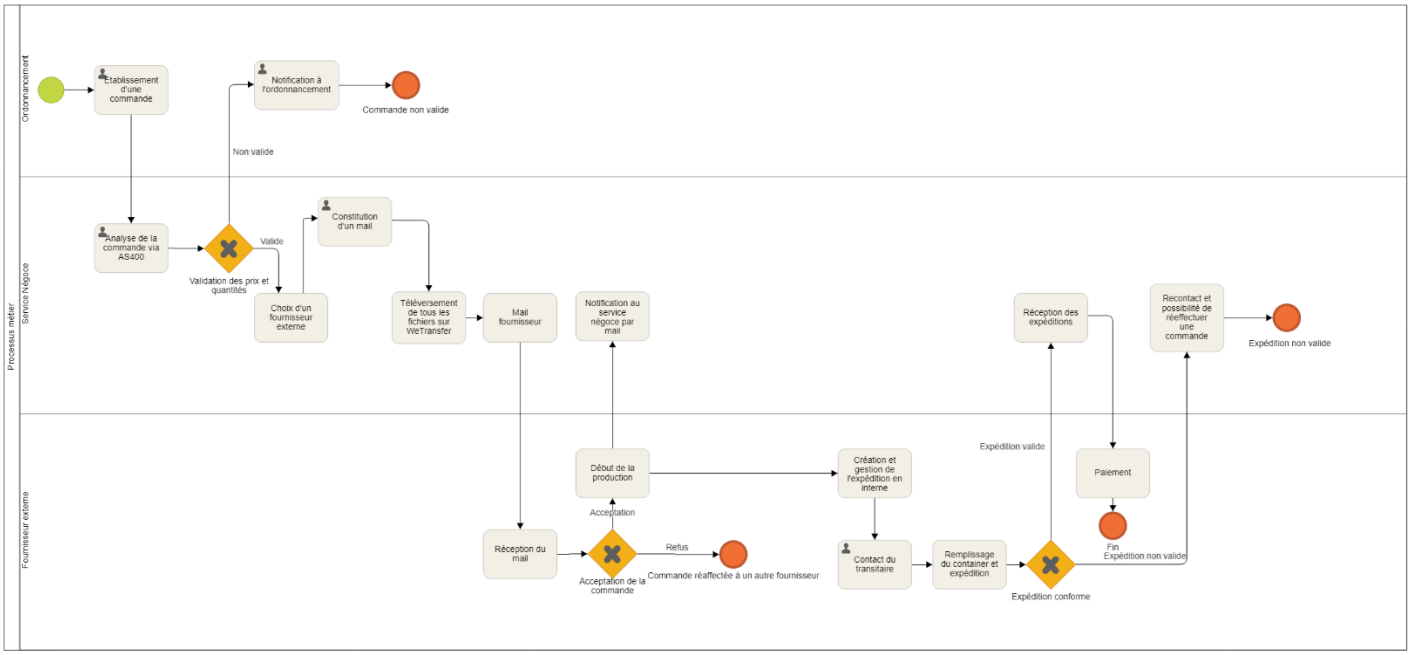
Elle est à ce moment-là renseignée par un document technique, qui regroupe l'ensemble des informations nécessaires à la confection des produits, d'une charte qui établit des règles quant à la production, l'envoi, ou toutes autres opérations regardant ce domaine, ainsi que la commande d'achat contenant l'ensemble des produits commandés, leur quantité et leur prix. Peu après, ces documents sont envoyés par e-mail au responsable du fournisseur.

Le fournisseur doit alors accuser réception de la commande par mail, l'examiner, et ensuite indiquer au Négoce qu'il s'apprête à commencer la production, ce qui fait passer le statut de la commande à « **acceptée »**, sous-entendu par le fournisseur. Le fournisseur, par la suite, envoie les produits confectionnés, qui seront reçus sur le port de l'usine, où un traitement de qualité sera effectué sur une petite partie des produits.

Suite à la réception des produits, et en assumant qu'ils passent avec succès le contrôle qualité (dont s'occupait récemment mon collègue et tuteur Vincent Descreux, au niveau informatique), ils sont enregistrés en base et comptabilisés, jusqu'au moment où la commande est considérée remplie par EMINENCE. Ceci donne le statut final de « **soldée** » à la commande, qui est ensuite payée au fournisseur.

> *Il est important de noter que tous ces échanges se font par mail, et le statut de la commande est changé dans l'AS400 manuellement. Aucun mécanisme ne permet au service Négoce ou au fournisseur de changer le statut via une application.*

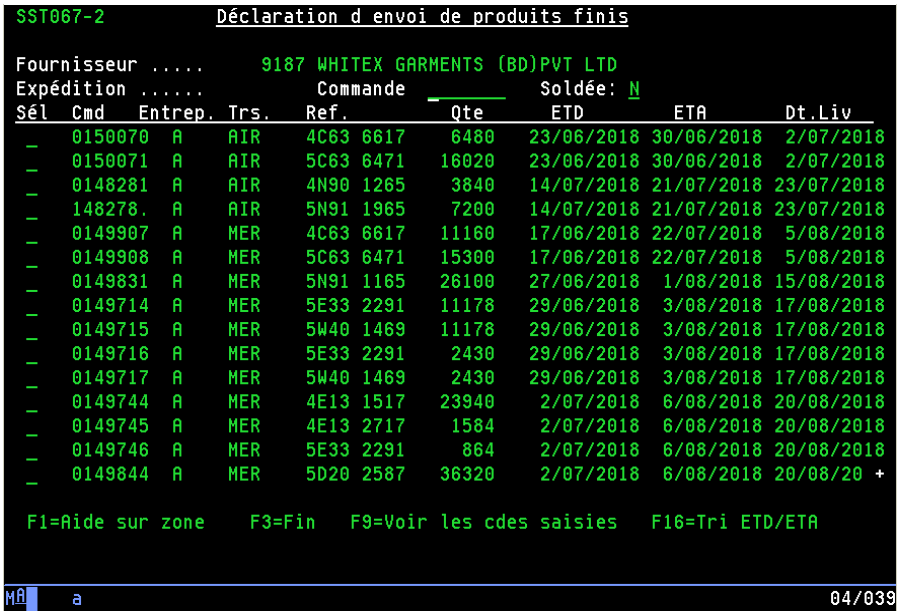
Voici la modélisation du processus métier :



Outils du service Négoce

Les deux services, Négoce et Sous-traitance, étant similaires, on pourrait penser qu'ils utilisent tous deux les mêmes ressources. Cependant, le service de sous-traitance a fait, il y a quelques années, une demande au service informatique concernant une refonte du portail de sous-traitance.

En effet, avant la création de ce portail (par mon collègue Sébastien Rousselle), les deux services se reposaient sur un écran AS400.



Ecran typique de l'AS400, pour le service négoce.

Il s'agit d'une sorte de console, certes différente du DOS que l'on peut retrouver sur Windows ou du terminal sur Mac, mais dont le fonctionnement s'en approche.

Des zones de fonctionnement sont définies lors de la programmation des pages, et l'utilisateur utilise les touches fléchées **<↑>** **<↓>** **<←>** **<→>** afin de se déplacer sur les champs éditables, qui sont repérables par les suites de tirets « underscore » **<_>**. Il est également possible de se déplacer sur lesdits champs éditables via la touche **<Tab>**. Le curseur se positionne alors automatiquement sur la zone suivante, ou précédente si la touche **<Shift>** est pressé en même temps.

Le service sous-traitance a donc fait appel au service informatique afin qu'il puisse disposer d'une interface web simplifiée permettant de réaliser les opérations de manière plus simple et avec plus de visibilité. Sébastien a donc mis au point un outil qui s'est avéré être plus efficace qui a permis au service de pouvoir opérer de meilleure manière.

Le service Négoce, a donc décidé de faire une demande similaire auprès du service informatique : disposer d'une interface web plus intuitive.

> *Il faudra cependant souligner que la nature et les tâches assignées des deux services étant différentes, il ne suffisait pas de trivialement adapter l'interface réalisée par Sébastien afin qu'elle fonctionne pour le service Négoce. Le type des produits, les opérations à réaliser sur la commande et le fait que les fournisseurs doivent pouvoir accéder à une partie du nouveau portail sont d'autant de facteurs induisant la nécessité de création d'un tout nouveau portail, exclusivement destiné au service Négoce ainsi qu'aux fournisseurs qui y sont rattachés.*

Si la demande initiale n'a pas évolué, les réunions qu'ont eu lieu au sein du service négoce, auxquelles nous avons été conviés Vincent et moi, ont ajouté des petits détails sur l'interface ou sur le fonctionnement de tel ou tel composant web. Je ne l'ai pas réalisé tout de suite, mais j'ai pu ici comprendre l'importance du retour utilisateur. En effet, la manière dont nous regardons une interface en tant que développeur est fondamentalement différente de celle de l'utilisateur « classique ».

De plus, étant ouvert aux technologies et ayant habitude de manipuler de tels outils, nous faisons parfois usage de raccourcis mentaux ou avons connaissance de certaines fonctionnalités que nous jugeons basiques et connues de tous. En réalité un certain pourcentage des employés n'est absolument pas familier avec les outils informatiques d'aujourd'hui, et encore moins avec les fonctionnalités dernier cri que s'emploient à déployer les entreprises de services technologiques tournant autour du Web, comme Google par exemple.

Le cahier des charges est donc un point clé de ce projet, permettant de poser des lignes de direction, mais il est important d'analyser en amont les problèmes rencontrés par le service, afin de répondre à un besoin qu'ils ne sont pas toujours capables de formuler.

Analyse de la demande et planification

Cahier des charges

Pour donner suite à la demande initiale, un premier cahier des charges a été réalisé :

- Un onglet « Commandes » dans lequel on pourra retrouver :
  - La liste des commandes passées par l'ordonnancement :
    - Informations de la commande
    - Informations du fournisseur
    - Pouvoir afficher le détail de la commande :
      - Produits avec le prix, libellé, tailles.
      - Quantité des produits et prix total
  - Une action pour chaque commande permettant :
    - D'envoyer les documents nécessaires à la confection du produit (Dossier technique, Commande d'achat)
    - D'envoyer un mail à des responsables que l'on pourra sélectionner, devant en même temps contenir les fichiers mais également les stocker à disposition des fournisseurs sur leur partie du portail
- Un onglet « Expéditions » permettant de :
  - Voir, pour chaque employé du négoce, les expéditions effectuées par les fournisseurs dont il est en charge, consulter leur statut et la quantité reçue.
  - Afficher le détail pour chaque expédition
- Deux onglets accessibles par les fournisseurs :
  - Un onglet « Commandes » qui, à l'instar de celui pour le Négoce, leur montre les commandes qui lui ont été passées, avec :
    - Visualisation des documents envoyés lors du passage de la commande
    - Menu pour accepter ou non la commande
  - Un onglet « Expéditions » qui permet :
    - De consulter les expéditions déjà réalisées
    - De créer une expédition :
      - Pouvoir sélectionner parmi les commandes envoyées
      - Pour chaque commande sélectionnée, de pouvoir saisir une quantité de produit

Problèmes actuels

Les problèmes liés aux outils actuels peuvent être séparés en deux parties : les problèmes liés à l'interface qui est présentée, ainsi que les problèmes liés à l'AS400 en général.

Interface

L'outil actuel dispose d'une interface rudimentaire : il ne s'agit ni plus ni moins d'une interface console améliorée. La souris peut être utilisée, mais il faudra bien sûr se passer de clic droit car aucun menu contextuel n'est présent. Les différentes opérations sont gérées à la main et selon l'écran : Il faut se positionner devant une ligne, celle sur laquelle on souhaite faire l'opération, et insérer un chiffre correspondant à l'opération désirer. Pour connaître toutes les opérations il faut appuyer sur la touche **F1** pour avoir une liste affichée sur l'écran. Il est donc préférable d'utiliser la touche tabulation dans la plupart des cas pour accéder aux zones éditables. D'ailleurs, les champs éditables n'ont pas de valeur interne : Ils considèrent l'ensemble des « pixels » qui composent la zone comme étant l'intégralité de la donnée. Si les caractères nuls sont ignorés à gauche, ils ne le sont en revanche pas à droite : Il y a donc souvent des problèmes de saisie puisqu'il faut que le dernier digit soit en fin de zone.

> *Par exemple, pour accéder à différents programmes, il existe un menu principal où il faut entrer un numéro de commande, par exemple 2758. Pour revenir à l'écran de connexion, il faut inscrire 99. Mais la zone fait 4 caractères de large : Si bien que si l'on tape 99, on obtient « 99__ » ce qui n'est pas reconnu. Il faut taper donc espace deux fois avant 99 pour inscrire correctement __99, ce qui ramène au menu de connexion. Par ailleurs, inutile d'essayer d'insérer des caractères entre ceux déjà saisis, l'intégralité de l'interface est bloquée en mode « Inser » …*

Par ailleurs, la fenêtre est en réalité une émulation réalisée par le logiciel IBM Client Access. Il est possible de mettre des couleurs, mais seulement deux différentes pour les éléments non textuels et les données. Aucune image ne figure dans les écrans, les champs acceptant uniquement les caractères alpha numériques. Certains vétérans de l'AS400 se donné du mal pour créer le logo Eminence en « ASCII art » sur la page de connexion, mais il s'agit bien là de l'unique prouesse artistique qu'est capable d'offrir cette interface désuète. Ceci est bien dommage, lorsqu'on se réfère à l'adage « A picture is worth a thousand words » (traduit en français par « un bon croquis vaut mieux qu'un long discours ».)

Pour finir, les menus ne sont pas accessibles via un menu principal classique déroulant comme on pourrait le trouver sur n'importe quelle interface web, ou bien sur quelques scripts Windows ou Linux bien ficelés. Pour accéder à un certain menu il faut connaître, pour la plupart par cœur, le numéro de commande qui correspond à ce menu. Au lieu du traditionnel Clic de souris pour dérouler les menus jusqu'à l'option souhaitée, il faut entrer des combinaisons de commandes (2758 puis 1 puis 14 pour un des menus du service Négoce par exemple). Ceci rend la transmission de connaissances compliquée puisqu'il faut retenir par cœur les commandes de menu pour pouvoir accéder à tel ou tel programme.

On pourra donc constater une interface tout bonnement inadaptée à des opérations telles qu'un visionnage de liste, d'édition de valeurs ou de navigation.

AS400

Parallèlement à l'interface, le fonctionnement spécial de l'émulateur AS400 et les technologies utilisées rendent son utilisation, son développement et sa maintenance retors.

Il n'existe pas de notions d'onglets : Il faut rouvrir une autre session avec un autre écran (certains relancent le programme). Un bon point que l'on pourra noter est que l'on peut voir, si l'on dispose des autorisations nécessaires, l'intégralité des utilisateurs/ordinateurs connectés. Ceci permet de voir les opérations en cours, le statut de ces opérations (utile pour M. Jean-Noël Ciscar qui occupe la position d'agent d'exploitation, et qui regarde à certaines heures les programmes sensés sauvegarder les données), et si les utilisateurs sont toujours utilisés (utile toujours pour notre agent d'exploitation qui peut donc terminer les sessions à distance, afin d'économiser des ressources.)

Mais concernant les autorisations, on se retrouve là aussi dans une situation délicate : Les menus font références à des « objets » à l'intérieur de l'AS400. Les autorisations sont très complexes à gérer (J'ai pu assister à un changement de permissions effectué en direct par mon collègue M. Phetthavone Douangsavath, ce qui ne m'a semblé ni plus ni moins comme de la magie noire...)

Enfin, la modification ou la création de nouveaux menus est encore un problème : Il faut utiliser un logiciel spécifique nommé Adélia Studio, et connaître le langage de programmation propre au logiciel. Il est également possible de s'attaquer au RPG, langage proche de l'assembleur qui est encore moins facile à prendre en main. Lorsqu'on veut modifier un programme, il sera nécessaire de recompiler le programme en question, ainsi que tous les autres programmes qui y font référence ou qui l'utilisent. Cette manœuvre est chronophage et pousse à régler les problèmes au dernier moment en cas d'urgence. Les technologies utilisées impliquent donc de trouver des employés avec des formations très spécifiques dans ce domaine.

Indépendamment de l'interface, l'outil actuel est peu adapté à l'utilisation qui en est faite aujourd'hui : La communication avec les différents se fait par mail uniquement, donc pas de traçabilité des échanges si les mails ne sont pas constamment envoyés en copie à M. Frédéric Beaumer ou à M. Stéphane Tur. Par ailleurs, les fichiers sont en général assez volumineux puisqu'il s'agit de dossiers complets sur certains produits, ce qui dépasse la limite autorisée de taille des pièces jointe qui a été mise en place par le service informatique. Les membres du service négoce doivent donc saisir un par un les documents et les envoyer via WeTransfer, site de transmission de fichier en ligne. Le site a l'avantage de proposer un téléversement de fichiers allant jusque 2Gb pour un utilisateur gratuit, ce qui est bien au-dessus de la taille moyenne des fichiers envoyés, mais ne permet pas un envoi multiple. De plus, les fichiers ne sont conservés qu'une semaine sur les serveurs distants, ce qui veut dire que si le fournisseur n'ouvre pas en temps et en heure la ressource proposée par le lien envoyé par mail, le service Négoce devra réeffectuer l'opération une seconde fois.

D'autre part, M. Frédéric Beaumer a formulé une demande stipulant qu'il voulait pouvoir recevoir les mails en copie, avec les pièces jointes. Ceci n'est pas possible actuellement car si l'on inclue les pièces jointes dans le mail destiné au fournisseur, celui-ci est bloqué. Il faudrait donc faire deux mails distincts, ce qui double encore la charge de travail pour un détail minime.

On pourra noter un manque d'organisation générale au niveau des menus : les ressources sont éparpillées et peu faciles d'accès. Pour effectuer certaines opérations il est nécessaire de changer de menu. Ceci entraîne une perte de visibilité générale et oblige les utilisateurs à faire des allers retours entre les programmes pour traiter une seule commande.

Après ce premier travail d'analyse, il convient maintenant de choisir une solution qui pourrait répondre aux problèmes de la solution actuelle. Il ne faut cependant pas perdre de vue les contraintes inhérentes au fonctionnement du service Négoce.

Choix et contraintes

Lorsqu'on regarde le fonctionnement du service Négoce, il devient clair que la solution proposée doit remplir certains critères :

- **Intégrabilité :** L'application doit pouvoir s'intégrer facilement au sein de l'entreprise Eminence, respecter sa politique. On doit pouvoir s'en servir comme les autres outils de l'entreprise, sans que cela ne soit dépaysant.

- **Ouverture d'accès :** La solution doit permettre un accès facilité au sein du service mais également, à terme, aux fournisseurs externes. Le but est de mettre fin à l'usage de WeTransfer pour envoyer les fichiers, mais également d'adapter une nouvelle politique : Faire venir les fournisseurs sur le portail au lieu de converser sur un terrain neutre.

- **Intuitivité:** L'interface doit prendre en compte le profil des utilisateurs venant à s'en servir. Beaucoup de personnes de l'entreprise sont ne sont que très peu voire pas habituées aux nouvelles technologies. Il conviendra alors de réaliser une solution facile d'utilisation qui réponde aux problèmes soulevés mais qui ne perde pas de vue les conséquences de tels changements.

- **Evolutivité :** La solution doit s'adapter aux différents changements proposés ou demandés par les différents services concernés. Le cahier des charges doit pouvoir être modifié sans que l'interface doive être reprise à zéro. Les nouvelles fonctionnalités doivent pouvoir s'intégrer facilement et rapidement dans la solution (sans prendre en compte bien sûr les temps de développement)

- **Dynamisme :** L'application doit respecter le workflow (ou flux de travail) des outils actuels, mais en reflétant l'état des ressources à l'instant T. Elle doit montrer à l'utilisateur le travail en cours, rester claire et concise, et indiquer précisément quelles opérations sont à disposition de l'utilisateur.

Il convient donc de s'intéresser aux technologies utilisées par Eminence pour savoir quel type de solution choisir, comment l'implémenter, et comment assurer le respect des critères précédemment mentionnés.

Base de données

EMINENCE fonctionne avec la base de données IBM DB2. Ceci est dû au fait que l'entreprise utilise des serveurs AS400 pour stocker l'intégralité des fichiers. La base de données est le lieu d'échanges constants, et des rotations de scripts sont organisées afin de garantir son bon fonctionnement. Certains scripts s'exécutent à intervalles définis, d'autres sont exécutés à partir d'une certaine heure. Pour ces derniers, il faut en général qu'une personne veille à leur bon déroulement, en plus d'effectuer d'autres opérations. Ce rôle incombe à une personne spécifique au sein du service informatique, mais les autres membres du service informatique peuvent l'assumer en cas d'absence ou d'indisponibilité.

Portail

Plusieurs portails, dont certains existent pour le développement, se situent sur des AS400 différents. Une partie du portail est réalisée en PHP procédural, tandis que l'autre tourne sur le Framework Symfony, d'ailleurs récemment mis à jour. Ces deux portails disposent d'un menu vertical sur le côté gauche donnant accès à plusieurs ressources en fonction du rôle de l'utilisateur connecté. Si cet accès est mis en place de manière stricte est régulée via un système de rôles sur l'Active Directory sur le portail Symfony, il est en revanche modifié à la main sur le portail PHP classique.

Gestionnaire de sources

En 2017 lors de mon arrivée, service informatique se servait encore du logiciel Tortoise SVN. Après les modifications, il suffisait de faire un clic droit dans le dossier principal et envoyer une notification de changement sur ce logiciel. Ceci avait le mérite d'être simple mais ne permettait pas réellement d'assurer une bonne sauvegarde du code, et ne permettait pas d'avoir une bonne vue d'ensemble sur les différentes sauvegardes. Il n'était en effet pas possible de changer le nom des sauvegardes en question, ce qui donnait lieu à des listes de noms plus tordus les uns et les autres, rendant difficile la décision de restaurer telle ou telle sauvegarde en cas de problème.

Mais récemment, la décision a été prise de passer sous le très célèbre système de versioning décentralisé Git. Ce changement a été accueilli avec joie pour la plupart. Pour ma part, ayant eu l'opportunité de découvrir ce logiciel de gestion de version au cours de mes années de formation, la mise à niveau n'a été que bénéfique pour la construction de la solution.

En parallèle, un serveur GitBucket a été mis en place, constituant une origine commune pour les projets. Le répertoire de base est le projet Symfony, sur lequel chacun des développeurs effectue ses modifications. Chacun possède sa propre branche, synchronisée en local et sur le serveur, sur lesquelles les autres développeurs peuvent « pull », c'est-à-dire récupérer les modifications effectuées et sauvegardées sur le serveur distant. Une branche plus générale nommée « develop » permet de regrouper de temps à autre les modifications effectuées par les différents développeurs du service informatique. Chacun étant sur une partie différente du portail, il n'y eu jusqu'à ce jour aucun problème lors de la fusion. Cependant, les fichiers inclus dans le portail tels que les scripts javascript, les feuilles de style css ou encore les assets utilisés par Symfony restent communs à l'ensemble du portail. Il incombe donc de rester vigilant quant à leur modification (qui n'a généralement pas lieu).

Cette politique vise à réduire les conflits entre les différentes versions des fichiers, modifiés simultanément par plusieurs collaborateurs. Une opération de ce genre impliquait généralement qu'une erreur de fusion entre les fichiers puisse poser des problèmes, demandant parfois des heures à consacrer à la résolution. De plus, la politique de BYOD (Bring Your Own Device) mise en place dans beaucoup d'entreprises, qui consiste à faire travailler chacun des membres sur ses appareils personnels, peut également poser des problèmes de compatibilité, entre notamment des postes sous Windows et Mac OS. Des erreurs sur les fichiers tendent à se multiplier et peuvent grandement nuire à la productivité générale, les problèmes ne venant pas de la capacité à mener le projet mais aux différences entre les multiples environnements de travail. Par ailleurs, travailler sur ses machines personnelles soulève également des questions quant à la sécurité du travail en train d'être réalisé, les applications et ressources étant accessibles de l'extérieur.

La politique d'Eminence vise à ne pas reproduire ce schéma, et a opté pour une architecture strictement locale. Bien qu'il soit possible d'accéder aux différents sites marchands d'Eminence depuis l'extérieur, il est cependant rigoureusement impossible d'accéder aux serveurs locaux de l'entreprise par ce biais. Les serveurs ont des IP locales de classe A (10.0.0.0 à 10.255.255.255) et ne sont accessibles que depuis les postes de l'entreprise via Ethernet (prise RJ45 connectée généralement au poste téléphonique).

Analyse des solutions possibles

Plusieurs solutions, au vu de l'analyse, pouvaient voir le jour :

- Adapter l'interface de l'AS400 directement pour le Web
    - Outils comme aXes, iSeries Web Services tools ou encore Silverdev
    - Reprogrammer les écrans en utilisant le COBOL ou le RPG pour le web
- Créer une ressource web sur le portail interne Eminence
    - Sur le portail classique, en PHP procédural
    - Sur le portail Symfony, en créant un module pour l'application

Simplement reprendre l'interface de l'AS400 et l'adapter pour le web semble de prime abord être la solution la moins fastidieuse : Moins de développement, mise au goût du jour, et l'application résultant de cette opération serait semblable aux outils actuels du service Négoce, tout en ajoutant des fonctionnalités nouvelles permettant une meilleure utilisation de la solution.

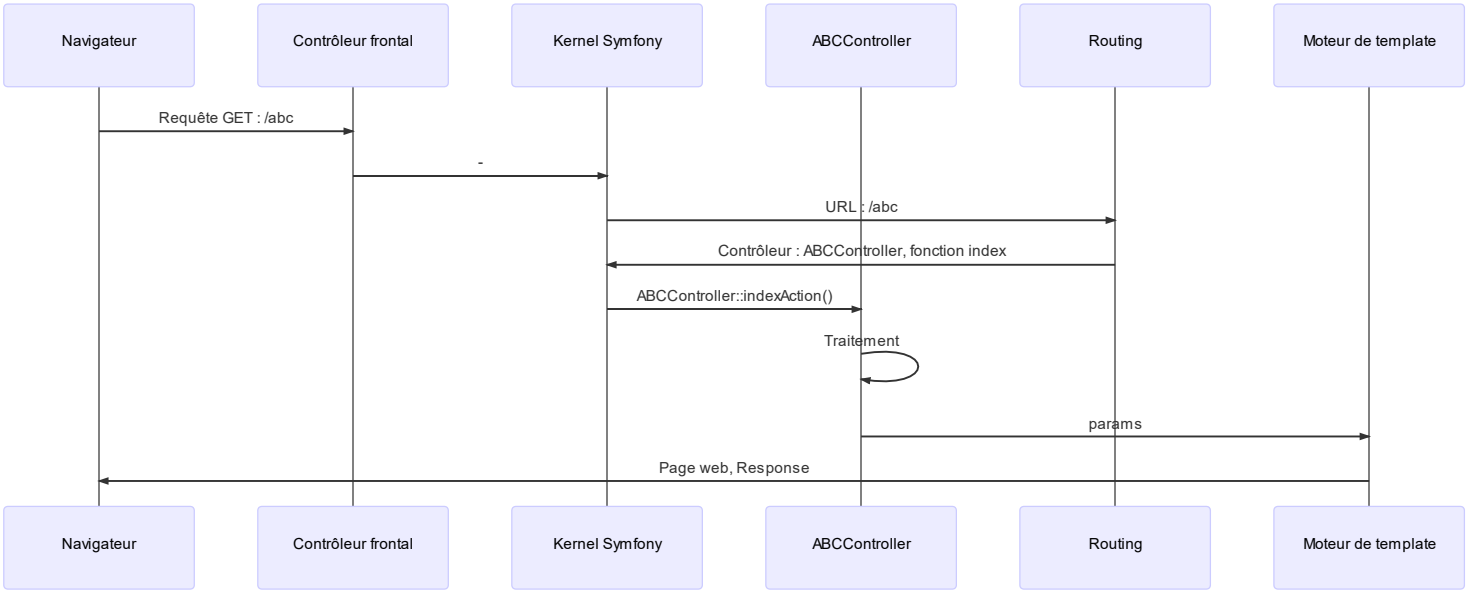
Mais si l'idée est séduisante, elle implique de faire des choix douteux : Tout d'abord il faut confier l'architecture à un outil externe. Si la politique d'Eminence a pour point clé de ne pas donner l'accès au public, pourquoi alors risquer une exposition ? Ce choix n'est donc pas le plus logique. Et si l'entreprise entreprend de reprogrammer l'intégralité des outils, il faudrait impérativement réaliser la solution en COBOL ou RPG, qui sont des langages vieux et avec peu de débouchés. Il faut garder à l'esprit que la solution doit rester évolutive, dynamique et ouverte au changement. Choisir des technologies désuètes semble donc une mauvaise décision pour respecter les critères de réussite de l'application.

Reste alors la création d'une interface web, sur l'un des portails web internes d'Eminence. Si le PHP procédural est pratique dans le cas de petits développements légers, il est en revanche peu adapté dans le cas de modifications successives pour donner suite aux changements progressifs du cahier des charges. De plus, ne pas passer par des librairies ou environnements déjà conçus et préinstallés (dits « frameworks ») impliquerait de devoir développer la structure de l'environnement avant de commencer le développement de la solution. Réinventer la roue reste chronophage peu importe le projet, et une opération de ce genre nécessite des arguments précédant sa réalisation. Or, il n'y a aucun intérêt à redévelopper un environnement PHP de toute pièce, surtout si un framework est déjà mis en place au sein de l'entreprise (ce qui est le cas).

Il y a donc tout intérêt à développer un nouveau module sur le portail Symfony. Il suffit de créer un nouveau module avec la console de Symfony, définir les rôles qui y auront accès, et ainsi créer une nouvelle entrée sur le portail.

A noter que cet automatisme est réalisé avec le bundle FOSUserBundle, dont l'installation et le développement ont été réalisés par Vincent et Sébastien.

Symfony fonctionne sur le modèle MVC (Modèle Vue Contrôleur), avec la vue pointant sur un fichier de routing, qui selon la ressource peut appeler un contrôleur faisant un traitement PHP et/ou envoyant les informations à un moteur de template, qui se chargera d'interpréter la vue comme un « texte à trou », complétant les emplacements définis par les variables issues des données du contrôleur.



A présent il convient donc d'étudier les différents moyens utilisés aujourd'hui pour construire une interface web.

Etablir un site web sur le net a en général plusieurs buts : Etablir une présence et une image digitale, donner des informations sur le métier, apporter une visibilité de l'entreprise vers l'extérieur, générer du revenu dans le cas d'un site e-commerce...

Mais bien entendu tout le monde ne dispose pas des compétences nécessaires à la réalisation d'un site de ce genre. Les entreprises ont donc généralement recours à deux procédés : Soit recruter un développeur capable de réaliser à la main un site moyennant rémunération, soit utiliser un système de gestion de contenu (ou CMS pour Content Management System en anglais). Ces logiciels font abstraction de la couche logicielle pour se concentrer uniquement sur l'interface utilisateur. On pourra citer les plus connus comme WordPress, Drupal pour les sites de type vitrine ou blog, ou encore les CMS davantage orientés e-commerce comme Shopify, Prestashop ou Magento (ce dernier est d'ailleurs utilisé au sein d'Eminence). Dans notre cas cette solution ne nous intéresse que très peu, car les besoins du service Négoce sont bien particuliers et nécessiteraient une modification intense des fonctionnalités prévues par défaut dans ces CMS, qui ont pour vocation de satisfaire le grand public. Ils s'orientent donc vers des options assez généralistes et manquent parfois de souplesse quant à leur adaptation à une solution.

Construire l'interface à la main semble donc être l'option la plus appropriée. Mais sachant que nous allons utiliser Symfony, il faut s'orienter vers un framework dit « Front-end », c'est-à-dire qui s'oriente exclusivement sur l'interface utilisateur.

On cible donc le HTML, le CSS et le JavaScript qui sont interprétés par le navigateur pour donner vie au site internet chez le client.

Trois grands frameworks JavaScript émergent alors, faisant aujourd'hui partie des plus utilisés : ReactJS, VueJS et AngularJS. Il convient donc de les comparer pour connaître leurs différences et lequel est le mieux adapté pour réaliser la solution. Il est important de prendre la bonne décision : En effet, il sera par la suite difficile d'en changer au cours du développement, voire impossible à moins de reprendre le projet depuis la case départ. De plus, un tel choix doit prendre en compte l'évolutivité grandissante de toutes ces technologies ; Il faut donc s'appliquer à choisir un framework qui puisse répondre correctement au cours du temps aux besoins des utilisateurs.

Le site formationjavascript.com fait état des besoins des applications Web aujourd'hui :

- L'encapsulation des fonctionnalités
- La communication avec l'API/le back-end (Ajax)
- Le templating
- Le routing
- Le stockage local
- La gestion des erreurs

Et pour les projets ambitieux, ne pas oublier une architecture solide et des procédures de tests.

Etant sur Symfony, nous disposerons déjà de certaines de ces fonctionnalités : le templating avec le moteur de template Twig, le routing avec le fichier routing.yaml.

ReactJS est une bibliothèque ayant pour but de construire des composants web réutilisables au sein d'une application web monopage. Ce n'est pas un framework MVC mais bien un ensemble d'outils uniquement centré sur la vue. Pour faciliter l'écriture de celle-ci, l'équipe créatrice de React, chez Facebook, a créé un tout nouveau langage nommé JSX (pour JavaScript XML). Il s'agit d'une extension de la syntaxe classique du JavaScript. Très similaire en apparence au HTML, JSX permet de créer des classes ou composants avec une syntaxe équivalente. Au lieu de faire appel à des fonctions de type « constructeur », ce langage permet de rendre compte de la structure du composant avec des balises « < » et « > », les enfants de chaque élément étant compris entre les deux balises de l'élément parent comme en HTML. De plus, les éléments créés au biais de React peuvent être directement utilisés au sein de ce langage, permettant une conception modulaire hiérarchisée.

Par exemple, un composant Modal qui contient 3 divs peut être utilisé au sein d'un autre composant en tant que tel. React décomposera le composant en éléments HTML et fera la concaténation.

```
<Modal>
	<div>Titre</div>
	<div>Message</div>
<Modal>
...
<App>
	<button>Afficher dialogue</button>
	<Modal></Modal>
</App>
```

VueJS est un framework évolutif lui aussi drestiné à construire des interfaces utilisateur. Il a été conçu pour pouvoir accueillir les changements de manière favorable. Il se focalise lui aussi sur la partie vue, pouvant être intégré avec d'autres frameworks ou bibliothèques existants. Il est souvent utilisé dans le cas d'une migration d'un site web vers une application monopage, notamment destinée au mobile.

Vue s'attache à garder le DOM classique intact, en rajoutant des zones spéciales interpolées par le moteur de template, pour y rajouter par la suite les informations nécessaires via un composant Vue.

Voici ci-dessous un exemple pour afficher un Hello World classique.

```
<div id=''app''>
	{{texte}}
</div>

var app = new Vue ({
	el : '#app',
	data : {
		texte : ''Hello World !''
	}
})
```

Le composant vue indique l'ancre, c'est-à-dire le composant auquel il va s'attacher (ici décrit avec le querySelector '#app', sous-entendu « Element dont l'id est "app" »)

Le template peut également devenir très réactif et dynamique via l'utilisation de **directives** : Il ne s'agit ni plus ni moins d'une fonction qui prend en paramètre un élément et qui indique à Vue les opérations à faire sur cet élément. On pourra retrouver des directives comme `v-bind` pour lier les composants à une donnée interne, ou encore `v-if` pour afficher le composant en fonction de la valeur passée à la fonction (afficher le composant si la fonction reçoit `true`, ne pas l'afficher si elle reçoit `false`). Elles sont préfixées de `v` pour une meilleure visibilité.

Dans un composant Vue existe également ce qu'on appelle des propriétés calculées : armées de getters et setters, elles permettent d'avoir une logique plus complexe au travers d'une fonction constamment mise à jour et qui se met à jour dans la vue de manière automatique.

Prenons l'exemple d'une vue voulant afficher le prénom et le nom :

```
var app = new Vue({
	el : '#app',
	data : {
		prenom : 'Sherlock',
		nom : 'Holmes'
	},
	computed : {
		nomComplet : {
			$get: function () {
				return this.prenom + ' ' + this.nom
			}
		}
	}
})

app.nomComplet // 'Sherlock Holmes'
```

Pour ne pas compliquer inutilement l'explication, le setter facultatif n'a pas été inclus.

Etant donné que seul le getter est utilisé il est également possible d'utiliser une simple fonction au lieu d'un objet :

```
nomComplet : function () {
	return this.prenom + ' ' + this.nom
}
```

La propriété calculée agit comme une propriété classique, mais lors qu'on y a accès elle fait appel à la fonction getter et retourne son résultat.

Lorsqu'on change sa valeur, la fonction setter est appelée avec en argument la nouvelle valeur, ce qui va modifier le contenu de la propriété calculée. La nouvelle valeur est ensuite affichée après un nouveau cycle d'appel à la fonction getter.

AngularJS est lui un framework MVC complet conçu et maintenu par Google. A la différence des deux autres technologies mentionnées précédemment, AngularJS contient toutes les directives et composants nécessaires à la création d'une nouvelle application. Via des modules il permet l'inclusion de fonctionnalités diverses et variées comme le routing, les animations, etc. Ces modules ne sont que facultatifs, ce qui permet de choisir uniquement les fonctionnalités d'Angular voulues et pouvoir ainsi développer l'application.

On inclut en général le javascript dans un autre fichier, référencé dans le fichier HTML :

```
<body ng-app="app">
	<div ng-controller="main">
		<p>Hello {{text}}!</p>
</div>
	<script src="main.js"></script>
</body>
```

Le script, quant à lui, se présentera comme ceci :

```
var demoApp = angular.module('app', [])
demoApp.controller('main', ['$scope','$http', function ($scope, $http){
```

```
            $scope.text = 'World';
      }]);
```

La syntaxe est fondamentalement différente des deux autres technologies. On inclue la directive `app` sur un élément haut dans la hiérarchie, ce qui va fixer le script sur cet élément. Les différentes parties de l'application peuvent fonctionner avec des contrôleurs différents, qui sont des fonctions spécifiques gérant l'état des variables internes.

Une variable très importante est le `$scope`. Il s'agit là d'une variable interne disponible dans tous les éléments enfants de l'élement auquel le contrôleur est attaché. On peut y stocker tout type d'objet, variable, fonction. L'objet est ensuite accessible dans la vue, interpolé via les doubles accolades `{{ }}`.

Il est possible de changer la méthode d'interpolation, dans le cas par exemple de conflits avec un éventuel moteur de template. Twig fonctionnant également avec les doubles accolades, il y a deux solutions :

- Soit inclure une balise spéciale conçue par Twig qui lui permet d'ignorer les doubles accolades, `verbatim`
- Soit changer la méthode d'interpolation d'Angular via le module `$interpolateProvider` :

```
      var demoApp = angular.module('app', [])
      demoApp.controller('main', ['$scope','$http', '$interpolateProvider', function
($scope, $http, $interpolateProvider){
            $interpolateProvider.startSymbol('[{') ;
            $interpolateProvider.endSymbol('}]') ;
            $scope.text = 'World';
      }]);
```

Les directives d'Angular sont diverses et variées, permettant le développement d'une interface riche et dynamique. Il est également possible de concevoir ses propres directives, qui fonctionneront de la même manière. Dans AngularJS les directives permettent d'attacher un comportement spécifique à un élément du DOM, ou bien de transformer l'élément en composants HTML. Lors du chargement de la page, le compilateur HTML installé par défaut dans AngularJS parcours l'application et recherche les éléments auxquelles la directive est appliquée, et exécute la fonction interne de celle-ci pour modifier le DOM ou attacher l'événement.

Les directives peuvent s'attacher à 4 types d'éléments selon l'attribut qui est défini : les noms d'éléments (E), les attributs (A), les noms de classes (C) et les commentaires (M). Ceci permet un développement versatile et complet, personnalisable en fonction des besoins si jamais les nombreuses directives proposées par défaut par AngularJS venaient à ne pas suffire (ce qui n'est généralement pas le cas).

Maintenant que les technologies sont présentées, on peut déjà séparer en deux les solutions proposées : React et Vue sont deux librairies qui proposent des outils pour concevoir et développer des interfaces web. Chacun dispose d'un fonctionnement spécifique, Vue étant une sorte de mélange entre React et Angular, mais ces deux outils se focalisent exclusivement sur l'interface. Même s'ils permettent de développer des composants réutilisables, ce qui augmente le taux d'évolutivité de l'application, ils ne permettent pas en revanche de fonctionner seuls.

Même s'ils paraissent simples d'utilisation de prime abord, ils mènent en réalité à une illusion : Il faudra tôt ou tard recourir à d'autres librairies pour pouvoir disposer des fonctionnalités nécessaires. On pourra prendre l'exemple de Redux pour React. Ce qui doit être simple d'utilisation devient très compliqué et fastidieux.

Certains insisteront sur la flexibilité de React et Vue. Il convient de s'étendre sur cette notion, qui regroupe beaucoup trop de définitions vagues. Pour certains, il s'agit de la capacité à un composant ou à un logiciel de pouvoir réaliser des tâches ou concevoir des fonctionnalités personnalisées en fonction des besoins. Pour d'autres, il en est plutôt de l'adaptabilité de ce composant à plusieurs autres solutions. Vue n'en reste pas moins intéressant, mais plus orienté dans une optique de progression sur un développement concernant un site web déjà fonctionnel, ce qui n'est ici pas le cas.

Angular permet de réaliser une application de toutes parts, que ce soit par le biais de toutes ses fonctionnalités, ou via son utilisation en tant que framework front-end. Il n'en reste pas moins flexible et facile à prendre en main. Son utilisation se rapproche de la programmation classique, ce qui permet de ne pas être dépaysé comme on pourrait l'être en programmant avec React ou Vue. Pouvant être adapté sur une infrastructure existante, ce framework est une solution de choix pour notre application.

Il est intéressant de noter qu'Angular possède plusieurs versions. En effet, AngularJS désigne la première version, tandis qu'Angular est un terme représentant toutes les versions qui ont suivi à partir de la version 2. Ces versions sont un redéveloppement complet du framework AngularJS, utilisant notamment le langage TypeScript plutôt que le JavaScript d'AngularJS.

Même s'il est alléchant de se tourner vers les nouvelles versions, la perspective d'apprendre un nouveau langage n'en est pas moins une possible perte de temps, pas ou peu de personnes au sein d'Eminence étant familières à ce langage. AngularJS, fonctionnant sur le classique JavaScript, reste donc la solution retenue.

Mise en place du projet

Il faut maintenant mettre en place le projet. Le cahier des charges incombe d'avoir un développement toujours ouvert aux modifications, sans que celles-ci ne perturbent le fonctionnement actuel.

Pour mettre en place un projet Web il faut en général :

- Définir le besoin utilisateur
- Choisir l'équipe de développement
- Planifier les tâches
- Développer l'application selon un cycle de projet
- Mettre en place des procédures de test/recettage
- Déployer la solution

Dans notre cas, il va falloir légèrement changer cette approche compte tenu de la structure du service informatique et des besoins utilisateurs. En réalité il va falloir adopter une méthode de gestion de projet souple et ouverte au changement. Pour notre solution, il faut :

1. Constituer une politique de structure et d'affichage
2. Planifier les étapes de développement en relation avec les fonctionnalités
3. Procéder à ces étapes de développement
4. Mettre à disposition un environnement de test
5. Déployer la solution
6. Retour client

> *La solution résultant d'un développement de plusieurs années, soutenu et maintenu, il faudra répéter les opérations 2 à 5 à chaque fois que le cahier des charges est modifié.*

La solution sera donc développée selon un cycle itératif, pour les étapes 2 à 5. La première étape, quant à elle, sera effectuée avant le début de la planification et du développement. Il sera judicieux de choisir une approche méticuleuse quant à l'interface de la solution, étant le principal changement à voir venir pour le client.

Une fois cette politique mise en place, il sera intéressant de mettre en place des indicateurs clés de performance spécifiques à notre solution, afin de vérifier qu'elle respecte ladite politique tout au long du développement.

Politique de structure et d'affichage

L'application nécessite deux parties bien distinctes : une partie pour les membres du service Négoce, ainsi qu'une partie pour les fournisseurs externes. Même si les fonctionnalités offertes par ces deux parties doivent différer, il n'en est pas moins important de garder la même identité visuelle sur l'intégralité de l'application. Cette idée est notamment renforcée par le fait deux ressources doivent être partagées et accessibles par les deux entités clients, qui sont les commandes et les expéditions. Le service Négoce doit disposer d'une liste de commandes avec des actions, ainsi que de la liste des expéditions. Les fournisseurs quant à deux, doivent avoir les actions sur la liste des expéditions. Bien sûr, le service Négoce aura la visibilité sur toutes les commandes et expéditions, tandis que les fournisseurs ne pourront interagir qu'avec les commandes et expéditions qui les concernent.

Trop de pages au sein d'une application peuvent ralentir son fonctionnement et entraver les actions de l'utilisateur. Les technologies d'aujourd'hui offrent de nombreuses alternatives au changement de page, et permettent d'afficher aux utilisateurs des interfaces dynamiques et réactives. Dans l'idéal, deux pages pour chaque client type semblent être la solution. Cela implique donc de devoir donner la possibilité à l'utilisateur d'effectuer les différentes actions sur les éléments au sein de la page. Il convient donc de trouver un outil permettant d'afficher des éléments dynamiquement sur la page, tout en gardant la même identité visuelle sur l'ensemble de l'application. De plus, un outil compatible avec Angular semble être la meilleure approche, l'adaptabilité de cet outil permettant l'inclusion de modules au sein de son application. L'inclusion des modules se fait dans cette déclaration :

```
	var demoApp = angular.module('app', [])

	demoApp.controller('main', ['$scope','$http', '$interpolateProvider', function
($scope, $http, $interpolateProvider){

	}]);
```

Le premier `[]` permet d'inclure différents modules qui seront utilisés tout au long de l'application. Le contrôleur, lui, définit les modules qu'il sera en mesure d'utiliser, afin que les variables soient définies sur toute l'étendue du contrôleur. Il est ainsi possible d'inclure des modules dans l'application, mais de ne les utiliser que dans les contrôleurs nécessaires. Cela permet d'économiser des ressources et de clarifier le code.

La réponse : AngularJS Material.

AngularJS Material est une implémentation de la nouvelle politique de Google sur les interfaces nommée Material Design. Jusqu'ici la mode était au Flat Design ou Design Plat, qui était un style d'interface maximisant l'utilisation de formes géométriques simples et de couleurs unies, sans effets particuliers d'ombres, de lumière ou de transparence. La majeure partie des interfaces tentaient de réaliser des interfaces toujours plus compliquées et sophistiquées, perdant de vue le côté sobre nécessaire à une interface digne de ce nom. Un exemple d'une telle politique est l'ancien design des anciens appareils Apple, qui ont vu un design « glossy » ou brillant être maintenu pendant quelques années, avant le déploiement de leur système d'exploitation iOS 7 en septembre 2014.

Cependant, si le Design Plat devenait la mode en manière de style d'interfaces, il n'en demeurait pas moins un simple guide plus qu'une politique de design. Ceci donnait lieu à des idées farfelues, certaines judicieuses mais la plupart perdues dans les méandres des créations des développeurs à l'époque. Un manque de cohésion général entre les différents composants graphiques conçus à l'époque se faisait ressentir, et il était courant de voir des interfaces au style peaufiné alors que leur utilisation et leur visibilité laissait considérablement à désirer.

Material Design, proposé par Google, est selon **Wikipedia** un « ensemble de règles de design », s'appliquant à l'interface graphique dont disposent les applications et par extension les appareils. Annoncé en Juin 2014, il est maintenant présent sur tous les appareils Android qui disposent d'un système d'exploitation de version supérieure à la 5.0. Il a fait l'objet d'une recherche étendue menée par une équipe de taille réduite dans ses débuts, qui a probablement pris de l'ampleur lorsqu'ils ont réalisé que leurs aboutissements allaient donner lieu à une percée dans le milieu du design applicatif. La théorie de base de cette nouvelle idéologie visuelle était de reproduire la sensation et l'apparence du papier au sein d'une page Web, copiant les phénomènes d'épaisseur et d'ombres régis par les lois de la physique. Par ailleurs, la versatilité des composants Web d'aujourd'hui a donné l'idée à cette équipe de modifier contextuellement lesdites propriétés de ces éléments, les adaptant ainsi à tout type de contexte. En résultent des règles de conception précises, relativement faciles à respecter, et qui permettent de créer des interfaces uniformes en peu de temps.

AngularJS Material se veut révolutionnaire en incluant cette politique dans son fonctionnement : alliant ainsi le Material Design aux composants Angular, elle met ainsi à disposition un set complet et exhaustif de multiples composants graphiques, facilement utilisables, personnalisables et stables, pouvant être utilisés dans n'importe quelle application Angular. De composants simples comme des boutons à des composants plus complexes comme des barres de menus, en passant par des indicateurs de progression ou encore un outil de thème intégré, sont immédiatement disponibles dès l'inclusion du module dans l'application. Il suffit d'inclure le script JavaScript au préalable pour pouvoir accès aux différentes ressources qui ne sont que du JavaScript et du CSS.

L'utilisation de cette librairie permet de répondre immédiatement aux questions posées précédemment : Comment donner l'option à l'utilisateur le choix de faire des actions concernant des éléments de la page, sans pour autant rafraichir celle-ci ?

La solution se trouve dans un service imbriqué nommé `$mdDialog`. Ce module permet de créer dynamiquement une fenêtre au sein de la page, qui peut être facilement modulable grâce aux paramètres passés lors de la construction de l'élément.

Il suffit d'inclure la directive `ng-click` sur un élément pour ouvrir le dialogue :

```
<div ng-app="demoApp" ng-controller="EmployeeController">
<div>
<md-button ng-click="ouvrirDialog()">Ouvrir</md-button>
</div>
 </div>
```

Un simple dialogue d'alerte se présente comme l'exemple ci-dessous :

```
function ouvrirDialog() {
      dialog = $mdDialog.alert({
        title: 'Message',
        textContent: 'Exemple de message affiché à l'écran',
        ok: 'Fermer'
      });
      $mdDialog
        .show( alert )
        .finally(function() {
          dialog = undefined;
        });
    }
```

Il est également possible de développer des dialogues beaucoup plus complexes, fonctionnant sur des templates. En soit, il ne s'agira que de pages Web encapsulées au sein du dialogue, dont l'utilisation rend leur accès dynamique, mais faisant elles-mêmes preuve de réactivité grâce aux fonctionnalités définies pour ces pages spécifiques.

Planifier les étapes de développement en relation avec les fonctionnalités

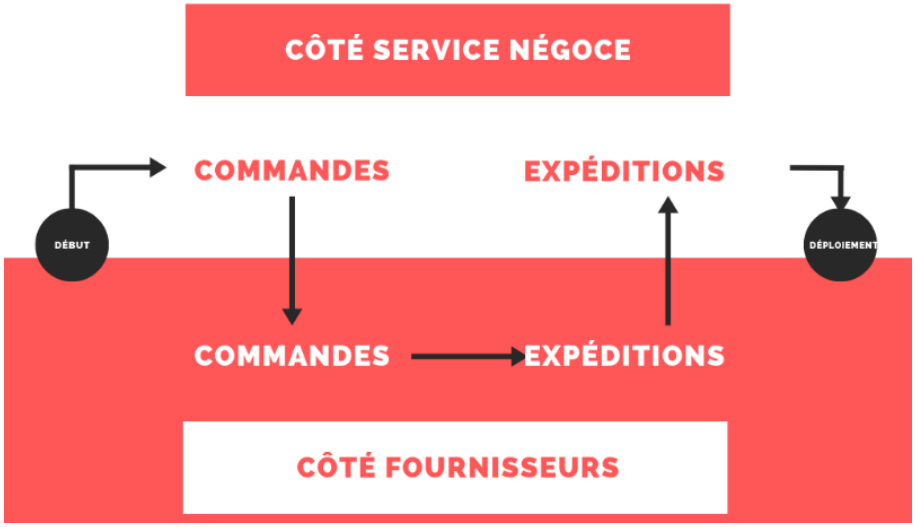
Les ressources partagées par les employés du Négoce et les fournisseurs étant les mêmes, il semble judicieux de construire la ressource la plus complexe et nécessitant le plus de conception et de développement, pour ensuite retirer des fonctionnalités et la restreindre aux options disponibles à l'autre entité client.

Suivant ce schéma, un fil directeur serait tout d'abord constitué par la création de l'interface pour les commandes, côté Négoce. Les employés doivent être capables de parcourir l'intégralité des commandes en cours, d'en sélectionner une, de constituer les produits qui la constituent, et d'effectuer des actions diverses comme consulter les expéditions en cours pour cette commande, joindre des fichiers ou encore envoyer une notification à un fournisseur.

Pour le fournisseur, seuls les documents sont d'une utilité. Il dispose donc de moins de fonctionnalités que le service Négoce.

En revanche, le fournisseur doit être capable de saisir des expéditions sur son interface, ce qui est rigoureusement impossible pour le service Négoce, qui n'ont qu'une vue sur la composition des produits et les quantités envoyées.

On pourra donc dégager le plan de développement suivant :

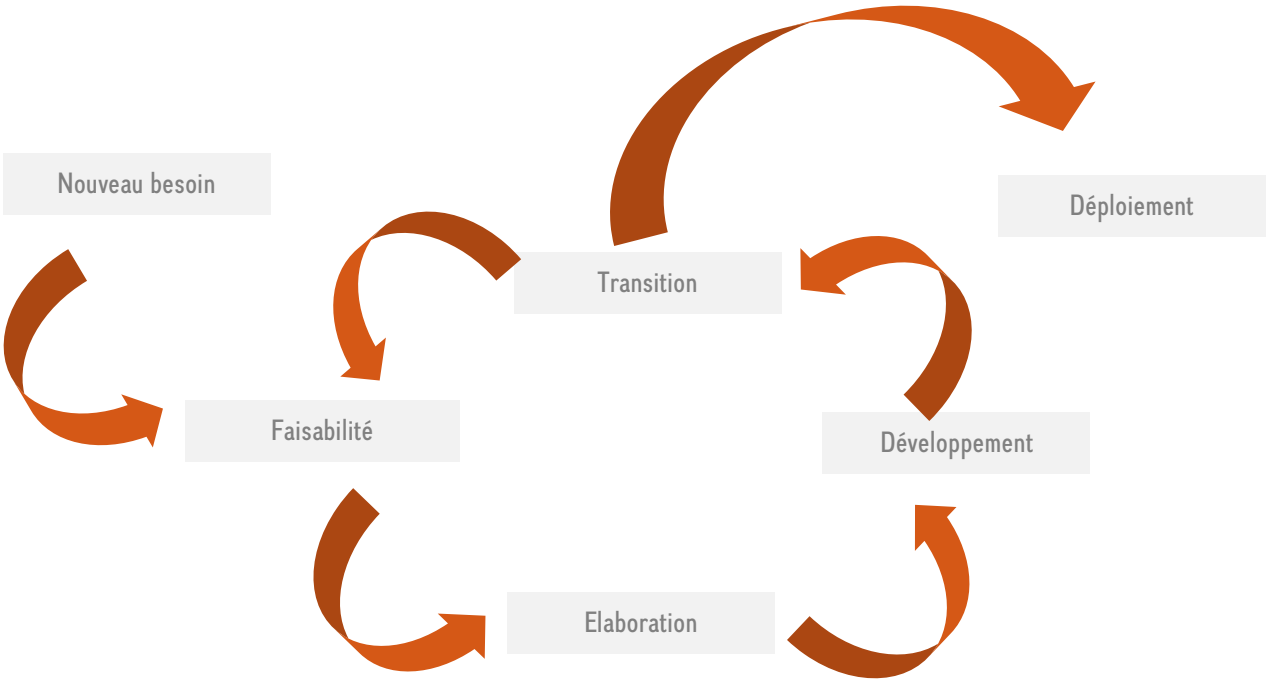


Développement de la solution

Le développement de la solution doit s'effectuer de manière modulaire : le cycle de développement doit laisser lieu à une analyse des besoins, pour ensuite embrayer sur des développements cycliques.

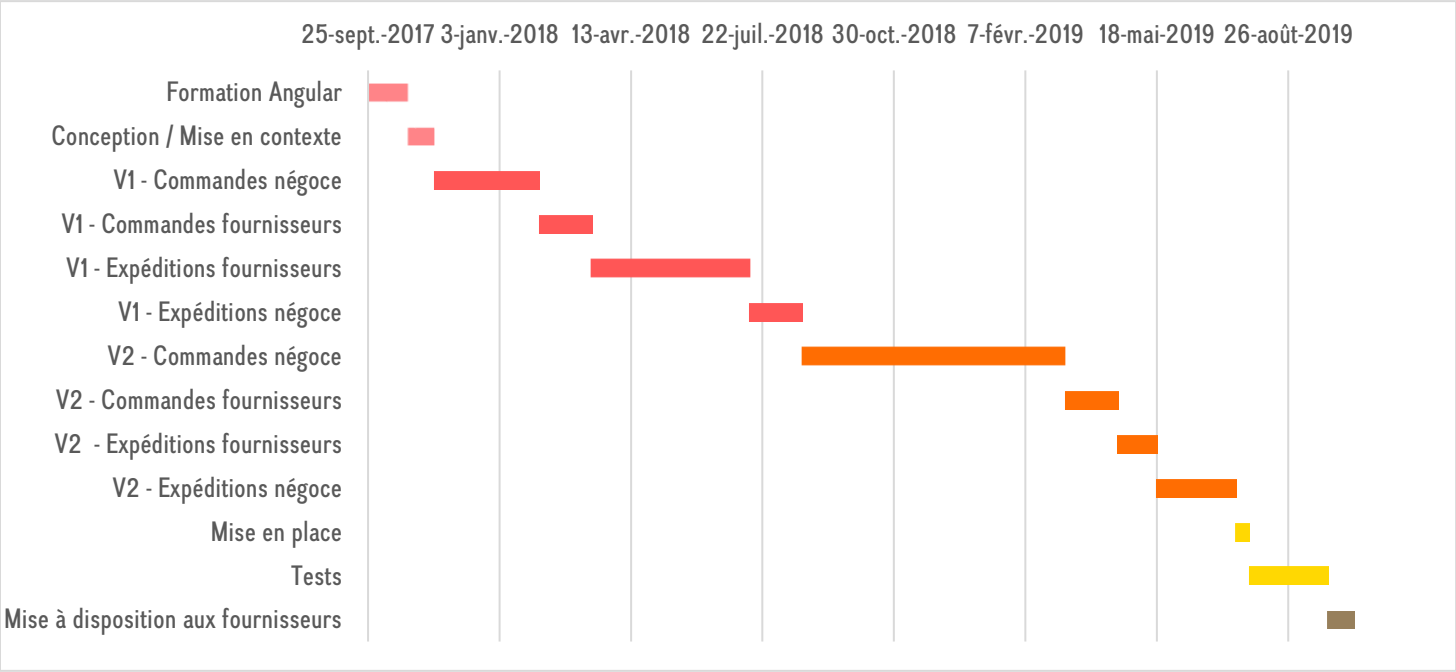
On s'orientera donc vers un cycle itératif.

Pour rappeler son fonctionnement :



Les développements seront réalisés de manière cyclique, avec un reporting fait tous les deux mois lors de réunions conviant le service Négoce pour observer les résultats. Ceci permet d'éviter ce que j'appelle personnellement le « syndrôme de la taupe ». Il s'agit du phénomène constaté lors des développement superflus ou peu cernés. Dans ces situations, on développe de nombreuses fonctionnalités que l'on juge sur le moment indispensables, mais qui ne sont pas nécessaires ou bien n'entrent pas en compte dans les besoins du client. La plupart du temps, ces développements emmènent le projet bien au-delà des limites posées, ce qui implique des développements subséquents pour redresser la barre et obtenir à nouveau une solution correcte. Ces développements sont dus à une mauvaise direction prise de la part de l'équipe de développement, pas corrigée assez vite. On creuse donc en ligne droite, très profond, ayant manqué l'objectif et devant creuser plus que nécessaire pour arriver à un résultat équivalent.

Voici le diagramme Gantt qui décrit les itérations du développement.



Le développement a été itératif de deux manières différentes : La première étant de développer les composants logiciels de manière modulaire et de faire des réunions à intervalles réguliers. La seconde manière a été de développer une seconde version de l'application, après constatation de problèmes irrésolvables lors des tests de la première version.

Il en sera ainsi pour toutes les autres versions de l'application. Le fait de construire l'interface la plus complexe pour ensuite s'en servir de modèle pour l'autre entité client permet d'économiser considérablement des ressources et du temps, comme on peut le voir sur le Gantt. La seconde interface prend considérablement moins de temps. Il est certain que l'interface des commandes négoce de la version 2 ne serait à ce jour pas achevée si le plan de développement en avait été autrement.

Mettre à disposition un environnement de test

Cette partie du développement ne pose que peu de problèmes.

En effet, Eminence possède plusieurs AS400 sur lesquels les ressources sont accessibles. Notamment, il existe un serveur que l'on appelle la recette qui est en soit une copie du serveur principal effectuée à un instant T.

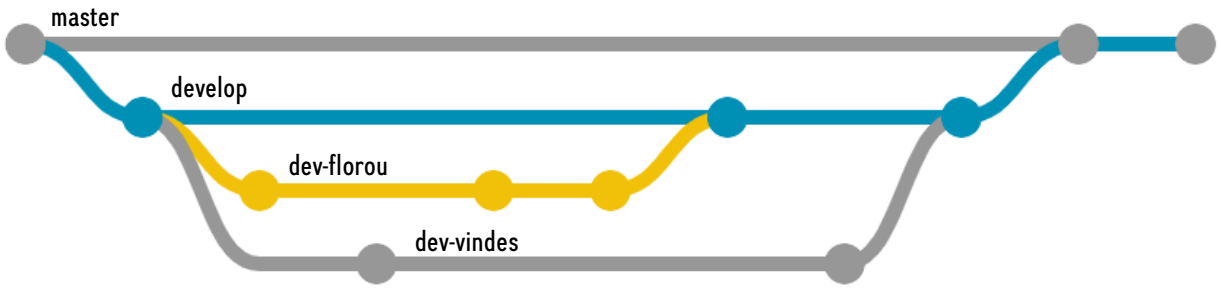
Ce système permet d'avoir un environnement de test réaliste, avec des jeux de données provenant de situations réelles puisque les fournisseurs et commandes ne changent pas. Quelques problèmes peuvent avoir lieu, comme par exemple la création d'un nouveau produit sur le serveur principal qui ne serait pas répercutée sur la recette. J'ai par exemple rencontré

quelques déboires, notamment lors de la synchronisation entre les deux serveurs : la table que j'avais créé avait disparu, rendant la page vide de texte puisqu'il s'agissait d'un dictionnaire. Fort heureusement il ne s'agit que de problèmes temporaires, qui n'affectent pas de manière durable le fonctionnement de l'application.

Le module a donc directement été créé sur le portail Symfony présent sur la recette, permettant un développement dans des conditions saines. En permanence alimenté par des données réelles, l'application a pu être développée en ne perdant pas de vue tous les différents cas pouvant se présenter, comme par exemple les disparités en matière de structure de produits.

Déploiement de la solution

Pour comprendre comment la solution a été déployée, jetons un œil à l'architecture de gestion de version présente au sein du service informatique, sur le serveur GitBucket :



La branche master est la branche de production. C'est sur les fichiers qu'elle contient que tourne le portail Symfony de production. Sur la recette, il existe un dossier Symfony sur lequel les fichiers sont synchronisés par rapport à la branche develop. C'est donc ici que l'on fait des tests pour voir si la version actuelle tient la route avant de la mettre en production. Enfin, chacun dispose de son propre dossier correspondant à sa branche (dev-florou pour moi). On peut donc travailler sur toutes les parties du portail sans être entravé par les modifications des autres membres du service informatique.

La solution est d'abord développée sur la branche correspondant au développeur. Ensuite, elle est poussée sur develop pour faire des tests d'intégration, avec les modifications apportées par tous les autres développeurs. Enfin, si les résultats sont concluants, elle est déployée sur la branche master. Pour mettre à jour le serveur de production, il suffit simplement de tirer les modifications de la branche master. Ceci ne prend gère plus de 30 secondes.

Une fois la solution déployée, une documentation sera rédigée pour expliquer son fonctionnement. (Documentation fournisseur disponible en Annexe.)

Choix des développeurs

Nous avons défini ce qu'il faut développer et comment il faut s'y prendre, il reste cependant la question du qui : A qui va-t-on confier ce développement ?

L'entreprise a alors deux choix possibles : le développement effectué par un prestaire externe, ou bien un développement interne au sein du service informatique. Pour effectuer cette décision il convient d'étudier l'impact du recours au service d'un prestaire, ainsi que de la faisabilité de cette manœuvre.

Avantages d'un prestataire externe

Disposer de l'expertise d'un prestataire externe permettrait sans nul doute un développement rapide. Adepte des nouvelles technologies, une équipe de développeurs confirmés est capable de se mettre à jour sur le fonctionnement de l'entreprise et de venir à bout du projet. Gestion de projet, structure et reporting sont des éléments clés du développement que sont sans nul doute capable d'offrir ces développeurs.

Limites au niveau développement

Cependant, leur expertise est parfois synonyme d'habitude : Des développeurs trop habitués aux nouvelles pratiques et fonctionnements ne seraient peut-être pas à l'aise avec le fonctionnement d'Eminence, dont les racines technologiques sont présentes depuis plusieurs dizaines d'années. D'un autre côté, les développeurs ayant des compétences dans les frameworks paléontologiques qui font fonctionner l'AS400 risquent de mettre l'accent sur cette partie de leurs compétences, et ainsi proposer leurs services à des coûts beaucoup plus important.

Environnement de travail

Faire appel à un prestataire externe implique évidemment de devoir mettre à disposition les outils nécessaires au développement à celui-ci. Si le prestataire est capable de choisir des outils dont il a l'habitude d'utiliser, il est revanche forcé de devoir utiliser les ressources qu'offre l'entreprise, c'est-à-dire les accès web et les bases de données.

Or, comme énoncé précédemment, les accès sont strictement internes chez Eminence. De plus, il ne s'agit pas de créer un nouvel environnement de travail mais bien d'intégrer l'outil dans la structure déjà mise en place par le service informatique. L'entreprise serait alors forcée d'ouvrir ses portes et de donner accès aux ressources via le web, ou bien d'accueillir le prestaire au sein de son entreprise. Les deux solutions présentent des risques importants en matière de sécurité : l'entreprise est souvent visée dans le cas de l'espionnage industriel malgré son taux de sécurité important, ce qui laisse imaginer le désastre si l'on venait à prendre de tels risques.

Durée et continuité du développement

Développer une telle solution implique de constituer la solution sur le long terme. Ainsi, un partenariat avec une entreprise prestaire implique de prolonger le contrat de manière indéterminée. Par ailleurs, il est important de noter que la plupart des informations nécessaires à la création du projet, ainsi que les retours clients, ne sont possibles que via le service Négoce. Un prestataire externe présentant la solution devrait alors interagir avec un autre service interne de l'entreprise, si ce n'est plusieurs lorsqu'on considère l'implication du service de l'ordonnancement lors du passage des commandes vers le Négoce.

Par ailleurs, la solution développée serait connue principalement du prestataire externe, ce qui serait dangereux i l'on venait à mal comprendre, même en suivant une éventuelle documentation, le fonctionnement de la nouvelle interface. Ceci est certainement à prévoir étant donné les difficultés rencontrées par les utilisateurs actuels des outils négoce, qui ne sont pas toujours habitués aux nouvelles technologies.

Il faudrait donc, en plus du service de développement, faire appel au prestataire en qualité de support exclusif sur la solution. Ceci implique soit le déploiement d'un membre de l'entreprise prestaire au sein du service informatique ou du service négoce, ou bien nécessite l'entreprise prestataire d'offrir un service de maintenance et support continu, sachant que la continuité d'un support informatique constitue un autre set de compétences à prévoir dans le recrutement de cette entreprise prestataire.

Le développement de cette solution est donc ici plus adapté dans le cas d'un développement interne. Moins de risques se présent lors du recours à un membre du service informatique. Celui-ci est plus à même d'être accoutumé aux façons de faire de l'entreprise et de sa structure, sans compter le fait qu'étant sur place il est plus aisé pour lui d'obtenir des informations des services concernés par ce nouveau développement ; Il offre également beaucoup plus de disponibilité quant à son service, et la solution intégrée directement dans les serveur de l'entreprise offre également la possibilité aux autres développeurs du service informatique d'assurer le support en cas d'absence. Le fait de choisir un développement interne est également plus d'usage en vue d'un développement toujours cyclique, les versions ultérieures désirées, regorgeant de fonctionnalités juteuses étant déjà mentionnées en réunion, avec recul et humour certes.

Le développement sera donc assuré par moi-même. Alternant dans l'entreprise depuis mon entrée en septembre 2017, j'ai été après la présentation du projet confiant sur mes compétences en matière de développement, estimant pouvoir mener le projet à terme. De plus, les présences chaleureuses mais non moins professionnelles et techniques de Vincent Descreux et Sébastien Rousselle étaient certaines de m'apporter les réponses nécessaires en cas de lenteurs sur le développement. J'avais entièrement confiance en la capacité de Vincent d'encadrer la mission, tant en qualité de développeur confirmé que de tuteur ; Cette confiance ne m'a pas trahi et je la ressens encore aujourd'hui.

Développement de la solution

Outils

Base de données

Eminence utilisant la base de données DB2, un logiciel est installé par défaut sur les machines du service informatique : SQLView.

SQLView est un logiciel basé sur le moteur de base de données présent par défaut dans Windows. Utilisant un système de « connexions » qui permet d'accéder à certaines bases, pourvu qu'on connaisse les identifiants, il offre une interface simple mais concise qui permet autant de faire des requêtes de vérification que de concevoir des scripts fonctionnant au sein des applications.

L'interface dispose d'un champ de texte permettant d'y insérer des requêtes SQL, dont le langage est reconnu et syntaxiquement colorisé. Le résultat est affiché dans la seconde partie qui est un tableau.

Le logiciel offre également d'autres fonctionnalités comme l'analyse des requêtes ou encore la modification des données directement en mode grille (qui n'est malheureusement pas possible ici). Il sera intéressant de noter que le logiciel effectue les requêtes de manière partielle : Les résultats sont limités et affichés à l'écran suivant le nombre de résultats, les suivants étant recalculés lorsqu'on défile avec la souris. Ceci permet un gain de temps considérable, les requêtes de grande envergure n'étant pas entièrement évaluées.

Poste de travail

La solution sera développée sur un ordinateur muni du système d'exploitation de Windows. Il s'agit ici de la norme établie en entreprise, et il serait peu judicieux d'installer un ordinateur Apple ou bien muni d'un système Unix. Les ordinateurs des clients étant également sur Windows, on aura ainsi un meilleur aperçu des performances lors du développement, qui seront alors semblables à celles que le client pourra constater. Il faut d'ailleurs souligner que Symfony rajoute une couche logicielle lors du développement à des mesures de test, qui ralentissent considérablement le fonctionnement de la page. Il faut donc considérer un fonctionnement beaucoup plus rapide sur le serveur de production.

Durant l'intégralité de la mission, le logiciel Visual Studio Code sera utilisé. Il s'agit ici d'une version simple et épurée du logiciel Visual Studio, sans pour autant manquer de fonctionnalités ou de possibilités.

En effet, la modification de cet IDE (ou EDI en français pour environnement de développement intégré) par le biais de réglages et de plug-ins additionnels lui permet de s'adapter au développeur, en lui proposant des fonctionnalités supplémentaires telles que de l'autocorrection sur un langage spécifique, une possibilité de déboguer des éléments, de se connecter sur une base de données distante, et bien d'autres. Le logiciel permet également de s'implanter dans le répertoire et de proposer une inclusion de Git au sein de celui-ci, donnant une meilleure vue d'ensemble et donnant l'occasion d'automatiser et/ou faciliter les processus de mise à jour ou de sauvegarde. Il est plus aisé de voir les modifications effectuées sur l'ensemble de projet, et les opérations de fusion ou « merge », qui sont un véritable calvaire à effectuer par le biais des outils git classique, devient un jeu d'enfant et ne laisse pas la place à des erreurs indépendantes du développement qui pourraient ralentir inutilement le développement de la solution.

L'IDE Atom a été considéré pendant un certain temps au début du développement. Cependant, j'ai pu constater que bien qu'il dispose de possibilités d'adaptation, comme Visual Studio Code, il est en revanche incapable de fonctionner avec trop de plug-ins installés, surtout sur une machine avec peu de ressources comme celle sur laquelle la solution s'est vue développée. Visual Studio code est bien plus stable, et offre une bibliothèque de plug-ins en ligne, ainsi qu'une documentation détaillée, à la différence d'Atom. On aurait pu comprendre un tel manque d'information dans le cas d'une technologie ou d'un outil balbutiant, mais Atom a vu le jour en 2015, ce qui ne justifie pas l'absence de partage de connaissance.

Avant de commencer le développement, il faut s'étendre sur la meilleure manière de satisfaire les besoins du service négoce en matière d'interface. L'enjeu ici est de totalement se détacher de l'interface existante, trouver des moyens clairs, efficaces et parlant d'eux-mêmes pour réaliser les opérations qu'ils effectuent aujourd'hui sur l'interface actuelle. Il est également intéressant de penser au-delà et d'adapter l'interface pour effectuer ces opérations d'une meilleure manière, sans dépayser l'utilisateur.

UX/UI : de l'interface à l'utilisateur

L'interface utilisateur étant le point clé de cette solution, il convient de s'intéresser aux deux notions principales en matière de conception d'interfaces graphiques : l'UX et l'UI.

L'UI ou (interface utilisateur en français) est la zone d'action physique logicielle au travers de laquelle un humain peut interagir avec une machine. Ces interactions ont pour but de contrôler ladite machine ou d'influer sur une partie de ses composants, tant au niveau logiciel qu'au niveau applicatif. Une interface utilisateur, lors de son fonctionnement, dispose de deux phases :

- La **phase d'interaction**, pendant laquelle un utilisateur entre en contact via le biais d'un capteur physique avec l'interface. Ce capteur peut-être un périphérique d'acquisition externe, comme une souris ou une tablette tactile, interne (capacitif ou résistif comme un écran tactile) ou bien un convertisseur de signal, comme un microphone ou un capteur de mouvement.
- La **phase de « feedback »** ou **retour**, qui est la réponse affichée sur l'interface en fonction de l'interaction utilisateur. Elle apparaît sous la forme d'une réponse graphique, comme par exemple un bouton qui change de forme, un lien qui change de couleur ou encore une animation. Ces réponses sont affichées sur un périphérique de restitution comme un moniteur, un écran ou des périphériques audio comme des haut-parleurs.

Tous les appareils d'aujourd'hui proposent ainsi une interface, dont les styles varient en fonction des usages, de la mode, du type d'appareil ou encore de l'utilisateur moyen utilisant l'application.

Peu importe le fonctionnement, l'objet et le but de la conception d'une UI est d'offrir à l'utilisateur un moyen simple, facile d'utilisation et proche du fonctionnement humain d'interagir avec un système informatique. Les interfaces de type console étant davantage utilisées par les développeurs et les administrateurs systèmes, les utilisateurs préfèrent voir s'afficher des formes géométriques, des boutons et des pages qui donnent une certaine tangibilité et un côté concret aux ressources utilisées.

L'UX, ou expérience utilisateur, est d'un autre côté le ressenti physique et psychologique qu'un utilisateur éprouve en manipulant ladite interface. Les deux notions sont donc étroitement liées, mais l'UX tend plus à s'orienter sur le lien entre le fonctionnement de l'interface et la logique métier de l'application. Une interface peut donc être facile d'utilisation mais ne pas constituer une expérience utilisateur agréable.

Par exemple, si l'on veut effectuer plusieurs suppressions, l'interface a beau afficher une icône magnifique d'une poubelle à côté des items à enlever et proposer un dialogue de confirmation, il n'en reste pas moins une mauvaise expérience utilisateur que de devoir effectuer ces suppressions les unes à la suite des autres. Une UX digne de ce nom proposerait un bouton pour sélectionner plusieurs items à supprimer d'un seul coup, rendant l'opération simple et efficace. Dans ce cas-ci l'UI doit aussi être au point, pour pouvoir afficher des cases à côcher à côté des éléments concernés.

L'étude de la conception en matière d'expérience utilisateur est en soi complexe puisque chaque utilisateur est différent et peut réagir à un comportement logiciel de manière totalement différente. Il en vient une étude psychologique poussée et l'établissement de guide ou de points clés permettant de cibler les comportements et réactions génériques des utilisateurs à la suite d'un stimulus graphique. Il s'agit en quelque sorte de la mise en place ergonomique d'une interface graphique.

Une interface graphique est donc le mariage harmonieux d'une UI et d'une UX, chacune des deux conçues l'une avec l'autre. Beaucoup d'auteurs et de néo-gourous s'emploient à définir aujourd'hui les points qui définissent une interface réussie, comme Jesse James Garret ou encore Magnus Revang.

On pourra dégager quelques notions importantes, selon lesquelles une interface se doit d'être :

- **Uniforme** : Le style ainsi que les actions que l'interface offre doivent être cohérents entre eux. Si pour supprimer un article il faut cliquer sur une icône, il ne faut pas utiliser un autre procédé pour afficher la même opération sur un autre item semblable. De plus, le style graphique de l'interface doit indiquer clairement le type d'action pouvant être réalisé.

- **Exhaustive** : L'interface doit proposer à l'utilisateur de manière claire les options qui sont mises à sa disposition.

- **Structurée** : En fonction des informations présentes sur la page, il convient d'avoir une interface présentant de la meilleure manière ce type de contenu. Qu'il s'agisse de la taille des caractères pour un texte ou de la manière d'afficher des items, cette structure définit la politique d'affichage du contenu.

- **Logique** : L'interface doit réagir d'une manière attendue à l'utilisateur. Si un bouton est cliqué, un changement doit apparaître à l'écran si ce n'est sur le bouton lui-même. L'interface doit également disposer de plusieurs états en fonction de ce que l'utilisateur peut effectuer.

- **Epurée** : Une interface disposant de toutes les fonctionnalités du monde n'est pas nécessairement la plus pratique à utiliser. Un design sobre, élégant, et donnant à chaque élément la place qu'il doit occuper en fonction de son importance reste un point crucial lors de la conception d'une interface utilisateur. Paul Morris Fitts, psychologue né en 1912, a établi la loi de Fittz en 1954 stipulant que l'indice de la difficulté d'une tâche est calculé par rapport au temps requis pour aller de la position de départ à la position de destination, tout ceci défini par la distance à la cible et à la taille de celle-ci.

On pourra donc se servir de ces critères pour définir une politique graphique régissant le fonctionnement de l'application.

Si on établit un diagramme de cas d'utilisation, il sera plus facile pour nous de définir les différents éléments graphiques que comportera notre solution.

Il sera évident dans un premier temps de séparer les deux notions de commandes et expéditions.

Affichage des items

L'affichage des items doit suivre l'idée de la console AS400, si ce n'est en l'améliorant. Les éléments doivent être affichés horizontalement, avec des sections verticales pour les attributs. Ceci permet un affichage uniforme de tous les items, sans compter que la flexibilité des pages web permet d'afficher plus d'informations et d'offrir plus de fonctionnalités que la liste console présente aujourd'hui dans l'AS400. Les champs du tableau, qui sera utilisé via une balise `<table>` doivent comporter :

- L'ensemble des informations nécessaires à la bonne compréhension de l'item (comme par exemple le numéro de commande, le fournisseur, les différents types de dates, etc.)
- Un emplacement pour les actions pouvant être effectuée sur cet item (afficher le détail, consulter les documents, etc.)

Le meilleur emplacement pour ce composant sera le milieu de page.

On inclura également un mécanisme pour rechercher les commandes/expéditions ou les produits qui les constituent. Cette barre de recherche sera disponible en haut de la page pour être accessible dès le chargement de la page.

Détail des items

On pourra distinguer deux cas : Le cas où les items doivent voir leur contenu s'afficher à côté ou en dessous, et le cas où cette liste est trop importante et doit être affichée différemment.

On mettra donc en place deux mécanismes de détail :

- Un premier mécanisme d'affiche en dessous des items, accessible via une action rapide sur un champ des items
- Un premier mécanisme affichant un dialogue au milieu de la page, permettant de changer de contexte

Opérations successives

Dans les deux parties de la solution, l'entité client peut inclure des documents ou renseigner des informations afin de les transmettre au service cible.

Ces actions se font généralement par étape, et il convient de penser à un moyen d'afficher ces étapes successivement. L'écran doit alors tour à tour changer de contenu pour afficher l'étape suivante, tout en laissant entrevoir les étapes suivantes ou précédentes.

On adoptera un système d'onglets couplé à un dialogue : Le dialogue permettra l'accès dynamique à ces étapes, tandis que leur succession sera assurée par le changement d'onglet. Leur fonctionnement n'implique pas la nécessité d'apparaître toutes en même temps, ce qui sera pris pour avantage par le système d'onglets. Il sera également pratique de désactiver lesdits onglets, dans le cas où les étapes ne sont pas complétées intégralement, rajoutant une couche de vérification.

La plus-value des technologies choisies se fait déjà ressentir : Angular dispose, avec AngularJS Material, de tous les éléments nécessaires à la conception de ces composants selon cette politique de design.

Réalisations

Base de données

L'application doit pouvoir utiliser exactement les mêmes ressources que l'application actuelle. Le traitement fait sur les données doit donc être uniquement le fruit de requêtes portant sur la base existante, et non la création intégrale d'une toute nouvelle base de données. Durant le développement, il est important de garder à l'esprit que DB2 ne dispose pas nécessairement de toutes les fonctions dernier cri qu'offrent aujourd'hui la plupart des SGBD (ou système de gestion de base de données).

La base de données a été conçue dès le début suivant les limitations de l'AS400 ; A l'époque le système était incapable de gérer des noms de tables de plus de trois lettres. Il a fallu aux développeurs de l'époque beaucoup d'inventivité pour tenter de rendre les noms de tables explicites.

Bien que leurs efforts aient été conséquents, on peut observer des noms de tables comme F£POSUM2, FKPSTRUC, FLVENDM. D'autres relatent légèrement plus de leur contenu, comme CDETET pour les entêtes de commandes ou EXPPRD pour les produits des expéditions. Il ne m'est pas possible de montrer un schéma complet de la base de données, celle-ci étant

constitué de plusieurs centaines de tables. Il a fallu choisir avec précision les tables concernées et les lier pour obtenir les informations nécessaires à afficher tout au long de la solution.

Au sein de l'application, la base peut être interrogée de deux manières différentes :

- Lors du chargement initial de la page : Cette requête est alors définie dans le contrôleur Symfony qui va exécuter la requête et inclure le résultat dans la page via Twig. Les éléments sont ensuite récupérés en javascript pour être passés dans Angular.
- Lors d'un appel AJAX : Cette requête est alors l'objet d'une fonction spécifique, prenant parfois des arguments, appelée via la commande `Routing.generate()`. Symfony résout la route et renvoie un JSON contenant les informations extraites de la base, passées ensuite dans une variable javascript et enregistrée dans la variable globale `$scope`.

Une des fonctions créée par le service informatique permet d'accéder au moteur DB2 depuis n'importe quelle page.

Il n'existe pas de procédure stockée dans la base. Cependant, il aurait été possible d'en créer avec la syntaxe suivante, par exemple pour obtenir le nom d'un fournisseur à partir de son numéro :

```
CREATE PROCEDURE GET_NAME(
        IN NUM_FRN          VARCHAR(6),
        OUT NOM_FRN         VARCHAR(50)
)
        LANGUAGE SQL
BEGIN
SELECT VMNAME INTO NOM_FRN
FROM SIMOBJ.FLVENDM
WHERE VMNO = NUM_FRN;
END
```

Le mot clé `IN` permet de spécifier une variable d'entrée, tandis que `OUT` spécifie la variable de sortie (il faut bien sûr rediriger la sortie du select vers cette variable).

Si beaucoup de requêtes de type LMD (Langage de manipulation de données) ont lieu, il n'y a en revanche pas de requêtes de type LCD (Langage de contrôle des données). Elles ont une structure différente et des fonctions bien séparées de celles précédemment montrées. A ce jour il existe 6 types de fonctions différentes :

- GRANT : donne l'autorisation à un utilisateur pour effectuer une action
- DENY : interdit à un utilisateur l'accès ou le lancement d'une action
- REVOKE : annule une précédente opération de changement d'autorisation
- COMMIT : valide une opération en cours
- ROLLBACK : annule une opération en cours
- LOCK : empêche les modifications

Si je voulais par exemple me donner l'autorisation de sélectionner un résultat dans une table, ceci est la syntaxe correcte :

```
GRANT SELECT
ON SIMOBJ.FLVENDM
TO FLOROU;
```

Ceci n'a que peu d'usage dans Eminence, étant donné que nous nous connectons à SQLView avec un utilisateur disposant de toutes les autorisations. Il n'y a aucun conflit, donc aucune nécessité de recourir à de telles opérations.

En matière de sauvegarde, il n'y a eu que de très rares fois besoin de dupliquer une table. Cette opération est faisable avec une requête de ce genre :

```
SELECT VMNO, VMNAME
FROM SIMOBJ.FLVENDM
INTO DEV001.FLVENDM
WHERE VMNO > 1000
```

A noter que le sélecteur * doit être utilisé pour copier tous les attributs. Ici, seuls les fournisseurs dont numéro est strictement supérieur à mille seront copiés.

En revanche, j'ai dû créer une table agissant comme un dictionnaire anglais-français, en utilisant la requête suivante :

```
CREATE TABLE DEV001.TRDFRN (
       TRFCOD              VARCHAR(50),
       TRFENG              VARCHAR(50),
       TRFFRA              VARCHAR(50),
       TRFPAG              VARCHAR(50)
) ;
```

`TRFCOD` représente le code utilisé dans la page, qui sera remplacé par la traduction. Deux langages sont disponibles pour l'instant, et `TRFPAG` permet de restreindre ces traductions à une seule page.

Enfin, il a été fait usage d'extractions SQL, qui ont permis de traiter un échantillon déjà extrait de la base de données. Une extraction permet de réeffectuer des opérations sur des données résultant d'une requête précédente, grâce au mot-clé `WITH`. Elle s'utilise avec la syntaxe suivante :

```
WITH tableTemp (valeur) as
    (SELECT avg(age)
    FROM Person),
    SELECT nom, age
    FROM Person
    WHERE Person.age > tableTemp.valeur;
```

Structure des pages et scripts

Pour une meilleure visibilité lors du développement, une structure au niveau des fichiers a été adoptée.

On séparera tout d'abord les pages dites « classiques » des dialogues apparaissant de manière dynamique, avec deux dossiers `View` et `Dialog`.

Ensuite, on sépare les pages et dialogues différents en fonction des fournisseurs, via deux dossiers `Negoce` et `Frn`.

Enfin, chaque page se voit attribuer un dossier ; Il y a en général un fichier dans le cas d'une page, tandis qu'il peut exister plusieurs fichiers pour les dialogues sur une même page. Ces différentes pages et ces dialogues se verront appelés via le routing de Symfony.

Pour les fichiers JavaScript, on préfixera le nom par « `ang` » pour signifier qu'il s'agit d'un script angular, suivi d'un underscore, puis de `frn` ou `neg` en fonction de la partie concernée, pour ensuite finir par le nom de page. Le script angular concernant les expéditions fournisseurs s'appelle donc `ang_frnExped.js` .

Enfin, pour les contrôleurs, chaque partie de l'application se voit attribuer un contrôleur qui porte le même nom. Ainsi, `CdeController` désigne le contrôleur pour les commandes côté négoce. Pour faciliter l'usage de l'AJAX au sein de l'application, deux autres contrôleurs ont été mis en place, `JsonController` et `JsonFrnController`. Ces deux contrôleurs ne renvoient jamais vers une vue mais retournent toujours une donnée de type JSON.

Fonctionnement général d'une page

Une page fonctionne généralement de cette manière :

Elle est accédée via la méthode render du contrôleur qui va rechercher le template et l'afficher avec Twig. Les variables passées en paramètres à cette fonction seront interprétées par Twig, ce qui permettra leur apparition dans la page ou le code. Le reste des variables entourées de doubles accolades {{ }} sera lui pris en charge par Angular.

Les scripts angular sont structurés de la sorte :

- Les fonctions d'usage utilisées tout au long de la page
- Les variables courtes, qui constituent des réglages par défaut
- Les fonctions complexes, définissant la logique de l'application

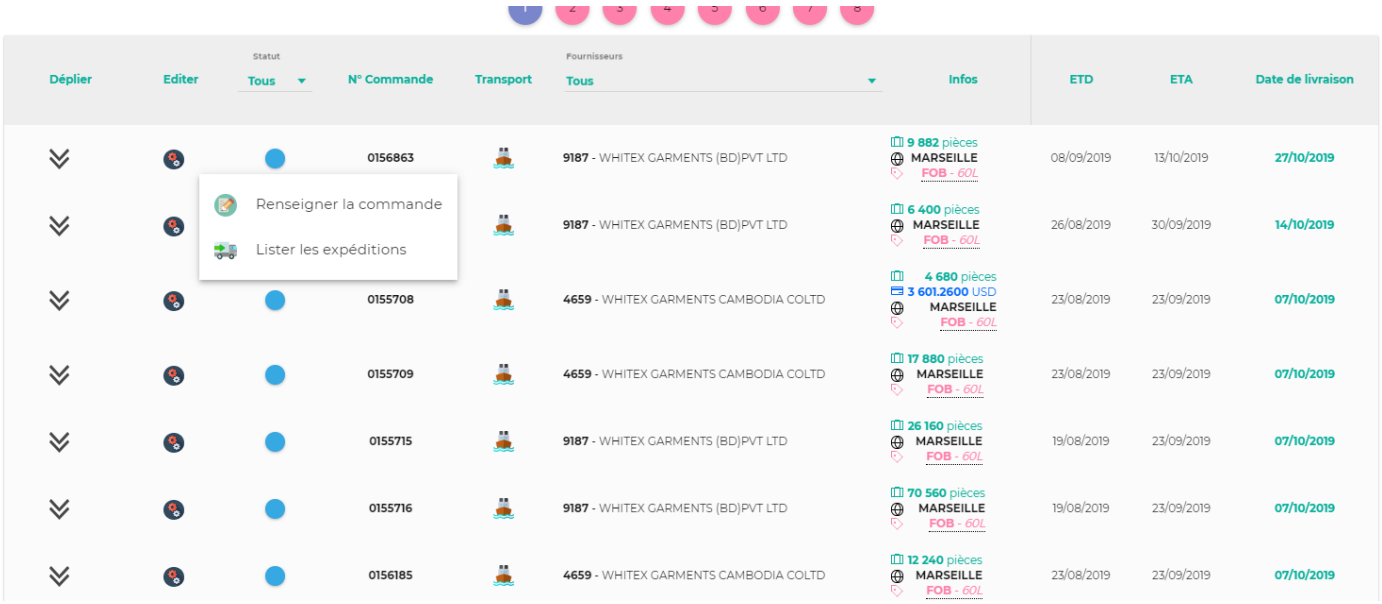
Implémentation de la politique graphique

En premier lieu, les items sont affichés grâce à une balise html de type <table.

On commence par afficher les actions disponibles sur la gauche, puis l'ensemble des attributs qui caractérisent ces items. Pour les items disposant d'un détail accolé, l'expérience utilisateur a été repensée en regroupant les deux types d'actions différentes : On garde le « déploiement » du détail sur un bouton interrupteur simple, tandis que toutes les autres options sont regroupées dans un menu, fonctionnalité possible et ridiculeusement facile à mettre en place grâce au service `$mdMenu` proposé par AngularJS Material. Voici l'exemple d'une telle structure :

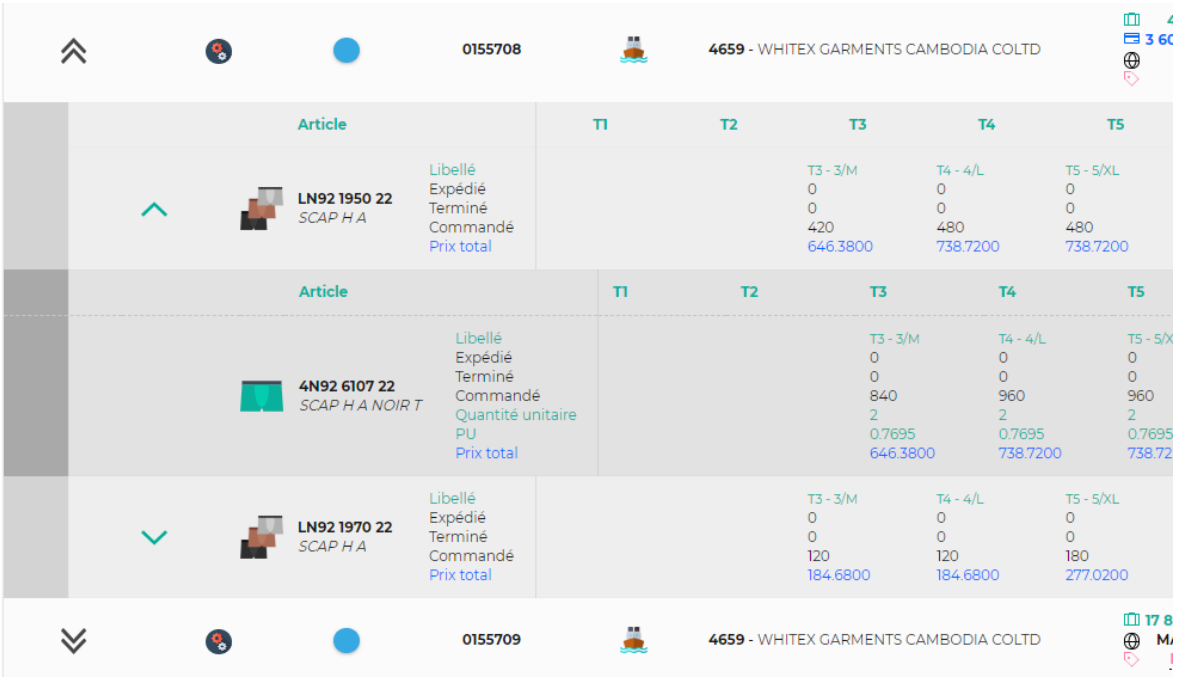
```
<md-menu>
    <md-button aria-label="Actions" class="md-icon-button" ng-click="$mdOpenMenu()">
        <md-icon md-svg-icon="{{ asset('bundles/EminenceQualite/Images/settings.svg') }}"></md-icon>
    </md-button>
    <md-menu-content width="4">
        <md-menu-item>
            <md-button ng-click="showTabDialog(cde)">
                <md-icon md-svg-icon="{{ asset('bundles/EminenceNegoce/Images/edit-pen.svg') }}"></md-icon>
                Renseigner la commande
            </md-button>
        </md-menu-item>
        <md-menu-item>
            <md-button ng-click="showExpeds(cde)">
                <md-icon md-svg-icon="{{ asset('bundles/EminenceNegoce/Images/lorry.svg') }}"></md-icon>
                Lister les expéditions
            </md-button>
        </md-menu-item>
    </md-menu-content>
</md-menu>
```

Les options ne sont en réalité que des appels à différentes fonctions, effectués par la directive angular `ng-click`.



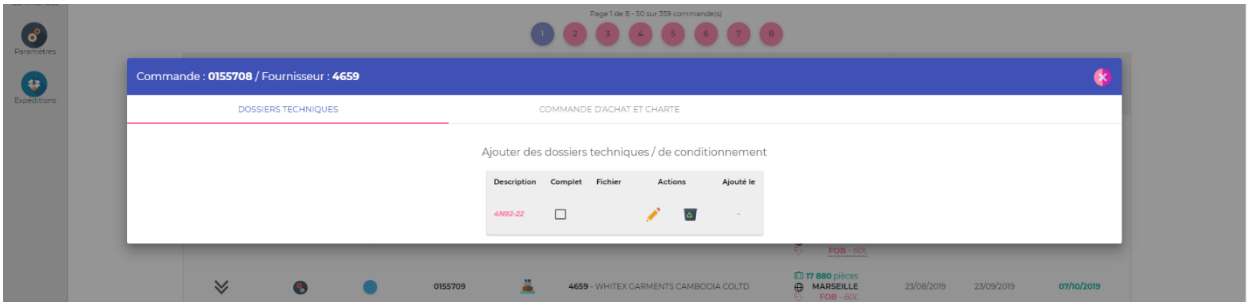
Le détail des items est disponible de deux manières : Soit au sein de la page, soit dans un dialogue. On peut voir l'implémentation du premier type dans la page des commandes côté Négoce.

L'affichage est possible grâce à l'inclusion d'une balise <td> faisant toute la ligne (possible avec l'attribut colspan), dans laquelle est affichée un ensemble de sous-tables, réagissant au type de produit concerné.

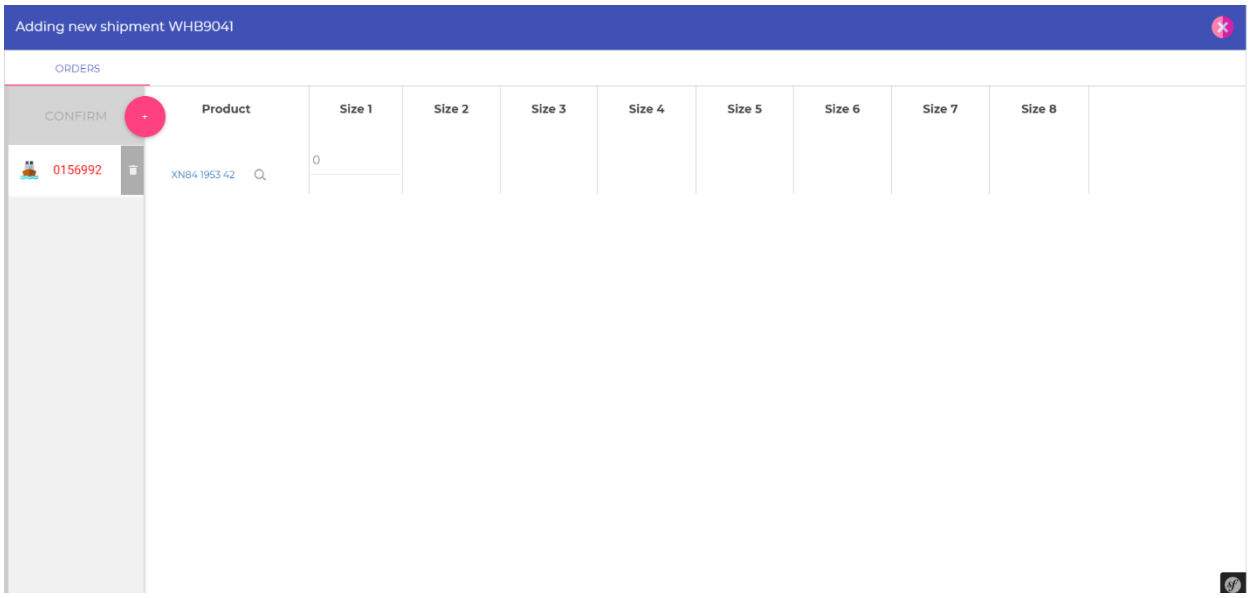


Les actions effectuées de manière successives ont été regroupée dans ce qu'il convient d'appeler un TabDialog. Il s'agit là d'un dialog disposant d'onglets qui permettent de changer son contenu. Il est donc possible de passer d'une étape à une autre d'un simple clic. La beauté d'angular permet de changer cet onglet de manière manuelle ou via une variable. De plus, il suffit d'utiliser la directive ng-disabled affublée de la condition appropriée pour désactiver un tel changement d'étapes, assurant ainsi la validité des données ou opérations sur l'étape en cours.

Un tel exemple est visible sur la page des commandes Négoce : Impossible d'envoyer le mail tant que les fichiers ne sont pas renseignés.



Un dialog personnalisé existe également dans les expéditions côté fournisseur. L'interface permet d'abord de saisir les commandes à remplir, pour ensuite afficher un récapitulatif de confirmation avant de sauvegarder les modifications.



Le dynamisme de tous ces éléments est possible via l'utilisation d'une technologie JavaScript appelée les `Promises`. Il s'agit là d'un objet qui représente, au moment de sa création, la possibilité ou non qu'une opération asynchrone réussisse, ainsi que la valeur de retour que cette opération doit renvoyer. L'objet est donc vide jusqu'à complétion de l'opération, et prend ensuite la valeur de retour dès l'opération complétée.

Ceci permet d'attendre le résultat avant de continuer la suite des opérations : On lance la promise et on effectue le reste des étapes avec le mot-clé `then` qui prend en paramètre la valeur de retour automatiquement renvoyée par la promise. Voici un exemple, au sein des expéditions côté fournisseur qui permet de charger le dictionnaire de manière asynchrone.

```
$scope.dictionary = []

$scope.loadDictionary = function() {
    return new Promise(resolve => {
        $.get(
            Routing.generate('eminence_negoce_frn_json_dictionary', {
                page : $scope.pageName
            })
        ).then(response => {
            resolve(response)
        })
    })
}

$scope.loadDictionaryAsyncCall = async function() {
    $scope.dictionary = await $scope.loadDictionary()
}
```

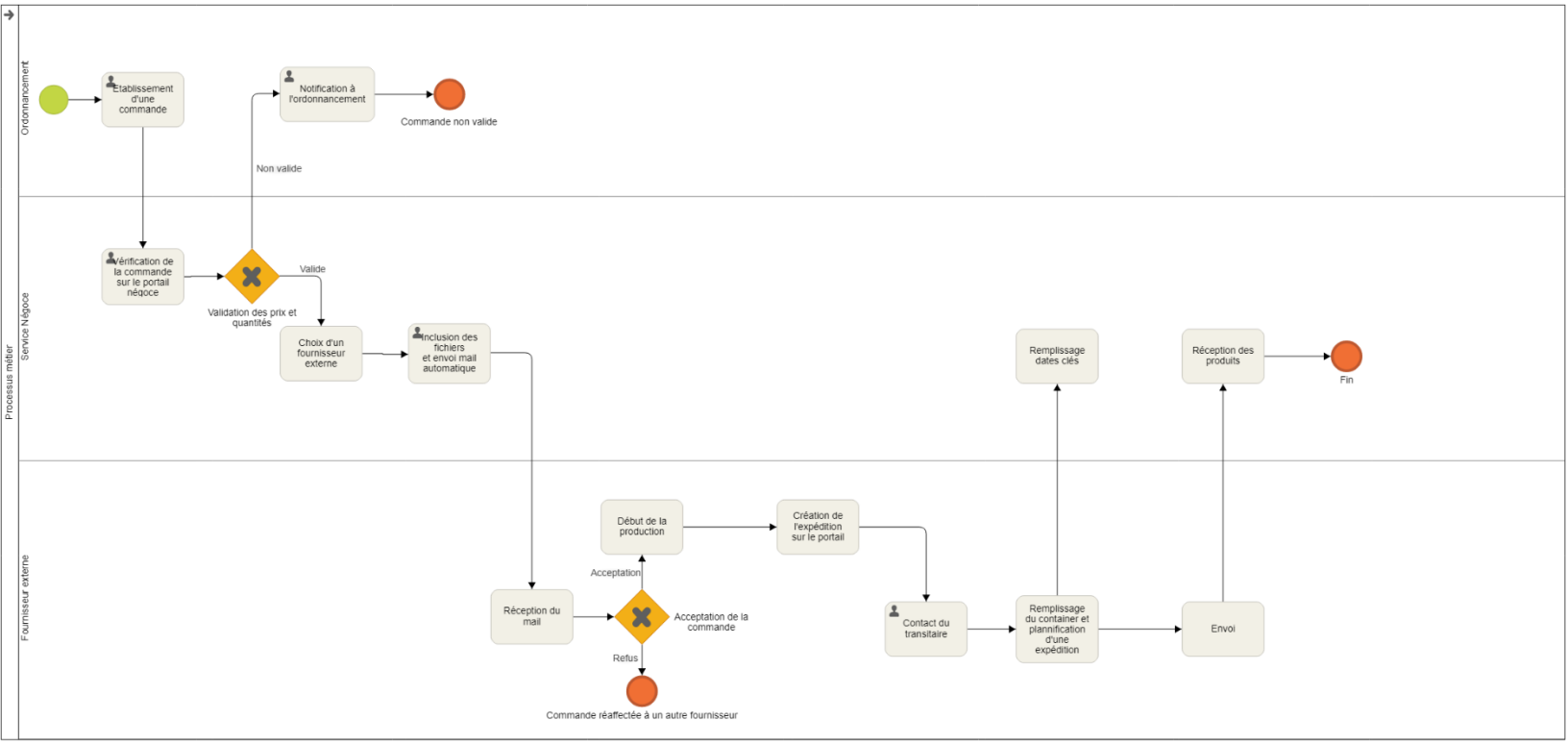
Il est également possible de lancer plusieurs Promises en même temps. C'est le cas de la sauvegarde des expéditions côté fournisseur. On attend le résultat de la sauvegarde des produits de l'expédition dans chaque Promise avec la fonction all, et on fait une seconde insertion dans la table des entêtes si toutes les opérations se sont passées avec succès.

```
Promise.all($scope.expedLines.map(line => sendLineExped(line, $scope.loadingStep)))
    .then(e => {
        $timeout(() => {
            $scope.loadingSave = 100
        }, 0)
        var message =
            $scope.status == 'edit'
                ? 'UPDATED_SHIPMENT_SUCCESS'
                : 'NEW_SHIPMENT_SUCCESS'
        $scope.showSimpleToast($scope.dictionary[message][$scope.currentLocale] + '.')
        $scope.expedSaved = true
        $timeout(() => {
            $scope.errorSave = false
        })
        $.get(
            Routing.generate('eminence_negoce_frn_json_saveDexLine', {
                step   : '1',
                values : JSON.stringify({
                    DEXMTR : $scope.dexmtr,
                    DEXEXP : $scope.expedNum,
                    DEXFOU : $scope.fournisseur
                })
            })
        ).then(response => {
            console.log(
                '%c dex line',
                response == true ? 'color:green' : 'color:red',
                response
            )
            $scope.refreshExpeds()
            $mdDialog.hide()
        })
```

Evaluation des réalisations

Pour constater de l'efficacité, il convient de comparer les processus métiers avant et après mise en place de la solution.

Voici le processus métier après la mise en place de la solution :



On peut voir une nette simplification du processus lors de l'envoi de la commande au fournisseur. Plus besoin de passer par mail et de s'encombrer avec les pièces jointes, tout est fait automatiquement par étape via l'interface. Les pièces jointes sont disponibles sur le serveur d'Eminence après cette opération, et les fournisseurs devront se connecter sur leur partie du portail pour y avoir accès.

Côté expéditions, on donne accès au service Négoce à un aperçu des livraisons : Cela donne une vue d'ensemble sur la complétion de la commande et permet, si besoin, d'accélérer le processus et/ou de changer le moyen de transport d'une partie de la commande (ce qui était voulu avant mais qui n'est possible que maintenant. Le service négoce demandait à l'ordonnancement de créer une nouvelle commande avec un nom modifié rien que pour pouvoir changer le mode de transport).

Enfin, on élimine complètement la possibilité qu'une expédition soit invalide, puisque que sa saisie est régie par l'interface. Le fournisseur sait donc exactement ce qu'il peut envoyer et ne peut pas envoyer plus que nécessaire sauf en cas de marge. Il a également vue sur ses propres expéditions, ce qui lui donne un aperçu de ce qu'il lui reste à envoyer.

Conclusion

L'interface utilisateur est au centre du processus métier, faisant le lien entre l'utilisateur et l'outil qu'il utilise. La recherche constante de meilleures interfaces est un point clé dans l'augmentation des performances et de la satisfaction utilisateur. Se concentrer sur l'apparence ne suffit pas, et il est d'usage de construire une méthodologie d'approche pour comprendre les besoins applicatifs et développer une solution en accord avec les désirs des utilisateurs. De tels processus requièrent des analyses bien précises, que doivent mener toutes les entreprises voulant effectuer ce type d'opération.

La société Eminence se retrouve confrontée aujourd'hui à des difficultés : Son fonctionnement est certes stable mais est enraciné dans le passé, ayant été développé au début des années 1950. Les technologies qu'elle emploie n'ont pas été utilisées dans un but évolutif mais bien pour construire la structure de la société et établir le fonctionnement de son usine ; Aujourd'hui une de dernières entreprises à avoir réussi à subsister dans le milieu hostile du textile, elle doit son succès à l'établissement de normes solides qui régissent sa politique, faisant d'elle une marque sûre.

Mais au fil des années les besoins se sont complexifiés et il est ardu de mettre à jour les outils de l'entreprise. Basant son architecture sur les célèbres machines d'IBM, les IBM System I communément appelées AS400, elle éprouve des difficultés à moderniser les fonctionnalités que celles-ci offrent, au détriment des employés dont les services en requérant l'utilisation intensive. En particulier, l'interface en mode console qu'affiche l'AS400 est peu adaptée aux tâches que doivent effectuer certains employés. Le service Négoce, service important qui relègue une partie de la fabrication des produits à l'étranger, est aujourd'hui gravement affecté par ce problème, suffoquant dans ses tâches quotidiennes qui impliquent entre autres la vérification des commandes et l'envoi de fichiers aux fournisseurs partenaires d'Eminence, aux quatre coins du monde.

Le service souhaite aujourd'hui changer de fonctionnement et disposer ainsi de nouveaux outils pour faciliter ses opérations. La solution envisagée doit permettre de résoudre les problèmes constatés par les employés concernés et offrir, au biais des nouvelles technologies, des fonctionnalités assistant les utilisateurs dans les tâches devant être effectuées. Un tel développement, s'il est réussi, implique également qu'une possible modification pourrait être réalisée sur d'autres services de l'entreprise, mettant ainsi en marche une révolution informatique. Gain de performance, optimisation des tâches et encadrement des processus actuels sont les aspects recherchés dans l'application à développer.

Ce mémoire fait état de la marche à suivre pour développer une telle solution. En commençant par une analyse des besoins, une visualisation du processus métier est indispensable pour comprendre les problèmes rencontrés aujourd'hui et sur quels points la solution applicative doit axer sa résolution de problèmes pour venir à bout de ces difficultés. Tout en examinant scrupuleusement le cahier des charges, directe conséquence de l'expression des besoins client, on pourra en extraire des critères de réussite, qui sont des points importants que la solution doit valider à tout instant lors de son utilisation et son développement. Il faut cependant évaluer les outils de l'entreprise et les limitations auxquelles elle est contrainte, car ce sont des facteurs importants qui vont influer sur la qualité et la méthode de développement.

Plusieurs solutions et outils se dégagent alors pour réaliser la solution, et il est impératif de comparer ces solutions, leurs avantages et leurs inconvénients, afin de décider de la meilleure alternative à prendre. Le mot-clé principal ici étant l'évolutivité de la solution, il faut bien évidemment s'orienter vers des éléments stables et ouverts aux changements que pourrait subir le cahier des charges, en directe corrélation avec la transformation progressive des besoins utilisateurs. L'entreprise doit également évaluer les risques liés à l'appel d'un prestataire externe pour réaliser l'application, ceci impactant grandement son fonctionnement et son support sur le long terme. Dépendre d'une autre structure est généralement synonyme d'échec en matière d'évolutivité puisque celle-ci est restreinte par le progrès personnel d'un environnement extérieur. Un développement interne, dans le cas de la structure d'Eminence, est fortement recommandé.

La science et la philosophie derrière l'interface doivent également être considérée, et accorder beaucoup d'importance à l'UX/UI de la solution constitue un aspect fondamental. La théorie des éléments graphiques doit être étudiée en fonction des besoins utilisateurs et un plan de développement doit être réalisé, détaillant de manière exhaustive une manière uniforme, logique et efficace de présenter un type d'information ou d'offrir une fonctionnalité.

Le développement doit être structuré et effectué selon le plan préétabli, en utilisant des méthodes de développement efficaces, logiques, et soucieuses de la bonne maintenance de l'application dans les années à venir. Bien que chacun dispose de ses propres prédispositions en matière de développement, il convient donc de s'appliquer à établir une structure particulière sur laquelle fonctionnera l'application, tant au niveau graphique qu'au niveau interne.

La méthodologie exprimée par ce mémoire peut ensuite être réutilisée pour effectuer d'autres développements similaires sur l'ensemble des services de l'entreprise. Par analyse successive, il est possible de prendre les décisions les plus judicieuses pour chaque étape du développement, de la conception à la réalisation, en passant par l'apparence. La théorie de l'interface y est abordée, étant une notion à laquelle trop peu y accordent une quelconque importance. Etant la zone d'action contrôlant l'application, synonyme de pouvoir octroyé au client sur l'outil, il est impératif qu'il dispose d'une interface claire, logique et dont l'expérience utilisateur en fasse un outil agréable à utiliser. Si les yeux sont le miroir de l'âme, l'interface est une fenêtre grande ouverte sur l'application.

Glossaire

AS400 : Aussi appelé IBM system I, est un ordinateur de la gamme IBM. Créé dans les années 60 par Frank G. Soltis, il est basé sur des nouvelles technologies pour l'époque et suit 5 grands principes :

- Indépendance vis-à-vis de la technologie
- Conception basé objet
- Intégration matérielle
- Intégration logicielle
- Espace adressable unique

DB2 : C'est un des systèmes de gestion de base de données propriétaire d'IBM. Il utilise le langage SQL. Ecrit en C, il est capable de tourner sur toutes les grandes plateformes : mainframes, systèmes UNIX, Windows, Mac/OS et linux.
SQL : signifie Structured Query Language, en français langage de requête structurée. C'est un langage informatique normalisé servant à exploiter des bases de données relationnelles. Il permet de rechercher, ajouter, modifier ou supprimer des données dans les bases de données relationnelles. AngularJS : framework JavaScript libre de construction d'applications mobiles développé par Google. Une page Web conçue avec AngularJS suit le patron MVC et assure ainsi le couplage faible entre la présentation, les données et les composants métier.

Framework : Un framework fait référence à un socle de développement destiné à faciliter la création de composants logiciels. Ils offrent pour la plupart un modèle d'architecture à respecter et sont spécialisé dans un domaine (application web, gestion d'une base de données etc..).

HTML : HyperText Markup Language est un langage de balisage destiné à représente une page web. Il permet de structurer les pages webs, inclure des ressources telles que des images, des liens ou des formulaires de saisi.

MVC : Modèle-Vue-Contrôleur est une architecture logicielle destiné aux interfaces graphiques, très populaire pour les applications web. Le modèle représente la couche qui contient les données. La vue représente la couche qui correspond à la présentation visuelle de l'application. Le contrôleur représente la couche de logique métier, c'est la couche qui est chargée de manipuler les données.

Opensource : ou « Code source ouvert » s'applique aux logiciels dont la licence respecte les contraintes définies par l'Open Source Initiative. Quand on parle d'opensource on parle de la possibilité de redistribuer gratuitement le code source d'un projet.

PHP : **Hypertext Preprocessor**, est un langage impératif orienté objet opensource, utilisé pour produire des pages webs dynamiques via un serveur http, il peut être déployé sur toutes les grandes plateformes.

Symfony : est un framework PHP pour les applications Web, ainsi qu'un set de composants PHP et de librairies réutilisatbles. Logiciel gratuit depuis 2005, il fonctionne sur l'architecture logicielle MVC.

UI : User Interface fait référence à l'analyse de l'interface graphique d'une application. Son but est d'améliorer l'interaction entre l'utilisateur et l'interface homme-machine. L'UI se base sur des règles et méthodes dédiées à l'ergonomie des composants graphiques et textures d'une interface.

UX : User eXperience fait référence à l'analyse de l'expérience de l'utilisateur, cette analyse a pour but d'améliorer le ressenti des utilisateurs lors de la navigation sur l'application. L'objectif est de rendre la navigation fluide et accessible.

Bibliographie/Webographie

Wikipedia

www.geekforgeeks.org

www.eminence.fr

uxplanet.org – What is UI design ? What is UX design ? UI vs UX : What's the difference

www.interaction-design.org

www.fabernovel.com – Vivons-nous dans l'opulence numérique ou le minimalisme technologique ?

sqlpro.developpez.com

contentsquare.com – Qu'est-ce que l'UX et pourquoi est-elle indispensable ?

www.esokia.com – UX / UI : qu'est-ce que ça signifie ?

www.wkf.fr – Sous-traitance et contrat de prestation de service

www.captaincontra.com – Contrat de prestation de services informatique : ce qu'il faut savoir

material.angularjs.org

blog.lesjeudis.com – MOA / MOE : quelles sont les différences ?

ux-fr.com – Principe de design : La loi de Fittz

www.ionos.fr – Interface graphique : pour une ergonomie Web optimale

www.blogdumoderateur.com – 5 outils pour réaliser une interface web ou mobile

academy.visiplus.com – En quoi consiste une bonne UI interface

academy.visiplus.com – 10 conseils pour une meilleure conception d'interface utilisateur

planzone.fr – Qu'est-ce que la méthode du Chemin Critique ?

planzone.fr – Qu'est-ce que la méthodologie Waterfall ?

www.appvizer.fr – Gestion de projet web : 9 conseils à suivre et erreurs à éviter

Annexe – Liste des compétences

Bloc	Compétence	Démontrée	Non démontrée	Pages
Bloc 1 – Fonction d'encadrement (10/19) - 52.6%				
	1.1 – Management des SI (4/7)			
	Identifier et analyser les besoins de l'entreprise au niveau organisationnel et système d'information			7-8, 12-15, 17-18, 26, 31, 4
	Elaborer un schéma directeur à partir d'orientations stratégiques			15
	Mettre en oeuvre des composants de management des processus (Business Process Management) en utilisant un logiciel de pilotage des gestions des flux			
	Modéliser et cartographier les processus métier en utilisant une méthode			7-8, 40, 48
	Assurer la confidentialité et l'intégrité des échanges en utilisant les outils de cryptographie et les certificats tiers de confiance			
	Elaborer une politique de sécurité pour l'utilisation des outils nomades tels que les smartphones, tablettes			16-17
	Mettre en oeuvre une politique de sécurité en utilisant des normes telles ISO27000			
	1.2 – Management des ressources (4/6)			
	Superviser et piloter une équipe projet			17-19, 26, 34-35
	Coordonner une équipe projet et intégrer la culture managériale de l'entreprise pour la faire partager			34-37
BLOC COMMUN	Identifier les leviers d'économie à actionner pour améliorer les processus Qualité			
	Elaborer des propositions d'amélioration de la Qualité en s'appuyant sur les principes du Lean Management			
	Utiliser de façon adéquate un contrat de services en relation avec un prestataire informatique			34-35
	Identifier les tactiques et stratégies pour choisir un prestataire tout en évaluant les risques pour l'entreprise			34-35
	1.3 – Ethique & Evolution du SI (2/4)			
	Déterminer les indicateurs green IT pour les utiliser dans la gestion informatique quotidienne			
	Evaluer les impacts des solutions informatiques en termes de responsabilités sociales et écologiques			
	Evaluer les impacts au sein du SI des solutions innovatrices proposées			48
	Rechercher et proposer des solutions innovatrices pour anticiper les évolutions et attentes de son secteur d'activité			17-25, 28, 30
	1.4 – Stratégie financière (0/2)			
	Lire et interpréter les indicateurs financiers d'un tableau de bord pour piloter son activité			
	Analyser un bilan et un compte de résultats pour mettre en place des indicateurs financiers			
Bloc 2 – Méthodes & Projet (4/8) - 50%				
	2.1 – Management de Projet (4/6)			
	Participer à la rédaction du cahier des charges d'un service à produire			11, 15
	Mettre en place une structure projet adéquate en analysant les besoins utilisateurs			26, 30-35
	Conduire un projet dans un contexte de forte contrainte de temps et produire les documents de suivi de son avancement			
BLOC COMMUN	Planifier un portefeuille de projets pour en assurer la coordination et les interdépendances			
	Participer à la présentation du cahier des charges d'un service à produire			11, 15
	Extraire et traiter l'information pertinente vers les publics ciblés			40, 48
	2.2 – Méthode de Gestion de Projet (0/2)			
	Gérer un projet en utilisant les principes de l'agilité de la méthode SCRUM			
	Mettre en place des indicateurs de mesure propres aux méthodes agiles			
Bloc 3 – Gestion des Données & Business Intelligence (6/11) - 54.5%				
	3.1 – Gestion des données (6/6)			
	Concevoir et utiliser la rétro-conception des bases de données			43
	Interroger une base de données au sein d'un système de gestion de bases de données client-serveur en utilisant le langage SQL			43, 45
	Maintenir et réutiliser du code au sein d'une base de données en développant des procédures stockées			43
	Déployer et administrer une base de données			44
	Utiliser une méthode de sauvegarde et de restauration d'une base de données appropriée			44
BLOC COMMUN	Sécuriser une base de données en utilisant le langage SQL du Système de Gestion de Bases de données			44
	3.2 – Business Intelligence (0/5)			
	Définir une architecture Business Intelligence			
	Modéliser un entrepôt de données en structurant différentes sources hétérogènes pour en faire une source homogène			
	Modéliser un datamart			
	Extraire des connaissances (Data Mining)			
	Assurer la qualité des données en utilisant les outils de gestion de la Qualité de données (Data Quality Management)			
Bloc 4 – Etudes & Développement (4/8) - 50%				
	Développer des composants logiciels en environnement objet			46-47
	Développer et déployer des solutions applicatives mobiles			
1 bloc au choix suivant sujet mémoire et parcours	Développer des solutions applicatives pour systèmes embarqués			
	Réaliser des tests unitaires en utilisant des règles d'erreur et exceptions			32-33
	Concevoir et exécuter des procédures de tests systématiques			32-33
	Gérer l'Intégration continue en utilisant des outils d'intégration			
	Elaborer des documents techniques décrivant un produit, un service ou une application			33
	Elaborer des modèles pour des publications techniques partagées			

Annexe – Attestation de non plagiat

EPSI l'École d'ingénierie informatique

ATTESTATION DE NON-PLAGIAT
(A inclure dans le mémoire professionnel)

Je soussigné(e)

Nom : Roura…….. Prénom : Florian…… ……

Auteur du mémoire professionnel pour le Titre de niveau 1 RNCP – Expert en informatique et Système d'information

L'évolution d'une application B2B de l'AS400 vers Angular……..

……

Déclare :

- Que ce mémoire est un document original, fruit d'un travail personnel
- Avoir obtenu les autorisations nécessaires pour la reproduction d'images, d'extraits, de tableaux, figures ou graphiques
- Ne pas avoir contrefait, falsifié, copié tout ou partie de l'œuvre d'autrui afin de la faire passer pour mienne ;

Atteste :

- Que toutes les sources d'information utilisées pour ce travail de réflexion et de rédaction sont référencées de manière exhaustive et claire dans la bibliographie/webographie de mon mémoire professionnel
- Etre informé(e) et conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, et que le plagiat est considéré comme une faute grave et sanctionné par la Loi.

Date et Signature :

Fait le…19/08/19……. …… A ……Aimargues……

Signature :