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.