Mémoire professionnel

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

# 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.

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

# Abstract

# Table des matières

[Remerciements 2](#_Toc16965668)

[Résumé du mémoire 3](#_Toc16965669)

[Abstract 4](#_Toc16965670)

[Table des matières 5](#_Toc16965671)

[Introduction 1](#_Toc16965672)

[Contexte 2](#_Toc16965673)

[Eminence 2](#_Toc16965674)

[Historique 2](#_Toc16965675)

[Aujourd’hui 3](#_Toc16965676)

[Ordonnancement et Service Négoce 4](#_Toc16965677)

[Workflow 6](#_Toc16965678)

[Outils du service Négoce 7](#_Toc16965679)

[Analyse de la demande 10](#_Toc16965680)

[Cahier des charges 10](#_Toc16965681)

[Problèmes actuels 10](#_Toc16965682)

[Interface 11](#_Toc16965683)

[AS400 12](#_Toc16965684)

[Choix et contraintes 14](#_Toc16965685)

[Base de données 15](#_Toc16965686)

[Portail 15](#_Toc16965687)

[Gestionnaire de sources 15](#_Toc16965688)

[Analyse des solutions possibles 17](#_Toc16965689)

[Mise en place du projet 26](#_Toc16965690)

[**Politique de structure et d’affichage** 27](#_Toc16965691)

[Planifier les étapes de développement en relation avec les fonctionnalités 30](#_Toc16965692)

# Introduction

# 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.

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.

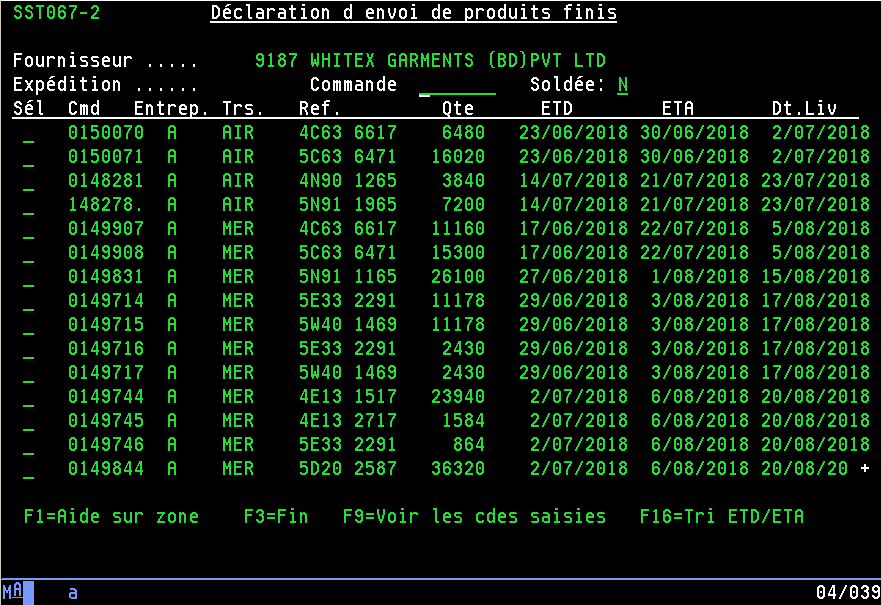
Suite à la réception des produits, et en assumant qu’ils passent avec succès le contrôle qualité (dont s’occupait récemment mon collègue et tuteur Vincent Descreux, au niveau informatique), ils sont enregistrés en base et comptabilisés, jusqu’au moment où la commande est considérée remplie par EMINENCE. 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.

## 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

## 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 :

* **Intégration :** 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.
* **Ouverte :** 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.
* **Intuitive :** 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.
* **Evolutive :** 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)
* **Dynamique :** 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

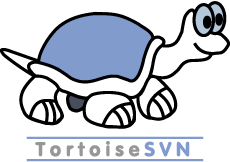
EMINENCE fonctionne avec la base de données IBM DB2. Ceci est dû au fait que l’entreprise utilise des serveurs AS400 pour stocker l’intégralité des fichiers. La base de données est le lieu d’échanges constants, et des rotations de scripts sont organisées afin de garantir son bon fonctionnement. Certains 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

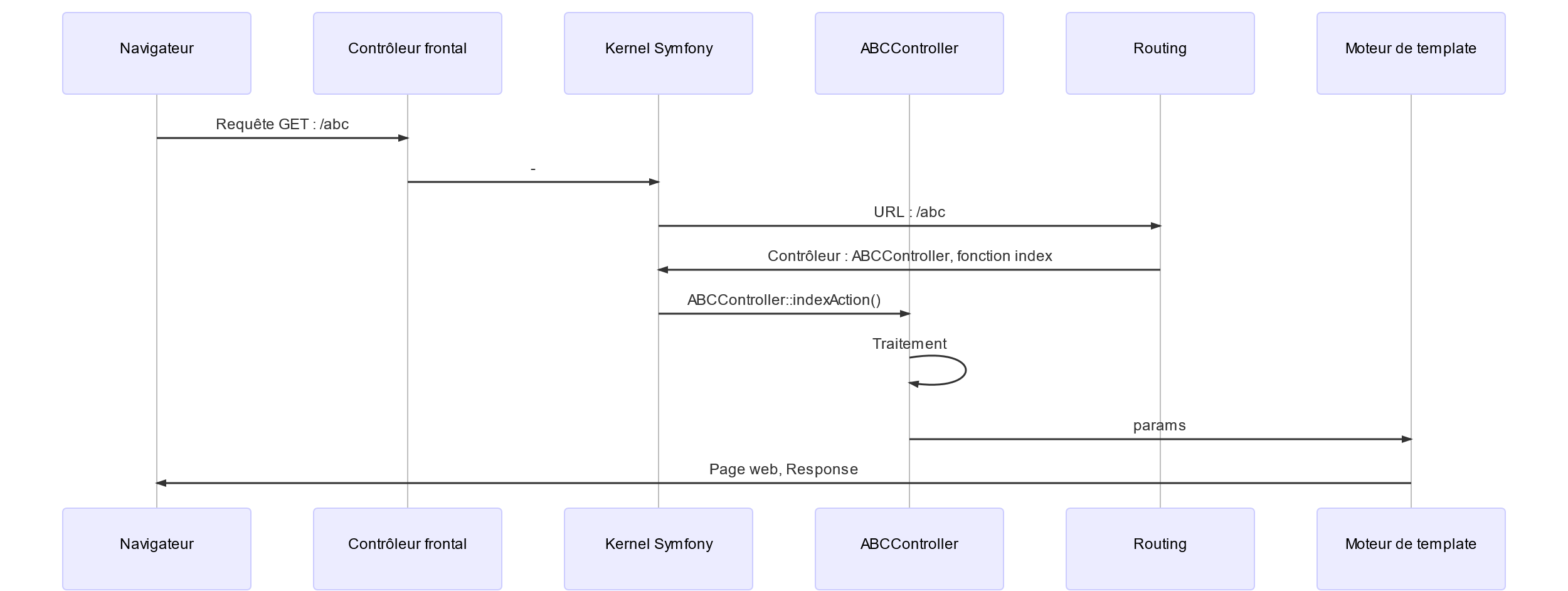
Simplement reprendre l’interface de l’AS400 et l’adapter pour le web semble de prime abord être la solution la moins fastidieuse : Moins de développement, mise au goût du jour, et l’application résultant de cette opération serait semblable aux outils actuels du service Négoce, tout en ajoutant des fonctionnalités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

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 :

