Département Informatique
Année spéciale
Promotion 2019-2020

Développement d'une application de gestion de commandes de matériel pour la Banque de Moselle

Projet soutenu le 19 juin 2020

Membres du Groupe

[redacted]

Yannis ARCORIO

[redacted]

Projet effectué au département informatique de l'IUT de Metz

Table des matières

I) Sujet du projet [redacted]

1) Répondre aux besoins des employés

Notre projet de synthèse a pour objectif de créer une application permettant la gestion des commandes d'une entreprise : la Banque de Moselle. Cette entreprise divisée en succursales comprenant chacune moins de 10 employés, souhaite que ces derniers puissent effectuer des commandes, de mobilier comme que de travaux de faible ampleur, auprès de l'un des fournisseurs se trouvant dans la liste à sa disposition. Nous devons donc développer une solution disposant d'une gestion des droits permettant aux employés, ici les utilisateurs, de formuler leurs besoins en matériel de travail et de faire valider cette demande avant de l'envoyer au fournisseur concerné.

2) Une modernisation nécessaire

Cette solution nous est demandée par la Banque de Moselle dans une optique de modernisation de son processus de commande. En effet, à ce jour, une seule personne nommée « responsable des commandes » s'occupe de répondre aux besoins des employés et donc, de passer toutes les commandes en utilisant des fiches de tableur, puis de les archiver. Ce fonctionnement rend tout le processus fastidieux et difficile à prendre en main. C'est pourquoi, l'application proposée va faciliter la rédaction des bons de commande grâce à une auto-complétion des champs et une simplification des démarches à suivre, permettant une plus grande automatisation du passage des commandes.

Le second problème, inhérent au fonctionnement actuel, concerne la gestion des documents à chaque étape de la commande avec un manque de centralisation des documents, et une utilisation de supports papier. C'est pourquoi l'utilisation d'une base de données et la gestion du bon de commande à chacune des étapes depuis l'application permettront un gain de temps et une réduction de la marge d'erreur dans le traitement des commandes.

La gestion actuelle des commandes pose également un problème de sécurité qui sera résolu facilement par l'application grâce à l'utilisation d'un identifiant, unique pour chaque utilisateur, couplé à un système de droits. Cela permettra de réguler l'accès aux données en fonction de l'identifiant de l'utilisateur. Toujours sur le plan de l'accès aux données, l'application offrira également un système de recherche beaucoup plus efficace grâce à l'utilisation de filtres permettant d'accélérer les recherches par rapport au système actuel, manuel et peu performant.

3) Une efficacité assurée par la répartition des tâches

L'application est destinée à être utilisée par les employés de la Banque de Moselle. Ces utilisateurs se verront attribuer un ou plusieurs rôles, et de ce fait, fonctions, par le biais du système de gestion des droits inclus dans l'application. Ce système vise à permettre une plus grande efficacité dans le traitement des bons de commandes par le truchement d'une division du processus.

Le premier rôle attribué est celui de Rédacteur. Le Rédacteur se trouve dans une succursale et c'est lui qui va se charger de rédiger le bon de commande suite à la demande d'un employé pour du matériel ou de la succursale pour des travaux.

Une fois rédigé, le bon de commande est transmis, par le Rédacteur, au Viseur. Le Viseur auquel est envoyé le bon de commande se trouve dans la même succursale que le Rédacteur de ce bon. Le Viseur va apposer un visa sur le bon de commande afin d'en confirmer la validité avant de l'envoyer au siège de l'entreprise.

Le rôle suivant est celui de Signataire, il est attribué à la personne au siège chargée d'officialiser la commande en la signant, puis de la transmettre au fournisseur.

Lorsque la commande est livrée, c'est le rôle du Réceptionnaire de confirmer la bonne réception par la succursale de la commande. Il fait donc partie de la même agence que le Rédacteur et que le Viseur de ce bon de commande.

Le dernier rôle dans l'application est celui d'Administrateur, c'est un employé dont la fonction est d'une part de gérer les droits d'utilisation de l'application, et d'autre part de maintenir à jour la base de données concernant les fournisseurs et leurs catalogues.

4) Un exemple concret

Pour illustrer le fonctionnement de l'application nous pouvons examiner la procédure d'envoi du bon de commande au fournisseur. Dans cette procédure, le Signataire se connecte à l'application et clique sur l'onglet « Bon signés », il est alors redirigé sur la page des bons de commande déjà signés. Il suffit alors au Signataire de cliquer sur le bouton « Envoyer » se situant à côté de la dénomination des bons. Il lui reste alors à confirmer l'envoi dans la pop-up qui vient d'apparaître et le bon est envoyé au fournisseur. En même temps que l'envoi du bon au fournisseur s'effectue, la commande est sortie de la liste des bons signés et est automatiquement placée dans la liste des bons envoyés.

II) Gestion du projet [redacted]

1) Répartition du travail

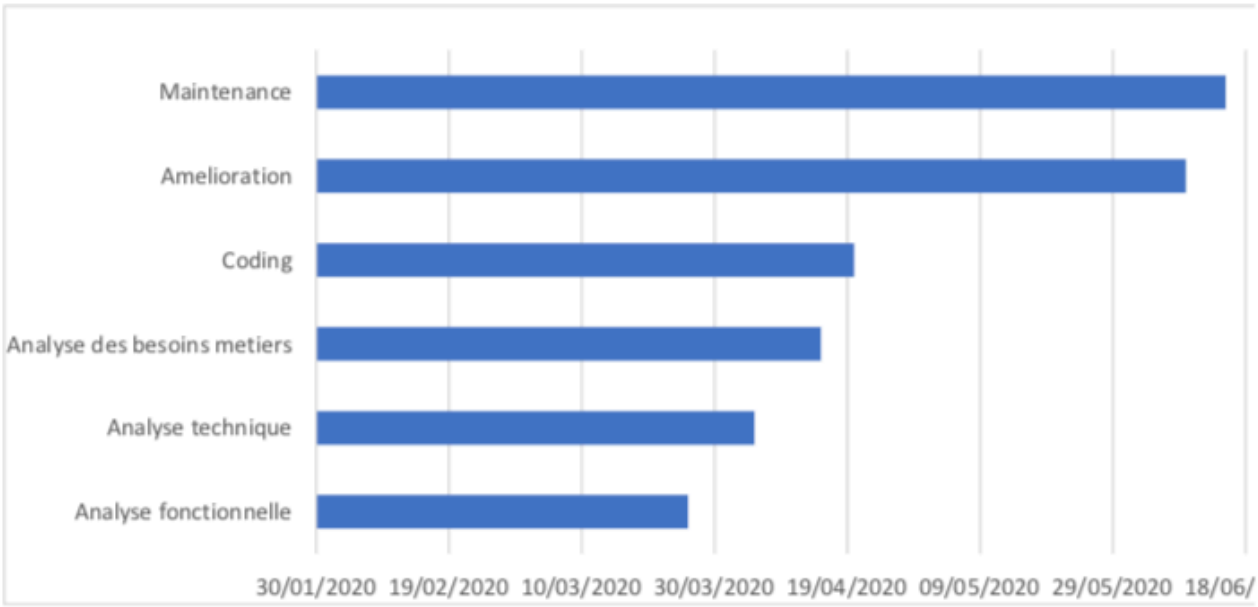
De par la conjecture actuelle, nous avons rencontré des difficultés pour ce qui concernait la gestion du projet. Ne pouvant pas nous voir physiquement, nous avons dû utiliser les avantages du numérique afin de mener à bien notre projet.

Un certain temps fut passé sur l'analyse fonctionnelle, en nous répartissant diverses tâches (diagrammes, user stories, schéma entité-association, schéma d'application), et sur l'élaboration du cahier des charges.

Nous avons également, dans le cadre de notre projet, une partie client et une partie serveur. Avec les membres du groupe nous nous sommes donc séparés afin d'agir sur tous les fronts. Ainsi donc nous avons pour la partie client : Xavier et Albin et pour la partie serveur Yannis, Matthieu et Shayan.

Malgré cette répartition du travail, nous avons très souvent travaillé ensemble et avions mis en place de fréquentes réunions virtuelles. Étant donné que nous fûmes en situation de découverte, recherche et d'apprentissage, l'application demandée n'a pas pu être achevée dans les délais, en plus des difficultés techniques notamment l'appréhension des nouveaux frameworks à savoir Spring Boot et Vue.js.

2) Diagramme de Gantt



Etape	Date début	Durée	Date de fin
Analyse fonctionnelle	26/03/2020	10	05/04/2020
Analyse technique	05/04/2020	10	15/04/2020
Analyse des besoins m	15/04/2020	5	20/04/2020
Coding	20/04/2020	50	09/06/2020
Amelioration	09/06/2020	6	15/06/2020
Maintenance	15/06/2020	6	21/06/2020

III) Analyse du sujet [redacted]

La Banque de Moselle est équipée en matériel orienté Microsoft. Elle possède des serveurs IIS et le système d'exploitation des ordinateurs de l'entreprise est Windows 10 Pro.

Un serveur IIS (Internet Information Services) « *est un serveur Web (HTTP) des différents systèmes d'exploitation Windows NT.* »[1]

1) Le type d'application

Au vu des besoins de la Banque de Moselle, nous avons décidé de développer une application de type client léger. Ce type est en effet plus adapté qu'un client lourd dans le cas où les mêmes informations doivent être accessibles par plusieurs usagers.

Un client léger communique avec un serveur central de traitement, ce qui n'oblige pas à installer l'application sur chaque poste de travail : l'utilisateur n'a besoin que d'un navigateur (dans notre cas, Firefox) pour accéder aux services de l'application. D'autres avantages offerts par un client léger sont la facilité de déploiement et de maintenance de l'application.

Ce type d'application présente aussi des désavantages : on peut considérer qu'elles sont moins sécurisées et il faut disposer d'une connexion d'internet pour pouvoir les utiliser.
Les applications de type client lourd, quant à elles, ne nécessitent pas de serveur central. Le traitement est effectué localement, ce qui implique d'installer l'application sur chaque poste et rend alors plus difficile la mise en place de l'application et possiblement ses mises à jour. Les applications de type client lourd ont cependant l'avantage de peu dépendre du réseau et elles sont donc souvent plus sécurisées.

2) Les technologies utilisées

A) Le coté serveur de l'application

- Java et le framework Spring Boot :

La partie serveur est codée en Java et s'appuie sur le framework Spring Boot.

« *Spring Boot est un nouveau framework créé par l'équipe de chez Pivotal, conçu pour simplifier le démarrage et le développement de nouvelles applications Spring. Le framework propose une approche dogmatique de la configuration, qui permet d'éviter aux*

[1] https://fr.wikipedia.org/wiki/Internet_Information_Services

développeurs de redéfinir la même configuration à plusieurs endroits du code. Dans ce sens, Boot se veut d'être un acteur majeur dans le secteur croissant du développement d'applications rapide. »[2]

En Java, Spring Boot est un framework rapide et efficace qui nous permet de réaliser une API de web service.

- Base de données et identification :

Pour stocker et gérer la base de données nous avons utilisé SQL Server à la demande de l'entreprise.

« *Microsoft SQL Server est un système de gestion de base de données (SGBD) en langage SQL incorporant entre autres un SGBDR (SGBD relationnel) développé et commercialisé par la société Microsoft*. »[3]

Ce moteur est capable de supporter un grand nombre de base de données et il est de plus performant en environnement Microsoft. L'administration et l'optimisation en sont facilitées par des outils.

Étant donné que plusieurs rôles ont été définis dans l'application, nous avions besoin d'un outil interne à l'application pour identifier et gérer les droits des utilisateurs. Nous avons donc utilisé à cette fin un autre service de Microsoft, l'Active Directory (AD).

« *Active Directory (AD) est la mise en œuvre par Microsoft des services d'annuaire LDAP pour les systèmes d'exploitation Windows. L'objectif principal d'Active Directory est de fournir des services centralisés d'identification et d'authentification à un réseau d'ordinateurs utilisant le système Windows, MacOs et encore Linux. Il permet également l'attribution et l'application de stratégies ainsi que l'installation de mises à jour critiques par les administrateurs. Active Directory répertorie les éléments d'un réseau administré tels que les comptes des utilisateurs, les serveurs, les postes de travail, les dossiers partagés (en), les imprimantes, etc. Un utilisateur peut ainsi facilement trouver des ressources partagées, et les administrateurs peuvent contrôler leur utilisation grâce à des fonctionnalités de distribution, de duplication, de partitionnement et de sécurisation de l'accès aux ressources répertoriées.* »[4]

En particulier pour cette application, on a utilisé l'un des protocoles de l'Active Directory, LDAP.

« *LDAP (Lightweight Directory Access Protocol) est un protocole d'application permettant d'interroger et de modifier des éléments dans des fournisseurs de services d'annuaire comme Active Directory, qui prend en charge une forme de LDAP* »[5]

[2] https://www.infoq.com/fr/articles/microframeworks1-spring-boot/
[3] https://fr.wikipedia.org/wiki/Microsoft_SQL_Server
[4] https://fr.wikipedia.org/wiki/Active_Directory
[5] https://www.it-swarm.dev/fr/active-directory/quelles-sont-les-differences-entre-ldap-et-active-directory/957891010/

B) Le coté client de l’application

Nous avons développé la partie client et l’interface de l’application à l’aide d’un framework de JavaScript, Vue.js, qui est aujourd’hui un outil très utilisé dans le développement web.

« *Vue.js (aussi appelé plus simplement Vue), est un framework JavaScript open-source utilisé pour construire des interfaces utilisateur et des applications web monopages. Vue a été créé par Evan You et est maintenu par lui et le reste des membres actifs de l'équipe principale travaillant sur le projet et son écosystème*. »[6]

Vue.js est un cadre réactif pour la création d'applications Web. Vue est différent des autres frameworks JavaScript dans la mesure où il n'est pas aussi intrusif. Outre la bibliothèque principale, Vue se compose d'un ensemble de bibliothèques facultatives qui peuvent être combinées pour créer des applications plus avancées. Cela signifie que le cadre peut être adopté progressivement, et vous pouvez facilement faire fonctionner certaines parties de votre application par Vue. C'est un énorme avantage, car les applications existantes peuvent être réécrites en petites étapes au lieu de réécrire la totalité de l'application en une seule fois. C'est quelque chose qui est beaucoup plus facile à réaliser avec Vue.js que de nombreux autres frameworks tels que Angular, qui ont tendance à prendre plus de contrôle sur votre application.

C) Architecture de l’application

L’architecture de l’application se fonde sur le design pattern MVC et l’architecture REST.

« *Modèle-vue-contrôleur ou MVC est un motif d'architecture logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.*

- *Un modèle (Model) contient les données à afficher.*
- *Une vue (View) contient la présentation de l'interface graphique.*
- *Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.* »[7]

MVC est un modèle architectural sur lequel peut s’établir la construction d’un logiciel. L'idée de base est de séparer les modèles de données internes de l'interface utilisateur via le contrôleur et la vue. Il domine le développement Web et mobile, et bien que certaines alternatives existent, presque tous les logiciels côté serveur pertinents sont développés avec un framework compatible MVC.

[6] https://fr.wikipedia.org/wiki/Vue.js
[7] https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur

« *REST (representational state transfer) est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web. Les services web conformes au style d'architecture REST, aussi appelés services web RESTful, établissent une interopérabilité entre les ordinateurs sur Internet. Les services web REST permettent aux systèmes effectuant des requêtes de manipuler des ressources web via leurs représentations textuelles à travers un ensemble d'opérations uniformes et prédéfinies sans état. D'autres types de services web tels que les services web SOAP exposent leurs propres ensembles d'opérations arbitraires.* »[8]

Dans le style architectural REST, l'implémentation du client et l'implémentation du serveur peuvent se faire indépendamment sans se connaître. Cela signifie que le code côté client peut être modifié à tout moment sans affecter le fonctionnement du serveur, et le code côté serveur peut être modifié sans affecter le fonctionnement du client. Tant que chaque partie connaît le format des messages à envoyer à l'autre, elles peuvent être modulaires et séparées.

En séparant les problèmes d'interface utilisateur des problèmes de stockage de données, nous améliorons la flexibilité de l'interface entre les plates-formes et améliorons l'évolutivité en simplifiant les composants du serveur. De plus, la séparation permet à chaque composant d'évoluer indépendamment.

En utilisant une interface REST, différents clients atteignent les mêmes points de terminaison REST, effectuent les mêmes actions et reçoivent les mêmes réponses. Lorsque nous créons des API, nous voulons que nos modèles fournissent quatre types de fonctionnalités de base. Le modèle doit pouvoir créer, lire, mettre à jour et supprimer des ressources. Les informaticiens se réfèrent souvent à ces fonctions par l'acronyme CRUD. Un modèle doit avoir la capacité d'exécuter au plus ces quatre fonctions pour être complet.

Dans un environnement REST, CRUD correspond souvent aux méthodes HTTP : POST, GET, PUT et DELETE, respectivement. Ce sont les éléments fondamentaux d'un système de stockage persistant.

REST nécessite qu'un client fasse une demande au serveur afin de récupérer ou de modifier des données sur le serveur. Avec HTTP que nous utilisons dans les demandes d'interaction avec les ressources dans un système REST :

- GET – permet de récupérer une ressource spécifique (par id) ou une collection de ressources
- POST – permet de créer une nouvelle ressource
- PUT – permet de mettre à jour une ressource spécifique (par id)
- DELETE – permet de supprimer une ressource spécifique (par id)

[8] https://fr.wikipedia.org/wiki/Representational_state_transfer

D) Choix des outils de développement

-IntelliJ Idea : le projet est développé sur l'IDE Intellij Idea, cet IDE a été choisi parce qu'il nous permet de développer à la fois les parties back end et front end du projet. Il supporte le framework Spring Boot et il intègre aussi des outils de gestion projet comme Git et Maven.

-Git : Un service de gestion de version (Version Control System) qui nous permet d'enregistrer des versions différentes d'un projet. Un outil très pertinent, surtout pour travailler en équipe à distance.

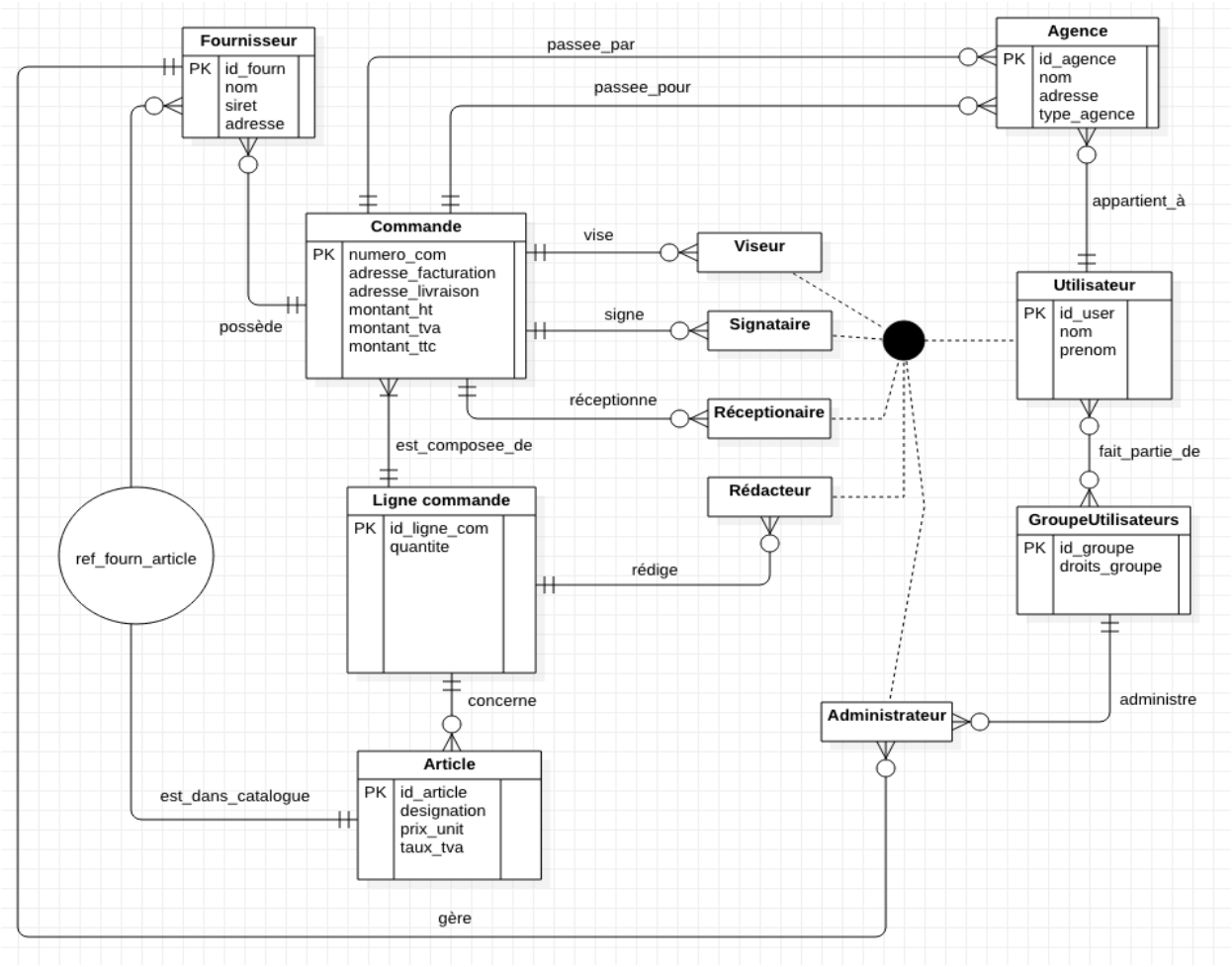
3) Le schéma entité-association de l'application

« *Le modèle entité-association (EA) (le terme « entité-relation » est une traduction erronée largement répandue), ou diagramme entité-association ou (en anglais « entity-relationship diagram », abrégé en ERD), est un modèle de données ou diagramme pour des descriptions de haut niveau de modèles conceptuels de données. Il a été conçu par Peter Chen dans les années 1970 afin de fournir une notation unifiée pour représenter les informations gérées par les systèmes de gestion de bases de données de l'époque. Il fournit une description graphique pour représenter des modèles de données sous la forme de diagrammes contenant des entités et des associations. De tels modèles sont utilisés dans les phases amont de conception des systèmes informatiques.* »[9]

Un diagramme EA montre la relation entre des ensembles d'entités. Un ensemble d'entités est un groupe d'entités similaires et ces entités peuvent avoir des attributs. En termes de SGBD, une entité est une table ou un attribut d'une table dans la base de données, donc en montrant la relation entre les tables et leurs attributs, le diagramme EA montre la structure logique complète d'une base de données.

Dans notre cas pour gérer la relation des différents rôles on a employé un modèle SGBD de spécialisation. L'idée derrière la spécialisation est de trouver les sous-ensembles d'entités qui ont peu d'attributs distincts. Par exemple, considérons un employé d'entité qui peut être classé comme sous-entités Technicien, Ingénieur & Comptable car ces sous-entités ont des attributs distincts.

[9] https://fr.wikipedia.org/wiki/Mod%C3%A8le_entit%C3%A9-association



(*Schéma réalisé par Yannis, Matthieu, Shayan*)

IV) Réalisation (Yannis Arcorio)

1) Vers un cahier des charges

Durant la première partie du projet, notre but a été la mise en place d'une analyse fonctionnelle et technique afin de réaliser un cahier des charges des plus détaillés et des plus clairs pour le client.

L'analyse fonctionnelle se découpe en plusieurs parties majeures. La présentation de l'entreprise et du sujet, l'analyse du besoin métier et des contraintes où l'on présente un diagramme d'activité et d'états-transitions.

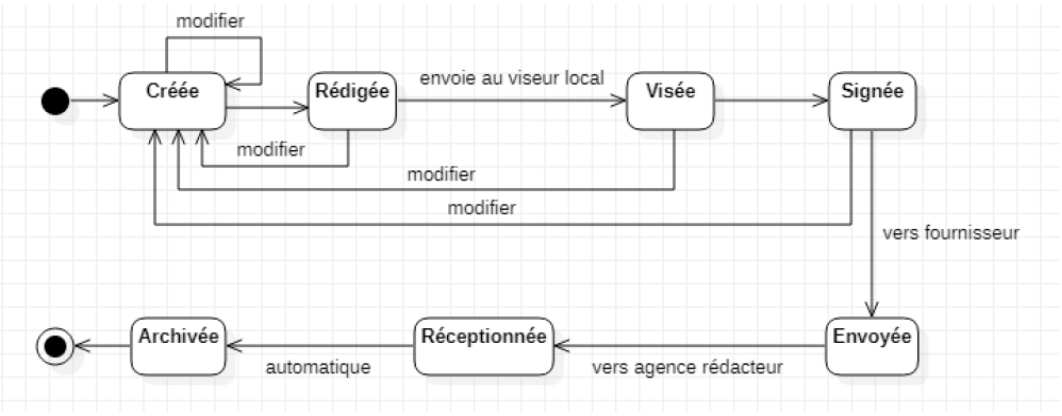


Figure 1 : Diagramme d'états-transitions

Les acteurs :

Nom acteur	Descriptif	Attentes
Rédacteur	Rédige un bon de commande (niv. local)	Rédiger un bon de commande et le transmettre
Viseur	Appose un visa aux bons de commande reçus (niv. local)	Viser le bon de commande
Signataire	Signe les bons de commande et les transmets aux fournisseurs (niv. siège)	Signer le bon de commande et le transmettre (signature)
Réceptionnaire	Réceptionne la commande passée	Confirmer la réception de la commande
Administrateur	Gère les utilisateurs et ajoute des fournisseurs et leurs catalogues	Gérer les droits d'utilisation de l'application / Alimenter les banques de données

Les besoins :

Besoin n°	Nom	Description	Problème(S) Soulevé(S)	Solutions
1	Centraliser le processus des commandes	Ajout (création), modification et archivage d'un bon de commande	Toutes les données ne sont pas numérisées et problème de centralisation de ces données	Centraliser le processus à l'aide d'une bdd
2	Assurer la sécurité des commandes	Assurer une bonne gestion des droits utilisateurs	Le système de droits lecture/ecriture des fichiers est fastidieux	Donner l'accès aux données selon un id utilisateur
3	Simplifier les démarches	Simplifier la rédaction des bons de commande	Pas assez automatisé. Une seule personne connait parfaitement les procédures	Automatiser la rédaction par une autocomplétion des champs. Simplification des démarches et procédures à suivre
4	Faciliter l'accès aux données	Rechercher des bons de commande par filtre	Système de recherche pas assez performant	Création d'un système de recherche rapide et efficace (filtres)

Et pour finir, les cas d'utilisation à l'aide de « User Story » et de diagrammes de cas d'utilisation.

User Story / / Diagramme cas d'utilisation n°1

Scénario : se connecter au système (cas de succès)

En tant que collaborateur de la Banque de Moselle, je souhaite me connecter à l'application de commande de matériel. Étant donné que je possède un compte utilisateur numérique dans l'Active Directory de l'entreprise, lorsque je lance l'application je saisis alors mes identifiants habituels. Puis, je valide la saisie en cliquant sur le bouton « Valider »

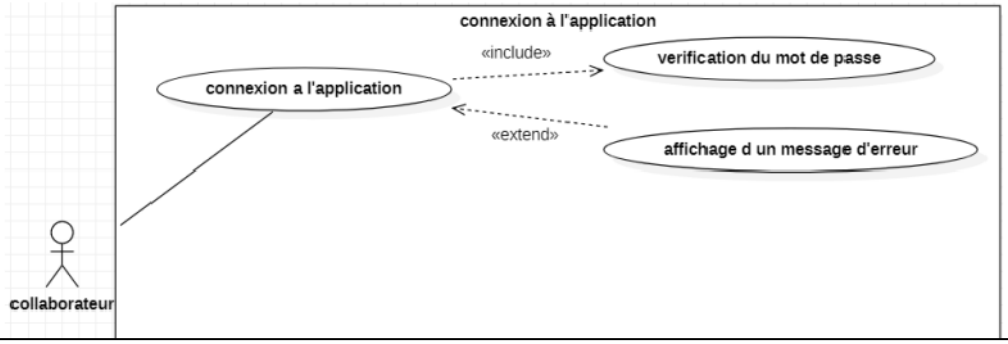
Alors, je suis redirigé vers l'interface utilisateur correspondant à mes droits d'accès au service*

*extension , mon login ou mon mot de passe est erroné alors, l'application me prévient en affichant un message d'erreur : « Login et/ou mot de passe incorrect(s) »

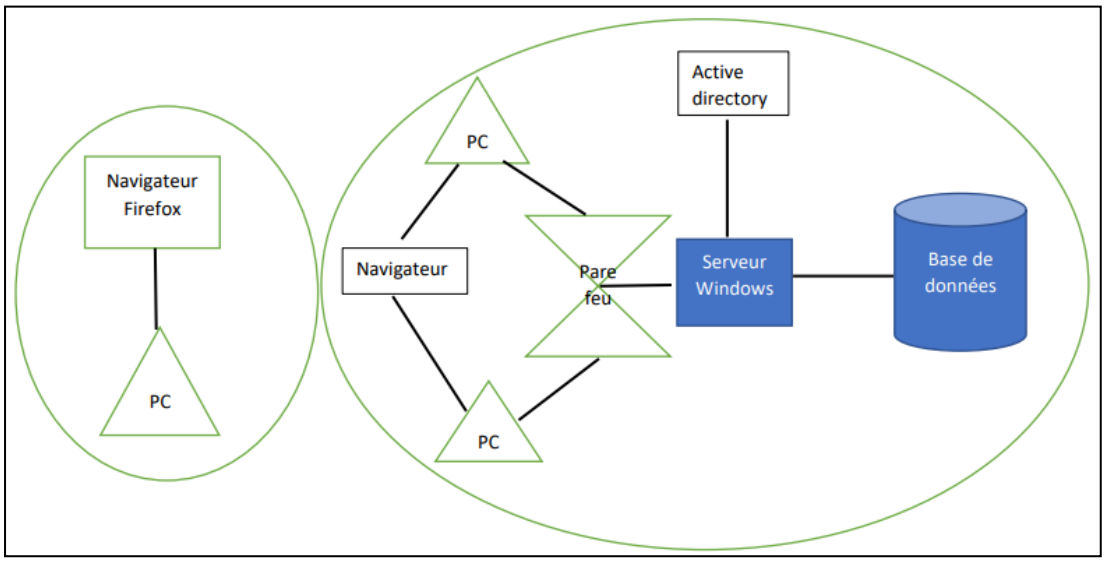
J'ai la possibilité de saisir de nouveaux identifiants pour réessayer de me connecter au service

(Retour à l'étape de saisie et de validation de la User Story 1).

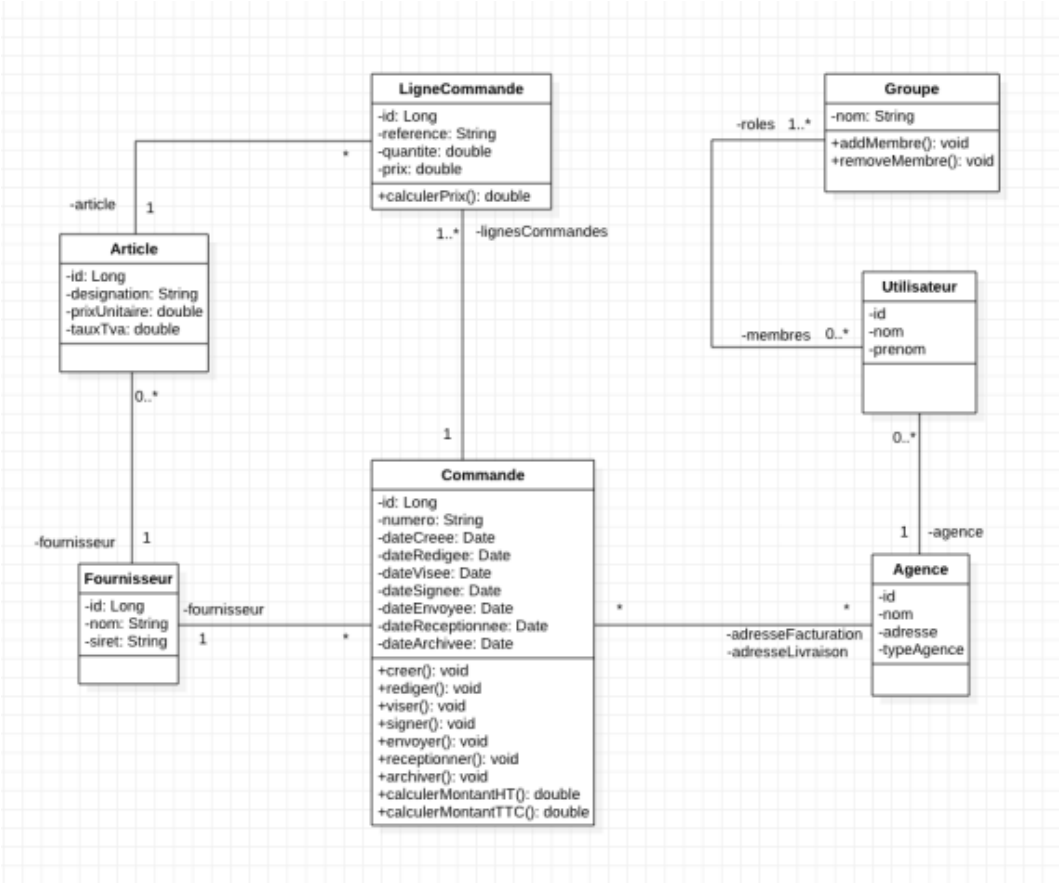
Diagramme cas d'utilisation



Après quoi nous avons effectué une analyse technique du projet. Pour cela nous avons réalisé un schéma de l'application :



Ainsi qu'un diagramme de classe :

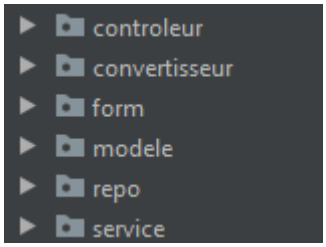


Tout ce travail, réalisé sur les cinq premières semaines du projet, nous a permis de fournir un cahier des charges clair et précis (cf. Cahier des charges).

2) Une partie serveur

L'application se base sur le design pattern MVC (Model-View-Controler). De plus, nous avons utilisé l'architecture REST pour la partie Client-Serveur. Le code est écrit en Java et nous nous sommes appuyés sur le framework Spring Boot. En ce qui concerne le lien avec la base de données, nous utilisons un ORM (Object Relational Mapping), Hibernate.

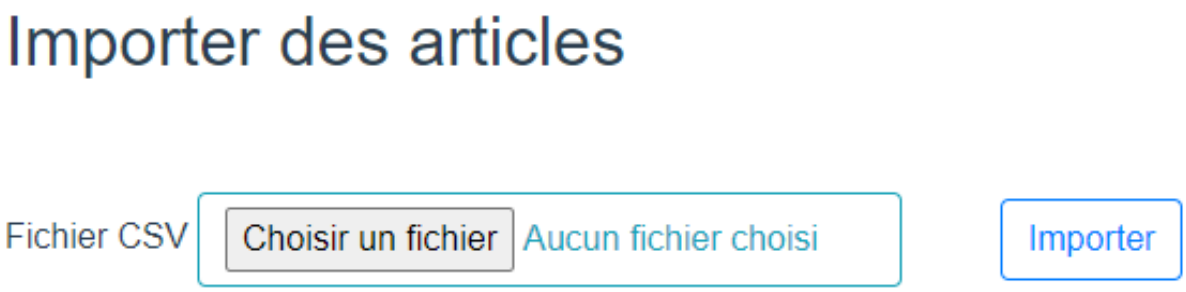
Nous avons donc découpé le code en plusieurs parties. Du côté serveur on y trouve les parties modèle, service, repository, formulaire et contrôleur ce qui rend le code très lisible et facile à modifier en cas d'implémentation de nouvelles fonctionnalités.



Pour l'authentification, le client exigeait une connexion via l'Active Directory Microsoft déjà en place au sein de la Banque de Moselle. L'Active Directory de Microsoft s'appuie en particulier sur le protocole LDAP (Lightweight Directory Access Protocol) qui définit des pratiques d'accès à des annuaires. De cette façon nous pouvons définir des

groupes (en fonction des acteurs) pour gérer les droits. De plus, nous pouvons même récupérer l'adresse de l'agence du Rédacteur par exemple, pour que la livraison se fasse automatiquement à cette adresse mais c'est une fonctionnalité que nous n'avons pas mise en place.

La commande, objet central de l'application, peut avoir plusieurs états. Nous avons utilisé le design pattern « State » pour mettre ces différents états en place. Une commande possède plusieurs lignes de commande et à chaque ligne correspond un article et une quantité. Nous avons donc mis en place, pour le Gestionnaire, une possibilité d'import de fichiers CSV correspondant aux collections d'articles du fournisseur afin d'alimenter la BdD, ce qui permet au Rédacteur une sélection plus facile des articles dans une commande.

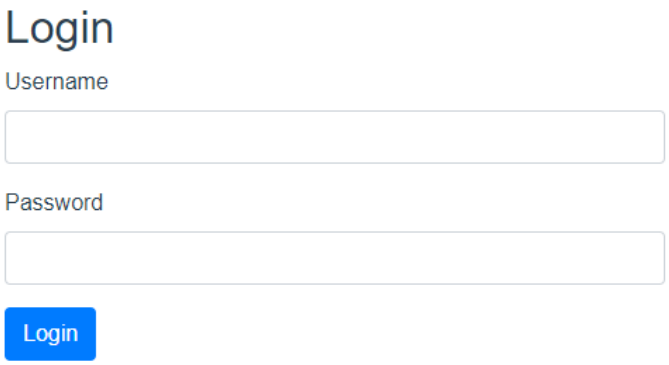


Cependant, il nous reste encore du travail à faire au niveau des droits utilisateurs, des tests unitaires (partie importante lors du développement).

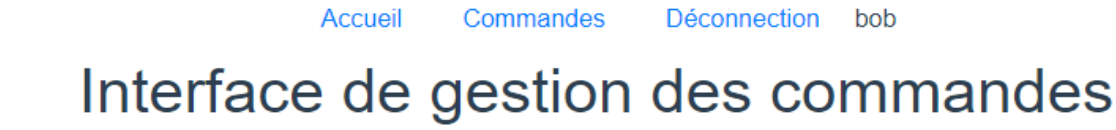
3) Une partie client

Cette partie repose sur l'utilisation de Vue.js, un framework JavaScript inspiré d'AngularJS mais qui reste plus « léger ». Pour l'aspect visuel nous utilisons le framework open source Bootstrap.

Pour l'authentification à l'application, nous avons réalisé une simple page de connexion classique mais efficace.

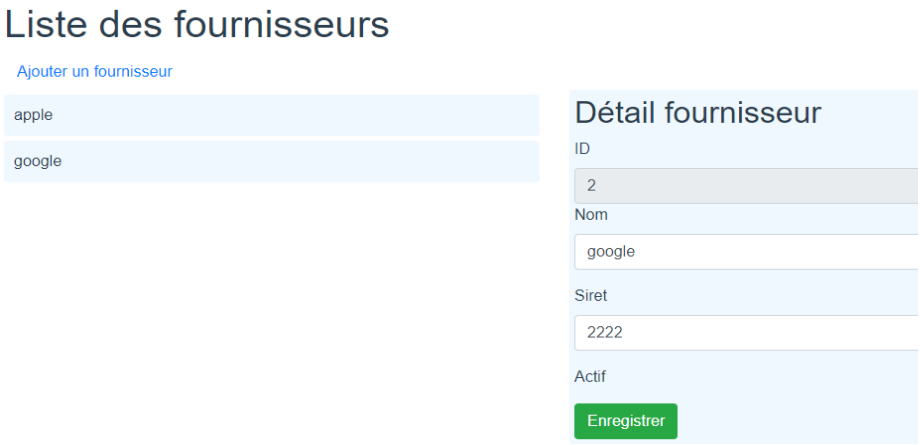


Une fois l'utilisateur connecté, l'application se présente comme ci-dessous :

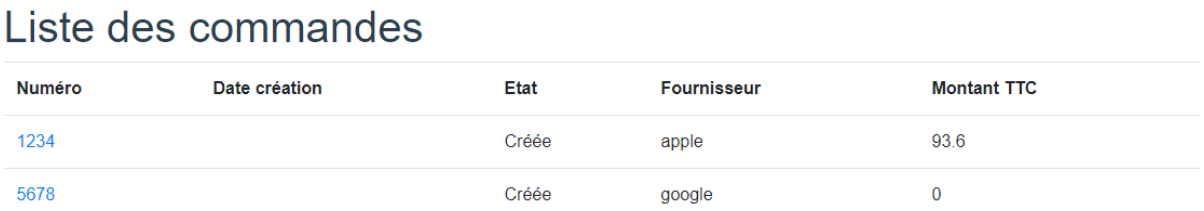


Nous pouvons voir que « bob » est connecté et qu'il a accès aux pages « Accueil », « Commandes » et « Déconnection ». Tandis que « ben », qui n'a pas les mêmes droits que « bob », n'aura forcément accès aux mêmes pages.

Nous pouvons gérer les fournisseurs, accéder à leurs détails, les modifier ou en créer. Nous avons créé un booléen permettant à chaque fournisseur d'avoir un état « actif » ou « inactif » conditionnant son affichage côté client tout en ne l'effaçant pas complètement de la BdD en cas de suppression par l'utilisateur.

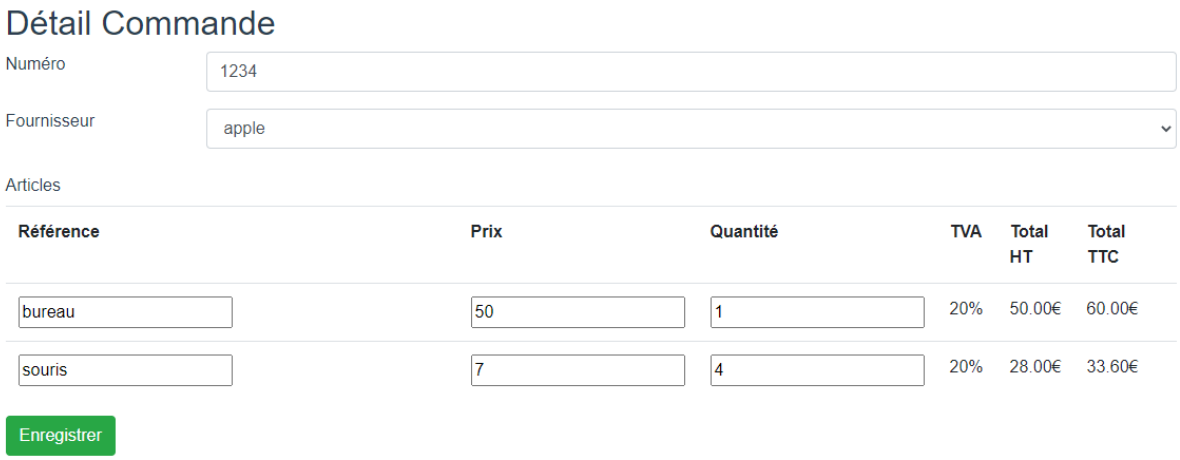


C'est sur ce même principe qu'est construite la page « Liste des commandes », où l'on peut voir un listing des commandes, avec leur numéro, leur date de création (pas encore



implémenté), leur état, leur fournisseur et leur montant total TTC. Les détails d'une commande apparaissent lorsque l'on clique dessus.

Nous n'avons affiché pour l'instant que quelques champs mais il reste encore à implémenter les droits de modification, faire apparaitre quelques dates importantes comme la date de la dernière modification par exemple.



Le travail sur les commandes n'est pas encore fini côté client, il reste beaucoup de fonctionnalités à mettre en place pour que tout se déroule correctement et le plus simplement possible car nous ne perdons pas de vue que l'objectif premier est de rendre plus simple la création et la gestion des bons de commandes.

V) Conclusion et perspectives [redacted]

1) L'importance de l'analyse du besoin métier

Le projet Banque de Moselle nous aura permis de mettre en pratique différents outils et connaissances que nous avons acquises tout au long de cette année de D.U.T. Plus important encore, la réalisation de ce projet de synthèse nous aura fait prendre conscience des différentes étapes à mettre en place dans le cadre d'un développement d'application. Ainsi, nous avons pu constater l'importance de l'analyse fonctionnelle dans la définition du besoin métier du maître d'ouvrage : comprendre la demande à satisfaire fut en effet essentiel avant de procéder à toute opération de programmation. Celle-ci nous a assuré un développement plus aisé et répondant de manière précise au besoin initial. Cette étape d'analyse, accompagnée de l'établissement de divers diagrammes et schémas (état, transition, schéma entité-association et diagramme de classe), nous aura donc permis de tracer une véritable feuille de route pour notre développement.

L'analyse technique, découlant elle aussi de la demande du client, nous a fait nous rendre compte de l'importance de la compréhension d'une demande quant aux choix technologiques à effectuer en fonction des contraintes matérielles et logicielles d'un client ou encore par rapport à la nature du besoin auquel il convient d'apporter une réponse logicielle. La demande formulée par la Banque de Moselle a en effet eu une influence dans le choix du type d'application développé (application web retenue pour la légèreté du modèle client-serveur et sa facilité de déploiement) et des architectures mises en place (architecture REST conférant une bonne séparation client-serveur pour une flexibilité et une évolutivité accrues). Nous nous sommes donc aperçus de la nécessité en tant que développeurs d'être polyvalents et capables de nous adapter afin d'aborder de nouveaux outils pour satisfaire au mieux une demande.

2) Une appréhension de technologies employées dans le milieu professionnel

Sur le plan de la programmation, le projet de synthèse aura été l'occasion pour nous de renforcer notre compréhension des concepts de la programmation orientée objet que nous avions déjà rencontrés et d'en appliquer de nouveaux (injection de dépendances, patrons de conception « État » et « Convertisseur » entre autres). Nous avons aussi pu appréhender des outils couramment employés dans le monde professionnel (utilisation de Spring Boot, d'un ORM via Hibernate, du framework JavaScript Vue.js et de l'Active Directory). De plus, le travail en équipe nous aura fait manipuler des outils standards de développement, tels que Github et également fait comprendre l'importance des tests unitaires dans le cadre d'un développement collaboratif afin d'éviter les régressions. Ce projet s'est donc avéré être un excellent moyen de consolider les connaissances que nous avons acquises les mois passés tout en étant très riche en découvertes.

3) De la nécessité d'une gestion de projet rigoureuse

Nous avons tous pu faire le constat de la nécessité de mettre en place une gestion de projet rigoureuse lors du développement d'applications nécessitant un travail en équipe. Nous avons eu des difficultés à nous organiser de manière efficiente afin d'avancer de manière cohérente, ce qui a sans doute contribué à l'inachèvement de notre application dans le délai qui nous était imparti. Nous avons parfois fait le choix d'amener l'ensemble de l'équipe à essayer de mettre en place les mêmes éléments, et ce, dans un but d'apprentissage primant alors sur l'efficacité qu'aurait apportée une division stricte des tâches en fonction des compétences et des appétences de chacun.

4) Perspectives d'évolution

Nous avons, en cours de développement, souvent pensé à des améliorations possibles par rapport à notre projet de départ mais ces dernières ne faisaient pas partie des contours dessinés en amont avec le maître d'ouvrage.

Tout d'abord, notre application n'est optimisée que pour un contexte d'utilisation très précis (système d'exploitation Windows 10, utilisation de Firefox sur des PCs de bureau). Dans un souci d'adaptabilité, une évolution possible pour l'application consisterait à s'assurer de son bon fonctionnement sur d'autres navigateurs. Nous pourrions également envisager d'adapter notre client à d'autres supports, voire créer une version mobile de l'application, afin qu'elle puisse s'exécuter convenablement sur des tablettes ou des smartphones. Ensuite, nous avions imaginé qu'il pourrait être intéressant d'intégrer ultérieurement à l'application un module de gestion budgétaire pour les Viseurs et le Gestionnaire. Celui-ci permettrait à ces acteurs de juger plus aisément de la pertinence d'une commande rédigée, relativement au budget de l'antenne concernée. Enfin, nous pourrions aussi proposer une interface permettant à des fournisseurs de mettre à jour directement leur catalogue d'articles afin de garantir une meilleure fiabilité des informations sur lesquelles s'appuie notre application.

5) Quelques remerciements

L'ensemble de notre groupe souhaite enfin remercier très chaleureusement notre tuteur, M. Iggiotti, pour son aide, sa patience et sa bienveillance tout au long du projet de synthèse. Sa passion pour la transmission de connaissances s'est ressentie dès le commencement de ce projet et a su susciter notre enthousiasme, ce qui nous a permis d'enrichir considérablement notre compréhension du processus de développement d'applications et de la programmation.