République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab de Blida

N° D'ordre : …….

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

Allali Mustapha Khelifa **Benali Yousri**

Thème : « Developpement d'une application pour la gestion

Des licences en temps réel »

En vue d'obtenir le diplôme de master **D**omaine : Mathématique et informatique

Filière : Informatique **S**pécialité : Informatique

Option : Ingénierie des Logiciels (IL)

Promotrice : Mme.BOUMAHDI Fatima Maître de conférences

Encadreur : Mr. NEBHI Mourad Chef de département information et Technologies

Sonatrach Hydra.

Promotion

2018 / 2019

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab de Blida

N° D'ordre : …….

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

Allali Mustapha Khelifa **Benali Yousri**

Thème : « Developpement d'une application pour la gestion Des licences en temps réel »

En vue d'obtenir le diplôme de master **D**omaine : Mathématique et informatique

Filière : Informatique **S**pécialité : Informatique

Option : Ingénierie des Logiciels (IL)

Promotrice : Mme.BOUMAHDI Fatima Maître de conférences

Encadreur : Mr. NEBHI Mourad Chef de département information et Technologies

Sonatrach Hydra.

Promotion

2018 / 2019

Résumé

*Parmi les importantes missions de la division **PED (Petrolium Enginnering & Développement)** au sein de la **SONATRACH**, est le suivi des gisements pétrolières l'élaboration des plans de développement et d'exploitation pétroliers.*

*Pour ce faire, les ingénieurs métiers de la division **PED**, utilisent une multitude de logiciels de nature géoscience (géologiques, géophysiques et pétro-physiques) provenant de plusieurs prestataires, principalement Schlumberger et Haliburton. Le droit d'utilisation et l'usage de ces logiciels sont régis par des licences **Flexnet**.*

*Le présent projet de fin d'études s'inscrit dans le cadre de développement d'une solution pour la gestion des licences flexnet, par **la Programmation Réactive**. Et la récupération des données liées à l'utilisation des licences logiciels.*

*Le choix porte sur la programmation « **Réactive** », afin de rendre le résultat disponible via les **WebFlux**.*

Un traitement en background doit être développé pour récupérer les données provenant des serveurs de licences et devrait être le plus performant possible. À cet effet, les threads seront utilisés pour pallier cette contrainte.

*Le module à développer doit aussi assurer le rôle du **Server Réactive SSE (Réactive Server Sent Event)**, moyennant les **WebFlux** pour véhiculer les résultats en format **JSON (RESTFUL** en continu).*

Mots-clés

Reactive SSE, Web Flux, NoSQL, JSON, RESTFUL, FlexNet, API.

Abstract

Among the important missions of the Petrolium Enginnering & Development Division (PED) within SONATRACH is the oilfield monitoring and development of petroleum development and exploitation plans.

To do this, PED's business engineers use a multitude of geoscience software (geological, geophysical and petro-physical) from several service providers, mainly Schlumberger and Haliburton. The right to use and use of such software is governed by Flexnet licenses.

This end-of-studies project is part of the development of a solution for the management of flexnet licenses through reactive programming. And the recovery of data related to the use of software licenses.

The choice is on "reactive" programming, in order to make the result available via WebFlux.

Background processing must be developed to recover data from license servers and should be as efficient as possible. For this purpose, the threads will be used to pal-link this constraint.

The module to be developed must also ensure the role of the Reactive Server SSE (Reactive Server Sent Event), by means of WebFlux to convey the results in JSON format (RESTFUL in continuous).

Keywords

Reactive SSE, Web Flux, NoSQL, JSON, RESTFUL, FlexNet PAI.

ملخص

من بين المهام المهمة لقسم بتروليوم للهندسة والتطوير (PED) في سوناطراك مراقبة حقول النفط وتطوير خطط تطوير واستغلال النفط.

للقيام بذلك، يستخدم مهندسو الأعمال في PED عددًا كبيرًا من برامج علوم الأرض (الجيولوجية والجيوفيزيائية والبتروكيميائية) من العديد من مقدمي الخدمات، ولا سيما Schlumberger وHaliburton. يخضع حق استخدام مثل هذه البرامج واستخدامها لتراخيص Flexnet.

يعتبر مشروع نهاية الدراسات هذا جزءًا من تطوير حل لإدارة تراخيص Flexnet من خلال البرمجة التفاعلية. والغرض منه هو تطوير وحدات وخدمات لاستعادة البيانات المتعلقة باستخدام تراخيص البرمجيات.

الخيار هو البرمجة "التفاعلية"، من أجل إتاحة نتائج العلاجات عبر WebFlux. بمعنى آخر، سيتم إنشاء البيانات وإتاحتها بشكل مستمر على خادم ويب عبر دفق بيانات مستمر، بغض النظر عن وجود عميل على تدفق (أي: على رابط الاتصال).

يجب تطوير معالجة الخلفية لاستعادة البيانات من خوادم الترخيص ويجب أن تكون فعالة قدر الإمكان. لهذا الغرض، سيتم استخدام مؤشرات الترابط للتغلب على هذا القيد.

يجب أن تفترض الوحدة التي سيتم تطويرها أيضًا دور حدث إرسال الخادم التفاعلي (Réactive Server Sent Event)، عن طريق WebFlux لنقل النتائج بتنسيق JSON (RESTFUL بشكل مستمر).

الكلمات المفتاحية

Réactive SSE، WebFlux، NoSQL، JSON، RESTFUL، Flexnet ، واجهة برمجة تطبيق.

Remerciements

Nous exprimons nos vifs remerciements à Dieu le tout-puissant qui nous a donné la force, la volonté et le courage pour que ce travail puisse voir le jour.

Notre profonde gratitude à la grande société nationale **SONATRACH** de Hydra qui nous a accueillis et mis à notre disposition tous les moyens nécessaires pour la réussite de notre stage et à Monsieur **NEBHI Mourad** notre encadreur de stage qui nous a, non seulement, proposé le thème de ce mémoire, mais aussi il nous a orienté, dirigé, fourni tous les ingrédients utiles pour notre projet de fin d'études. Un grand Monsieur que nous ne cesserons jamais de le remercier pour sa patience, pour son engagement permanent, sa maitrise totale du thème, son expérience, son soutien constant et la confiance totale qu'il nous a accordé. Nous remercions aussi toute l'équipe de la division **PED**.

Nous tenons également à remercier messieurs les membres de jurys pour l'honneur qu'ils nous ont fait en acceptant de siéger à notre soutenance. Notre gratitude à Mme. **BOUMAHDI Fatima** pour avoir suivi et pris en charge ce travail, tout en la remerciant pour l'intérêt qu'elle a porté à ce mémoire, ainsi que pour ses précieux conseils et remarques.

Finalement, Nous tenons à remercier nos chers parents, qui ont toujours été là pour nous, tous les membres de nos familles pour leur amour, soutien et encouragement durant toutes ces années ainsi que nos frères, sœurs et amis qui ont toujours été là pour nous.

Table des matières

Tables des Figures

Liste des Tableaux

Liste des abréviations

ACID *: Atomicité Cohérence Isolement Durabilité*
AD *:Active Directory*
API *: Application Programming Interface*
BPMN *: Business Process Modeling Notation*
CLUF *: Contrat de Licence Utilisateur Final*
CRUD *:Create Read Update Delete*
CSS *: Cascading Style Sheets*
DMN *: Decision Model and Notation*
DOS *: Disk Operating System*
DRD *: Decision Requirements Diagram*
GPO *:Group Policy Object*
GUI *: Graphical User Interface*
HTML *: Hyper Text Markup Language*
IP *: Internet Protocol*
IT *:Informatique et Technique*
Jdk *: Java Development Kit*
JSON *: JavaScript Object Notation*
LDAP *: Lightweight Directory Access Protocol*
LMGRD *: License Manager Guard*
LMTOOLS *: Licence Manager Tools*
MAC *: Media Access Control, : Media Access Control*
NoSQL *:No Structured Query Language*
NPM *: Node Package Manager*
MLM *: Multi Level Marketing*
OMG *: Object Management Group*
PED *: Petroleum Engineering and Developement*
POM *: Project Object Model*
SGBD *: Système de Gestion de Base de Données*
SGBDR *: Système de Gestion de Base de Données Relationnelle*
SGML *: Standard Generalized Markup Language*
SQL *: Structured Query Language*
SSE *: Server Sent Event*
URL *: Uniform Resource Locator*
XML *: Extensible Markup Language eXtensible Markup Language*

Introduction Génerale

Parmi les importantes missions de la division PED (Petrolium Enginnering & Developement) au sein de la SONATRACH est le suivi des gisement pétroliers et l'élaboration des plans de développement et d'exploitation pétroliers.

1.1 Problématique

Vue la taille de l'entreprise « SONATRACH » qui est constituer de plusieurs divisions, ou chaque division est un groupement de département et que chaque département utilise une multitude de logiciels. Chaque logiciel nécessite au moins une licence pour pouvoir être utiliser, des budgets pharamineux (plusieurs milliards de dinars pour une volumétrie de 38 logiciels qui utilisent 261 licences) sont consacré chaque année pour l'achat ou au renouvellement des licences logicielles avec tout ce que ça engendre comme frais de maintenance et d'inverventions de la part des prestataires.

À date cette opération couteuse ce fait de façon arbitraire ou les besoins sont exprimés par des ordres de services mais il y'a une absence totale de suivi ni de visibilité réelle sur l'utilisation de ces licences, autrement dit il n'existe aucun moyen pour déterminer.

- si une licence est utilisée ou pas ?

- qui utilise quelle licence ?

- combien de temps une telle ou telle licence a été utilisé ?

On ne sait pas si le besoin existe sur le terrain et on se base sur des besoins exprimés verbalement ou par ecrit.

De plus les licences acquises sont gérées de façon manuelle via des commandes fournis par l'environement Flexnet et la gestion de ces dernières nécessite l'apprentissage d'une dizaine de commandes en plus de la ressource humaine qui doit etre formé et consacré à plein temps pour les gérer, l'absence de la ressource peut ralentir le travail et couter enormément d'argent de plus de la rémunération de cette tache.

1.2 L'objectif

Le présent projet s'inscrit dans le cadre de développement d'une solution pour la gestion des licences Flexnet, par la programmation Réactive. Il a pour but le développement de modules et services permettant la récupération des données liées à l'utilisation des Licences Logiciels l'affichage de ces derniers de façon instantané et dynamique via un tableau de bord, les stocker dans une base de données non relationnelle en vue de leurs volumétries pour qu'ils puissent être exploiter lors de la gestion de ces derniers en plus d'une interface agreable simple à utilisé qui nécessite pas une formation particulière. L'exploitation des données aura des fins décisionnelles, en résumé l'application permettra :

- De construire une base de donnes pour avoir des statistiques
- La gestion des licences (attribuer / libérer / affecter)
- La gestion des utilisateurs des licences
- La gestion des serveurs des licences
- Visualiser l'historique et usages des licences
- Confirmer ou infirmer les besoins des licences
- Maitriser les dépenses (optimisation des coûts)

1.3 Structure du mémoire

Le présent rapport est structuré en Cinq chapitres :

- Le premier chapitre présente les concepts théoriques de gestion des licences, l'acquisition et fonctionnement de ces dernières, La définition de l'environnement Flexnet et la présentation des fichiers LOG utilisé pour l'élaboration de notre projet.
- Le deuxième chapitre est consacré aux présentations, et définitions de la programmation réactive, Les threads, les multithreading et une introduction des bases de données non relationnelles (nosql).
- Le troisième chapitre définis les notations standards Decision Modèle Notation (DMN) et Business Process Modeling Notation (BPMN) utilisées pourmodélisation de la solution proposée.
- Quant au quatrième chapitre, il est devisé en deux parties principales :

- La modélisation qui reprend une modélisation métier, une modélisation informationnelle et une modélisation décisionnelle nécessaire à la prise de décision et à la concrétisation des objectifs
- La conception générale qui présente la solution ainsi l'architecture proposée et une conception détaillée des différents modules ou composants de notre architecture.

- Le chapitre Cinq reprend l'implémentation de la solution, les choix techniques et enfin les tests avec la présentation de notre tableau de bord final, nos interfaces et nos résultats.

Pour conclure, ce rapport est clôturé par une synthèse de cette expérience et une illustration des perspectives.

Chapitre 1 : La gestion des Licences

1. Introduction

Dans ce chapitre, Nous présentons le Fonction des licences, les licences FlexNet et les fichers Log.

2. Acquisition et fonctionnement des licences

Une licence de logiciel est un contrat « par lequel le titulaire des droits du logiciel autorise un tiers à poser des gestes qui autrement les enfreindraient. » [1]

Pour avoir le droit d'utiliser un logiciel, il faut que le titulaire déteneur des droits l'autorise. La licence est le document dans lequel il énumère les droits qu'il accorde au licencié (installer le logiciel, l'utiliser, faire une copie de sauvegarde). Utiliser sans licence un logiciel dont on n'est pas l'auteur revient à violer le droit d'auteur. [1]

Souvent, le titulaire des droits ne se contente pas de concéder la licence, il ajoute également des exigences comme l'interdiction d'utiliser le logiciel à plusieurs, d'étudier le logiciel, de publier des mesures de ses performances, etc. Pour le grand public, l'achat d'un logiciel revient en fait à obtenir une licence, puis à accepter le contrat de licence utilisateur final (CLUF). [1]

2.1 Formats des licences

Les licences fournis par les prestataires peuvent être en différents formats comme démontré ci-dessous :

- Dongle : un dongle de protection logicielle (communément appelé dongle ou clé) est un dispositif de protection contre la copie électronique et le contenu. Lorsqu'ils sont connectés à un ordinateur ou à un serveur, ils déverrouillent les fonctionnalités du logiciel ou décodent le contenu. La clé matérielle est programmée avec une clé de produit ou un autre mécanisme de protection cryptographique et fonctionne via un connecteur électrique relié à un bus externe du serveur.

- Licence Électronique : chaque licence constitue un contrat de droit commercial dans lequel l'éditeur intègre ses conditions. L'acheteur est censé les accepter dès lors qu'il installe le logiciel. Dans tous les cas, il conviendrait donc de bien lire chaque licence lors de l'installation et surtout, de s'en rappeler les clauses.

Chez un même éditeur les formes de licences sont nombreuses et évoluent rapidement dans le temps. Vu le nombre de logiciels installés sur une machine, connaître en permanence ses droits sur un parc important relève du casse-tête.

Il est donc primordial de simplifier la situation en définissant des règles communes d'achat de licences pour l'ensemble de l'entreprise et de signer des contrats d'achat cadre avec les fournisseurs.

2.2 Types des licences Electronique

- Licence fixe : est conçue pour être installée sur un ordinateur particulier. Elle peut utiliser une caractéristique spécifique à cet ordinateur, comme son adresse Media Access Control (MAC) pour vérifier et contraindre la conformité de l'usage de la licence. [2]
- Licence nominative : qui a remplacé la licence fixe, a pour but de permettre l'utilisation d'un logiciel sur un ordinateur défini avec une adresse MAC par exemple [2]
- Licence flottante : fonctionne avec un ordinateur serveur de licence(s). Celui-ci décompte le nombre de licences utilisées à un instant « T » sur le réseau. Tant qu'au moins une licence reste disponible, tout ordinateur du réseau réclamant une licence se la verra affecter temporairement durant le temps d'utilisation du logiciel concerné. [2]

- Licences libres : est une forme particulière de licence : les licences libres qui garantissent quatre droits fondamentaux aux utilisateurs :

- Usage de l'œuvre.
- Étude de l'œuvre pour en comprendre le fonctionnement ou l'adapter à ses besoins.
- Modification (amélioration, extension et transformation) ou incorporation de l'œuvre en une œuvre dérivée.
- Redistribution de l'œuvre, c'est-à-dire sa diffusion à d'autres usagers, y compris commercialement. [2]

Dans notre cas nous allons utiliser les licences flottantes, car ces dernières sont fournies par le fournisseur des logiciels suite à une convention signée par la division.

2.3 L'achat d'une licence

Comme mentionnée plus haut la société SONATRACH ne fait pas de production ou de développement informatique, et sous-traite les différents logiciels utilisés dans les divisions de production ci-après les étapes d'achat des licences logiciels :

- Expression des besoins (nombre et type du logiciel) par la division.
- La division envoi un ordre de service au département IT.
- Le service IT transmet l'ODS au prestataire concerné.
- Livraison des licences de la part du prestataire.
- Installation des licences dans les serveurs.
- Utilisation des licences de la part des employés de la division.
- La maintenance est assurée par le prestataire (Mensuelle/trimestrielle) selon le contrat.
- Le renouvellement et la régénération des licences est annuelle (ou selon le besoin).
- Les prestataires peuvent donner des versions d'évaluation de logiciel pour des tests (à des fins commerciales).

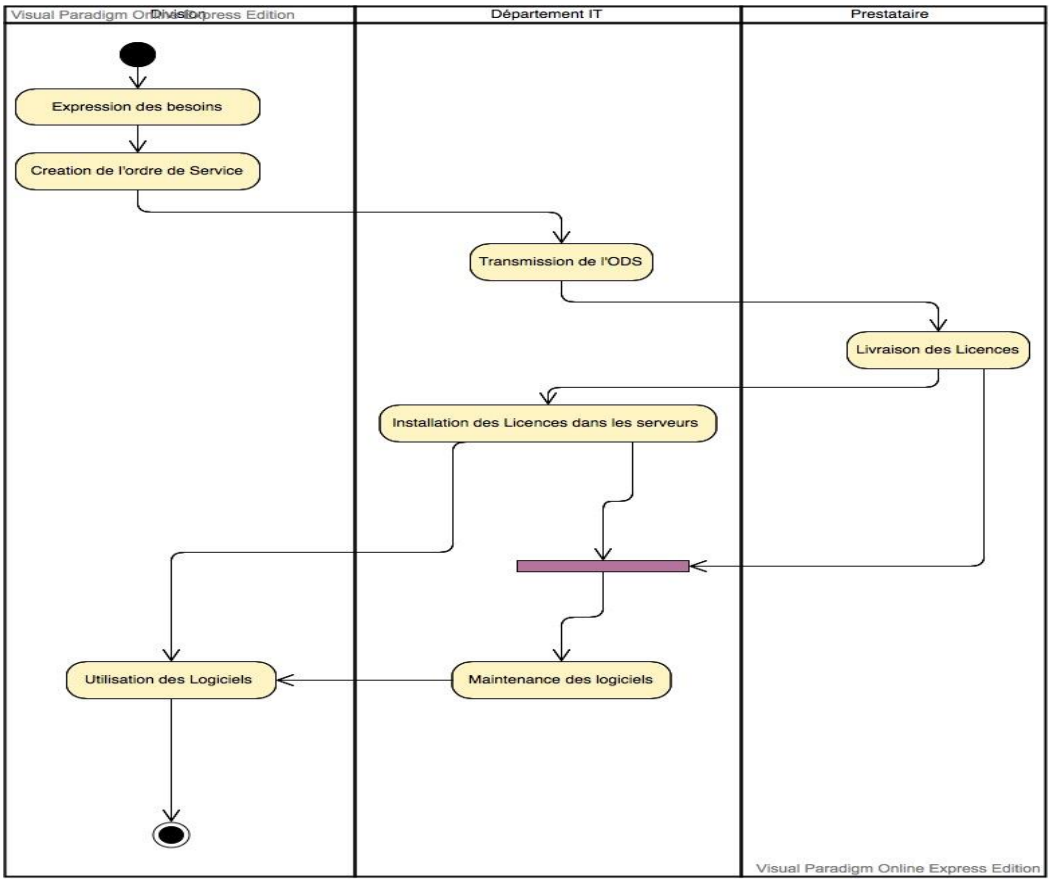


Figure 1- Diagramme d'activité de l'achat d'une licence

2.4 La Convention entre la division et le prestataire

Une convention et signée entre le fournisseur des Logiciels et la division de SONATRACH, elle est le référentiel qui contient tous les articles liés au contrat et à l'achat des licences logiciels ci-dessous quelques articles en relation avec notre projet :

- LOGICIEL : le terme logiciel désigne le programme informatique en sa version production, la plus récente, incluant les correctifs, la documentation qui s'y rapporte et éventuellement tout accessoire nécessaire à son utilisation. [3]

- MODULE LOGICIEL : le terme « module logiciel » désigne un ou ensemble de composants, déjà existants ou qui pourraient être intégrés dans un logiciel, pour fournir des fonctionnalités supplémentaires.

- LICENCE LOGICIEL : le terme « licence logiciel » correspond au droit d'utilisation perpétuel de tout ou partie d'un logiciel dans la version achetée sans limitation dans le temps et selon les termes et conditions correspondants. Une licence comprend un ou plusieurs modules. [3]

- LICENCES COMPLEMENTAIRES SUPPLEMENTAIRES : le terme « licences supplémentaires » désigne les licences additionnelles de logiciels à ajouter et similaires aux licences de logiciel actuellement installées dans les divisions, requises dans le cadre des projets de la gestion d'études de caractérisation et de développement de champs. [3].

- UTILISATEUR : le terme « utilisateur » signifie toute personne faisant partie du Groupe SONATRACH exploitant une licence de logiciel.

- DUREE DE LA CONVENTION : la présente convention est conclue pour Cinq (05) années, à compter de la date d'entrée en vigueur tel que défini à l'article 31.

- TRANSFERT DES LICENCES : dans le cadre de la présente convention, Le prestataire procèdera sans frais au transfert à SONATRACH, des licences déjà acquises et celles à acquérir, par les opérateurs en association avec SONATRACH, et ce après expiration des contrats d'association. [3].

- ENVIRONNEMENT DE FONCTIONNEMENT DES LOGICIELS : SONATRACH assurera un environnement adéquat au bon fonctionnement des équipements hébergeant les logiciels, soit, à titre indicatif mais non exhaustif, l'alimentation électrique, la connexion réseau, la climatisation et la protection d'accès physique. [3]

- LICENCESET MODULES COMPLEMENTAIRES : des licences additionnelles nouvelles ou de licences complémentaires pourront être acquises, par SONATRACH, pour améliorer la performance des solutions existantes, conformément aux conditions des annexes B. [3]
- FLEXIBILITÉ DANS LA GESTION DES LICENCES : a la demande de SONATRACH, le nombre de licence sous maintenance pourra être réduit avec un préavis minimum de 30 jours avant la période de renouvellement.

2.5 Architecture de fonctionnement des licences

Ci-dessous un schéma qui résume l'architecture des licences au sein du département PED (Petrolium Engenieering Division)

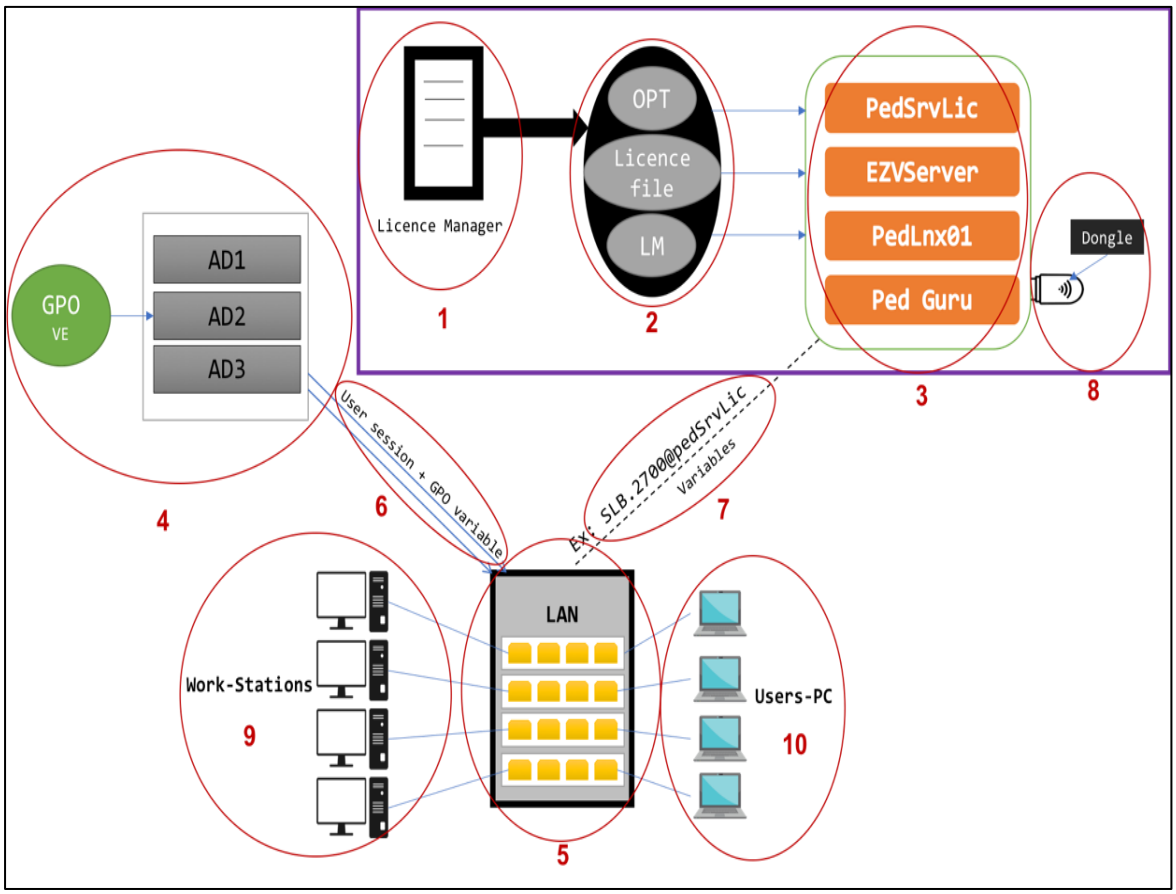


Figure 2 - Architecture de fonctionnement d'une licence

Description de l'architecture actuelle

1/ Licence Manager : Outil de gestion de logiciel utilisé par les éditeurs de logiciels indépendants ou par les organisations d'utilisateurs finaux pour contrôler où et comment les produits logiciels peuvent être exécutés.
2/ Fichier des licences + fichier option + Fichier de LM TOOLS
3/ Différentes licences implémenté dans les serveurs
4/ Active directory (AD) : est la mise en œuvre par Microsoft des services d'annuaire LDAP (Lightweight Directory Access Protocol) 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. Il permet également l'attribution et l'application de stratégies, 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, les imprimantes, etc.
5/ Serveur de Réseau local
6/ Identifiant session Windows + mots de passe de sessions + Variables GPO (Group Policy Object)
7/ Adresse du serveur + Numéro de port
8/ Un dongle (ou sentinelle) est un composant matériel se branchant sur les ordinateurs ou les serveurs, il s'agit d'une licence physique.
9/ Station de travail et Administration réseau
10/ Ordinateurs personnels

3. Les licences FLEXnet

Les licences fournis par les prestataires sont gérées par un outil de gestion des licences open source appelé Flexnet détaillé ci-dessous :

3.1 Définition de FLEXnet

FLEXnet est un outil de gestion de licences populaire sur le marché. Il permet aux logiciels de "flotter" sur un réseau et de ne pas être liés à une machine particulière. Cela implique une relation serveur-client qui nécessite qu'un ordinateur client extrait d'abord avec succès une licence du serveur afin qu'une application puisse être utilisée sur cet ordinateur client. Les critères d'extraction d'une licence peuvent varier en fonction de la configuration de la gestion des licences. [4]

3.2 Fonctionnement de FLEXnet

FLEXnet comprend quatre composants principaux :

- Démon du gestionnaire de licences (lmgrd) : établit le premier contact avec l'application cliente et démarre et redémarre les démons du fournisseur.
- Démon vendeur (MLM) garde une trace du nombre de licences extraites et du détenteur de chaque licence en accédant à la mémoire et en autorisant ou non les extractions de licence.
- Fichier de licence (license.dat) contient les données de licence dans un fichier texte. Il est créé par le fournisseur du logiciel (par exemple, The MathWorks) et modifié lors de l'installation.
- Programme d'application communique avec le démon du fournisseur pour demander une licence d'extraction. [4]

Voici le détail du processus de demande de licence :

Les applications sous licence FLEXnet peuvent être comptées ou non comptées. Compté signifie qu'un gestionnaire de licence est nécessaire pour permettre le contrôle de la licence. Les licences non comptées n'utilisent pas de gestionnaire de licence. La gestion des licences dépend uniquement du contenu du fichier de licence. Lorsque vous exécutez une application comptée sous licence FLEXnet, les événements suivants se produisent [4] :

- Le module de licence de l'application client trouve le fichier de licence, qui comprend le nom d'hôte du nœud du serveur de licences et le numéro de port du serveur avec lequel le démon du gestionnaire de licences, lmgrd, communique.
- Le client établit une connexion avec le démon du gestionnaire de licences sur le serveur (lmgrd) et lui indique le démon du fournisseur avec lequel il doit parler.
- Lmgrd détermine quelle machine et quel port correspond au démon fournisseur principal et renvoie ces informations au client.
- Le client établit une connexion avec le démon fournisseur spécifié et envoie une demande de licence.
- Le démon vendeur vérifie si des licences sont disponibles et envoie un message d'accord ou de refus au client.
- Le module de licence de l'application accorde ou refuse l'utilisation de la fonctionnalité, selon le cas. [4]

Voici un organigramme du processus de demande de licence comportant les étapes ci-dessus intitulées :

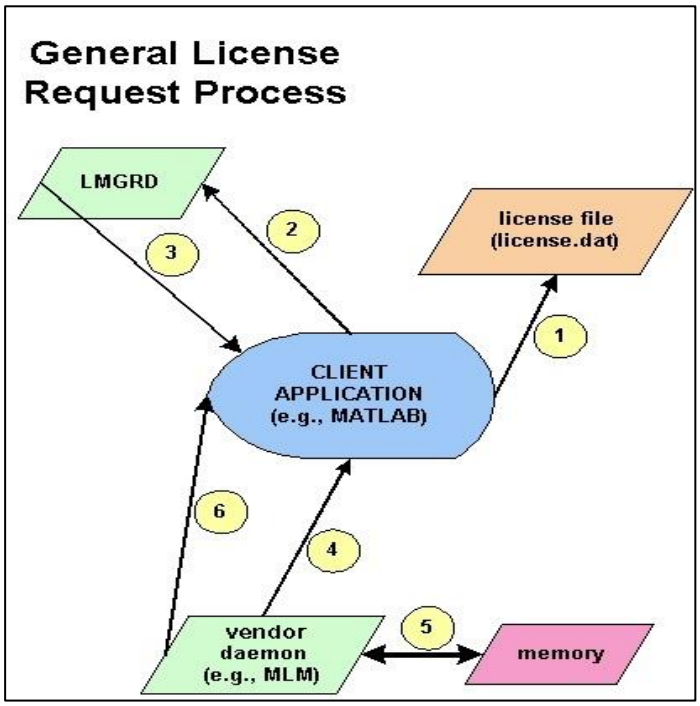


Figure 3 - Fonctionnement de FlexNet

3.3 Liste des utilitaires ou scripts FLEXnet

Flexnet fournit des utilitaires à l'administrateur de licences pour les aider à gérer les activités liées aux licences sur le réseau. Veuillez noter que l'ensemble spécifique d'utilitaires fournis avec une application peut varier d'un fournisseur à l'autre. Vous trouverez ci-dessous un tableau des utilitaires les plus couramment utilisés avec FLEXnet :

Com-mande	Explication
lmdown	Arrête tous les démons de licences (les démons lmgrd et tous les vendeurs) sur le nœud du serveur de licences (ou sur les trois nœuds dans le cas de serveurs redondants à trois serveurs).
lmdebug	(UNIX uniquement) Place l'état de la licence, les informations de débogage et la sortie du journal de débogage de la licence dans un seul fichier à des fins de dépannage
lmhostid	Indique le host id d'un système
lmremove	Libère une licence bloquée dans le pool de licences libres
lmstat	Affiche le statut d'un serveur de licences. (Lorsqu'il est utilisé avec l'indicateur –a, cet utilitaire affiche toutes les fonctionnalités extraites, par quel utilisateur et par quel ordinateur.)
lmstart	Termine les processus existants du gestionnaire de licences et démarre un nouveau processus (redémarrage du gestionnaire de licences).
lmver	Indique la version FLEXnet d'une bibliothèque ou d'un fichier binaire.
Imdiag	Diagnostiquer les problèmes de contrôle de licence

Tableau 1 - Les des commandes FlexNet

Ces commandes sont saisies dans une fenêtre DOS, dans le répertoire racine du logiciel utilisé. Il existe également une interface utilisateur graphique (GUI) pour ces commandes sous Windows appelée LMTOOLS qu'on abordera un peu plus tard dans le mémoire.

4. Fichier LOG

Un fichier log est un fichier comprenant les informations (logs) enregistrées au niveau des serveurs lorsqu'une requête de chargement de fichiers est effectuée lors d'une visite sur un site web. Les fichiers logs comprennent, entre autres et pour chaque requête effectuée, l'adresse IP du demandeur, les fichiers demandés, les heures, des données relatives à l'environnement de navigation…, Les logs bruts sont difficilement interprétables car les données sont volumineuses et disparates (l'affichage d'une page peut donner plusieurs lignes de codes). [5]

4.1 Utilité du fichier LOG

Dans notre projet nous nous basons sur les fichiers logs qui sont généré automatiquement et dynamiquement dans les serveurs des licences, et qui comprennent toutes les informations liées à l'usage des licences, tels que (heures d'usage de licences, nom d'utilisateurs, nom des licences, les erreurs constatées sur les serveurs…etc.)

De ce fait nous allons procéder à un traitement des informations contenus dans le fichier LOG, en passant par un nettoyage de ces derniers, et extraire que les informations liées à l'usage des licences afin de les afficher en temps réel et les stocker dans notre base de données pour des fins décisionnelles.

5. Conclusion

Ce chapitre définit les différents formats et types des licences logicielles, l'aquistion de ces dernières au niveau de la division PED de Sonatrach, l'architecture de la configuration actuelle au niveau de la division. Aussi il présente les concepts fondamentaux liées à l'environement flexnet en expliquant le fonctionnement et la liste des utilitaires utilisés, il finit par la défintion du fichier LOG sur le quel l'application se base pour extraire les informations sur l'utilisation des licences logicielles,

Dans le prochain chapitre nos aborderons le concept de la programmation réactive utilisée lors du développement de notre projet et aussi la notion des bases de données non relationnelles.

Chapitre 2 : La Programation Réactive et les Bases de Données NoSQL

1. Introduction

Ce chapitre présent la programation réactive dans le cadre du développement d'une solution pour la gestion des licences flexnet, Il a pour but le développement de modules et services permettant la récupération les informations liées à l'utilisation des licences logicielles.

2. Programmation Réactive

Avant d'éxpliqué le choix de cette nouvelle technologie pour le développement de l'application, nous allons d'abord définir les principes de la programmation réactive.

2.1 Qu'est-ce que la programmation réactive

La notion de système réactif désigne des systèmes qui maintiennent une interaction permanente avec un environnement donné en particulier, un système réactif doit être en mesure de réagir aux sollicitations successives de son environnement. La famille des langages synchrones [6] regroupe un ensemble de langages de programmation dédiés à la conception de tels systèmes. De manière générale, ces langages permettent de décrire le comportement de composants parallèles qui s'exécutent de manière synchrone, relativement à une horloge logique sur laquelle repose un mécanisme de diffusion instantanée de l'information. Parmi les langages synchrones, on distingue principalement deux paradigmes de programmation : la programmation orientée flots de données [7], la programmation orientée contrôle. [8]

Dans tous les cas, ces langages sont conclus de manière à permettre un ordonnancement statique des composants parallèles, elle est une compilation des programmes vers du code séquentiel, des automates à états finis ou des circuits. En particulier, sous certaines contraintes, on peut assurer que pour toute sollicitation de l'environnement, la réaction du système existe (réactivité) et est unique (déterminisme).

En contrepartie de l'obtention de propriétés aussi fortes, les contraintes imposées sur ces langages limitent leur utilisation à des domaines très spécifiques. Comme exemples de fonctionnalités non supportées par les langages synchrones, on peut citer : l'introduction dynamique de composants parallèles (e.g. par création de threads ou mobilité du code), la mobilité des noms (au sens des algèbres de processus), la définition de fonctions récursives et la manipulation de types de données dynamiques. [9]

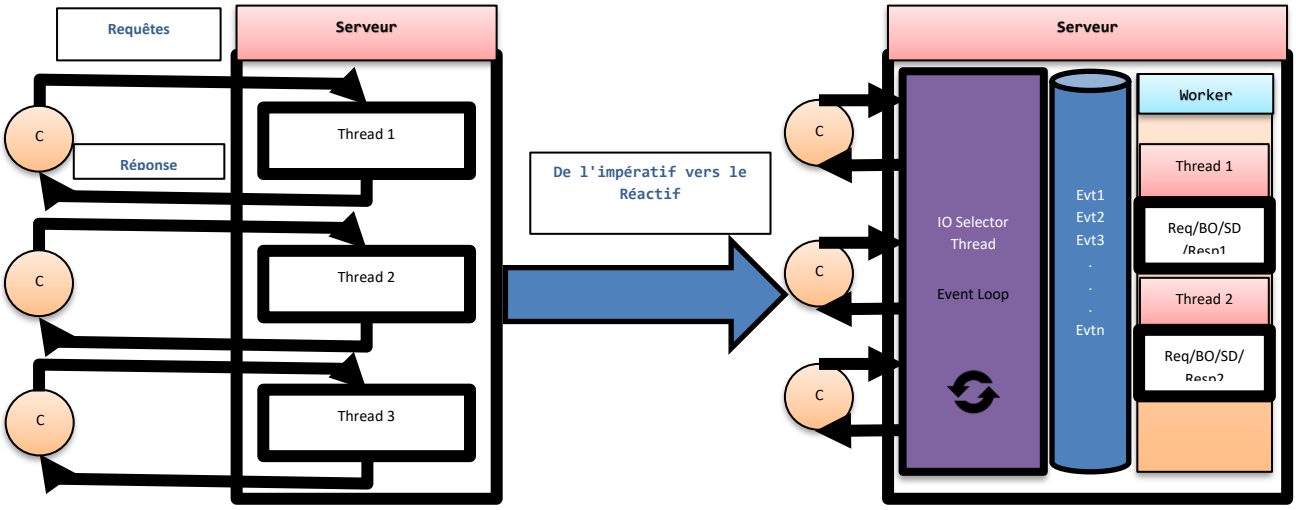


Figure 4 – Passage De l'impératif vers le Réactif

2.2 Les Avantages de la programmation réactive

La plupart des applications contemporaines sont réactives, c'est-à-dire qu'elles répondent, en calculant, à des événements qui leur parviennent [10].

Cela permet aux utilisateurs d'avoir une meilleure interaction avec le système, une réponse beaucoup plus rapidement et donc une satisfaction [11] .

La réactivité est désormais présente partout, et principalement dans les interfaces graphiques [10].

Les applications réactives sont importantes aujourd'hui car elles doivent être :

- Disponibles : Le système doit répondre rapidement quoi qu'il arrive.
- Résilientes : Le système doit toujours rester disponible, même en cas d'erreur.
- Souples : Le système doit continuer à vivre même s'il est surchargé.
- Orientées messages : Le système utilise des messages asynchrones.

Ce qui correspond aux exigences des utilisateurs d'aujourd'hui [10].

2.3 Les principes de la programmation réactive

Il s'agit d'un paradigme de programmation différent du paradigme le plus courant « la programmation impérative ». Un paradigme de programmation est un style de programmation, la manière de traiter les problèmes informatiques. La programmation réactive est asynchrone, non bloquante et fonctionnelle, en particulier durant les interactions avec les ressources externes :

- Responsive

‣Réponse en temps voulu, si possible

‣Temps de réponses rapides et fiables (limites hautes)

- Résilient

‣Résiste à l'échec

‣Principes: Réplication, conteneurs, isolement, délégation

On fait en sorte qu'un échec n'impacte qu'un seul composant

- Élastique

‣Le système reste réactif en cas de variation de la charge de travail.

‣Pas de point central

‣Pas de goulot

‣Distribution des entrés entre composants

- Message Driven

‣ Passage de messages asynchrones

‣ Couplage faible, isolation

‣ Pas de blocage, les composants consomment les ressources quand ils peuvent

- Deux façons de gérer les données

‣ Donnés qui changent (mutable) : on utilise un état (state)

‣ Donnés qui ne changent pas (immutable) : on utilise des propriétés (props)

‣ On essaie de minimiser les données qui changent quitte à refaire des calculs [12]

2.4 Pourquoi la programmation réactive ?

Le choix porte sur la programmation « Réactive », afin de rendre le résultat des traitements disponible via les WebFlux. Autrement dit, la donnée sera générée et disponible en continu sur un serveur web via un flux de données continuel, indépendamment de l'existence d'un client sur le flux (ie : sur le lien de communication).

Un traitement en background doit être développé pour récupérer les données provenant des serveurs de licences et devrait être le plus performant possible. À cet effet, les threads seront utilisés pour palier à cette contrainte.

2.5 Serveur Réactive SSE (Server Sent Event)

Server-sent events est une norme décrivant comment les serveurs peuvent initier la transmission de données vers les clients une fois que la connexion initiale du client a été mise en place. Cette technologie est couramment utilisée pour envoyer des mises à jour de messages ou de flux de données en continu à un navigateur client. Elle a été conçue pour améliorer nativement le support du streaming de données multi-navigateurs à l'aide d'une API JavaScript appelé EventSource, par laquelle un client demande une URL particulière afin de recevoir un flux d'événements. [13]

2.6 Web Flux

Un flux Web (ou « news feed ») est une technique de l'Internet pour transmettre des nouvelles ou des changements récents. Les distributeurs de contenus, grâce à la syndication de contenu, proposent un canal de transmission auquel peuvent s'abonner des internautes (on parle alors de « syndicated feed »). Les données transmises généralement en langage XML, se limitent le plus souvent à quelques phrases suivies d'un hyperlien vers le site Web émetteur. Les formats de flux Web (Atom, RSS) n'étant pas directement lisibles par l'utilisateur, ils nécessitent l'usage d'un logiciel dédié comme un agrégateur, permettant de regrouper plusieurs flux, ou d'un navigateur web compatible.

3. Les Threads

Un thread ou fil (d'exécution) ou tâche est similaire à un processus car tous deux représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur. Du point de vue de l'utilisateur, ces exécutions semblent se dérouler en parallèle. Toutefois, là où chaque processus possède sa propre mémoire virtuelle, les threads d'un même processus se partagent sa mémoire virtuelle. Par contre, tous les threads possèdent leur propre pile d'exécution. [14]

3.1 Le multithreading

Il existe différents moyens d'exécuter plusieurs threads sur un même cœur de processeur, selon la façon dont leurs temps d'exécution respectifs sont répartis.

Les threads peuvent s'exécuter à des moments différents (temporal multithreading), soit en changeant de thread à chaque cycle d'horloge (interleaved multithreading), soit lors d'un événement comme un défaut de cache (block multithreading). Ils peuvent aussi s'exécuter simultanément en partageant les unités d'exécution d'un processeur super scalaire (simultaneous multithreading). [15]

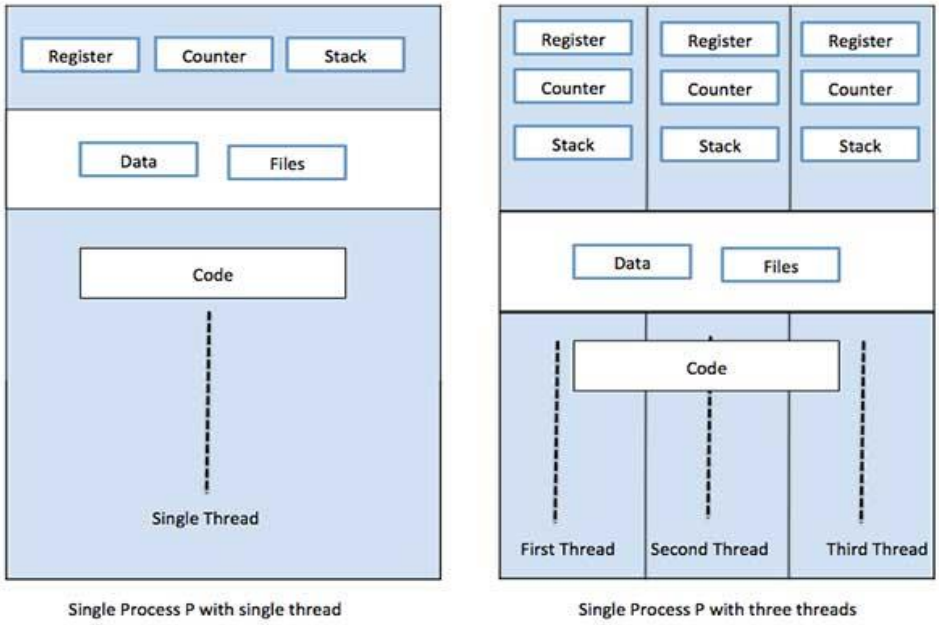


Figure 5 – Multithreading

4. Les base de donnes NoSql

Les évolutions logicielles suivent assez naturellement les évolutions matérielles. Les premiers SGBD étaient construits autour de mainframes et dépendaient des capacités de stockage de l'époque. Le succès du modèle relationnel est dû non seulement aux qualités du modèle lui-même mais aussi aux optimisations de stockage que permet la réduction de la redondance des donnés. [16]

4.1 Qu'est-ce qu'une base de données NoSQL ?

NoSQL est une approche de la conception de base de données pouvant prendre en charge une grande variété de modèles de données, qui compris les formats clé-valeur, document, en colonnes et graphique. NoSQL, constitue une alternative aux bases de données relationnelles traditionnelles dans

les quelles les données sont placées dans des tables et dont le schéma de données est soigneusement conçu avant la construction de la base de données. Les bases de données NoSQL sont particulièrement utiles pour travailler avec des grands ensembles de données distribuées. [17] Dans cette partie, nous allons aborder les caractéristiques générales des moteurs NoSQL, conceptuellement et techniquement, en regard des bases de données relationnelles, mais aussi indépendamment de cette référence.

4.2 Types de base de données NoSQL

Il existe plusieurs types et architectures de bases de données NoSQL, cependant, elles ont toutes un point commun, c'est l'abandon des contraintes ACID (Atomicité Cohérence Isolement Durabilité). Ces bases stockent soit des paires clé-valeur soit des documents JSON (JavaScript Object Notation). Elles ne possèdent pas de schémas et de contraintes sur les objets. Le développeur est libre de stocker et organiser les données dans la base comme il veut. C'est lui qui définit au travers de son application la structure et les règles des éléments qui y seront manipulés et sauvés. [18]

Les quatre types de bases de données NoSQL sont présentés dans ce qui suit :

- Les entrepôts clé-valeur

Les bases de données NoSQL de type clé / valeur s'articulent sur une architecture très basique. On peut les apparenter à une sorte de HashMap, c'est-à-dire qu'une valeur, un nombre ou du texte est stocké grâce à une clé, qui sera le seul moyen d'y accéder. Leurs fonctionnalités sont tout autant basiques, car elles ne contiennent que les commandes élémentaires du CRUD (Create, Read, Update, Delete), cela permet aux données totalement hétérogènes entre elles d'être stockées. Ces modèles peuvent assurer une performance élevée en lecture/écriture.
Redis et Riak sont les principales bases de données de type entrepôt clé-valeur. [19]

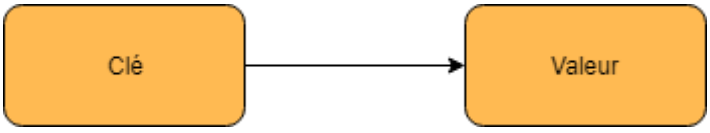


Figure 6 - Modèle clé-valeur [19]

- Bases orientées documents

Les bases de données orientées documents se rapprochent du modèle clé-valeur. La valeur étant cette fois caractérisée par un document dans un format hiérarchique semi-structuré dont la structure est libre. Il est identifié par une clé unique. Ce type de modèle permet de stocker des données

non-planes. Ce qui signifie des relations avec des jointures de type One to Many. C'est particulièrement approprié pour les applications web. On retrouve principalement MongoDB et CoucheBase comme solutions basées sur le concept de base documentaire. [18]

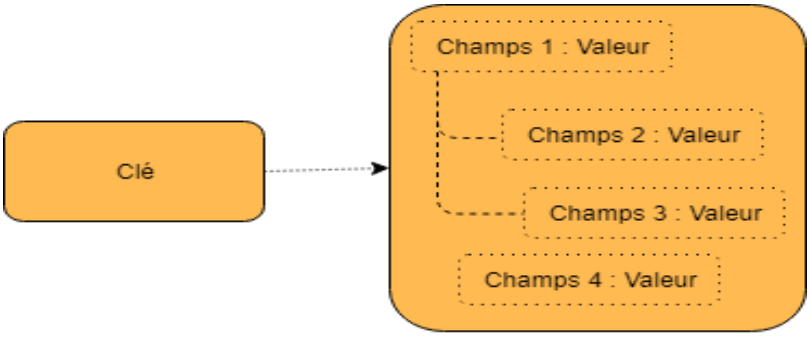


Figure 7 - Modèle orienté document [18]

- Bases orientées colonnes

Les bases de données orientées colonne ont été conçues par les géants du web afin de faire face à la gestion et au traitement de gros volumes de données. Elles intègrent souvent un système de requêtes minimalistes proche du SQL.

Bien qu'elles soient abondamment utilisées, il n'existe pas encore de méthode officielle ni de règles définies pour qu'une base de données orientée colonne soit qualifiée de qualité ou non.

Le principe d'une base de données colonne consiste dans leur stockage par colonne et non par ligne, les données sont organisées de façon à ce que toutes les données d'une même colonne soient stockées ensemble. Ces bases peuvent évoluer avec le temps, que ce soit en nombre de lignes ou en nombre de colonnes. Autrement dit, et contrairement à une base de données relationnelle où les colonnes sont statiques et présentes pour chaque ligne, celles des bases de données orientées colonne sont dite dynamiques et présentes donc uniquement en cas de nécessité. De plus il n'y a pas de stockage de valeur « null » [20]

Les solutions les plus connues se basant sur ce concept sont HBase et Cassandra.

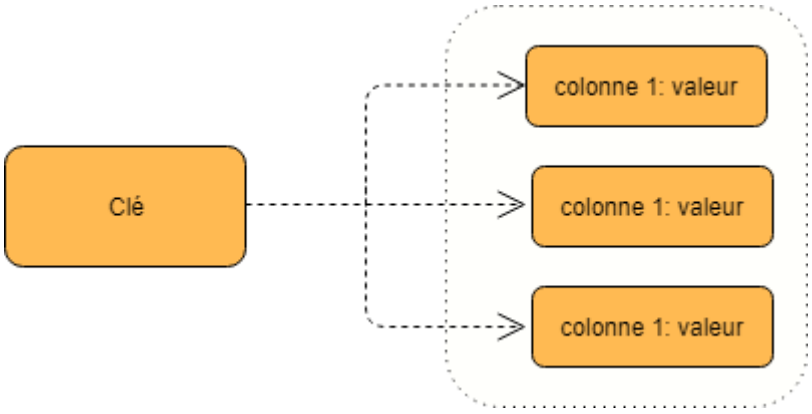


Figure 8 - Bases orientées colonnes [20]

- Bases orientées graphes

Les bases de données NoSQL orientées graphes cherchent avant tout à répondre à des problèmes complexes voire impossibles pour des SGBDR. Elles offrent une réponse adéquate à un modèle de données très volumineux et fortement connecté [18]

Les réseaux sociaux (Facebook, Twitter, etc..), où des millions d'utilisateurs sont reliés de différentes manières, constituent un bon exemple : amis, fans, famille etc. Le défi ici n'est pas le nombre d'éléments à gérer, mais le nombre de relations qu'il peut y avoir entre tous ces éléments. Evidement et comme son nom l'indique, ces bases de données reposent sur la théorie des graphes. [18]

Ce type de base de données NoSQL est le moins utilisé. Il répond à une problématique très spécifique qui intéresse encore peut les entreprises (hormis quelques grands acteurs des réseaux sociaux).

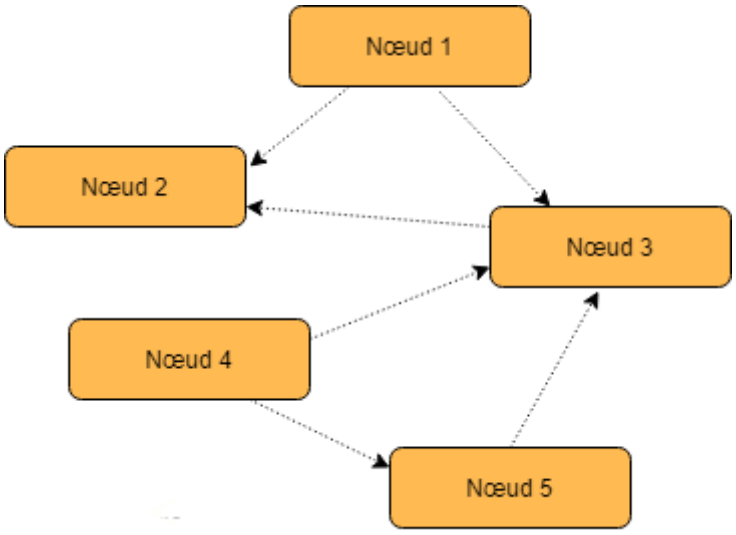


Figure 9 - Bases orientées graphes [18]

4.3 Avantages du NoSQL

- Plus évolutif

NoSQL est plus évolutif. C'est en effet l'élasticité de ses bases de données NoSQL qui le rend si bien adapté au traitement de gros volumes de données. [21]

- Plus flexible

N'étant pas enfermée dans un seul et unique modèle de données, les applications NoSQL peuvent donc stocker des données sous n'importe quel format ou structure, et changer de format en production. En fin de compte, cela équivaut à un gain de temps considérable et à une meilleure fiabilité. [21]

- Plus économique

Les serveurs destinés aux bases de données NoSQL sont généralement bon marché, de plus, la très grande majorité des solutions NoSQL sont Open Source, ce qui reflète une économie importante sur le prix des licences. [21]

- Plus simple

Les bases de données NoSQL ne sont pas forcément moins complexes que les bases relationnelles, mais elles sont beaucoup plus simples à déployer. La façon dont elles ont été conçues permet une gestion beaucoup plus légère. [21]

4.4 Inconvénients du NoSQL

Les bases de données NoSQL étant open source ont un support différent de celui offert par les sociétés commerciales à leurs produits. Il n'y a pas de relation entre les tables comme le SQL. [22]

Manque d'expérience : La nouveauté NoSQL signifie qu'il n'ya pas beaucoup de développeurs et d'administrateurs qui connaissent la technologie, ce qui empêche les entreprises de trouver des personnes disposant des connaissances techniques appropriées. Au contraire, le monde SGBDR compte des milliers de personnes hautement qualifiées. [22]

4.5 Comparaison entre les bases de données SQL et NoSQL

Souvent, lorsque l'on souhaite comprendre le fonctionnement d'un procédé, on se retrouve face à un tableau de données du type ci-dessous. Dans ce tableau, on souhaite voir quelle est la différence entre le SQL et NoSQL :

	SQL	**NoSQL**
Stockage de données	Stocké dans un modèle relationnel, avec des lignes et des colonnes. Les lignes contiennent toutes les informations relatives à une entrée / entité spécifique et les colonnes représentent tous les points de données distincts ; Par exemple, vous pouvez avoir une ligne sur une voiture spécifique, dans laquelle les colonnes sont "Marque", "Modèle", "Couleur", etc.	Le terme « NoSQL » englobe une multitude de bases de données, chacune avec des modèles de stockage de données différents. Les principaux sont : document, graphique, valeur-clé et colonne
Schémas et Flexibilité	Chaque enregistrement est conforme au schéma fixe, ce qui signifie que les colonnes doivent être définies et verrouillées avant la saisie des données et que chaque ligne doit contenir des données pour chaque colonne. Cela peut être modifié, mais cela implique de modifier toute la base de données et de passer hors ligne.	Les schémas sont dynamiques. Des informations peuvent être ajoutées à la volée et chaque 'ligne' (ou l'équivalent) ne doit pas nécessairement contenir de données pour chaque 'colonne'.
Scalabilité	La mise à l'échelle est verticale. Essentiellement, plus de données signifie un serveur plus gros, ce qui peut coûter très cher. Il est possible de faire évoluer un SGBDR sur plusieurs serveurs, mais il s'agit d'un processus difficile et fastidieux	La mise à l'échelle est horizontale, ce qui signifie sur tous les serveurs. Ces serveurs multiples peuvent constituer du matériel bon marché ou des instances cloud, ce qui le rend beaucoup plus économique que la mise à l'échelle verticale.
Conformité ACID (Atomicité, Cohérence, Isolement, Durabilité)	La grande majorité des bases de données relationnelles sont compatibles ACID.	Varie entre les technologies, mais de nombreuses solutions NoSQL sacrifient la conformité ACID pour la performance et la scalabitlité

Tableau 2 - Compraison entre SQL et NoSQL [23]

4.6 Environnement de développement des bases de données nosql

Dans le but de mieux comprendre le tableau ci-dessous présente quelques environnements de développement des bases de données nosql :

Bases	Année de lancement	Schéma de données	Editeur / prestataire de support	Positionnement
Cassandra	2008	Orienté colonnes	DataSax (ex Riptano)	Adoptée par les géants du web et les start-up, Cassandra permet de gérer de gros volumes de données.
Couchbase	2010	Orienté documents	Couchbase	A la différence de MongoDB, Couchbase dispose d'un outil de requêtage normalisé SQL qui facilite sa prise en mains par des développeurs rompus aux bases SQL.
Elasticsearch	2004	Index inversé	Elasticsearch	Connu avant tout pour son moteur de recherche distribué, Elasticsearch tire sa force de l'indexation et de l'analyse des données.
HBase	2006	Orienté colonnes	Hortonworks	Souvent comparé à Cassandra, HBase joue la carte de la très forte volumétrie. Un produit complexe qui exige un gros travail de structuration.
MongoDB	2007	Orienté documents	MongoDB	Base NoSQL la plus populaire, MongoDB est saluée pour la souplesse de sa structure et sa capacité à répondre à un grand nombre de besoins.
Redis	2009	Clé-valeur	Redis Labs	Base de données en mémoire, Redis privilégie la vitesse d'exécution. En contrepartie, ses capacités de requêtage sont limitées.

Tableau 3 - Les différentes Technologies Existantes pour le NoSQL [24]

Dans le tableau précèdent on voit les défèrent logiciel pour chaque ligne et Schéma de données et Editeur nous nous interessons à MongoDB car il est dynamique comme il contient quelques propriétés de SQL que nous aborderont en détails dans le dernier chapitre de l'implémentation

5. Conclusion

Ce chapitre présente les concepts fondamentaux, et aussi le choix de cette technologie car suite à la volumétrie de données importante qui sera générée par le traitement en background lors de l'exécution de notre programme réactif (séquentiel) sur plusieurs serveurs en parallèle, ces données nécessitent d'être stocker pour être la base de l'élaboration des statistiques d'utilisation des licences, aussi pour les prises de décisions qu'on abordera lors d'un prochain chapitre.

Chapitre 3: Les langages de Modelisation

1. Introduction

Ce chapitre presente les langages de modélisation utilisés pour illustrer notre proposition.

Notre archetecture comporte une partie importente d'aide a la décision, qui sera modélisée avec la notation DMN (Decision Model and Notation), cette derniére se base sur la notation BPMN (Business Process Modeling Notation).

En effet, dans ce chapitre Nous présentons la modélisation décisionnelle avec notation DMN et puis la modélisation métier avec le BPMN.

2. DMN (Decision Model and Notation)

Une nouvelle notation nommée DMN (Decision Model and Notation) est venue compléter le bouquet de standards proposés par l'OMG (Object Management Group) dans le cadre de la modélisation métier. [25]

Cette notation DMN permet de modéliser les prises de décision et les règles métier.

Son objectif principal est de fournir une représentation de la prise de décision compréhensible par toutes les parties prenantes (acteurs métier qui gèrent et pilotent les décisions, analystes métiers qui spécifient les exigences relatives aux prises de décisions, développeurs responsables de l'automatisation de tout ou partie de la prise de décision). [25]

2.1 Portée et utilisations de DMN

La modélisation des décisions est réalisée par des analystes métier afin de comprendre et de définir les décisions prises dans une entreprise. [26]

Trois utilisations de DMN peuvent être discernées [26]:

1. Pour modéliser la prise de décision humaine.

2. Pour modéliser les exigences pour la prise de décision automatisée.

3. Pour la mise en œuvre de la prise de décision automatisée.

2.2 Les éléments graphiques principaux

Le niveau DRD (Decision Requirements Diagram) de DMN represente les exigences d'une prise différents élément graphiques qu'utilisés dans DRD.

Un DRD contient les éléments suivants :

– *Input Data* (Donnée d'entrée, au moins une, généralement plusieurs).

– *Decision* (Décision, au moins une, la Décision est en fait la donnée de sortie).

– *Knowledge Source* (Source de connaissance, un expert ou un document de référence qui fait autorité dans le métier concerné).

– *Business Knowledge Model* (Modèle de connaissance métier, contenant une logique de décision, qui peut être réutilisé dans plusieurs diagrammes DRD). [25]

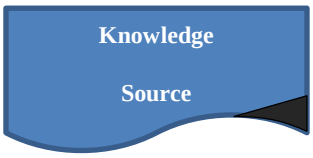
élément DMN	Description
	Un élément de données en entrée désigne les informations utilisées comme entrée par une ou plusieurs décisions. Lorsqu'il est inclus dans un modèle de connaissance, il désigne les paramètres du modèle de connaissance
	Une décision désigne l'acte consistant à déterminer une sortie à partir de plusieurs entrées, en utilisant une logique de décision pouvant faire référence à un ou plusieurs modules de connaissance métier
	Une source de connaissance désigne une autorité pour un modèle de connaissance métier ou une décision
	Un modèle de connaissance métier désigne une fonction encapsulant une connaissance métier, par ex. sous forme de règles métier, de table de décision ou de modèle analytique.

Tableau 4 - Les 4 éléments graphiques de DRD de DMN [27]

Il existe trois types de liens différents, les « *Requirements* », permettent de représenter les relations entre ces quatre éléments graphiques principaux :

– Information Requirement, trait plein terminé par une pointe, qui relie :

- Soit une Donnée d'entrée vers une Décision.

- Soit une Décision secondaire vers une Décision principale.

– Authority Requirement, trait pointillé terminé par un point, qui symbolise l'invocation d'une Source de connaissance.

– Knowledge Requirement, trait pointillé terminé par une pointe, qui symbolise l'invocation d'un Modèle de connaissance métier.

Tous ces composants graphiques du DMN (les quatre éléments principaux et les trois liens différents) sont utilisés dans le diagramme DRD de la Figure 11. [25]

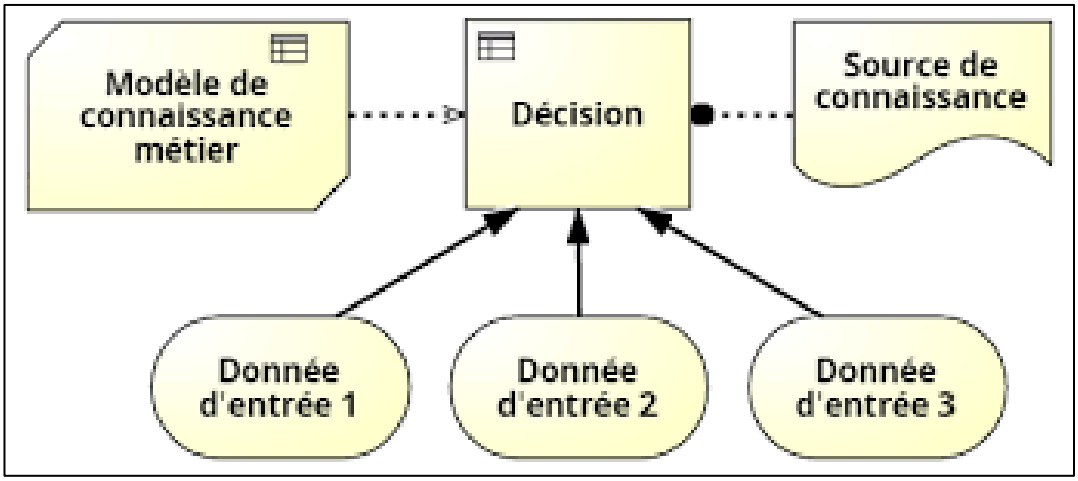


Figure 10 - les composants graphiques de DMN dans un diagramme DRD [25]

2.3 Tables de décision

La notation graphique relativement simple du DRD ne permet pas de représenter la logique de décision elle-même. Celle-ci est représentée généralement (surtout lorsque les critères sont relativement nombreux) par une table de décision [28].

Le formalisme choisi dans la spécification DMN pour représenter les tables de décision, qui datent de quelques décennies [29]. La formalisation des règles métier est une spécialité à part entière.

Dans les tables de décision dites horizontales (représentation recommandée), les règles métier sont listées sous forme de lignes. Les données d'entrée, qui sont donc les critères de décision, sont listées sous forme de colonnes, ainsi que la donnée en sortie (résultat de la décision).

Il doit y avoir une séparation claire entre les conditions (données en entrée) et les actions (décisions en sortie) [28]. On peut préciser les types de données (nombre, booléen, etc.) et leurs domaines de valeur respectifs (min., max.).

La formalisation des règles métier est une spécialité à part entière.

Les outils spécialisés dans la notation DMN - une vingtaine sont déjà disponibles. [30] permettent de vérifier automatiquement la complétude et la cohérence des tables de décision, ce qui est un gage de qualité

Décision				
U	Entrées			Sortie
	Donnée 1	Donnée 2	Donnée 3	Décision
1	Valeur 1	Valeur 11	Valeur 30	Valeur A
2	Valeur 2	Valeur 12	Valeur 32	Valeur B
3	Valeur 3	Valeur 13	Valeur 38	Valeur C

Tableau 5 - Tableau des decisions [25]

La table de décision est un format de connaissances métier, spécifiquement pris en charge par DMN. Tel que le modèle de connaissance de l'entreprise peut être noté à l'aide d'une table de décision.

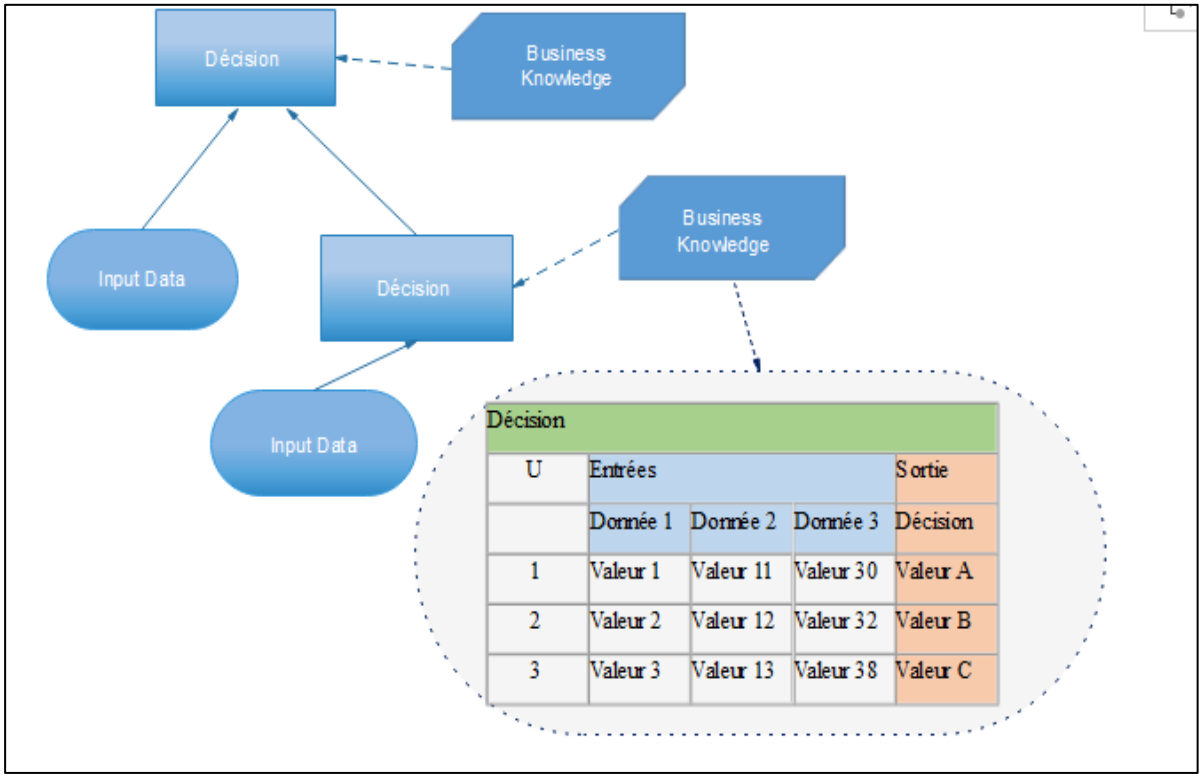


Figure 11 - DMN et la table de décision

3. BPMN (Business Process Modeling Notation)

BPMN est un acronym pour Business Process Modeling Notation. Il est défini comme « une notation graphique qui illustre les étapes d'un processus métiers. »

Business Process Modeling Notation a été développée par la Business Process Management Initiative (BPMI), et est maintenant maintenue par l'Object Management Group (OMG). Le but principal de BPMN est de fournir une notation qui soit réellement compréhensible par tous les utilisateurs. [31]

L'objectif de BPMN : de fournir une notation qui est facilement compréhensible par tous les utilisateurs professionnels, des analystes métier qui créent la version initiale du processus, aux développeurs techniques chargés de l'application de la technologie qui va exécuter ces processus, et finalement, les personnes qui permettront de gérer et de contrôler ces processus. Ainsi, BPMN crée un pont standardisé pour l'écart entre la conception des processus d'affaires et l'implémentation des processus [32].

3.1 Les éléments de la notation BPMN

L'utilisation d'une notation réduira et enlèvera n'importe quelle confusion entre les différents utilisateurs à tous les niveaux, pour cela, BPMN utilise une riche bibliothèque de symboles qui couvrent tous les détails de processus métiers.

3.2 Les principaux éléments graphiques de BPMN

Les différents éléments sont organisés en quatre catégories [33] :

1. Objets de flux
2. Objets de connexion
3. Swimlanes (Couloirs de nage)
4. Artefacts

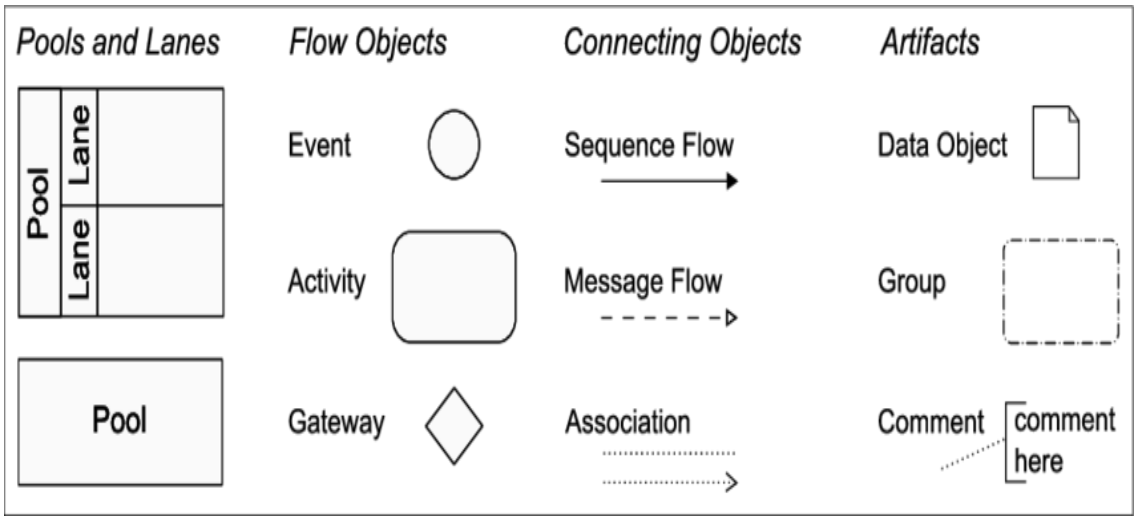


Figure 12 - Les éléments du BPMN [33]

3.2.1 Objets de flux

3.2.1.1 Les évènements

Un événement est représenté par un cercle. Les évènements servent à identifier un état particulier dans le processus. Il représente quelque chose qui survient, il déclenche alors une action ou le résultat d'une action. Il n'effectue aucune tâche. Les évènements peuvent être de trois types :

- début

- intermédiaire

- fin

Chaque événement représente quelque chose qui arrive pendant le processus métiers. Ils peuvent affecter le flux du processus et ont généralement une cause ou une conséquence.

Les évènements de début et de fin doivent toujours être présents sur un processus BPMN.

De même, lorsqu'une tâche débouche sur une exception, l'évènement de fin doit aussi être représenté pour le cas où la tâche se déroule normalement [33].

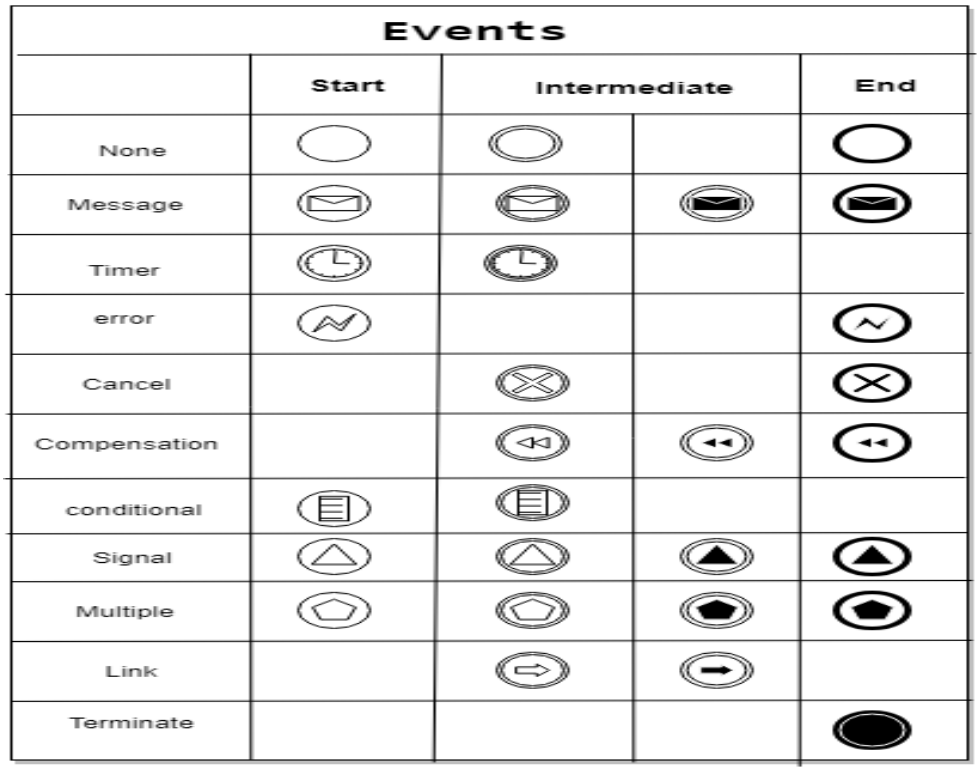


Tableau 6 - Les évènements du BPMN [33]

3.2.1.2 Les activités

Une activité peut être un processus, un sous-processus ou une tâche. Elle est représentée par un rectangle aux coins arrondis [33].

Une tâche est un élément indivisible. Elle représente une action. Chaque tâche a un début et une fin, par conséquent une tâche ne peut débuter que si la tâche précédente est terminée.

Les différents types d'activités sont les suivants : tâche, sous-processus, tâche en boucle et sous-processus tâche en boucle. Voir la Tableau 7 [33].

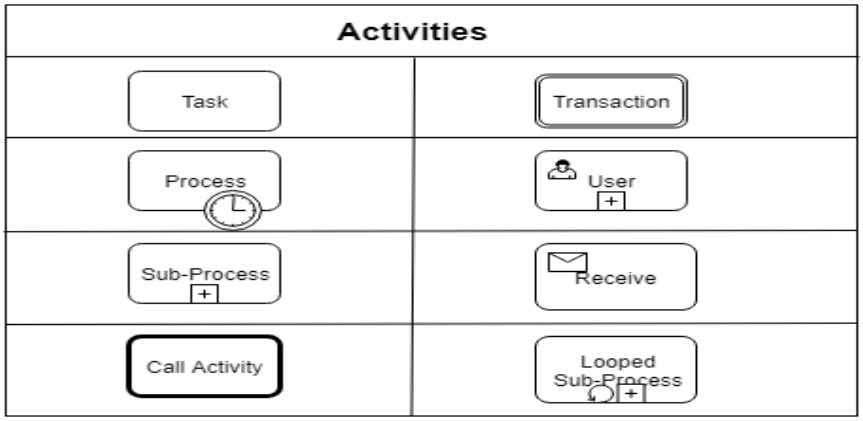


Tableau 7 - Les Activités en BPMN [33]

3.2.1.3 Les objets de connexion

Les objets de connexion sont utilisés pour connecter des objets de flux au sein d'un diagramme. [33]

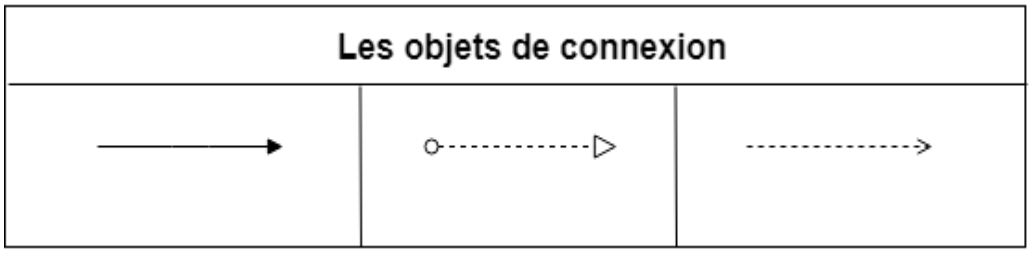


Figure 13 - Les objets de connexion en BPMN [33]

3.2.1.4 Les branchements

Le branchement est un objet essentiel dans la norme BPMN. Il sert à représenter la condition de routage entre les flux d'entrée et les flux de sortie. Il est représenté comme une forme de losange. Le symbole à l'intérieur du losange sert à identifier le comportement du branchement.

Les branchements sont autant utilisés pour diviser un flux en plusieurs flux que pour réunir plusieurs flux en un seul. Le branchement n'est pas une tâche et n'effectue aucune action.

Lorsque le losange est vide, chaque sortie est une alternative et il n'y a pas de différenciation entre les sorties. Le losange vide est utilisé lorsque que l'on ne veut pas compliquer une vue ou bien lorsque la règle de traitement n'est pas connue.

Il est donc possible de définir un comportement de traitement précis. [33]

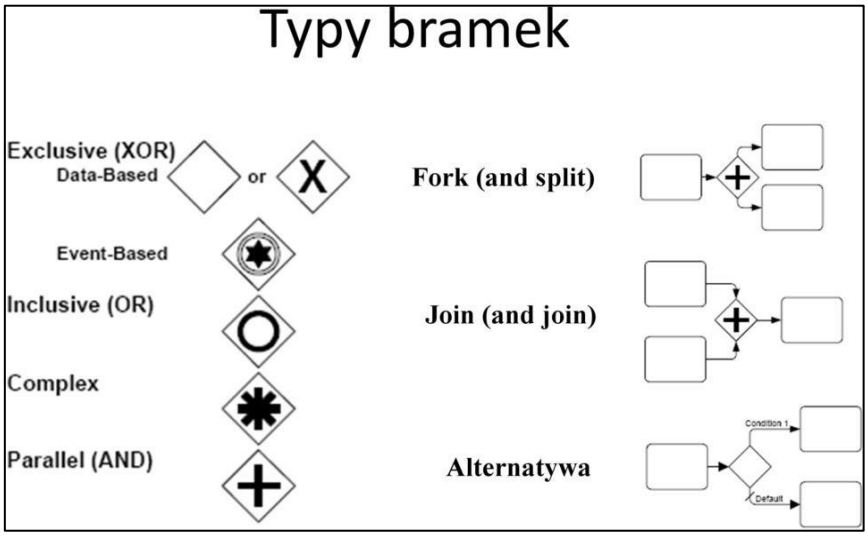


Figure 14 - Les branchements en BPMN [33]

3.2.4.5 Les couloirs

Les « couloirs de nage » sont utilisés pour organiser et regrouper les éléments BPMN en catégories visuelles, montrant des fonctionnalités ou des responsabilités différentes.

Les deux types d'objets sont les suivants :

- **Piscine** : Elle représente un participant à un processus (département, service), il est habituellement utilisé dans des situations B2B ou interservices. Une piscine peut contenir plusieurs couloirs (correspondant à des rôles : manager, assistant, collaborateur).
- **Couloir** : C'est une partition en sous ensemble dans une piscine. Ils sont utilisés pour organiser et classer les activités. Pour illustrer les échanges entre 2 couloirs on utilise des flux de messages. [33]

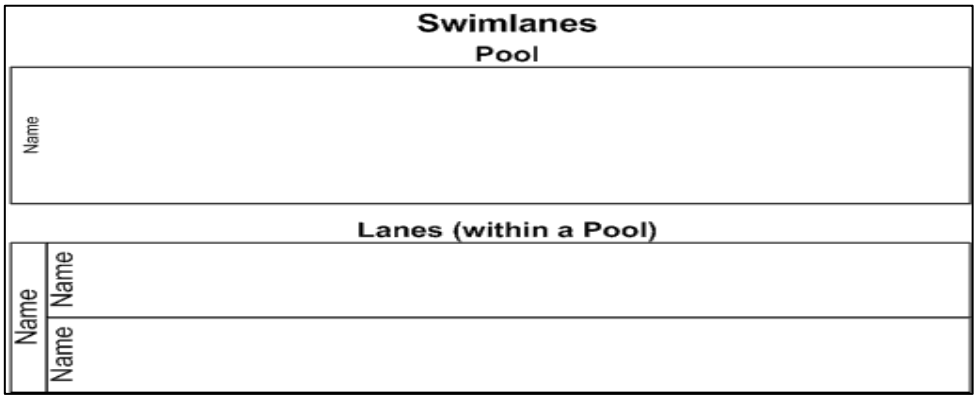


Figure 15 - Les couloirs en BPMN [33]

4. Conclusion

Ce chapitre rentre le cadre de la modélisation de la solution dans le but de relier les tâches des décideurs aux sources des données, nous avons opter à l'utilisation du DMN et BPMN pour faire une analyse du système et faciliter l'échange d'idées il est plus judicieux de passer par la norme DMN et BPMN pour construir la solution proposée,qui sera abordée dans le prochain chapitre.

Chapitre 4 : La Solution proposée

1. Introduction

Il existe différents types de cycles de développement dans la réalisation d'un logiciel dont toutes les étapes de la conception sont prises en comptes.

Dans notre projet nous avons opté pour le modèle en cascade (en anglais : waterfall model), un modèle de gestion linéaire qui divise les processus de développement en phases de projet successives. Contrairement aux modèles itératifs, chaque phase est effectuée une seule fois. Les sorties de chaque phase antérieure sont intégrées comme entrées de la phase suivante. Le modèle en cascade est principalement utilisé dans le développement des logiciels.

En pratique, plusieurs versions du modèle en cascade sont utilisées. Les modèles les plus courants divisent les processus de développement en cinq phases. Les phases 1, 2 et 3 définies par Royce sont parfois regroupées en une seule et même phase, qualifiée d'analyse des besoins [34].

1. Analyse : planification, analyse et spécification des besoins
2. Conception : conception et spécification du système
3. Implémentation : programmation et tests des modules
4. Test : intégration du système, tests du système et de l'intégration
5. Exploitation: livraison, maintenance, amélioration

Le schéma suivant illustre pourquoi le modèle de gestion linéaire est qualifié de « modèle en cascade ».

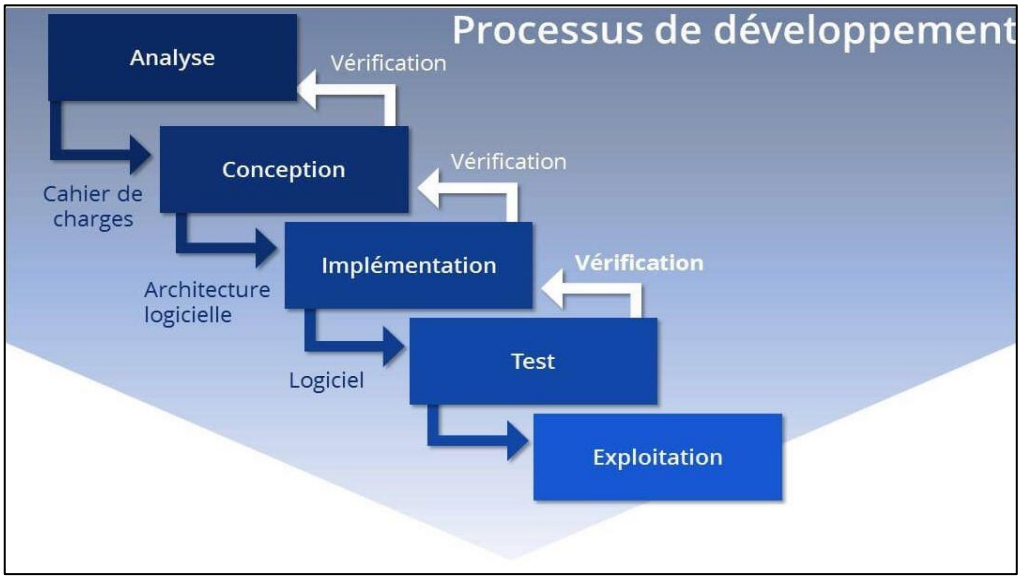


Figure 16 - Processus de développement [34]

Vue d'ensemble des avantages et inconvénients du modèle en cascade [34]

Avantages	Inconvénients
Une structure simple grâce à des phases de projet clairement délimitées.	Les projets complexes ou à plusieurs niveaux ne peuvent que rarement être divisés en phases de projet clairement définies.
Une bonne documentation du processus de développement par des étapes clairement définies.	Une faible marge pour les ajustements du déroulement du projet en raison d'exigences modifiées.
Les coûts et la charge de travail peuvent être estimés dès le début du projet.	L'utilisateur final est uniquement intégré dans le processus de production après la programmation.
Les projets structurés d'après le modèle en cascade peuvent être représentés facilement sur un axe temporel.	Les erreurs sont parfois détectées uniquement à la fin du processus de développement.

Tableau 8 - Avantages et Inconvénients du modèle en cascade

Pour préenter notre solution, nous suivons le modèle cascade, en effet dans ce chapitre nous illustrons notre analyse avec les notations : UML, BPMN et DMN, puis nous proposons une nouvelle architicture qui présente notre solution dans l'étape « conception ».

2. Étape 1 : Analyse

Dans le but de représenter toutes les facettes de notre sujet, nous allons définir 3 Types d'analyse de notre solution qui sont :

- Analyse Métier
- Analyse Informationnelle
- Analyse Décisionnelle

2.1 Analyse Métier

Déjà vu précédemment dans cette partie nous utilisons la figure suivante montre laspet métier de notre solution.

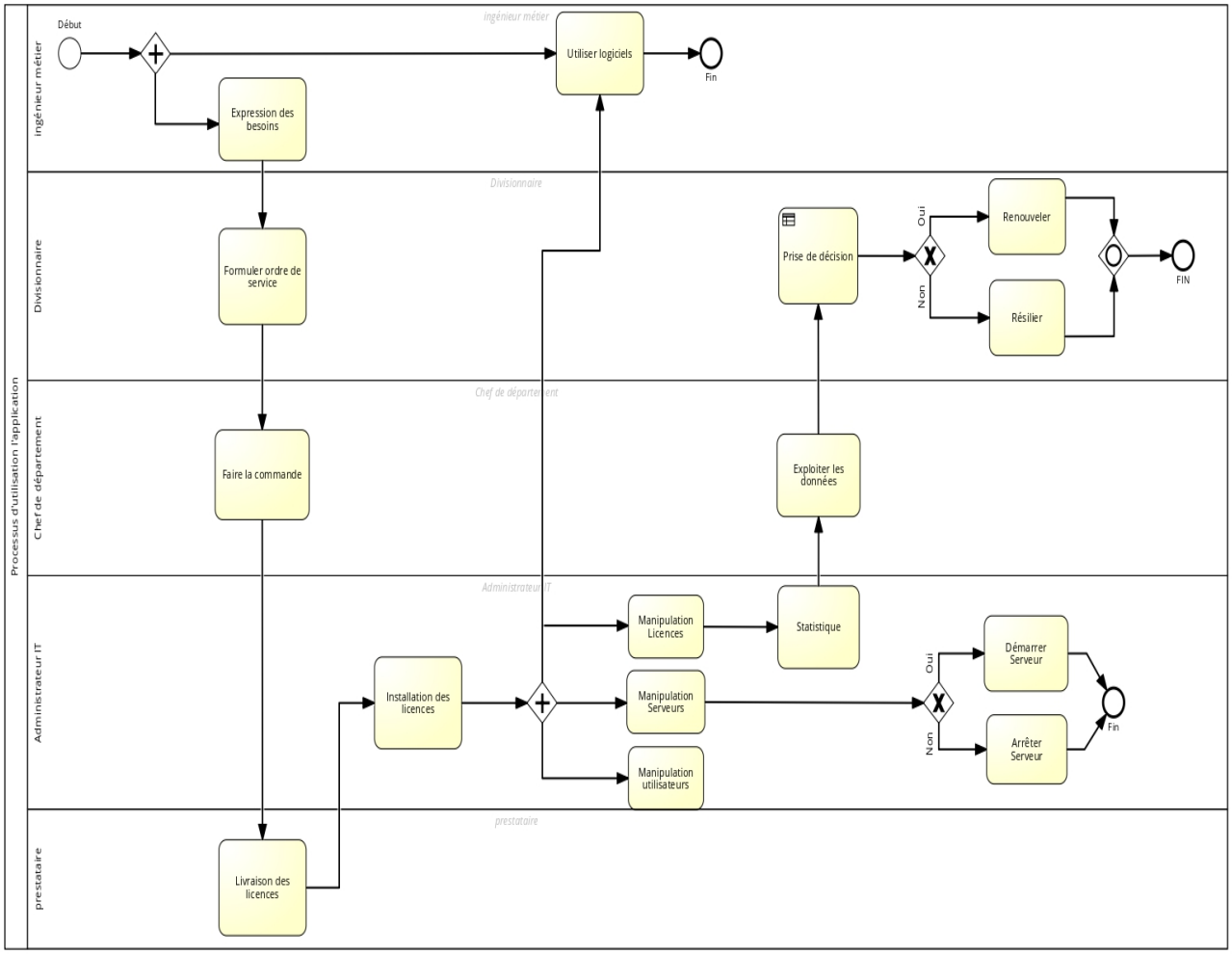


Figure 17 – La modélisation métier

Aspect Métier de l'application :

Acteur : Ingénieur Métier

Il s'agit des différents Ingénieurs de la division PED de SONATRACH qui font l'expresion des besoins et utilisent divers logiciels métier et à qui on attribut les droits d'usage des licences Logiciels

Acteur : Administrateur

Il représente l'administrateur des Licences logiciels métier et utilisateur unique de notre programme et qui passe par plusieurs scénarios :

- Gestion d'accés : l'administrateur doit s'identifier avec un nom d'utilisateur et un mot de passe pour pouvoir accéder à l'interface de l'application

- Accès à l'interface de gestion : après l'authentification l'administrateur à le choix entre plusieurs fonctionnalités qui sont
    - Aide à la décision : c'est la partie qui permet de visualiser les statistiques d'utilisation des licences par mois/année.
    - Manipulation les serveurs : Lancer et arrêter les serveurs des licences.
    - Manipulation les Licences
    - Manipulation Utilisateurs

Acteur : Chef de département

Il représente le responsable du département informatique qui fais la commande et exploite principalement les données de notre application et définit les besoins en matière des licences

Acteur : Divisionnaire

Il s'agit du haut responsable de la division de production à qui revient le pouvoir de formuler ordres de service et prendre des décisions quant à l'achat ou non des licences en se basant sur les statistiques liées à l'utilisation de ces deniers généré par notre solution.

2.2Analyse Informationnelle

Dans le but de mieux comprendre notre solution et les interactions avec les utilisateurs, dans cette partie nous allons détailler les scenarios des principaux cas d'utilisation.

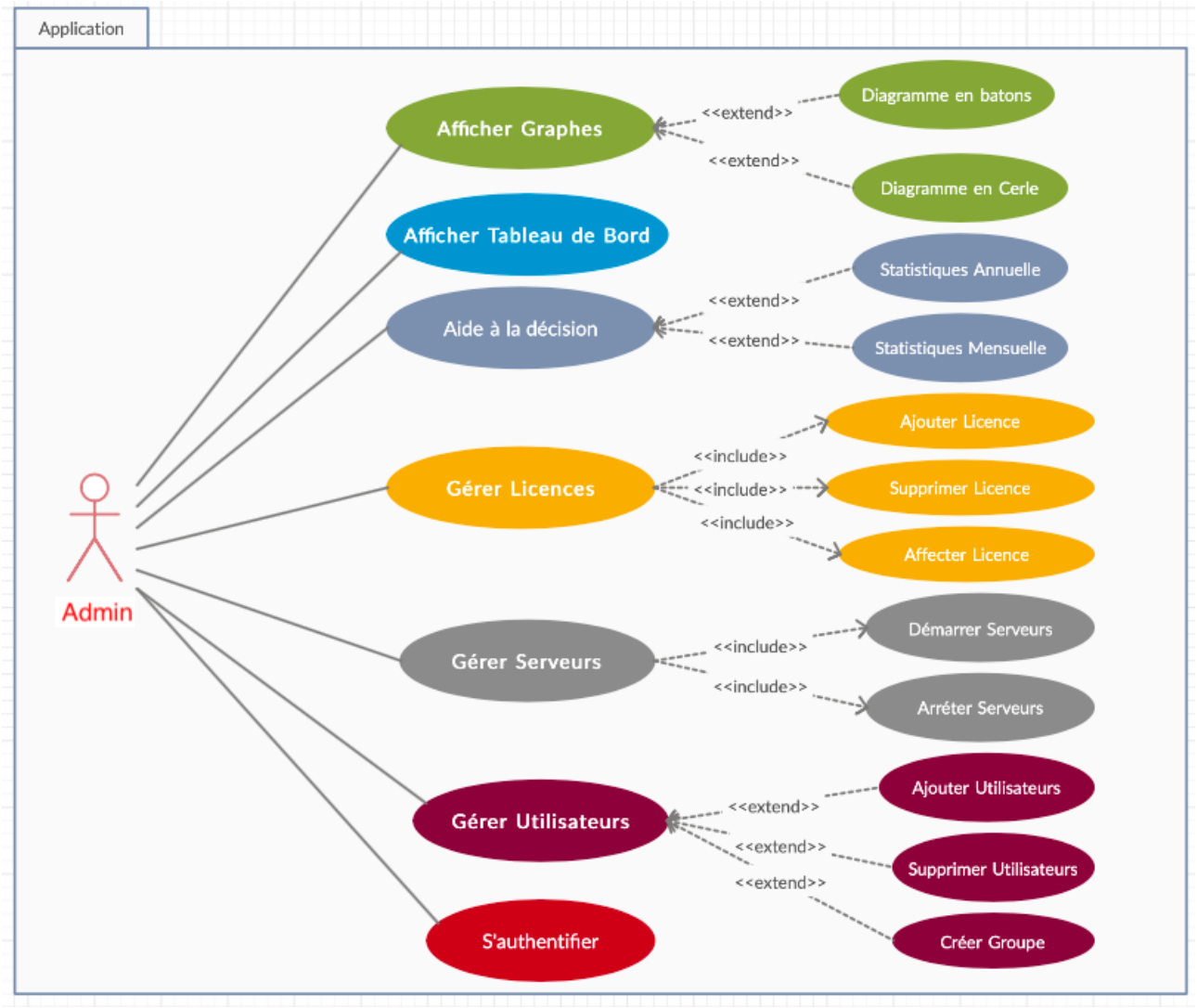


Figure 18 - Diagramme de cas d'utilisation de l'application

Description textuelle des principaux cas d'utilisation

Cas d'utilisation N° 1	Manipulation des Licences
Résumé	Permet à un administrateur de manipule licence.
Acteurs	Administrateur.
Précondition	Identification pour l'utilisation de l'application.
Scénario nominal	« Début » 1. Le système permet à l'administrateur d'ajouter des licences pour les exploiter par les utilisateurs.

	2. Le système permet de supprimer une ou plusieurs licences. 3. Affecter une licence à un utilisateur ou un groupe d'utilisateurs « Fin »

Tableau 9 - Description textuelle du cas d'utilisation « gérer licences »

Cas d'utilisation N° 2	Manipulation des serveurs
Résumé	Permet à l'administrateur de manipule l'états de serveur.
Acteurs	Administrateur.
Précondition	Identification pour l'utilisation de l'application
Scénario nominal	« Début » 1. Le système permet de lancer la commande pour démarrer le serveur. 2. Le système permet de lancer la commande d'Arrêt du serveur. « Fin »

Tableau 10 - Description textuelle du cas d'utilisation « gérer serveurs »

Cas d'utilisation N° 3	Manipultaion des Utilisateurs
Résumé	Permet à l'administrateur de manipule tous les utilisateurs qui utiliser les licences.
Acteurs	Administrateur
Précondition	Avoir déjà un compte par Active directory
Scénario nominal	« Début » 1. L'administrateur ajoute un utilisateur au système pour pouvoir utiliser des licences. 2. L'administrateur supprime un utilisateur du système. 3. L'administrateur peut créer un groupe d'utilisateurs « Fin »

Tableau 11 - Description textuelle du cas « Gérer utilisateurs »

Cas d'utilisation N° 4	Consulter tableau de bord
Résumé	Permet à l'administrateur de consulter et voir l'état des licences.
Acteurs	Administrateur.
Précondition	L'administrateur doit être connecté au système (authentification).
Scénario nominal	« Début » 1. Le système affiche un tableau de bord qui permet de donner une visibilité en temps réel sur l'utilisation des licences. 2. Le système importe les informations de la base de données. 3. L'administrateur peut examiner l'états des données réactives dans le tableau. 4. L'administrateur peut voir les statistiques des licences via un graphe par mois et années. « Fin »
Scénario alternative	Dans le cas où une information est incomplète un message d'erreur est affiché et le system continue l'exécution réactives.

Tableau 12 - Description textuelle du cas « Affichage tableau de bord »

Cas d'utilisation N° 5	Aide a la decision
Résumé	Permet à l'administrateur de voir la la traçabilité des licences est des utilisateur.
Acteurs	Administrateur.
Précondition	L'administrateur doit être connecté au système (authentification).
Scénario nominal	« Début » 4. Le système offre un bouton qui affiche un tableau qui permet de donner une visibilité sur les licences annuelles et un tableau que permet de donner une visibilité sur les utilisateurs. 4. Le système offre un bouton qui affiche un tableau qui permet de donner une visibilité mensuelle des licences. « Fin »

Tableau 13 - Description textuelle du cas « Aide a la decision »

2.3 Analyse Décisionnelle

Après avoir élaboré les diagrammes des cas d'utilisations et la description textuelle, nous allons nous focaliser sur la modéisation de l'aspect décisionel avec le DMN.

Notre diagramme DMN est basé sur quatre données d'entrée pour la prise des décisions à l'achat et l'utilisation des licences :

- Nombre d'utilisateurs d'une licence.
- Nombre d'occurrence d'une licence.
- Durée de vie.
- Temps d'utilisation d'une licence.

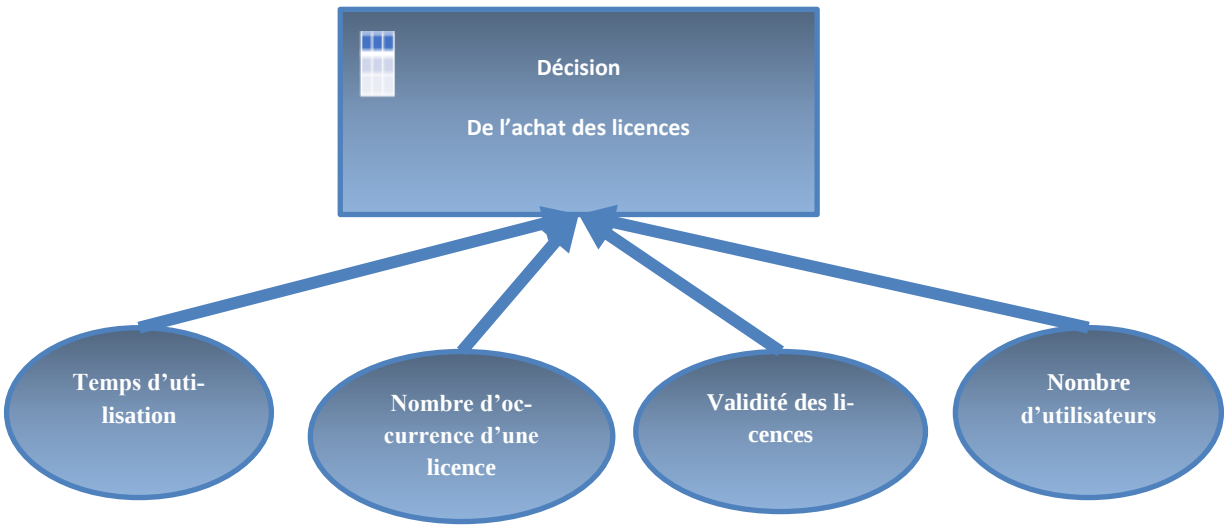


Figure 19 – Model de besoin de DMN

La figure suivante illustre le diagramme DMN pour la prise de décision « Achat »

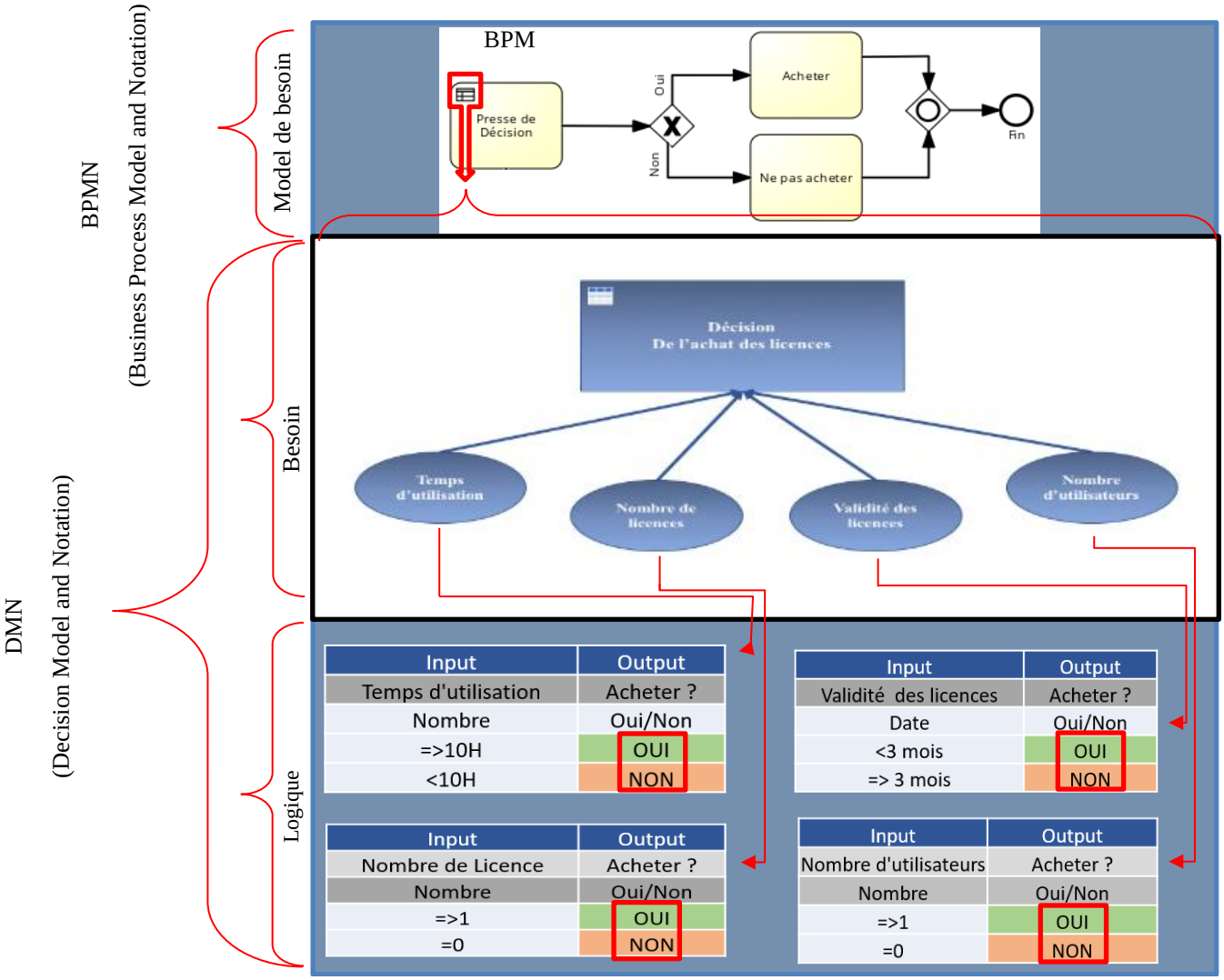


Figure 20 –Diagramme DMN pour la décision d'achat

La table ci-dessus nous permet de prendre les décisions selon les données entrantes décrit dans les colonnes d'entrer et sortir avec des décisions optimales généré sur la partie sortie du tableau de manière détaillée.

Description:

- Nombre d'utilisateurs des licences

Input	Output
Nombre d'utilisateurs	**Acheter?**
Nombre	Oui/Non
=>1	OUI
=0	NON

Tableau 14 – Table de décision du nombre d'utilisateurs

Input : Nombre d'utilisateurs des licences

Output : Décision Acheter ou ne pas acheter des licences

Conditions : =>1 ou =0

Décision : Si le nombre d'utilisateurs est => 1 nous achetons des licences car il existe un besoin au niveau de la production métier

- Nombre d'occurrence d'une licence.

Input	Output
Nombre de Licence	**Acheter?**
Nombre	Oui/Non
=>1	OUI
=0	NON

Tableau 15 - Table de décision d'occurrence d'une licence

Input : Nombre d'occurrence d'une licence

Output : Décision Acheter ou ne pas acheter des licences

Conditions : =>1 ou =0

Décision : Si le nombre d'occurrence d'une licence est => 1 nous achetons la licence en question car cette dernière est utilisée au moins une fois.

- Validité d'une licence

Input	Output
Validité en cours	**Acheter?**
Date	Oui/Non
<3 mois	OUI
=> 3 mois	NON

Tableau 16 - Table de décision de la validité en cours d'une licence

Input : Validité en cours d'une licence

Output : Décision Acheter ou ne pas acheter des licences

Conditions : =>3 mois ou >3 mois

Décision : Si la validité d'une licence est <3 mois nous procédons au renouvellement de cette dernière

- Temps d'utilisation d'une licence.

Input	Output
Temps d'utilisation	**Acheter?**
Nombre	Oui/Non
=>10H	OUI
<10H	NON

Tableau 17 - Table de décision de temps d'utilisation d'une licence

Input : Temps d'utilisation d'une licence

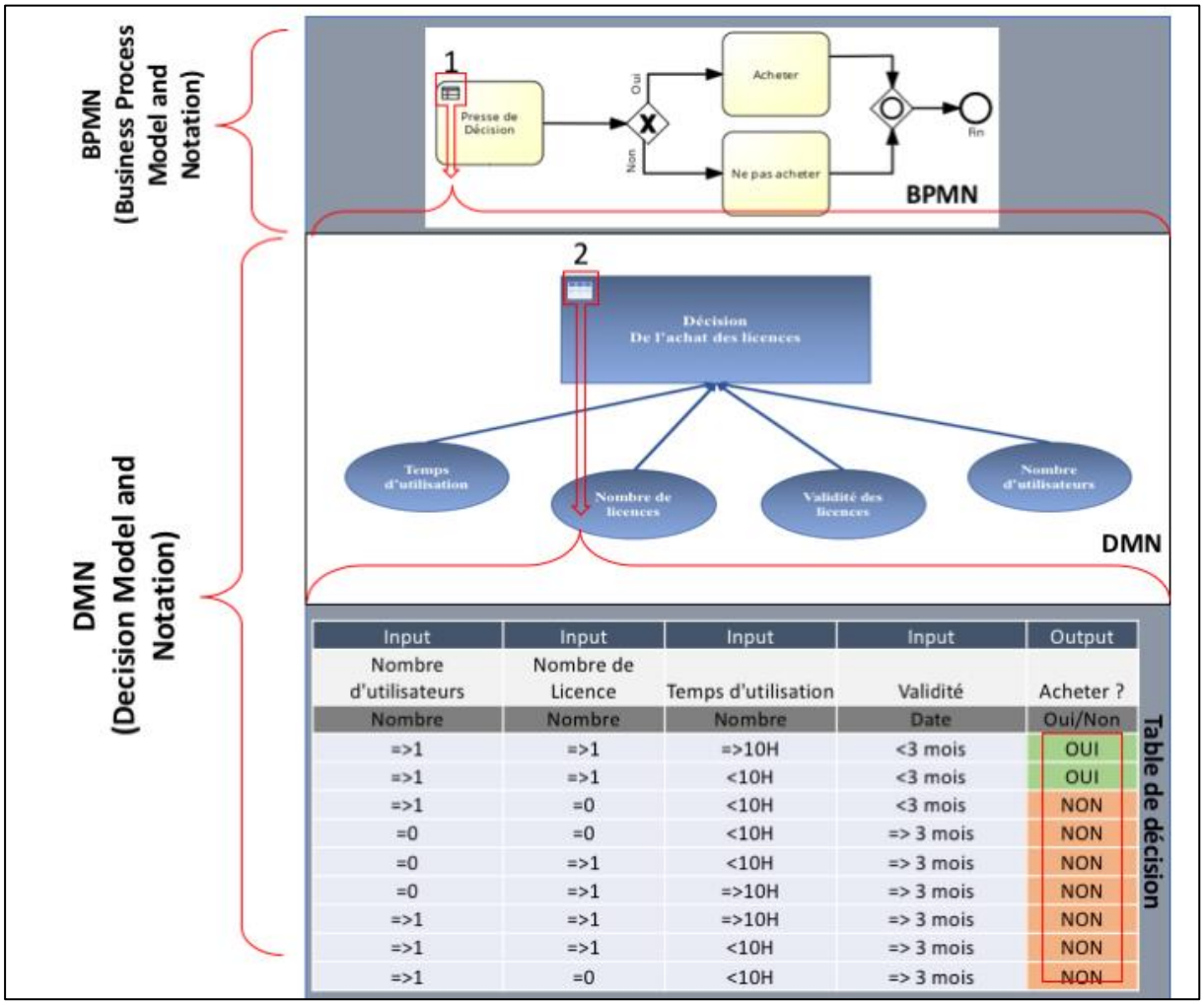
Output : Décision Acheter ou ne pas acheter des licences

Conditions : =>10 Heures ou >10H Heures

Décision : Si la durée d'utilisation d'une licence est >10 Heures nous jugeons qu'il existe un besoin réel de cette licence et nous procédons à l'achat

- **Tables décisionnelles globale de DMN**

Ci-après le Table décisionnelle sur laquelle on se base pour prendre les décisions liées à l'achat des licences :



Input	Input	Input	Input	Output
Nombre d'utilisateurs	Nombre de Licence	Temps d'utilisation	Validité	Acheter ?
Nombre	Nombre	Nombre	Date	Oui/Non
=>1	=>1	=>10H	<3 mois	OUI
=>1	=>1	<10H	<3 mois	OUI
=>1	=0	<10H	<3 mois	NON
=0	=0	<10H	=> 3 mois	NON
=0	=>1	<10H	=> 3 mois	NON
=0	=>1	=>10H	=> 3 mois	NON
=>1	=>1	=>10H	=> 3 mois	NON
=>1	=>1	<10H	=> 3 mois	NON
=>1	=0	<10H	=> 3 mois	NON

Figure 21 - Table des Décisions globales pour la décision « Achat »

Input :

- Temps d'utilisation d'une licence
- Nombre d'occurrence d'une licence
- Nombre d'utilisateur des licences
- Validité d'une licence

Output : Décision Acheter ou ne pas acheter des licences

Conditions : Nombre d'utilisateurs =>1 / Nombre de licence => 1 / Temps d'utilisation =>10 ou Temps d'utilisation <10H Heures / Validité <3 mois

Décision : nous Achetons des licences dans le cas où toutes les conditions sont réunies, à noter aussi que la condition principale pour prendre une décision est que la durée de validité de la licence en question soit < 3mois.

3. Étape 2 : La Conception

Dans cette étape nous allons passer à la conception de l'application, l'élaboration du schéma global du projet, ce dernier sera par la suite scindé en (05) cinq modules principaux, qui seront abordés d'une manière détaillée.

3.1 La conception générale

Ci-dessous le schéma qui synthétise l'architecture proposée au sein du département informatique de la division PED « en bleu », et les modules conçus dans le cadre de notre projet « en rouge ».

Description de l'architecture

- **Traitement Back end :**

Base de données : pour avoir une architicture performante, nous utilisons MongoDB, ce dernier est utilisé pour repndre aux besoins de performances.

Nous avans opté pour une base de données NonSQL qui permet : le stockage dynamique séquentiel des données issues des serveurs, contenant les informations sur l'état des licences utilisées à l'instant t.

Programmation Réactive :

Ça représente le programme réactive basé principalement sur les webflux, développés par springboot avec langage Java qui s'exécute de manière continue, où il envoi des requêtes en forme de multithread en simultané pour générer les fichiers LOG dans les serveurs des licences, et qui contiennent toutes les informations liées aux utilisateurs des licences et l'heures de début et de fin de leurs utilisations, aussi il permet l'envoi des informations collecter sur les serveurs vers le front end.

Serveurs de Licence :

C’est les serveurs ou sont installer les différentes licences fournis par les prestataires fournisseurs des logiciels.

- **Traitement front END :**

Poste de Travail:

Il s’agit des différents postes de travail où sont installé les logiciels métiers utilisés par les ingénieurs ou les collaborateurs de la division de production, connectés entre eux, les serveurs de licence et les postes d’administration via le réseau local de l’entreprise.

Administrateur :

Il s’agit de l’administrateur réseau et/ou serveurs de licences qui utilise une interface Front END de l’application, lui permettant la gestion des licences, serveurs et utilisateurs. Cette interface est développée par Angular et assure un affichage dynamique en temps réel des taux d’utilisation des licences logicielles et donne des statistiques d’aide à la décision.

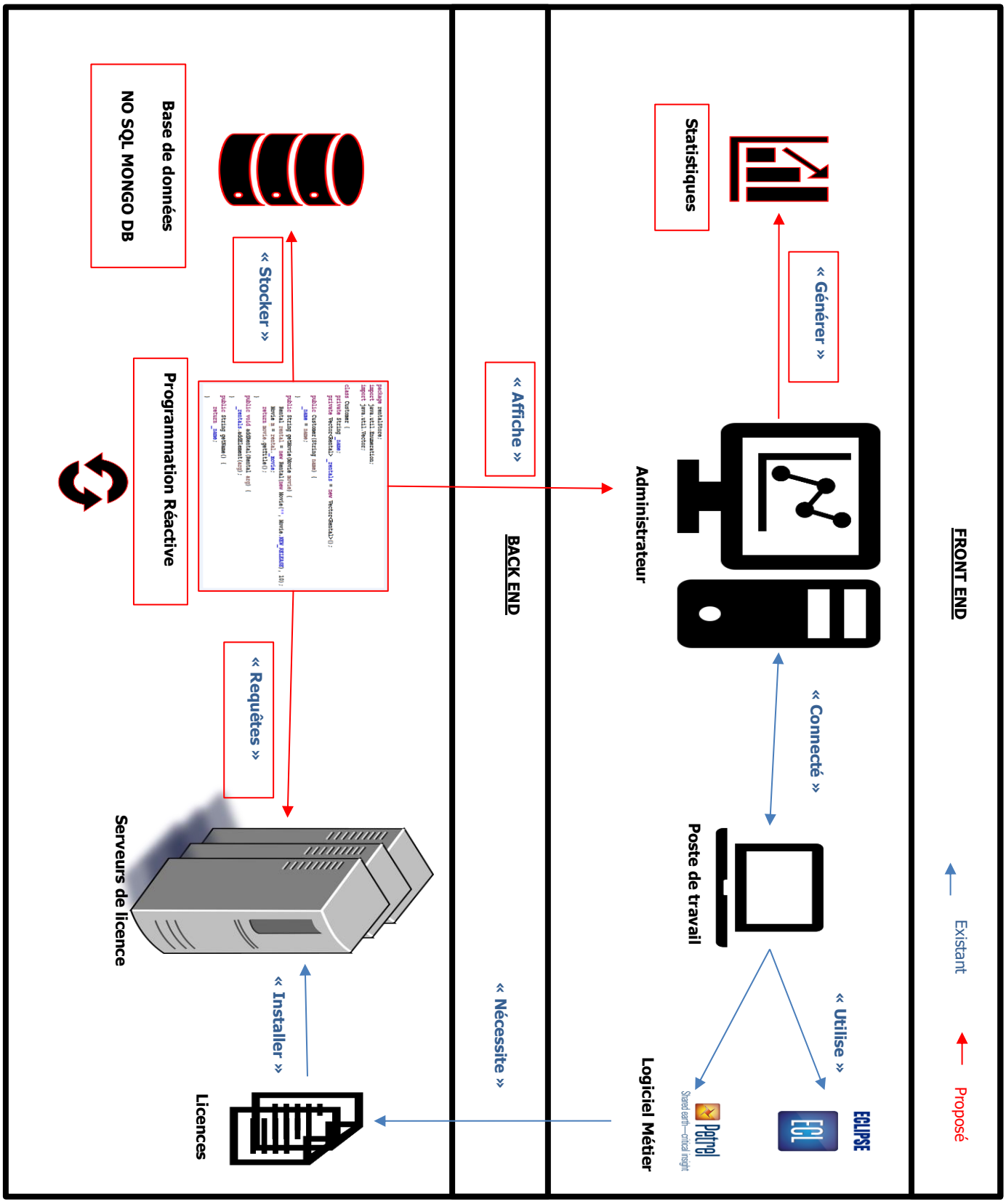


Figure 22 - Architecture Proposée

3.2 La conception détaillée

Dans cette partie nous allons présenter l'architecture à implémenter, constituée de (04) quatre modules détaillés dans ce qui suit.

3.2.1 Module 1: Lancement de l'application

Le programme est une application Web, installée sur un serveur de la division et accessible via une URL, comme montré dans la figure suivante (adresse serveur + port) :

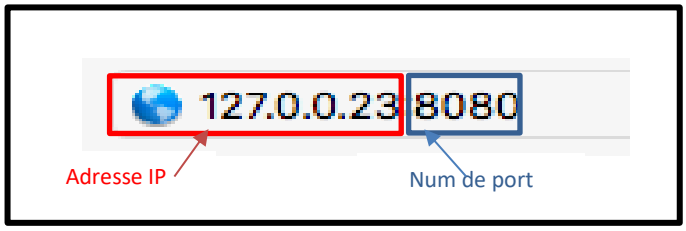


Figure 23 - Adresse IP + N° port

L'authentification de l'administrateur se fait via le même login et mot de passe utiliser lors de l'authentification dans la session de travail Windows. Les informations des comptes sont gérés par l'annuaire Active directory.

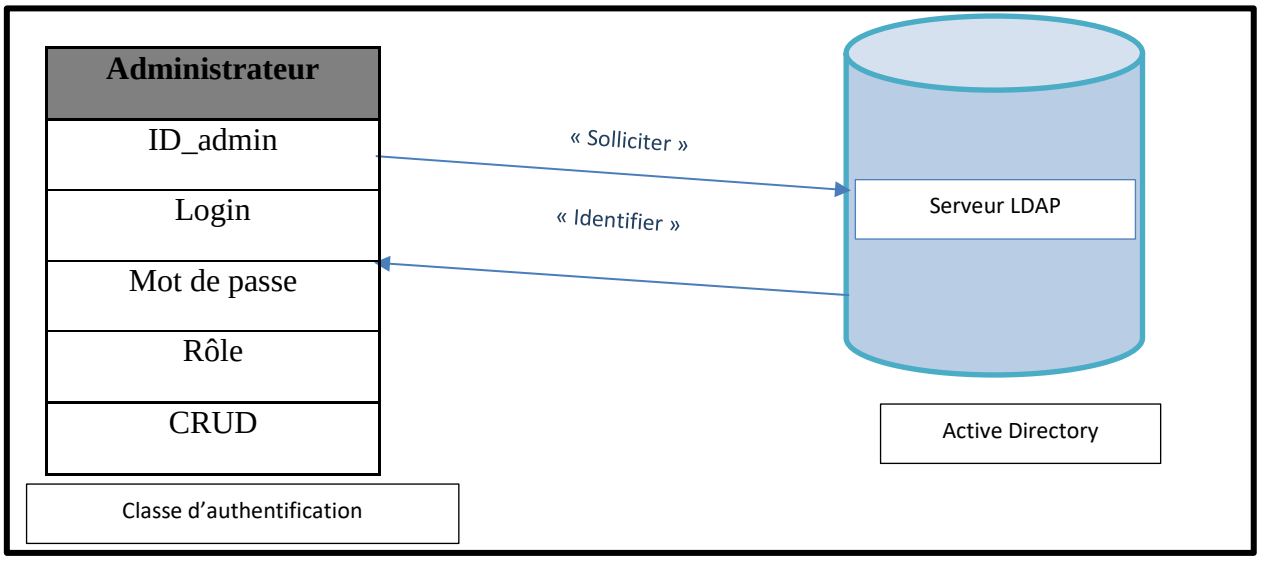
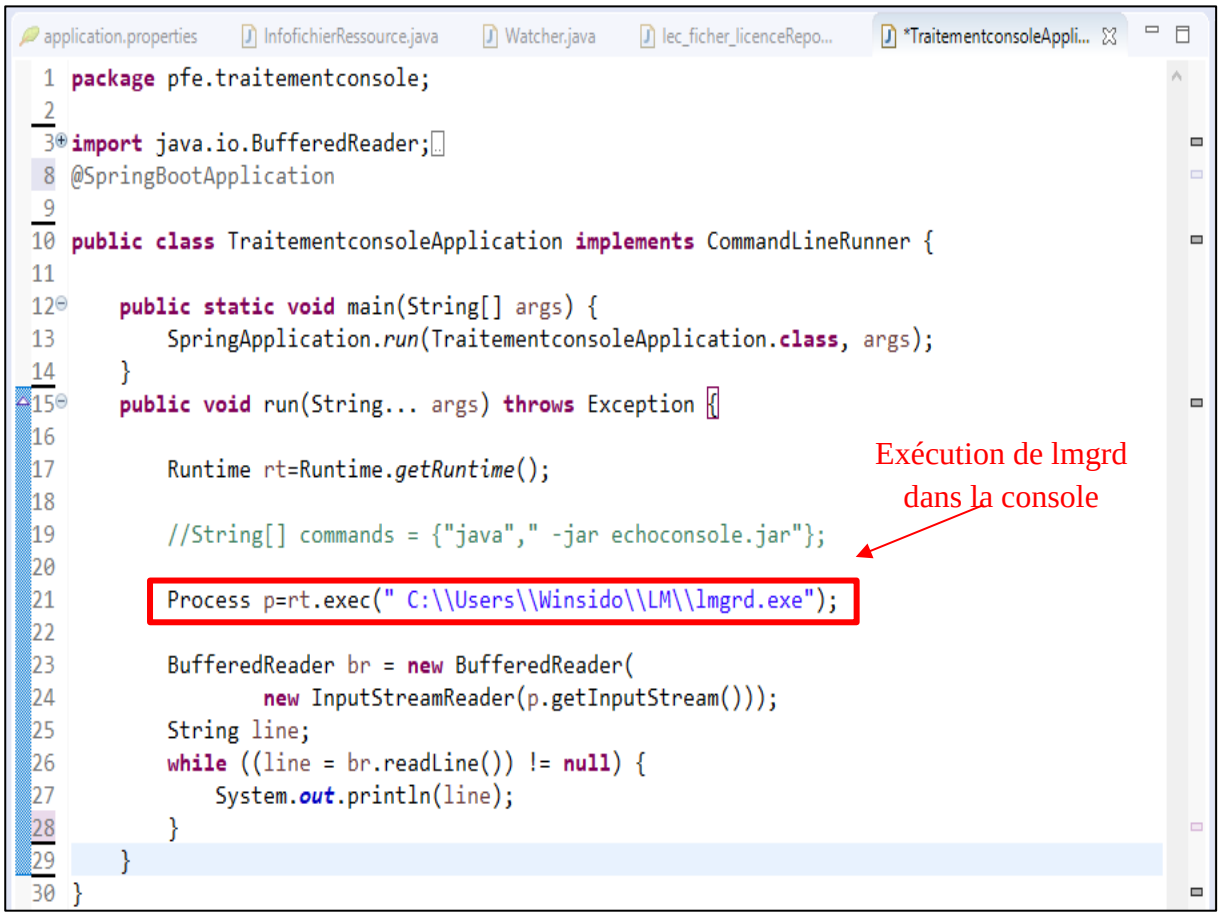


Figure 24 - Architecture d'authentification

Le lancement de l'application entraine l'exécution de la commande « lmgrd » déjà présentée. Cette commande assure la remontée instantanée des informations liées à l'utilisation des licences et les renseigne dans un fichier log.

La figure suivante montre une portion du code développé :




```
1  package pfe.traitementconsole;
2
3  import java.io.BufferedReader;
8  @SpringBootApplication
9
10 public class TraitementconsoleApplication implements CommandLineRunner {
11
12     public static void main(String[] args) {
13         SpringApplication.run(TraitementconsoleApplication.class, args);
14     }
15     public void run(String... args) throws Exception {
16
17         Runtime rt=Runtime.getRuntime();
18
19         //String[] commands = {"java"," -jar echoconsole.jar"};
20
21         Process p=rt.exec(" C:\\Users\\Winsido\\LM\\lmgrd.exe");
22
23         BufferedReader br = new BufferedReader(
24                 new InputStreamReader(p.getInputStream()));
25         String line;
26         while ((line = br.readLine()) != null) {
27             System.out.println(line);
28         }
29     }
30 }
```

Figure 25 - Programme de lancement de la commande « lmgrd »

3.2.2 Module 2 : Lecture du fichier LOG

Le fichier log maintenu à jour par la commande « lmgrd » existe sur le serveur de license sous le format suivant (le concept est déjà présenté dans le chapitre 1) :

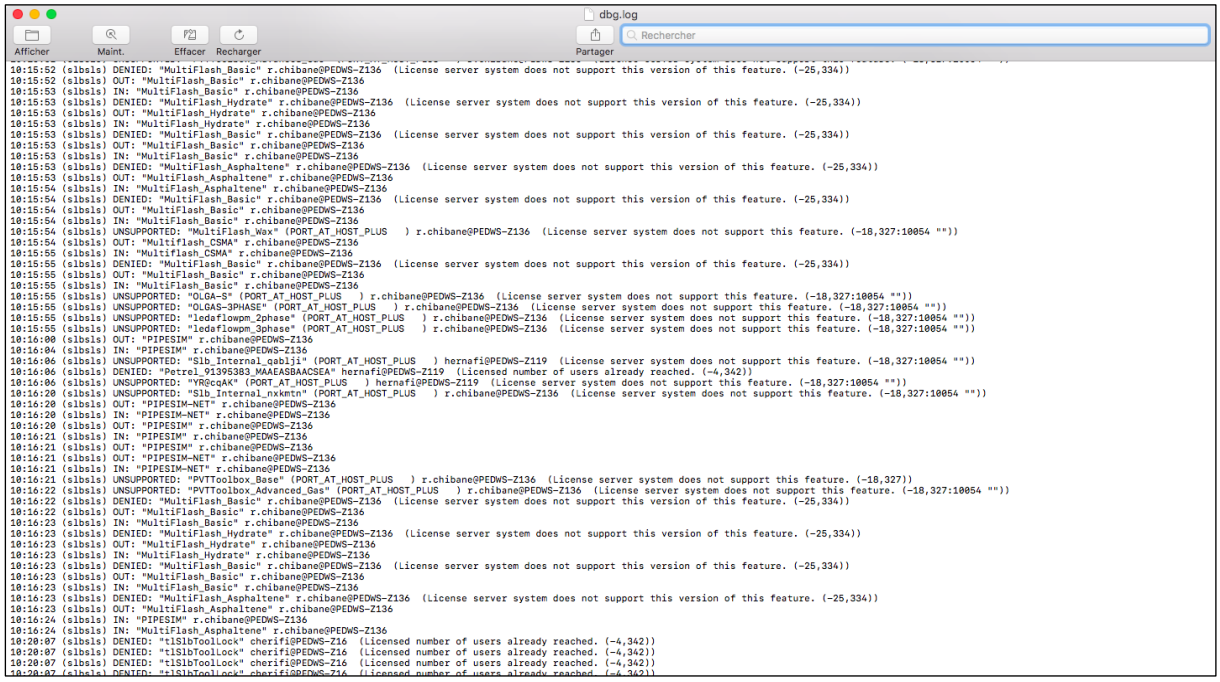


Figure 26 - Fichier LOG en état brut

La Logique de Traitement :

La lecture des informations du fichier log suit la logique suivante :

- Pointer le curseur de lecture sur la première ligne
- Chercher les Mots clés « IN » et « OUT » qui représentent l'accès à la licence et la libération de la licence respectivement
- Ignorer les lignes qui ne comportent pas les mots clés car il s'agit d'informations non liées forcément à l'utilisation des licences
- Prendre les informations utiles à notre étude délimitée par des caractères bien définis sur lesquels on se base pour distinguer les données comme le démontre la figure ci-dessous :

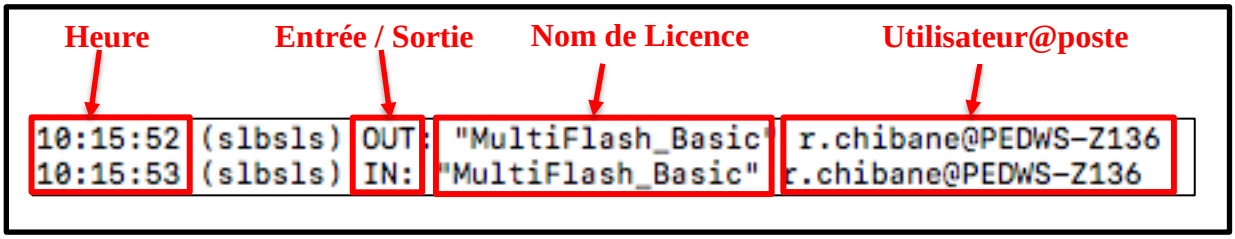
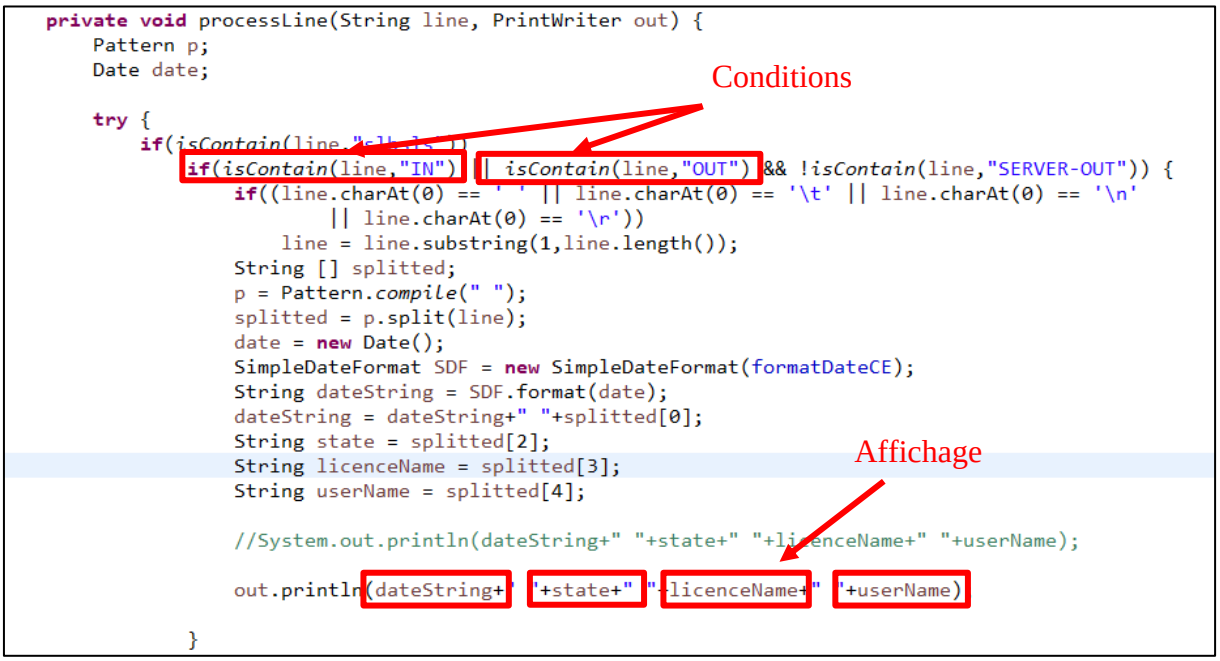


Figure 27 - Fichier LOG détaillé

- Le curseur pointera vers la dernière ligne et lira chaque nouvelle ligne rajoutée, autrement dis c'est une sorte de sentinelle qui surveille toute activité sur le fichier.

Ci-dessous le programme déterminant la lecture du fichier :




```
private void processLine(String line, PrintWriter out) {
    Pattern p;
    Date date;

    try {
        if(isContain(line,"selkat-1"))
            if(isContain(line,"IN") || isContain(line,"OUT") && !isContain(line,"SERVER-OUT")) {
                if((line.charAt(0) == ' ' || line.charAt(0) == '\t' || line.charAt(0) == '\n'
                        || line.charAt(0) == '\r'))
                    line = line.substring(1,line.length());
                String [] splitted;
                p = Pattern.compile(" ");
                splitted = p.split(line);
                date = new Date();
                SimpleDateFormat SDF = new SimpleDateFormat(formatDateCE);
                String dateString = SDF.format(date);
                dateString = dateString+" "+splitted[0];
                String state = splitted[2];
                String licenceName = splitted[3];
                String userName = splitted[4];

                //System.out.println(dateString+" "+state+" "+licenceName+" "+userName);

                out.println(dateString+" "+state+" "+licenceName+" "+userName)

            }
```

Figure 28 - Algorithme de lecture du fichier log

Probléme rencontré :

Le fichier LOG enregistre les heures d'utilisation des licences mais ne comporte pas de date calendaire requises pour notre étude

Solution proposée :

Ajout d'une fonction pour l'extraction de la date du système de gestion du serveurs et l'enregistrée dans la base de données avec l'heures de l'usage.

3.2.3 Module 3 : Affichage sur Tableau de Bord

Une fois l'accès à l'application est établi, l'utilisateur peut choisir le mode d'affichage selon plusieurs modes :

- Tableau dynamique avec rajout d'une case systématiquement s'il y'a usage d'une licence
- Diagramme en batons (X/Y) avec X axe du temps et Y axe des nombres d'utilisateurs
- Diagramme en cercle qui reprend le nombre d'utilisateurs de chaque licence à l'instant T

Les données visualisées (affichées) sont récupérées en temps réel, dynamiquement de la manière reactive.

3.2.4 Module 4 : Manipulation des licences/serveurs/utilisateurs

La Gestions des Licences, utilisateurs ou serveurs se fait via des commandes et scripts Flexnet vu dans le chapitre 1, nous allons nous concentrer sur les commandes utilisées pour la gestion prédéfinit de ces derniers :

	Fonction	Commande
Manipulation des licences	**Ajouter Licence**	**lmstart #lic**
	Supprimer Licence	**Lmremove #lic**
	Affecter Licence	**RESERVE #lic feature type type_name**
Manipulation des serveurs	**Démarrer Serveur**	**startserver server_name**
	Arrêter Serveur	**stopserver server_name**
Manipulation des utilisateurs	**Ajouter Utilisateur**	**INCLUDE feature type type_name**
	Supprimer Utilisateur	**EXCLUDE feature type type_name**
	Créer Groupe	**GROUP group_name user_list**

Tableau 18 - Les commandes de gestion

3.2.5 Module 5 : Stockage/Déstockage de la base de données

Connexion avec la base de données :

Afin d'exploiter les données lues depuis le fichier log ces derniers doivent être stocker dans une base de données non relationnelle comme vu précédemment de ce fait nous devons créer une base de données et la reler avec notre solution via Spring Tools comme le démontre la figure ci-dessous :

Uri : Lien de la base de données sur le serveur local

Username : Nom d'utilisateur

Passoword : mot de passe d'accès à la base de données

Port : Numéro de port de l'application

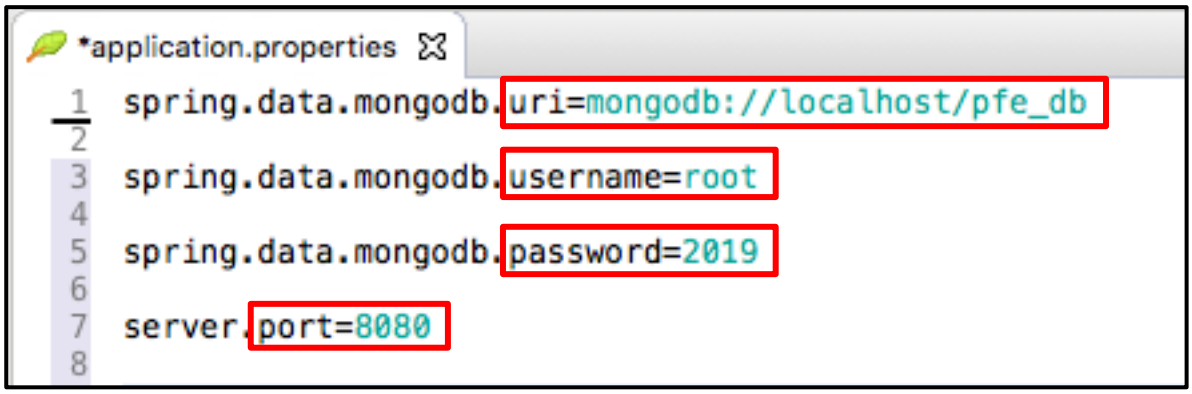


Figure 29 - Connexion à la base de données

Stockage dans la base de données :

Après lecture du fichier log et identification des informations mentionnées dedans nous procédons au stockage de ses informations dans une base de données nosql comme déjà vu auparavant de cette manière :

Figure 30 - Stockage dans la base de données

Les informations sont stockées et une visualisation de la base de données est possible via MongoDB sur la figure ci-dessous nous pouvons voir comment les données sont stocker à l'intérieure de notre base :

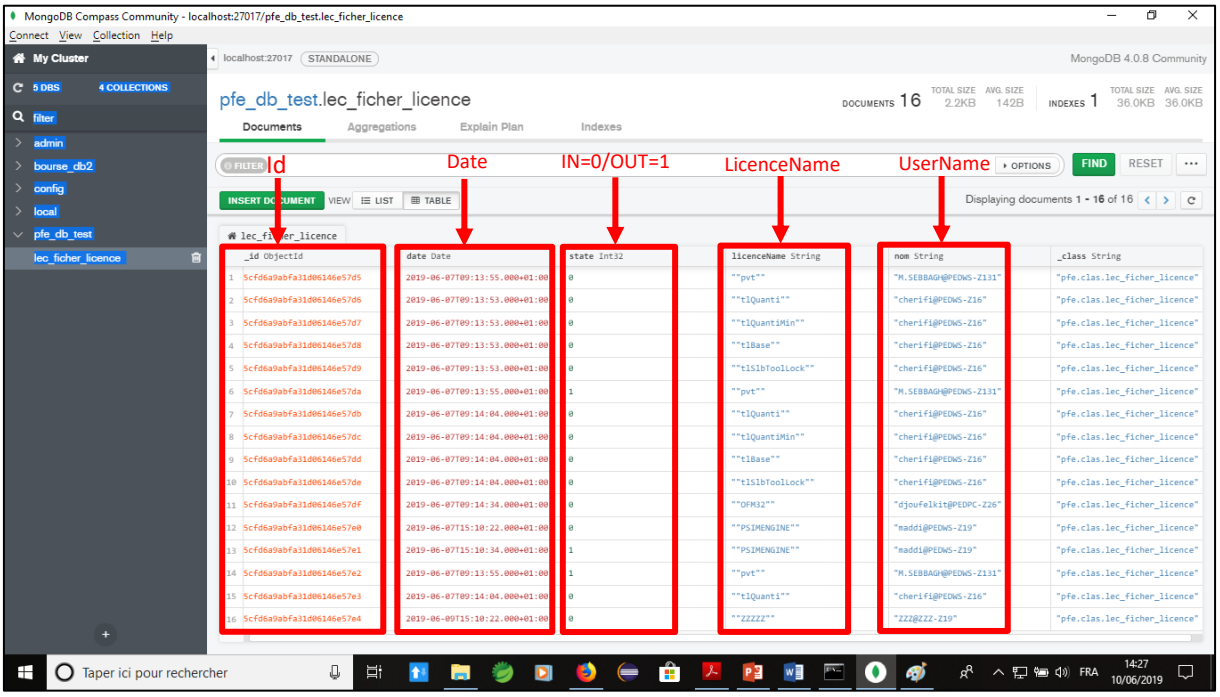


Figure 31 - Aperçu de la base de données

Déstockage de la base de données :

Afin d'afficher les informations stockées dans la base de données à savoir les noms des licences, les noms d'utilisateurs, les date d'utilisation…ect nous n'allons pas passer par des requêtes SQL Classiques mais nous allons plutôt utiliser une nouvelle forme de requêtes qui font appel directement aux données dont on a besoin sans parcourir toute la base de données.

Nous établissons une connexion entre Angular et SpringTools via une Liaison HTTPClient il s'agit d'une connexion qui passe par un URL comme nous montre dans la figure Ci-dessous :

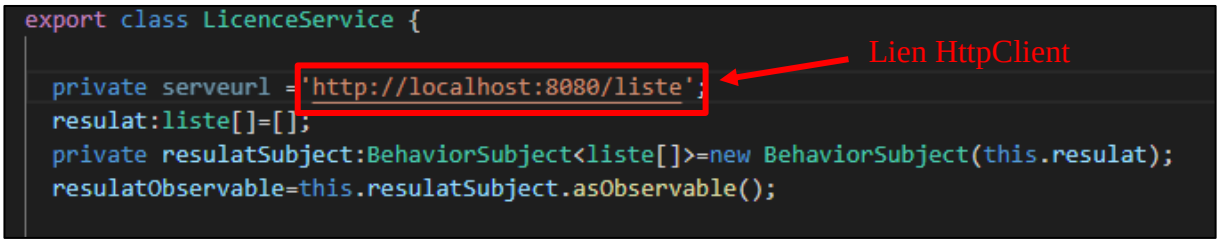


Figure 32 - Connexion Springtool/Angular

Voici la vision coté Springtool Vs la vision coté Angular :

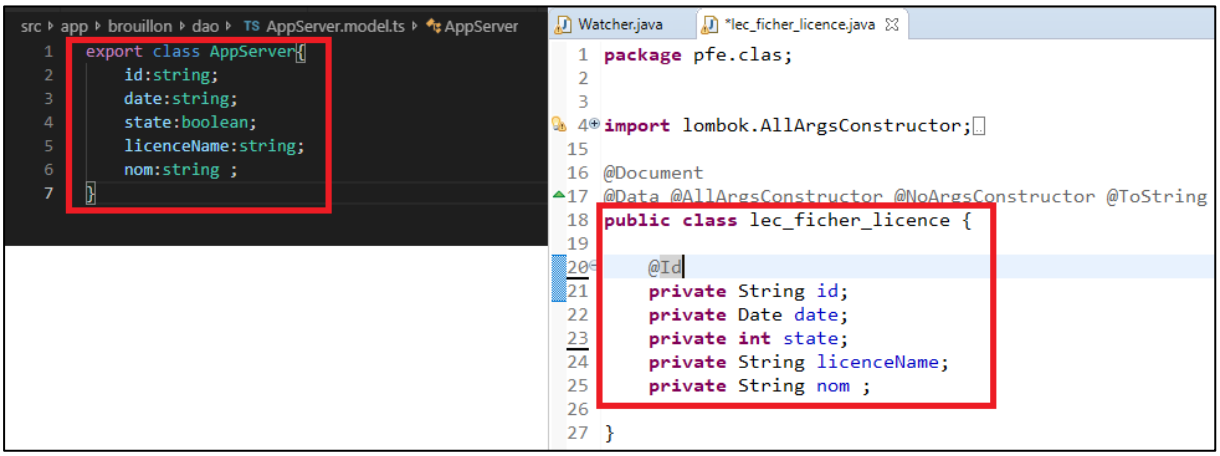


Figure 33 - Class SpringTool Vs Class Angular

4. La conclusion

Dans ce chapitre, nous avons présenté les deux premières étapes de notre processus de développement à savoir l'analyse et la conception en plus de l'architecture proposée de notre solution puis une conception détaillée des différents composants de notre système.

Dans le prochain chapitre nous aborderons les parties, implémentation et tests de notre projet ainsi que la description des choix techniques et les utilisés pour la réalisation de notre solution.

Chapitre 5 : Implémentation et Tests

1. Introduction

Notre but étant fixé et sa conception élaborée, en utilisant le modèle en cascade et les langages de modélisation BPMN et DMN, nous allons passer maintenant à l'implémentation de notre solution basée sur la programmation réactive, qui est considérée comme étant la concrétisation finale de l'architecture proposée.

Il est suivi par la présentation des outils techniques choisis pour le développement, accompagné d'une présentation des interfaces des principaux volets de notre solution. Pour finir, nous allons présenter quelques tests concernant l'efficacité de la programmation réactive et le résultat obtenu.

2. Étape 3 : Implémentation

2.1 Les choix Techniques

Pour la réalisation du projet, nous avons opter pour le modèle vue contrôleur (MVC) pour définir l'architecture, puis nous avons utilisé plusieurs environnements de développement chaque environnement est utilise avec des Frameworks et des langages de programmation spécifiques.

2.1.1 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. [35]

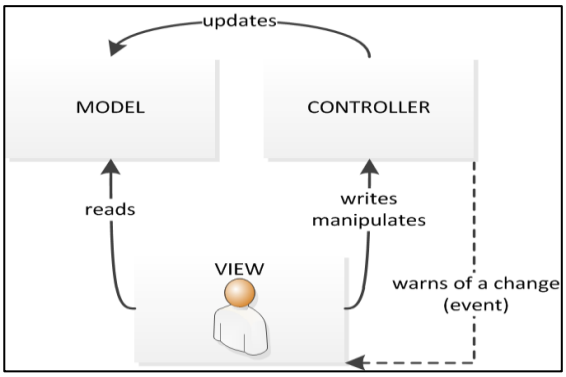


Figure 34 - Architecture MVC

2.1.2 Le développement en Back-end

Le Back end, c'est un peu comme la partie immergée de l'iceberg. Elle est invisible pour les visiteurs, mais il représente une grande partie du développement d'un projet web. Sans Back-end le site web reste une coquille vide... Les technologies que nous avons utilisé pour le Back end, sont :

2.1.2.1 Les Environnements en back-end

En programmation informatique, un environnement de développement est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels [36]. Il comporte un éditeur de texte destiné à la programmation, des fonctions qui permettent, par pression sur un bouton, de démarrer le compilateur ou l'éditeur de liens ainsi qu'un débogueur en ligne, qui permet d'exécuter ligne par ligne le programme en cours de construction [37]. Certains environnements sont dédiés à un langage de programmation en particulier [38],

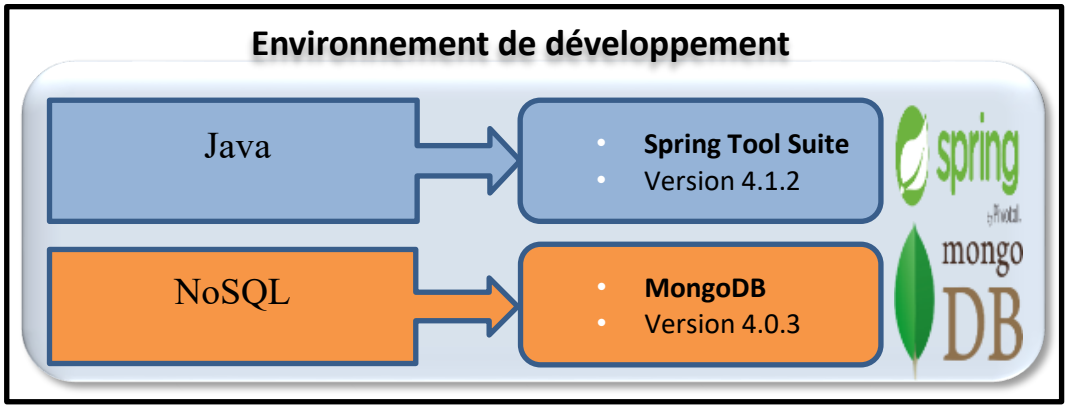


Figure 35 - Environnements de développement back-end

Spring Tool Suite

Spring Tool Suite (STS) fournit le meilleur environnement de développement basé sur Eclipse pour la création d'applications d'entreprise utilisant Spring. STS fournit des outils pour toutes les dernières technologies d'entreprise Java et Spring, et s'ajoute aux dernières versions d'Eclipse.

STS prend en charge le ciblage d'applications sur des serveurs locaux, virtuels et basés sur le cloud. Il est librement disponible pour le développement et les opérations commerciales internes sans limite de temps. [39]

MongoDB

MongoDB (d'origine « humongous » ie « énorme ») est un système NoSQL de type orienté document. C'est une solution open source, écrit en C++, proposé par la société10gen, développé depuis Octobre 2007. Sa popularité est grandissante, vu sa simplicité d'utilisation du point de vue de développement client, ainsi que ces performances remarquables. Il est flexible et peut fonctionner parfaitement avec des masses de données importantes. Il gère des collections de documents JSON (JavaScript Object Notation), équivalentes à des tables dans MySQL, stockées dans un format binaire (BSON). [40]

Les Avantages de MongoDB

- Répond aux besoins de performances
- Garantit la scalabilité horizontale (réplication et sharding)
- Nombreuses fonctionnalités (count, group by, order by, SUM, MIN, etc.)
- Supporte l'indexation pour optimiser les performances

2.1.2.2 Les Langages utilisés en back-end

En informatique, un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire et de significations1,2

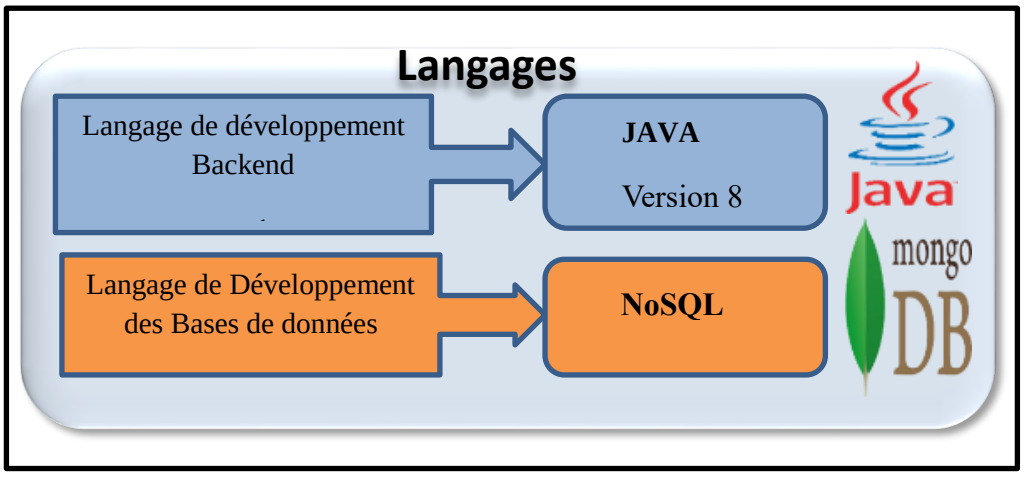


Figure 36 - Langages de développement back-end

JAVA

Java est un langage puissant qui peut être utilisé de plusieurs façons. Il existe trois éditions, Java Standard Edition (Java SE), Java Enterprise Edition (Java EE) et Java Micro Edition (Java ME). Java SE peut être utilisé pour développer des applications autonomes ou des applets côté client. Java EE peut être utilisé pour développer des applications côté serveur, telles que les servlets Java et les pages Java Server. Java ME peut être utilisé pour développer des applications pour les appareils mobiles, tels que les téléphones cellulaires. [41]

NoSQL

La technologie a été abordée dans le chapitre 2 avec la notion des bases de données

2.1.2.3 Les Frameworks utilisés en back-end

Un Framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un « squelette » de programme, un canevas. Il est souvent fourni sous la forme d'une bibliothèque logicielle et accompagné du plan de l'architecture cible du framework. [42]

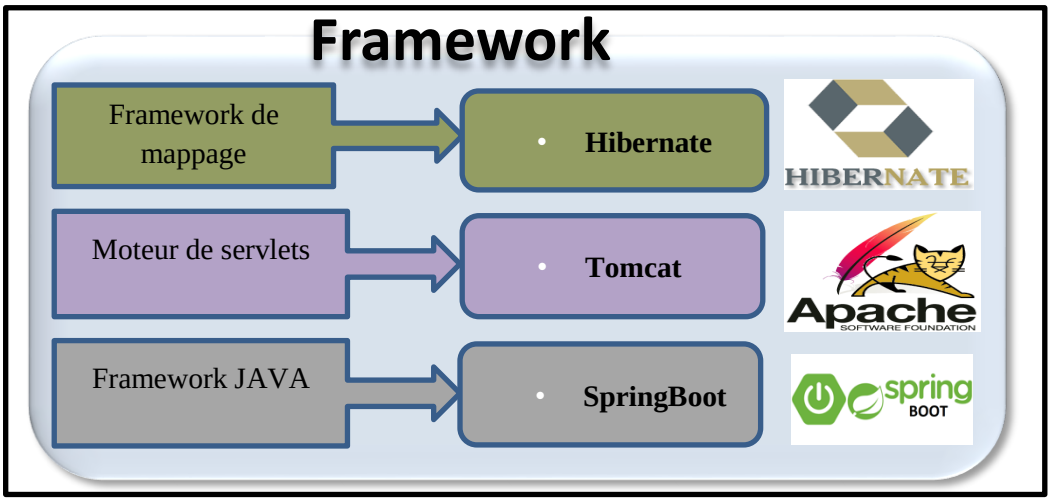


Figure 37 - Frameworks en back-end

Hibernate

Hibernate permet aux développeurs de produire du code évolutif, fiable et efficace. C'est un médiateur qui relie l'environnement orienté objet à l'environnement relationnel. Il fournit des services de persistance pour une application en effectuant toutes les opérations requises dans la communication entre les environnements orientés objet et relationnels. Le stockage, la mise à jour, la suppression et le chargement peuvent être effectués indépendamment de la forme persistante des objets. En outre, Hibernate augmente l'efficacité et les performances de l'application, rend le code moins détaillé et permet au code d'être plus axé sur les règles métier que la logique de persistance [43]

Tomcat

Tomcat est un serveur de servlets (des simples classes) Java et un serveur web de la fondation Apache Software. Il fournit à la fois des technologies Java servlet et JSP (Java Server Pages) (en plus de servir des pages statiques traditionnelles et des programmes CGI (Common Gateway Interface ; littéralement « Interface de passerelle commune ») externes écrits dans n'importe quel langage de programmation). Le résultat est que Tomcat est un bon choix pour une utilisation en tant que serveur web pour de nombreuses applications, y compris en l'utilisant comme un serveur web de production haute performance ; il est open source et un moteur JSP. Il peut être utilisé seul et en conjonction avec d'autres serveurs Web. [44]

2.1.3 Le développement en Front-end

Il s'agit des éléments visibles de l'application, ce que l'on voit à l'écran et avec lesquels on peut interagir. Ces éléments sont composés de HTML, CSS et de JavaScript contrôlés par le navigateur web de l'utilisateur. Ci-après les définitions des langages utilisés dans notre projet :

2.1.3.1 Environnements en front-end

Les environnements que nous avons utilisés pour le front end sont : Angular Material, Visual Studio

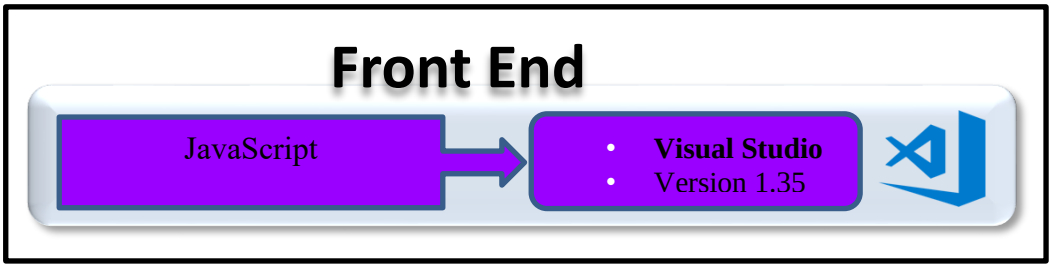


Figure 38 - Environnements de développement Font-end

Visual Studio Code

Visual Studio Code est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et dispose d'un écosystème riche d'extensions pour d'autres langages (tels que C ++, C #, Java, Python, PHP, Go) et les environnements d'exécution (tels que .NET et Unity) [45]

2.1.3.2 Les Langages utilisés en front-end

Pour la partie visible du projet nous avons utilisés plusieurs langages de programmation web qui sont représenté ci-dessous :

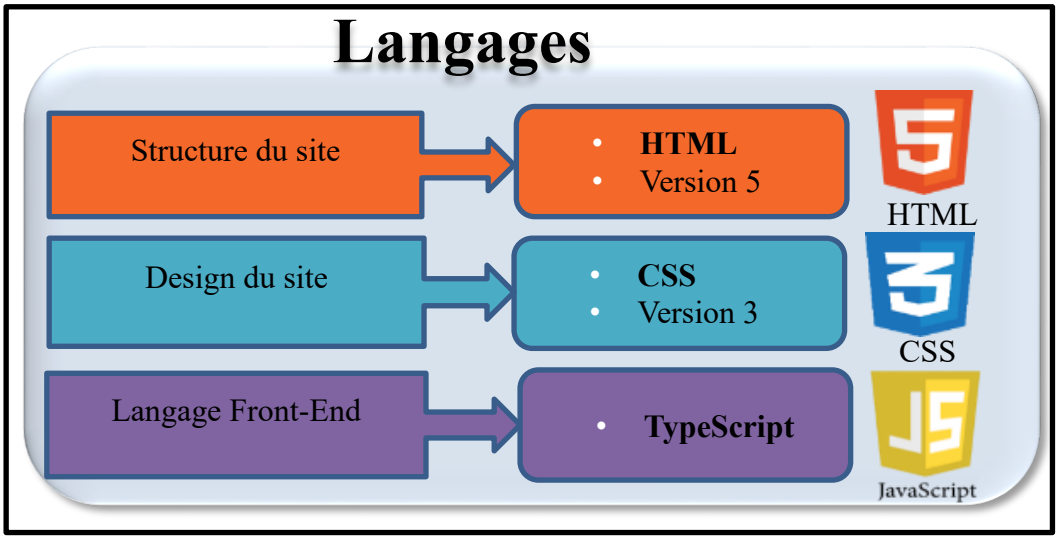


Figure 39 - Langages en Front-end

HTML (Hyper Text Markup Language)

(Hyper Text Markup Language) est un langage de balisage qui contient des codes pour indiquer la mise en page et le style (tels que le gras, l'italique, les paragraphes, le placement de graphiques, etc.) dans un fichier texte. Le langage de balisage est entièrement diffèrent d'un langage de programmation. HTML est utilisé pour créer de l'hypertexte documents qui sont indépendants de la plateforme. Un document HTML est un fichier texte contient les éléments qu'un navigateur utilise pour afficher du texte, des objets multimédias et des hyperliens. Un lien hypertexte relie un document à un autre document. [46]

CSS (Cascading Style Sheets)

CSS fonctionne en associant les règles aux éléments HTML. Ces règles imposent la manière dont le contenu des éléments spécifiés doit être affiché. Il permet aussi de créer des règles qui spécifient comment le contenu d'un élément doit apparaitre. Par exemple, vous pouvez spécifier que l'arrière-plan de la page est blanc, tous les paragraphes doivent apparaître en gris à l'aide de la police de caractères Arial, ou que tous les en-têtes de niveau un doivent être en bleu, italique, Times. [47]

TypeScript

TypeScript est un langage de programmation libre et open source développée par Microsoft qui a pour but d'améliorer et de sécuriser la production de code JavaScript. C'est un sur-ensemble de JavaScript (c'est-à-dire que tout code JavaScript correct peut être utilisé avec TypeScript). Le code TypeScript est transcompilé en JavaScript, pouvant ainsi être interprété par n'importe quel navigateur web ou moteur JavaScript. Il a été cocréé par Anders Hejlsberg, principal inventeur de C# 2,3,4, 5,6.

TypeScript permet un typage statique optionnel des variables et des fonctions, la création de classes et d'interfaces, l'import de modules, tout en conservant l'approche non-contraignante de JavaScript. [48]

2.1.3.3 Les Frameworks utilisés en front-end

Comme pour le back-end nous avons fait appels à plusieurs framework de langages différents afin de donner vie à notre interface d'application :

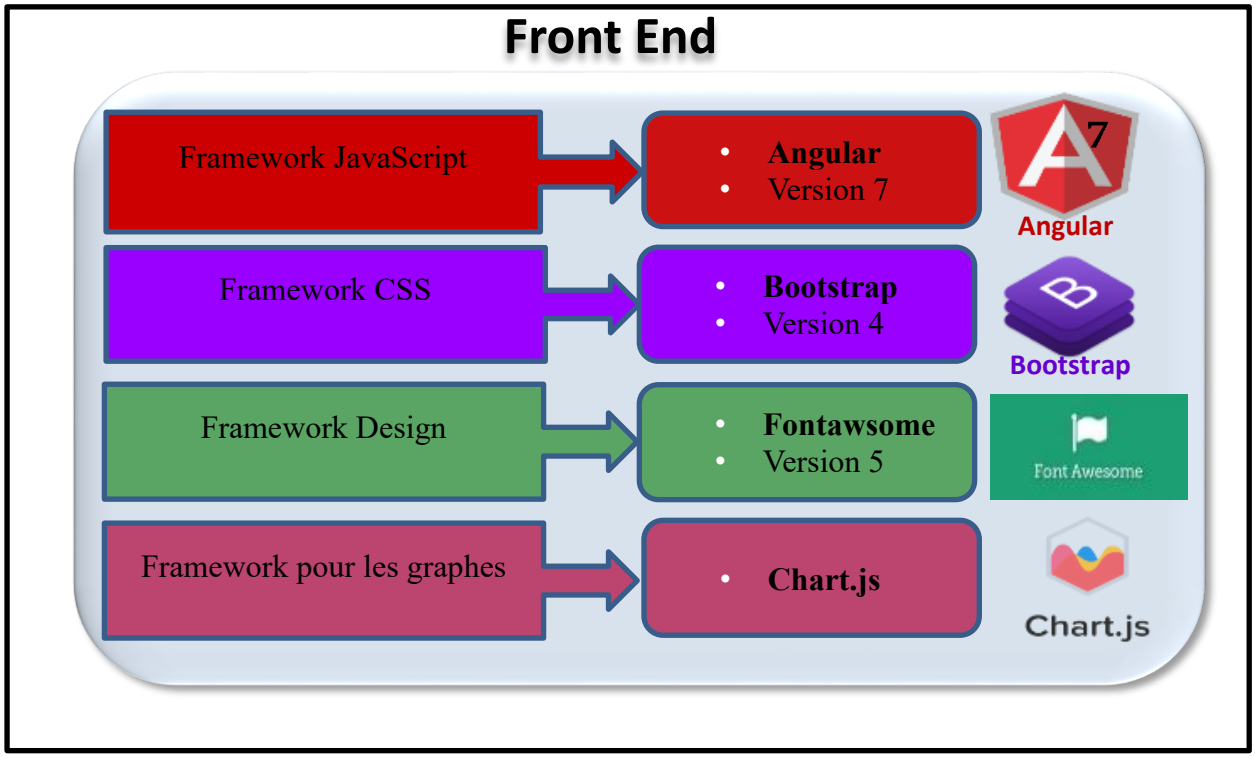


Figure 40 - Framework en Front End

Bootstrap

Bootstrap est un framework CSS. C'est par la suite que ce dernier a évolué vers des composants HTML et JavaScript qui permettent aujourd'hui d'offrir un service complet répondant parfaitement aux attentes du Web et de ceux qui le font évoluer. Depuis l'arrivée de Bootstrap en tant que service open source, de nouvelles possibilités s'offrent au plus grand nombre d'entre nous. N'importe quel individu est capable de réaliser sans effort particulier son projet web, qui demande pourtant en amont de nombreuses connaissances techniques. [49]

Fontawesome

C'est une police de caractères qui permet d'afficher des icônes, des pictogrammes sur un site Web, mais aussi à l'intérieur d'un document Photoshop, Illustrator, Word, Open Office… Crée initialement pour être utilisée avec Bootstrap, elle peut s'utiliser avec n'importe quel projet. Toutes les icônes sont gratuites, personnalisables et redimensionnables à souhait, car sous format vectoriel. [50]

Angular

Angular est un Framework JavaScript Open Source développé par Google est une plateforme qui facilite la création d'applications avec le Web. Angular combine des modèles déclaratifs, une injection de dépendance, des outils de bout en bout et des meilleures pratiques intégrées pour résoudre les problèmes de développement. Angular permet aux développeurs de créer des applications compatibles avec le Web, le mobile ou le bureau. [51]

2.2 La réalisation du projet

Ci-dessous la structure détaillée de l'implémentation de notre projet en utilisant le Framework SpringBoot

2.2.1 Structure du projet

Ci-dessous la structure globale de l'implémentation de notre projet

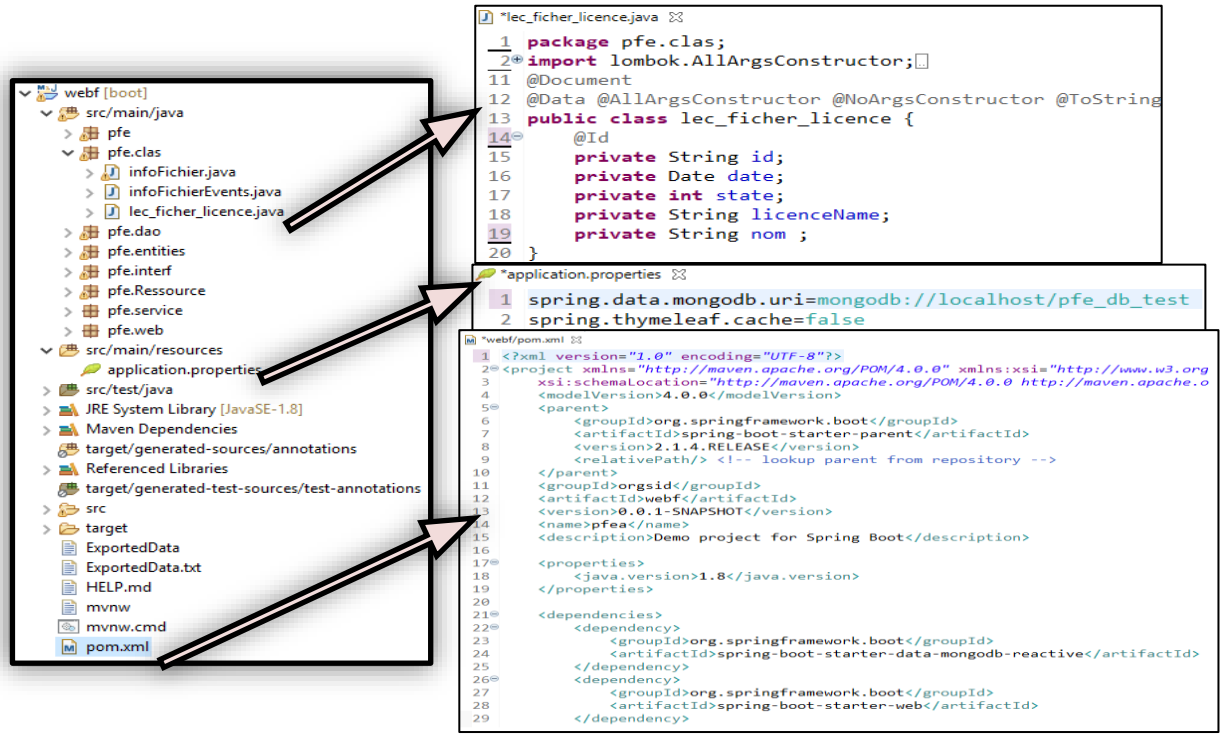


Figure 41 - Structure globale de l'implémentation

D'après la figure on définit les composants en détails dans ce qui suit :

2.2.2 POM (Project Object Model)

Nous devons commencer par créer un fichier Maven pom.xml. Un Project Object Model ou POM est l'unité de travail fondamental de Maven. C'est un fichier XML qui contient des informations sur le projet et les détails de configuration utilisés par Maven pour construire le projet. Lors de l'exécution d'une tâche ou d'un objectif, Maven recherche le POM dans le répertoire en cours. Il lit le POM, obtient les informations de configuration nécessaires, puis exécute l'objectif.

Certaines des configurations qui peuvent être spécifiées dans le POM sont les dépendances du projet, les plugins ou les objectifs qui peuvent être exécutés, les profils de construction, etc. D'autres informations telles que la version du projet, la description, les développeurs, les listes de diffusion et autres peuvent également être spécifiées.

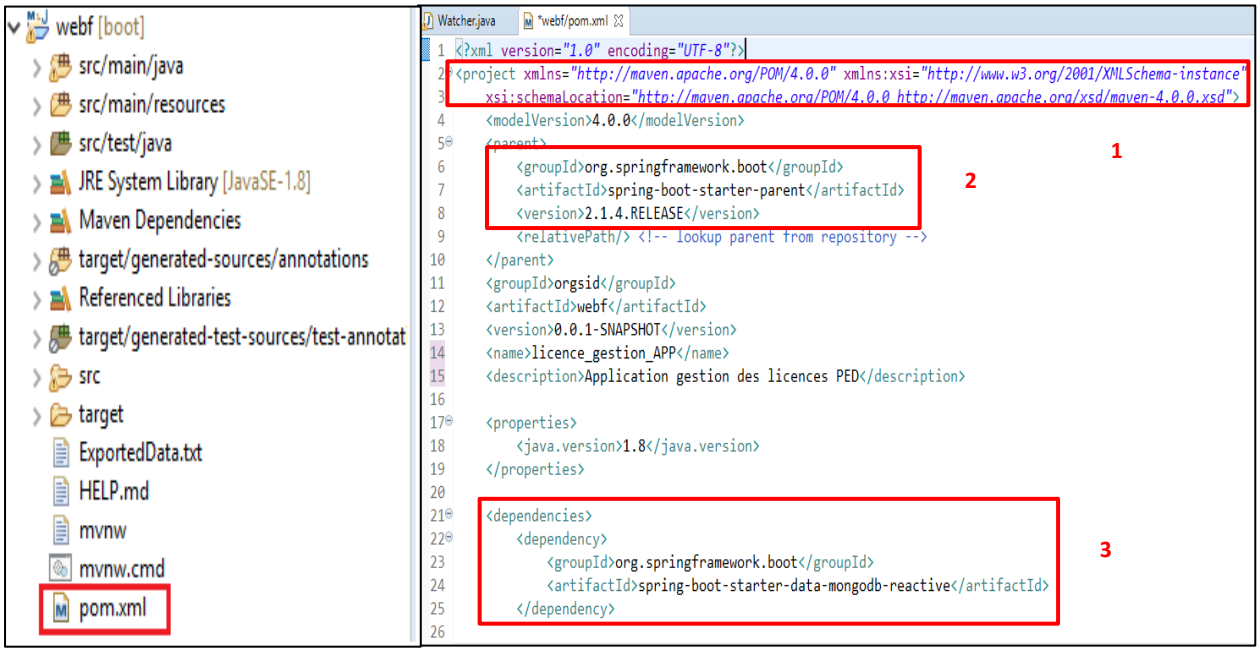


Figure 42 - Structure du fichier POM

1 : Lien vers Apache Maven qui est un outil de gestion et de compréhension de projets logiciels. Basé sur le concept de modèle d'objet de projet (POM), Maven peut gérer la construction, les rapports et la documentation d'un projet à partir d'une information centrale.

2 : Nom du framework spring boot, ses composants et sa version.

3 : Les dépendances Qui sont des fonctionnalités centrales dans Maven est la gestion des dépendances, Le mécanisme de gestion des dépendances de Maven est organisé autour d'un système de coordonnées identifiant des artefacts individuels tels que des bibliothèques de logiciels ou des modules.

2.2.3 Les entités

Le package « pfe.clas » contient les classes Java des entités du projet :

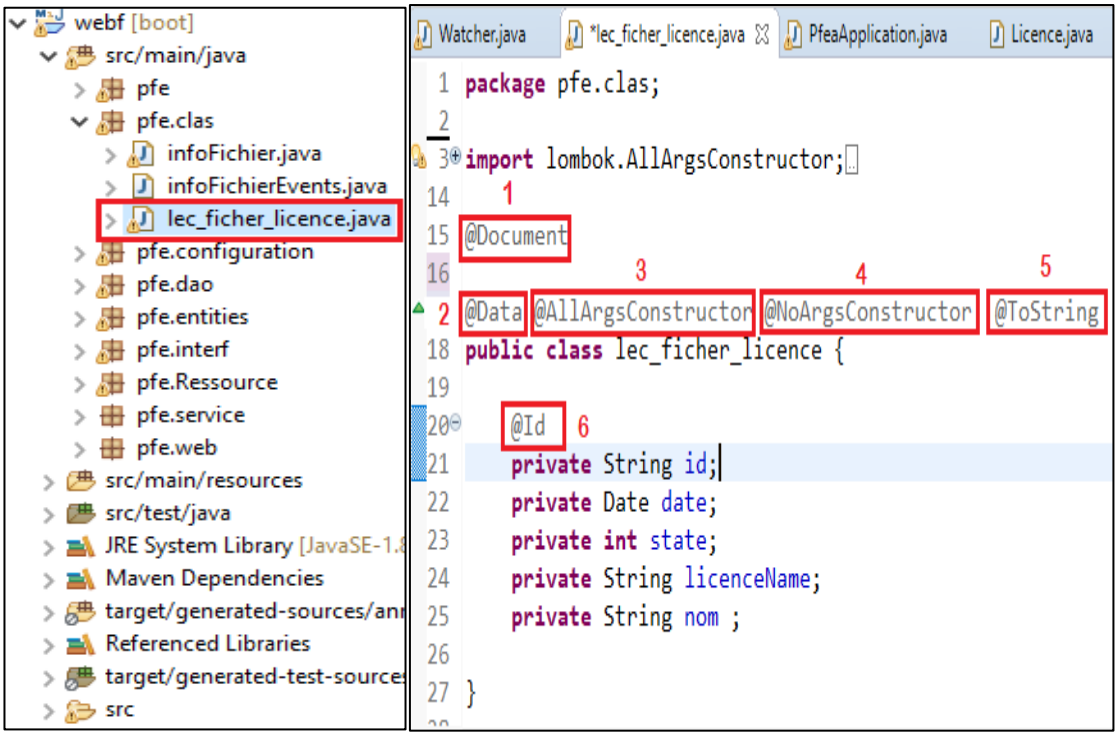


Figure 43 - Les entités du projet et un exemple du code source

Chaque entité créée comporte une interface et son implémentation.

1-@Document : Pour marquer un modèle à utiliser avec Spring Data MongoDB (base de donnes).

2-@Data : est une annotation de raccourci pratique qui regroupe les fonctionnalités des getters et setter.

3-@AllArgsConstructor : défini toutes les annotations répertoriées ici sont placées sur le constructeur généré.

4-@NoArgsConstructor : va générer un constructeur sans paramètre.

5-@ToString : pour permettre de générer une implémentation de la méthode toString (). Par défaut, le nom de votre classe, ainsi que chaque champ.

6-@Id : ce champ est un identifiant.

2.2.4 Repository

Le package « pfe.interf » contient les interfaces Java des entités du projet :

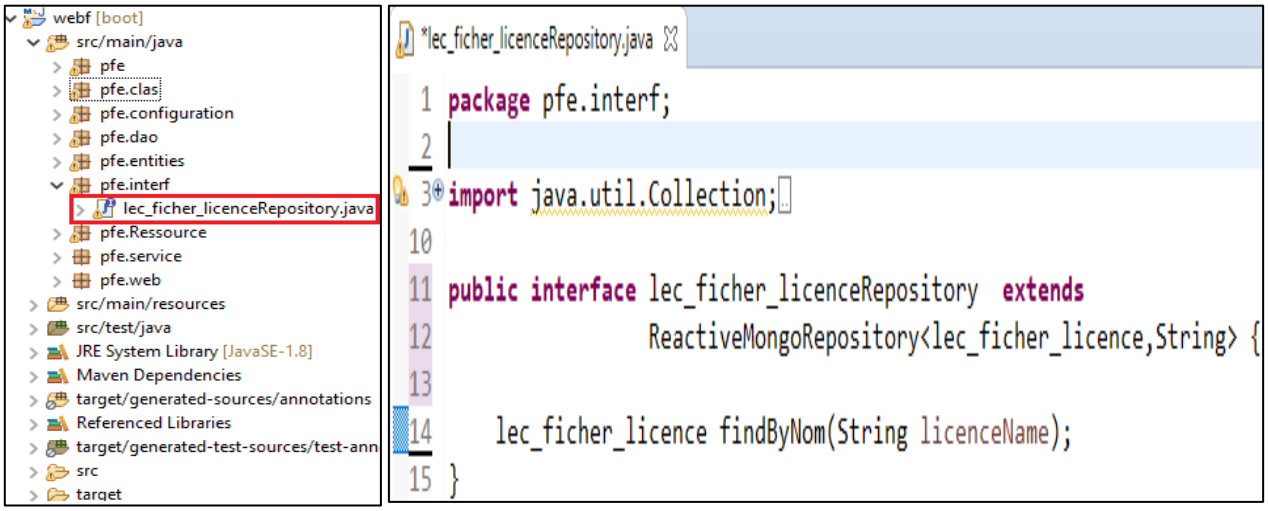


Figure 44 - Les repositories du projet et un exemple du code source

2.2.5 Les contrôleurs

Le package « pfe.Ressource » est structuré comme suite :

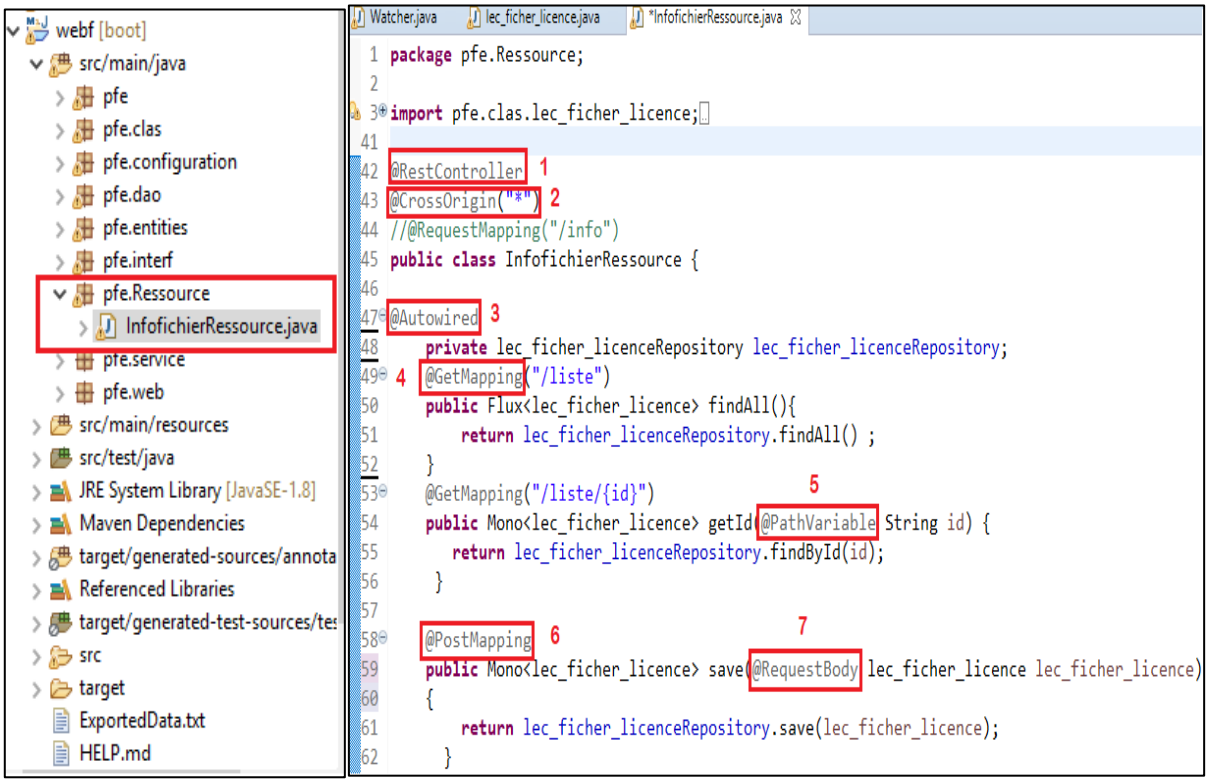


Figure 45 - Les repositories du projet et un exemple du code source

1- @RestController pour dire que c'est un contrôleur Spring MVC (Model View Controller).
2- @CrossOrigin ("*") pour permettre à Angular accéder à tous les url de notre projet à créer à voir s'authentifier.

3-@Autowired permet de faire l'injection de dépendances (signifie que le code précédé de @Autowired peut être réutilisé directement dans d'autres application sans besoin de le redéfinir).

4-@GetMapping Annotation pour le mappage des requêtes HTTP GET sur des méthodes de gestionnaire spécifiques de la base de donnes.
5-@PathVariable Annotation indiquant qu'un paramètre de méthode doit être lié à une variable de modèle d'URI.
6-@PostMapping pour dire que c'est une mise à jour ou une création

7-@RequestBody pour récupérer les valeurs entrer par l'utilisateur.

3.1.1 Les services

Le package « pfe.configuration » est structuré comme suite :

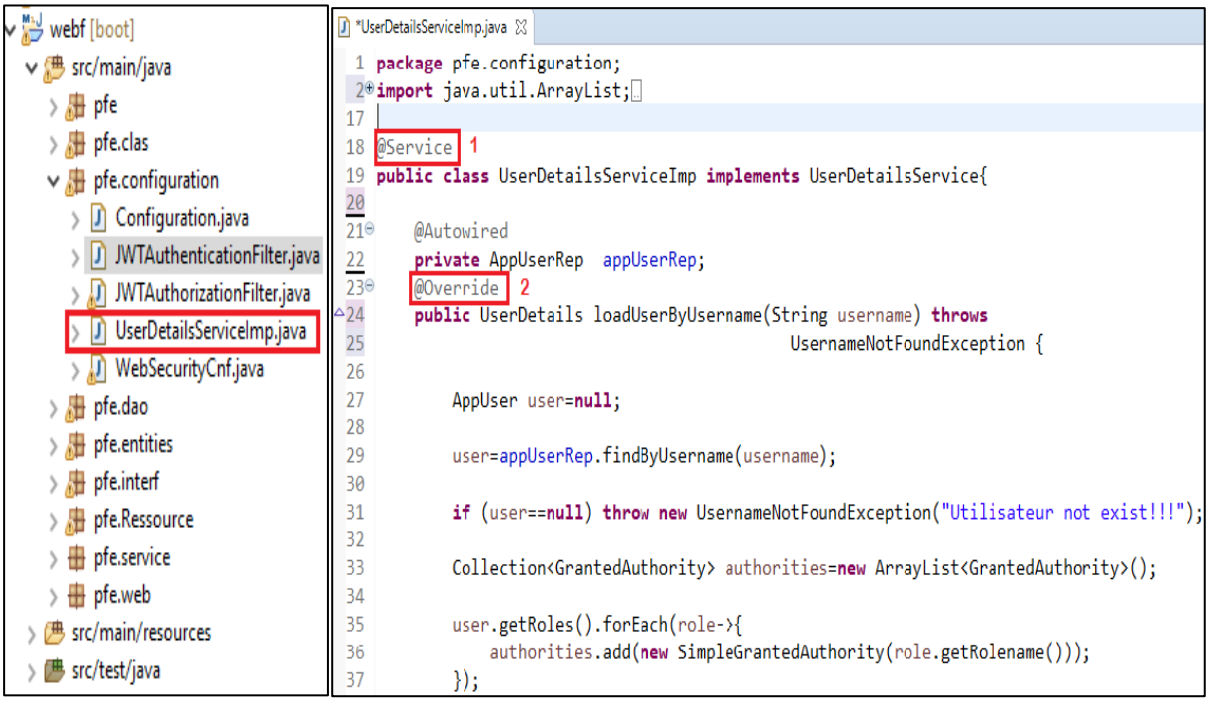


Figure 46 - Les Services du projet et un exemple du code source

1-@Service: Il est utilisé pour marquer la classe en tant que fournisseur de services est (elle fournit certaines fonctionnalités métier).

2-@Override: pour dire que c'est une implémentation de cette méthode

3. Étape 4 : Tests

Dans cette partie nous passons aux différents tests de nos modules implémentés précédemment

3.1 Configuration de LMtools

Comme abordé précédemment LMtools est l'interface utilisateurs graphique des utilitaires flexnet ci-dessous la configuration requise pour la gestion des licences :

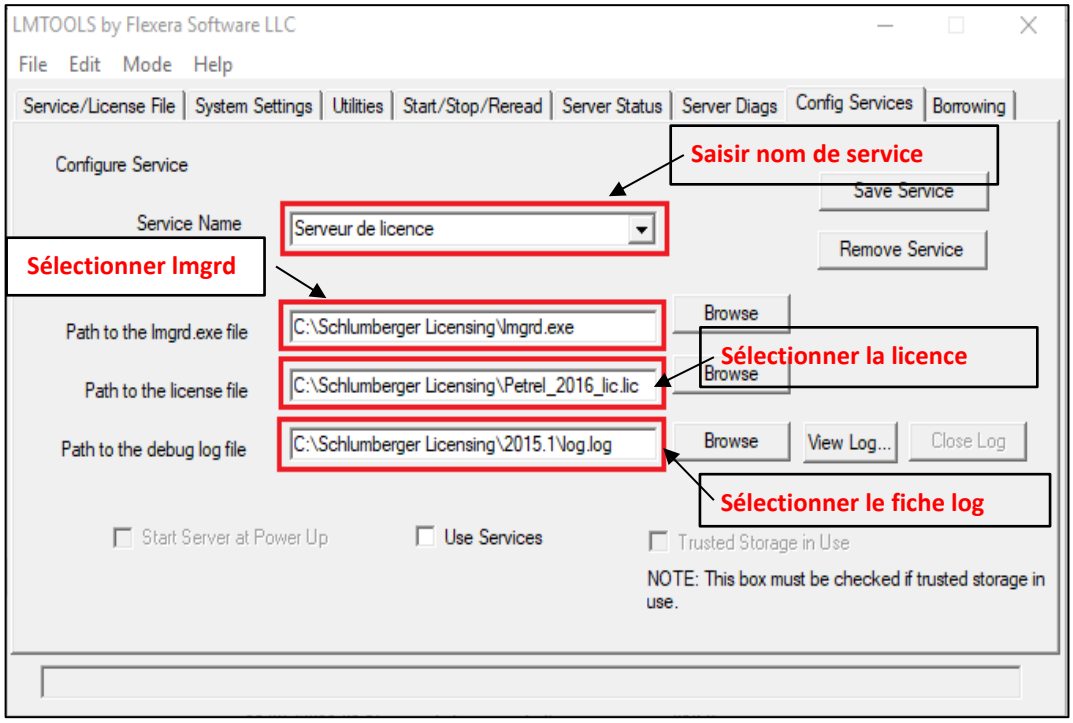


Figure 47 - Configuration de LMtools

Après l'installation de LMtools nous devons configurer le service qui va gérer les licences

De ce fait pour chaque logiciel on définit :

- Le nom du service qu'on veut implémenter.
- Sélectionner l'exécutable de la commande lmgrd.
- Sélectionner le fichier de licence.

- Crée un fichier log et l'implémenter afin de définir le chemin d'historisation des activités sur le serveur.

Après cette étape nous devons lancer le serveur qu'on vient de configurer pour permettre l'usage de la licence qu'on vient de lancer

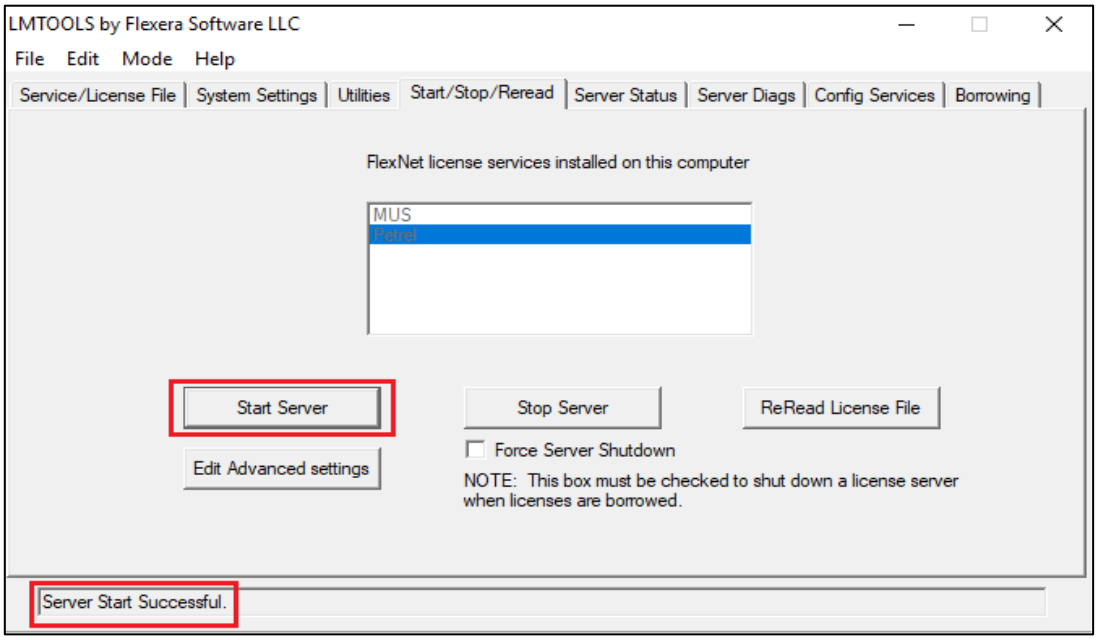


Figure 48 - Lancement du serveur depuis LMTOOLS

3.2 Le Fichier Licence

Comme déjà expliqué dans le chapitre 1 l'utilisation des licences et définit par un fichier de licence généré par les prestataires et fournis avec les logiciels ces dernier sont implémenté dans les serveurs et comportent plusieurs informations comme nous montre la Figure :

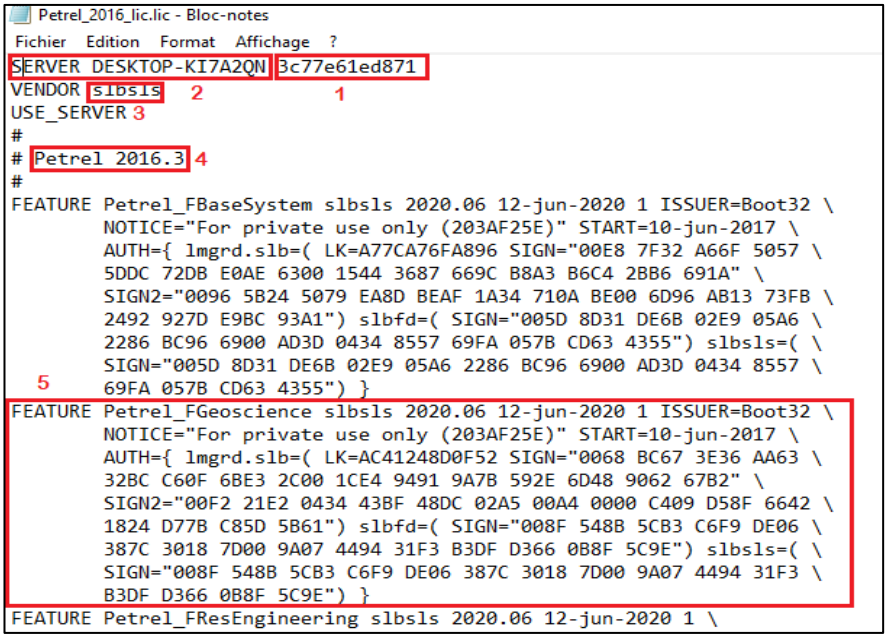


Figure 49 - Aperçu du fichier licence

1 : Nom du serveur (serveur local « localhost » pour notre démo)

2 : Adresse Mac du serveur

3 : Nom du Vendor (déjà expliqué en chapitre 1)

4 : Version du logiciel

5 : Clé de la licence

3.3 Etude de cas : Le logiciel metier petrel

Parmi les logiciels métier les plus utilisé par la division PED est l'outil « Petrel » qui est une plateforme logicielle utilisée dans le secteur de l'exploration et de la production de l'industrie pétrolière. Il permet à l'utilisateur d'interpréter les données sismiques, d'effectuer une bonne corrélation, de construire des modèles de réservoir, de visualiser les résultats de la simulation de réservoir, de calculer des volumes, de produire des cartes et de concevoir des stratégies de développement permettant de maximiser l'exploitation du réservoir. Le risque et l'incertitude peuvent être évalués tout au long de la vie du réservoir. Petrel est développé et construit par Schlumberger.

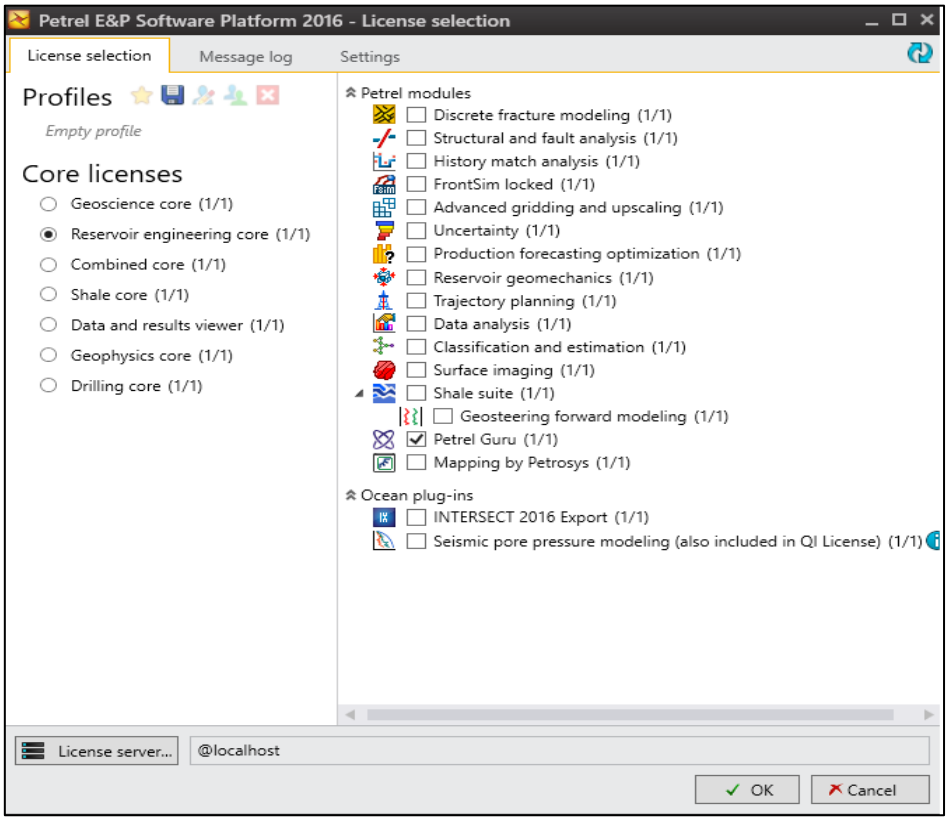


Figure 50 - Aperçu du Logiciel « Petrel »

Après le lancement de notre logiciel ce dernier reconnait bien les licences implémentées et nous affiche tous les composants de l'outil, nous allons par la suite lancer la commande lmgrd manuellement afin de voir les informations remontées sur la console concernant l'ouverture de l'outil et l'usage des licences.

3.4 Exécution de la commande Lmgrd

Le démon du gestionnaire de licences (lmgrd) gère le contact initial avec le Applications FLEX sous licence lm, en transmettant la connexion aux applications appropriées démon vendeur. Il démarre et redémarre également les démons fournisseurs. Contient des informations sur configuration et démarrage du démon du gestionnaire de licences dans votre environnement.

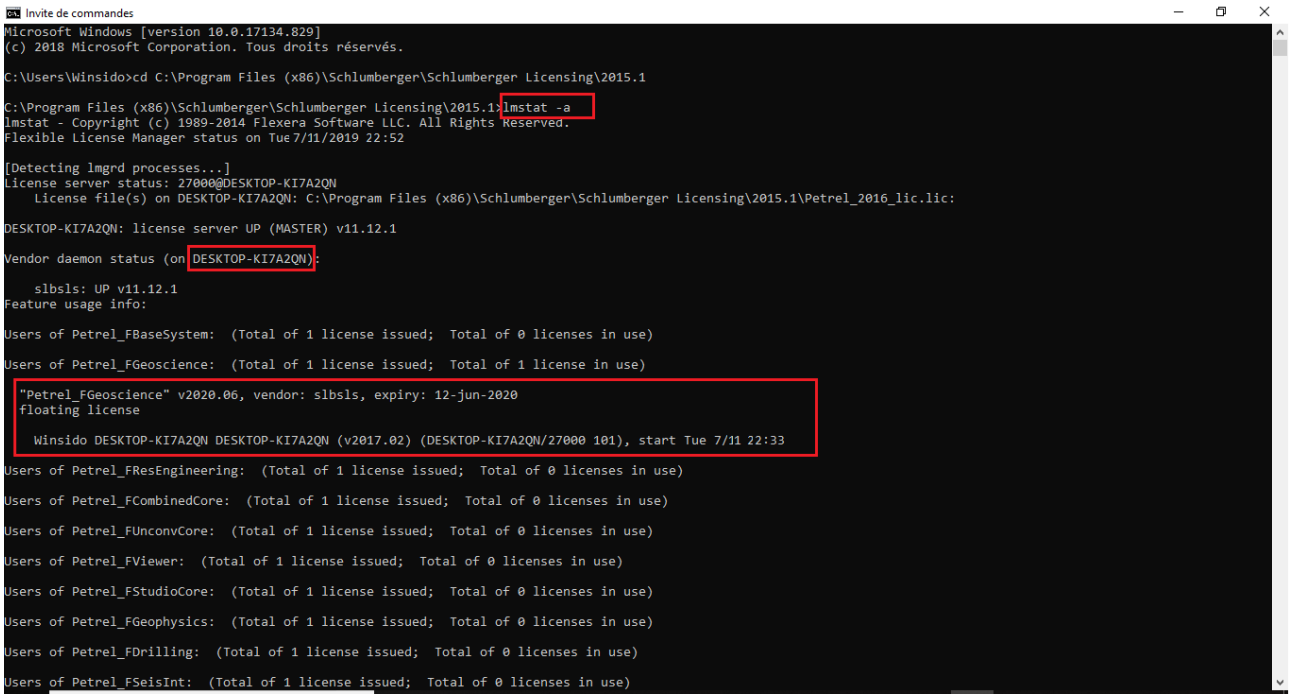


Figure 51 - Console de la commande « lmgrd »

Nous constatons que sur la figure 51 on voit bien l'heure d'accès à la licence, le nom de la machine qui utilise la licence avec le nom de l'utilisateur et le nom des licences utilisées.

Nous allons passer maintenant au fichier LOG qui est généré automatiquement par la commande « lmgrd » comme expliqué auparavant.

3.5 Le fichier LOG

Le fichier journal de débogage contient des messages d'état et d'erreur utiles au débogage, le serveur de licences. Une partie de la sortie du journal de débogage décrit les événements spécifiques à lmgrd et une partie de la sortie du journal de débogage décrivent des événements spécifiques à chaque démon vendeur.

Ci-dessous un aperçu du fichier LOG généré après le lancement de notre logiciel et l'éxécution de la commande « lmgrd »

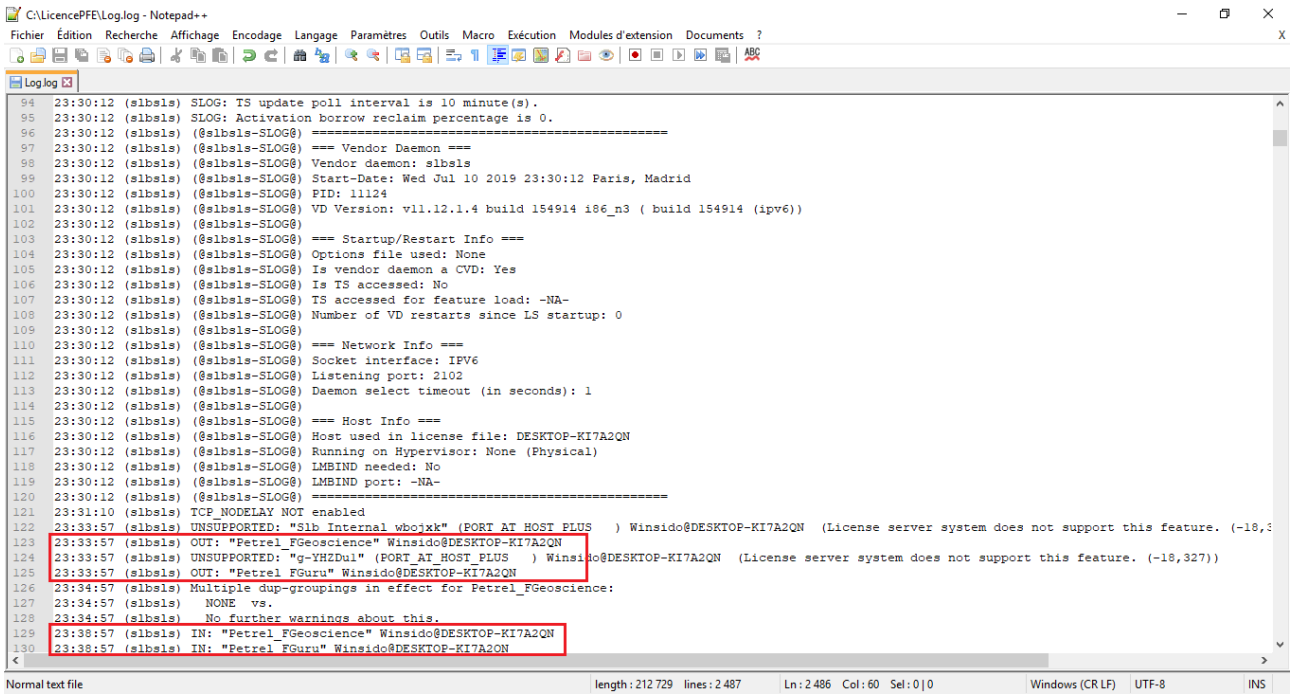


Figure 52 : Fichier LOG généré

Nous remarquons que la ligne encadrée correspond aux informations remontées par la commande « lmgrd » dans la console ou on retrouve exactement les mêmes informations remontées dans la console.

3.6 Visibilité Base de données

Après le lancement de notre application cette dernière lit le fichier log qui traite les informations écrites sur le fichier et les stocks dans une base de données non relationnelle dans la figure ci-dessous on peut voir comment les informations lues sur le fichier log apparaissent dans notre BDD depuis MongoBD de façon claire et lisible, et nous montre bien les informations de notre démo qui apparaissent sur les dernières lignes,

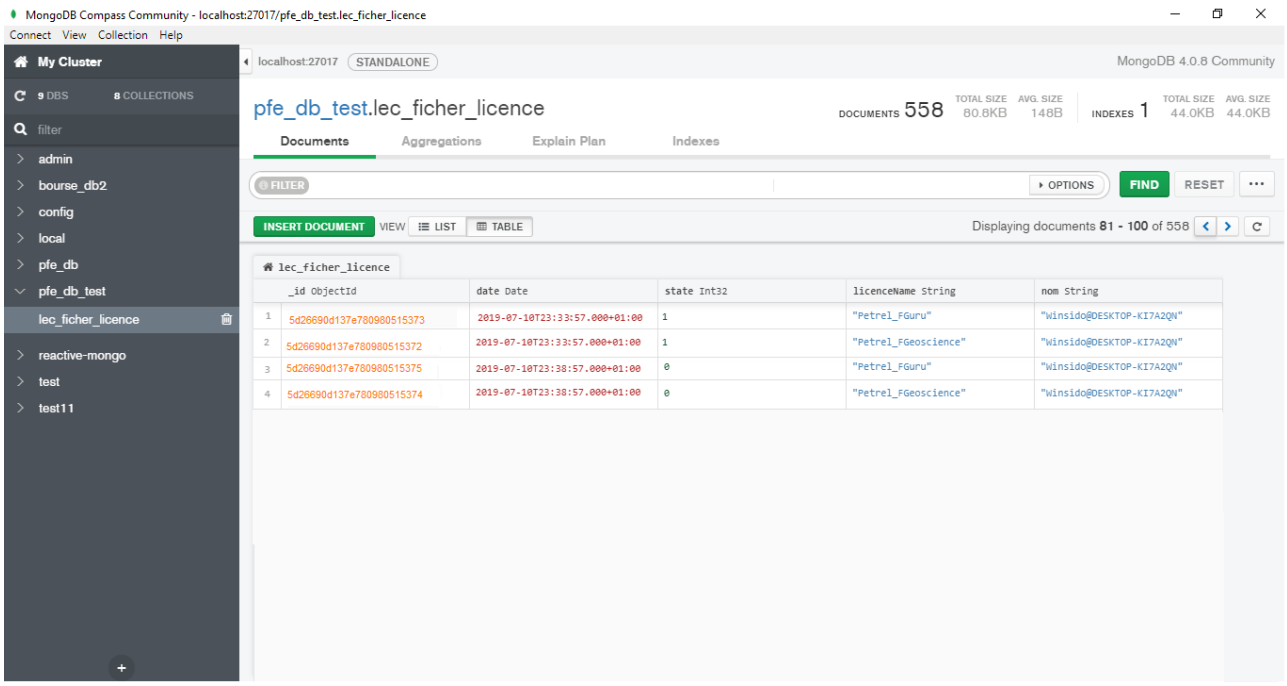


Figure 53 - Aperçu de la Base de données

Nous avons la date et l'heure d'ouverture de la licence, le statut (1 correspond à l'ouverture / 0 correspond à la fermeture), le nom de licence utilisée, la session de l'utilisateur et la machine utilisée.

Nous allons passer ensuite à l'affichage de ces informations depuis notre lien http pour vérifier la liaison de notre programme avec l'interface web, de ce fait nous allons utiliser un API « Restlet Client » que nous aborderons dans ce qui suit.

3.7 Restlet Client (REST API Testing)

Restlet Client (anciennement appelé DHC) permet d'interagir avec les services REST. Il apporte de nombreuses fonctionnalités qui améliorent l'utilisation des acteurs finaux, il gagne un temps précieux lors du débogage des appels HTTP ou en partageant les requêtes avec d'autres utilisateurs.

Restlet Client, permet de créer des scénarios complexes qui imitent l'utilisation réelle de notre API (combiner plusieurs demandes d'API, réutiliser les données des réponses précédentes, etc.). Il permet d'effectuer de nombreux tests de réponse d'API, notamment sur la valeur des en-têtes et des parties du corps ou temps de réponse. [52]

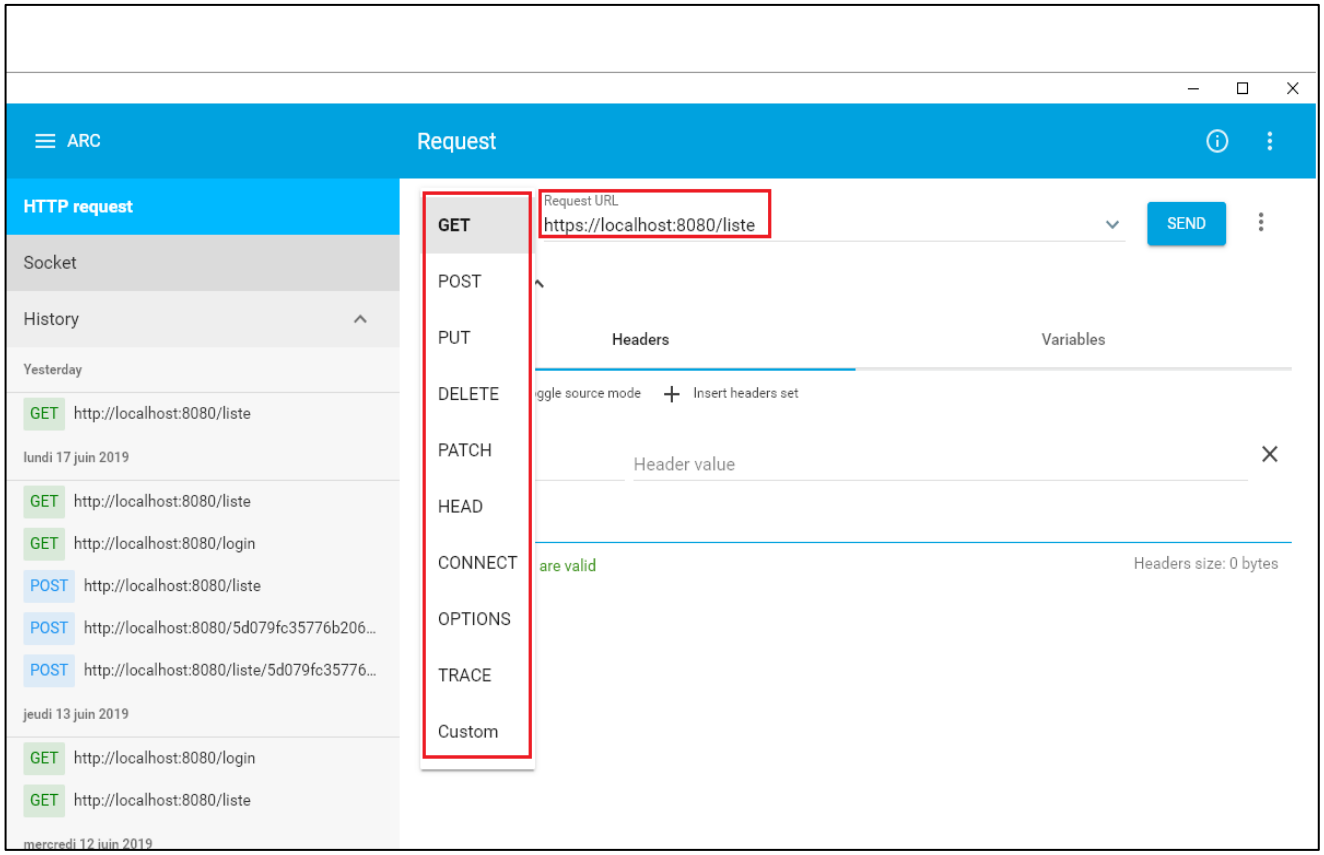


Figure 54 - L'interface « Restlet Client »

« Restlet Client » comporte plusieurs méthodes, on s'intéresse aux principales méthodes suivantes :

- GET= pour récupérer
- POST=pour la mise à jour
- PUT= pour créer
- DELETE=pour supprimer

Test de l'affichage du projet via Rest API

Après avoir choisi la méthode qui convient, on spécifier le chemin (path) de la requête qu'on veut effectuer (exemple : http://localhost:8080/liste => c'est pour dire qu'on va effectuer une requête sur les projets en général.

http://localhost:80/projects => c'est pour dire qu'on va effectuer une requête sur le projet qui a un id=1.)

Après avoir spécifier le chemin et envoyer la requête (=> Send) le résultat va être afficher dans le 'BODY' (avec un response 200 en cas de succès).

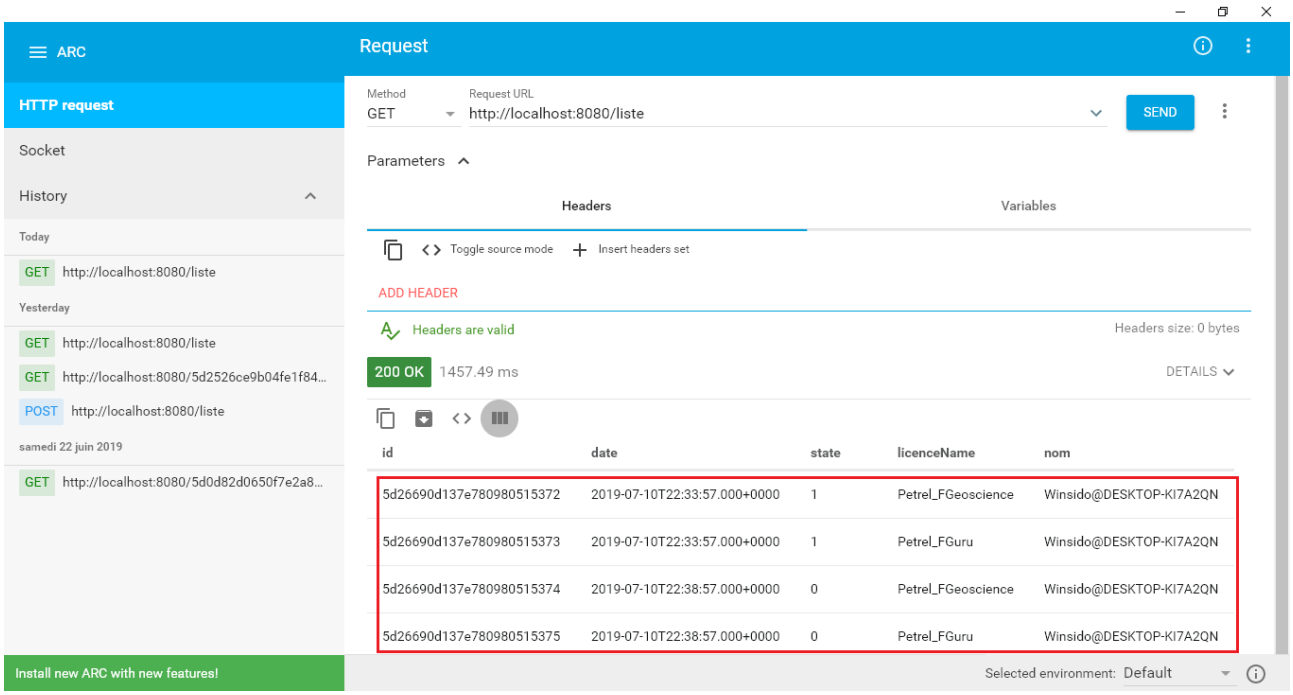


Figure 55 - Test de l'affichage de notre projet

On remarque la liaison est bien active car depuis notre navigateur on peut visualiser les informations traitées par notre programme et stocker dans notre base donnée.

3.8 Lancement de l'application

Pour le lancement de l'application on doit introduire le lien du serveur de l'application web et spécifier le numéro de port, après cette étape nous trouvons une page d'authentification comme nous montre la Figure suivante :

3.8.1 Interface d'authentification

La figure ci dessous représente l'interface d'authentification de notre application web

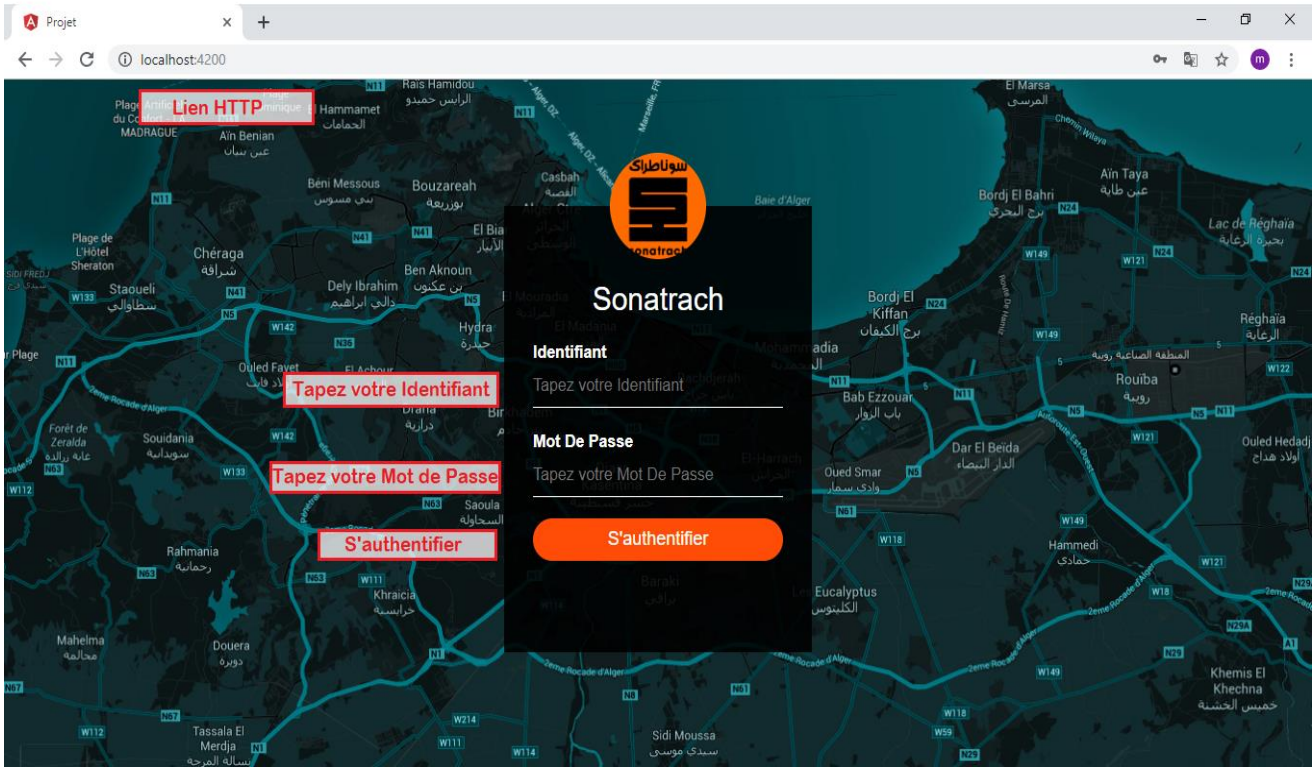


Figure 56 - Interface d'authentification

L'utilisateur de l'application doit saisir son nom d'utilisateur et son Mot de passe qui sont générés de manière automatique en se basant sur une authentification LDAP (comme vu précédemment) qui gère la génération des ID session et des MDP automatiquement.

3.8.2 Interface de l'application

Après l'authentification l'utilisateur accès à notre application qui se compose de plusieurs onglets :

1 : Affichage du tableau d'utilisations des licences

2 : Affichage du diagramme en batons d'utilisation des licences

3 : Affichage du diagramme en cerle d'utilisation des licences

4 : Table de Synthèses des licences détenus par la division

5 : Aide a la d'écision (Statisques d'utilisations des licences)

6 : Onglet de manipulation des utilisateurs des licences

7 : Onglet de manipulation des Serveurs

8 : Onglet de manipulation des licences dans nos serveurs

9 : Bouton pour se déconnecter

Figure 57 – Onglets de L'application

3.8.3 Affichage du tableau de bord

Ce qui représente l'affichage en temps réel des utilisateurs entrants/sortants qui utilise les licences à l'instant T

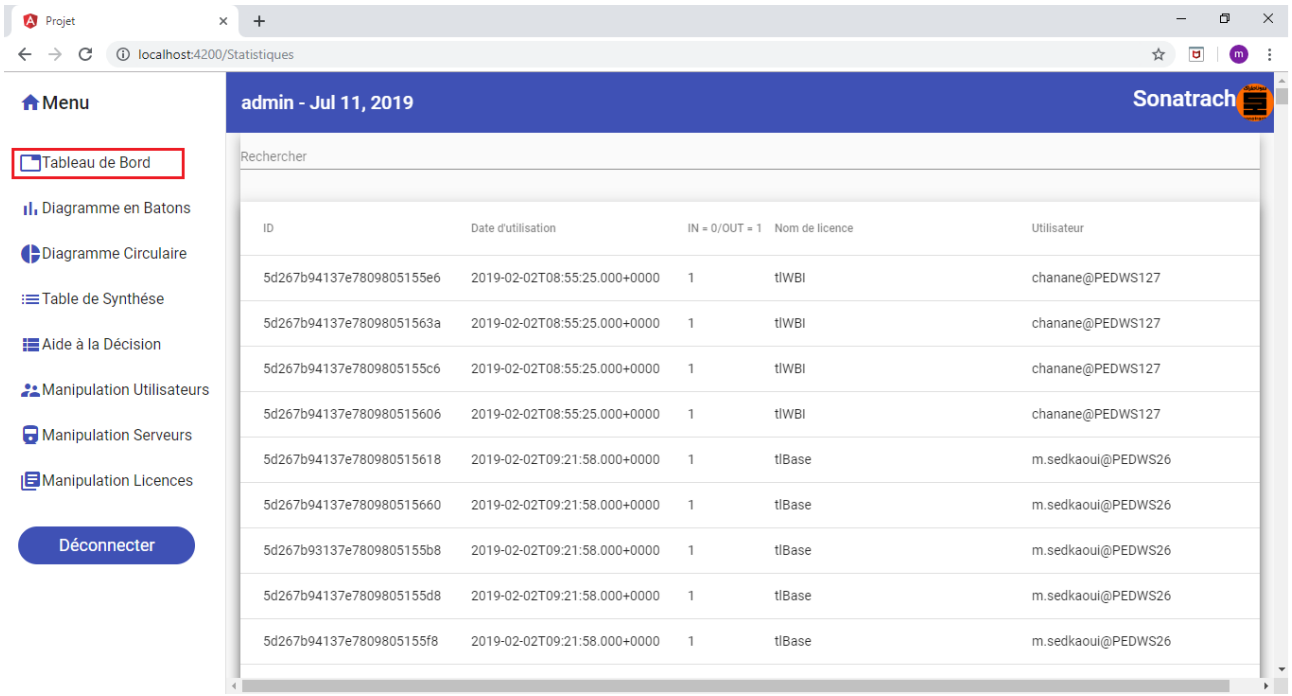


Figure 58 - Tableau de bord

On remarque que sur notre tableau de bord on voit bien les informations de l'usage des licences qui remontre de façon très lisible et claire pour le gestionnaire ou l'administrateur de l'application.

3.8.4 Affichage du diagramme en batons

L'utilisateur peut avoir une visibilité en forme de graphe qui apparait sous forme de diagramme en batons comme nous illustre la Figure ci après :

Figure 59 - Aperçu du diagramme en batons

Ou comme mentionnée auparavant une visibilité en forme de graphe en cercle peut être sélectionner depuis les onglets de l'application

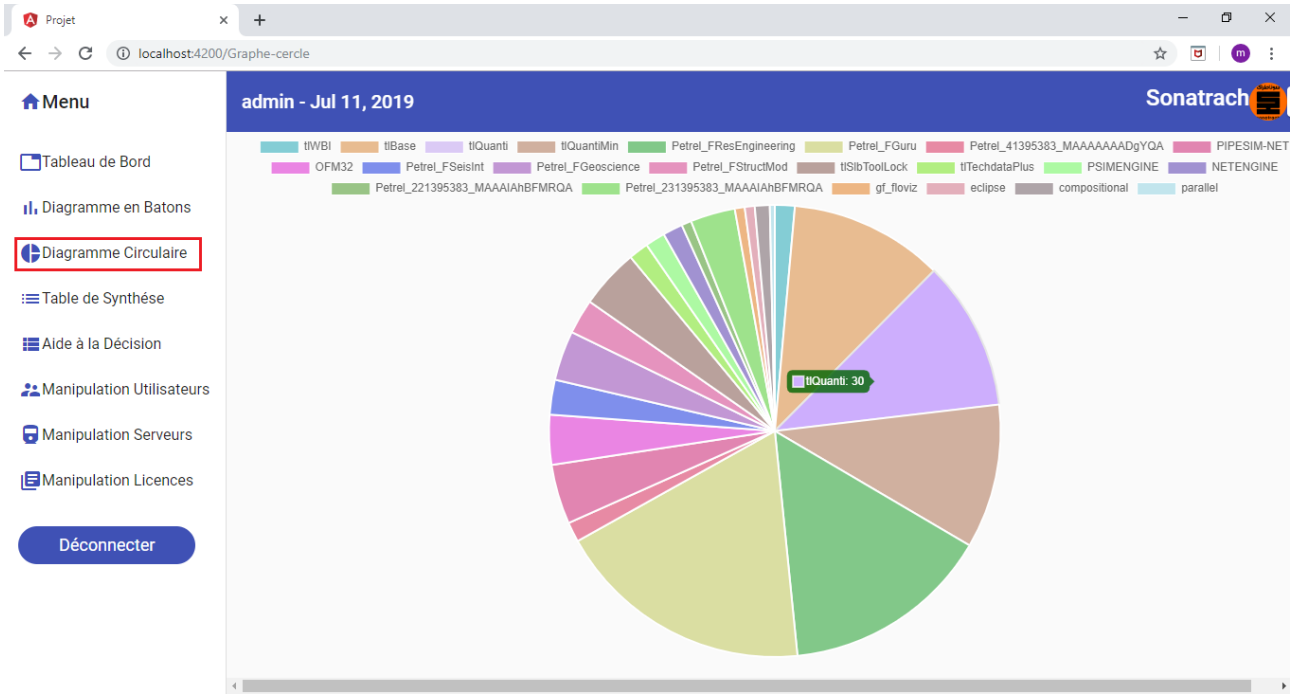


Figure 60 - Diagramme en Cercle

3.8.5 Table de synthèse

Dans notre application nous avons rajouté un onglet qui contient la synthèse des licences détenus par la division pour une meilleure visibilité sur les volumes de ces dernières :

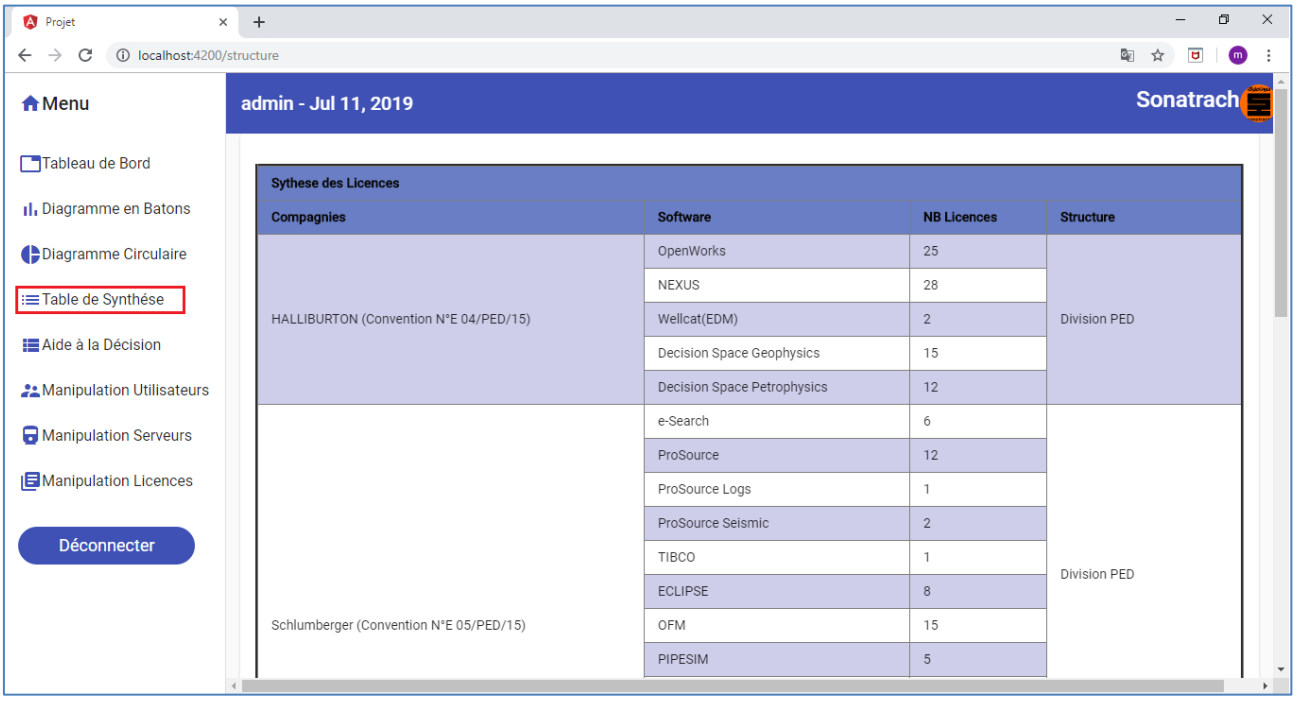


Figure 61 - Table de synthèse

3.8.6 Manipulation des Utilisateurs

La figure ci dessous montre l'onglet de manipulation des utilisateurs et ses différentes fonctionnalitées

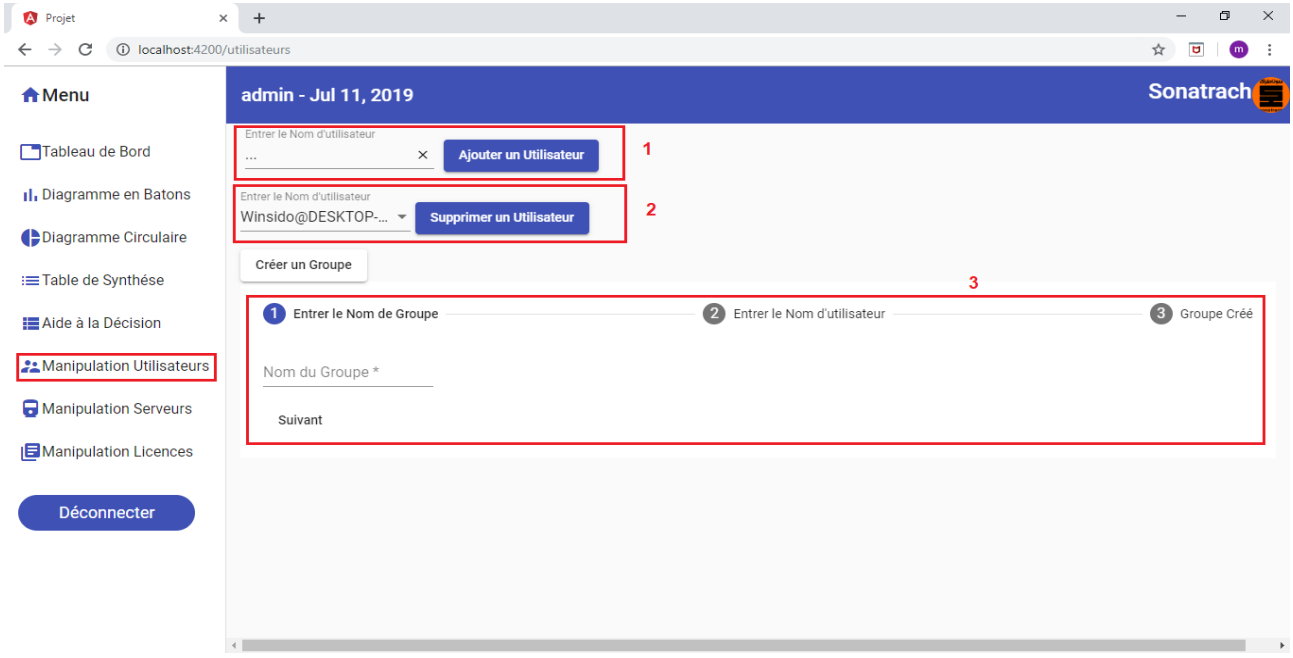


Figure 62 - Manipulation des utilisateurs

Fonctionnalitées :

1 : Ça permet d'ajouter un nom d'utilisateur en tapant son identifiant.

2 : Selectionner le nom d'utilisateur parmis ceux enregistrer dans la base de données.

3 : Donner un nom au groupe qu'on veut créer puis on affecte des utilisateurs au groupe puis on valide.

3.8.7 Manipulation des Serveurs

La figure ci dessous démontre l'onglet de manipulation des serveurs et ses différentes fonctionnalitées

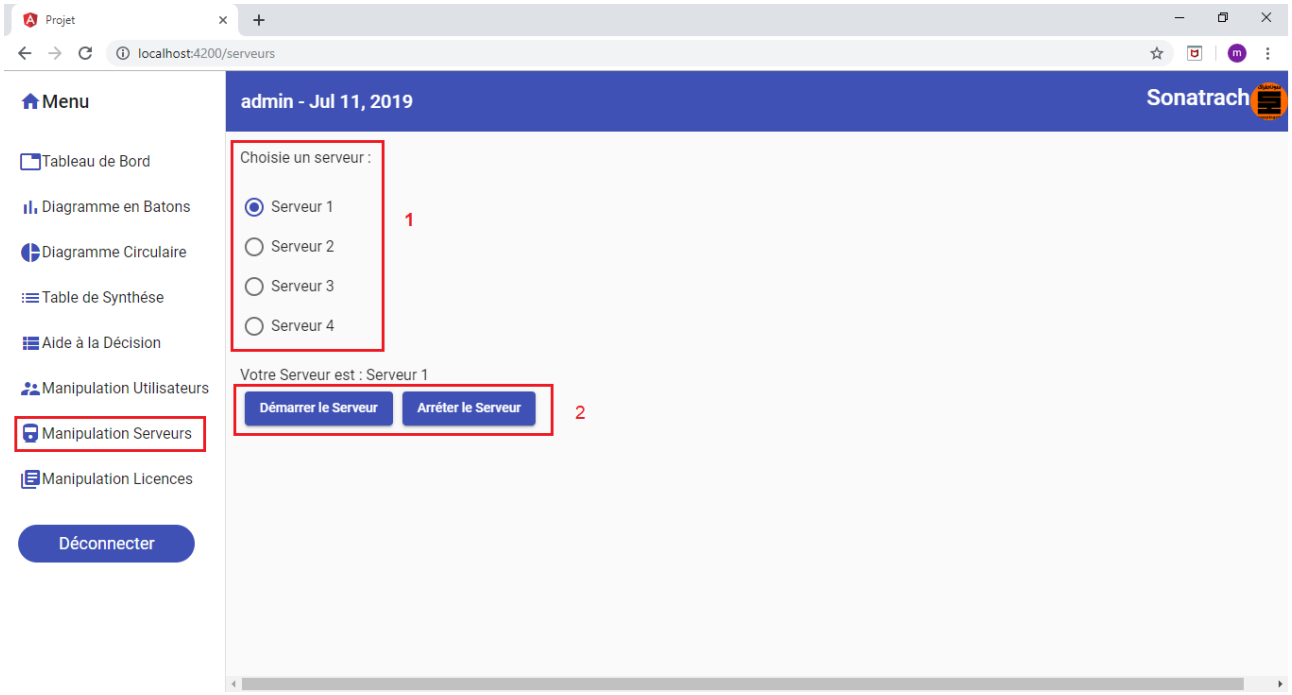


Figure 63 - Manipulation des serveurs

Fonctionnalitées :

1 : Ça permet de selectionner le serveur sur lequel on veut appliquer l'action.

2 : Choisir entre Démarrer ou Arréter un des serveurs des licences.

3.8.8 Manipulation des Licences

La figure ci dessous montre l'onglet de Manipulation des licences et ses différentes fonctionnalités

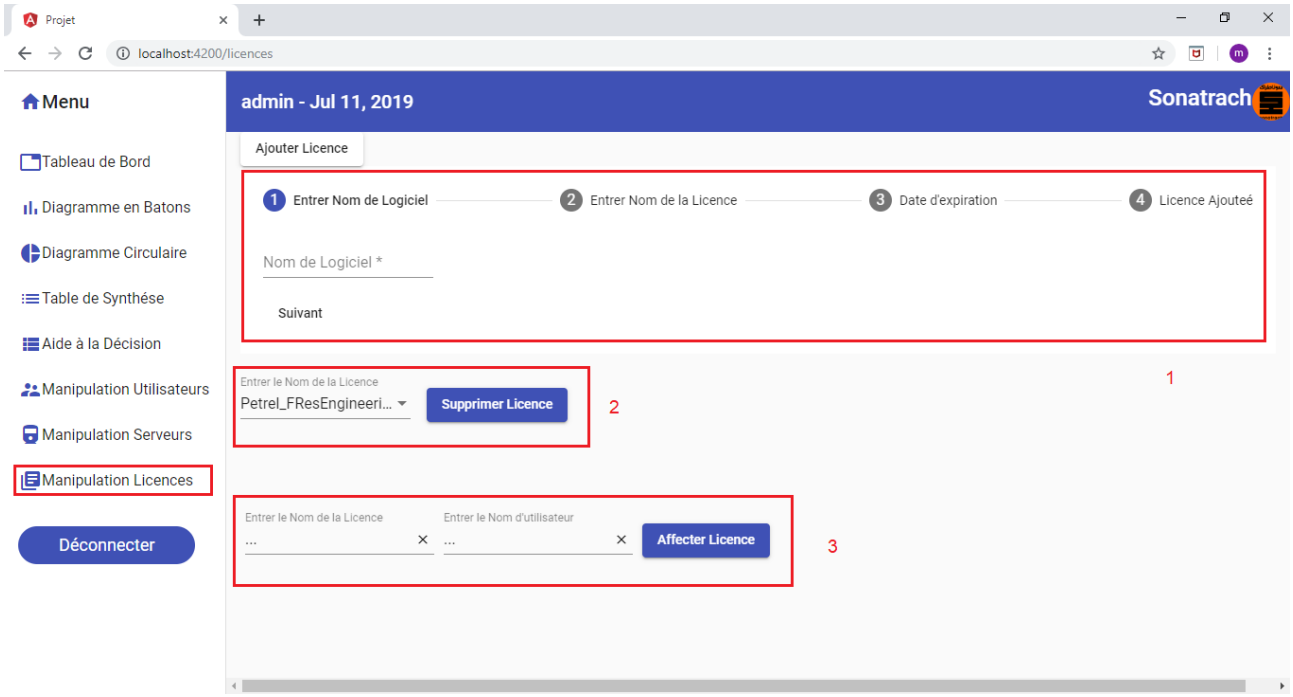


Figure 64- Manipulation des Licences

Fonctionnalitées :

1 : Ça permet d'ajouter le nom du logiciel aux quel appartient la licence, saisir le nom de la licence et préciser sa date d'expiration.

2 : Sélectionner parmis la liste des licences celle à supprimer.

3 : Choisir le nom de la licence qu'on veut affecter, et préciser parmis la liste déroulante le nom de l'utilisateur vers qui on veut affecter cette licence.

4. Etape 5 : Exploitation

Dans notre application nous avons intégré un onglet d'aide à la décision qui nous permet d'avoir une visibilité sur l'historique d'usage des licences par utilisateur comme le montre la figure ci-dessous :

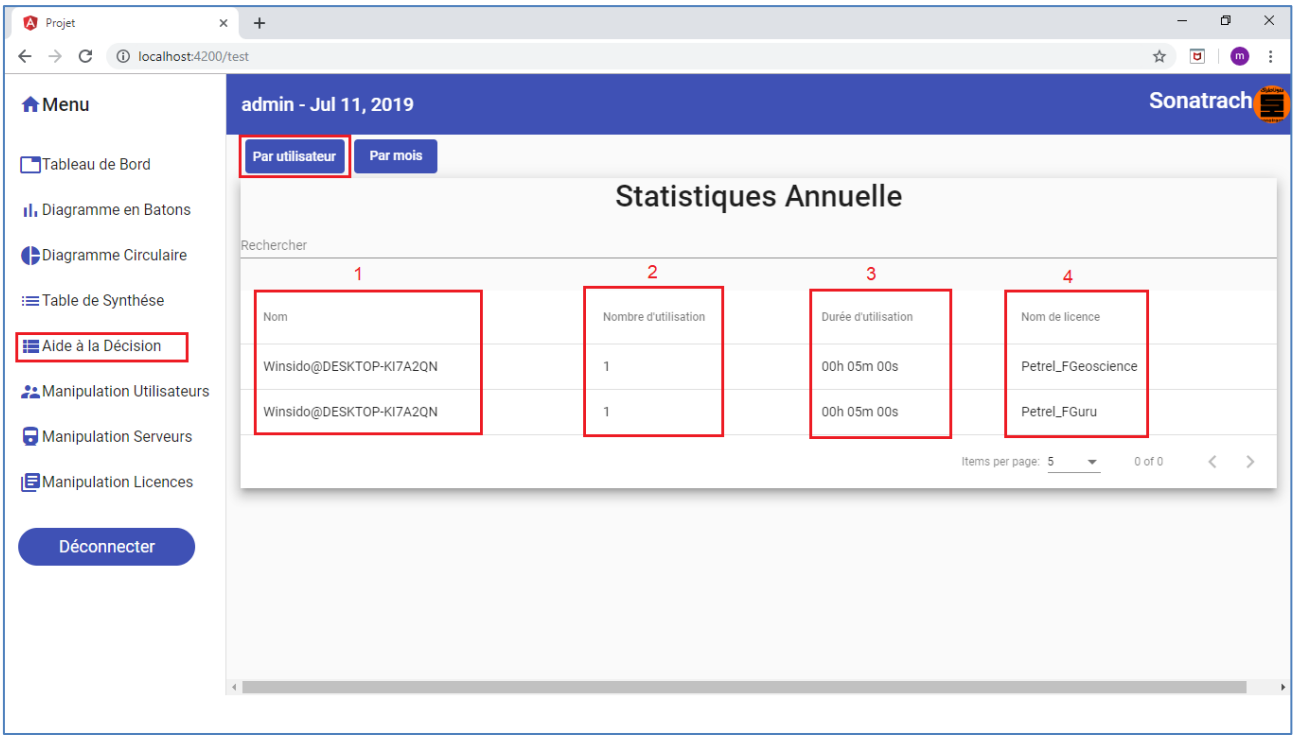


Figure 65- Statistiques Annuelle par Utiilsateur

Description :

1 Nom d'utilisateur.

2 Nombre d'utilisation de chaque licence.

3 Durée d'utilisation de chaque licence.

4 Nom de la licence.

Aide à la décision :

Cette partie s'inscrit dans l'optique de prise des decisions sur l'achat des licences (selon le DMN vu dans la modélisation décisionnlle « étapes 2 »), elle sera principalement exploitée par les preuneurs de décisions (divisionnaire selon notre BPMN vu dans la modélisation Métier « étapes 2 »), elle se base sur ces informations liées à :

- Nombre d'utilisateurs d'une licence.
- Nombre d'occurrence d'une licence.
- Durée de vie.
- Temps d'utilisation d'une licence.
- La synthèse de toutes les licences possédées

Le tableau dans la figure 66 représente les statistiques d'utilisation des licences mensuelle

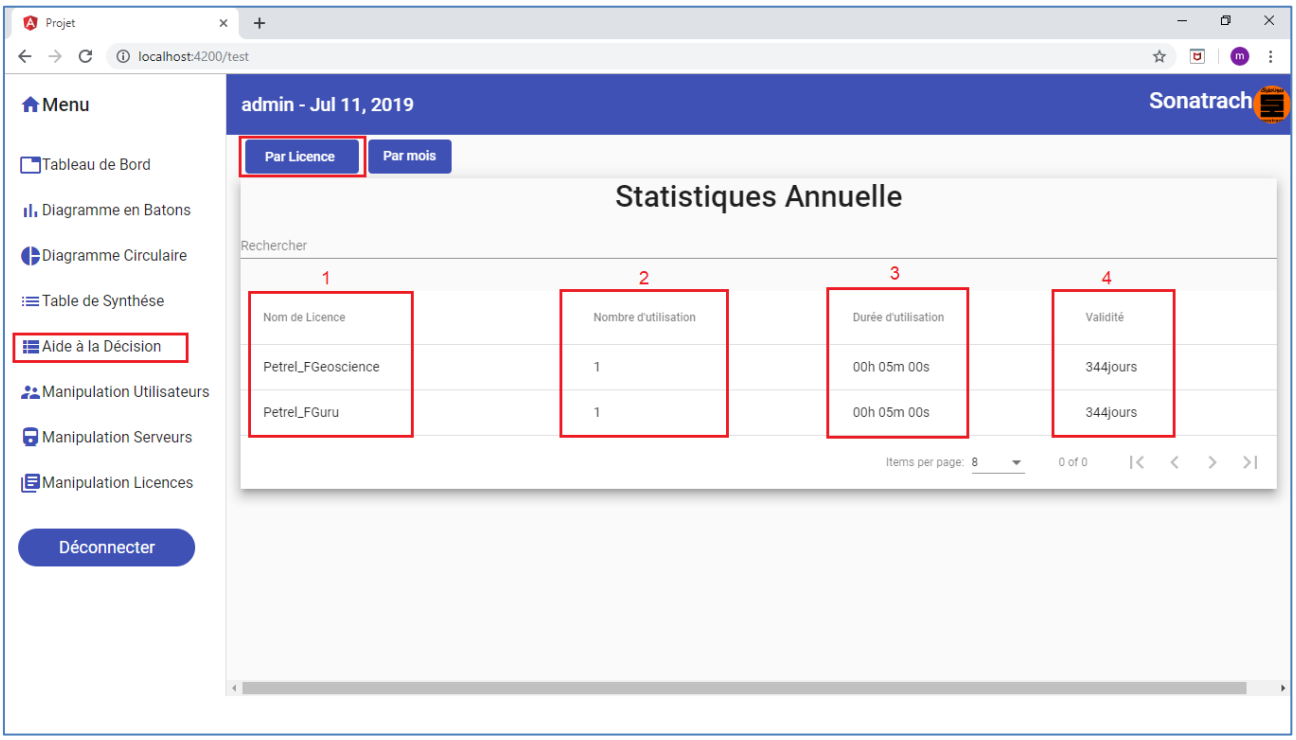


Figure 66: Statistiques Annuelle par Licence

Description :

5 Nom des Licences

6 Nombre d'utilisateur de chaque licence

7 Durée d'utilisation de chaque licence

8 Validité de chaque licence

Le tableau dans la figure 67 représente les statistiques d'utilisation des licences mensuelle

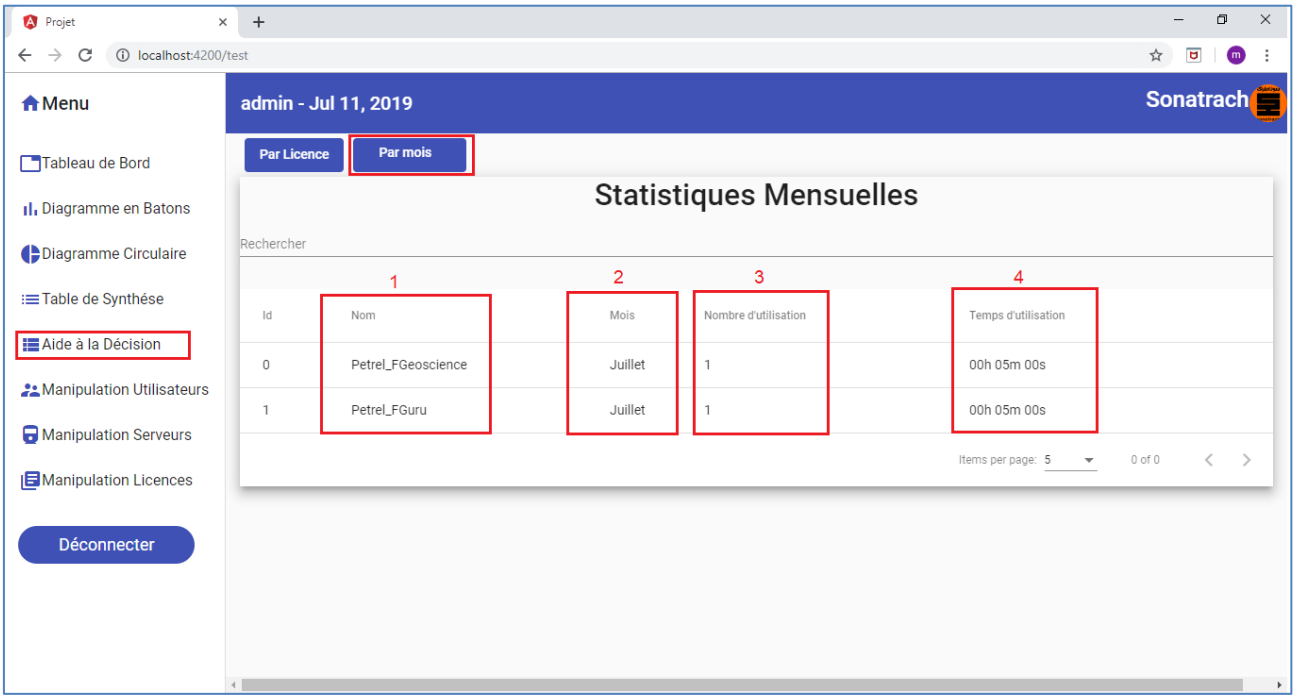


Figure 67 - Statistiques Mensuelle des licences

Description :

1 Nom des Licences.

2 Mois d'utilisation de la licence.

3 Nombre d'utilisation (occurrence) de chaque licence.

4 Temps d'utilisation de chaque licence.

5. Conclusion

Au cours de ce chapitre, nous avons réalisé la partie web qui assure le suivi des licences et d'aide à la décision, en respectant toutes les étapes du cycle de développement, à savoir : l'implémentation, les tests et l'exploitation des données générées par le traitement du fichier log de licences.

Par ailleurs, le chapitre consacré à la conception, a permis de mettre en exergue la logique de développement. Les tests ont été effectués sur un cas réel d'utilisation des licences du logiciel d'interprétation et de modélisation géosciences « Petrel », où nous avons dérouler le processus d'exploitation des licences jusqu'à l'affichage des taux et données d'utilisations, à travers le tableau de bord mis en place.

Conclusion Générale

La diversité et la complexité des études géosciences obligent les entreprises en conquête des marchés pétroliers, à se doter des logiciels adéquats de modélisation, d'interprétation et de simulation des gisements et réservoirs pétroliers.

Désormais, l'utilisation de ces logiciels métiers impliquent des budgets colossaux liés à l'utilisation de leurs licences. A cet effet, une gestion minutieuse de ces licences s'impose afin d'optimiser les budgets et les coûts d'acquisition.

C'est dans cette optique, que notre projet de fin d'études apporte à la Division PED, l'outil nécessaire à la gestion et le suivi des licences logicielles, et d'aide à la décision.

La solution que nous avons proposée, s'articule sur des concepts et technologies nouvelles à savoir la programmation réactive et les bases de données non sql, pour de meilleures performances dans les traitements back end et le système de stockage (gestion) des données.

L'environnement Flexnet a été exploité en vue de l'extraction des données et informations utiles sur l'utilisation des licences, qui aideront par la suite à la prise de décision pour l'achat, le renouvellement ou l'abondons d'une licence (ou plusieurs).

L'architecture de la solution est fondée sur les statistiques élaborées par le traitement backend et les tableaux de bords front end qui servent à la visualisation des résultats.

A présent, notre solution répond pratiquement aux objectifs énoncés au début du stage.

Lors de l'élaboration de notre projet nous avons rencontré quelques difficultés liées au manque de documentation sur les technologies utilisées vu qu'elles sont très récente ou aussi quelques problématiques techniques qu'on a pu détourner ou solutionnée.

Notre projet n'est pas arrivé à ses fins car des perspetives d'amélioration de l'application sont à prévoir notament sur la partie gestion des utilisateurs et la gestion des serveurs ou nous souhaiterons optimiser de mieux ses fonctions en implémentant de nouvelle fonctionnalités déjà abordé dans la partie modélisation telle que la création de groupe d'utilisateurs, suppression ou ajout d'utilisateurs, lancement et arrèt des serveurs ou aussi implémenter une authentification via le processus LDA

Annexes

Présentation de la société SONATRACH

Sonatrach est une compagnie nationale algérienne d'envergure internationale c'est la clé de voute de l'économie algérienne.

Elle a été créée, le 31 décembre 1963, pour répondre au souci d'une mobilisation des ressources du rentré pétrolier perçue très tôt comme un élément moteur dans le développement de l'Algérie, au fil des années, elle devient un puissant élément d'intégration nationale et de stabilité, de développement économique et social.

Le groupe pétrolier et gazier Sonatrach intervient dans l'exploration, la production, le transport par canalisation, la transformation et la commercialisation des hydrocarbures et de leurs dérivés. Elle se développe également dans les activités de pétrochimie, de génération électrique, d'énergies nouvelles et renouvelables, de dessalement d'eau de mer et d'exploitation minière.

Sonatrach opère en Algérie et dans plusieurs régions du monde, notamment en Afrique (Mali, Tunisie, Niger, Libye, Égypte, Mauritanie), en Europe (Espagne, Italie, Portugal, Grande-Bretagne, France), en Amérique latine (Pérou) et aux États-Unis.

L'entreprise emploie 41 204 salariés (120 000 avec ses filiales), génère 30 % du PNB de l'Algérie. En 2005, sa production était de 232,3 millions de TEP, dont 11,7 % (24 millions de TEP) pour le marché intérieur.

En 2009, son chiffre d'affaires s'élevait à 77 milliards US$. Par le chiffre d'affaires, Sonatrach est de loin la première compagnie africaine, toutes activités confondues. Elle devance la filiale sud-africaine de l'assureur Old Mutual, classée deuxième. Sonatrach est le 12^{e} groupe pétrolier au niveau mondial, le premier en Afrique et dans le Bassin méditerranéen, le 4^{e} exportateur de GNL, le 3^{e} exportateur de GPL et le 5^{e} exportateur de gaz naturel. [53]

Présentation La division Petroleum Engineering Developement PED-SONATRACH

PED ou Petroleum Engineering and Development est une division qui a une importance capitale au niveau de la Sonatrach. Elle constitue un moteur clé dans le suivi de l'avancement des projets pétroliers et gaziers et la gestion des collaborateurs et des infrastructures. [53]

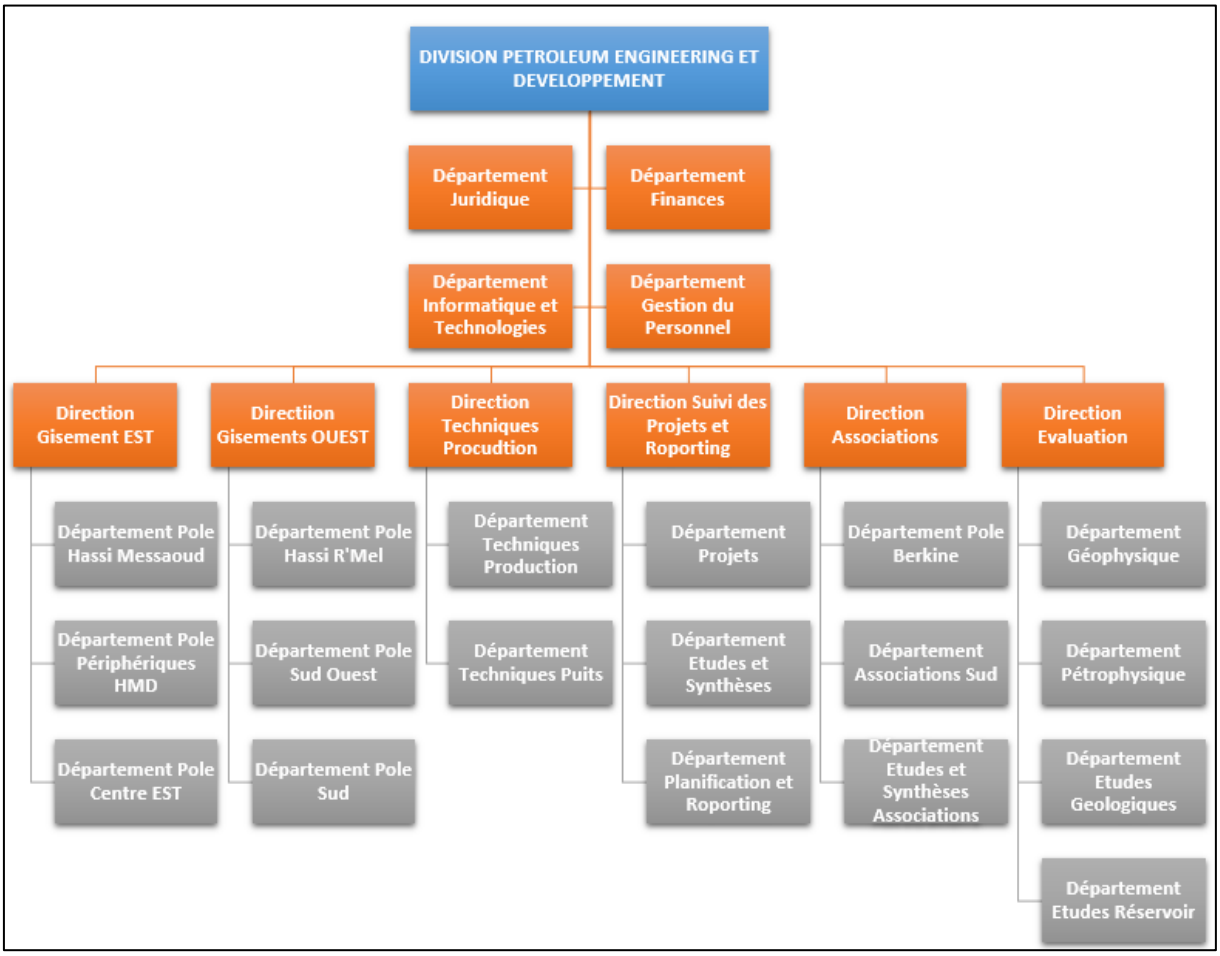


Figure 68- Représentation de la division PED

Bibliographie

[1] P.-P. LEMYRE, *Les logiciels libres sous l'angle de la responsabilité civile*, Montréal, Maîtrise en droit des technologies de l'information Université de Montréal, 2013.

[2] P.Lemeyre, "Types de licences de logiciels," 05 03 2019. [Online]. Available: http://isn-2.e-monsite.com/pages/types-de-licences-de-logiciels.html.

[3] D. H.Biscay, *ACQUISITION , MAINTENANCE DE LOGICIELS*, Hydra, PED, 2018.

[4] "FLEXlm End Users Manual," April 2000. [Online]. Available: http://www.globetrotter.com.

[5] *Flexlm End User*, 2003.

[6] N. Halbwachs, *Synchronous programming of reactive systems*, Kluwer Academic Pub, 1993.

[7] P. C. P. R. Nicolas Halbwachs, *and Daniel Pilaud. The synchronous data- flow programming language LUSTRE*, 1991, p. 1305–1320.

[8] G. G. Gérard Berry, *The Esterel synchronous programming language: Design, semantics, implementation*, 1992, p. 87–152.

[9] F. Boussinot, *Java fair threads*, INRIA, 2002.

[10] A. M. T. Guido Salvaneschi, "Reactive Programming: a Walkthrough," 2015.

[11] M. M. Guido Salvaneschi, *Towards Reactive Programming for Object-oriented*, Software Technology Group.

[12] A. Tabard, *Programmation Réactive*, Lyon, Université Claude Bernard, 2018.

[13] "Server-SentEvents,"03February2015.[Online].Available: https://www.w3.org/TR/eventsource/.

[14] G. Laurent, *Techniques audiovisuelles et multimédias*, 3e ed., 2012.

[15] J. L. H. David A. Patterson, *Computer Organization and Design ARM Edition*, 2016.

[16] F. B. N. S. e. B. C. Soutou, *L'émergence du Big Data et des bases NoSQL,* Rudibruchez, Ed., Paris: Groupe Eyrolles, 2015.

[17] M.Rouse,“techtarget,”mars2017.[Online].Available: https://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL. [Accessed 06 02 2019].

[18] F. Hadrien, *SQL, NoSQL, NewSQL stratégie de choixMémoire présenté en vue de l’obtention du diplôme Bachelor HES,* Genève, Haute École de Gestion (HEG-GE), 2015.

[19] L. Heinrich, *Architecture NoSQL et réponse au Théorème CAP*, Genève, Haute École de Gestion (HEG-GE), 2012.

[20] A. G. PIAZZA, *NoSQL Etat de l’art et benchmark,* Genève, Haute École de Gestion de Genève (HEG-GE), 2013.

[21] A. w. H.Mattallah, *Etude comparative des bases de données SQL et la nouvelle génération NewSQL MySQL vs VoltDB,* Tlemcen, Université Abou Bakr Belkaid, 2017.

[22] A. A. S. B. Fatima-Zahra Benjelloun, *LE QUOI? LE POURQUOI ET LE COMMENT,* Kénitra, Université Ibn Tofail, 2015.

[23] “GeeKsforGeeKs,” [Online]. Available: https://www.geeksforgeeks.org/difference-between-sql-and-nosql/. [Accessed 02 02 2019].

[24] X.Biseul,“JDN,”09062017.[Online].Available: https://www.journaldunet.com/solutions/dsi/1194284-base-nosql-laquelle-choisir-pour-quels-besoins/. [Accessed 03 02 2019].

[25] J.-P. B. M. B. Thierry Biard, *DMN (Decision Model and Notation) : De la Modélisation à l’Automatisation des Décisions,* Toulouse: <hal-01532837>, 2017.

[26] T. D. Gero Decker, *Decision Modeling Using DMN 1.0,* Nürnberger, 2015.

[27] OMG, “Decision Model and Notation,” January 2019. [Online]. Available: https://www.omg.org/spec/DMN.

[28] E. J.Vanthienen, *Decision Tables: Refining theConcept and a Proposed Standard,* 1994.

[29] Codasyl, *A modern appraisal of decision tables,* 1982.

[30] OpenRules, “Decision Model and Notation (DMN) Supporting Tools.,” 2016. [Online]. Available: http://openjvm.jvmhost.net/DMNtools/.

[31] J. D. M.Kassis, *Développement d’une méthode de conduite de changement,* Université de Tunis El Manar institut supérieur d’informatique, 2011.

[32] J. D. Costa, *BPMN 2.0 pour la modélisation et l'implémentation de dispositifs pédagogiques orientés processus,* Genève , Université TECFA, 2014.

[33] A. Shraideh, *Integrating a business process analysis and optimization step using BPMN model,* Lille, Ecole Centrale: <NNT : 2009ECLI0021>, 2009.

[34] "Digitalguide," 11 03 2019. [Online]. Available: https://www.ionos.fr/digitalguide/sites-internet/developpement-web/modele-en-cascade/. [Accessed 01 06 2019].

[35] S. Walther, *ASP.NET MVC Framework Unleashed,* Sams Publishing , 2009.

[36] G. B. Firesmith, *Henderson-Sellers,* Pearson Education, 2002.

[37] M. Hamilton, *Software Development: Building Reliable Systems,* Prentice Hall Professional, 1999.

[38] A. K.-H. Kingsley-Hughes, *Beginning Programming,* 2005.

[39] P. Rader, "spring-tool-suite," [Online]. Available: https://stackoverflow.com/tags/spring-tool-suite/info. [Accessed 16 06 2019].

[40] N. DRISSI, "Expert consultant bases de données," [Online]. Available: http://www.alphorm.com. [Accessed 12 02 2019].

[41] D. Liang, *Introduction to java programming, Eighth Edition, Armstrong Atlantic State University,* 2011, p. 73 pages.

[42] K. Chazotte, *Build Your Own Framework with Visual FoxPro,* Hentzenwerke, 2004.

[43] A. R. Seddighi, *pring Persistence with Hibernate,* BIRMINGHAM , SRM University: PACKT Publishing, 2016, p. 265 pages.

[44] S. M. Keith Mike, *Schincariol Merrick,* 2009.

[45] G. V. Liew, "Visual Studio Code," [Online]. Available: https://code.visualstudio.com/docs. [Accessed 27 06 2019].

[46] Infosys, *HTML, CSS, JavaScript,* 2005.

[47] J. Duckett, *Design and Build,,* vol. 336, 2012, p. 1078 pages.

[48] G. V. Liew, "Visual Studio Code," [Online]. Available: https://code.visualstudio.com/docs. [Accessed 26 06 2019].

[49] Benoît Philibert, “Bootstrap 3 Le framework 100 % web design,” 2011. [Online]. Available: www.editions-eyrolles.com.

[50] D. ALLARD, “Fontawesome.[En ligne],” 2016. [Online]. Available: http://www.studiovita-mine.com/blog/blogcomment-utiliser-la-bibliotheque-dicones-font-awesome. [Accessed 09 06 2019].

[51] “Angular,” 2019. [Online]. Available: (https://angular.io/docs). [Accessed 01 06 2019].

[52] UNOMENA, “What Restlet Client does ?, [En ligne],” 2018. [Online]. Available: https://restlet.com/documentation/. [Accessed 19 06 2019].

[53] B. Yakoub, *cloud privé,* H2IT Ibn Roch en partenariat avec 3il Limoges, 2017, p. 86 pages.