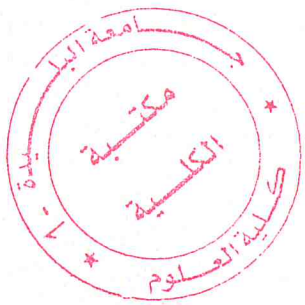
IIA-004-488_1

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 :

MISSOUM Imène KHELID Nihal

Thème : « Programmation distribuée par les Microservices pour la gestion de projet »

En vue d'obtenir le diplôme de master
Domaine : Mathématique et informatique
Filière : Informatique
Spécialité : Informatique
Option : Système Informatique et Réseau (SIR)

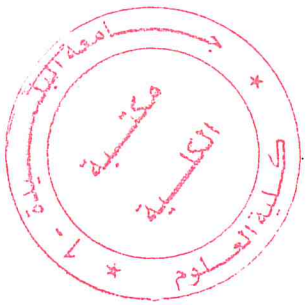
Structure d'accueil

Devant la commission des jurys :

Présidente: Mme.MANCER Yasmine Maître assistant à l'Université de Saad Dahleb Blida.
Examinatrice: Mme.GUESSOUM Dalila Maître assistant à l'Université de Saad Dahleb Blida.
Promotrice: Mme.BOUMAHDI Fatima Maître de conférences à l'Université de Saad Dahleb.
Encadreur : Mr. NEBHI Mourad Chef de département information et Technologies Sonatrach Hydra.

Promotion
2017 / 2018

MA-004-488-1

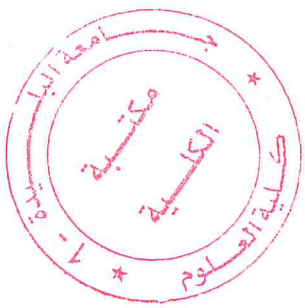

جامعة البليدة
مكتبة
الكلية
كلية العلوم

Résumé

Les entreprises subissent une pression constante pour s'adapter et améliorer leurs processus afin de rester compétitives. La migration vers les systèmes distribues est la solution adaptée par la plupart des sociétés. A cet égard, et afin de pallier à des problèmes récurrents dans le processus de la gestion de projet, les responsables de la direction PED (Petroleum Engineering and Development) de l'entreprise nationale SONATRACH ont émis le souhait de disposer d'une solution interne distribué basé Microservices pour la gestion de projet dépond des attentes de la division concerner afin de garder trace des projets en temps réels. La gestion de projet est l'activité stratégique de l'entreprise, c'est l'utilisation d'un savoir, d'habilités, d'outils et de techniques dans le cadre des activités d'un projet, en vue de satisfaire ou de dépasser les exigences et les attentes des parties prenantes à l'égard d'un projet. Dans le modèle Microservices, les applications sont divisées en une série de services, qui sont interconnectés fondés sur les capacités opérationnelles. Chaque service prend en charge une série de fonctionnalités distinctes, agissant en fait comme une mini application dotée de sa propre logique opérationnelle. Dans ce mémoire, nous montrons qu'au lieu d'utiliser des logiciels génériques tels que MS Project, la gestion de projet par les Microservices est une bonne solution dédiée à la direction PED. Notre implémentation est plus performante que les autres propositions existantes, sans sacrifier aucune fonctionnalité, répond au besoin et gère les problèmes de la division concernée.

Mots-clés

Microservice, gestion des projets, PED, Systèmes distribué, MS Project.



Abstract

Companies are under constant pressure to adapt and improve their processes in order to remain competitive. Migration to distributed systems is the right solution for most companies. In this regard, and in order to overcome recurring problems in the process of project management, the managers of the PED (Petroleum Engineering and Development) management of the national company SONATRACH have expressed the wish to have a distributed internal solution by Microservices of project management responds to expectations of the division concerned to keep track of projects in real time. Project management is the strategic activity of the company, it is the use of knowledge, skills, tools and techniques within the framework of the activities of a project, in order to satisfy or exceed the requirements and expectations of stakeholders for a project. In the Microservices model, applications are divided into a series of services and they are interconnected based on operational capabilities. Each service supports a series of distinct features, efectively acting as a mini-application with its own business logic. In this thesis, we show that instead of using generic software such as MS Project, project management by Microservices is a good solution dedicated to the PED direction. Our implementation is more ecient than the other existing proposals, without sacricing any functionality, responding to the need and managing the problems of the division concerned.

Keywords

Microservice, Project Management, PED, Distributed System, MS Project.

ملخص

الشركات تحت ضغط مستمر للتكيف وتحسين عملياتها من أجل البقاء قادرة على المنافسة. الهجرة إلى النظم الموزعة هو الحل الأمثل لمعظم الشركات. في هذا الصدد، ومن أجل حل المشاكل التي يواجها فرع:

(Petroleum Engineering and Development) PED بالشركة الوطنية سوناطراك في عملية إدارة المشاريع أعرب رغبة في أن يكون هناك حل داخلي يستند على الميكرو سرفيس إدارة توزيع ويستجيب المشروع لتوقعات الفرع المعني بتتبع المشاريع في الوقت الحقيقي.

إدارة المشاريع هي الأعمال الاستراتيجية للشركة وهو استخدام المعارف والمهارات والأدوات والأساليب المستخدمة في أنشطة المشروع، وذلك لتلبية أو لتتجاوز متطلبات وتوقعات أصحاب المصلحة في المشروع.

في نموذج الميكرو سرفيس، تنقسم التطبيقات إلى سلسلة من الخدمات. الميكرو سرفيس مترابطة على أساس القدرة التشغيلية. كل خدمة تدعم سلسلة من السمات المميزة، والعمل بفعالية باعتبارها مصغرة التطبيق مع منطق الأعمال الخاصة بها. في هذا المقال، تبين لنا أنه بدلا من استخدام برامج عامة مثل أمس بروجكت، إدارة الأعمال من قبل الميكرو سرفيس هو الحل جيد مخصص لفرع **PED**.

النتيجة التي توصلنا لها هي أكثر كفاءة من المقترحات الأخرى الموجودة، من دون التضحية بأي وظيفة، ويلبي حاجة وتدير مشاكل الفرع.

الكلمات المفتاحية

الميكرو سرفيس، **PED**، إدارة المشاريع، النظام الموزع، أمس بروجكت.

Remerciement

Nous exprimons nos vifs remerciements à toute personne ayant contribué, de près ou de loin, à la réalisation de ce travail.

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 Mme **GUERRACHE Faiza** pour son précieux aide et nous tenons à remercier aussi Monsieur **FAKHAR** et 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 exprimer notre profonde gratitude à nos familles qui nous ont toujours soutenues et à tout ce qui a participé à réaliser ce mémoire. Ainsi que l'ensemble des enseignants qui ont contribué à notre formation.

Dédicaces

Merci mon Dieu le tout puissant de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout de mon rêve.

A la mémoire de ma tante paternelle Salèha, qui nous a quitté au début du mois sacré de Ramadhan 2018.

A mes Très Chers Parents
*Chère **Maman,** cher **Papa**, vous représentez pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement, vous n'avez pas cessé de m'encourager et de prier pour moi.*
Vos prières et bénédictions m'ont été d'un grand secours pour mener à bien mes études.
Aucune dédicace ne saurait être assez éloquente pour exprimer ce que vous méritez, pour tous les sacrifices que vous n'avez cessé de me donner depuis ma naissance, durant mon enfance et même à l'âge adulte.
Que ce travail, que je vous dédie, en témoignage de mon profond amour, reflète ma profonde affection et ma grande reconnaissance.
Puisse Dieu, le tout Puissant, vous préserver et vous accorder santé, longue vie et bonheur pour que vous demeuriez le flambeau illuminant mon chemin.

*A ma chère tante **Hassina** que j'aime beaucoup.*

A ma deuxième mère

*ma tante **BELLAL Hakima**.*

A mes Très Chères Sœurs
Yasmine et Cerine,
Vous êtes merveilleuses, des étoiles brillantes, vous êtes la joie et le bonheur. Que Dieu vous garde et vous réalise tous vos rêves et souhaits.
*A mon Très Cher petit Frère **Anis**.*
A Tous mes Professeurs,
Saisis cette occasion pour vous exprimer ma profonde gratitude tout en vous témoignant mes respects et ma considération.

*A ma très chère **KHELID Nihal**,*
Je suis très contente de travailler avec toi durant cette année.

A Toutes mes Amies et Collègues de promotion et d'études.
A mes grandes mères, mes tantes et mes oncles, et a tous ceux ou celles qui me sont chers et que j'ai omis involontairement de citer, je vous dédie ce modeste travail.

MISSOUM Imène

Au nom d'Allah, le Miséricordieux, par essence et par excellence

Merci avant tout à Dieu.

Dédicaces

A l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, que dieu te garde dans son vaste paradis, à toi ***mon père****.*

A la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur; ***maman*** *que j'adore.*

A celui qui m'a soutenue tout au long de ce projet: mon mari ***Mohamed****, et bien sûr*
A mes sœurs: ***Inès****,* ***Nada*** *et* ***Tasnime****, sans oublié* ***mes grands-parents*** *et* ***ma belle-mère*** *que j'aime.*

A toute ma famille, et mes amis,

A mon binôme ***Imène*** *et toute la famille* ***Missoum****.*

Et à tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous dis merci.

Khelid Nihal

Table des matières

Liste des Figures

Liste desTables

Liste des abréviations

ACM Queue	Association for Computing Machinery Queue
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BPMN	Business Process Modelling Notation
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
ESB	Enterprise Service Bus
GNL	Gaz Naturel Liquéfié
GPL	Gaz de Pétrole Liquéfié
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
ISO	Organisation internationale de normalisation
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
Java SE	Java Standard Edition
JPA	Java Persistence API
JSON	JavaScript Object Notation
JSP	Java Server Pages
JVM	Java Virtuel Machine
MS Project	Microsoft Project
MVC	Model View Controller
PED	Petroleum Engineering and Development
PERT	Program and Evaluation Review Technic
PHP	Hypertext Preprocessor
PM	Project Manager
PMI	Project Management Institut
PNB	Produit National Brut
REST	Representational State Transfer
RMI	Remote Method Invocation
SGML	Standard Generalized Markup Language
SOA	Architecture Orientée Services
SOAML	Service-oriented architecture Modeling Language
SOAP	Simple Object Access Protocol
SPR	Suivi des projets et Reporting
SQL	Structured Query Language
TEP	Tonne d'Equivalent Pétrole
UML	Unified Modeling Langage
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

Introduction générale

Introduction générale

Une entreprise qui mène des projets est une entreprise en évolution. Du plus simple au plus complexe, les projets peuvent être gérés avec des méthodes différentes selon leurs natures et leurs exigences. La mise en place d'un projet est un des enjeux fondamentaux pour les entreprises et les organismes soucieux d'optimiser l'utilisation de leurs ressources humaines et matérielles.

La science et la technologie sont essentielles à la réalisation des objectifs de développement des organismes. Dans le processus de développement, l'information joue un rôle fondamental dans le transfert des connaissances scientifiques et techniques essentiels à la réalisation d'une amélioration.

Dans ce contexte, La division Petroleum Engineering and Development (PED),

souhaite bénéficier de l'évolution connue dans le monde informatique pour une amélioration dans tous les domaines. Ses projets d'études sont en collaboration avec des structures émettrices qui

utilisent les nouvelles technologies existantes.

- **Problématique**

Un des objectifs principaux de la division a été la formation des ressources humaines, en particulier au sein de projets, afin d'assurer le renforcement permanent des compétences locales pour assurer I 'exécution des activités après la fin du projet. Toutefois, la formation relative à la gestion des projets a reçu jusqu'ici peu d'attention parce que, ils travaillent encore avec des logiciels standards tels que MS Project. L'information à un instant donné est centralisée, absence de l'accès à distance des utilisateurs d'où la problématique d'avoir une application distribuée dédié à la division concerné.

- **L'objectif**

Soucieuse d'assurer une gestion des projets saine, et de veiller à ce qu'ils soient effectivement menés à terme, la division espère que le présent projet de fin d'études saura favoriser la réalisation d'une solution distribuée basée Microservices, dédiée au leurs besoins.

Dans cette optique, s'inscrit notre mémoire qui consiste à créer une application de gestion des projets d'une architecture distribuée, basée Microservices.

- **La structure du mémoire**

Le présent rapport est structuré en quatre chapitres :

- Le premier chapitre présente les concepts théoriques de gestion des projets, utiles pour l'élaboration de notre projet.
- Le deuxième chapitre est consacré à des rappels, et des définitions des architectures logiciels anciennes et la présentation des Microservices.
- Le troisième chapitre représente le choix conceptuel.
- Quant au dernier chapitre, il est dédié au développement des Microservices utilitaires de notre application, les différents services et le reste de l'application, il présente ainsi l'architecture générale du système en Microservices, les choix techniques et enfin 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
Gestion de projet

La conduite d'un projet en maîtrise d'œuvre requiert une continuité dans le processus total que ce soit dans la démarche appliquée ou dans l'équipe de maîtrise d'œuvre. Aussi le projet ne peut supporter de rupture de responsabilités d'une phase à l'autre sans nécessiter une remise à plat des moyens, des délais et des coûts. (Durand, 2004)

1.4 Le processus d'un projet

Les processus de gestion de projet sont présentés comme des composants distincts ayant des interfaces clairement définies. Les particularités d'un projet sont définies sous forme d'objectifs à réaliser en fonction de la complexité, du risque, de la taille, des délais, de l'expérience de l'équipe de projet, de l'accès aux ressources, de la quantité d'informations historiques, de la maturité de la gestion de projet dans l'organisation, du secteur d'activité et du champ d'application. Les groupes de processus nécessaires et les processus qui les constituent sont des guides qui aident à appliquer correctement au cours du projet la connaissance et les compétences en management de projet. De plus, cette application des processus de management de projet étant itérative, beaucoup de processus sont répétés et revus pendant le projet. Le chef de projet et son équipe ont la responsabilité de déterminer quels processus, dans les groupes de processus, seront employés, par qui, et avec quelle rigueur ils seront exécutés pour réaliser l'objet voulu du projet.

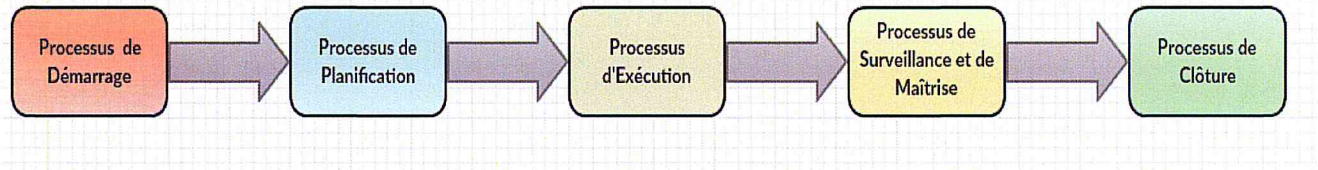


Figure 1-2 Les groupes de processus gestion de projet

Cette section identifie et décrit les cinq groupes de processus de management de projet nécessaires à tout projet. Ces cinq groupes présentent des dépendances internes nettes et doivent être exécutés selon la même séquence pour chaque projet.

Les groupes de processus et les processus qui les composent sont souvent réitérés avant l'achèvement du projet. Les processus d'un groupe peuvent avoir des interactions tant au sein du groupe qu'avec les autres groupes.

Les groupes de processus ne sont pas des phases du projet, dans le cas de projets de grande envergure ou complexes, tous les processus des différents groupes de processus peuvent se répéter normalement pour chaque phase ou chaque sous-projet. (PMI, 2004a)

Les cinq groupes de processus sont :

1.4.1 Processus de démarrage

Le processus qui définit et autorise le projet ou une phase du projet, est constitué des processus qui facilitent l'autorisation formelle de démarrer un nouveau projet ou une nouvelle phase du projet. (PMI, 2004b) Les processus de démarrage sont souvent exécutés, hors du périmètre de contrôle du projet, par l'organisation ou par des processus relatifs aux programmes. (PMI, 2004b)

1.4.2 Processus de planification

Le processus qui définit et affine les objectifs, et planifie le déroulement des actions requises pour atteindre ces objectifs ainsi que le contenu pour lequel le projet a été entrepris, les processus qui le constituent et leurs interactions, pour planifier et conduire avec succès un projet de l'organisation. Le groupe de processus de planification aide à collecter des informations à partir de nombreuses sources à différents niveaux d'exhaustivité et de confiance. Les processus de planification permettent l'élaboration du plan de gestion du projet. (PMI, 2004b)

Il y a deux méthodes de planification :

- **Diagramme de Gantt** : créée vers 1918 (El Montassir, 2007). On peut en utiliser la technique sans pour autant présenter le diagramme. Elle consiste à déterminer la meilleure manière possible de positionner les différentes tâches d'un projet à exécuter sur une période déterminée en fonction :
  - des durées de chacune des tâches,
  - des contraintes d'antériorité entre les différentes tâches,
  - des délais à respecter,
  - des capacités de traitement (qui peuvent évoluer en fonction des heures supplémentaires accordées, des investissements réalisés). (El Montassir, 2007)

Exemple :

Nous avons choisi un exemple excessivement simple pour expliquer la manière dont un Gantt se construit. Supposons qu'on cherche à ordonnancer la réalisation des tâches d'un projet ayant les caractéristiques suivantes :

Tâches à réaliser :

- Tâche A : durée 3 jours
- Tâche B : durée 6 jours

- Tâche C : durée 4 jours
- Tâche D : durée 7 jours
- Tâche E : durée 5 jours

Liens entre les opérations :

- B et D après A ;
- C après B ;
- E après D

Le diagramme de Gantt se présente sous la forme d'un tableau quadrillé où chaque colonne correspond à une unité de temps et chaque ligne à une opération à réaliser.

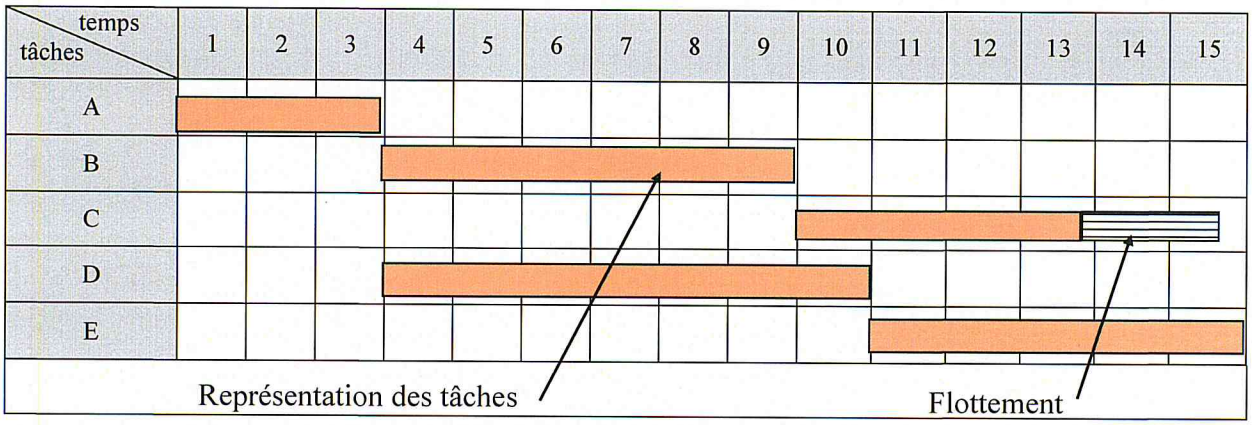


Tableau 1-1 Exemple de Gantt

On définit une barre horizontale pour chaque tâche ; la longueur de celle-ci correspond à la durée de la tâche. La situation de la barre sur le graphique est fonction des liens entre les différentes tâches.

- **Méthode Pert** : PERT signifie Program and Evaluation Review Technic (Technique d'élaboration et de contrôle des projets). Il date de 1958(El Montassir, 2007). L'outil PERT permet non seulement de visualiser un réseau d'antériorités, mais, dans le cas d'un projet, d'en déterminer dates et marges, d'en assurer contrôle et suivi.

Contrairement à celle du GANTT, la méthode PERT s'attache surtout à mettre en évidence les liaisons qui existent entre les différentes tâches d'un projet et à définir le chemin dit « critique ». Comme pour le GANTT, sa réalisation nécessite tout d'abord de définir :

- Le projet à réaliser
- Les différentes opérations et les responsables

- Les durées correspondantes
- Les liens entre ces différentes opérations

Le graphe PERT est composé d'étapes et d'opérations. On représente les étapes par des cercles, les opérations par des flèches. La longueur des flèches n'a pas de signification, il n'y a pas de proportionnalité de temps. Seront développées par la suite, les typologies de l'outil PERT. (El Montassir, 2007)

Exemple :

Nous avons choisi un exemple simple pour mieux visualiser comment ce diagramme marche :

Tâches à réaliser :

Tâche	Dépendances	Durée (en jour)
A	Aucune	3
B	A	2
C	B	7
D	C	5
E	C	3
F	D, E	5
G	F	5

Tableau 1-2 Exemple de diagramme PERT : l'ensemble des tâches à réaliser .

Les étapes à faire :

- Poser la date de démarrage de projet.

Figure 1-3 Exemple PERT : étape 1

- Placer les différentes tâches tout en renseignant dans un premier temps les dates au plus tôt de celles-ci :

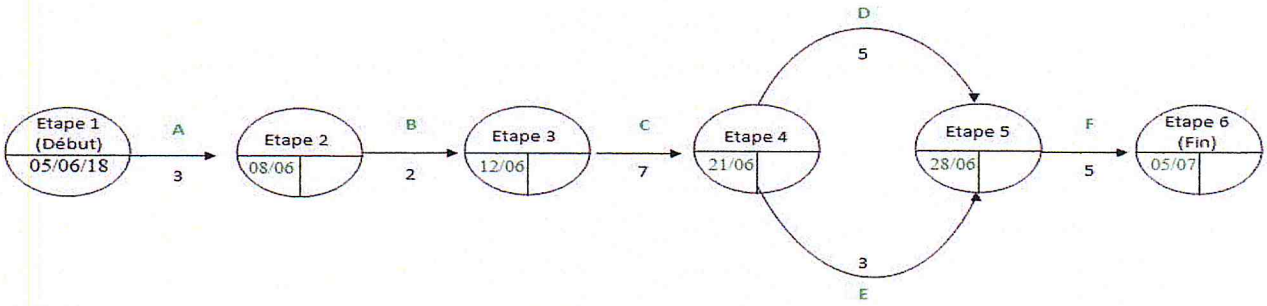


Figure 1-4 Exemple PERT : étape 2

- Une fois l'exactitude des données contrôlée, on s'attelle aux dates au plus tard. Pour cela, il vous faudra regarder si, certaines tâches peuvent se décaler dans le temps sans pour autant mettre en péril le délai global du projet. Dans notre cas, les dates au plus tard ne bougeront pas :

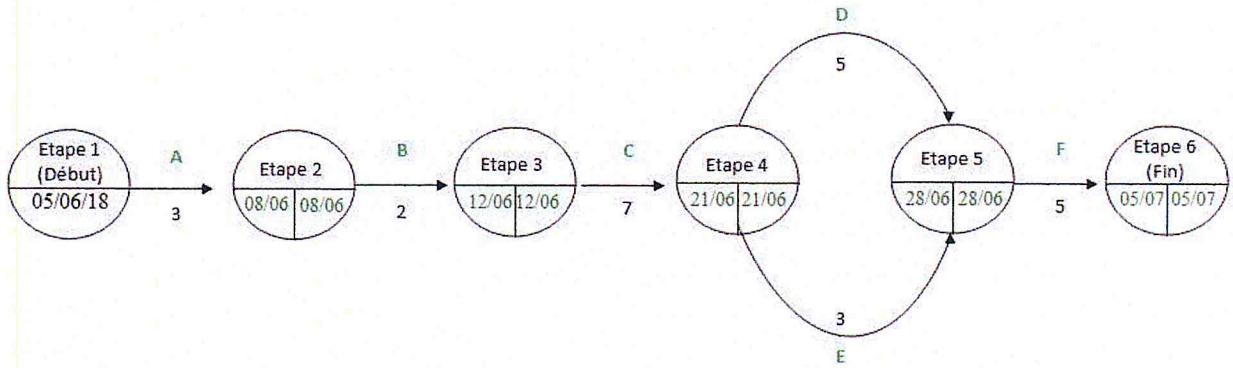


Figure 1-5 Exemple PERT : Dernière étape

- Une fois tous ces éléments renseignés, nous avons le diagramme de PERT prêt.

1.4.3 Processus d'exécution

Le processus qui intègre les personnes et autres ressources pour exécuter le plan de gestion du projet, constitué par les processus utilisés pour exécuter le travail défini dans le plan de gestion du projet pour satisfaire aux exigences du projet (PMI, 2004b). L'équipe de projet devrait déterminer quels sont les processus requis pour le projet concerné. Ce groupe de processus implique la coordination des personnes et des ressources ainsi que l'intégration et l'exécution des activités du projet conformément au plan de gestion du projet. (PMI, 2004b)

1.4.4 Processus de surveillance et de maîtrise

Le processus qui mesure et surveille régulièrement la progression du projet pour identifier les écarts par rapport au plan de management du projet, de manière à permettre les actions correctives nécessaires et à atteindre les objectifs du projet, constitué des processus utilisés pour observer l'exécution du projet, afin de pouvoir identifier les problèmes potentiels en temps voulu et entreprendre au besoin des actions correctives pour maîtriser l'exécution du projet. L'équipe de projet devrait déterminer quels sont les processus requis pour le projet concerné. (PMI, 2004b)

1.4.5 Processus de clôture

Le processus qui formalise l'acceptation du produit, du service ou du résultat, et qui conclut le projet ou une de ses phases de manière ordonnée, englobe les processus utilisés pour mettre formellement fin à toutes les activités d'un projet ou d'une phase de projet, remettre le produit achevé à d'autres ou clore un projet annulé. Une fois achevé, ce groupe de processus permet de vérifier que les processus définis sont également achevés pour tous les groupes de processus afin de clore le projet ou une phase du projet, selon le cas, et il confirme formellement que le projet ou la phase du projet est terminé.(PMI, 2004b)

1.5 Les indicateurs à suivre

La stratégie de suivi et de pilotage est définie à chaque projet, en fonction de sa taille ou de sa criticité, du nombre d'acteurs ou encore du contexte contractuel : processus de suivi, indicateurs retenus, rôles, fréquence de collecte et de diffusion, mode de communication. La démarche adoptée, qui se caractérise par une approche soit prédictive, soit agile, et par une formalisation plus ou moins outillée du planning, va impacter les modalités de suivi du projet. Les indicateurs de contrôle et les outils de pilotage varient selon la stratégie retenue pour la planification et la conduite du projet dans son ensemble. (Rota, 2008)

Les progiciels ou logiciels les plus utilisés sont : PSN7, MS Project, Sure Track, PRIMAVERA, Super Project, Artémis,…

1.5.1 La performance

La performance d'un projet à un instant t est la part de l'ensemble des travaux qui a été réellement réalisée (Rota, 2008). La difficulté réside dans la valorisation de cette performance; on peut dénombrer un nombre d'activités réalisées, un nombre de fonctionnalités ou d'exigences développées et validées, un nombre de points de fonction ou de cas d'utilisation implémentés.

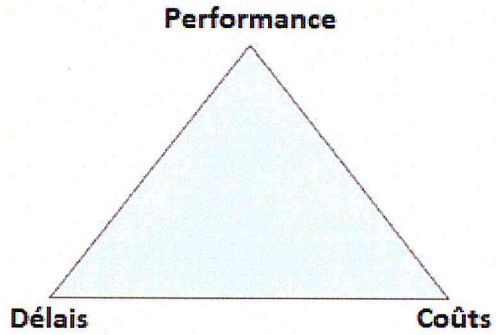


Figure 1-6 Le triangle de la triple contrainte (Roger,2011)

1.6.1 Le respect de la performance

La maitrise de la performance est la plus sûre garantie de réussite d'un projet, car les deux autres paramètres, coûts et délais en dépendent directement (Roger, 2011). Négliger le travail préparatoire relatif au paramètre « performance » entrainera une défaillance d'ordre technique ou organisationnelle qui donnera lieu irrémédiablement à des retards et des surcoûts. (Roger, 2011)

1.6.2 Le respect des coûts

Le paramètre « coûts » représente l'objectif économique du projet, qu'il s'agisse des recettes ou des dépenses. Ce paramètre essentiel caractérise la réussite ou l'échec économique d'un projet.

Ce paramètre sera respecté si :(Roger, 2011)

- L'on estime avec précision le détail, poste par poste, des coûts du projet ; la bonne estimation repose sur une connaissance précise du développement projet, des achats à réaliser et des tâches à exécuter ;
- L'on maitrise les dépassements de coûts internes et externes qui peuvent survenir sur la durée du projet en les analysant et en les négociant (contrôle des coûts) ;
- L'on négocie financièrement toutes les nouvelles demandes, exprimées par le client, qui interviennent en écart par rapport au contrat.

1.6.3 Le respect des délais

Le paramètre « délais » représente le respect de la date de livraison du projet. Essentiel, il caractérise la réussite ou l'échec calendaire d'un projet.

Ce paramètre sera respecté si : (Roger, 2011)

- L'on estime avec précision les délais d'approvisionnement et les durées de l'ensemble des tâches du projet ; la bonne estimation repose sur une connaissance précise du plan de développement projet ;
- L'on maitrise le dépassement de durée (internes et externes) qui peuvent survenir sur la durée du projet, en les analysant et en les renégociant (contrôle des délais) ;
- L'on répercute, dans le calendrier contractuel, toutes les nouvelles demandes exprimées par le client qui interviennent en écart par rapport au contrat. (Roger, 2011)

1.7 Le tableau de bord

Le vocable « tableau de bord » est apparu vers 1790. Le tableau de bord est défini par le lexique de gestion Dalloz comme une « *représentation synthétique chiffrée des principales informations nécessaires aux dirigeant (entrepreneurs, gouvernements) pour le contrôle de l'exécution d'un programme d'action et d'orientation en cas d'écarts par rapport aux projet ou projections* ». (Roger, 2011)

Cette définition, en introduisant les notions de chiffre, d'écart par rapport à une référence et de tendance, caractérise ce document comme un outil de management permettant : (Roger, 2011)

- De piloter des activités et des projets ;
- De contrôler des dépenses ;
- De respecter des budgets ;
- D'analyser des tendances ;
- De mesurer des écarts ;
- D'exploiter des résultats ;
- D'évaluer les risques ;
- De consulter un bilan d'activités ;
- De prendre connaissance, à travers de nombreux états, du fonctionnement d'une direction ;
- De lancer diverses actions (réunions, audit…).

Cet ensemble d'information, cohérentes et précises, de nature différente, couvrant l'ensemble du périmètre d'activités et regroupées dans un même document, permet ainsi de prendre les décisions les plus adaptées au contexte de l'entreprise.

Le tableau de bord doit retenir l'information juste et utile, il rendra compte de la vie du projet et de ses contraintes internes et externes.(Roger, 2011)

1.8 Gestion de projet au niveau de la société SONATRACH

Sonatrach est une compagnie nationale algérienne d'envergure internationale c'est la clé de voûte 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. (Benzarara, 2017)

1.8.1 La division Petroleum Engineering and Development (PED)

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. (Benzarara, 2017)

Pour faire le suivi des projets la direction SPR (Direction Suivi des projets et Reporting) utilise l'outil de planification MS Project.

MS Project est le logiciel de planification le plus utilisé aujourd'hui. La maîtrise d'un outil de ce type est indispensable pour tout chef de projet et planificateur. Les fonctionnalités de l'outil et la littérature sont si abondantes qu'il est difficile de saisir l'essentiel en termes de bonnes pratiques afin de créer simplement un planning correct. (Moine, 2013)

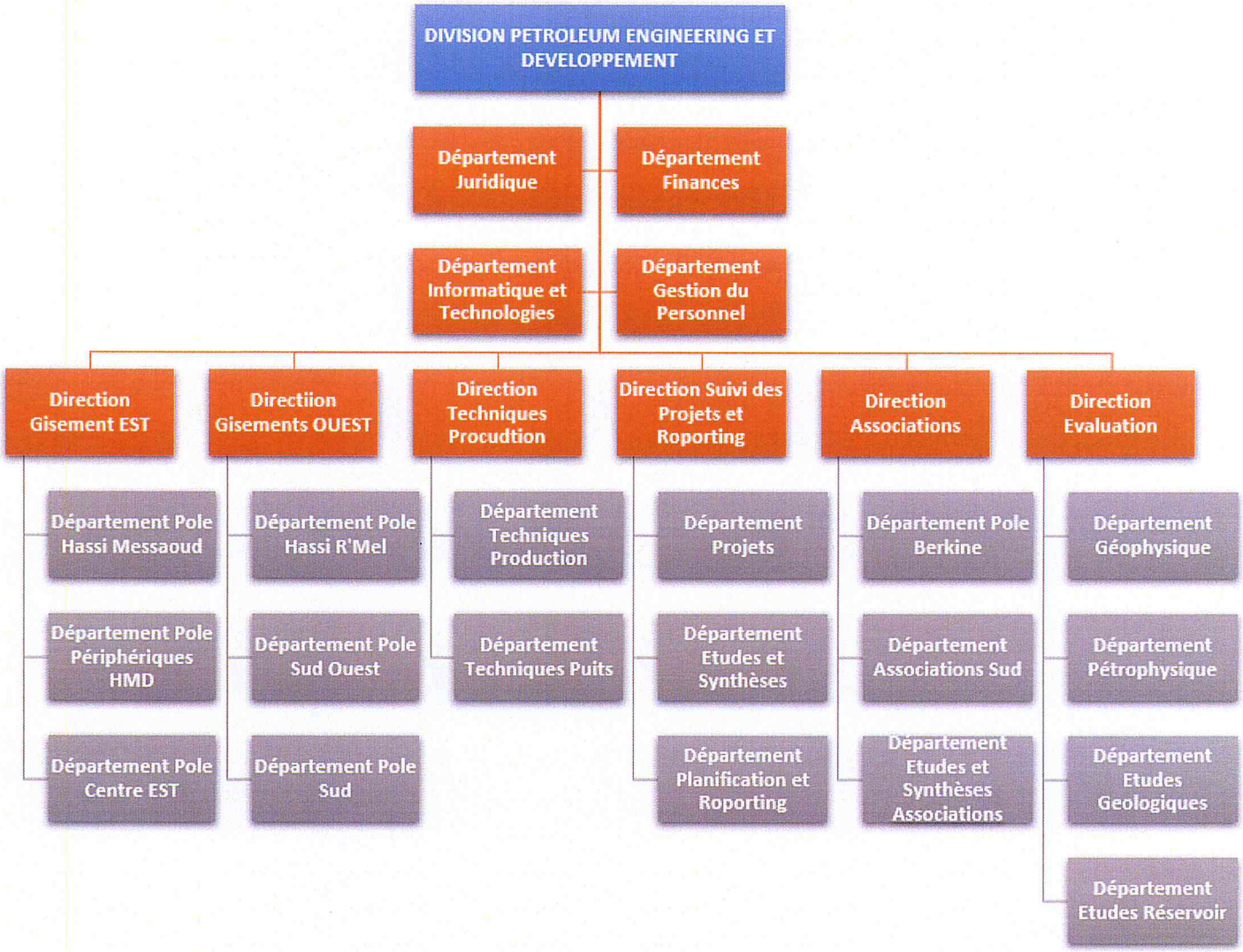


Figure 1-7 Représentation de la Division Petroleum Engineering and Development

1.9 Conclusion

Dans ce chapitre introductif, qui est réservé à présenter les concepts théoriques relatifs à notre projet, on a présenté les différentes généralités sur la gestion de projets. Dans le prochain chapitre, nous abordons les concepts de base reliés à une architecture Microservices.

Chapitre 2
Les Microservices

- Elles limitent l'évolutivité. La stratégie habituelle pour gérer les incréments de demandes entrantes est de créer de nouvelles instances de la même application et de diviser la charge parmi les instances. Cependant, il pourrait être le cas que l'augmentation le trafic souligne seulement un sous-ensemble des modules, ce qui rend l'allocation des nouvelles ressources pour les autres composants gênant.
- Elles représentent également un verrou technologique pour les développeurs, qui sont liés utiliser le même langage et les mêmes cadres de l'application originale.
- Le style architectural Microservices a été proposé pour faire face à de tels problèmes. Dans notre définition du Microservices, nous utilisons le terme «cohésif» pour indiquer qu'un service n'implémente que des fonctionnalités fortement liées au souci qu'il est destiné à modéliser.(Mazzara & Meyer, 2017)

Les applications monolithiques sont donc développés en un seul bloc, toutes les aspects fonctionnels d'un problème sont centralisés et implémentés au sein de la même application, contrairement au Microservices qui structure une application comme une collection de petits services autonomes, faiblement couplés , remplissant chacun une seule tâche fonctionnelle ou technique comme indique la figure suivante :

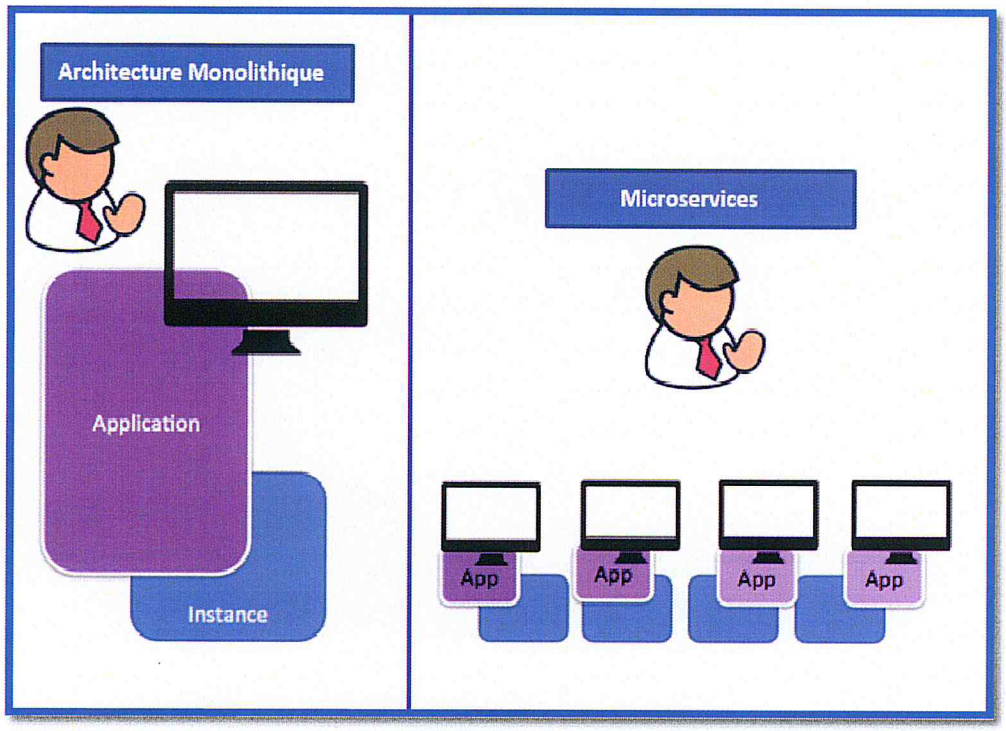


Figure 2-1 Différence entre architecture Monolithique et les Microservices

2.3 Les architectures orientées services

Afin de comprendre les micros services, il semble nécessaire de discuter de leur relation avec leur cousin plus âgé : un style architectural communément appelé architecture orientée services (SOA). Bien que les SOA aient fait fureur au début des années 2000, il y a beaucoup d'idées souvent contradictoires autour de ce qu'est vraiment une SOA. Martin Fowler soulève ce point dans un billet de blog 2005, et sa tentative est de classer les opinions dominantes peut se résumer ainsi : (Carneiro & Schmelmer, 2016)

- Une architecture SOA est conçue pour exposer les logiciels via des services Web, impliquant généralement backbone commun, généralement via HTTP, pour que les applications d'une organisation puissent travailler avec.
- Elle vise à faire disparaître les applications monolithiques et à les remplacer par un ensemble de services de base qui exposent les fonctionnalités et les données de l'entreprise. Données fournis par les services de base seront regroupées dans des interfaces utilisateur distinctes.
- Un SOA implique nécessairement une technologie de bus de message asynchrone, qui communique les événements professionnels via des "documents" (parfois appelés messages) entre les systèmes impliqués. Le SOA est définis comme un principe architectural dans lequel il y a un biais à construire des systèmes grâce à des services implémentés séparément, déployés, évolutifs et maintenus. Dans une configuration comme celle-ci, toutes les applications «front-end» clientes se basent uniquement sur un ensemble de (back-end) services qui exposent les fonctionnalités que les applications frontales consomment pour faire leur travail.(Carneiro & Schmelmer, 2016)

2.3.1 Service

La définition d'un service d'un logiciel est très proche de la définition du dictionnaire américain d'Oxford (non technique) de tout service, en ce sens qu'il s'agit d'un «système répondant à un besoin public». Les services logiciels doivent répondre aux besoins d'une ou de plusieurs applications client internes ou externes.(Carneiro & Schmelmer, 2016)

2.3.2 Avantage

- Obligation d'avoir une modélisation poussée.
- Possibilité de découpler les accès aux traitements.
- Localisation et interfaçage transparents (ouverture accrue).
- Possibilité de mise en place facilitée à partir d'une application objet existante.

- Réduction des coûts en phase de maintenance et d'évolution.
- Facilité d'amélioration des performances pour des applications importantes (répartition des traitements facilitée).

2.3.3 Inconvénient

- Coûts de conception et de développement initiaux plus conséquents.
- Nécessité d'appréhender de nouvelles technologies.
- Existant non SOA dans les entreprises.
- Performances réduites pour des traitements simples (couche supplémentaire).

Les architectures orientés services et les Microservices partagent le même fonctionnement basé sur les services, mais SOA est un terme très large par rapport aux Microservices qui est un style architectural inspiré de l'orientée services qui a récemment commencé à gagner en popularité. La principale différence entre eux réside dans la taille et la portée. Un Microservice doit être significativement plus petit que ce que SOA a tendance à être et est principalement un petit service déployé indépendamment. D'un autre côté, un SOA peut être plusieurs Microservices, donc le SOA est plus général comme indique la figure suivante :

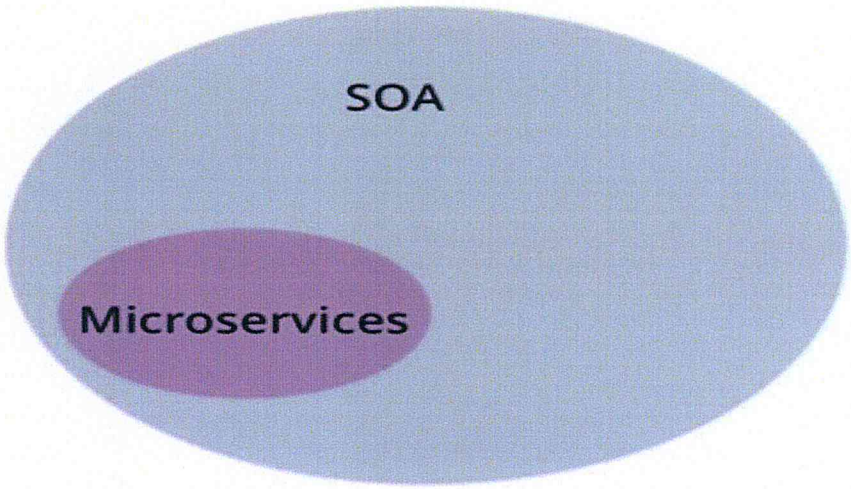


Figure 2-2 Comparaison entre Microservices et SOA

2.4 Microservices

Il est difficile de déterminer exactement qui a utilisé le terme micro service pour la première fois et quand. Alors qu'il semble plus probable que le terme soit apparu sur la scène autour de 2011, certaines entreprises ont adopté ce style beaucoup plus tôt. Amazon est un exemple, qui a brisé son application Obidos précédemment monolithique dans de nombreux petits services. Une autre définition celle de Microsoft Fellow Jim Gray pour ACM Queue, Amazon CTO Werner Vogels décrit comme une «orientation de service» générale ce qui est très proche de la définition des micros services ; telleque l'orientation des services consiste à encapsuler les données avec le logique métier qui fonctionne sur les données, avec le seul accès via une interface de service publiée. Pas de base de données directe l'accès est autorisé depuis l'extérieur du service, et il n'y a pas de partage de données entre les services.(Carneiro & Schmelmer, 2016)

L'ancien dirigeant d'eBay et de Google, Randy Shoup, résume les caractéristiques de micro service comme ayant trois caractéristiques principales : (Carneiro & Schmelmer, 2016)

- Ils ont une portée très ciblée qui se concentre sur un très petit nombre de fonctionnalités (Parfois unique), exprimé par une API petite et bien définie.
- Ils sont tous très modulaires, et indépendamment déployée, élevant essentiellement le Règles d'encapsulation logicielle et de modularité du programme au déploiement niveau de l'unité.
- Enfin, les micros services ont une persistance isolée, ce qui signifie qu'ils ne partagent pas stockage persistant, tel qu'une base de données commune.

La principale différence que nous avons trouvé est que les micro services ont le trait commun de décomposer une grande application dans un ensemble de petits services, concentrés, visionnés, et faiblement couplés, qui sont très peu coûteux tourner et maintenir.

Une autre définition des micros services est celle de Fowler et Lewis, comme un style architectural (Carneiro & Schmelmer, 2016) et une approche pour développer une application unique en tant que suite de petits services, chacun avec son propre processus et permet de communiquer avec des mécanismes légers, souvent une API de ressources HTTP. (Carneiro & Schmelmer, 2016)

Ces services sont construits autour de capacités métier et déployée indépendamment par des machines de déploiement entièrement automatisées. Il y a un nu minimum de gestion centralisée de ces services, qui peuvent être écrits dans des langages de programmation et utilisent différentes technologies de stockage de données. Cette approche est un changement important par rapport aux implémentations SOA à l'ancienne, qui ont fini par être beaucoup plus précisément décrit comme un

ensemble d'applications monolithiques de taille moyenne à grande, cousues ensemble via des solutions ESB (Enterprise Service Bus) coûteuses et souvent complexes. (Carneiro & Schmelmer, 2016)

Cela a souvent conduit à beaucoup plus systèmes étroitement couplés, où de nombreux contrats d'inter application n'étaient pas explicites, mais plutôt implicites et caché à l'intérieur de l'ESB "couche de colle". En revanche, les architectures de micro services ont tendance à fonctionner de manière très explicitement publiée, de petite taille et plutôt contrats simples entre les applications de service unique. Les implémentations de micro service les plus performantes appliquer la loi de Postel d'être "conservateur dans ce que vous faites, soyez libéral dans ce que vous acceptez des autres" à tous services qui composent l'ensemble du système.(Carneiro & Schmelmer, 2016)

2.4.1 Propriétés de l'architecture des Microservices

- Autonome : peut être déployé indépendamment. Toutes les communications via des appels réseau, expose une API.
- Hétérogénéité technologique : Différentes parties des systèmes peuvent utiliser une pile technologique différente si nécessaire pour de meilleures performances. Mais de multiples technologies s'accompagnent d'une compréhension opérationnelle de plusieurs piles de technologie.
- Résilience : Les micros services peuvent fournir de la résilience en introduisant de la redondance.
- Mise à l'échelle : c'est la scalabilité, désigne la capacité d'un système à s'adapter à un changement d'ordre de grandeur de la demande.
- Facilité de déploiement : les micros services individuels peuvent être déployés indépendamment du déploiement de Monolithique entier.
- Organisation : différents micro services peuvent appartenir à différentes équipes.
- Réutilisation : réutilisation des fonctionnalités, tel que le même micro service peut être utilisé à différentes fins.

2.4.2 Avantage de l'architecture des Microservices

- Chaque micro service est développé, testé et déployé séparément des autres.
- Chaque micro service est développé en utilisant une technologie qui peut être différente des autres (JAVA, C++, PHP, Python…).

- Chaque service tourne dans un processus séparé donc si un micro service plante il ne va pas affecter les autres.
- La combinaison des micros services peuvent réaliser des opérations très complexes.
- Ils sont faiblement couplés puisque chaque micro service est physiquement séparé des autres.
- Indépendances relative entre les différentes équipes qui développement les différents micros services.
- Facilité des tests et du déploiement.
- Livraison continue d'où le client ne va pas attendre longtemps pour avoir les premières versions et il va recevoir son produit rapidement.

2.4.3 Inconvénient de l'architecture des Microservices

- Découpage de fonctionnement de chaque service et le délimiter puisque il faut créer des services les plus simples possibles faisant un minimum de choses.
- la baisse de la performance causée par les appels de processus distants.

2.4.4 L'intégration des Microservices

Obtenir l'intégration correcte est l'aspect le plus important de la technologie associée Avec Microservices. Il existe une panoplie d'options sur la façon dont un Microservice peut communiquer avec un autre, et comment assurer la sécurité des transferts effectués et gardé trace a toutes échanges.

2.4.4.1 Les protocoles

Le client et les services peuvent communiquer à travers différents types de communication, chacun ciblant un scénario et des objectifs différents. Initialement, ces types de communications peuvent être classés en deux axes. Le premier axe définit si le protocole est synchrone ou asynchrone.

2.4.4.1.1 HTTP (Protocole de transfert hypertexte)

HTTP (HyperText Transfer Protocol) est un protocole pour les systèmes d'information distribués, collaboratifs et hypermédias. Il est aussi, l'ensemble de règles régissant le transfert de fichiers (texte, images, son, vidéo, et autres fichiers multimédias) sur le Web (Fielding et al, 1999). Dès qu'un utilisateur se connecte au Web et ouvre un navigateur, il utilise indirectement le protocole http. Il est un protocole synchrone, Le client envoie une demande et attend une réponse du service. Cela est indépendant de l'exécution du code client qui pourrait être synchrone (thread est bloqué) ou asynchrone (thread n'est pas bloqué, et la réponse atteindra éventuellement un rappel).

Le point important ici est que le protocole (HTTP / HTTPS) est synchrone et que le code client ne peut poursuivre sa tâche que lorsqu'il reçoit la réponse du serveur HTTP. (Fielding et al, 1999)

2.4.4.1.2 REST (Transfert d'état représentatif)

Le style d'architecture REST (Representational State Transfer) est un moyen pour développer des services Web. Les services Web RESTful gagnent de plus en plus d'approches. Ils sont utilisés comme application interfaces de programmation (API) pour les services Web 2.9. (Fernandes et al., 2013)

Reposant Les sujets techniques des services Web deviennent populaires Le style REST comprend identifiant global de toutes les ressources (par exemple, un identifiant de ressource uniforme) et le client doit seulement Connaissez cette poignée et l'action requise. Il doit aussi savoir le bon format de représentation, qui est généralement un HTML, langage XML (eXtensible Markup Language) ou objet JavaScript Notation (JSON) métadonnées.

Services Web RESTful (REST API) spécifient un ensemble de ressources, qui comprend trois composants : l'URI du service Web, le type de données pris en charge (JSON, XML, etc.) et les opérations de support via des méthodes HTTP. (Fernandes et al., 2013)

2.4.4.1.3 AMQP (Protocole avancé de mise en file d'attente des messages)

Le protocole AMQP (Advanced Message Queuing Protocol) est un middleware de message standard ouvert, utiliser des messages asynchrones (Fernandes et al., 2013). Selon la norme AMQP, produits middleware écrits pour différentes plates-formes et dans différentes langues peuvent envoyer des messages d'un à un autre. (Fernandes et al., 2013)

AMQP est soutenu par un bon nombre d'acteurs clés, y compris Cisco Systems, Credit Suisse, Deutsche Borse Systems, Goldman Sachs, Banque JPMorgan Chase, Red Hat, et 29West. (Fernandes et al., 2013)

AMQP permet aux applications d'envoyer et de recevoir des messages. À cet égard, il fonctionne comme la messagerie instantanée ou email. AMQP diffère énormément des autres solutions disponibles, car il permet la spécification de quels messages peuvent être reçus et à partir de, et comment les compromis sont effectués en ce qui concerne sécurité, fiabilité et performance.

Systèmes conçus pour s'intégrer AMQP effectuer beaucoup mieux au fonctionnement sans surveillance ou "Lights-out" que d'autres solutions. Il y a plusieurs raisons de choisir l'AMQP sur la

compétition, y compris la commodité, la possibilité de connecter des applications sur différentes plates-formes, la possibilité de connecter des partenaires en utilisant un standard ouvert complet, et une position pour innovation fondée sur les fondements de l'AMQP.(Fernandes et al., 2013)

2.4.4.1.4 SOAP (Protocole d'accès aux objets simple)

Le protocole SOAP (Simple Object Access Protocol) est un protocole de messagerie basé sur XML pour l'échange d'informations entre ordinateurs. Il permet à des programmes qui s'exécutent sur des systèmes d'exploitation distincts (tels que Windows et Linux) de communiquer via Internet. SOAP peut étendre HTTP pour la messagerie XML. (Baude, 2008)

Il fournit le transport de données pour les services Web, et peut échanger des documents complets ou appeler une procédure distante, ou diffuser un message. SOAP permet aux applications client de se connecter facilement aux services distants et d'appeler des méthodes distantes.

Bien que SOAP puisse être utilisé dans une variété de systèmes de messagerie et puisse être distribué via divers protocoles de transport, SOAP se concentre sur les appels de procédure distants transportés via HTTP (Baude, 2008). D'autres frameworks tels que CORBA, DCOM et Java RMI fournissent des fonctionnalités similaires à SOAP. , mais les messages SOAP sont entièrement écrits en XML et sont donc uniquement indépendants de la plate-forme et du langage. (Baude, 2008)

2.4.4.2 Spring cloud

Spring cloud fournit des outils permettant aux développeurs de construire rapidement certains des modèles courants dans les systèmes distribués (par exemple, gestion de configuration, découverte de service, disjoncteurs, routage intelligent, micro-proxy, bus de contrôle, sessions, état du cluster). (Bottard et al., 2018)

La coordination des systèmes distribués conduit à des schémas de plaque de chaudière, et l'utilisation des développeurs Spring Cloud peut rapidement mettre en valeur les services et les applications qui implémentent ces modèles. Ils fonctionneront bien dans n'importe quel environnement distribué, y compris le propre ordinateur portable du développeur. Spring cloud est donc la boite à outils des Microservices. (Bottard et al., 2018)

2.4.4.2.1 Les services proxy

Les proxys web sont une alternative intéressante aux browsers et aux serveurs. Ce sont des serveurs intermédiaires par lesquels transitent tous les messages des clients qui y sont connectés, les proxys sont libres d'appliquer des traitements qu'ils veulent à ces messages. Par exemple, un proxy

gérant un cache renvoie les documents qu'il possède dans son cache et fait suivre au serveur les requêtes sur des documents inconnus ou mis à jour.

Un proxy peut être partagé par plusieurs clients. Ainsi il est possible de factoriser les services destinés à des groupes d'utilisateurs dans un proxy commun. A l'opposé, un service spécifique à un utilisateur peut être réalisé par un proxy qui lui est dédié. Les proxys peuvent être chainés et les messages peuvent transiter par plusieurs proxys offrant différents services. Enfin, ils sont très bien supportés par les clients ; leur configuration est simple et peut même, pour certains d'entre eux être automatique. (Dedieu, 2006)

2.4.4.2.2 Les services d'enregistrement

Le service d'enregistrement aide à garder une trace des instances des services. Il est une base de données contenant les emplacements réseau du service instances (Gadge & Kotwani, 2017). Chaque instance de service s'enregistre au démarrage et désinscrit à l'arrêt. La passerelle API utilise cette information pendant la découverte du service.

De plus, c'est la responsabilité du service de mettre à jour le registre à propos de sa disponibilité /indisponibilité pour répondre à la demande. En tant que composant essentiel, le service d'enregistrement doit être hautement disponible. (Gadge & Kotwani, 2017)

2.4.4.2.3 Les services de configuration

Service de configuration Spring, fournit un support côté serveur et côté client pour la configuration externalisée dans un système distribué (Bottard et al., 2018). Avec le serveur de configuration, la disposition d'un emplacement central pour gérer les propriétés externes des applications dans tous les environnements.

Les concepts sur le client et le serveur sont mappés de la même manière que les abstractions Spring Environnent et les propriétés des sources, de sorte qu'ils s'adaptent très bien aux applications Spring, mais peuvent être utilisés avec n'importe quelle application s'exécutant dans n'importe quelle langue (Bottard et al., 2018). Lorsqu'une application traverse le pipeline de déploiement, de la phase de développement à la phase de test, on peut gérer la configuration entre ces environnements et on peut assurer que les applications disposent de tout ce dont elles ont besoin pour migrer. Il est facile d'ajouter des implémentations alternatives et de les brancher avec la configuration Spring. (Bottard et al., 2018)

2.4.4.2.4 Les services de sécurités

Service de sécurités Spring est un cadre d'authentification et de contrôle d'accès puissant et hautement personnalisable, utilisé pour la sécurisation des applications Spring et sa facilité d'extension pour répondre aux besoins spécifiques. (Rayen Baxter et al, 2018)

2.5 Conclusion

Ce chapitre a présenté les concepts fondamentaux liés à notre problématique. Nous nous sommes intéressés particulièrement à la définition des Microservices, ses avantages, inconvénient, ses propriétés et leur l'intégration par des protocoles et des services pour assurer les échanges sécurisés entre eux et garder leurs traces. Le chapitre suivant étudie la conception de notre solution proposée.

Chapitre 3

Analyse et Conception

3.1 Introduction

Le défi de répondre aux challenges liés à ce projet nous à mener à concevoir notre solution, en se basant sur une architecture fonctionnelle et technique qui répond dans chacune de ses parties sur un ou plusieurs de ces challenges. Etant donné la complexité technique de ce travail due à la nouveauté des technologies et approches utilisées, nous avons opté pour le modèle d'architecture logicielle à «4+1» vues pour minimiser la complexité du projet. L'architecture logicielle s'occupe d'abstraction, de décomposition et de composition de style et d'esthétique. Pour décrire une architecture logicielle, nous allons utiliser un modèle composé de plusieurs vues. Aussi et à l'effet de traiter les architectures importantes et d'une approche difficile, nous allons décortiquer le modèle choisi.

3.2 Le modèle 4+1 vues

Il est basé sur l'utilisation de plusieurs vues différentes, permettant de décrire l'architecture de systèmes complexes où le logiciel a une influence essentielle sur la réalisation et l'évolution du système (Kruchten , 2015). L'utilisation de ces vues multiples permet de traiter séparément les intérêts des différentes «parties prenantes» de l'architecture : utilisateurs finaux, développeurs, ingénieurs systèmes, chefs de projet, etc., et de traiter séparément les exigences fonctionnelles et non-fonctionnelles.

Chaque vue est d'écrite, accompagnée d'une notation permettant de la représenter. Les vues ont été conçues en utilisant un processus de développement itératif centré sur l'architecture et piloté par les cas d'utilisation. Le modèle proposé sera composé de cinq vues principales détaillées dans la figure suivante :

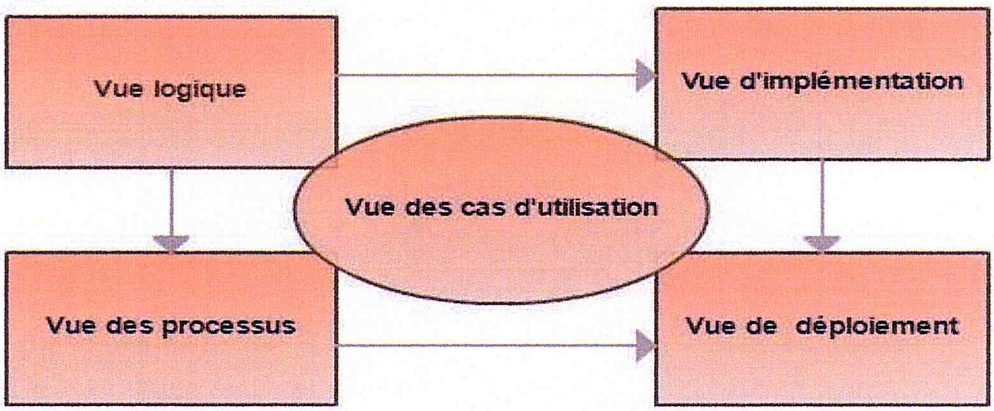


Figure 3-1 Le modèle d'architecture logicielle à «4+1» vues **(Kruchten , 2015)**

- **La vue logique**, est le modèle objet de la conception (quand une méthode de conception orientée objet est utilisée). Cette vue concerne « l'intégrité de conception».

Cette vue de haut niveau se concentre sur l'abstraction et l'encapsulation, elle modélise les éléments et mécanismes principaux du système.

Elle identifie les éléments du domaine, ainsi que les relations et interactions entre ces éléments « notions de classes et de relations » :

- Les éléments du domaine sont liés au(x) métier(s) de l'entreprise,
- ils sont indispensables à la mission du système,
- ils gagnent à être réutilisés (ils représentent un savoir-faire).
- Cette vue organise aussi (selon des critères purement logiques), les éléments du domaine en "Catégories" :
- Répartir les tâches dans les équipes.
- Regrouper ce qui peut être générique.
- Isoler ce qui est propre à une version donnée, etc...
- **La vue processus,** enregistre les aspects de concurrence et de synchronisation de la conception. Cette vue concerne « l'intégrité d'exécution ».

Cette vue est très importante dans les environnements multitâches ; elle exprime la perspective sur les activités concurrentes et parallèles. Elle montre ainsi :

- La décomposition du système en termes de processus (tâches).
- Les interactions entre les processus (leur communication).
- La synchronisation et la communication des activités parallèles (threads).
- **La vue de déploiement**, décrit le déploiement du logiciel sur le matériel et reflète son aspect distribué. Cette vue concerne « l'intégrité de performance ». Elle exprime la répartition du système à travers un réseau de calculateurs et de nœuds logiques de traitements. Cette vue est particulièrement utile pour décrire la distribution d'un système réparti. Elle montre :
- La disposition et nature physique des matériels, ainsi que leurs performances.
- L'implantation des modules principaux sur les nœuds du réseau.
- Les exigences en termes de performances (temps de réponse, tolérance aux fautes et pannes...).

des différentes phases qui composants les projets et même leurs tâches, et de même pour la création et la mise à jour (nous pouvant créer et mettre à jour des phases et même des tâches).

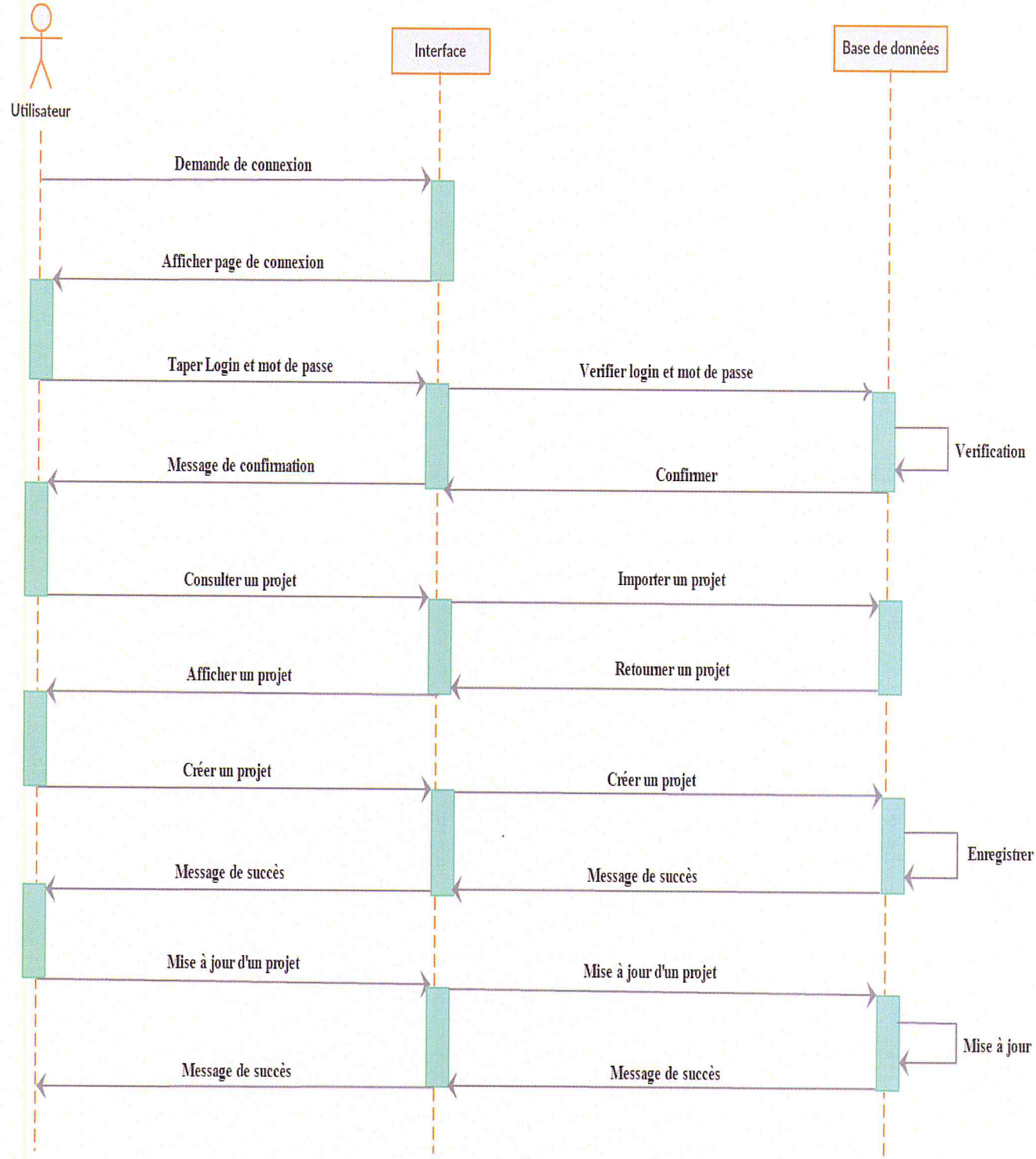


Figure 3-2 Diagramme de séquence général de notre système

3.2.1.2 Diagramme de classe

Le diagramme de classe présente les classes et les relations statiques entre eux.

La figure (3-3) décrit les classes qui composent notre système ainsi que les relations, les attributs et les méthodes de ces derniers. Une classe représente la structure d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui le composent. Elle est constituée d'attributs dont les valeurs représentent l'état de l'objet et des méthodes qui sont les opérations applicables aux objets.

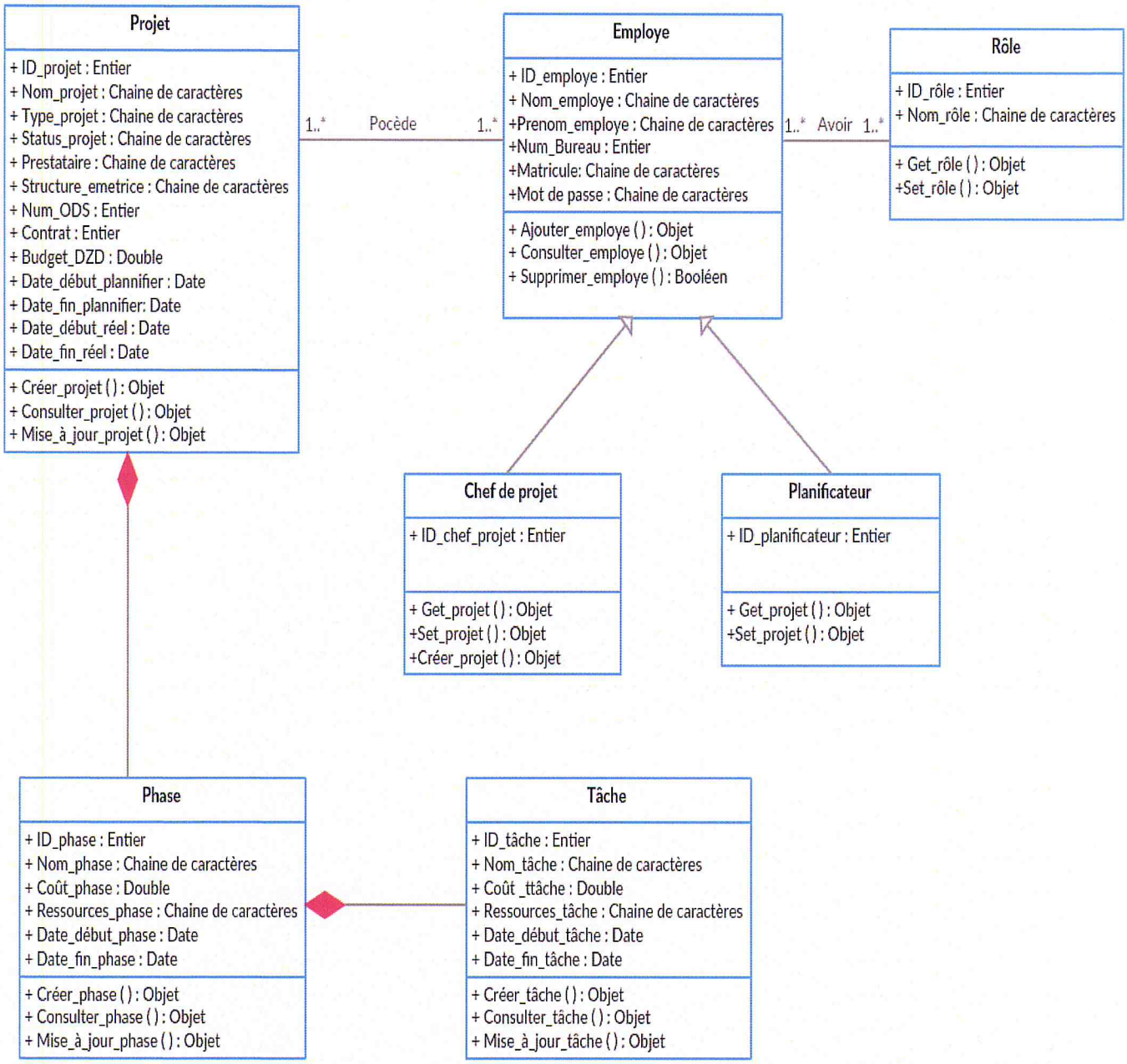


Figure 3-3 Diagramme de classe général de notre système

3.2.2 Vue de processus

Dans la vue de processus, nous utilisons le diagramme BPMN pour présenter les différents processus de notre système et montrer l'interaction entre eux.

3.2.2.1 Diagramme BPMN

BPMN (Business Process Modelling Notation) est un standard pour la modélisation de processus métier facilement compréhensible par les partenaires professionnels, prévu pour servir comme langage visant à combler un déficit de communication qui survient souvent entre le design des processus métier et l'implémentation. Il fournit une notation graphique permettant de définir des processus métier dans un diagramme de processus métier (BPM), basé sur une technique d'organigrammes très proche de celle utilisée par les diagrammes d'activité UML. (Tchokpon, 2017)

La figure (3-4) définit le processus général des étapes pour la création du projet de notre système.

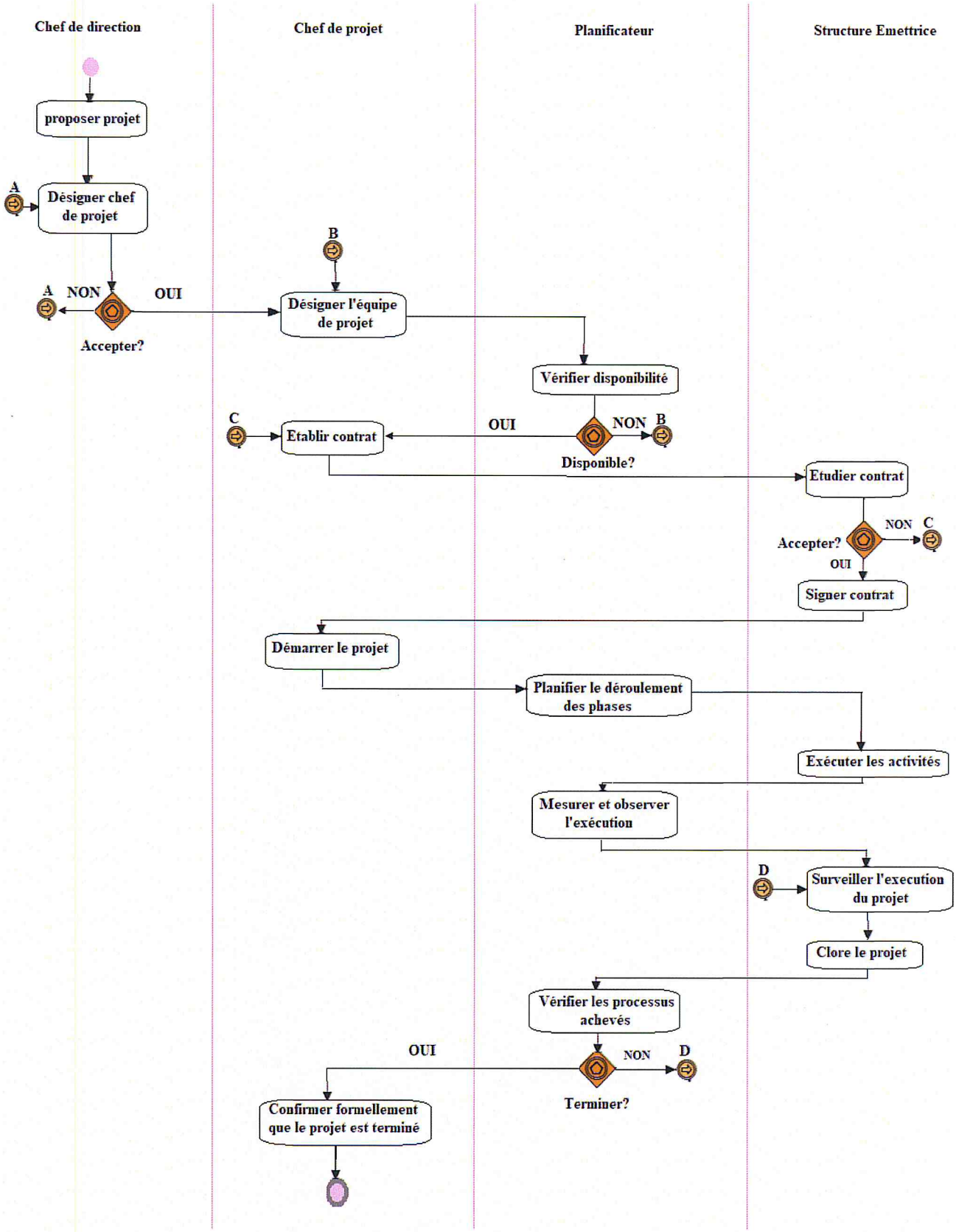


Figure 3-4 Diagramme BPMN de notre système

3.2.3 Vue de déploiement

La vue physique nous permet de voir dans quels environnements notre composant va être déployé, et ce, à travers le diagramme d'architecture de services et le diagramme de contrat de services (SOAML).

3.2.3.1 SOAML

Le profil SoaML permet de personnaliser les abstractions UML nécessaires à la modélisation des services. Il clarifie comment utiliser les métas classes UML pour modéliser une solution SOA afin de garantir que les modèles des services reflètent les cinq principes SOA suivants : couplage faible, abstraction, réutilisation, autonomie et composition.

La figure (3-5) décrit les différents acteurs participant pour la gestion des projets dans notre système.

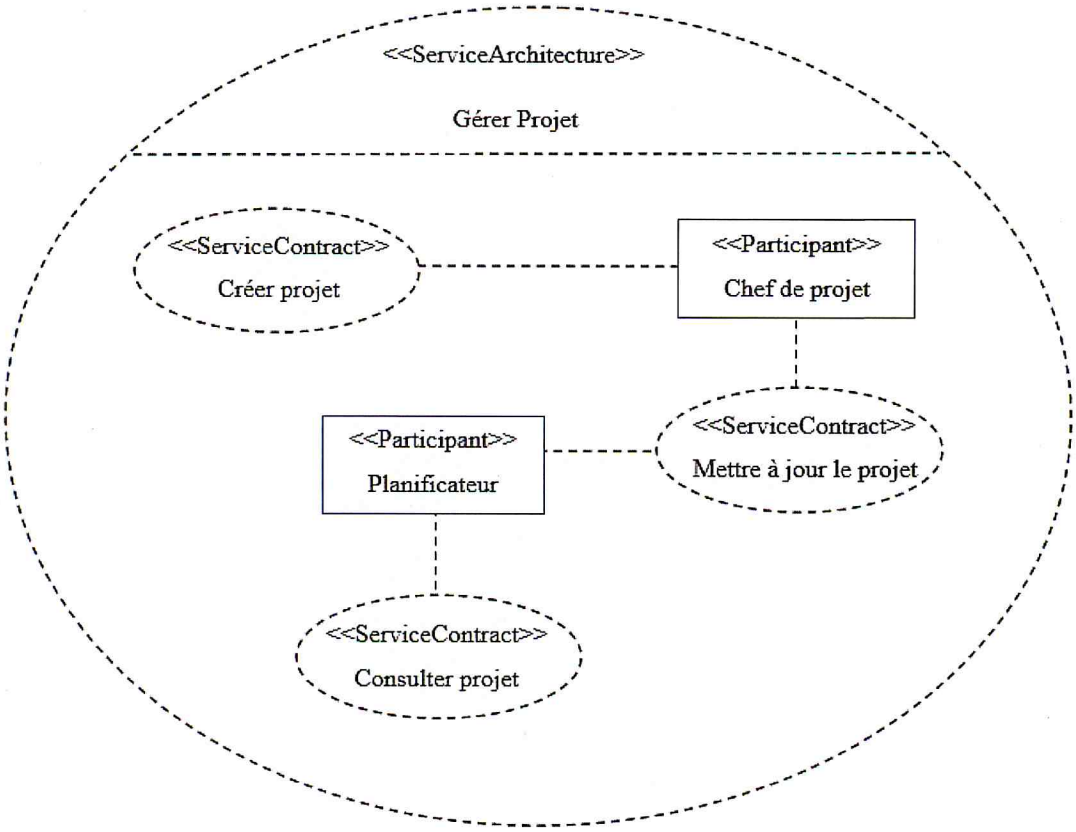


Figure 3-5 Diagramme d'architecture de service de notre système

La figure (3-6) définit les méthodes de création du projet dans notre système.

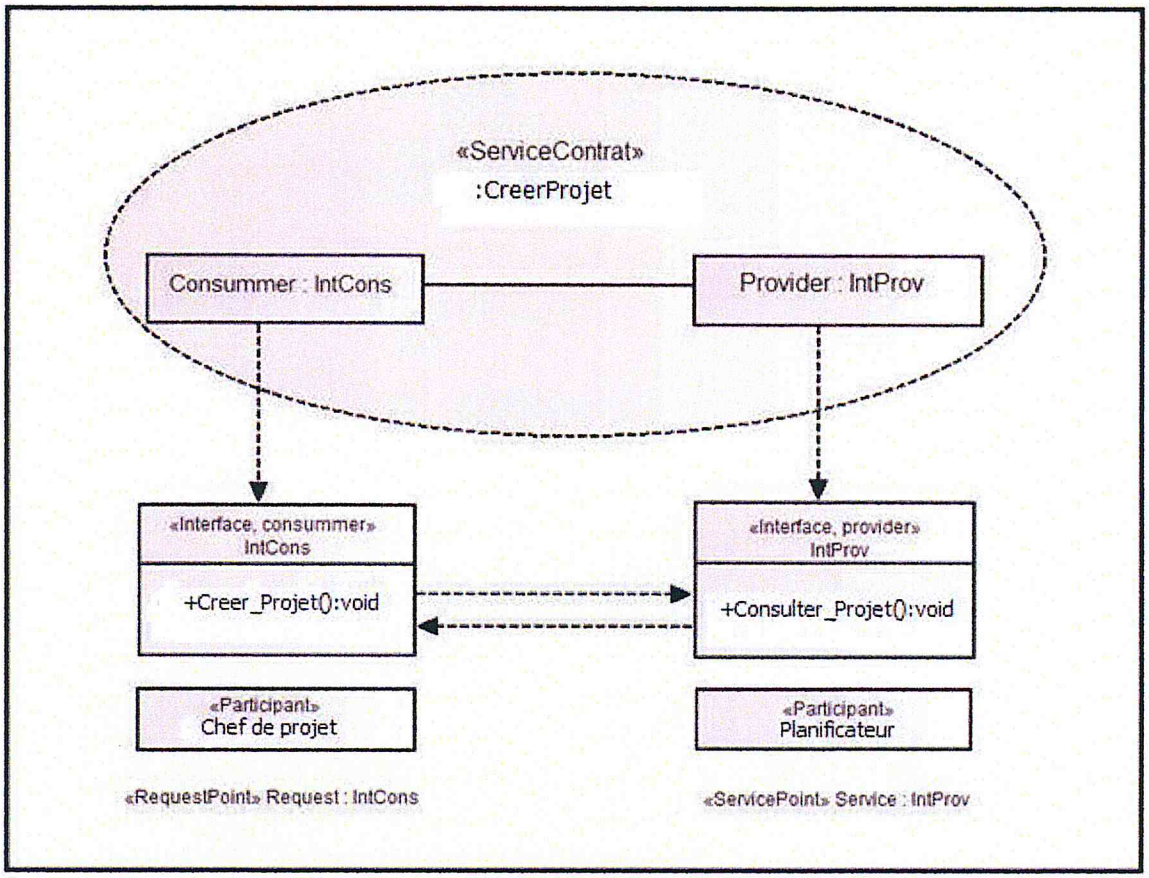


Figure 3-6 Diagramme de contrat de service de la création de projet

La figure (3-7) définit les méthodes de mise à jour du projet dans notre système.

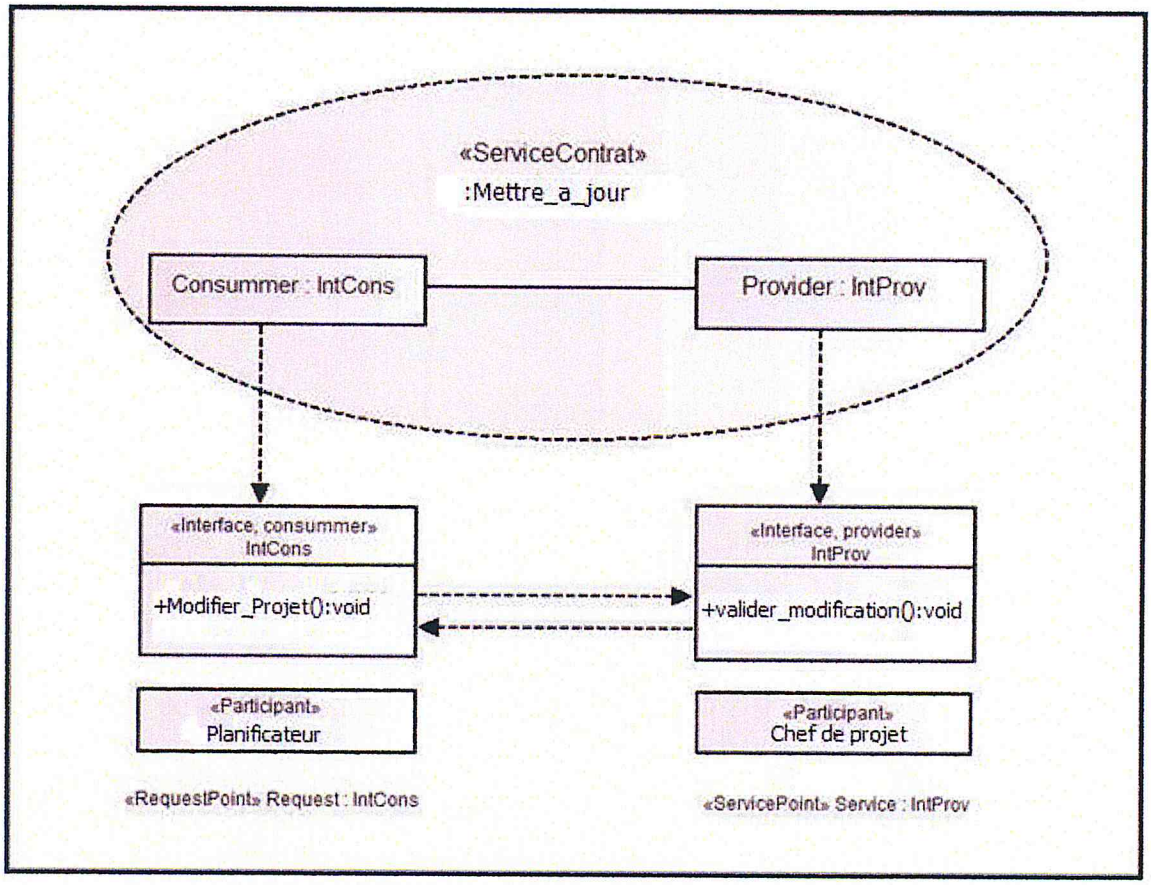


Figure 3-7 Diagramme de contrat de service de la mise à jour de projet

La figure (3-8) définit les méthodes de consultation du projet dans notre système.

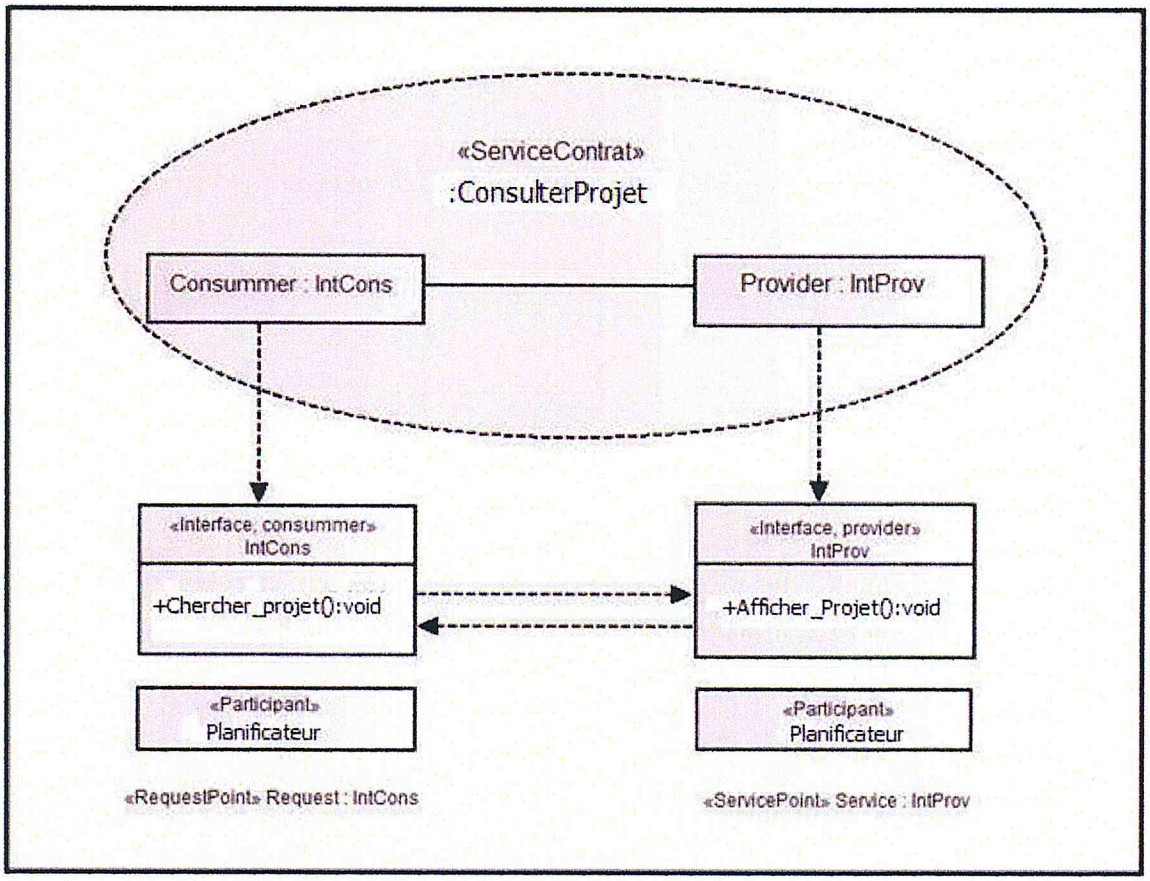


Figure 3-8 Diagramme de contrat de service de la consultation de projet

3.2.4 Vue d'implémentation

Dans la vue d'implémentation, nous utilisons le diagramme de composant pour spécifier les composants de notre diagramme et comment sont-ils dépendants les uns des autres.

3.2.4.1 Diagramme de composant

Le diagramme de composant comporte les composants d'implémentation et leurs relations.

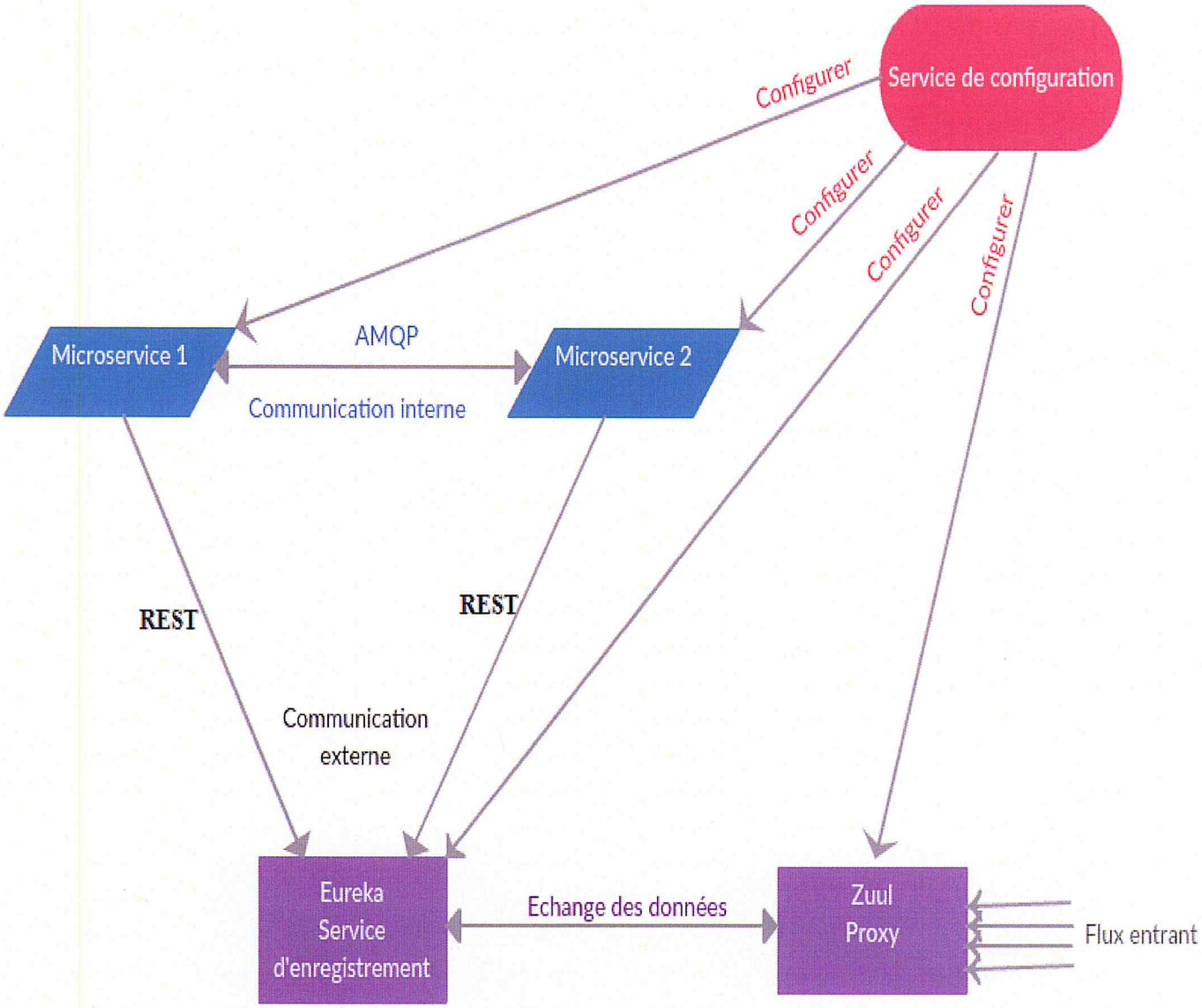


Figure 3-9 Schéma représentant l'architecture générale de notre solution

Notre système se déroule comme suit :

- Étape 1 : Le service de configuration va configurer les deux Microservices, le service d'enregistrement et le proxy (Chacun d'entre eux embarque la librairie client-config).
- Étape 2 : Chacun des Microservices embarque la librairie discovery-client et va s'enregistrer dans un serveur Eureka au démarrage.
- Étape 3 : Pour chaque flux entrant, Zuul va lui découvrir l'ensemble des Microservices connus par Eureka.
- Étape 4 : Zuul va mettre en place les routes http vers les Microservices, jouer le rôle de load balancer (équilibrage de charge ; est une technique spécifique aux serveurs et connections

réseaux qui permet de répartir le trafic entre plusieurs ordinateurs ou lignes physiques. ... Cette répartition de charge va en temps réel envoyer les requêtes sur l'ordinateur le moins utilisés au moment donné).

- Étape 5 : Une fois l'initialisation terminée, Zuul exposera de nouvelles uri, afin de rendre accessible les Microservices où qu'ils soient.

Les deux Microservices communiquent entre eux par le protocole AMQP, pour un échange de données ; et une communication externe par le protocole REST.

3.2.4.1.1 Design patterns utilisés

Les Design Patterns (Patrons de conception), sont des modèles de conception ou encore motifs de conception, ils sont un recueil de bonnes pratiques de conception pour un certain nombre de problèmes récurrents en programmation orientée objet. Pour les Microservices gestion des projets, nous avons choisi le modèle MVC, détaillé dans la figure suivant :

Modèle MVC pour la gestion des projets selon une architecture Microservice

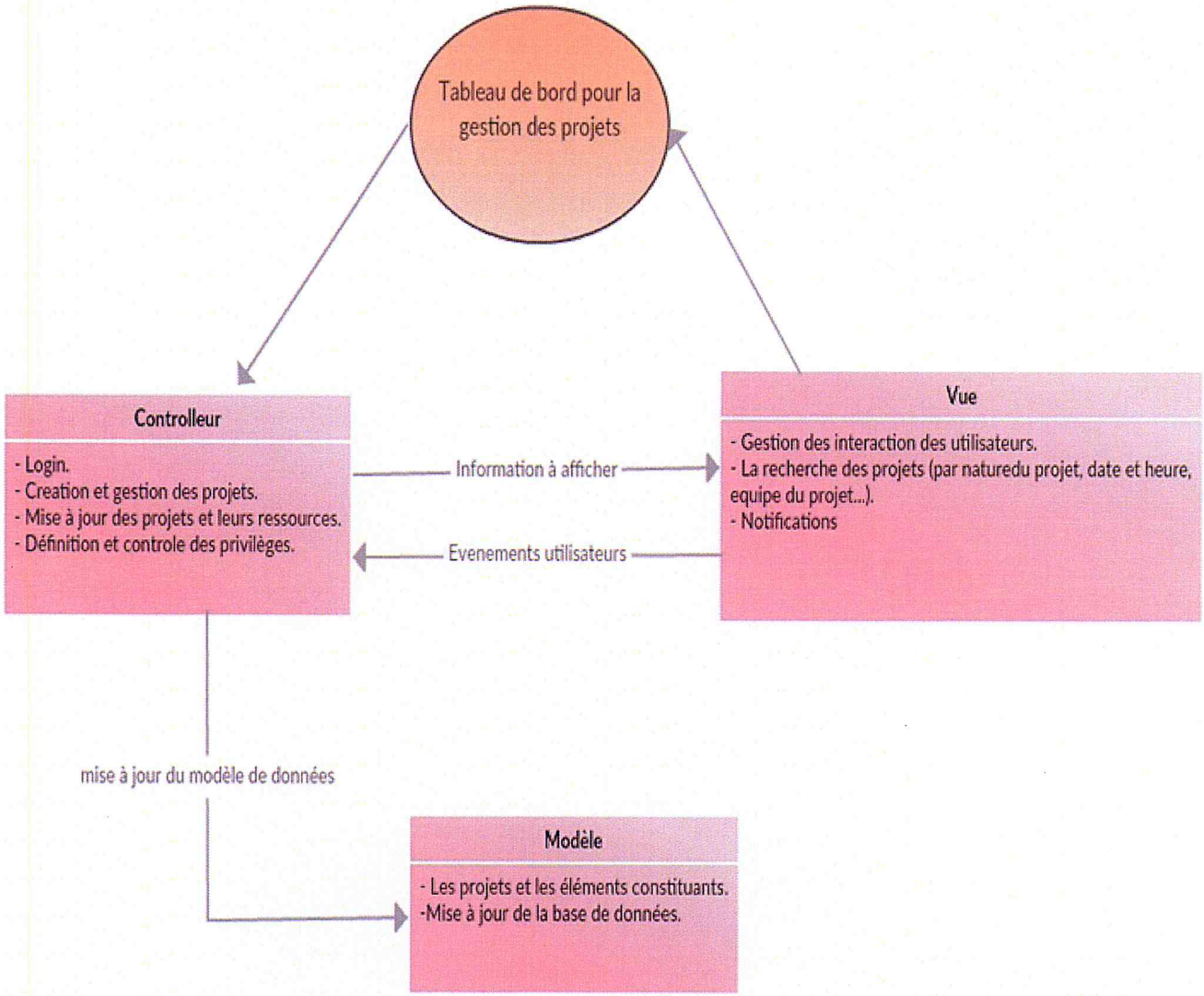


Figure 3-10 Schéma représentant l'architecture Microservice de notre solution

La figure (3-10) représente l'architecture d'un Microservice qui s'occupe de la gestion des projets. Le Microservice gestion des projets, est organisé selon le modèle MVC.

En termes de haut niveau, le modèle MVC signifie qu'une application MVC sera divisée en trois parties :

- Les modèles, qui contiennent ou représentent les données avec lesquelles les utilisateurs travaillent. Il peut s'agir de modèles de vue simples, qui représentent simplement des données transférées entre des vues et des contrôleurs ; ou ils peuvent être des modèles de domaine, qui

contiennent les données dans un domaine métier, ainsi que les opérations, les transformations et les règles de manipulation de ces données.

- Les vues, qui sont utilisées pour rendre une partie du modèle en tant qu'interface utilisateur.
- Les contrôleurs, qui traitent les demandes entrantes, effectuent des opérations sur le modèle et sélectionnent les vues à rendre à l'utilisateur. (Freeman, 2013)

3.2.5 Vue des cas d'utilisations

Dans la vue de cas d'utilisation, nous avons spécifié les besoins via le diagramme de cas d'utilisation. Avant de développer un système, il faut savoir précisément à quoi il devra servir. C'est-à-dire à quel besoin il devra répondre. Donc, la spécification des besoins permet de définir les besoins des utilisateurs du système. Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans pour autant imposer le mode de réalisation de ce comportement. Il permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera.

3.2.5.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation montre les acteurs et l'utilisation du système.

Ce premier diagramme de cas d'utilisation (Figure 3-11) nous donne un aperçu des tâches du chef du projet qui travaille en collaboration avec le planificateur et l'équipe du projet dans le système.

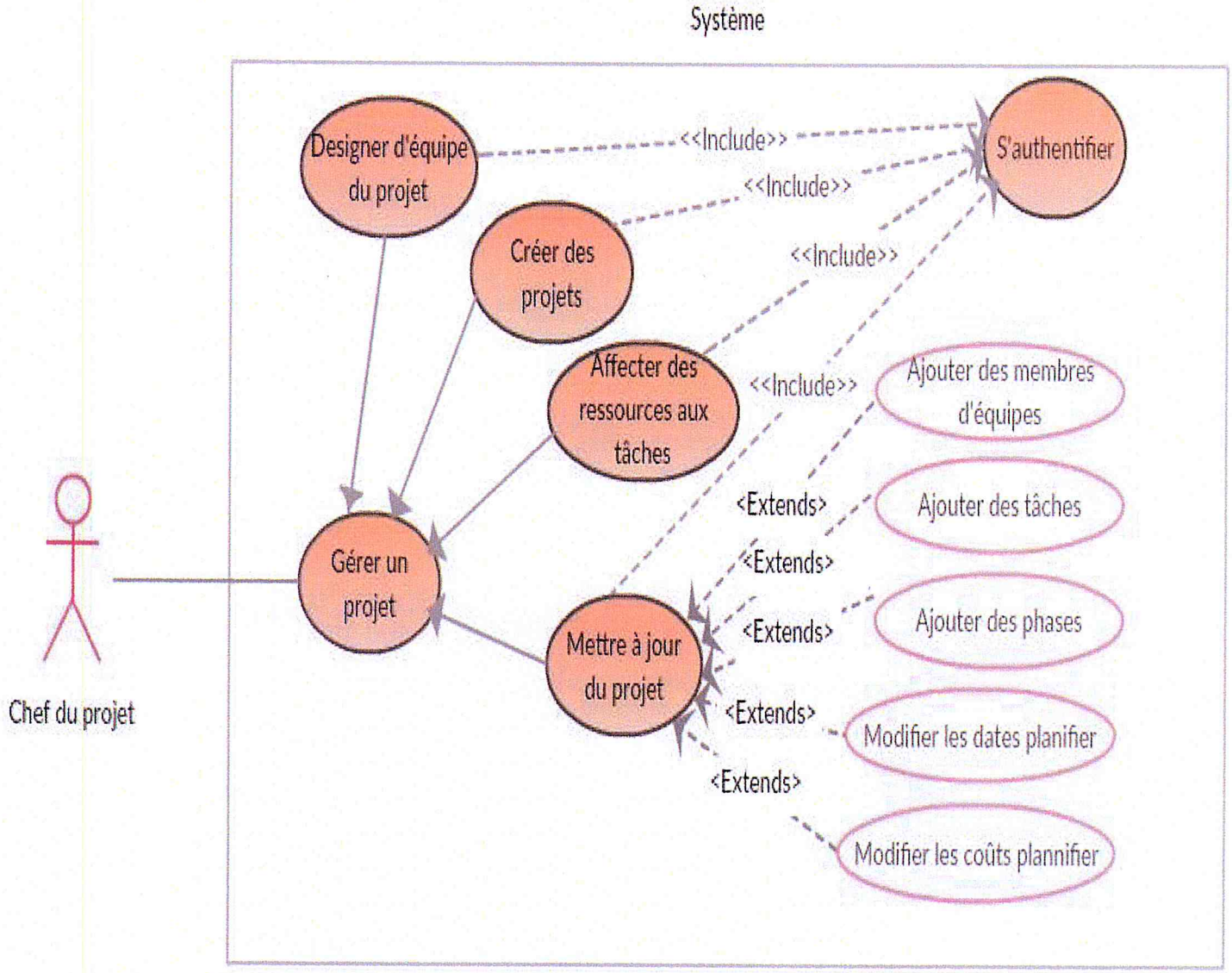


Figure 3-11 Diagramme de cas d'utilisation montre le rôle du chef du projet dans le système

Ce deuxième diagramme de cas d'utilisation (Figure 3-12) nous donne des explications sur la fonctionnalité du planificateur qui est en étroite collaboration avec le chef du projet et l'équipe chargé du projet dans le système.

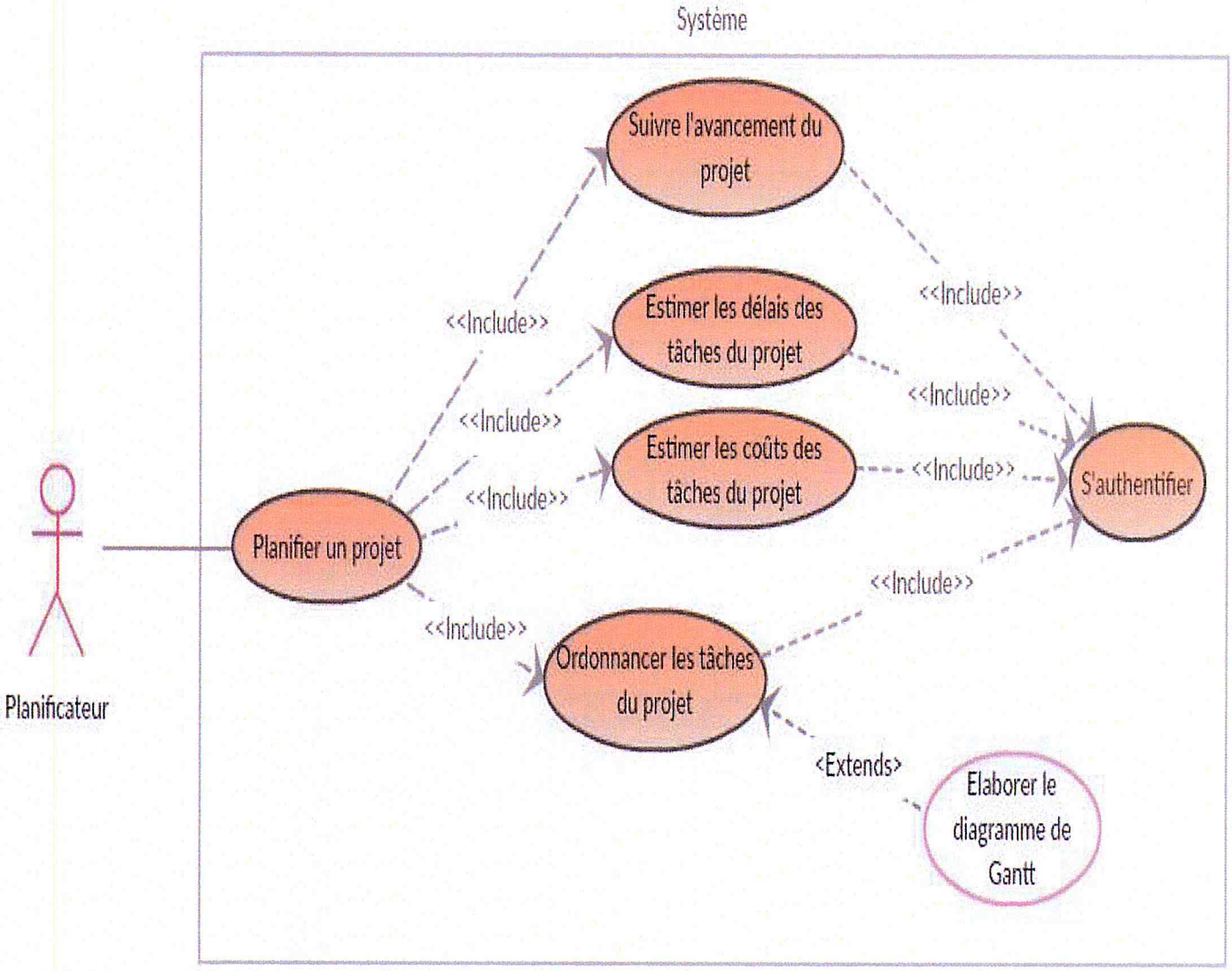


Figure 3-12 Diagramme de cas d'utilisation montre le rôle du planificateur dans le système

Ce dernier diagramme de cas d'utilisation (Figure 3-13) montre la création des comptes utilisateurs et administrateurs, par l'administrateur du système qui va gérer les comptes administrateurs, ces derniers sont les ingénieurs du département, qui vont s'occuper par la suite de créer les comptes utilisateurs. Dans notre système les utilisateurs sont les chefs des projets et les planificateurs qui sont déjà détaillé dans les diagrammes des cas d'utilisations précédents (figure3-11 et figure 3-12).

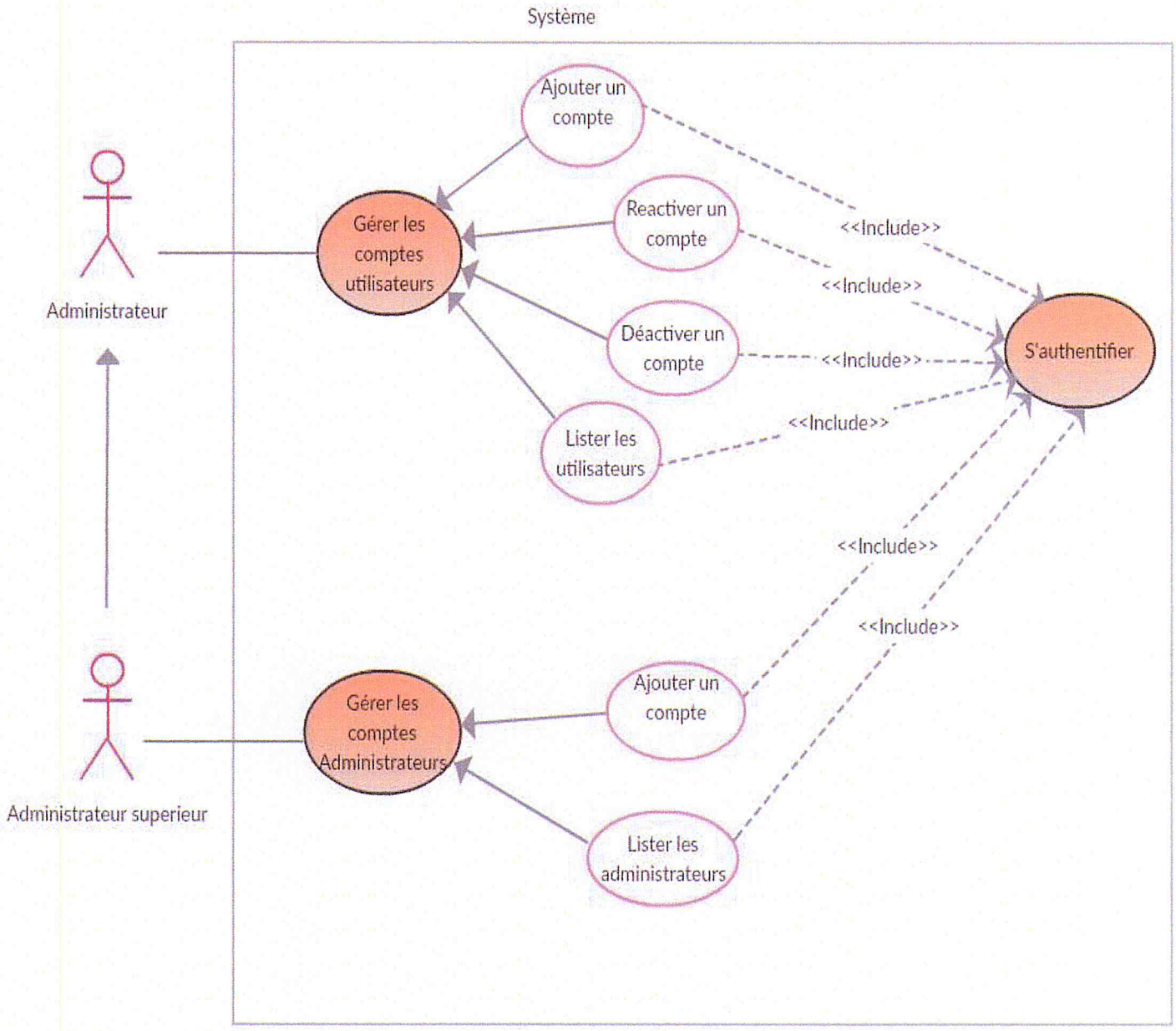


Figure 3-13 diagramme de cas d'utilisation montre la gestion des comptes dans le système

3.3 Conclusion

Dans ce chapitre, nous avons présenté toute la conception et l'analyse de notre système avec les différents diagrammes ainsi que le modèle choisi pour la réalisation de notre solution proposée nous avons décortiqué la problématique générale de notre travail.

Dans le prochain chapitre nous aborderons la partie réalisation ainsi que la description des choix techniques effectués pendant le développement du système demandé.

مكتبة الكلية

Chapitre 4

Réalisation et tests

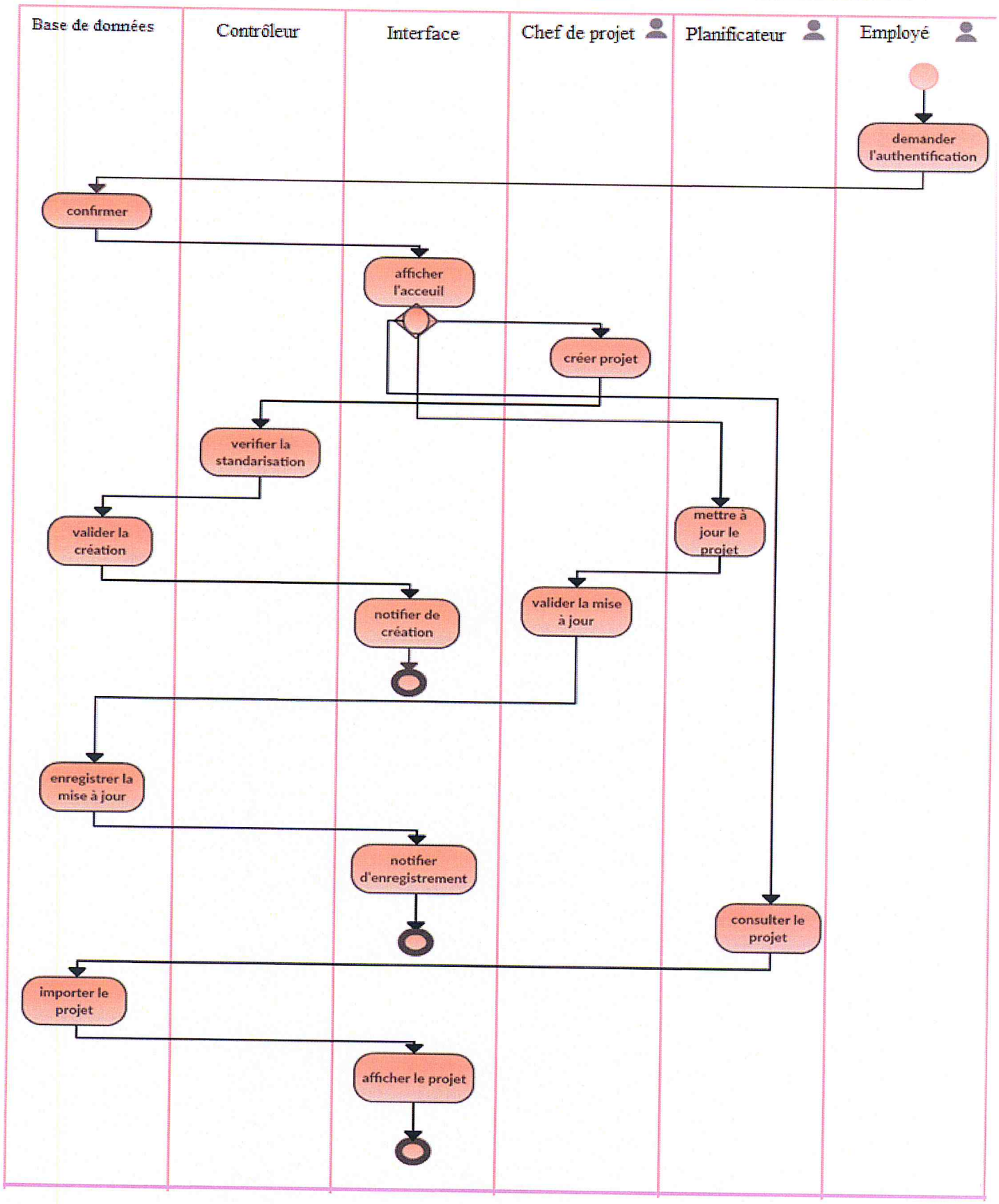


Figure 4-1 Le déroulement de l'architecture générale du système

Le diagramme (Figure 4-1) illustre le processus principal, L'employé déclenche le processus en envoyant une demande d'authentification, l'interface affiche une page d'accueil tout dépend des privilèges et fonctionnalités des employés, s'il était un chef de projet ; il crée le projet, valide les modifications apportés au projet par le planificateur. Et s'il était un planificateur il ne peut que mettre à jour le projet ou le consulter.

4.3 Outils de développement (IDE)

Durant le développement, l'IDE « Eclipse Jee Oxygen » a été utilisé dans sa version « Oxygen.3 (4.7.3) March 2018», sur la plateforme Windows. Cet IDE permet l'édition du code, dispose d'une auto-complétion, d'un débogueur, d'une coloration syntaxique… etc.

4.3.1 Les choix techniques

Pour la réalisation du projet, nous avons utilisé les langages de développent suivant :

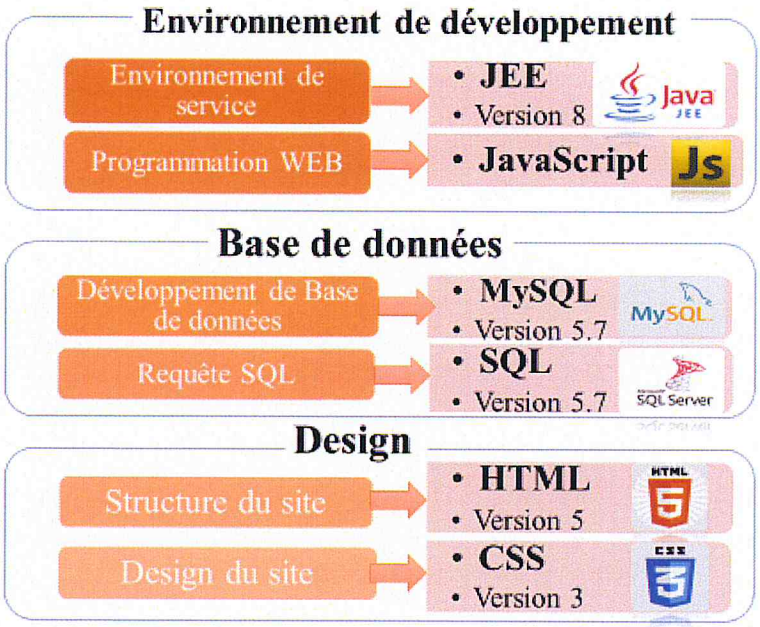


Figure 4-2 Les langages de développement utilisés

4.3.1.1 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.(Daniel Liang, 2011)

4.3.1.1.1 Java EE

La plateforme Java Enterprise Edition (Java EE) est la plate-forme standard de l'industrie pour développer des applications d'entreprise codées dans le langage de programmation Java. Basé sur le fondement solide de la plateforme Java Standard Edition (Java SE), Java EE ajoute les bibliothèques et les services système qui prennent en charge l'évolutivité, l'accessibilité, la sécurité, l'intégrité et les autres exigences des applications d'entreprise. L'objectif de l'évolution de la plateforme, selon la fondation, est de permettre de livrer des implémentations de serveurs d'applications Java compatibles qui répondent aux besoins des développeurs et administrateurs d'applications Java côté serveur et Microservices.

Pour le développement de notre application nous avons utilisé la version 8 de Java EE.

4.3.1.2 JavaScript

JavaScript est le langage de programmation du Web. La majorité des sites Web modernes utilisent JavaScript, et tous les navigateurs modernes (sur les ordinateurs de bureau, les consoles de jeu, les tablettes et les téléphones intelligents) incluent des interpréteurs JavaScript, ce qui fait de JavaScript le langage de programmation le plus omniprésent de l'histoire. (Flanagan, 2011)

JavaScript fait partie de la triade de technologies que tous les développeurs Web doivent apprendre : HTML pour spécifier le contenu des pages Web, CSS pour spécifier la présentation des pages Web et JavaScript pour spécifier le comportement des pages Web : il met vie à une page web qui serait autrement complètement statique. (Flanagan, 2011)

4.3.1.3 SQL

SQL (Structured Query Language) est un langage de base de données. Le résultat d'une requête SQL est une table (également appelée, dans ce contexte, un ensemble de résultats). Ainsi, une nouvelle table permanente peut être créée dans une base de données relationnelle simplement en stockant le jeu de résultats d'une requête. De même, une requête peut utiliser à la fois les tables

permanentes et les ensembles de résultats issus d'autres requêtes. Les perfectionnements ultérieurs ont conduit à de nouvelles versions de la norme SQL. Avec les améliorations apportées au langage de base, de nouvelles fonctionnalités ont été ajoutées au langage SQL pour incorporer des fonctionnalités orientées objet, entre autres. Le dernier standard, SQL : 2006, se concentre sur l'intégration de SQL et XML et définit un langage appelé XQuery qui est utilisé pour interroger des données dans des documents XML.(Beaulieu, 2009)

4.3.1.3.1 Base de données MYSQL

MYSQL est une base de données relationnelles open source. Cette base de données de taille modeste a introduit des millions d'utilisateurs d'ordinateurs et de chercheurs amateurs dans le monde des systèmes d'information puissants. Il peut être installé sans beaucoup de difficile et sophistiqué configuration. MySQL peut fonctionner sur un matériel très modeste et met très peu de pression sur les ressources système ; Beaucoup de petits utilisateurs fournissent des informations à leurs organisations en exécutant MySQL sur des systèmes de bureau modestes. La rapidité avec laquelle il peut récupérer des informations en a fait un favori de longue date des administrateurs Web. (Williams et al, 2007)

4.3.1.4 HTML

HTML (Hyper Text Markup Language) est un langage de balisage, dérivé du langage SGML (Standard Generalized Markup Language). SGML est un langage utilisé pour définir d'autres langues. Un langage de balisage est un langage 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 différent d'un langage de programmation. HTML est utilisé pour créer de l'hypertexte documents qui sont indépendants de la plate-forme. Un document HTML est un fichier texte contient les éléments qu'un navigateur utilise pour afficher du texte, des objets multimédia et des hyperliens. Un lien hypertexte relie un document à un autre document. Hyper liens dans Les textes sont faciles à repérer car ils sont généralement soulignés et d'une couleur différente du reste du texte. (Infosys, 2005)

4.3.1.5 CSS

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 apparaître. 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. (Duckett, 2012)

4.3.2 Framework utilisée

Le terme framework, issu de la langue anglaise, signifie littéralement « structure ». Dans le Web, un framework est en effet un ensemble de composants organisés de manière à proposer un service technique complet à l'utilisateur. En d'autres termes, un framework est une collection d'outils techniques (HTML, CSS, JavaScript, etc.) simplifiant l'organisation ou la réalisation d'un projet web. Il existe des centaines de frameworks destinés aux langages de programmation web comme PHP. Ces derniers s'adressent très souvent à un public averti et ciblé. (Philibert, 2015)

4.3.2.1 Le développent Back end

Le Back end, c'est un peu comme la partie immergée de l'iceberg. Elle est invisible pour les visiteurs, mais représente une grande partie du développement d'un projet web. Sans elle, le site web reste une coquille vide. . Les technologies que nous avons utilisé pour le Back end, sont : SpringBoot, JPA, Hibernate, Tom4.

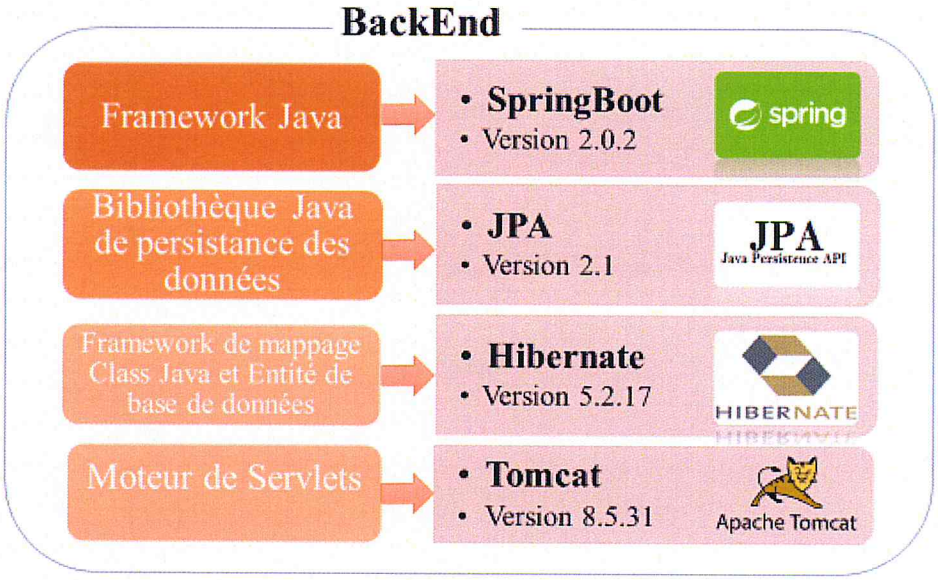


Figure 4-3 Les langages de développement utilisés Back End

4.3.2.1.1 Spring boot

Spring Boot est un framework orienté suivant l'approche " Convention sur la configuration", qui aide à construire des applications Spring rapidement et facilement. L'objectif principal de Spring Boot est de créer rapidement des applications basées sur Spring sans que les développeurs aient à écrire encore et encore la même configuration standard. Ces dernières années, l'architecture des Microservices est devenue le style d'architecture préféré pour la construction d'applications d'entreprise complexes. Spring Boot est un excellent choix pour construire des applications basées sur les Microservices en utilisant différents modules Spring Cloud. (Prasad Reddy, 2017)

4.3.2.1.2 JPA

JPA (Java Persistence API) est un framework de persistance en Java, définie sur la base d'Hibernate. Le modèle de JPA est simple, puissant et flexible. Il peut fonctionner avec d'autres moteurs de persistance. On définit la correspondance entre la structure des classes objet et le schéma relationnel de la base de données ; on manipule directement des objets dans le code ; le framework fait la lecture/enregistrement du contenu des objets sur le support physique. Plus besoin de coder des requêtes SQL, Même si un langage de requête orienté objet reste utile.(Cariou, 2016)

4.3.2.1.3 Hibernate

Hibernate est un framework Java open-source qui simplifie le développement d'applications Java / JEE à partir d'applications simples et autonomes exécutées sur une seule machine JVM, jusqu'à des applications d'entreprise complexes s'exécutant sur des serveurs d'applications complets. Hibernate permet aux développeurs de produire du code évolutif, fiable et efficace. Hibernate 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. (Ahmad Reza Seddighi, 2016)

4.3.2.1.4 Tomcat

Tomcat est un serveur de servlets (des simples classes) Java et un serveur web de la fondation Apache Software. Un serveur Web est, bien sûr, un programme qui affiche des pages Web en réponse à des demandes, par exemple, d'un utilisateur assis sur un navigateur Web. Mais les serveurs Web ne sont pas limités à servir des pages HTML statiques ; ils peuvent également exécuter des programmes en réponse aux demandes des utilisateurs et retourner les résultats dynamiques au navigateur de l'utilisateur. C'est un aspect du Web que Tomcat d'Apache est très bon parce que Tomcat 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 gratuit et un moteur JSP. Il peut être utilisé seul et en conjonction avec d'autres serveurs Web. (Keith & Schincariol, 2013)

4.3.2.2 Le développent Front end

Front end, s'agit des éléments du site (notre tableau de bord), 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. Les technologies que nous avons utilisées pour le front end sont : Angularjs, Bootstrap, jQuery.

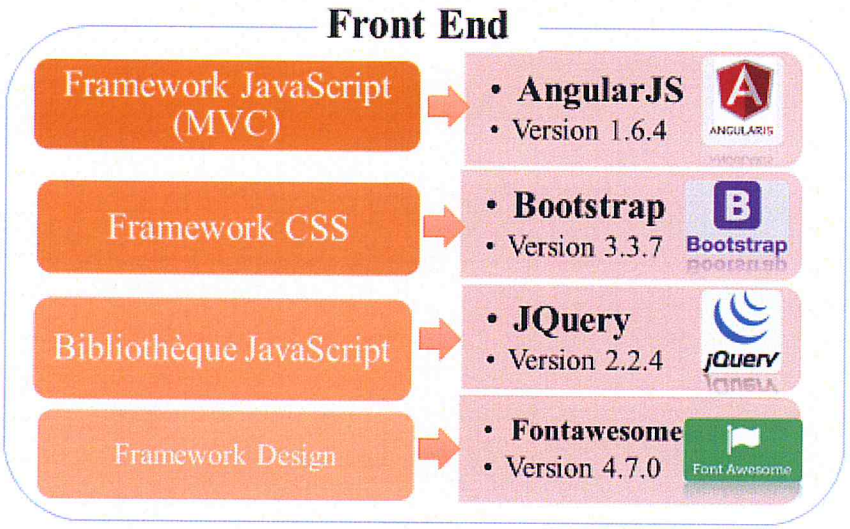


Figure 4-4 Les langages de développement utilisés Front End

4.3.2.2.1 Angularjs

Angularjs est un framework structurel pour les applications web dynamiques qui étend le HTML vocabulaire résultant en un balisage expressif et lisible. Il offre un ensemble d'outils pour la création de pages Web qui va au-delà des structures statiques permises par les versions à base de pain blanc de HTML. Alors que HTML permet au concepteur de créer des pages Web remplies de texte, d'images et de liens, il a peu de flexibilité en matière de logique dynamique et de workflow. Angularjs étend le vocabulaire HTML pour accueillir beaucoup plus de techniques et stratégies, y compris les techniques mêmes utilisées pour authentifier les utilisateurs. Angularjs est entièrement extensible et fonctionne bien avec d'autres bibliothèques. Chaque fonctionnalité peut être modifiée ou remplacée pour répondre à vos besoins uniques en termes de workflow de développement et de fonctionnalités. (Scotch, 2017)

4.3.2.2.2 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. (Philibert, 2015)

4.3.2.2.2.1 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. (ALLARD, 2016)

4.3.2.2.3 JQuery

JQuery est une bibliothèque ou une API (application programming interface) JavaScript c'est à dire un ensemble de fonctions qui permettent d'écrire de manière condensée, les lignes de codes les plus utilisées en JavaScript qui seraient très compliquées à faire soi-même en JavaScript pur. Elle permet de coder plus vite et plus simplement. Son slogan est « write less, do more » et de fait, cette librairie à l'avantage d'avoir une syntaxe courte et facile à comprendre. Elle permet de faire en une ligne ce qui demanderait des centaines de lignes de code en JavaScript pur. De plus, jQuery s'occupe tout seul de la compatibilité avec les différents navigateurs, Sa syntaxe est claire et facilement compréhensible, le seul point négatif étant qu'il impacte les performances techniques car la librairie est assez lourde à charger. (Bauwens ,2013)

4.3.3 Les services utilisés

Nous avons utilisé les trois services suivants :

4.3.3.1 Spring Zuul

Zuul est un service de périphérie qui transfère des demandes à plusieurs services de support. Il fournit une «porte d'entrée» unifiée à votre système, ce qui permet à un navigateur, une application mobile ou une autre interface utilisateur de consommer des services de plusieurs hôtes sans gérer le partage des ressources d'origine et l'authentification pour chacun. On peut intégrer Zuul à d'autres projets Netflix comme Hystrix pour la tolérance aux pannes et Eureka pour la découverte de services, ou l'utiliser pour gérer les règles de routage, les filtres et l'équilibrage de charge sur notre système. (François b. ,2017)

4.3.3.2 Spring Eureka

Eureka est un service basé sur REST (Representational State Transfer) qui est principalement utilisé dans le cloud AWS pour localiser des services à des fins d'équilibrage de charge et de basculement de serveurs de niveau intermédiaire. La découverte de services est l'un des principes clés d'une architecture basée sur les Microservices. La configuration manuellement de chaque client

ou une forme de convention peut être très difficile à faire et peut être très fragile. Eureka est le serveur de découverte de service Netflix et le client. Le serveur peut être configuré et déployé pour être hautement disponible, chaque serveur répliquant l'état des services enregistrés sur les autres. (François b. ,2017)

4.3.3.3 Spring config

Service de configuration Spring Cloud Config, permet de centraliser les configurations du système distribué. Cela dans le but de rendre plus aisé sa maintenance. Cependant, Spring Cloud Config peut être utilisé avec n'importe quelle application exécutée dans n'importe quelle langue. (JULIE, 2016)

4.3.4 Implémentation

La dernière étape pour ce projet est le mettre en fonctionnement.

4.3.4.1 Installation de Spring Tool Suite (STS)

Nous avons installé Spring Tool Suite (STS) dans Eclipse (celui avec JEE).

La méthode la plus simple est de l'installer à partir du Marketplace.

Une fois STS installé, vous avez à disposition une perspective 'Spring'.

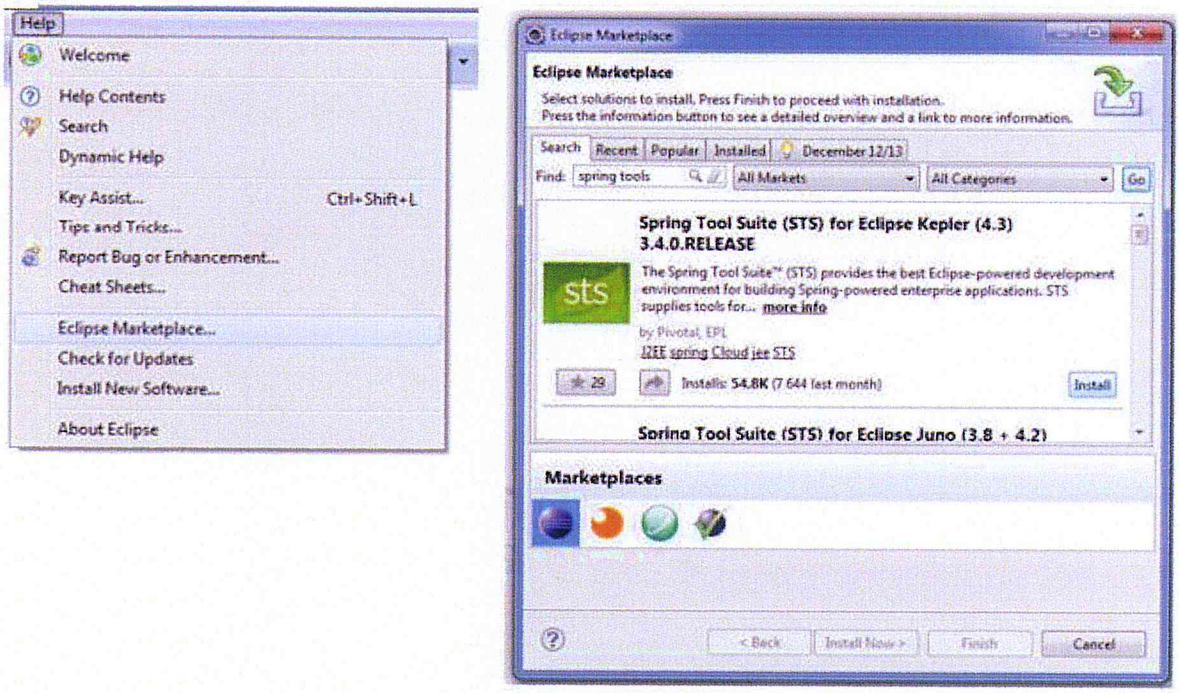


Figure 4-5 Installation du plugin Spring Tools pour eclipse

Un plugin est un outil composé d'un ensemble de fichiers informatiques, et qui permet d'installer des nouvelles fonctionnalités en marge d'un logiciel auquel il est rattaché.

4.3.4.2 Création d'un projet Spring boot

Après l'installation du STS, Eclipse doit être redémarré pour pouvoir créer notre premier projet Spring boot (File => New => Spring Started Project).

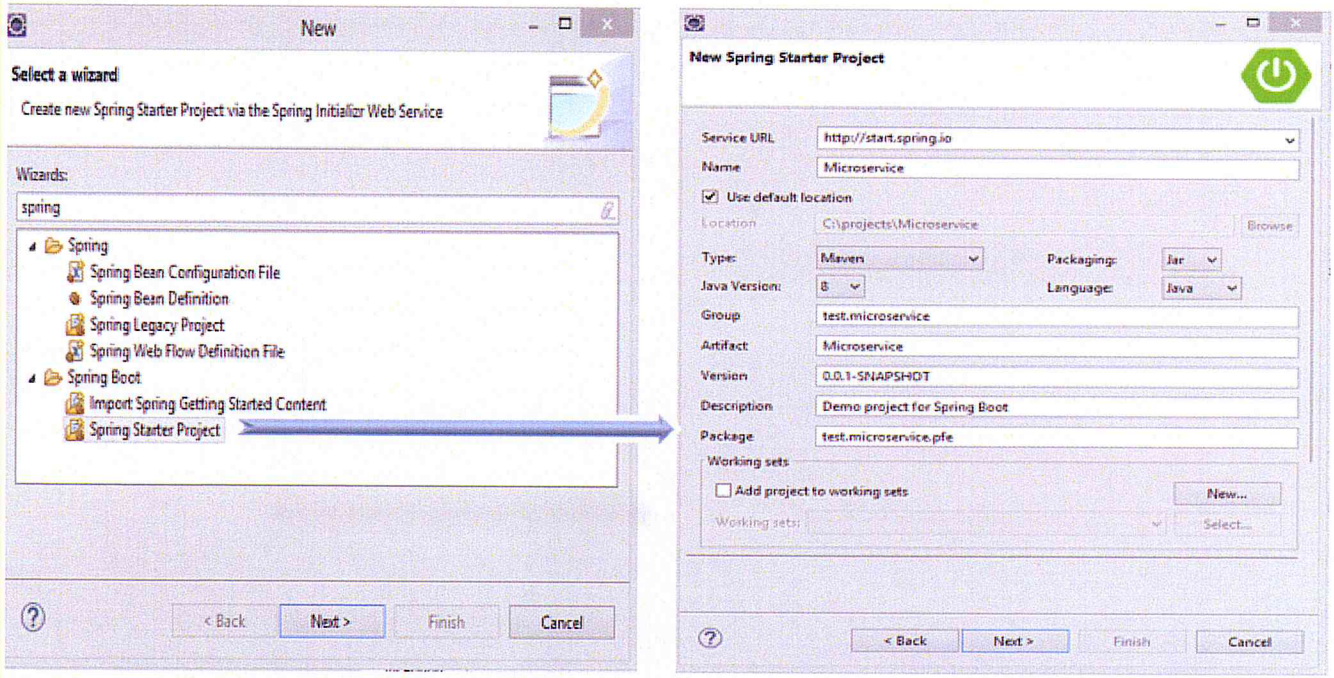


Figure 4-6 La création d'un projet Spring boot sur eclipse

Après avoir nommé le projet (=>Next), nous avons ajouté les dépendances nécessaires pour notre Microservice « Projects_management » (gestion des projets) et spécifier la version de Spring boot utilisée (2.0.2), (puis => Finish).

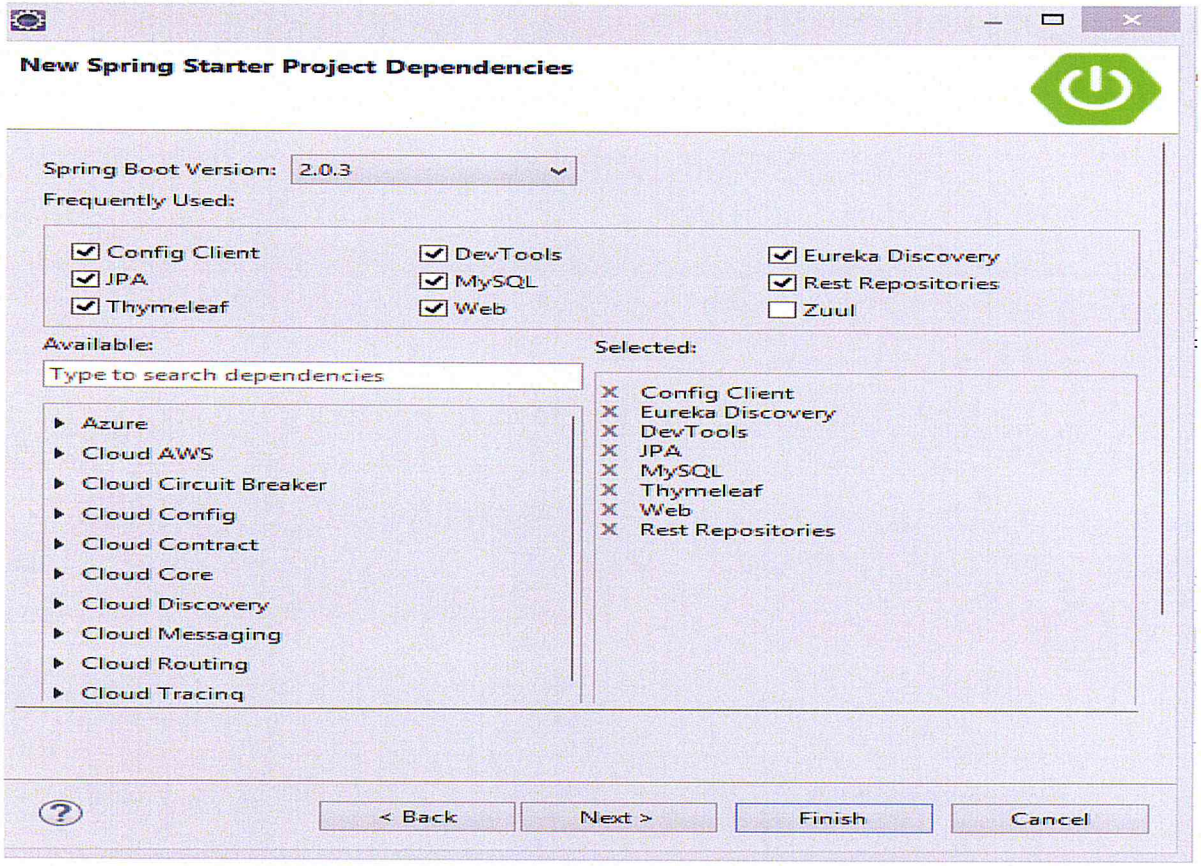


Figure 4-7 L'ajout des dépendances pour le projet Spring boot

4.3.4.3 La structure du projet

Nous avons commencé notre application par la création des Microservices, notre premier Microservice nommé « Projects_management » qu'on va dupliquer par la suite pour avoir deux Microservices.

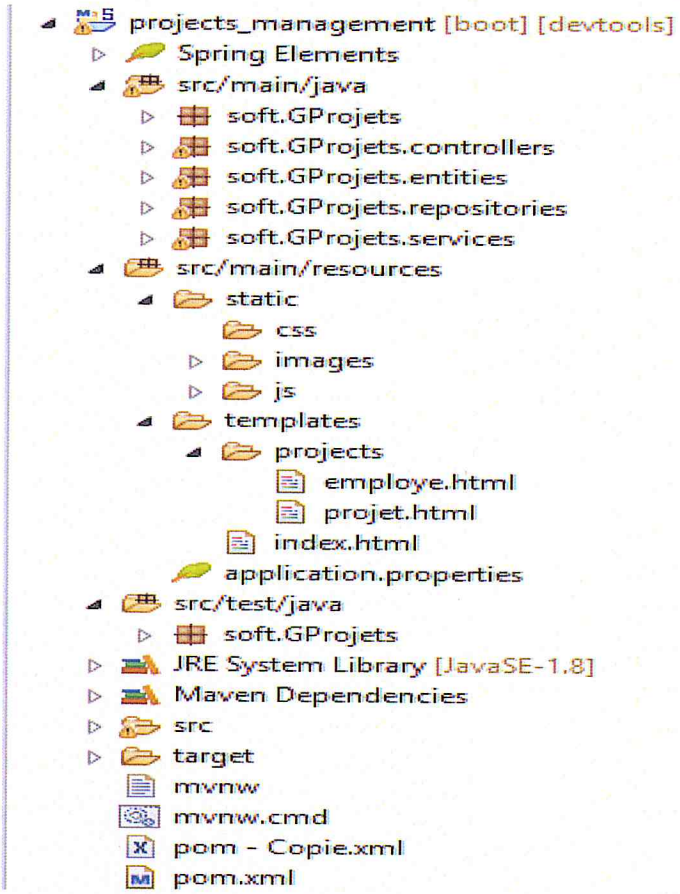


Figure 4-8 La structure du projet Spring boot Microservice « Projects_management »

L'analyse de l'arborescence squelette projet spring boot pour notre premier projet Microservice « Projects_management » :

4.3.4.3.1 Maven

Maven, géré par l'organisation Apache Software Foundation. (Jakarta Project), est un outil pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier. L'objectif recherché est de produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.

Apports :

- Simplification du processus de construction d'une application.
- Fournit les bonnes pratiques de développement.
- Tend à uniformiser le processus de construction logiciel.
- Vérifier la qualité du code.
- Faciliter la maintenance d'un projet.

Pour ne pas stocker toutes les bibliothèques dans notre projet, c'est Maven, qui va télécharger les bibliothèques et les rende disponibles. C'est un outil pour gérer les bibliothèques. Si un besoin d'une nouvelle version de la bibliothèque, il va changer la version et le projet est prêt. Par exemple, Spring peut avoir besoin d'autres bibliothèques, XML, etc. Une fois que la déclaration d'une dépendance sur Spring est faite, Maven téléchargera Spring et toutes les dépendances du Spring.

4.3.4.3.1.1 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 fondamentale 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.

4.3.4.3.1.2 Les dépendances

Une fonctionnalité centrale 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. L'exemple POM référence les coordonnées en tant que dépendance directe du projet. Un projet qui a besoin, disons, de la bibliothèque Hibernate doit simplement déclarer les coordonnées du projet Hibernate dans son POM. Maven télécharge automatiquement la dépendance et les dépendances dont Hibernate a besoin (appelées dépendances transitives) et les stocke dans le référentiel local de l'utilisateur.

4.3.4.3.2 Application.properities (Spring Properties File)

Le fichier des propriétés Spring contient les informations sur la base de données (url, nom de la base de données, nom d'utilisateur, mot de passe…)

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/pfe?usesSSl=false
spring.datasource.username=root
spring.datasource.password=2021
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
#spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.ImprovedNamingStrategy
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL55Dialect


# Keep the connection alive if idle for a long time (needed in production)
#spring.datasource.testWhileIdle = true
#spring.datasource.validationQuery = SELECT 1

server.port=80

#spring.jpa.properties.jadira.usertype.autoRegisterUserTypes = true
#spring.http.multipart.max-file-size=100MB
#spring.jpa.properties.hibernate.current_session_context_class=org.springframework.orm.hibernate5.SpringSessionCo
#spring.jackson.parser.allow-unquoted-field-names=true
#spring.jackson.parser.allow-single-quotes=true

spring.thymeleaf.cache=false
```

Figure 4-9 Le fichier application.properities

4.3.4.3.3 Les entités

Le package « soft.GProjets.entities » contient les classes Java des entités du projet :

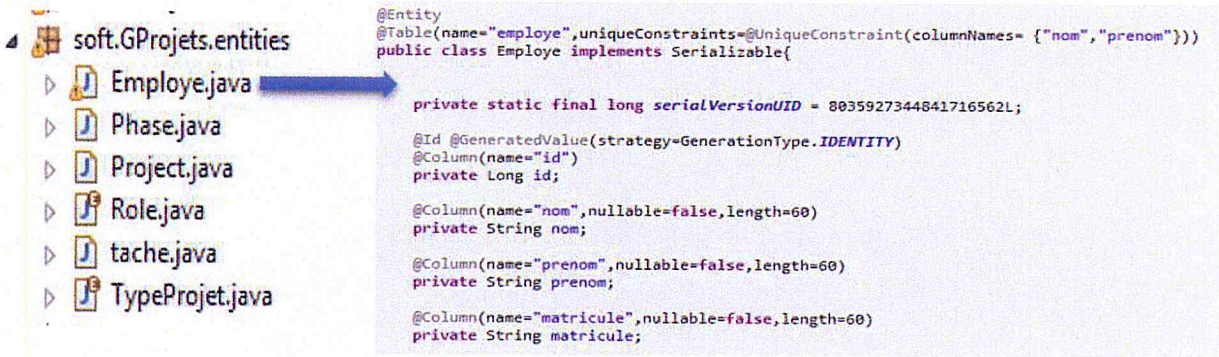


Figure 4-10 Les entités du projet et un exemple du code source

L'annotation @Entity pour dire que cette classe est une entité, @Table : cette classe est une table dans la base de données, @Column : ce champ est une colonne dans cette table, @UniqueConstraint : ce couple (nom, prénom) est unique, @GenerateValue : ce champs est généré automatiquement, @Id : ce champ est un identifiant.

La classe « TypeProjet » est une classe énumération incluse dans la classe « Project », pour pouvoir exiger que le type du projet soit interne ou bien externe seulement d'où l'utilisateur ne peut pas entrer d'autres valeurs. Ainsi la classe « Role », est une classe énumération gère les rôles des employés (utilisateur, administrateur, administrateur supérieur).

On lance Eclipse pour tester la création automatique des entités (Projects_management => Run As => Spring Boot App) :

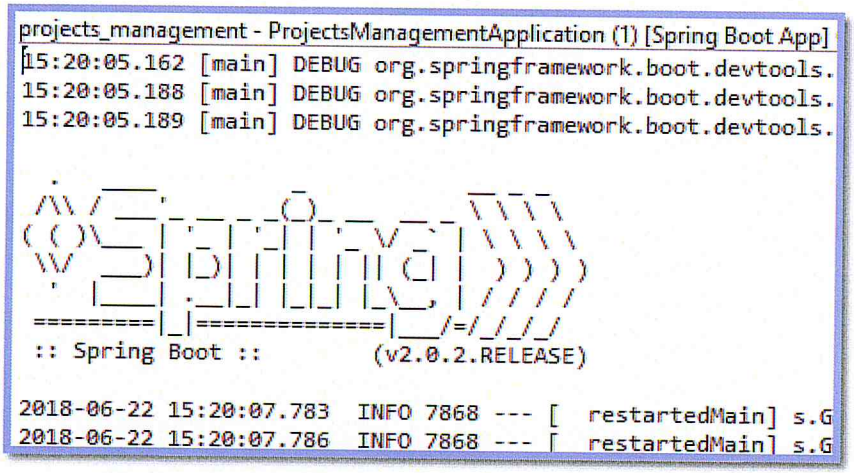
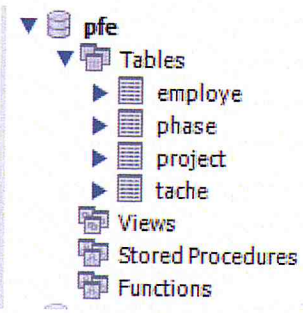


Figure 4-11 Démarrage de l'application Spring boot

Ensuite, on vérifier la création automatique des tables des entités dans notre base de données



« pfe » à partir des entités du projet « Projects_management » :

Figure 4-12 La base de données « pfe » avec les tables crées automatiquement

4.3.4.3.4 Repository

Le package « soft.GProjets.repositories » contient les interfaces Java des entités du projet :

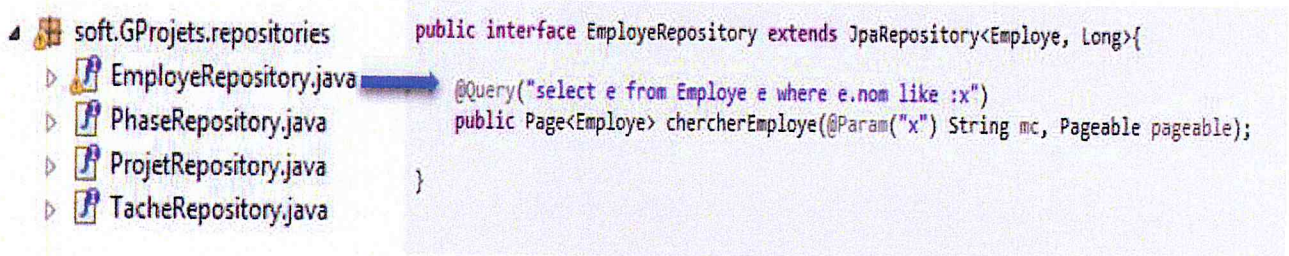


Figure 4-13 Les repositories du projet et un exemple du code source

Repository hérite du JPA et utilise l'annotation @Query pour passer une requête SQL dans un code java, @Param fournit le nom, le type et la description d'un paramètre de fonction. Chaque entité du projet comporte une interface Repository de même que « EmployeRepository ».

4.3.4.3.5 Les services

Le package soft.GProjects.services est structuré comme suite :

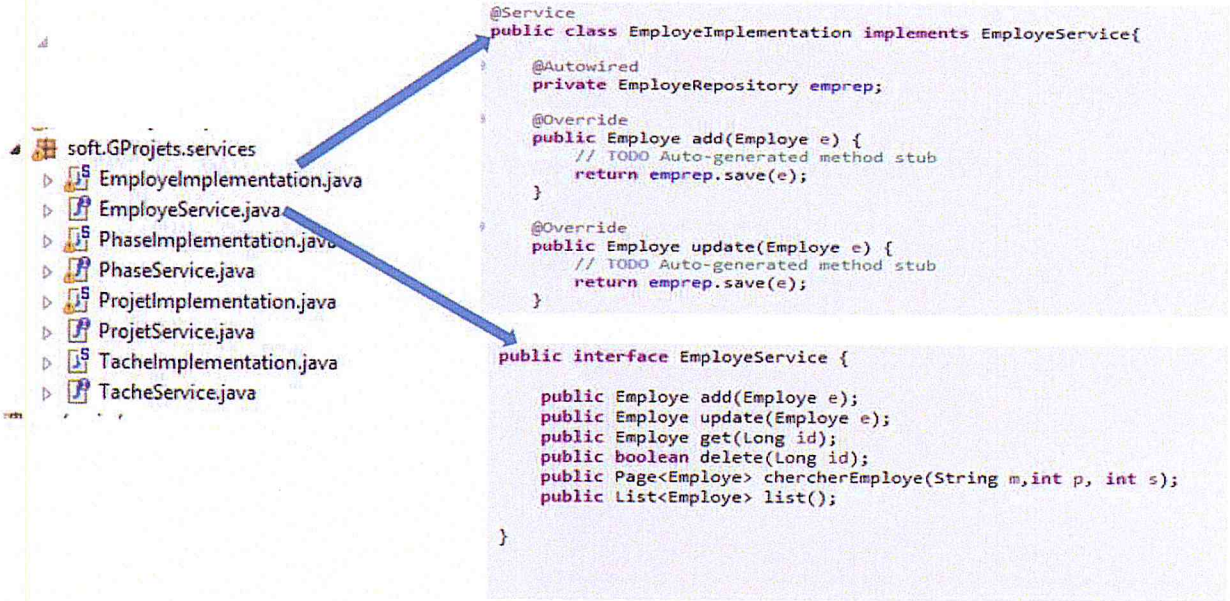


Figure 4-14 Les Services du projet et un exemple du code source

Chaque entité créée comporte une interface et son implémentation. L'interface « EmployeService» contient les différentes méthodes possibles pour la gestion des employés. La classe « EmployeImplementation » contient l'implémentation de toutes les méthodes définies dans l'interface « EmployeService». Pour dire que cette classe est un service, on a l'annotation @Service, @Override pour dire que c'est une implémentation de cette méthode, @Autowired permet de faire l'injection de dépendances (détaillé dans l'annexe) ; signifie que le code précédé de @Autowired peut être réutilisé directement dans d'autres application sans besoin de le redéfinir.

4.3.4.3.5.1 Les contrôleurs

Le package soft.GProjects.controllers est structuré comme suite :

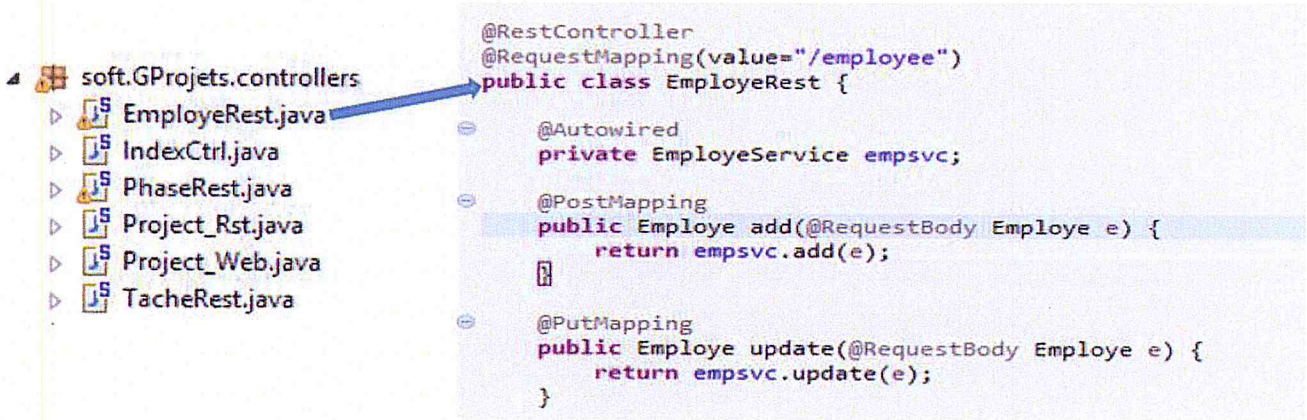


Figure 4-15 Les contrôleurs du Spring MVC et un exemple du code source

Chaque entité créée comporte un contrôleur Spring MVC pour gérer les requêtes. Pour dire que cette classe est un contrôleur, on doit spécifier des annotations. L'annotation @RestController pour dire que c'est un contrôleur Spring MVC, il est généralement utilisé en combinaison avec une annotation @RequestMapping qui mappe les requêtes HTTP aux méthodes de gestion des contrôleurs MVC et REST. Comme dans l'exemple « /employee » va être manipulé par les méthodes : add () et update (). Les annotations @PostMapping et @PutMapping pour dire que c'est une mise à jour ou une création, @RequestBody pour récupérer les valeurs entrer par l'utilisateur.

4.3.5 Teste

Après l'implémentation de notre solution proposée, nous avons testé la fiabilité de l'application.

4.3.5.1 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. (UNOMENA ,2018)

Pour utiliser « Restlet Client » il suffit d'ajout l'extension à chrome (Plus d'outils => Extensions => Ouvrir le chrome web store => Restlet Client => Ajouter à chrome).

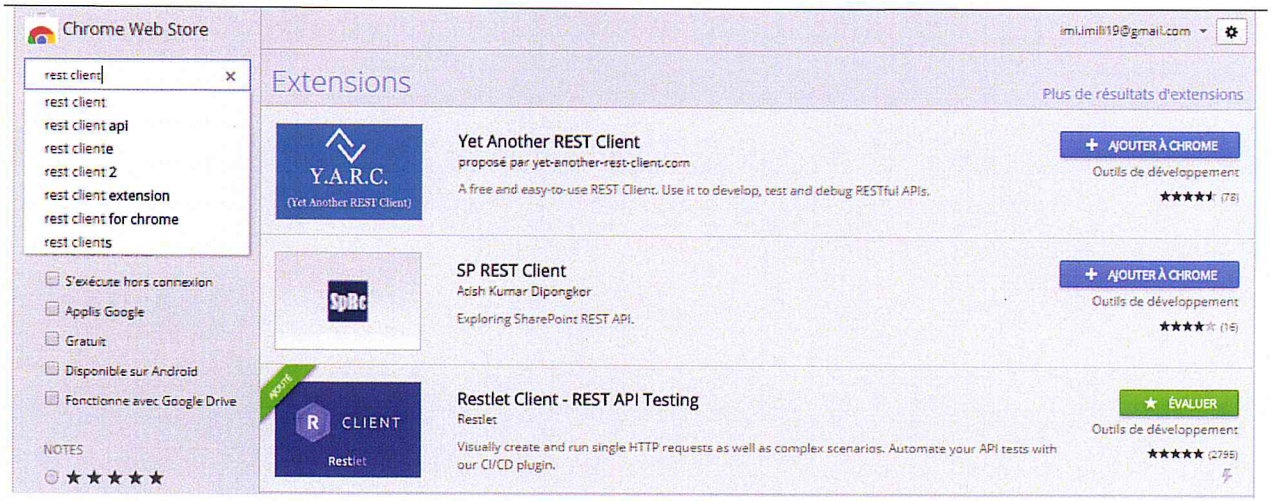


Figure 4-16 L'ajout de l'extension « Restlet Client » à chrome

On lance Eclipse et on exécute notre projet « Projects_management » ensuite on lance « Restlet Client » pour commencer les tests.

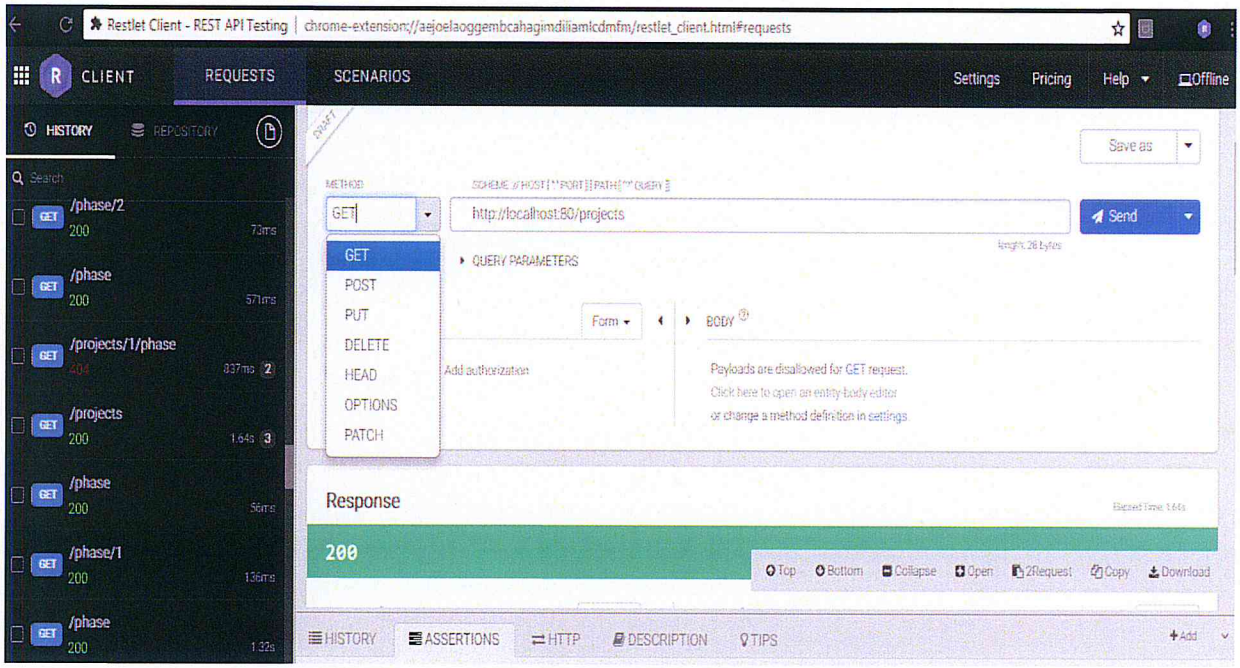


Figure 4-17 L'interface « Restlet Client »

« 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

Après avoir choisi la méthode qui convient, on spécifier le chemin (path) de la requête qu'on veut effectuer (exemple : http://localhost:80/projects => c'est pour dire qu'on va effectuer une requête sur les projets en général.

http://localhost:80/projects/1 => 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).

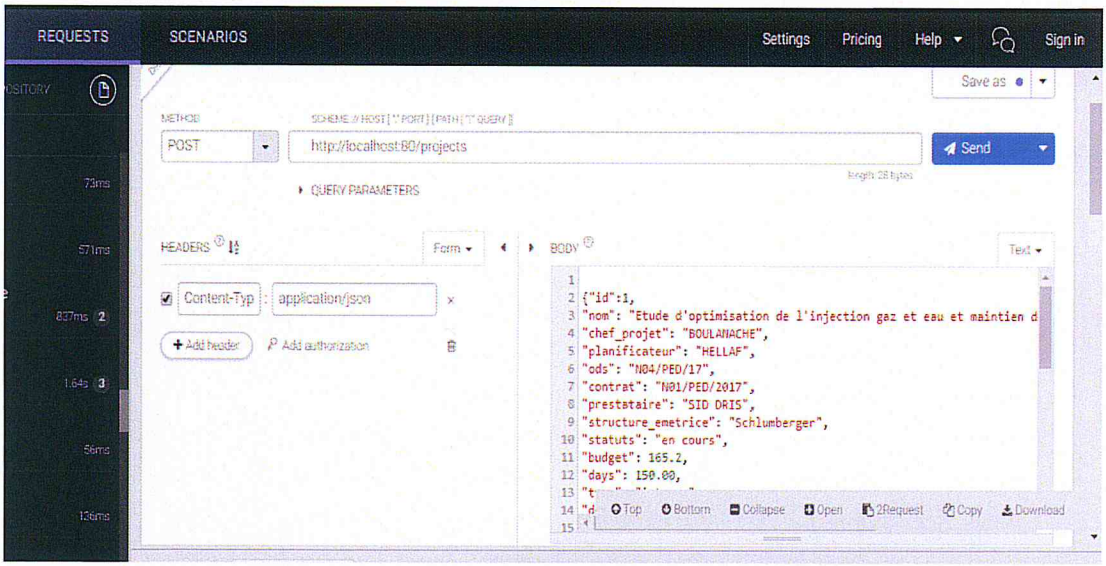


Figure 4-18 Test de mise à jour d'un projet avec « Restlet Client »

On a spécifié le champ « id » du projet parce que c'est une modification par contre la création le champ « id » est créé automatiquement.

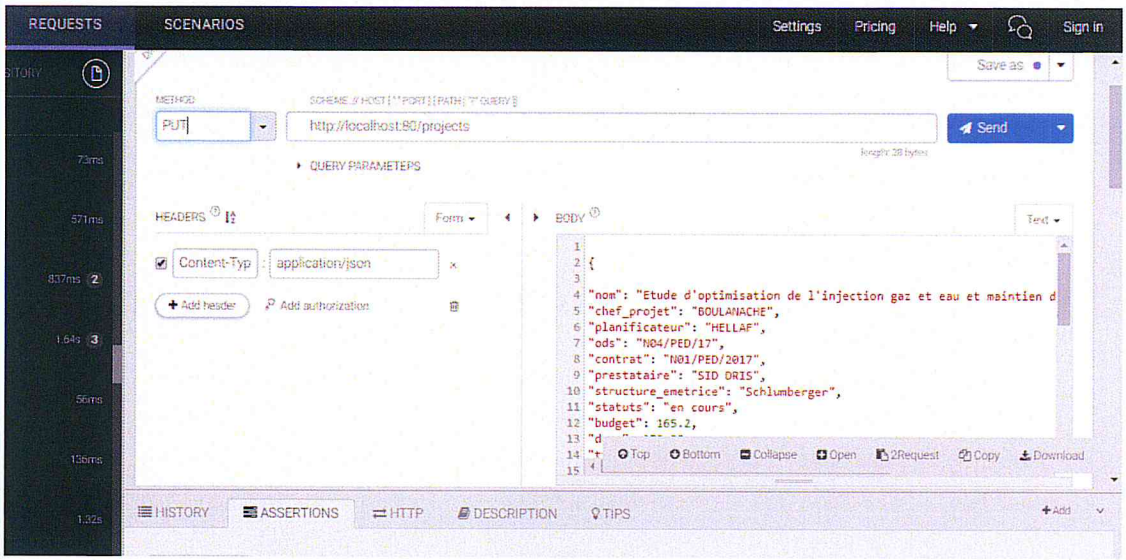


Figure 4-19 Test de création d'un projet avec « Restlet Client »

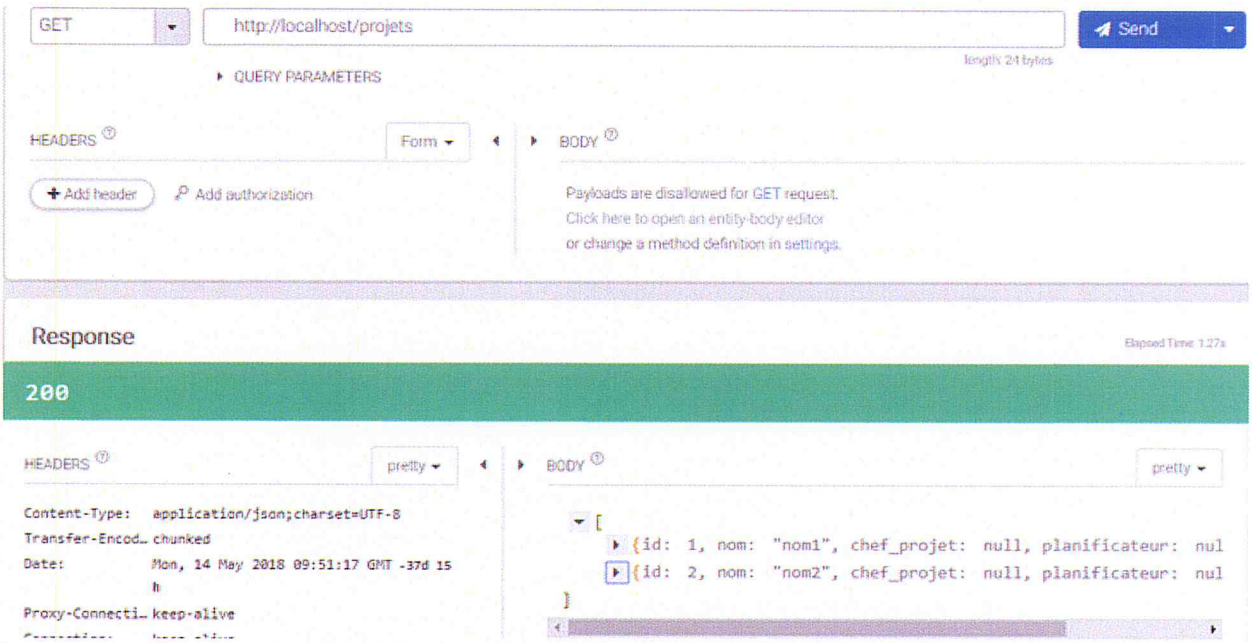


Figure 4-20 Test de récupération de tous les projets existants avec « Restlet Client »

4.3.6 Les interfaces de l'application

Le résultat final de l'application est un tableau de bord, la figure (4.21) montre la page d'accueil et d'authentification :

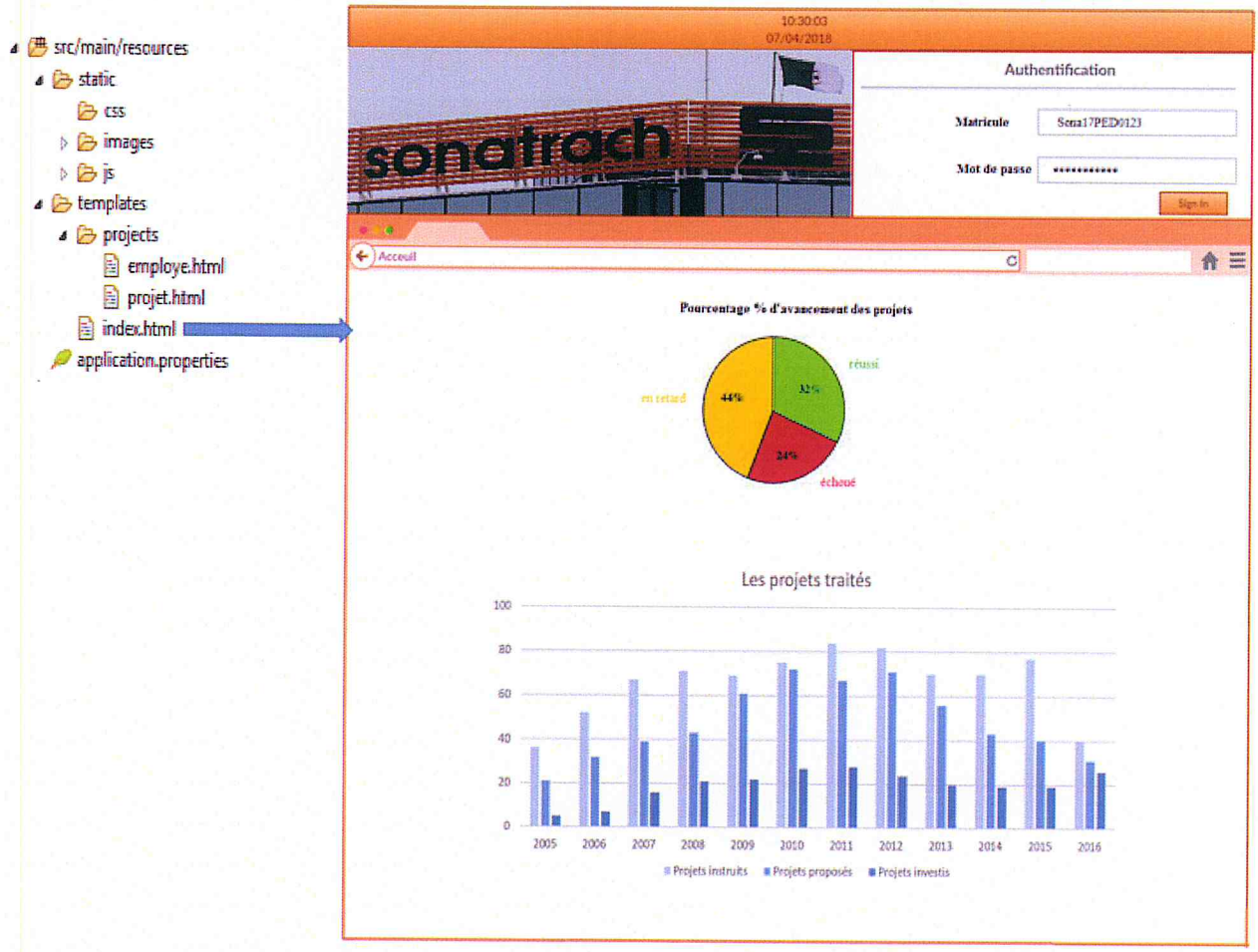


Figure 4-21 L'interface d'accueil et d'authentification

Cette interface comporte des statistiques sur les projets stockés, date et heure actuelles et une zone pour l'authentification. Ensuite une page de gestions des projets va apparaitre avec une listes des projets existants selon les privilèges du l'employé authentifié.

Nous exposons les interfaces liées au Microservice « Projects_management ».

La figure (4.22) présente l'interface d'ajout, de recherche et de mise à jour d'un nouveau projet.

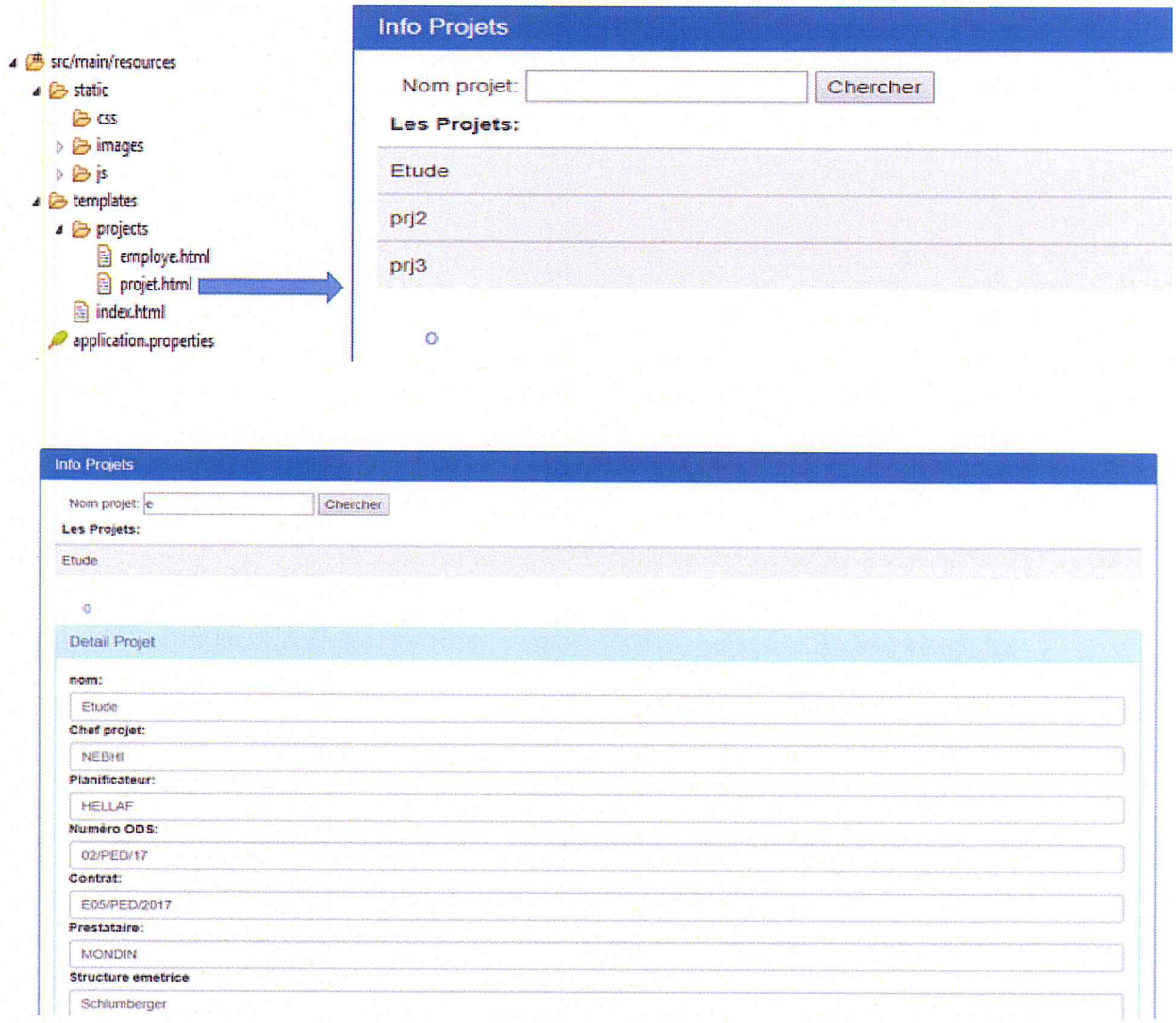


Figure 4-22 L'interface d'ajout, de recherche et de mise à jour d'un nouveau projet

Une liste des projets comporte 3 projets par page, les pages sont numérotés à partir du 0.

Pour chercher un projet il suffit d'entrer son nom, et pour consulter ses détails il suffit du cliquer sur son nom dans la liste des projets. De même pour modifier il suffit de remplacer les champs à modifier puis cliquer sur le bouton « Save » pour enregistrer les modifications apportées. Pour l'ajout, il suffit de cliquer sur le bouton « Add » les champs serons vide pour pouvoir les remplir, comme illustre la figure suivante :

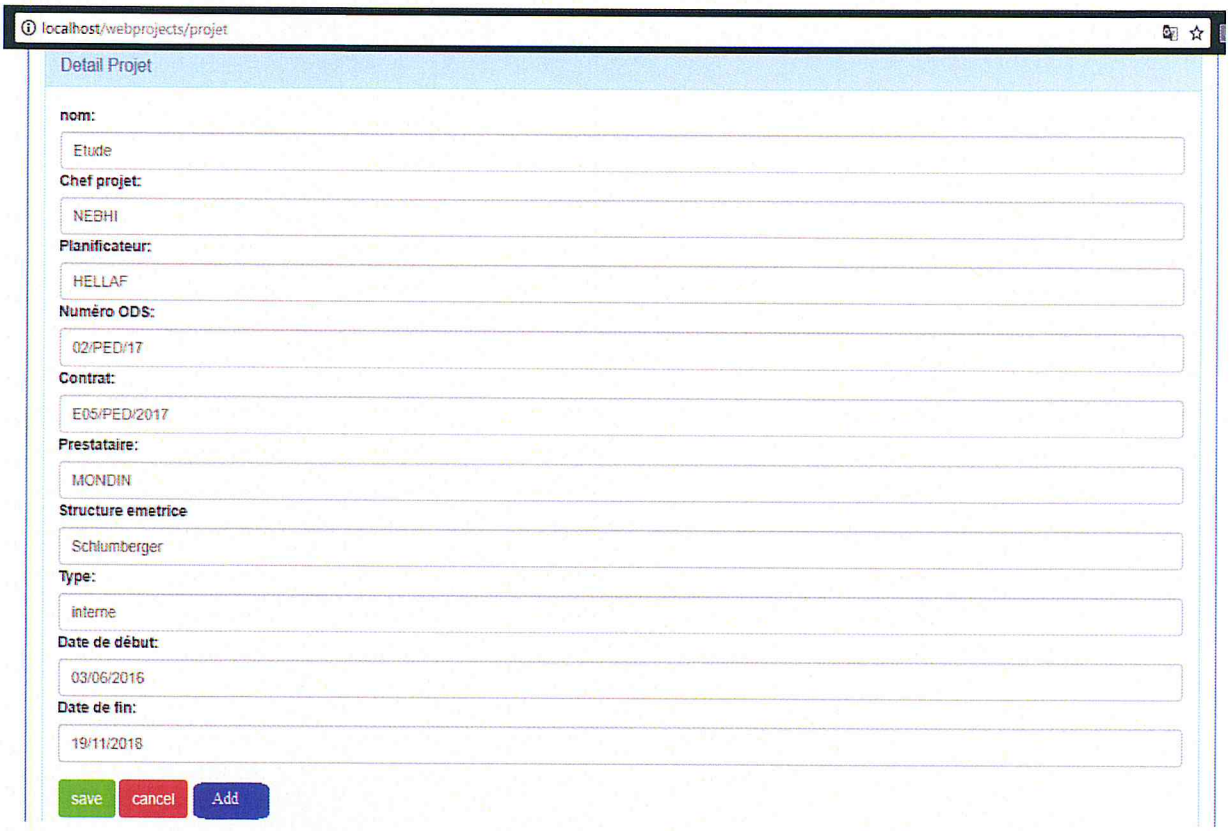


Figure 4-23 L'interface d'ajout, de recherche et de mise à jour d'un nouveau projet

Chaque projet contient une liste des phases et chaque phase contient une liste des tâches, donc en cliquant sur le projet, la liste des phases est affichée de la même manière que celle du projet et de même pour les tâches.

Figure 4-24 L'interface d'ajout, de recherche et de mise à jour d'une nouvelle phase

De même pour les tâches si une phase comporte une liste des tâches, on a la possibilité de les modifier, ajouter, consulter.

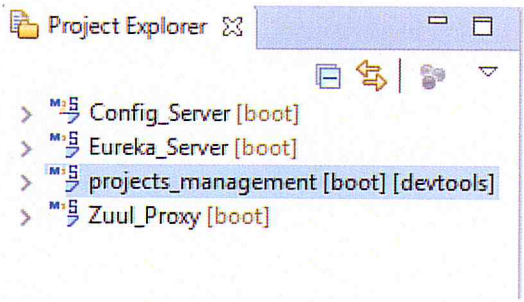


Figure 4-25 La structure des projets de l'application finale

Après avoir bien détaillé le projet « Projects_management », on a aussi ajouter les projets :

- « Config_Server » service de configuration, intégré facilement dans une application Spring boot en utilisant l'annotation @EnableConfigServer , pour localiser les données à configurer on doit spécifier dans application.proprieties (l'url, chemin…) du projet « Config_Server » ainsi le numéro de port (de méme pour les autres services Eureka et Zuul).

```
import org.springframework.cloud.config.server.EnableConfigServer;

@EnableConfigServer
@SpringBootApplication
```

Figure 4-26 La classe « Config_service » du projet « Config_Server »

- « Eureka_Server » service d'enregistrement , intégré facilement dans une application Spring boot en utilisant l'annotation @EnableEurekaServer

```
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@EnableEurekaServer
@SpringBootApplication
```

Figure 4-27 La classe «Eureka_service » du projet « Eureka_Server »

- « Zuul_Proxy » service proxy, intégré facilement dans une application Spring boot en utilisant l'annotation @EnableZuulProxy

```
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@EnableZuulProxy
@SpringBootApplication
```

Figure 4-28 La classe « Zuul_service » du projet « Zuul_Proxy »

Après avoir réaliser les services et le Microservice on lancer notre application :

On va dupliquer le Microservice «Projects_management» pour assurer la disponibilité, on lance le même projet sur deux ports différents, comme ça on a deux Microservices pour la gestion des projets (Run As=> Run Configurations => Arguments et on spécifier numéro de port –server 83).

4.4 Conclusion

Au cours de ce dernier chapitre, nous avons réussi à développer le Microservice « Projects_management » et les services : proxy, enregistrement et de configuration (Zuul, Eureka, config). Nous avons aussi détaillé l'architecture globale de notre solution présentée dans le chapitre précédent. Nous avons réussi à tester les communications entre notre Microservices et l'utilisateur en utilisant « Restlet client » déjà mentionné dans ce chapitre. Nous avons aussi réussi à tester les communications entre les Microservices et les différents services en suivant la chaine de déploiement décrite dans chapitre précédent.

Conclusion générale

Conclusion générale

Notre projet de fin d'études effectué au sein de Sonatrach a abouti à une refonte de l'application .Nous avons commencé par une identification des limites liées à une architecture Microservices et un diagnostic technique de la solution existante (l'utilisation du logiciel MS Project).

Ensuite, nous étions responsables de la mise en place de l'architecture de la solution proposée ainsi que la collecte des besoins, la conception qui comporte, La réalisation et les tests. Le travail réalisé n'a pas été une simple migration mais une création complète de la solution. Nous avons pu rependre à travers cette refonte aux besoins liées aux limites du MS Project.

Le choix conceptuel de l'architecture 4+1 vues a bien décortiqué la problématique en 5 vues chacune son objectif, les points de vues qu'elle traite, une notation pour le plan architectural correspondant et les outils que l'on utilise pour la décrire et la gérer. L'utilisation des Microservices comme une solution pour la gestion des projets a pu résoudre les problèmes de disponibilités et d'accessibilités en temps réel causé par l'absence d'une plateforme distribuée dédiée à la division PED.

Ce stage était bénéfique, il nous a initiées au monde des Microservices qui est un sujet d'actualité, ainsi que l'utilisation des différentes technologies de programmations récentes tels que Spring boot, Angularjs, Bootstrap…

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

En effet, nous avons pu développer les besoins fonctionnels et non fonctionnels initialement dégagés et chaque Microservice a pu passer par la chaîne de déploiement avec succès.

Faute de temps, nous avons travaillé que sur la gestion de projets. Comme perspectives, Nous envisageons d'implémenter la gestion des contrats, des ressources humaines… prochaines objectives de l'équipe. Il sera toujours intéressant d'ajouter un module pour superviser l'application et les temps des réponses des composants de système.

Annexes

Annexe A : L'injection de dépendances

L'injection de dépendances est un concept clé avec AngularJS. Angular n'est pas le seul framework à utiliser ce système qui est extrêmement important aujourd'hui. Symfony2 exploite également l'injection de dépendances.

Dans la théorie, l'injection de dépendances permet à des modules de ne pas se soucier de l'instanciation des modules dont ils dépendent. Il suffit d'appeler les dépendances et Angular se charge de les instancier et de les injecter pour nous. (Rémi Michel, 2018)

Il existe de multiples intérêts à pratiquer l'injection de dépendances :

- **La simplicité :** c'est très pratique pour un développeur. Plus de soucis du comment instancier les modules que nous utilisons. Cela suit le principe du "least knowledge". Lorsqu'on développe quelque chose, on n'a pas envie de nous soucier des autres composants, on va juste les utiliser. (Rémi Michel, 2018)
- **La fiabilité :** Lorsque notre module est chargé, nous avons la certitude que toutes ses dépendances sont chargées et que nous avons la possibilité de les utiliser. (Rémi Michel, 2018)
- **La réutilisabilité :** Nous le verrons lorsque nous aborderons la partie des services, mais il s'agit d'un point très important. Lorsque nous développons des services permettant par exemple de faire des conversions de dates, il y a fort à parier que vous souhaiteriez pouvoir réutiliser ce module dans d'autres projets. L'injection de dépendances permet donc d'inciter les développeurs à créer de petits modules unitaires et à les assembler par la suite pour créer des systèmes plus conséquents. (Rémi Michel, 2018)
- **Les tests :** C'est un point extrêmement important. Si le module que nous souhaitons tester possède 10 dépendances, il est assez embêtant d'avoir à instancier les 10 modules afin de pouvoir juste tester notre module. À la place, nous allons dire au système d'utiliser des bouchons qui vont se comporter comme nos dépendances. (Rémi Michel, 2018)

Bibliographie

(ALLARD, 2016) Denis ALLARD. Fontawesome. [En ligne]. http://www.studiovitamine.com/blog/blogcomment-utiliser-la-bibliotheque-dicones-font-awesome/ [page consultée le 09/06/2018 à 18:30h].

(Baude, 2008) F. Baude, Audrey Occello Engineering School of Technology, University of Nice (UNS) – France (2008).

(Bauwens ,2013) Bauwens Nicolas, JavaScript/jQuery, Management Bruxelles Formation Cepegra Multimediatic, Belgique, (2013), 66 pages.

(Baxter et al., 2018) Rayen Baxter and the Spring team. Spring Security. [En ligne]. https://spring.io/projects/spring-security [page consultée le 10/06/2018 à 15h].

(Beaulieu, 2009) Beaulieu Alan, Learning SQL, Published by O'Reilly Media, Printed in the United States of America, (2009), 312 pages.

(Benzarara, 2017) Benzerara Yakoub, Etude et mise en place d'un cloud privé, H2IT Ibn Roch en partenariat avec 3il Limoges, Algerie,(2017), 86 pages.

(Bottard et al., 2018) Eric Bottard and the Spring team. Spring Cloud. [En ligne]. http://projects.spring.io/spring-cloud/[page consultée le 15/06/2018 à 05 :30h].

(Cariou, 2016) Eric Cariou, Java Persistence API, Université de Pau et des Pays de l'Adour

(Carneiro & Schmelmer, 2016) Cloves Carneiro Jr., Tim Schmelmer, Microservices From Day One, Library of Congress Control, USA, (2016), 246 pages.

(Daniel Liang, 2011) Y. Daniel Liang, Introduction to java programming, Eighth Edition, Armstrong Atlantic State University, (2011), 73 pages.

(Dedieu, 2006) Olivier Dedieu, Pluxy : un proxy Web dynamiquement extensible, Unité de recherche INRIA Rocquencourt Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex France, (24 Mai 2006), 22 pages.

(Duckett, 2012) Duckett Jon, Design and Build, (2012), volume 336, 1078 pages.

(Durand, 2004) Armel Durand, Maîtrise d'Œuvre des projets informatiques, Dunod, Paris, (2004), 270 pages.

(El Montassir, 2007) EL Montassir Amina, Management de projet Methodes gantt et pert, École nationale de commerce et de gestion de Settat, Maroc, (2007), 39 pages.

(Fernandes et al, 2013) J.Fernandes, Joel Rodrigues, Sana Ullah, I.Lopes, Performance evaluation of RESTful web services and AMQP Protocol, Institut de Télécommunication, University of Beira Interior, Covilhã, Portugal & King Saud University, Riyadh, Saudi Arabia, (juillet 2013), 815 pages.

(Fielding et al, 1999) R. Fielding, UC Irvine, J. Gettys, HyperText Transfer Protocol -- HTTP/1.1, USA, (juin 1999), 129 pages.

(Flanagan, 2011) Flanagan David, JavaScript : The Definitive Guide 6th Edition, Published by O'Reilly Media, Printed in the United States of America, (2011), 994 pages.

(François b. ,2017) François b. Les Microservices pour une architecture orientée web n°5. [En ligne]. https://blog.talanlabs.com/microservices-partie-5-spring-cloud/[page consultée le 10/06/2018 à 13h].

(Freeman, 2013) Adam Freeman, Pro ASP.NET MVC 5, Italie, (23 décembre 2013), 785 pages.

(Gadge & Kotwani, 2017) Sanjay Gadge, Vijaya Kotwani, Microservice Architecture : API Gateway Considerations, Edition GlobalLogic 1741 Technology Dr. San Jose, CA 95110, USA, (2017), 13 pages.

(Infosys, 2005) Infosys ; HTML, CSS, JavaScript ; France, (2005), 144 pages.

(Isoz, 2017) Vincent ISOZ, Ingénierie de GESTION DE PROJETS Guide non exhaustif pour scientifiques & ingénieurs V10.0, Paris, volume 1680, (2017-07-03).

(JULIE, 2016) Bastien JULIE. Spring Cloud- architecture Micro-services. [En ligne]. https://www.supinfo.com/articles/single/3638-spring-cloud-architecture-micro-services[page consultée le 12/06/2018 à 10h].

(Keith & Schincariol, 2009) Keith Mike, Schincariol Merrick, Pro JPA 2, Apress Edition, Printed and bound in the United States of America, 508 pages.

(Kruchten , 2015) Philippe Kruchten, Plans Architecturaux – Le Modèle d'Architecture Logicielle à « 4+1 » Vues, University of British Columbia – Vancouver, Rational Software Canada, (03 février 2015), 218 pages.

(Laude, 2016) Bruno Laude, Chapitre PMI France Livre annuel 2015-2016, PMI, France, (2016), 42 pages.

(Mazzara & Meyer, 2017) Manuel Mazzara, Bertrand Meyer Present and Ulterior Software Engineering, Springer International Publishing, Bertrand, (novembre 2017).

(Moine, 2013) Jean-Yves Moine, Le grand livre de la gestion de projet, AFNOR, France, (2013), 430 pages.

(Philibert, 2015) Benoît Philibert, Bootstrap 3 Le framework 100 % web design, ÉDITIONS EYROLLES 61, bd Saint-Germain 75240 Paris Cedex 05, www.editions-eyrolles.com. (2011), 311 pages.

(PMI, 2004a) Project Management Institute, Guide du Corpus des connaissances en management de projet Troisième édition, ANSI, USA, (2004), 390 pages.

(PMI, 2004b) Project Management Institute, A Guide to the Project Management Body of Knowledge Third Edition, ANSI, USA, (2004), 390 pages.

(Prasad Reddy, 2017) K.Siva Prasad Reddy, Beginning Spring Boot 2 : Applications and Micro-services with the Spring framework, Hyderabad, India, (2017), 304 pages.

(Rémi Michel, 2018) Rémi Michel, L'injection de dépendances, [En ligne]. https://openclassrooms.com/courses/developpez-vos-applications-web-avec-angularjs/l-injection-de-dependances-1 [page consultée le 20/06/2018 à 14 :45 h].

(Roger, 2011) Roger Aïm, les fondamentaux de la gestion de projet, AFNOR, France, (2011), 42 pages.

(Rota,2008) Véronique Messager Rota, Gestion de projet Vers les méthodes agiles, Eyrolles, France, (2008), 252 pages.

(Scotch, 2017) Scotch, AngularJS Best Practices Guide, SecureAuth Corporation, USA, (juin 2017), 63 pages.

(Seddighi, 2016) Ahmad Reza Seddighi, Spring Persistence with Hibernate, PACKT Publishing, BIRMINGHAM – MUMBAI, (2016), 265 pages.

(Tchokpon, 2017) Romaric Tchokpon, L'approche processus à l'heure du numérique : Le Business Process Management, Business Analyste- Expert BPM, University of Milano, Italie, (2017), 50 pages.

(UNOMENA ,2018) UNOMENA, What Restlet Client does ?, [En ligne]. https://restlet.com/documentation/client/user-guide/introduction [page consultée le 19/06/2018 à 21h].

(Williams et al, 2007) Williams, Seyed M.M. "Saied" Tahaghoghi and Hugh E., Learning MySQL, Published by O'Reilly Media, Printed in the United States of America, (2007) 622pages.

جامعة البليدة 1
مكتبة
الكلية
كلية العلوم