République Algérienne Démocratique et Populaire Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

Faculté de : Génie électrique et d'informatique
Département : Informatique

Mémoire de fin d'études
En vue de l'obtention du diplôme de
Master en Informatique
Spécialité : Ingénierie des systèmes d'informations

Présenté par :

LEHARA Yasmine et **TAMAZIRT Malika**

Thème : Migration d'une application monolithique vers l'architecture microservices (cas EPAL)

Proposé par : Mr. BOUFOULA

Encadrer et diriger par : Mme. BELKADI

Mr. BOUFOULA

Soutenue le **13/12/2020**

Devant le jury composé de :

Mme. AOUDJIT Rachida	Présidente de jury.
Mme. CHEMOUNE Karima	Examinatrice.
Mme. BELKADI Malika	Directrice de mémoire.
Mr. BOUFOULA Azeddine	Co-Promoteur.

Année universitaire : 2019/2020.

Remerciement

En guise de reconnaissance, nous tenons à témoigner mes sincères remerciements à toutes les personnes qui ont contribués de près ou de loin au bon déroulement de notre stage de fin d'étude et à l'élaboration de ce modeste travail.

Nos sincères gratitudes à Madame **BELKADI** pour la qualité de son enseignement, ses conseils et son intérêt incontestable qu'elle porte à tous les étudiants.

Nous tenons à remercier aussi Monsieur **BOUFOULA** pour sa patience, ces conseils pleins de sens et pour le suivi et l'intérêt qu'il a portait à nos travaux.

Dans l'impossibilité l'impossibilité de citer tous les noms, nos sincères remerciements vont à tous ceux et celles, qui de près ou de loin, ont permis par leurs conseils et leurs compétences la réalisation de ce mémoire.

Enfin, nous n'oserions oublier de remercier tout le corps professionnel de l'université Mouloud Mammeri, pour le travail énorme qu'il effectue pour nous créer les conditions les plus favorables pour le déroulement de nos études.

Dédicaces

Toutes les lettres ne sauraient trouver les mots qu'il faut. . .

Tous les mots ne sauraient exprimer la gratitude, L'amour, le respect, reconnaissance. . .
Aussi, c'est tout simplement que je dédie cette thèse :

À MES CHERS PARENTS

Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être.

Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours. Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez. Puisse Dieu, le Très Haut, vous accordent santé, bonheur et longue vie et faire en sorte que jamais je ne vous déçoive.

À MES CHERS FRÈRES ET MA CHÈRE SŒUR

HADIA ; la prunelle à mes yeux et la douce au cœur

LYES ; le jumeau a moi que j'aime profondément

NASSIM; mon petit frère que j'admire énormément

En témoignage de mon affection fraternelle, de ma profonde tendresse et reconnaissance, je vous souhaite une vie pleine de bonheur et de succès et que Dieu, le tout puissant, vous protège et vous garde. À LA MÉMOIRE DE MON CHER ONCLE "Dada MOH". J'aurais aimé que tu sois présent pour qu'on puisse partager ce moment de joie avec moi ; malheureusement la vie est ainsi personne ne la choisi. Que dieu le plus puissant t'accorde dans son paradis ; Je t'aime

À MA CHERE BINOME MALIKA

À MES GRANDS-PARENTS, MES CHERES TANTES ET MES CHERS ONCLES

Je vous remercie tellement pour votre encouragement de toujours et votre soutiens ; mon cœur est rempli d'affection vers vous tous

À MES CHERS AMI(E) S ET COLLEGUES D'ETUDE

En souvenir de notre sincère et profonde amitié et des moments agréables que nous avons passés ensemble ; Vous allez trouver dans ce travail l'expression de respect et de tendresse envers vous

L.Yasmine

Dédicaces

Avec l'expression de ma reconnaissance, je dédier ce modeste travail a ceux qui, quels que soit les termes embrassés, je n'arriverais jamais à leurs exprimer mon amour sincère

À l'homme mon précieux offre de Dieu qui doit ma vie, ma réussite et tout mon respect

Mon cher père

À la femme la plus précieuse à celle qui a souffert sans me laisser souffrir, qui n'a jamais dit non âmes exigences et qui n'a jamais cessé de me rendre heureuse

Ma chère maman

À mes chères sœurs Samira, Amélia, Lynda, Katia qui n'ont pas cessé de me conseiller, encourager et soutenir tout au long de mes études que Dieu les protège Et leurs offres la chance et le bonheur

À mon adorable petite sœur Wissam qui sait toujours comment procurer la joie Et le bonheur pour toute la famille

À mon très cher Frère Ghano que j'admire énormément

À mon ami et celui que je considère Mon frère celui qui me comprend et qui N'a jamais cessé de me soutenir Momoh

À mes amies Rosa Kahina

À ma binôme Yasmine

À tous ceux qui m'ont aidée de près ou de loin Je tiens à vous exprimer ma gratitude, mon amour et mon respect.

Merci pour votre bienveillance, merci pour votre patience, merci pour vos encouragements et pour vos efforts.

Merci pour tout.

T.Malika

Résumé

La digitalisation exponentielle de toutes les activités humaine avec la croissance des exigences et des besoins des clients, a mené pour développer un nouveau style architectural pour les applications logicielle entreprise, connu par le nom « microservices », en résoulant ainsi les difficultés rencontrés lors du traitement des grands projets en terme de temps et du coût. Ce travail concerne la migration d'une application monolithique adopté dans l'Entreprise Portuaire d'Alger (EPAL) vers une nouvelle approche « microservices », pour réduire le coût du développement et de la maintenance, et améliorer la qualité de service de cette application qui sera capable, à la fin, de répondre aux exigences des clients en temps réel.

Avant ce faire, nous avons présenté en premier lieu l'approche monolithique avec son impact sur le marché. Pour passer ensuite à effectuer une étude sur l'architecture microservices, où nous avons monté ces différents concepts et caractéristiques. Nous avons tiré les avantages de son utilisation avec l'approche DevOps et la méthodologie agile, en expliquant l'utilité de chacune de ces technologies. Nous avons entamé la section étude de l'existant et conception, après avoir présenté l'organisme d'accueil de l'entreprise, pour étudier la faisabilité de cette migration, dont nous avons montré les différents diagrammes qui vont nous aider dans la réalisation de cette application. Terminons par les différents outils à utiliser dans le développement et le déploiement de notre application.

Mots clés : Monolithique, Microservices, DevOps, EPAL, Docker, Kubernetes.

Abstract

The exponential digitization of all human activities with the growth of customer requirements and needs, has led to develop a new architectural style for enterprise software applications, known by the name "microservices", thus solving the difficulties encountered during processing. big projects in terms of time and cost.

This work concerns the migration of a monolithic application adopted in the Algiers Port Company (EPAL) towards a new "microservices" approach, to reduce the cost of development and maintenance, and improve the quality of service of this application which will be able, at the end, to meet customer requirements in real time.

Before doing this, we first presented the monolithic approach with its impact on the market. Then we move on to doing a study on the microservices architecture, where we built these different concepts and characteristics. We have reaped the benefits of using it with the DevOps approach and the agile methodology, explaining the usefulness of each of these technologies. We started the study of the existing and design section, after presenting the host organization of the company, to study the feasibility of this migration, of which we have shown the different diagrams that will help us in the realization of this application. Let’s finish with the different tools to use in the development and deployment of our application.

Keywords : Monolithic, Microservices, DevOps, EPAL, Docker, Kubernetes.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

1. Contexte du travail

Les technologies logicielles ne cessent d'évoluer pour faciliter le développement, le déploiement et la maintenance d'applications dans différents domaines. En parallèle, ces applications évoluent en continu, avec la croissance potentielle des exigences et des besoins des clients, pour garantir une bonne qualité de service qui deviennent de plus en plus complexe. Cette évolution implique souvent des coûts de développement et de maintenance plus élevés, auxquels peut s'ajouter une augmentation des coûts de déploiement sur des infrastructures d'exécution récentes telle que le Cloud. Réduire ces coûts et améliorer la qualité de ces applications sont actuellement des objectifs centraux du domaine du génie logiciel. Apparus il y a quelques années, une tendance à l'accélération, sous le nom des microservices au tant qu'exemple de technologie ou styles architectural favorisant l'atteinte de ces objectifs. En effet, l'approche des microservices a été inventé pour résoudre certaines difficultés causées par les gros projets, en offrant des avantages tangibles (concret ou encore sûr), notamment une augmentation de l'évolutivité, de la flexibilité et l'agilité. En remplaçant ainsi l'approche par défaut adopté dans le développement des applications, connue sous le nom « approche monolithique ». Le microservice alors est une méthode de développement d'applications logicielles en tant que suite de services modulables indépendant, dans lesquels chaque service exécute un processus qu'il communique à travers un mécanisme défini au préalable. Dans le domaine de la programmation par exemple, on forme des petites équipes qui ne s'occupent que d'un seul service. Et puis, dans le domaine du management du projet, on se concentre sur l'équipe et son indépendance, c'est-à-dire au lieu d'avoir une administration centralisée, chaque équipe est entièrement responsable de son produit final. De ce fait, de nombreuse entreprise considèrent que suivre cet exemple est le moyen le plus efficace de développer leurs activités, en basculant leurs processus de développement vers cette nouvelle architecture. Le cas d'Entreprise Portuaire d'Alger (EPAL).

2. Problématique

EPAL dispose d'une application monolithique avec une administration centralisé, c'est-à-dire pour faire communiquer les entités (modules) entre elle, il va falloir passer par un intermédiaire ou un administrateur qui va transmettre à son tour le message aux autres entités, ce problème engendre la duplication des bases de données. En d'autres termes, l'administration centralisée de ce processus nécessite la disposition de la base de données dans chaque entité participante, ce qui cause à son tour des problèmes en termes de temps d'exécution, le cout et la maintenance. Dans le cadre de ce travail, nous essayons de migrer de l'architecture monolithique adoptée par cette entreprise vers une nouvelle architecture microservices, en divisant 'le code' en plusieurs différents petits services, pour qu'ils soient traités indépendamment l'un des autres.

3. Contribution

Cette migration se base sur l'approche DevOps qui permet de collaborer et réunir les équipes de développement. Ainsi que la méthodologie « SCRUM », utilisé dans la gestion des projets, pour garantir un équilibre entre ces équipes au fur-et-à mesure de la progression du projet. En définitive, la contribution du présent travail est de donner des solutions à des problèmes liés aux :

- Temps d'exécution de l'application.
- Le coût et le déploiement de l'application.
- La maintenance et la mise à jour des modules.
- Etablissement d'un plan d'action

4. Organisation du mémoire

Notre mémoire est organisé en deux parties englobant en tout quartes chapitres. La première partie comporte les deux premiers chapitres.

Le premier chapitre présente les notions de base de notre étude, à savoir l'approche monolithique, microserices et DevOps, avec la présentation de notre organisme d'accueil. Le deuxième chapitre consacré pour l'étude de l'existant, dont nous avons représenté l'infrastructure de l'entreprise ainsi que la description de ces applications.

La deuxième partie comporte les deux derniers chapitres dédies au développement et au déploiement de notre application. Le troisieme chapitre présente les différents diagrammes UML utilisés dans la réalisation du travail, à savoir les diagrammes de cas d'utilisation, les diagrammes de séquence et le diagramme de classe. Le quatrième chapitre est dédié au développement de l'application. Nous commençons par la description de l'environnement du développement avec les différentes technologies utilisées. A la fin, nous présentons les captures de l'application réalisée avec son déploiement.

ChapitreI. Partie théorique et présentation de l’organisme

Partie 1 :

I.1 Introduction :

Les entreprises reposent sur les systèmes d'informations qui représentent l'ensemble des éléments participant à la gestion, au traitement, au transport et à la diffusion de l'information au sein de l'entreprise. Les systèmes d'informations sont de plus en plus complexes avec la démultiplication des applications, et les data.

Afin de faire tourner le système d'information dans une entreprise cette dernière a besoin d'implémenter un type d'architecture logiciel.

En effet il existe trois types d'architecture à savoir :

L'architecture monolithique qui est apparue vers les années 1990 elle est basée sur la construction de l'application en un seul bloc.

Avec le temps les entreprises qui adaptent ce type d'architecture ont rencontré des problèmes tels que la difficulté de mises à jour, fiabilité...etc. Ce qui à entrainer l'apparaissions d'un autre types d'architecture vers 2005, appeler l'architecture orienté service(SOA) qui se base sur la décomposition de l'application en quelque services chacun est responsable de fonctionnement d'un ensemble de fonctionnalités elle repose sur le protocole SOAP malgré cette décomposition en services mais les données étaient toujours centralisées dans un SGBD. la SOA n'a pas rencontrer le succès escompté : son usage s'est souvent limité aux gros projets dans les grandes entreprises..

Quelques ans plus tard et plus précisai ment en 2015 un autre types d'architectures est apparu appeler l'architecture en microservices qui est une amélioration de la SOA. Dans les microservices une application est conçu comme une suite de petits services, appelés micro-services, découplés le plus possible les uns des autres, dont chaque microservices répond à une fonctionnalité métier unique

Dans ce premier chapitre, nous nous intéressons aux concepts de base liés à notre travail. Il se compose de deux parties

Dans la première partie, Nous définissants l'application monolithique, ses avantages et ses limites, ensuite, nous décriront l'architecture microservices, ses avantages et inconvénients. Puis nous projetterons les concepts liés à ce style d'architecture à savoir l'approche de la

conception pilotée par le domaine (Domain Driven Design DDD) et nous présenterons l'approche de développement DevOPS (developpeurs et opérationels).

Enfin, nous nous intéresserons au cadre générale de notre projet qui s'agit d'une présentation générale de l'organisme d'accueil.

I.2 Application Monolithique :

I.2.1 Définition :

Une architecture monolithique représente le modèle traditionnel unifié de conception d'un programme informatique.

C'est une application développée en un seul bloc avec une même technologie et déployée dans un serveur d'application [1]. Une application monolithique est souvent associée à une base de données et à une interface utilisateur côté client. Ses composants sont interconnectés et interdépendants plutôt qu'associés de manière flexible comme dans le cas des programmes modulaires. Dans ce type d'architecture, chaque composant et ceux qui lui sont associés doivent être présents pour permettre l'exécution ou la compilation du code.

I.2.2 Avantage de l'architecture monolithique :

- Démarrer un nouveau projet et le développer est plus simple grâce à une architecture monolithique
- Il est beaucoup plus facile de tester une structure qui est un monolithe Des composants tels que des Framework, des modèles ou des scripts peuvent être facilement appliqués
- Le déploiement est très simple. Tout ce que vous avez à faire est de coller l'application préalablement préparée sur le serveur.

I.2.3 Limites de l'architecture monolithique :

L'architecture monolithique permet de développer des applications entières en un seul bloc. Cette architecture a fonctionné avec succès pendant de nombreuses décennies. Ce qui fait que de nombreuses entreprises sont toujours en cours de l'utiliser.

Mais avec l'évolution du marché et émergence de nouvelles technologies, les applications monolithiques ont eu de sérieux problèmes. Le tableau ci-dessous récapitule les principales limites de cette architecture et leurs impacts sur la plateforme.

Limites	Impact sur la plateforme
Limites technologiques	▪ Le choix des technologies est fixé avant que le développement de l'application commence. ▪ Une seule technologie est utilisée pour le développement de l'application.
Mise à l'échelle couteuse	▪ La mise à l'échelle d'une partie qui a besoin de plus de ressources requiert toute l'application.
Cout des tests élevé	▪ Durée des tests automatiques longue et durée de build longue.
Modification couteuse	▪ Une modification dans une petite partie de l'application requiert un rebuild et redéploiement entier
Tolérance aux pannes limitée	▪ Un dysfonctionnement sur une partie du système impacte tout le système.
Agilité limitée	▪ Agilité réduite de l'équipe et fréquence de livraison limitée à cause du couplage fort entre les composants. ▪ Effort important de montée en compétences pour un nouvel arrivant sur des composants fortement couplés. ▪ Nécessité de l'intervention de plusieurs personnes pour chaque modification.

Tableau 1: les impacts de l'architecture monolithique.

Pour remédier à ces limites une nouvelle architecture d'application dite Microservices a vu le jour.

I.3 Architecture Microservices :

Cette architecture n'a pas une définition unique donc nous donnons deux définitions différentes :

I.3.1 Définition 1 :

L'architecture microservices est une méthode de développement d'applications logicielles en tant que suite de services modulables et indépendamment [2], l'idée étant de découper un grand problème en petites unités implémentées sous forme de microservices, ou chaque microservice est responsable d'une fonctionnalité et il est développé, testé et déployé indépendamment des autres. Chaque microservice peut être développé en utilisant une technologie différente des autres et chaque microservice tourne dans un processus séparé.

La seule relation entre les différents microservices est l'échange de données effectué à travers les différentes API qu'ils exposent.

I.3.2 Définition 2 :

Le terme microservice a connu une émergence au cours de ces dernières années pour décrire un style d'architecture bien particulier. Cette approche se concentre sur la décomposition d'une application en petits services autonomes et déployés indépendamment. Les services communiquent entre eux à travers réseau grâce aux API [3]. Les microservices sont implémentés et exploités de manière indépendante.

I.3.3 Domain Driven Design (DDD) :

Domain Driven Design n'est pas un Framework ni une méthodologie mais plutôt d'après Eric Evans [4] c'est une approche qui favorise la création des systèmes informatiques autour des compétences métiers pour combler l'écart entre la réalité de l'entreprise et le code.

DDD vise à augmenter les taux de réussite en comblant le fossé de collaboration et de communication.

Pour permettre le partage fluide des connaissances, DDD appelle à l'utilisation d'un langage partagé, orienté domaine métier : le langage ubiquitaire (omniprésent).

Le langage omniprésent est le point de départ du DDD. Le principe de cet ubiquitous language est que chaque classe, méthode, variable etc. doit être nommée avec le plus grand soin afin que le code raconte au travers de ses objets l'histoire du métier, que le code retranscrive au plus près les réalités métier. L'objectif du recours à un langage omniprésent est que tout le monde parle le langage métier partout même dans le code, afin de s'assurer que le code n'est pas pollué par la technique et qu'il raconte le métier.

Comme le montre la figure ci-dessous [4] :

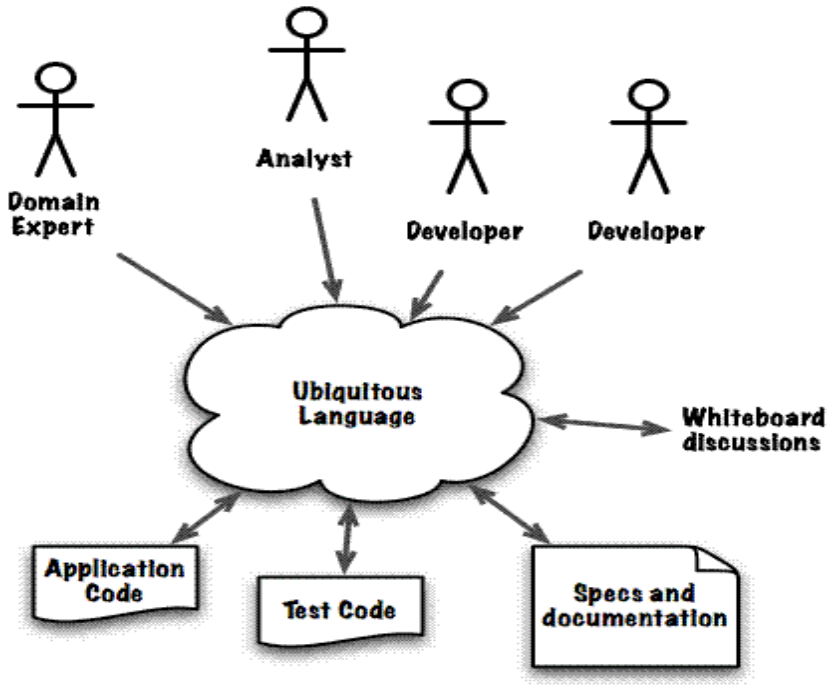


Figure 1:langage utilisé entre les équipes.

Le langage omniprésent devrait être le seul langage utilisé pour exprimer un modèle. Tout le monde dans l'équipe devrait pouvoir s'entendre sur chaque terme spécifique sans ambiguïté et aucune traduction ne devrait être nécessaire.

Le maintien d'un modèle dans un état pur est difficile quand il s'étend sur l'intégralité de l'entreprise. Donc il est préférable de tracer des limites à travers le pattern « Bounded Contexte» pour les définir.

I.3.4 Bounded contexte (contexte borné) :

D'après Martin Fowler [5] le contexte borné est un pattern central dans la DDD. Il est au centre de la section de conception stratégique. Bounded contexte traite de grands modèles en les divisant en différents contextes bornés. Ces relations entre les contextes bornés sont généralement décrites par une carte de contexte**.**

C'est avec le bounded contexte qu'on va décomposer notre application monolithique en différents microservices.

I.3.5 Carte de contexte :

Il est nécessaire de définir des frontières et des relations entre les modèles multiples. La carte de contexte est un document partagé entre ceux qui travaillent sur le projet [6]. Elle met en évidence les différents contextes bornés et leurs liaisons. Cette carte peut être un diagramme comme illustré dans la figure [7], ou n'importe quel document écrit dans des niveaux de détails variables.

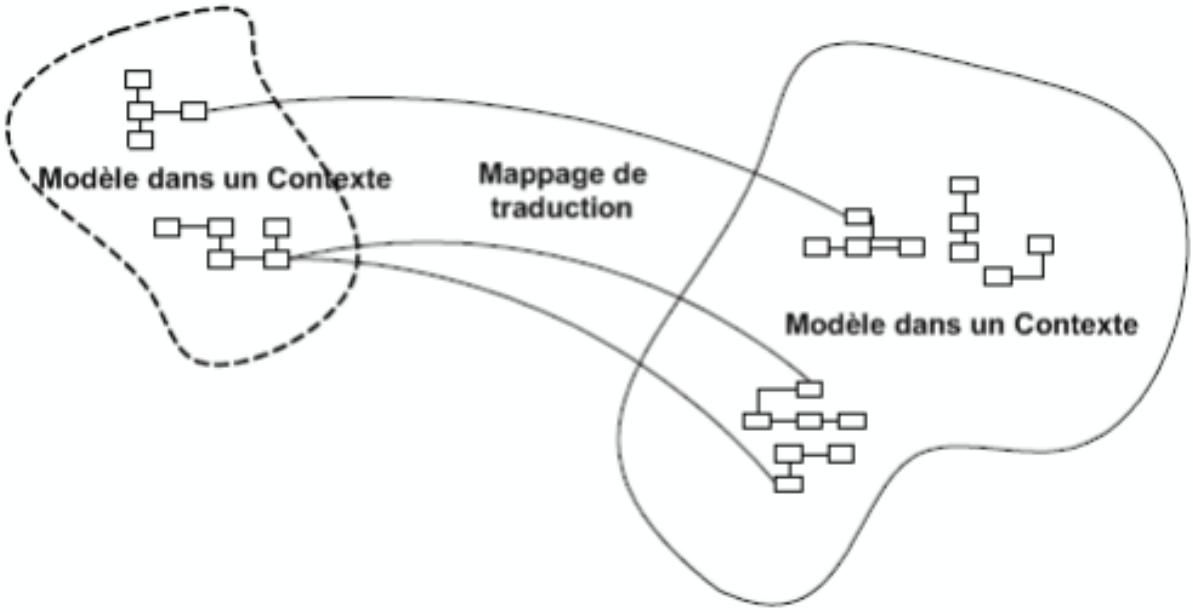


Figure 2:La carte de contexte.

I.3.6 Développement à base de composants :

Le développement à base de composants est une branche de l'ingénierie logicielle qui met l'accent sur la séparation des préoccupations à l'égard des vastes fonctionnalités disponibles à travers un système logiciel. [8]

Cependant, un système est un ensemble de préoccupations fonctionnelles et extra-fonctionnelles (aspects). Les préoccupations fonctionnelles sont les fonctionnalités métiers que le système doit assurer, alors que les préoccupations extra-fonctionnelles sont des services dont le système a besoin pour effectuer ses fonctionnalités métiers.

Comme exemple de préoccupations extra-fonctionnelles, on peut citer la sécurité, la synchronisation, etc. Dans les approches de développement de logiciels, notamment dans le cas de la programmation orientée objet, il apparaît que les deux types de préoccupations sont enchevêtrés et le code relatif aux préoccupations extra-fonctionnelles est éparpillé dans le code fonctionnel

Cette situation augmente la complexité, empêche la réutilisation et gêne l'évolution des systèmes. La séparation avancée des préoccupations permet de séparer les parties extra-fonctionnelles des parties fonctionnelles d'un système. L'objectif escompté étant d'offrir une meilleure réutilisation, faciliter la maintenance, réduire la complexité des systèmes et augmenter leur évolutivité.

I.3.7 Persistance polyglotte :

En 2006 le terme de programmation polyglotte a été inventé par Neal Ford pour exprimer l'idée que les applications devaient être écrites dans un mélange de langues pour tirer parti du fait que différentes langues conviennent pour résoudre différents problèmes. [9]

En 2008, Scott Leberknight inventa le terme de persistance polyglotte pour exprimer l'idée que les applications complexes devront naturellement utiliser plusieurs types de base de données pour répondre à chaque problème efficacement.

I.4 Caractéristiques des microservices :

➢ La division en composants via les services :

Cette caractéristique est héritée du génie logiciel à base de composants. Les microservices sont indépendamment développés, testés et déployés. Un changement dans un service ne nécessite que son déploiement et n'affecte pas l'intégralité du système. Les services permettent d'éviter le couplage fort entre les différents composants en utilisant des mécanismes d'appel distants explicites.

➢ L'organisation autours des capacités métiers :

La décomposition classique d'une application est souvent orientée vers les couches techniques. Le fractionnement est autour des capacités métier de l'entreprise. Chaque service est autonome vis-à-vis de la fonctionnalité qu'il réalise. Il possède son propre code, sa propre interface et gère ses propres donnés

➢ Un Produit, pas un Projet :

Le but d'utiliser les microservices est de livrer rapidement un morceau de logiciel qui est alors considéré comme terminé. Dans la vision microservices, une équipe est responsable d'un produit durant tout son cycle de vie. L'équipe de développement est entièrement responsable du logiciel en production.

➢ Les extrémités Intelligentes et les Canaux Stupides [10] :

Plusieurs entreprises s'investissent dans les canaux de communication intelligente entre les services, alors qu'avec les micros services, l'utilisation de communications stupides est favorisée. Ces communications non intelligentes ne font que transmettre les messages alors que le microservice s'en charge du reste. L'intercommunication entre les microservices via des protocoles ouverts est privilégiée et beaucoup d'autres interagissent les uns avec les autres via des API REST ou à travers des systèmes de file d'attente

➢ Une Gouvernance Décentralisée :

Il est difficile de trouver une seule technologie qui résout tous les problèmes d'une façon efficace. Donc, il est préférable d'utiliser le bon outil au bon moment. Dans les architectures microservices, chaque service pourra utiliser la technologie, le langage et la plateforme les plus adéquats pour ses besoins.

- **Gestion de Données Décentralisée :**

L'architecture micro service admet l'utilisation de plusieurs bases de données dans le cadre d'une application monolithique nous n'avons qu'une seule base de données logique pour les entités persistantes alors qu'une application en microservices, a son propre modèle conceptuel et gère ses propres bases de données.

- **Automatisation de l'Infrastructure :**

Les techniques d'automatisation de l'infrastructure ont connu une évolution considérable ces dernières années. L'évolution du Cloud a réduit la complexité opérationnelle de la construction, du déploiement et de l'exploitation de microservices. Les entreprises, qui ont migré vers l'architecture microservices, a gagné de l'expérience dans la livraison continue et l'intégration continue on utilise maintenant des outils d'automatisation de l'infrastructure.

- **Tolérance aux pannes :**

L'un des atouts majeurs des microservices est leurs conceptions pour être tolérants aux pannes. Dans une application en microservices, si un service échoue, les autres services ne sont pas affectés et adaptent leurs fonctionnements selon l'état du système dans lequel ils évoluent.

- **Une conception évolutive :**

La propriété clé d'un microservice est les notions d'indépendance et d'évolutivité, ce qui implique que nous pouvons réécrire un microservice sans affecter les autres. En général, l'évolution d'une application consiste en l'ajout de nouvelles fonctionnalités qui se traduit par la création de nouveaux microservices et ou par la mise à jour des services existants qui implique seulement la mise à jour et le redéploiement du microservice concerné.

I.4.1 Avantages et Inconvénients :

- **Avantages [11] :**

  - ✓ L'architecture microservices offre aux développeurs la liberté de développer et de déployer de manière indépendante
  - ✓ Un microservice peut être développé par une équipe assez petite
  - ✓ Intégration facile
  - ✓ Possibilité d'utiliser différentes technologies
  - ✓ Facilité des tests et du déploiement
  - ✓ Livraison contenue

- **Inconvénients [12] :**

  - ✓ En raison du déploiement distribué, dans certain cas les tests peuvent devenir compliqués et fastidieux
  - ✓ L'augmentation du nombre de services peut entraîner des barrières d'information
  - ✓ Les développeurs doivent mettre des efforts supplémentaires dans la mise en œuvre du mécanisme de communication entre les services
  - ✓ L'architecture entraîne généralement une consommation de mémoire accrue.

Les microservices et DevOps sont deux tendances importantes qui sont devenues de plus en plus précieuses pour les entreprises. Ces pratiques sont conçues pour offrir une meilleure agilité et efficacité à l'entreprise. Par conséquent, DevOps est un facteur clé de l'excellence des microservices.

I.5 Définition du DevOps :

DevOps est une approche qui permet de réunir et de faire collaborer les équipes de développement et d'exploitation afin de pouvoir créer et livrer des produits plus performants et plus fiables pour mieux répondre aux besoins du client. Cette collaboration et cette productivité améliorées permettent d'atteindre des objectifs commerciaux tels que la publication plus fréquente ou la mise à disposition plus rapide de versions, de fonctionnalités ou de mises à jour des produits, le tout en assurant les niveaux de qualité et de sécurité appropriés. Autre objectif : améliorer les délais de détection, de résolution de bogues ou d'autres problèmes et de republication d'une version, et aussi

- ✓ Raccourcir le délai de commercialisation
- ✓ S'adapter au marché et à la concurrence
- ✓ Améliorer le temps de récupération

. DevOps encourage la possibilité maximale d'automatisation en utilisant des outils et des scripts DevOps. L'une des plateformes les plus utilisées dans l'intégration et le déploiement continue CI/CD est Jenkins, un outil open source (qui peut cependant être difficile à prendre en main)

Il existe aussi des solutions payantes comme GitlabCI (que nous utilisons chez Padok), Bamboo, TeamCity, Concourse, CircleCI ou Travis CI.

La figure suivante montre collaboration des développeurs et opérations

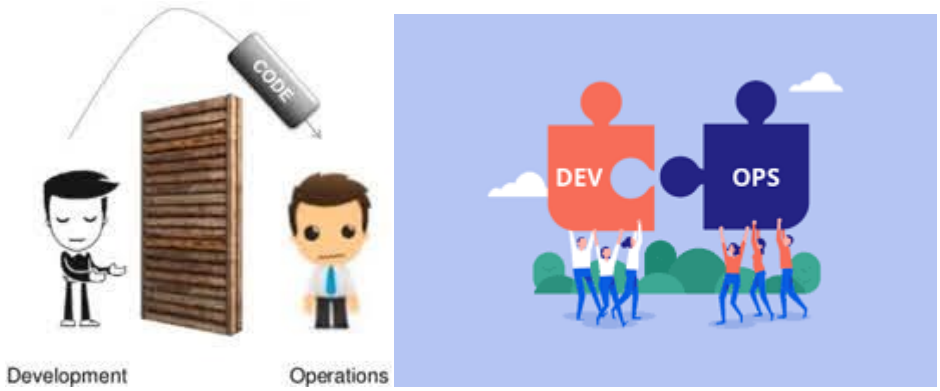


Figure 3: Approche DevOps.

I.5.1 Cycle de vie DevOps :

Le cycle de vie se compose de sept étapes comme le montre la figure ci-dessus [13] :

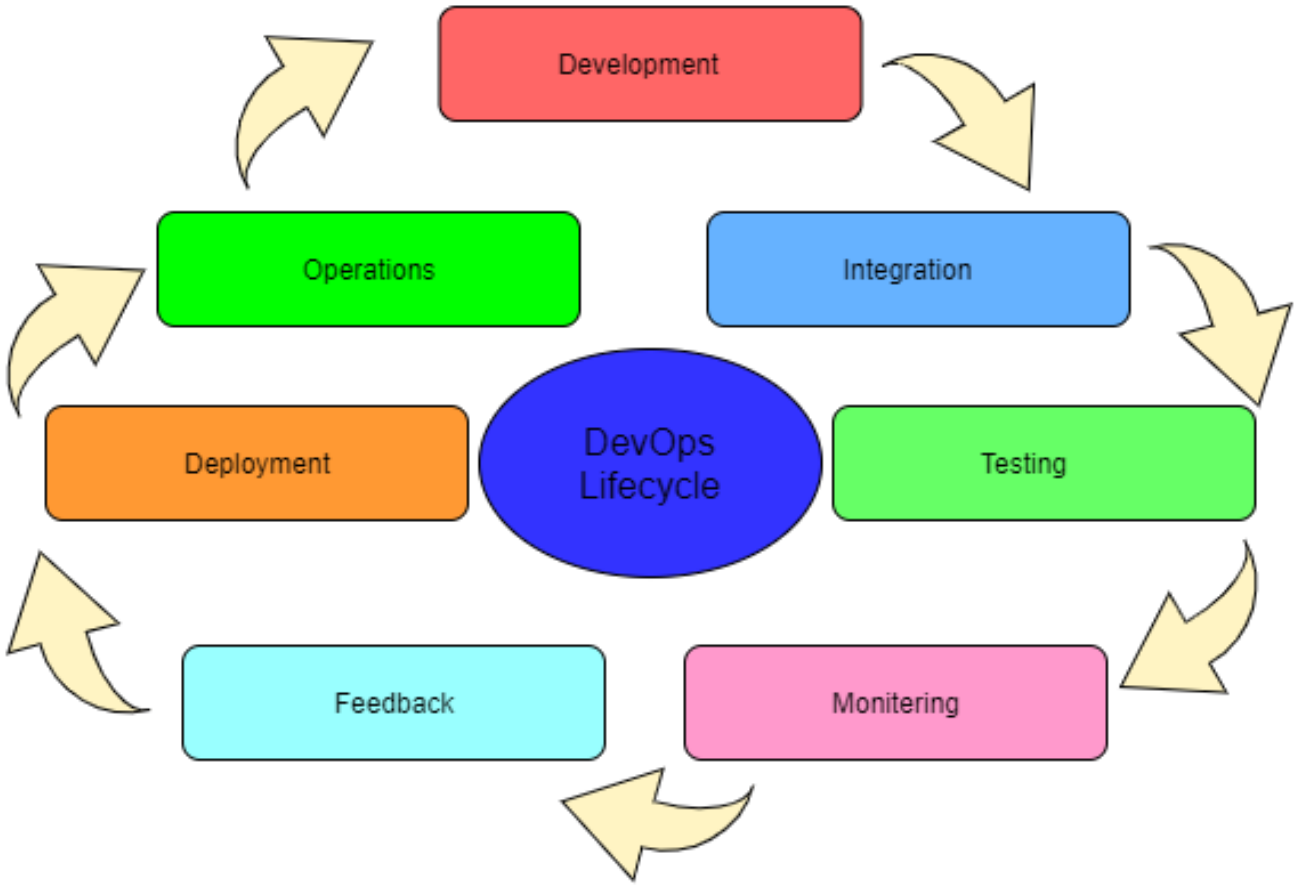


Figure 4:cycle de vie de l'approche DevOps.

- **Le développement continu (continuous development) :**

Le développement continu est la première étape de cycle de vie de l'approche DevOps. Cette étape veut dire qu'au lieu de faire les mises à jour pour le logiciel en un seul lot, les mises à jour seront effectuées en continu, bloc par bloc, permettant au code logiciel d'être livré aux clients dès qu'il est terminé et testé.

➢ L'intégration continue (continuous integration) :

Le processus d'intégration continue démarre automatiquement après le développement Il comprend plusieurs étapes comme

- Planification de développement.
- Compilation et intégration du code.
- Test du code.
- Mesure de la qualité de code.
- Gestion des livrables de l'application

➢ Les tests continus (continuous testing) :

Les tests continus dans DevOps signifient des tests sans interruption effectués sur le logiciel à chaque étape du cycle de vie du développement logiciel. L'objectif des tests continus est d'évaluer la qualité des logiciels à chaque étape du processus de livraison continue. Le code est continuellement développé, livré, testé et déployé.

➢ La surveillance continue (continuous monitoring) :

La surveillance continue est une partie essentielle de DevOps. Elle consiste à surveiller les logiciels (ressources) et les processus de développement (méthodes). Les ressources sont mesurées en continu dans tous les environnements pour détecter les erreurs le plus tôt possible.

➢ Rétraction continue (continuous feedback) :

Improvise le produit actuel et aide à publier de nouvelles versions rapidement

C'est une phase importante de l'application logicielle ou le retour client est un atout majeur pour améliorer le fonctionnement du logiciel actuel et publier rapidement de nouvelles versions en fonction de la réponse.

➢ Déploiement continue (continuous deployment) :

Processus de déploiement est effectué de telle manière que les modifications apportées au code ne devraient pas affecter le fonctionnement de l'application à fort trafic.

- **Opération continue (continuous operations) :**

Automatisez le processus de publication avec des cycles de développement plus court. Tous les opérations de DevOps sont basées sur la continuité avec une automatisation complète du processus de lancement et permettent aux organisations d'accélérer le temps global de mise sur le marché de manière continue.

I.6 La relation entre microservices et DevOps :

L'architecture micro services et l'approche DevOps présentent des pratiques qui offrent une plus grande agilité et une efficacité opérationnelle pour l'entreprise, Elles rendent l'entreprise plus productive que leurs concurrents, ce qui signifie qu'elles peuvent innover plus rapidement à moindre coût...

La combinaison entre microservices et DevOps a été utilisée pour la première fois par les grandes entreprises telles qu'Amazon, Netflix, Google ...etc.

Cette combinaison permet d'accélérer l'innovation, réduire les erreurs, améliorer la qualité des produits, augmenter la valeur commerciale, réduisez les coûts de développement logiciel, et rendre les équipes DevOps plus productives.

Un mélange des deux méthodologies tel qu'agile et DevOps peut être utilisé pour assurer une efficacité accrue. Les deux ont un rôle majeur à jouer en matière de développement et de déploiement de logiciels, et l'un peut être utilisé pour activer l'autre.

I.7 Définition de la méthodologie agile :

Agile est une approche itérative qui encourage la collaboration, l'organisation et la rétroaction entre les clients et les équipes de développements afin de mener à bien des projets plus rapidement et plus efficacement [14].

Parmi les méthodes agiles les plus utilisées on trouve la méthodologie SCRUM [15].

I.7.1 La méthodologie SCRUM :

Le nom SCRUM est un terme emprunté au rugby qui signifie « la mêlée ». Le principe de base était que l'équipe avance ensemble et soit toujours prête à réorienter le projet au fur-et-à-mesure de sa progression.

SCRUM est Considéré comme un cadre (Framework en anglais) de gestion de projet, elle se compose de plusieurs éléments fondamentaux :

- ✓ Des rôles,
- ✓ Des événements,
- ✓ Des artefacts,
- ✓ Des règles.

I.8 Relation entre agile et DevOps :

Agile et DevOps se concentrent tous deux sur la vitesse, l'efficacité et la qualité des résultats tout au long du cycle de vie du développement logiciel. Ils se concentrent également sur des cycles de publication plus courts[16]. Les deux méthodologies ne mettant pas beaucoup l'accent sur les niveaux de documentation et passent plus de temps sur l'automatisation et la collaboration. Au fur et à mesure de l'avancement des projets, le niveau de risque a tendance à diminuer lors de l'utilisation d'une approche Agile et/ou DevOps, alors que le risque a tendance à augmenter avec le temps avec d'autres approches tel que **Kanban**.

La figure ci-dessous représente la collaboration entre client et développeur et les administrateur systèmes :

Figure 5:Approche DevOps et approche Agile

Partie 2 :

Dans cette partie, nous présenter l'organisme d'accueil considéré comme étant un environnement ou encore l'apprenti de mise en œuvre du savoir et savoir-faire et surtout l'occasion de m émerger dans le domaine professionnel

Dans le cadre de notre projet nous avons opté pour l'organisme d « EPAL » qui sera présenté en détail dans cette partie

I.9 Présentation de l'organisme d'accueil :

Entreprise Portuaire d'Alger (EPAL) a été créée par décret N°82286 en aout 1982 à la suite d'une négociation du secteur portuaire. Devenue autonome en septembre 1989 et ayant le statut d'une SPA au capital social de 750 millions de dinars dépendant du *Holding Public* « services » dite société de gestion des ports et relevant du ministère des transports. Le conseil d'administration est l'organe décideur qui élit le directeur général comme il peut le révoquer.

I.9.1 Missions du port d'Alger :

L'EPAL est une Entreprise Publique Economique (EPE), elle a pour missions :

- **Mission d'autorité portuaire :**

Concerne la gestion du domaine publique portuaire

- En matière de police et de sécurité l'EPAL veille à la protection des installations du domaine public portuaire.
- Règlement et contrôle de la navigation dans le port, de même que l'affectation des quais d'accostage aux navires devant charger ou décharger les marchandises.
- Appliquer et faire respecter la réglementation en vigueur en matière d'exploitation et de développement des zones industrielles portuaires.

- **Mission commerciale :**

Concerne les prestations commerciales pour les navires (remorquage, ravitaillement en eau…), rentrant au port ainsi que les prestations touchant les marchandises en transit (chargement. Déchargement, entreposage, conservation et livraison aux propriétaires), les prestations de manutention et de relevage.

I.9.2 Présentation des différentes structures et leurs missions :

L'entreprise portuaire est structurée en trois directions à savoir :

La direction générale, la direction opérationnelle et la direction fonctionnelles qui s'adaptent à la densité et la diversité des activités portuaires, Des changements sont à prévoir dans les jours à venir suivant le nouveau mode économique national qui exige un changement d'organisation afin de répondre à l'accroissement du trafic maritime.

La figure suivante represente l'organigramme de l'entreprise EPAL :

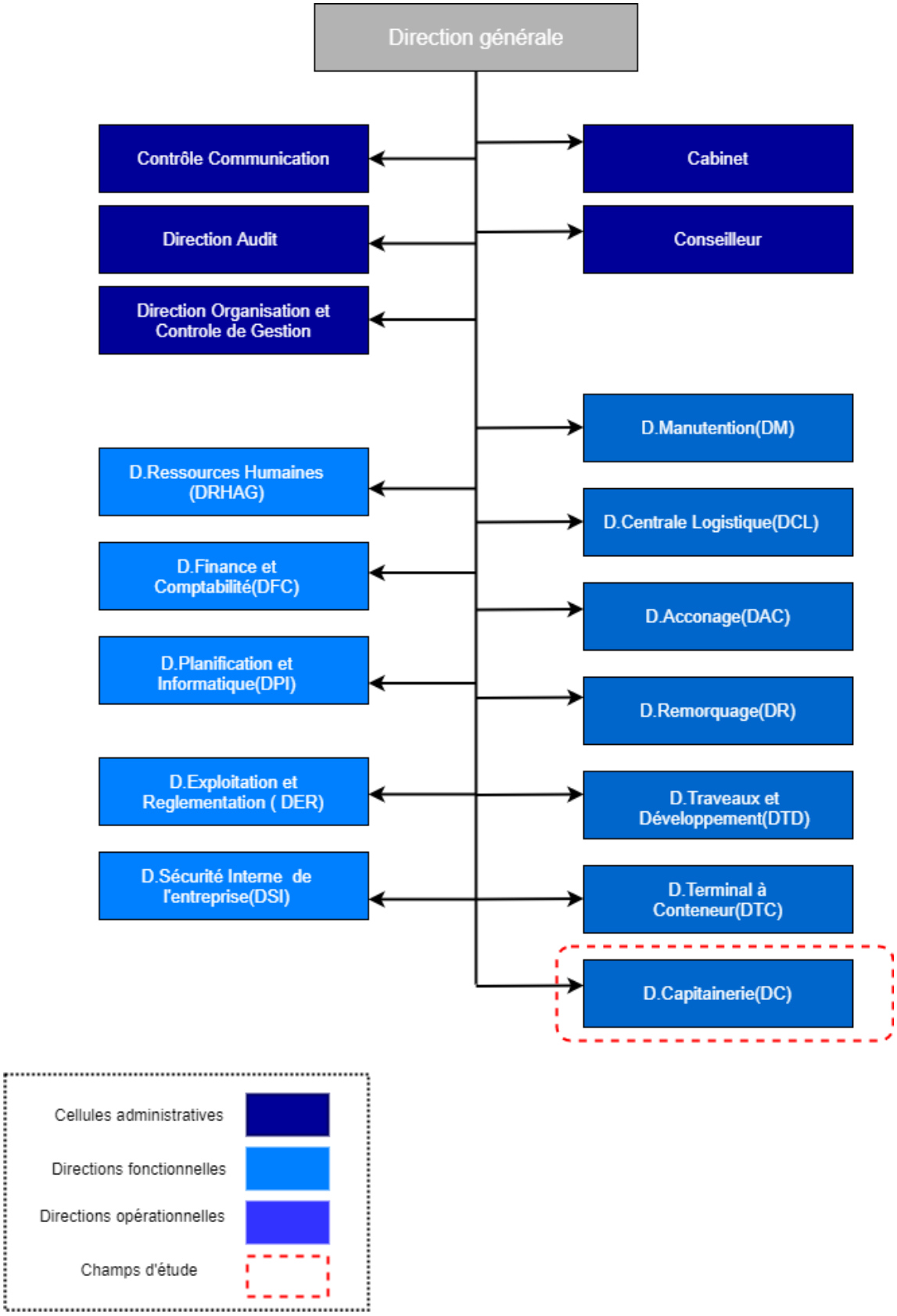


Figure 6:organigramme d'EPAL

➢ Directions fonctionnelles :

Elles assurent : l'animation, la coordination et la consolidation de la structure opérationnelle.

Elle organise la circulation, l'analyse et la synthèse des informations. Ces directions sont :

- **Direction Ressources Humaines et d'Administration Générale(DRHAG) :** Elle est chargée de :
  - La gestion des ressources humaines et l'administration des personnels.
  - La conception des règles et procédures liées à la gestion des ressources humaines.
- **Direction Finance et Comptabilité (DFC) :** Elle est chargée de :
  - Définir de mettre en œuvre la politique de gestion financière et de procédures en comptables de l'entreprise.
  - Procéder à des analyses financières périodiques.
  - Veiller à l'équilibre financier de l'entreprise.
- **Direction Planification et Informatique(DPI) :**

  C'est une direction fournissant des prestations de logiciels et matériels informatiques et procédures utilisées par les autres directions. Parmi les activités principales de la DPI :
  - L'élaboration des procédures utilisées dans l'entreprise, leur développement et la mise en place des structures de l'entreprise.
  - Adoption d'une architecture réseau et une stratégie de sécurité de données spéciales.
  - L'organisation du système, des circuits d'informations et de communication entre environ700 postes de travail dans de l'Entreprise.
  - Planification des projets de conception.
  - Réalisation et suivi des études de domaine.
  - Réalisation et suivi des études conceptuelles.
  - Implémentation des applications de gestion au niveau des directions concernées.
  - Maintenance du matériel informatique utilisé dans l'EPAL.

- Réalisation des extensions réseaux à la demande.

- **Direction Exploitation et Réglementation (DER):**

Elle est chargée de la gestion du domaine public portuaire, de l'application des textes législatifs et de l'actualisation de ceux jugés indiquant que la promotion du port d'Alger en tant que première place portuaire du pays.

- **Direction Sécurité Interne de l'entreprise(DSI) :**

  - La principale mission de cette direction est d'assurer et de veiller à la sécurité des biens et des personnes ainsi que des marchandises transitant par le port.
  - Le port d'Alger est classé parmi les premiers ports certifiés « code ISPS » en Afrique assuré par (03) trois brigades reparties sur trois zones dans le port, chaque brigade comprend :
    - Des maîtres-chiens.
    - Des gardiens portuaires de sécurité.
    - Des agents de sécurité en patrouilles mobiles.

➢ Directions opérationnelles :

Elles ont un rôle de gestion d'exploitation et de mise en œuvre des moyens matériels et humains. Elles fonctionnent suivant le principe d'autonomie de gestion par la réalisation d'un budget propre. Ces Directions sont :

- **Direction Manutention (DM) : Elle est chargée de :**

  - Opérations de manutention liées au chargement et à l'arrivage des marchandises à bord des navires, à l'import et à l'export.
  - La programmation des moyens humains et matériels nécessaires au bon déroulement des opérations de manutention.
  - La facturation des prestations fournies aux navires.

- **Direction Centrale Logistique(DCL)** : **Elle est chargée de :**

La gestion des engins de manutention portuaire, de leur affectation en fonction des besoins exprimés par les directions manutention et acconage, et de leur location aux autres usagers.

- **Direction Remorquage (DR)** : **Elle est chargée de :**

  - L'ensemble des opérations liées aux remorquages des navires (remorquages portuaires et de haute mer), conformément à la législation et à la réglementation en vigueur.
  - Sauvetage et de l'assistance des navires.

- **Direction Acconage (DAC) : Elle est chargée de :**

  - La réception et de la reconnaissance physique des marchandises, contradictoirement avec le bord au débarquement et les chargeurs ou leur représentant à l'embarquement.
  - L'entreposage et le gardiennage des marchandises débarquées.
  - La préservation des marchandises sur les aires d'entreposage.
  - La gestion et le suivi des affaires juridiques et contentieuses liées à l'activité.
  - Rentabiliser les aires d'entreposages (terre-pleins et magasin).
  - L'élaboration et la transmission des données statistiques d'acconage.

- **Direction Travaux et Développement (DTD):**

Cette direction est chargée de l'organisation, la coordination et le contrôle de l'ensemble des travaux d'entretien et de maintenance des installations, bâtiments, ouvrages et équipements du port.

- **Direction des Conteneurs(DTC) :**

L'évolution du trafic maritime de la marchandise conteneurisée, qui a atteint un volume important, a été pris en charge par la direction des conteneurs, pour ce faire, elle dispose d'équipes spécialisées dans le traitement des conteneurs à même de réaliser des rendements très performants, à cet effet elle est chargée de :

- La gestion et l'exploitation du terminal à conteneurs.
- Opérations de transport des conteneurs du quai jusqu'aux parcs de et inversement.
- La réception et le pointage des conteneurs.

- **Direction Capitainerie(DC) :**

La capitainerie représente le centre opérationnel du port, cette direction gère et contrôle les mouvements de la navigation. Elle est chargée de :

- La prise en charge des navires dès leur annonce d'arrivée.
- Assure les opérations de pilotage, de lamanage ainsi que l'avitaillement en eau.
- La gestion des produits dangereux.

I.10 CONCLUSION :

Dans ce chapitre nous avons défini les défférents concepts liés à notre thème et nous avons présenté notre organisme d'accueil. Dans le chapitre suivant nous allons aborder l'étude de l'existant et nous exposerons ensuite ses limites.

ChapitreII. Etude de l’existant

II.1 Introduction :

Après avoir présenté dans le premier chapitre l'organisme d'accueil, et un ensemble de concepts liés à notre thème, nous consacrons ce chapitre pour faire l'étude de l'existant dans le but de comprendre l'infrastructure de l'entreprise ainsi que leur système d'information et cerner les limites de l'existant, et cela nous aidera à mieux comprendre les objectifs de notre travail.

II.2 Infrastructure de l'entreprise :

II.2.1 Présentation de réseau EPAL :

- **Topologies du réseau d'EPAL :**

La topologie de réseau informatique de l'entreprise d'Alger c'est **Tree layer hiérarchique model** (modèle hiérarchique de Cisco de trois couches) à savoir la couche d'accès, la couche distribution et la couche cors.

La figure suivante représente la topologie de ce réseau :

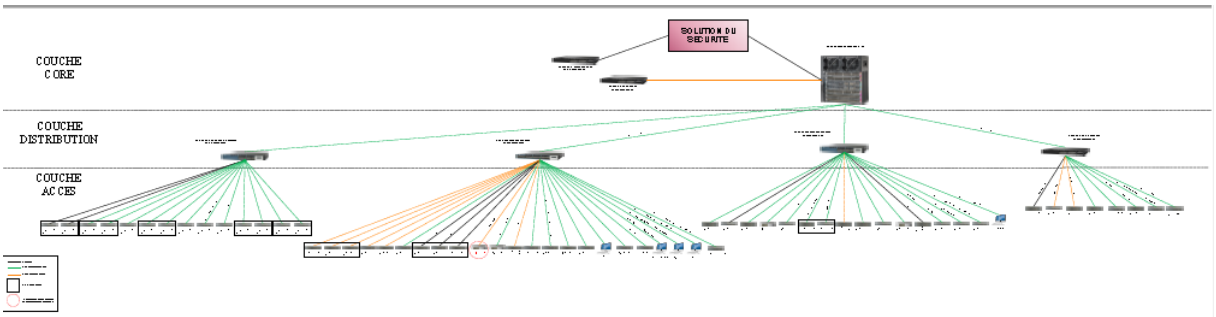


Figure 7:la topologie de ce réseau.

Le réseau est divisé en trois zones à savoir DMZ, WAN et LAN

- **DMZ zone :**

Cette partie représente la zone démilitarisée du réseau EPAL. Situé entre la partie Lan zone et la partie Wan zone derrière deux types de pare-feu différents et redondants. Elle est accessible depuis réseau interne et externe de l'entreprise. Le but est ainsi d'éviter toute connexion directe au réseau interne.

- **WAN zone :**

La Wan zone ou bien internet zone a pour rôle le regroupement de la partie internet avec les serveurs et les équipements de sécurité dans l'entreprise.

- **LAN zone :**

La partie LAN zone est la partie dorsale du réseau de l'EPAL. Elle représente toutes les connexions entre plusieurs switches de niveau 2.

Zone serveur :

L'armoire centrale (data center) dispose de cinq serveurs à l'intérieur, de deux serveursde base de données (oracle) et d'un serveur web.

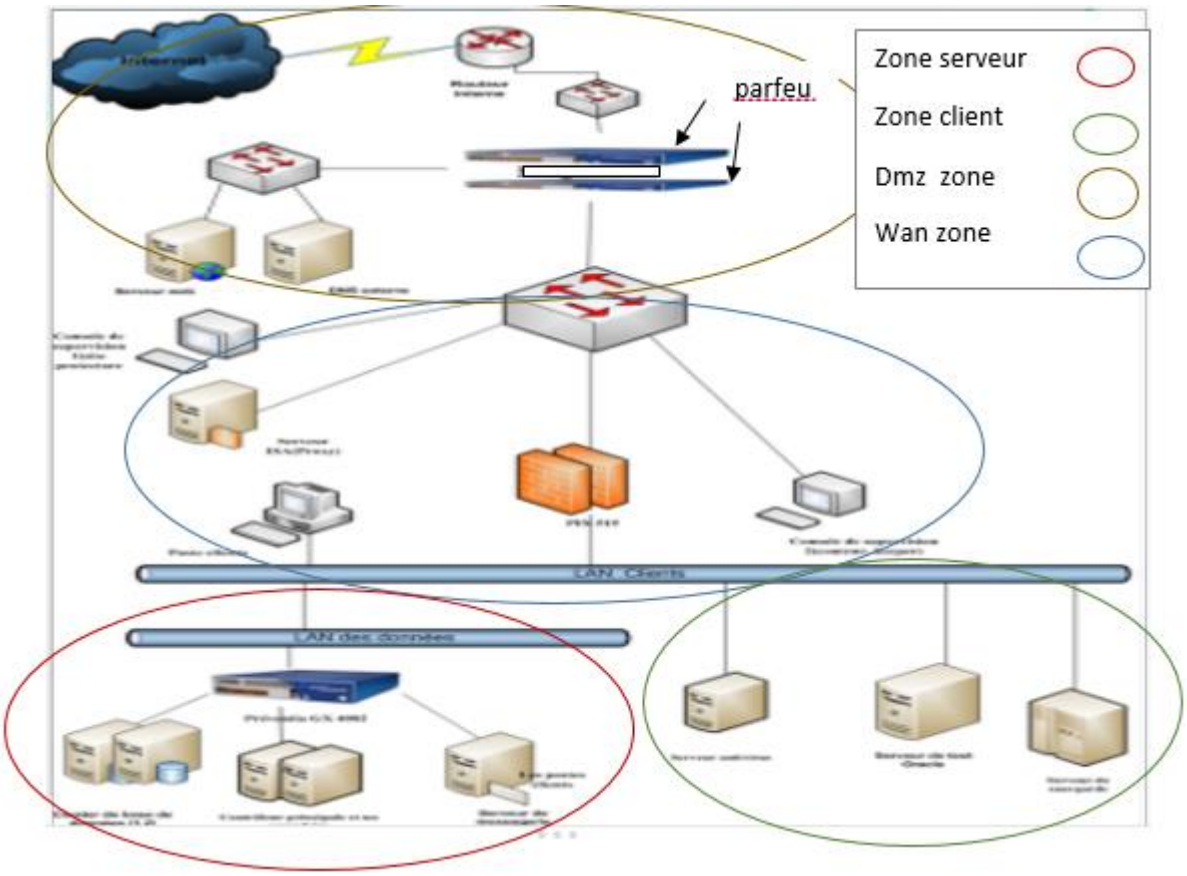


Figure 8:Les zones du réseau de l'EPAL.

- **Représentation géographique :**

Le port d'Alger a été divisé en trois zones à savoir zone nord, zone centrale et la zone sud, et cela pour faciliter la gestion du réseau. Tout le port est couvert de la fibre optique afin d'assurer la circulation des données.

La figure suivante représente les différentes zones :

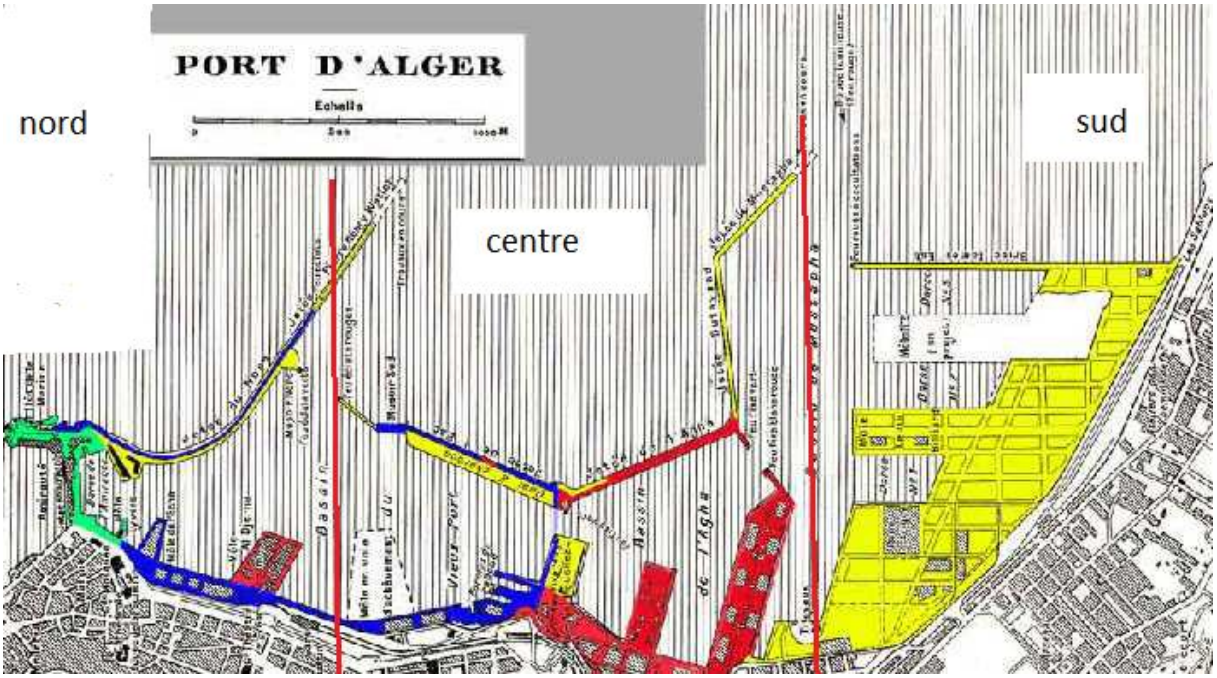


Figure 9: trois zones d'entreprise portuaire d'Alger.

Chaque zone dispose d'une armoire (distribution), qui permet la distribution du réseau entre les directions du port. Les trois armoires sont reliées à une armoire centrale.

II.2.2 Système d'informations de L'EPAL :

Le système d'informations de l'entreprise EPAL est considéré un élément central dans EPAL Il permet aux différents acteurs de véhiculer des informations et de communiquer grâce à un ensemble de ressources matérielles, humaines et logicielles.

➢ Le Synoptique de système d'informations :

La figure suivante représente la répartition des composants dans le système d'informations portuaire d'Alger:

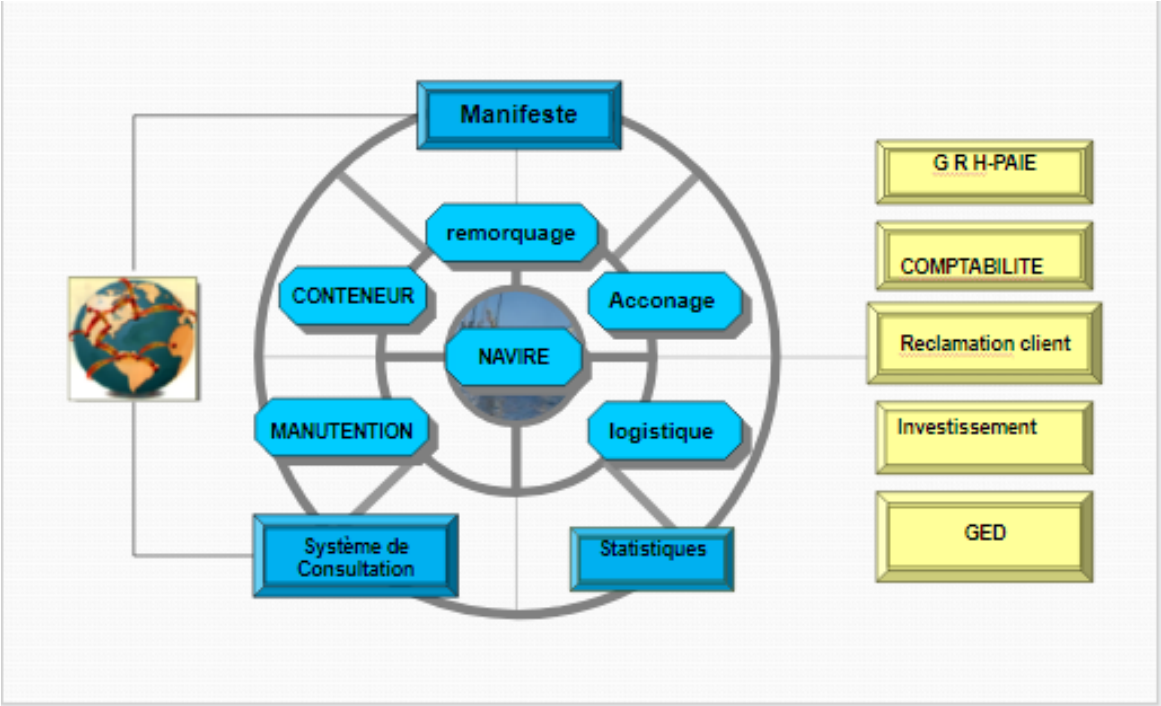


Figure 10: Le synoptique de système d'information de l'EPAL

II.3 Descriptions des applications et leurs modules :

Les activités opérationnelles utilisent la solution de GestPort. Cette dernière est une solution développée en centura, Elle assure la gestion de tout ce qui est opérationnel. Elle contient essentiellement cinq composants à savoir manutention, capitainerie, DTC (direction terminal et conteneur), acconage, statistiques. Chaque composant se devise en plusieurs modules. La solution Gestport est associée à une seule base de données oracle qui se compose de plusieurs schémas ou chaque schéma contient plusieurs tables. Chaque schéma est associé à un composant de la solution Gestport.

Les composants de la solution GestPort sont définis ci-dessous :

- **Manutention :**

C'est une application chargée des opérations d'embarquement et de débarquements des marchandises. Manutention se devise en plusieurs modules parmi ces derniers on trouve :

- Traitement des navires, assure l'embarquement et le débarquement des marchandises

Le composant manutention contient un schéma en base de données appelé Epalmanut

Ce schéma contient plus de 50 tables, on cite quelques-unes : AFFECTNUM, AGENT_ESC_PROV, DOUBLAGE, ENGIN, ESCALE, ESCALE_PROV, EVPSPEC, FACTURE…etc.

- **Acconage :**

Comprends les opérations tendant à assurer la réception, le pointage et la reconnaissance à terre des marchandises soient embarquées ou débarquées ainsi que leur gardiennage jusqu'à leur embarquement ou leur délivrance au destinataire.

Il existe deux types de marchandises les marchandises roulant (marchandises qui roulent tel que véhicules... etc.) ou divers (marchandises non conteneurisées et non roulent telles que les cartons ...etc.)

Acconage se compose de plusieurs modules à savoir :

- Gestion de manifeste : un manifeste est un document qui décrit l'ensemble des marchandises embarquées sur un naire, il est transmet par le consignataire par email dans un format Txt ce fichier est par la suite traité via l'application.

Acconage contient un schéma en base de données appelé **Acconage**

Ce schéma contient plus de 50 tables, on cite quelques-unes : ACTIVITE, COMMANDE, COMMANDEDACCES, COMPTE CONSIGNATAIRE, DETAFFECT_ENGIN …etc.

- **Conteneur :**

C'est un composant qui a les mêmes opérations avec acconage mais avec un traitement différent. Il se compose des modules suivants :

- Poussage
- Scanner

Conteneur contient un schéma en base de données appelé **Conteneur**

Ce schéma contient plus de 40 tables, on cite quelques-unes : AFFECTSCELL, CHARGER, CHOMAGE, DETAIPVDGX, DETBONGRUE, DETFACTSCANNER…etc.

- **Capitainerie :**

La capitainerie assure la gestion du trafic maritime dans le port pour les navires commerciaux et non commerciaux, leurs prises en charge total pendant la durée de leurs Escales dans le port.

Capitainerie se compose de plusieurs modules à savoir :

- **Suivi des navires** : il gère toutes les opérations d'arriver et de sortie des navires.
- **Exploitation :** ce module consiste à faire l'opération de pilotage[1] des navires.
- **Remorquage :** le remorquage consiste à l'assistance des navires à l'entrée et à la sortie du port ainsi qu'à tous les mouvements dans le port.
- **Facturation :** ce module consiste à faire la facturation pour le pilotage et la facturation pour le remorquage.

La capitainerie contient un schéma en base de données appelé **Capitainerie**

Ce schéma contient plus de 60 tables, on cite quelques-unes : CLIENT, COMMANDE, COMMANDER, CONTROLE, DETAILTAXER, DETJOURNAL …etc.

[1] Pilotage : Le pilotage consiste à l'assistance donnée aux capitaines des navires par les pilotes et les lamaneurs pour la conduite de tous les mouvements des navires dans le port et en rade

- **Statistiques :**

  Les statistiques sont consacrées à :

  - Faire une étude statistique sur le trafic global des marchandises ainsi que le trafic passager.
  - L'évolution (et / ou la diminution) des importations et des exportations.
  - L'étude de stabilité des importations.

II.4 Etude sur les bases de données :

Après avoir étudié et analyser le fonctionnement de la base de données de Gestport nous avons remarqué que :

- Il y a des mêmes tables qui apparaissent dans plusieurs schémas tel que la table navire qui existe dans le schéma capitainerie et epalmanut. Si on affecte une modification sur une de ces tables il faut qu'on affect les mêmes modifications dans toutes les instances de la table et cela est fait manuellement et donc ça prend beaucoup du temps et s'ils oublient de modifier dans toutes les instances de la table on aura des problèmes.

La figure suivante représente un exemple de table qui apparait dans plusieurs schémas :

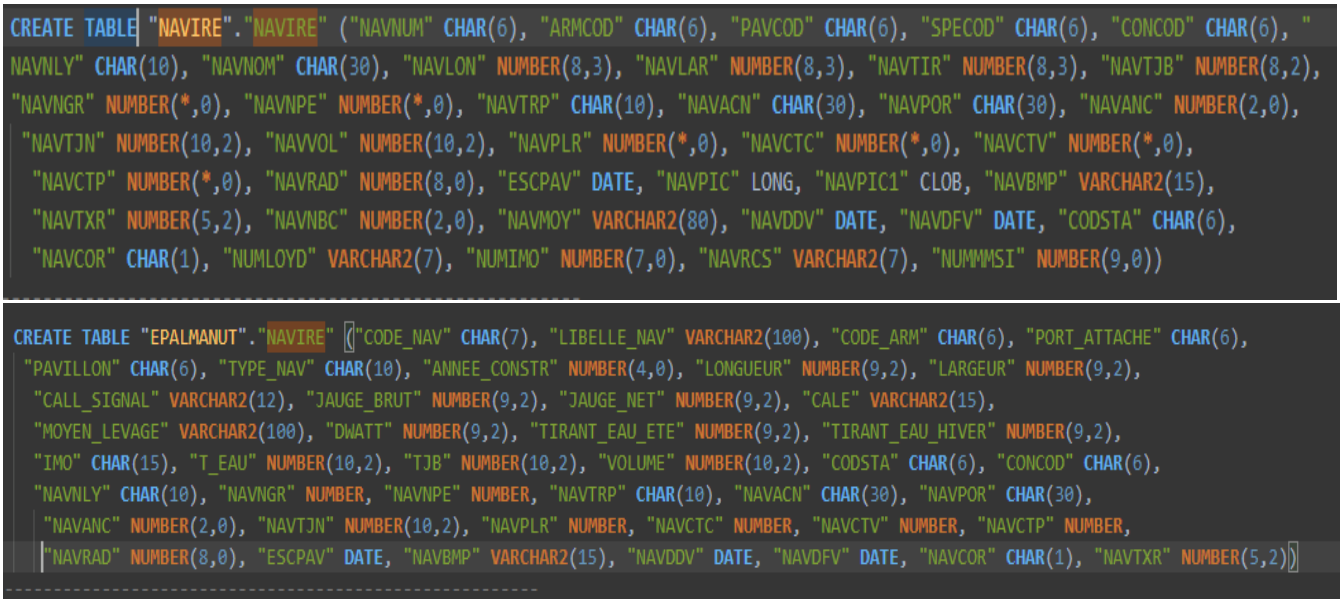


Figure 11: les tables de base de données apparient dans plusieurs schémas

- Il existe des tables qui sont sans relation comme : LETTRE, LIGNE_JOURNAL_FACTURE, JOURNAL_FACTURE

- Il existe des tables avec les mêmes attributs et la même clés primaire

si juste le nom de la table qui est différents comme la figure ci-dessous le montre :

```
CREATE TABLE "NAVIRE"."ACTIVITE" ("ACTNUM" NUMBER(8,0), "QAICOD" CHAR(6), "ESCNUM" NUMBER(8,0), "PRSNUM" NUMBER(8,0),
"UTICOD" CHAR(6), "SHFNUM" NUMBER(8,0), "ACTDRS" DATE, "ACTDAD" DATE, "ACTHED" DATE, "ACTDAF" DATE, "ACTHEF" DATE, "ACTTPJ" CHAR(1),
"ACTSNS" CHAR(1), "ACTOBS" VARCHAR2(60), "ACTPER" CHAR(2), "ACTDUR" NUMBER(2,0), "NAVNUM" CHAR(6), "AGEMAT" CHAR(6), "ACTEQU" VARCHAR2(120))

CREATE TABLE "NAVIRE"."ACTIVITREM" ("ACTNUM" NUMBER(8,0), "QAICOD" CHAR(6), "ESCNUM" NUMBER(8,0), "PRSNUM" NUMBER(8,0),
 "UTICOD" CHAR(6), "SHFNUM" NUMBER(8,0), "ACTDRS" DATE, "ACTDAD" DATE, "ACTHED" DATE, "ACTDAF" DATE, "ACTHEF" DATE,
  "ACTTPJ" CHAR(1), "ACTSNS" CHAR(1), "ACTOBS" VARCHAR2(60), "ACTPER" CHAR(2), "ACTDUR" NUMBER(2,0), "NAVNUM" CHAR(6),
  "AGEMAT" CHAR(6), "ACTEQU" VARCHAR2(120))
```

Figure 12 : représente les tables de base de données qui ont la même clé primaire et mêmes attributs.

- Des tables qui contiennent la même clé primaire comme le montre la figure suivante :

```
TIME$$" DATE, "DMLTYPE$$" VARCHAR2(1), "OLD_NEW$$" VARCHAR2(1), "
                CHANGE_VECTOR$$" RAW(255))

  CREATE TABLE "NAVIRE"."MLOG$_PLANIFIER" ("NAVNUM" CHAR(6), "PR
VNUM" CHAR(10), "QAICOD" CHAR(6), "PPVCOD" CHAR(6), "PLNORD" NUME
R(3,0), "SNAPTIME$$" DATE, "DMLTYPE$$" VARCHAR2(1), "OLD_NEW$$" VA
RCHAR2(1), "CHANGE_VECTOR$$" RAW(255))

CREATE TABLE "NAVIRE"."NAVIRE" ("NAVNUM" CHAR(6), "ARMCOD" CH
AR(6), "PAVCOD" *
CHAR(6), "SPECOD" CHAR(6), "CONCOD" CHAR(6), " NAVNLY" CHAR(10), "N
AVNOM" CHAR(30), "NAVLON" NUMBER(8,3), "NAVLAR" NUMBER(8,3), "NA
VTIR" NUMBER(8,3), "NAVTJB" NUMBER(8,2),
  "NAVNGR" NUMBER(*,0), "NAVNPE" NUMBER(*,0), "NAVTRP" CHAR(10), "N
AVACN" CHAR(30), "NAVPOR" CHAR(30), "NAVANC" NUMBER(2,0),
 "NAVTJN" NUMBER(10,2), "NAVVOL" NUMBER(10,2), "NAVPLR" NUMBER(*,0
), "NAVCTC" NUMBER(*,0), "NAVCTV" NUMBER(*,0),
    "NAVCTP" NUMBER(*,0), "NAVRAD" NUMBER(8,0), "ESCPAV" DATE, "NAV
PIC" LONG, "NAVPIC1" CLOB, "NAVBMP" VARCHAR2(15),
    "NAVTXR" NUMBER(5,2), "NAVNBC" NUMBER(2,0), "NAVMOY" VARCHAR
2(80), "NAVDDV" DATE, "NAVDFV" DATE, "CODSTA" CHAR(6),
   "NAVCOR" CHAR(1), "NUMLOYD" VARCHAR2(7), "NUMIMO" NUMBER(7,0),
"NAVRCS" VARCHAR2(7), "NUMMMSI" NUMBER(9,0))
```

Figure 13:les tables de base de données qui ont la même clé primaire

II.4.1 Limite des bases de données :

Après avoir analysé les schémas de la base de données de Gestport on a pu extraire les limites suivantes :

- Tables sans relation (tables isolées)
- Redondance des tables de base de données
- Des tables dupliquées dans les autres bases de données
- Problèmes de cohérence de données

La solution Gestport contient aussi d'autres limites au niveau de :

- **Développement :**
  - Problème d'ergonomie l'application est développée avec un ancien langage de programmation
  - L'utilisation de cette application est ennuyeuse
  - Problème d'intégration du code, si on affect une mise à jour d'une fonctionnalité d'un module on doit réintégrer tout le module pour qu'il puisse fonctionner cela provoque une perte du temps
- **Testes :**

  Il n'y a pas d'inconvénients à ce niveau
- **Déploiement :**

  Gestion de versions, faire des mises à jour à la solution GestPort nécessite un déplacement au niveau de chaque module.
- **Maintenance :**
  - L'augmentation des couts de la maintenance et de gestion.
  - Cette solution ne nous offre pas la liberté d'utiliser d'autres langages de programmation.

II.5 Conclusion :

Après avoir fait notre étude de l’existant et cerné les limites de la solution GestPort, nous allons entamer la partie conception de notre travail dans le prochain chapitre, en présentant l’approche et la méthodologie à utiliser pour faire la migration.

ChapitreIII. Analyse et Conception

Ce chapitre sera consacré à l'analyse et la conception de notre application, nous allons le décomposer en deux parties, la première partie sera consacrée à la décomposition de notre application en microservices et cela en appliquant les différents concepts théoriques définis dans le chapitre1, et la deuxième partie sera consacrée à la conception d'un microservice de la capitainerie c'est dans cette partie-là qu'on va identifier les acteurs de microservice à réaliser ,nous allons identifier nos besoins que ce soit les besoins fonctionnels ou non fonctionnels ensuite nous présenterons les différents diagrammes UML ainsi que la conception de notre base de donné

Partie 1 :

III.1 Introduction :

Cette phase est l'une des plus importantes étapes dans le cycle de développement du projet car elle aide le concepteur à planifier les tâches. En effet, c'est au cours de celle-ci qu'on va appliquer les différents concepts théoriques définit aux parts avant comme le Domaine Driven Design (DDD) qui va nous aider à extraire les différents microservices à partir de la plateforme existante.

III.2 Décomposition de la GestPort en microservice :

Pour passer d'une architecture monolithique vers une architecture en microservicesnous Suivrons le pattern du contexte borné (bounded contexte) de la conception pilotée par le domaine (DDD) afin de pouvoir deviser notre vaste modèle (contexte) en modèles plus petits avec une meilleure cohésion et des frontières entre chaque modèle. Dans un premier lieu nous définissons le modèle métier global de l'EPAL qui met l'accent sur les différents composants métiers. Ensuite nous présentons notre carte de contexte (représente les différents modèles) et enfin nous présenterons notre bounded contexte.

III.2.1 Le modèle métier global d'EPAL :

L'entreprise EPAL gère plusieurs fonctionnalités à savoir les navires, le suivi de marchandise, la gestion des taxes, les factures, etc. Pour mieux comprendre le fonctionnement de l'entreprise EPAL on va présenter le modèle métier global d'EPAL qui met l'accent sur les différents composants métiers. La figure ci-dessous représente le schéma du modèle métier global d'EPAL qui va nous aider à mieux comprendre nos composants métiers :

Figure 14 : Modèle métier global d'EPAL.

III.2.2 Décomposition du modèles métier global d'EPAL en sous modèles :

Afin de décomposer notre modèle métier global en plusieurs sous modèles métier

On va regrouper toutes les fonctionnalités métiers qui appartiennent aux mêmes contextes dans un modèle (nommé modèle A par ex.), et cela en suivant le pattern bounded contexte de la conception pilotée par le domaine (DDD) ensuite on va définir les liens entre les composants métiers de même modèle.

Ici on prend les composants qui sont en relation et qui partagent le même contexte on va les regrouper ensemble dans le même modèle c'est-à-dire on va limiter notre modèle à qui on va associer un nom pour le distinguer des autres et pour comprendre de quel modèle s'agit '-il par exemple on va créer un modèle qui s'occupe juste de la taxation un autre pour la facturation ainsi de suite

Maintenant passant à la décomposition et la définition de chaque modèle de l'entreprise EPAL (on a pu décomposer notre modèle métier grâce aux réunions que nous avons faite avec les experts métiers de l'EPAL)

- Le composant Manifeste on va le mètre dans un modèle qu'on va nommer gestion manifeste.
- Le composant gérer navire on va le mètre dans un modèle qu'on va nommer Administrateur.
- Les composants suivi marchandise, pointage débarquement, pointage embarquement, débarquement, embarquement vont être dans le même modèle qu'on nomme pointage
- Les composants sortie navire, arriver navire, suivi navire, placement, mouvement navire, prévision vont être regroupés dans le même modèle escale
- Les composants pilotage, moyens de servitude, chargement, remorquage, déchargement, gardiennage sont dans le modèle prestation
- Les composants taxation pilotage et taxation remorquage sont regroupés dans le même modèle nommé taxation
- Le composant poussage est seul dans le modèle nommé poussage
- Le composant facturation est dans le modèle facturation

La figure suivante représente les différents modèles de l'EPAL et la relation entre les composants métiers d'un modèle ainsi que le nom de chaque modèle :

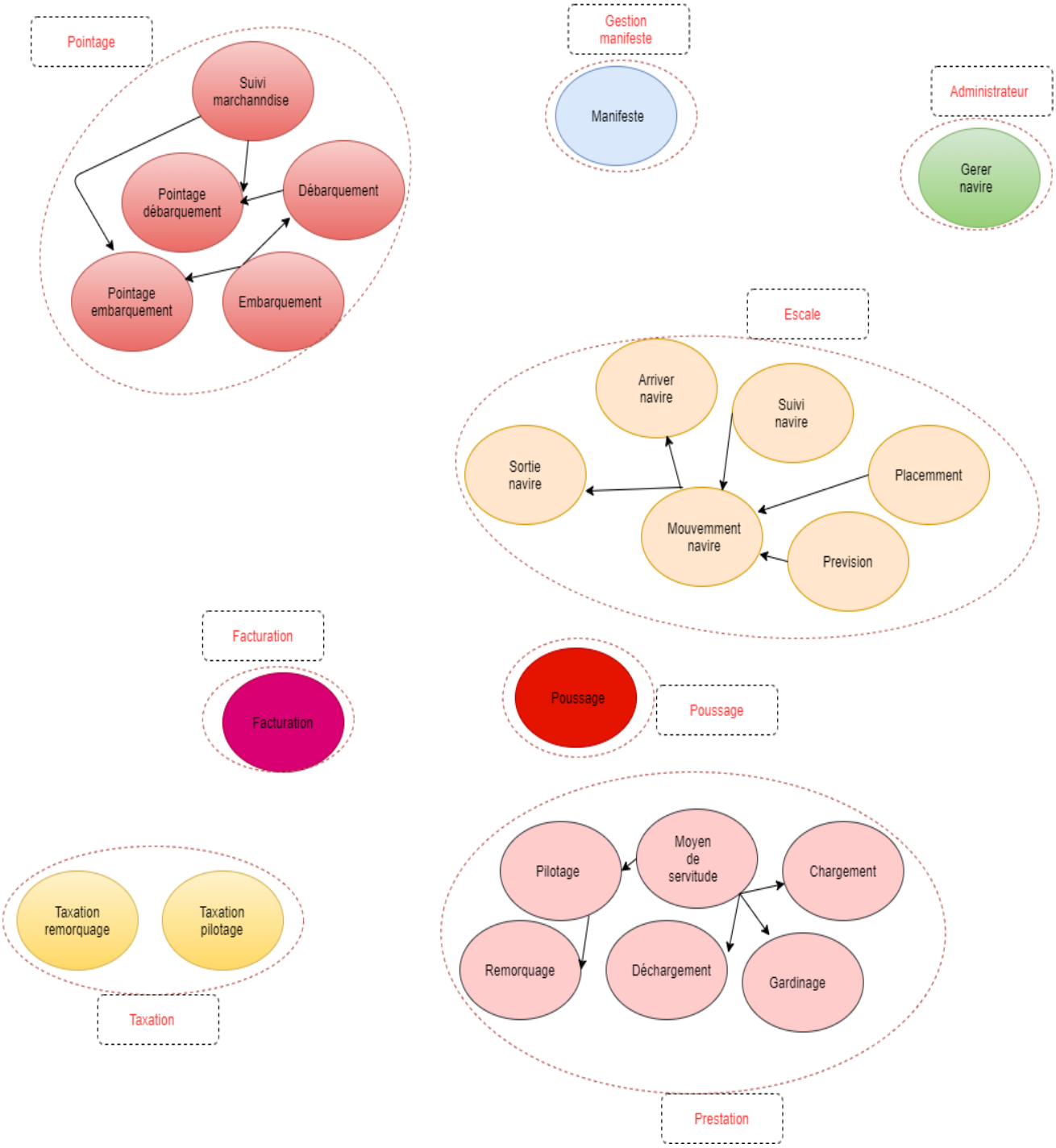


Figure 15:Le pattern bounded contexte.

III.2.3 Réalisation de la carte de contexte d'EPAL :

Après avoir défini les modèles on va essayer de faire les liens entre les composants des modèles définis auparavant (c'est-à-dire on définit les liens entre les composants de modèle A avec les composants de modèle B). Il est nécessaire de définir les frontières et les relations entre ces modèles et cela en se basant sur la carte de contexte du Domaine Driven Design.

La figure suivante représente notre carte de contexte :

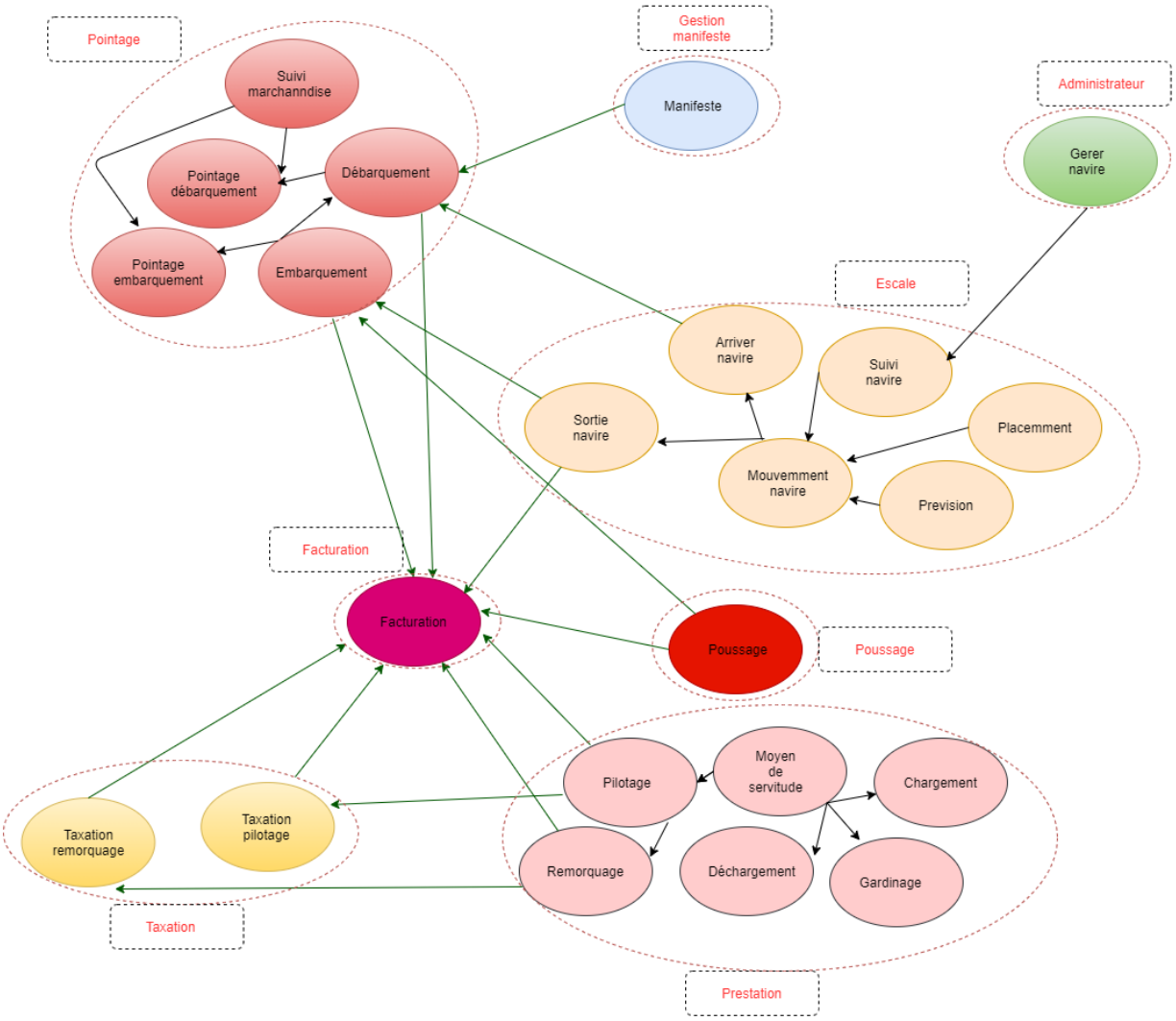


Figure 16 : La carte de contexte.

Chaque contexte présente un microservice autonome tel que la relation inter-microservices est forte et le couplage entre les microservices est faible.

Le paterne bounded contexte nous a permis d'extraire huit (8) microservices :

- Microservice Gestion Manifeste.
- Microservice Prestation.
- Microservice Escale.
- Microservice Pointage.
- Microservice Taxation.
- Microservice Facturation.
- Microservice Administrateur.
- Microservice Poussage.

Après avoir décomposé notre plateforme en microservices nous exposerons une démonstration de la solution proposée qui concernera les microservices de la capitainerie à savoir Escale.

III.3 Langage ubiquitus (langage omniprésent) :

Afin de ne pas avoir des incompréhensions liées aux termes utilisés pour désigner les différents éléments de notre modèle entre nous en tant que développer et les experts métier de l'EPAL nous avons défini le langage ubiquitaire lié à notre modèle Escale.

Par exemple :

- En tant qu'agent de commission je veux faire une prévision d'arriver navire.
- En tant qu'agent de mouvement je veux consulter la liste des navires en prévision de sortie
- En tant qu'agent de commission je veux consulter la liste des navires en prévision d'arriver.

Les termes expressions « agent de commission » « agent de mouvement » « prévision d'arriver navires » « navires en prévision d'arriver » « navires en prévision d'arriver » ... font partie du langage ubiquitaire de notre contexte.

Partie 2

III.4 Introduction :

Cette deuxième partie sera consacrée à la conception des microservices de la capitainerie à savoir Escale, on a choisi de réaliser ce microservice car la capitainerie est considérée comme le noyau de l'EPAL grâce à ses fonctionnalités et les opérations dont elle dispose.

Pour réaliser notre conception et afin de mieux expliquer notre conception on va utiliser les différents diagrammes UML. Tout d'abord nous allons définir c quoi UML et diagramme qui la compose ensuite nous présentant les acteurs de notre système après on identifiera les besoins fonctionnels et non fonctionnels. Ensuite nous exposerons les défirent diagramme UML utilisé à savoir les diagrammes de séquence, les diagrammes de cas d'utilisation et les diagrammes de classes et enfin nous présenterons la conception de notre base de données.

III.5 Langage UML :

UML, c'est un acronyme anglais pour « Unified Modeling Language » qu'on traduit par « Langage de modélisation unifié ». L'UML est un langage de modélisation qui permet de représenter graphiquement les besoins des utilisateurs [17] en utilisant des diagrammes qui donnent chacun une vision différente du projet à traiter.

III.5.1 Les diagrammes UML :

Il existe deux types de diagrammes à savoir les diagrammes statiques, et les diagrammes de comportements :

- **Diagramme statique (de structure) :**
  - Diagramme de classes
  - Diagramme d'objets
  - Diagramme de composants
  - Diagramme de déploiement
  - Diagramme des paquets
  - Diagramme de structure composite

➢ **Diagramme de comportement :**

- Diagramme des cas d'utilisation.
- Diagramme état transition.
- Diagramme d'activité.
- Diagramme de séquence.
- Diagramme de communication.
- Diagramme global d'interaction.
- Diagramme de temps.

III.6 Analyse de microservice « ESCALE » :

III.6.1 Identification des acteurs :

- **Acteur :** un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagit directement avec le système étudié [18].

➢ **Les acteurs de notre système sont :**

- **Agent de commission :** celui qui gère les prévisions des navires.
- **Agent de mouvement :** celui qui gère les escales des navires.

III.6.2 Spécification des besoins :

Les besoins sont divisés en deux catégories à savoir les besoins fonctionnels et les besoins non fonctionnels :

- **Besoins fonctionnels :**

Ce sont les actions et les réactions que le système doit faire suite à une demande d'un acteur

- **Besoins fonctionnels de l'agent de commission :**

L'application doit permettre à l'agent de commission de :

- Faire une prévision d'arrivée navire.
- Afficher la liste des navires.

- **Besoins fonctionnels de l'agent de mouvement :**

L'application doit permettre à l'agent mouvement de :

- Remplir l'observation procès-verbal.
- Afficher le procès-verbal.
- Modifier le procès-verbal.

- **Besoins non fonctionnels :**

Il s'agit des besoins qui caractérisent le système.

Les besoins non fonctionnels de notre application sont les suivants :

- **Tolérance aux pannes :**

  L'EPAL doit fonctionner même s'il y a des microservices en défaillance.

- **Temps de réponse :**

  L'EPAL doit rendre une réponse dans un temps minimal.

- **Maintenabilité :**

  L'architecture doit permettre l'évolution et assurer l'extensibilité de l'application.

- **Monitoring :**

L'architecture doit permettre la surveillance permanente de notre système dans un but préventif.

III.7 Analyse de microservice administrateur :

III.7.1 Identification des acteurs :

- **Administrateur :** celui qui gère toutes les fonctionnalisées de notre système.

III.7.2 Spécification des besoins :

Les besoins sont divisés en deux catégories à savoir les besoins fonctionnels et les besoins non fonctionnels :

- **Besoins fonctionnels :**

Ce sont les actions et les réactions que le système doit faire suite à une demande d'un acteur principal.

- **Besoins fonctionnels de l'administrateur :**

L'application doit permettre à l'administrateur de :

- Gérer les navires.
- Gérer les consignataires.
- Gérer les lignes.
- Gérer les Operations.

- **Besoins non fonctionnels :**

Il s'agit des besoins qui caractérisent le système.

Les besoins non fonctionnels de notre application sont les suivants :

- **Tolérance aux pannes :** L'EPAL doit fonctionner même s'il y a des microservices en défaillance
- **Temps de réponse :** L'EPAL doit rendre une réponse dans un temps minimal
- **Maintenabilité :** L'architecture doit permettre l'évolution et assurer l'extensibilité de l'application.

- **Monitoring :** L'architecture doit permettre la surveillance permanente de notre système dans un but préventif.

III.8 La conception de microservice « ESCALE » :

III.8.1 Diagramme de contexte :

Le diagramme de contexte met en évidence le champ d'application et les acteurs intervenant.

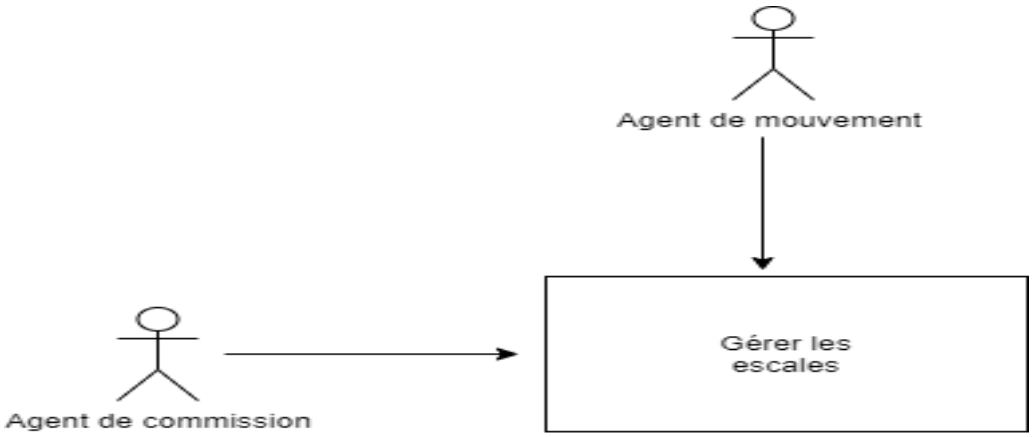


Figure 17: Diagramme de contexte pour le microservice escale.

III.8.2 Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation permet de représenter visuellement une séquence d'actions réalisées par un système, représenté par une boîte rectangulaire, produisant un résultat sur un acteur[19], appelé acteur principal, et ceci indépendamment de son fonctionnement interne.

- **Cas d'utilisation :** Un cas d'utilisation est un ensemble d'actions réalisées par le système en réponse à une action d'un acteur.
- **Relation« include » :** Cas d'utilisation source incorpore explicitement et de manière obligatoire le comportement décrit dans le cas d'utilisation destination.
- **Relation «extend »** : Le cas d'utilisation source incorpore implicitement et de manière facultative un autre cas d'utilisation à l'endroit spécifié. Ceci veut dire que le cas d'utilisation source « étend » (en anglais « extend ») ou précise le cas d'utilisation destination.

➢ Diagramme de cas d'utilisation de microservice Escale :

Le microservice escale doit gérer les prévisions des navires et le suivi des escales donc nous allons présenter les diagrammes de cas d'utilisation pour les prévisions ainsi que pour le suivi d'escales.

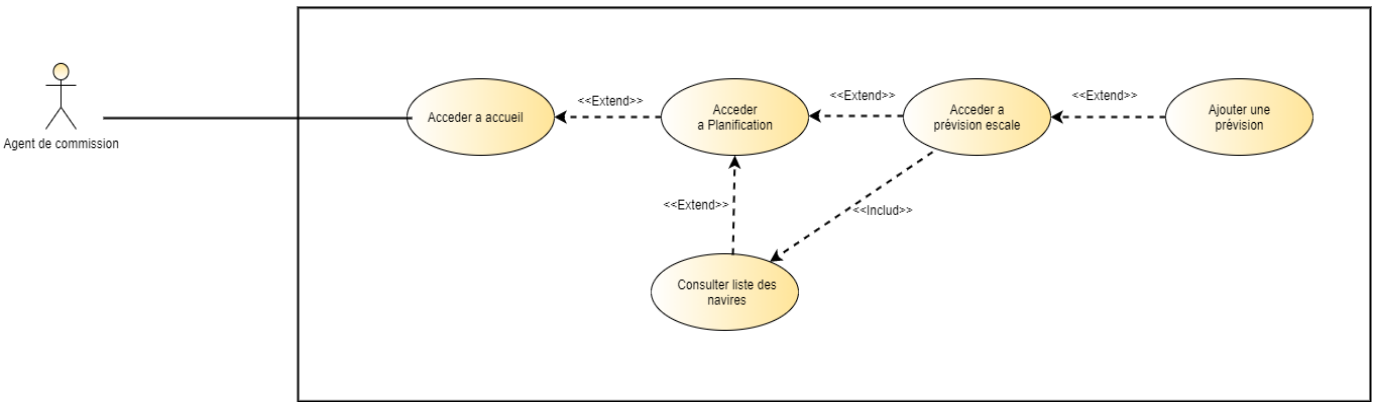


Figure 18 : Diagramme de cas d'utilisation de prévision d'escale.

L'agent de commission doit tout d'abord accéder à l'accueil ensuite il peut sélectionner prévision d'escale et il peut aussi faire une prévision d'arriver.

➢ Diagramme de cas d'utilisation de procès-verbal :

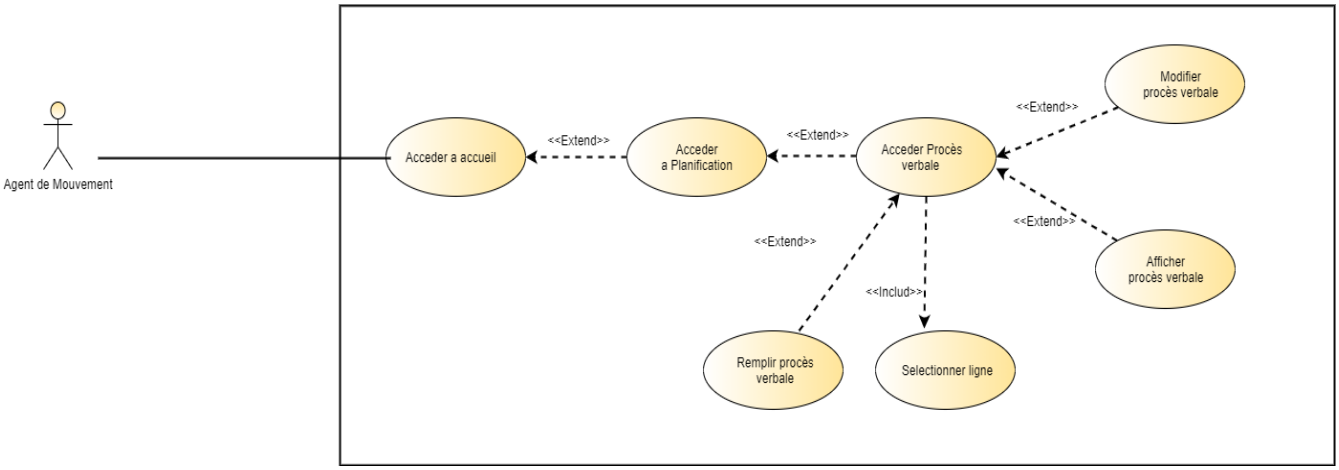


Figure 19:Diagramme de cas d'utilisation de procès-verbal.

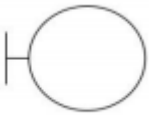
L'agent de mouvement doit tout d'abord accéder à l'accueil ensuite il doit accéder à procès-verbal et sélectionner une ligne après il peut afficher le procès-verbal ou bien le modifier ainsi qu'il peut remplir le formulaire de procès-verbal.

III.8.3 Diagrammes de séquence :

Diagramme de séquences : c'est un diagramme qui présente la vue dynamique du système. Il sert à représenter les interactions entre les objets en indiquant la chronologie des échanges. Dans cette étape nous allons déterminer les objets et les classes d'analyse, ces classes peuvent être réparties en trois catégories d'objets :

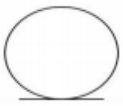
- **Les objets d'interface :** Ils représentent l'interface entre l'acteur et le système ou les pages Web complète dans le cas d'application web.

  L'icône :



- **Les objets entités :** Ce sont des objets d'écrits dans un cas d'utilisation mais qui lui survivront, c'est-à-dire qui se trouve dans d'autres cas d'utilisation.

  L'icône :



- **Les objets contrôlent :** Ils représentent le processus, c'est-à-dire les activités systèmes tel qu'un calcul ou une recherche, ils dirigent les objets entité et interface.

  L'icône :

- **Diagrammes de séquence pour consultation de la liste des navires :**

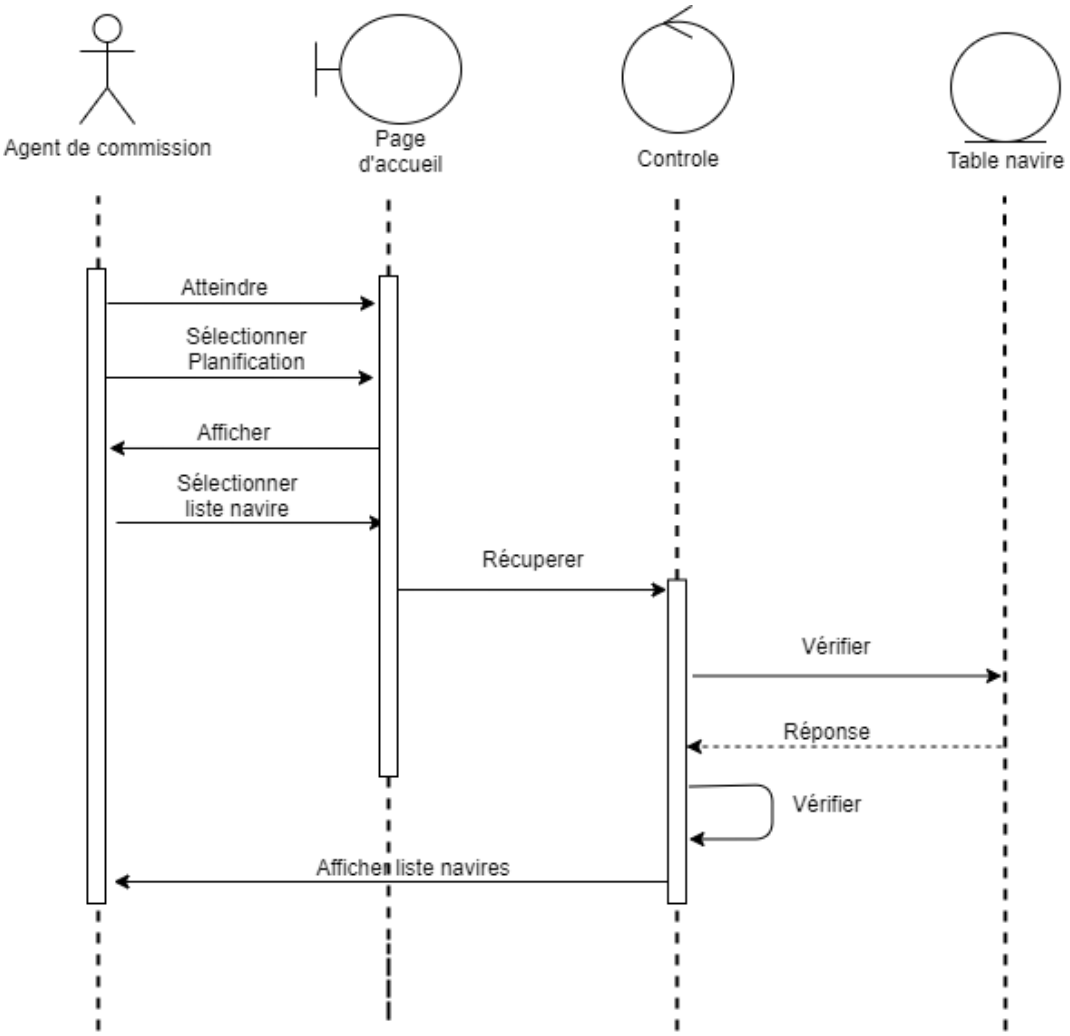


Figure 20 : Diagramme de séquence consulter liste navires.

1)L'agent de commission accède à la page d'accueil.

2)L'agent de commission sélectionne l'option Planification.

3)Le système lui affiche une liste d'options et il sélectionne l'option consulter navires

4)Le système récupère la liste des navires au niveau de la base de données et la liste des navires lui sera affichée

- **Diagrammes de séquence pour l'affichage de la liste des procès-verbaux :**

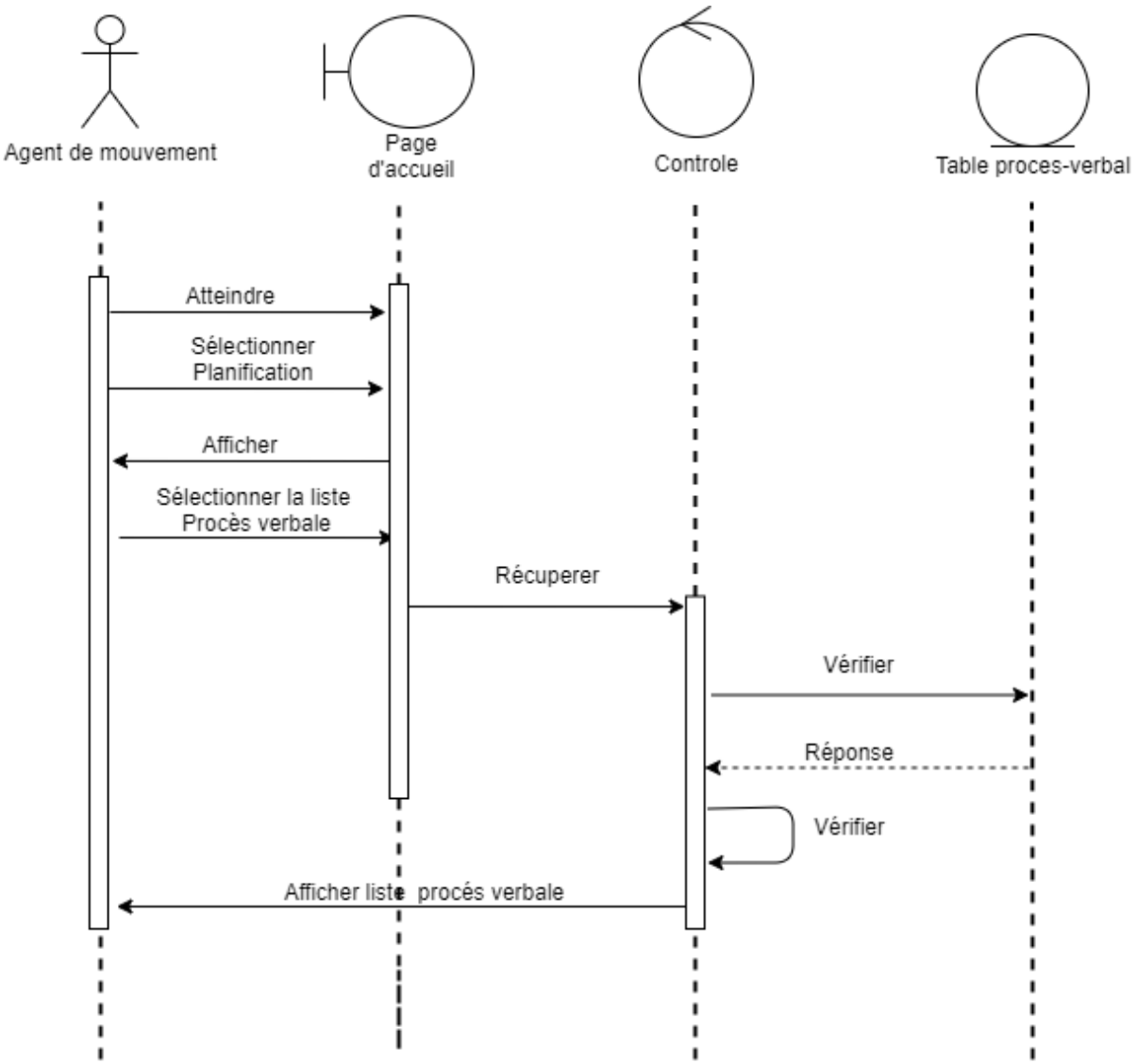


Figure 21 : Diagramme de séquence afficher procès verbale.

1)L'agent de mouvement accède à la page d'accueil.

2)L'agent de mouvement sélectionne l'option Planification.

3)Le système lui affiche une liste d'options et il sélectionne l'option afficher procès-verbal

4)Le système récupère la liste des procès-verbaux au niveau de la base de données et la liste des procès verbale lui sera affichée

- **Diagrammes de séquence pour la modification de procès-verbaux :**

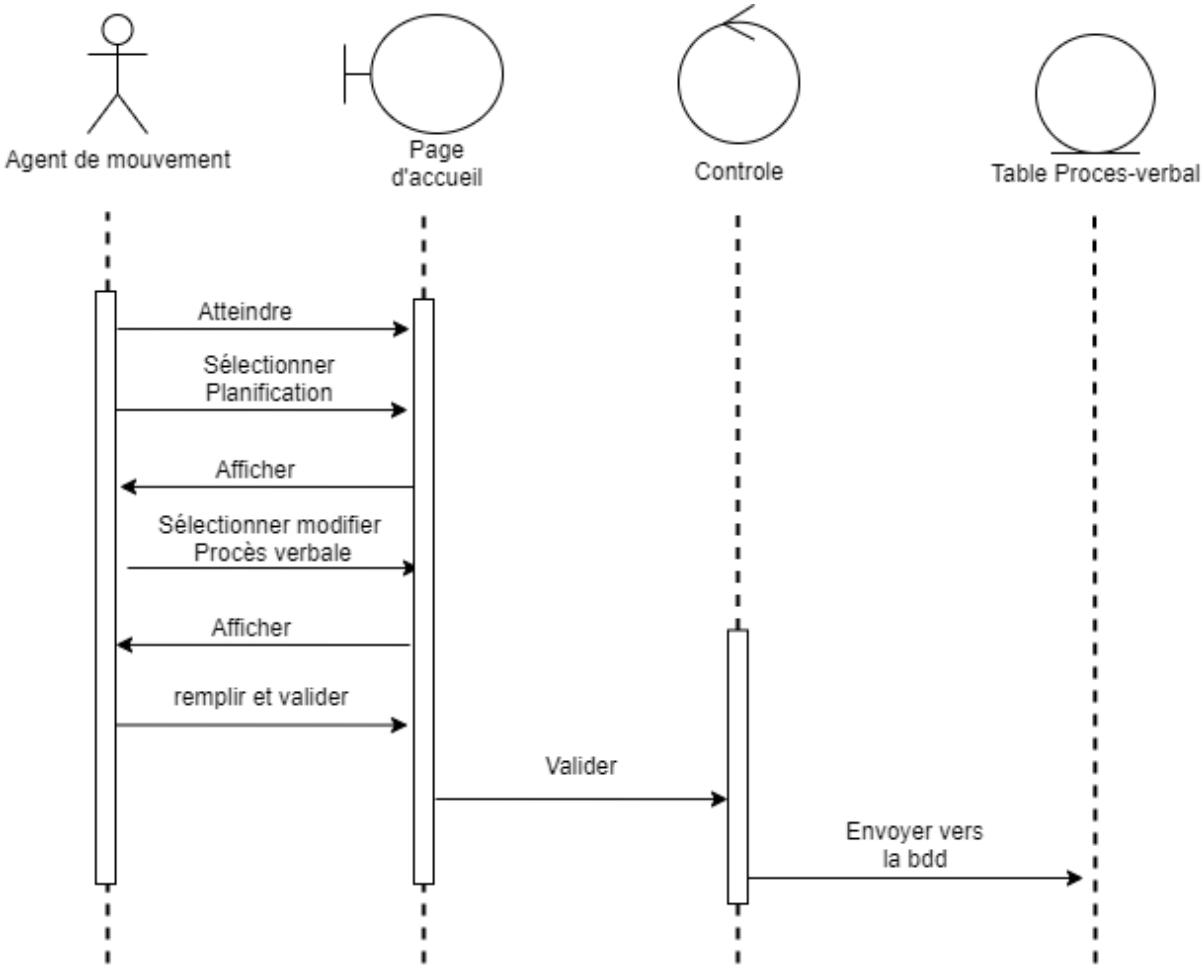


Figure 22:Diagramme de modifications de procès-verbal.

1. L'agent de mouvement accède à la page d'accueil.

2. L'gent de mouvement sélectionne l'option Planification.

3. Le système lui affiche une liste d'options et il sélectionne modifier procès-verbal

4. Le système lui affiche un formulaire et l'agent de mouvement le rempli et le valide

5. Le système enregistre les informations au niveau de la base de données

III.8.4 Diagramme de classe :

C'est un diagramme qui représente la vue statique des objets pages, son intérêt majeur est de modéliser les entités d'un système. Il représente l'architecture conceptuelle du système. Autrement dit, il exprime les relations existantes entre les pages client et serveur.

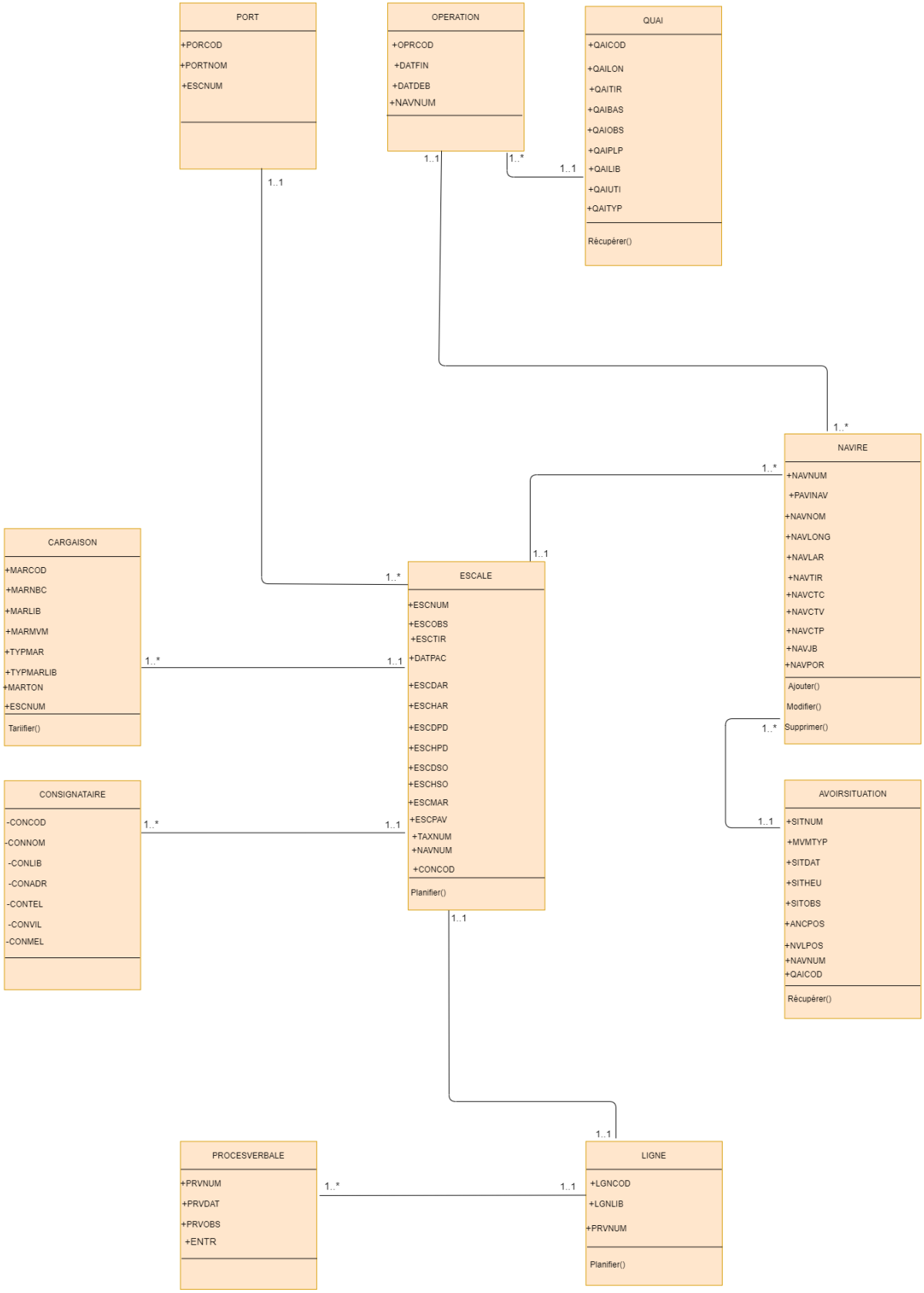


Figure 23 : Diagramme de classe ESCALE.

III.8.5 Conception de la base de données :

- **Règles de gestion :**

  - Une escale peut avoir un à plusieurs navires.
  - Un navire concerne une et une seule escale.
  - Une escale peut être planifié par un a plusieurs consignataires.
  - Un consignataire planifie une et une seule escale.
  - Une escale peut comporter une à plusieurs cargaisons.
  - Une cargaison concerne une et une seule escale.
  - Une escale suit une et une seule ligne.
  - Une ligne concerne une et une seule escale.
  - Une escale à un seul port de départ et un seul port d'arrivé.
  - Un port départ ou d'arrivé est concerné par plusieurs escales.
  - Un navire peut effectuer une et une seule opération
  - Une opération concerne un et un seul navire.
  - Un navire peut avoir une et une seule situation.
  - Une situation concerne un et un seul navire.
  - Une ligne peut avoir un à plusieurs procès-verbaux.
  - Un procès-verbal appartient à une et une seule ligne.
  - Une opération utilise un et un seul quai.
  - Un quai est utilisé pour une ou plusieurs opérations.

- **Le niveau logique de la base de données :**

- **ESCALE** (**ESCNUM**, ESCDAR, ESCHAR, ESCDPD, ESCHPD, ESCDSO, ESCHSO, ESCMAR, DATPAC, ESCTIR, ESCOBS, ESCPAV, *CONCOD, *NAVNUM)

- **NAVIRE** (**NAVNUM**, NAVNOM, NAVLONG, NAVLARG, NAVTIR, NAVCTC, NAVCTV, NAVCTP, NAVJB, NAVPOR)

- **CARGAISON (MARCOD**, MARLIB, MARMVM, TYPMAR, TYPMARLIB, MARTON, MARNBC, *ESCNUM)

- **CONSIGNATAIRE** (**CONCOD**, CONNOM, CONLIB, CONADR, CONTEL, CONVIL, CONMEL)

- **OPERATION** (**OPRCOD**, DATOPR, POSOPR)

- **QUAI** (**QAICOD**, QAILOG, QAITIR, QAIBAS, QAIOBS, QAIPLP, QAIUTI, QAITYP, QAILIB, *OPRCOD)

- **LIGNE** (**LGNCOD**, LGNLIB, * OPRCOD)

- **AVOIR-SITUATION**(**SITNUM** , SITHEU, SITDAT, SITOBS, MVMTYP, ANCPOS, *NAVNUM)

- **PROCES_VERBALE** (**PRVNUM**, PRVDAT, PRVOBS, ENTR)

- **PORT** (**PORCOD**, PORNOM, CODSTA, *ESCNUM)

➢ Le niveau physique de la base de données :

- **ESCALE :** correspond à un voyage

Nom du champ	Type de donnée	Description	Clef
ESCNUM	Integer(10)	Numéro de l'escale	**Primaire**
ESCDAR	Date	Date arriver	
ESCHAR	Time	Heure arriver	
ESCDPD	Date	Date prévue départ	
ESCHPD	Time	Heure prévue départ	
ESCDSO	Date	Date de sortie	
ESCHSO	Time	Heure de sortie	
ESCMAR	Varchar(255)	Marchandises	
DATPAC	Date	Date prévue d'accostage	
ESCTIR	Integer(10)	Tiran d'eau	
ESCOBS	Varchar(50)	Observation	
ESCPAV	Varchar(15)	Pavillon	
CONCOD	Varchar(15)	Code consignataire	**Etrangère**
NAVNUM	Varchar(15)	Numéro navire	**Etrangère**

Tableau 2: Niveau physique de la table « ESCALE».

- **NAVIRE :** correspond aux informations d'un navire

Nom du champ	Type de donnée	Description	Clef
NAVNUM	Integer(10)	Numéro navire	**Primaire**
NAVNOM	Varchar(15)	Nom navire	
NAVLONG	Integer(10)	Longueur	
NAVLARG	Integer(10)	Largeur	
NAVTIR	Integer(10)	Tiran d'eau	
NAVCTC	Integer (15)	Capacité de transport conteneur	
PAVINAV	Varchar(15)	Pavillon	
NAVCTV	Integer (15)	Capacité de transport véhicule	
NAVCTP	Integer (15)	Capacité de transport passagers	
NAVJB	Integer(10)	Jauge brute	
NAVPOR	Varchar(15)	Port d'attachement	

Tableau 3:Le niveau physique de la table « NAVIRE ».

- **AVOIR_SITUATION** : correspond à la situation actuelle du navire (en rade, en quai ...)

Nom du champ	Type de donnée	Description	Clef
SITNUM	Integer(10)	Numéro situation	**Primaire**
SITHEU	Time	Heure situation	
SITDAT	Date	Date situation	
SITOBS	Varchar(255)	Observation	
MVMTYP	Varchar(15)	Type de mouvement	
ANCPOS	Varchar(15)	Ancienne position	
NVLPOS	Varchar(15)	Nouvelle position	
NAVNUM	Integer (10)	Numéro navire	**Etrangère**

Tableau 4: Le niveau physique de la table « AVOIR_SITUATION ».

- **LIGNE :** correspond à la ligne suivie par le navire de son démarrage jusqu'à son arriver

Nom du champ	Type de donnée	Description	Clef
LGNCOD	Integer(10)	Code ligne	**Primaire**
LGNLIB	Varchar (25)	Libellé ligne	

Tableau 5: Le niveau physique de la table LIGNE

- **CARGAISON :** correspond aux marchandise**s**

Nom du champ	Type de donnée	Description	Clef
MARCOD	Integer(10)	Code marchandise	**Primaire**
MARLIB	Varchar(15)	Libellé marchandise	
MARMVM	Varchar(10)	Mouvement marchandise	
TYPMAR	Varchar(10)	Type marchandise	
TYPMARLIB	Varchar(20)	Libellé type marchandise	
MARTON	Integer (15)	Tonnage marchandise	
MARNBC	Integer (15)	Nombre de colis	
ESCNUM	Integer (10)	Numéro escale	**Etrangère**

Tableau 6: Le niveau physique de la table « CARGAISON ».

- **CONSIGNATAIRE :** concerne les informations de l'agent consignataire

Nom du champ	Type de donnée	Description	Clef
CONCOD	Integer(10)	Code consignataire	**Primaire**
CONNOM	Varchar(15)	Nom consignataire	
CONLIB	Varchar(15)	Libellé consignataire	
CONADR	Varchar(50)	Adresse consignataire	
CONTEL	Integer (15)	Numéro de téléphone consignataire	
CONVIL	Varchar(15)	Ville	
CONMEL	Varchar(30)	Email	

Tableau 7: Le niveau physique de la table « CONSIGNATAIRE ».

- **PORT :** correspond au port d'arriver ou bien de démarrage

Nom du champ	Type de donnée	Description	Clef
PORCOD	Integer(10)	Code port	**Primaire**
PORNOM	Varchar (25)	Nom du port	
CODSTA	Integer(10)	Code statut	
ESCNUM	Integer(10)	Numéro escale	**Etrangère**

Tableau 8:Le niveau physique de la table «PORT».

- **OPERATION :** correspond à la position du navire c a d c'est à dire est ce qu'il va entrer ou bien il va sortir

Nom du champ	Type de donnée	Description	Clef
OPRCOD	Integer(10)	Numéro opération	**Primaire**
DATOPR	Date	Date opération	
POSOPR	Varchar(15)	Position (entrer, sortie)	

Tableau 9:Le niveau physique de la table « OPERATION ».

- **QUAI :** correspond aux informations du quai

Nom du champ	Type de donnée	Description	Clef
QAICOD	Integer(10)	Code quai	**Primaire**
QAILOG	Integer(15)	Longueur	
QAITIR	Integer(15)	Tiran d'eau	
QAIBAS	Varchar(20)	Bassin du quai	
QAIOBS	Varchar(255)	Observation	
QAIPLP	Integer (20)	Poste multiple	
QAIUTI	Varchar (30)	Utilisation quai	
QAITYP	Varchar (5)	Type quai	
QAILIB	Varchar (60)	Libellé quai	
OPRCOD	Integer (10)	code opération	**Etrangère**

Tableau 10:Le niveau physique de la table «QUAI».

- **PROCES_VERBALE :** correspond aux informations des documents qui contiennent les informations sur les navires

Nom du champ	Type de donnée	Description	Clef
PRVNUM	Integer(10)	Code port	**Primaire**
PRVDAT	Date	Date procès verbale	
ENTR	Varchar (255)	Entreprise	
PRVOBS	Varchar (255)	Observation	

Tableau 11:Le niveau physique de la table «PROCES_VERBALE ».

III.9 Conception de microservice administrateur :

III.9.1 Diagramme de microservice administrateur :

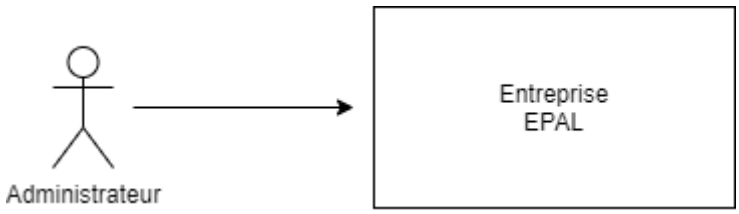


Figure 24:Diagramme de contexte pour le microservice administrateur

III.9.2 Diagramme de cas d'utilisation de l'administrateur :

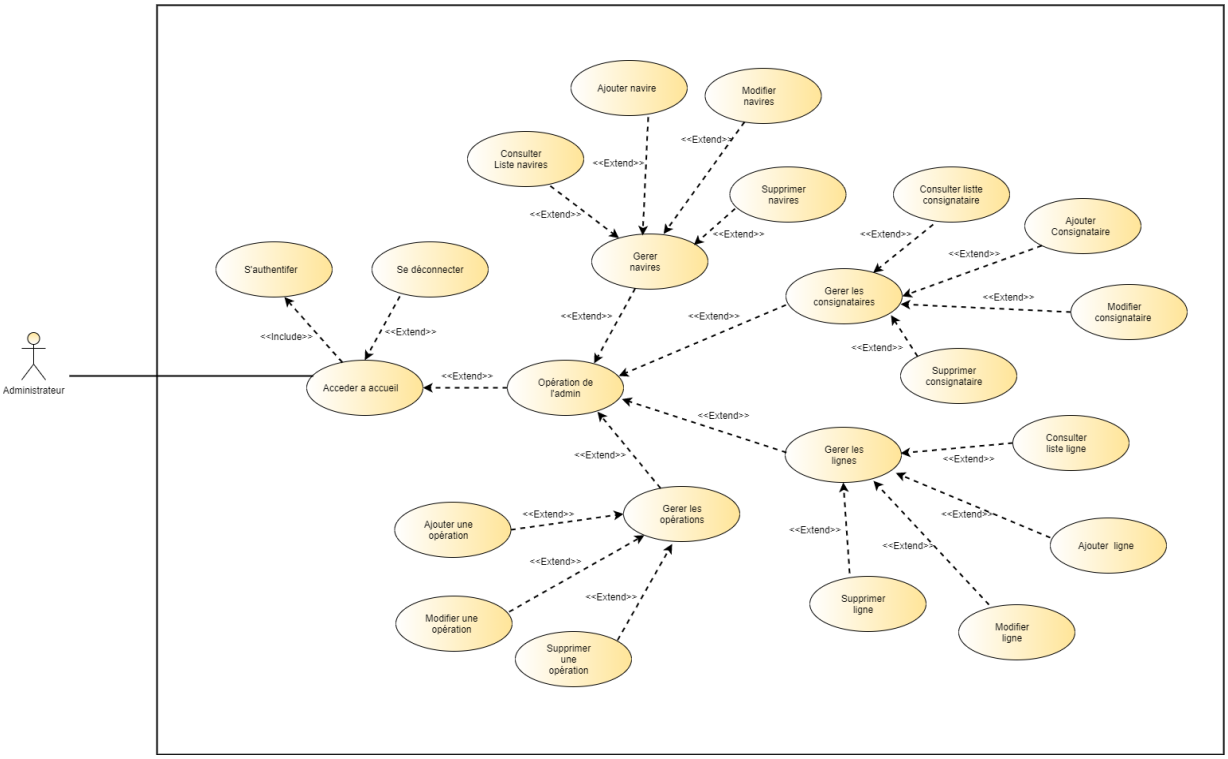


Figure 25 : Diagramme de cas d'utilisation pour administrateur

L'administrateur doit tout d'abord s'authentifier, accéder à l'accueil, il peut ajouter supprimer modifier un navire il peut consulter la liste des navires, il peut aussi ajouter, modifier supprimer un consignataire, et ajouter modifier supprimer une ligne, ajouter, modifier supprimer une opération il peut aussi consulter la liste des navires, la liste des consignataires et la liste des lignes et il peut se déconnecter

III.9.3 Diagramme de séquence :

- **Diagrammes de séquence pour l'authentification de l'administrateur :**

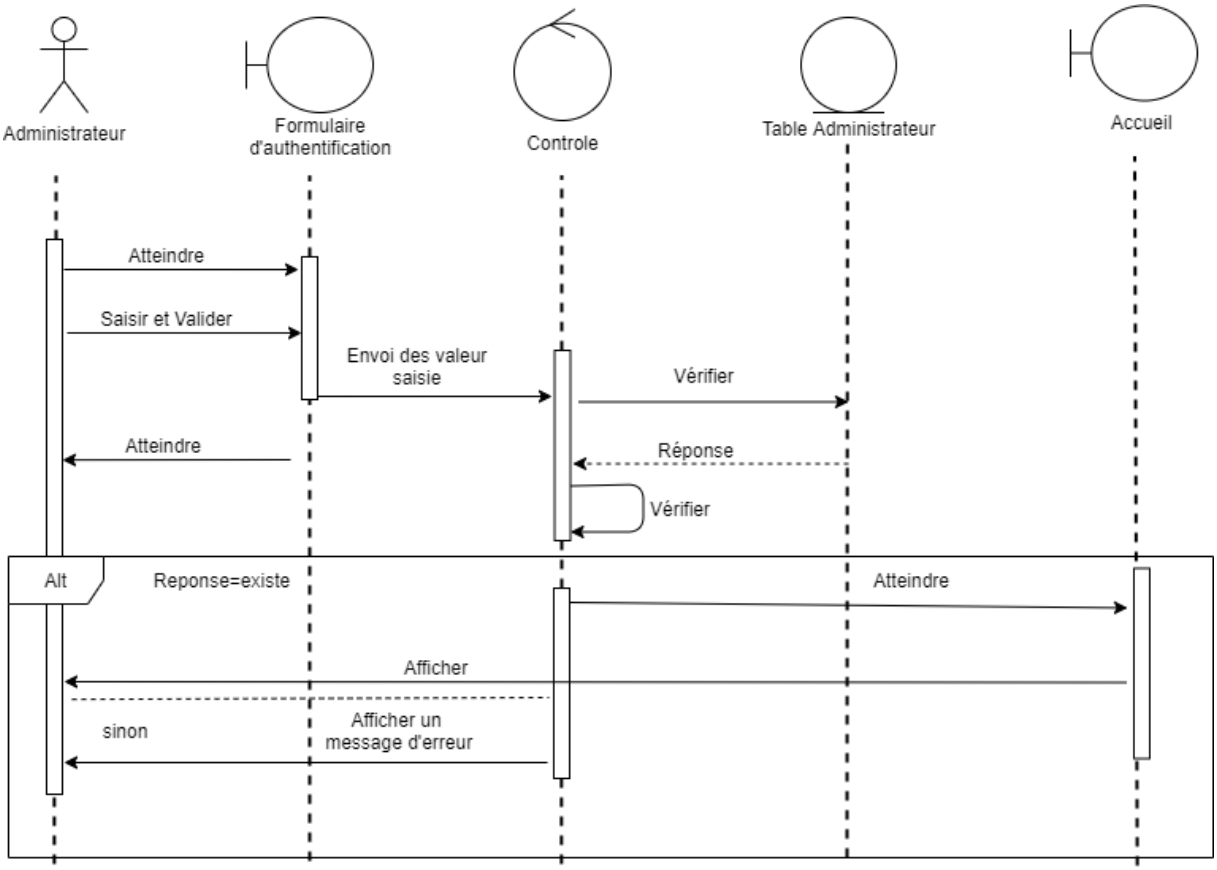


Figure 26:Diagramme de séquence pour le ca s d'utilisation S'authentifier.

1. Le système affiche le formulaire d'authentification.

2. L'administrateur remplit le formulaire et valide.

3. Le système fait les vérifications au niveau de la base de données si l'authentification n'est pas réussie, le système affiche un message d'erreur sinon l'administrateur accède à la page d'accueil.

- **Diagramme de séquence pour le cas d'utilisation ajouter navire :**

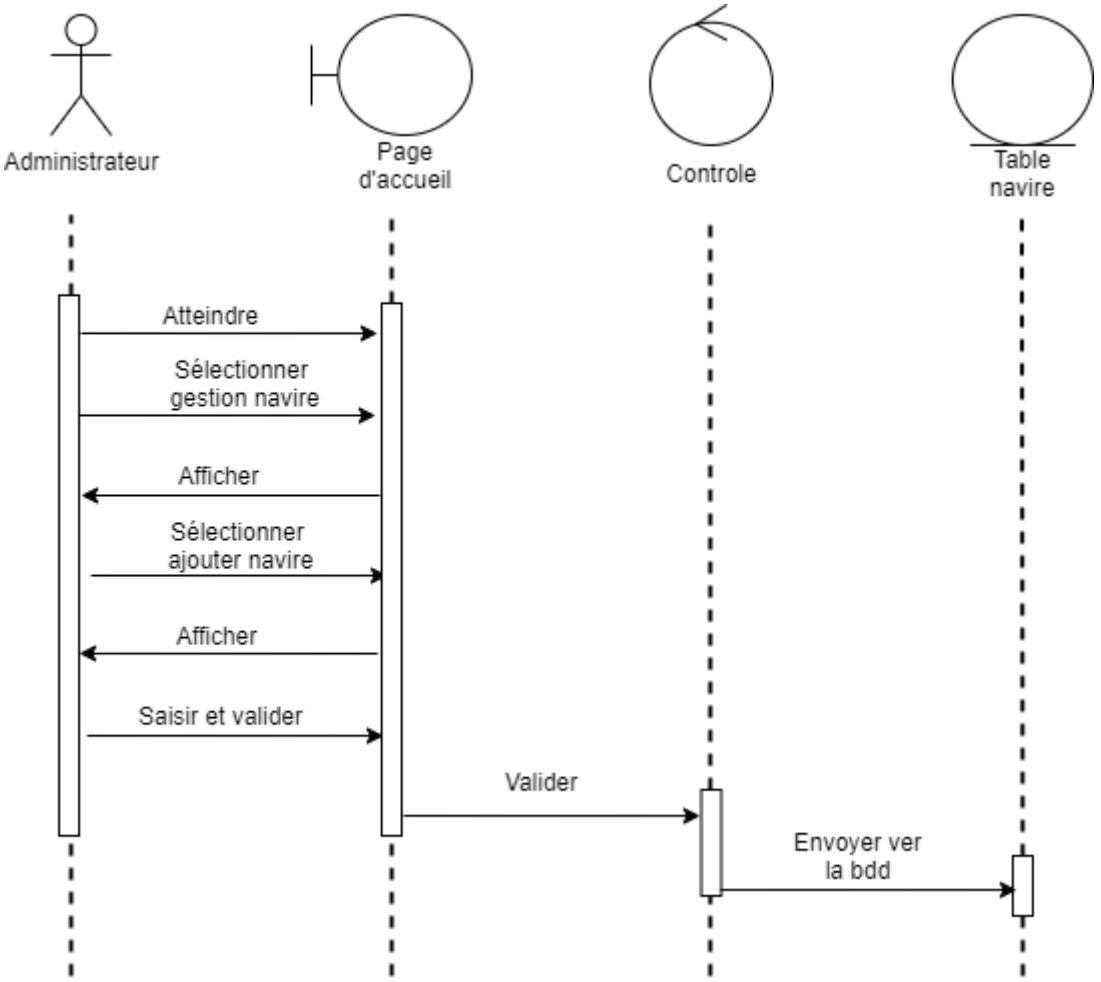


Figure 27 : Diagramme de séquence pour le cas d'utilisation Ajouter navire .

1. L'administrateur accède à la page d'accueil.

2. L'administrateur sélectionne l'option gestion navire.

3. Le système lui affiche une liste d'options et il sélectionne l'option ajouté navire

4. Le système lui affiche un formulaire et l'administrateur le rempli et le valide

5. Le système enregistre les informations au niveau de la base de données

III.9.4 Diagrammes de classe de microservice administrateur :

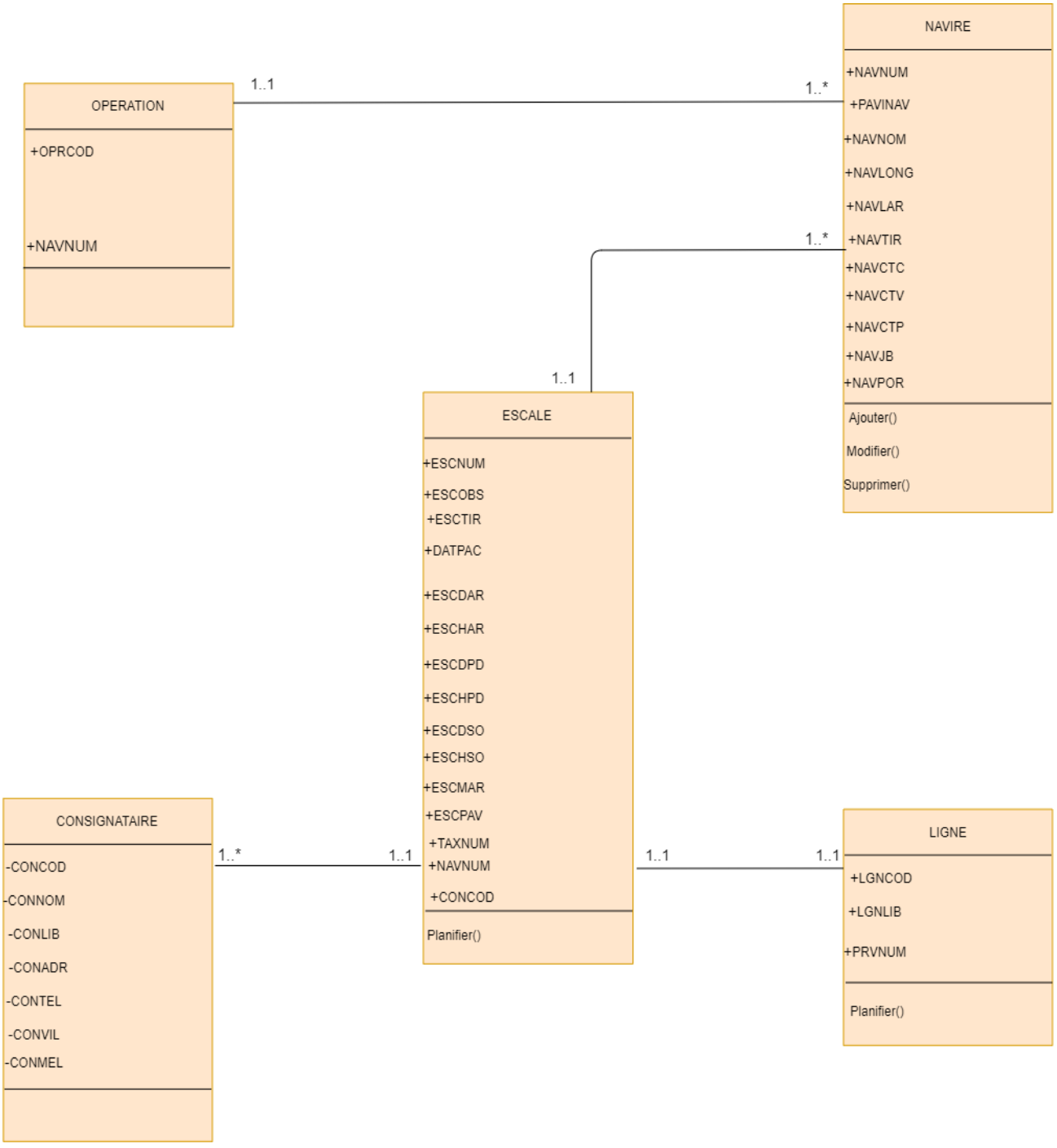


Figure 28:Diagramme de classe pour administrateur

III.9.5 Conception de la base de données :

- **Règles de gestion :**
  - Une escale peut avoir un a plusieurs navires.
  - Un navire concerne une et une seule escale.
  - Une escale peut être planifier par un a plusieurs consignataires.
  - Un consignataire planifie une et une seule escale.
  - Une escale suit une et une seule ligne.
  - Une ligne concerne une et une seule escale.
  - Un navire peut effectuer une et une seule opération.
  - Une opération concerne plusieurs navires.

- **Le niveau logique de la base de données :**

- **ESCALE (ESCNUM**, ESCDAR, ESCHAR, ESCDPD, ESCHPD, ESCDSO, ESCHSO, ESCMAR, DATPAC, ESCTIR, ESCOBS, ESCPAV, *CONCOD, *NAVNUM**)**
- **NAVIRE (NAVNUM**, NAVNOM, NAVLONG, NAVLARG, NAVTIR, NAVCTC, NAVCTV, NAVCTP, NAVJB, NAVPOR**)**
- **CONSIGNATAIRE (CONCOD**, CONNOM, CONLIB, CONADR, CONTEL, CONVIL, CONMEL**)**
- **OPERATION (OPRCOD**, DATOPR, POSOPR**)**
- **LIGNE (LGNCOD**, LGNLIB, * OPRCOD**)**

- **Le niveau physique de la base de données :**

✓ **ESCALE :** correspond à un voyage.

Nom du champ	Type de donnée	Description	Clef
ESCNUM	Integer(10)	Numéro de l'escale	**Primaire**
ESCDAR	Date	Date arriver	
ESCHAR	Time	Heure arriver	
ESCDPD	Date	Date prévue départ	
ESCHPD	Time	Heure prévue départ	
ESCDSO	Date	Date de sortie	
ESCHSO	Time	Heure de sortie	
ESCMAR	Varchar(255)	Marchandises	
DATPAC	Date	Date prévue d'accostage	
ESCTIR	Integer(10)	Tiran d'eau	
ESCOBS	Varchar(50)	Observation	
ESCPAV	Varchar(15)	Pavillon	
CONCOD	Varchar(15)	Code consignataire	**Etrangère**
NAVNUM	Varchar(15)	Numéro navire	**Etrangère**

Tableau 12: Le niveau physique de la table ESCALE

✓ **NAVIRE :** correspond aux informations d'un navire.

Nom du champ	Type de donnée	Description	Clef
NAVNUM	Integer(10)	Numéro navire	**Primaire**
NAVNOM	Varchar(15)	Nom navire	
NAVLONG	Integer(10)	Longueur	
NAVLARG	Integer(10)	Largeur	
NAVTIR	Integer(10)	Tiran d'eau	
NAVCTC	Integer (15)	Capacité de transport conteneur	
PAVINAV	Varchar(15)	Pavillon	
NAVCTV	Integer (15)	Capacité de transport véhicule	
NAVCTP	Integer (15)	Capacité de transport passagers	
NAVJB	Integer(10)	Jauge brute	
NAVPOR	Varchar(15)	Port d'attachement	

✓ CONSIGNATAIRE **:** concerne les informations de l'agent consignataire.

Nom du champ	Type de donnée	Description	Clef
CONCOD	Integer(10)	Code consignataire	**Primaire**
CONNOM	Varchar(15)	Nom consignataire	
CONLIB	Varchar(15)	Libellé consignataire	
CONADR	Varchar(50)	Adresse consignataire	
CONTEL	Integer (15)	Numéro de téléphone consignataire	
CONVIL	Varchar(15)	Ville	
CONMEL	Varchar(30)	Email	

Tableau 13:Le niveau physique de la table consignataire.

- ✓ **OPERATION :** correspond à la position du navire c a d c'est à dire est ce qu'il va entrer ou bien il va sortir.

Nom du champ	Type de donnée	Description	Clef
OPRCOD	Integer(10)	Numéro opération	**Primaire**
DATOPR	Date	Date opération	
POSOPR	Varchar(15)	Position (entrer, sortie)	

Tableau 14 : Le niveau physique de la table OPERATION.

- ✓ **LIGNE :** correspond à la ligne suivie par le navire de son démarrage jusqu'à son arrivée.

Nom du champ	Type de donnée	Description	Clef
LGNCOD	Integer(10)	Code ligne	**Primaire**
LGNLIB	Varchar (25)	Libellé ligne	
OPRCOD	Integer (10)	code opération	**Etrangère**

Tableau 15 : Le niveau physique de la table LIGNE.

III.10 Conclusion :

Ce chapitre est consacré à la conception de notre architecture en microservices. Nous avons présenté les différents diagrammes UML qui illustrent le comportement de notre système, après avoir décomposé l'architecture EPAL en différents microservices et spécifier les besoins fonctionnels et non fonctionnels en spécifiant les besoins

Le chapitre suivant sera réservé à la description de la solution proposée. Notre réalisation se base sur le développement du microservice (Escale).

Dans un premier lieu nous définirons les différentes technologies à utiliser dans notre réalisation que ce soit pour le développement ou bien pour le déploiement. Ensuite nous entamerons la réalisation et nous présenterons les interfaces de notre microservice. Finirons par la définition des outils utilisés.

ChapitreIV. Réalisation

IV.1 Introduction :

Après avoir décomposé l'application d'EPAL qui est GestPort en petit microservice, et avoir présenté les différents diagrammes UML. Nous passerons à la phase de réalisation qui se focalise sur le développement du microservice escale.

Premièrement nous commencerons par la présentation des différents outils a pour le développement et le déploiement. Deuxièmement nous présentons les interfaces de notre application et le déploiement de notre microservice.

IV.2 Choix de technologie :

Nous présentons dans cette partie les technologies à utiliser dans le développement de notre application.

IV.2.1 Choix des Framework de back-end :

Chaque service de l'architecture microservices peut être développée avec une technologie différente et un langage diffèrent, par exemple utiliser un Framework laravel et symfony pour langage PHP et Framework Spring boot, hiberner pour langage java.

Nous avons décidé de développer le microservice avec Spring boot parce qu'il est concédé un bon point pour mettre en œuvre l'architecture microservice.

- **Spring Boot :**

Spring boot c'est un Framework Open Source basé sur java, développé par Pivota Team.il permet de simplifier le démarrage et le développement des nouvelles applications en réduisant la complexité de configuration[20]. Il permet de créé rapidement des applications très puissantes et riches en fonctionnalités .il offre la possibilité de créer deux types d'applications :

- Java archive, correspondant à une application dont le paquetage est d'extension *.jar*.

- Web archive, correspondant à une application dont le paquetage est d'extension *.war*.

Nous avons choisi Spring boot pour développer le back end de microservice en raison des fonctionnalités et des avantages qu'il offre, tels que

- Il offre un moyen flexible de configurer les Java Beans, les configurations XML et les transactions de base de données.
- Il fournit un traitement par lots puissants et gère les points de terminaison REST.
- Dans Spring Boot, tout est configuré automatiquement ; aucune configuration manuelle n'est nécessaire.
- Il offre une application de ressort basée sur l'annotation
- Facilite la gestion des dépendances
- Il comprend un conteneur de servlet intégré

Spring boot présente de nombreux avantages compétitifs, en effet, il offre une gamme riche de choix au niveau du serveur d'applications utilisé, des implémentations REST, des outils de manipulation des JSON, des outils de logging et plusieurs autres intégrations.

IV.2.2 Choix de Framework de front-end :

Après avoir choisi le Framework, Spring boot pour le développement de backend nous devons maintenant choisir un Framework pour le front end de notre application.

- **Angular :**

C'est un framework JavaScript open source conçu pour aider les développeurs à créer des applications modernes. Ce Framework a été développé autour de trois concepts, testabilité, modularité et maintenabilité.

Nous avons choisi angular comme Framework pour développer le front end de notre application par ce que :

- Il facilite et simplifie le travail.
- Il se base sur le concept single page pour éviter la création de plusieurs pages avec plusieurs extensions (.css, .html, etc.).

IV.3 Approche de développement :

L'approche DevOps se concentre sur l'automatisation de plusieurs activités des équipes opérationnelles telles que l'intégration, le déploiement des applications et la surveillance des environnements de production (monitoring).

IV.4 Outils de déploiement :

Afin de déployer le microservice, nous avons opté pour un déploiement continu qui est l'un des concepts de cycle de vie de DevOps définis dans le premier chapitre.
Les outils de conteneurisation[2] jouent également un rôle essentiel dans la phase de déploiement continu de cycle de vie de DevOps. **Docker** est un outil populaire qui est utilisé dans le cycle de vie. Il assure la cohérence entre les environnements de développement, la préparation de test la préparation de production.
Docker assure la sécurité, l'évolutivité et la simplicité qu'il apporte au cycle logiciel, il est incontournable dans tout environnement DevOps.

IV.4.1 Docker :

Docker nous permet de travailler dans des environnements isolés appelés conteneurs, capables de communiquer entre eux et pouvant être déployés facilement sur nos serveurs.

Dans cas de conteneurs docker nous disposons d'un système d'exploitation et des ressources qui sont partagées entre les conteneurs.

Docker est une plateforme logicielle, destinée aux développeurs et administrateurs systèmes. Dans le but de faciliter le développement, la diffusion et le déploiement

[2]**La conteneurisation** est un type de virtualisation au niveau de l'application, qui permet de créer plusieurs instances d'espace utilisateur isolées sur un même noyau. Ces instances sont appelées conteneurs.

d'applications. Il repose sur le noyau linux et ses fonctionnalités de virtualisation par conteneurs, il s'appuie notamment sur :

- Cgroups qui est un composant pour Contrôller et limiter l'utilisation des ressources pour un ou plusieurs processus (utilisation de la RAM CPU entre autres).
- Espace de noms (namespaces) permet de créer des environnements sécurisés d'une manière à isoler les conteneurs.

La figure ci-dessous donne l'architecture de docker

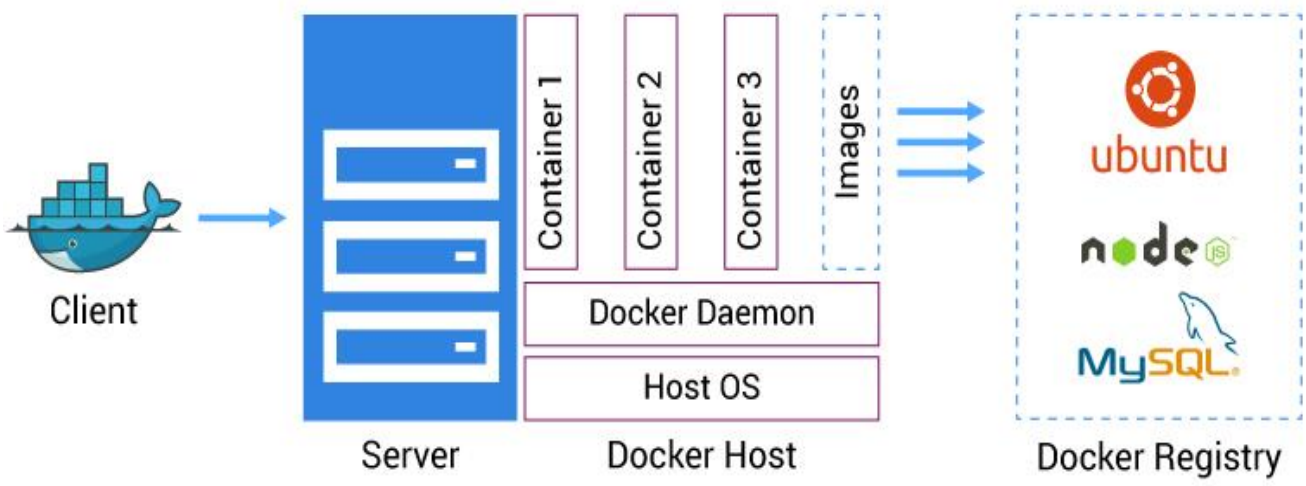


Figure 29: L'architecture docker.

Client docker créée une image docker et une commande de construction est émise vers le démon docker, ce dernier construira alors une image en fonction de nos entrées et l'enregistrera dans register (docker Hub[3]).

[3] Docker Hub C'est un dépôt en ligne où les images Docker (qu'elles soient officielles ou non) peuvent être publiées et utilisées par les utilisateurs de la communauté Docker. Docker Hub rassemble à la fois des dépôts publics ainsi que des dépôts privés

➢ Commande docker :

La figure suivante représente les différentes commandes de docker :

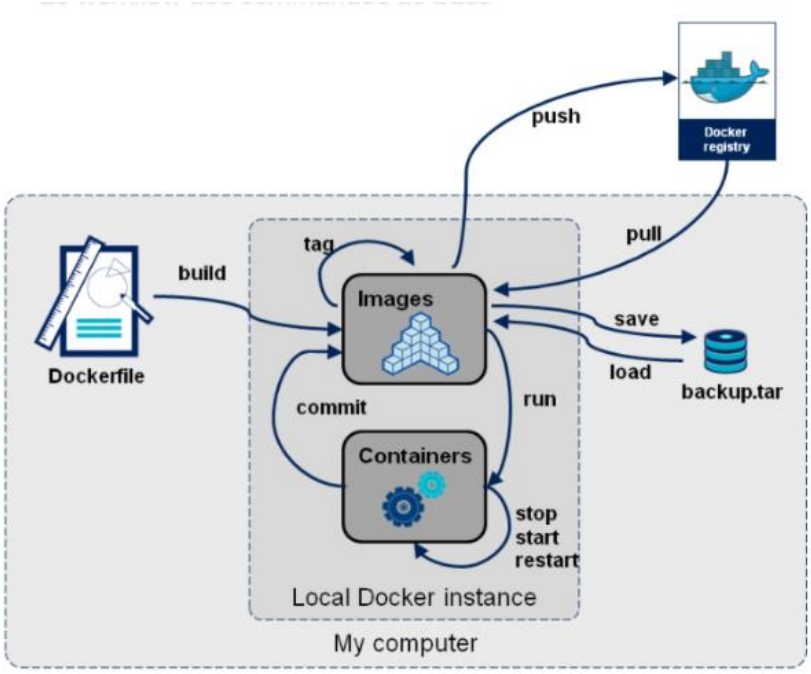


Figure 30 : Commande Docker.

- Création du fichier dockerfile et exécuter build pour construire une image docker.
- Publier l'image docker dans docker register avec push (docker hub).
- Récupérer l'image docker existée dans docker avec pull.
- Exécuter l'image docker en utilisant run.
- Sauvegarder des images docker,en utilisant la commande docker save qui produira une archive tar qui pourra être utilisée.
- Pour la restauration de ces containers nous utilisons la commande suivante : « **docker load ».**

➢ Les étapes de conteneurisation des applications dans docker :

On considère l'application spring boot et la base de données MySQL comme deux services différents. Pour que l'application et la base de données soient déployées on aura besoin de deux conteneurs docker. Ceux-ci devront fonctionner sur le même réseau pour pouvoir communiquer entre eux.

Nous présentons quelques étapes et commandes de conteneurisation de l'application et sa base de données

- **Conteneurisation de la base de données MYSQL :**
  - Création du réseau docker.
  - Utilisation de l'image fournie par docker hub pour exécuter en tant que conteneur en utilisant la commande suivante :

```
docker  container run --name mysqldb --network capitainerie-e MYSQL_ROOT
_PASSWORD = root -e MYSQL_DATABASE = capitaineriedb -d mysql: 8
```

  - Nous modifions l'application.properties dans l'application Spring Boot pour utiliser le nom du conteneur mysqlcapitaineriedb au lieu de localhost.

- **Déploiement de l'application spring boot :**
  - Construire des images docker à l'aide d'un fichier docker (docker file).
  - Cree un fichier image docker en utilisant la commande ci-dessous :

docker build –t <IMAGE_NAME>

  - Nous créerons un fichier docker-compose.yml pour définir les services qui composent l'application, et compléter/surcharger certaines des propriétés définies dans leurs docker files respectives.
    - ✓ Définir également les liens entre les services.
    - ✓ Pour chaque service on crée un lien vers le dossier dans lequel se trouve son dockerfile.
    - ✓ Lancer toute l'application avec docker-compose up.

Afin de gérer l'automatisation des tâches telles qu'approvisionnement, déploiement, la configuration et la planification et la sécurité des interactions entre les conteneurs …etc. On utilise l'orchestration des conteneurs.

Parmi les outils d'orchestrations des conteneurs les plus populaires : **« kubernetes ».**

IV.4.2 Kubernetes :

Kubernetes, k8s ou encore kube est l'orchestrateur du déploiement le plus populaire pour la gestion des conteneurs sur les clusters de serveurs, utilisés souvent avec Docker créé par l'équipe de Google qui l'ont ensuite donnée à Cloud Native Computing Fondation (CNCF) en devenant ainsi un outil open source[21]. En d'autres termes, Kubernetes permet de prendre en charge plusieurs kernels (noyaux), donc de pouvoir gérer les conteneurs sur différents serveurs hôtes, qu'ils soient physiques ou virtuels ou alors situés dans le Cloud publique, privés ou hybrides. Les fonctionnalités d'orchestration de Kubernetes permettent de créer des services applicatifs que ce soit Front-End ou Back-End sur plusieurs conteneurs, planifier l'exécution de ces conteneurs dans un cluster, garantir leur intégrité au fil du temps et assurer leur monitoring.

Avec Kubernetes, le développeur n'a plus à s'occuper de la gestion des machines virtuelles (VM), il a à disposer directement son environnement d'exécution qui est le conteneur pour pouvoir y déployer son code (il s'occupe uniquement des couches Application et Contexte d'exécution, il n'a pas besoin de savoir où sont les applications), c'est-à-dire, c'est à Kubernetes de s'occuper des couches infrastructures sous-jacentes (OS/ Stockage/ Réseau).

Kubernetes repose sur une architecture de cluster, généralement déployée sur plusieurs nœuds (nodes) (Kubernetes supporte des clusters de 1 à 5000 nœuds). Cette architecture est définie par deux types de nœuds, en suivant l'architecture maitre-esclaves :

- Un nœud maitre (master) : charger d'orchestrer le cluster (un serveur qui contrôle les nœuds). Il gère le cluster et constitue le point d'entrée pour toutes les tâches administratives.
- Des nœuds esclaves (nœuds) : ce sont des nœuds de calcul qui correspondent à des hôtes Docker. En d'autres termes des machines qui hébergent les hôtes Docker.

Les hôtes Docker exécutent des tâches assignées au sein des nœuds qui font tourner des Pods[4].

Le master gère l'utilisation des ressources sur chaque nœud, afin d'assurer que la charge du travail n'est pas en excès par rapport aux ressources disponibles. Pour ce faire, Kubernetes doit connaitre les ressources disponibles et celles actuellement assignées sur les serveurs en faisant appel à des Kubelets[5].Si un nœud tombe en panne, Kubelet le signale au master qui va vérifier à son tour le nombre de copies identiques demandé d'un pod et qui doivent s'exécuter dans le cluster (la gestion de la résilience des pods).

Kubernetes fournit un service de routage en assignant une adresse IP et un nom de domaine à un service pour équilibrer la charge du trafic vers différents pods. Les requêtes du service sont transformées par Kubernetes vers des pods du service, qui correspondent à des tiers applicatifs, c'est-à-dire Front-End avec le serveur Web ou Back-End avec la BDD.

- **Fonctionnement du Kubernetes :**

Kubernetes s'exécute au-dessus de l'OS et interagit avec les pods des conteneurs qui s'exécutent sur les nœuds. Le master Kubernetes reçoit les commandes de la part d'un administrateur ou d'une équipe DevOps et relie ces instructions aux nodes. Ce système de transfert fonctionne avec des services, le node le plus adapté pour cette tache va être choisi automatiquement. Il va ensuite allouer les ressources aux pods désignés dans ce nœud pour qu'ils effectuent la tâche requise. Lorsque le master planifie un pod dans un nœud, le Kubelet de ce nœud ordonne à Docker de lancer les conteneurs spécifiés (démarrer ou arrêter ce conteneur). Le Kublet collecte ensuite en continuant le statut de ces conteneurs via Docker et rassemble ces informations sur le serveur Master.

[4] Pods : sont un groupe d'un ou plusieurs conteneurs étroitement lié (environnement d'exécution d'un ou plusieurs Docker) qui partagent les mêmes ressources (le master va définir quel nœud va faire tourner un pod selon la disponibilité des ressources).

[5] Le Kubelet : est un composant exécuté sur des nœuds qui s'assure que les conteneurs définis ont démarré et fonctionnent comme prévu lors de leurs conceptions.

La figure suivante représente les composants de Kubernetes :

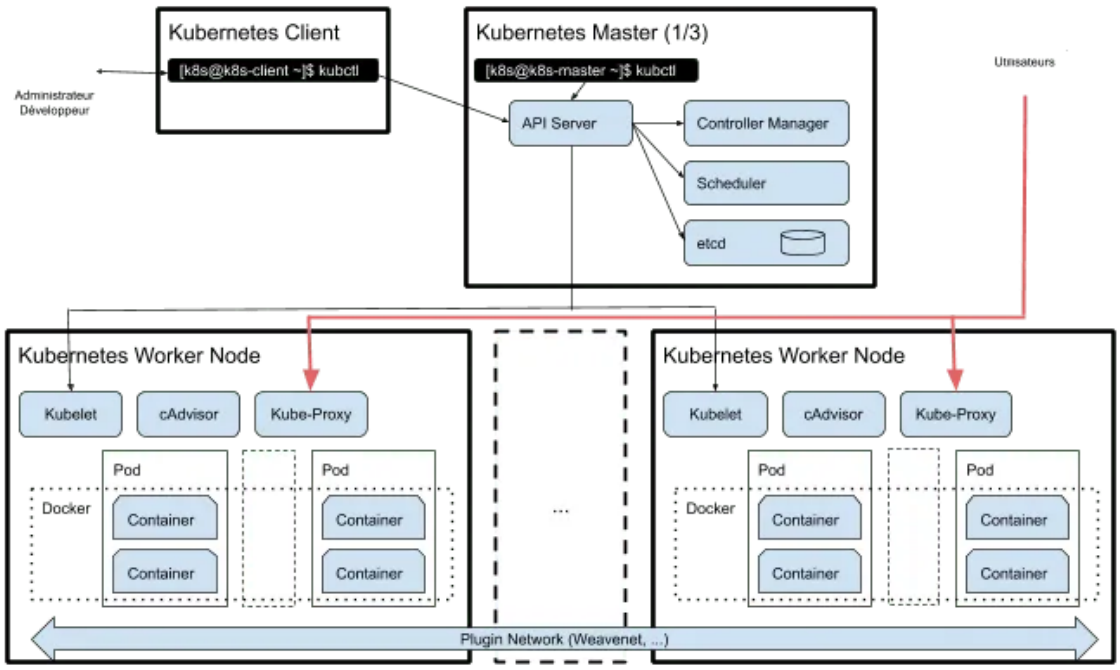


Figure 31: les composants de Kubernetes.

IV.5 Environnement de développement :

IV.5.1 Environnement physique :

La configuration de deux machines utilisées dans notre implémentation suit :

Machine 01 :

- ✓ Un pc portable Lenovo i3 CPU 2.3.0
- ✓ Ram de taille 8.00GO
- ✓ Disque dure de taille 500GO
- ✓ Nom de l'hôte : DESKTOP-M618O2G
- ✓ Nom du système d'exploitation : Microsoft Windows 10 Professionnel
- ✓ Version du système : 10.0.19041 N/A version 19041

Machine 02 :

- ✓ Un pc portable Dell i7-2640M CPU 2.8.0
- ✓ Ram de taille 8.00GO
- ✓ Disque dure de taille 500GO
- ✓ Nom de l'hôte : DESKTOP-J9LGP13
- ✓ Nom du système d'exploitation : Microsoft Windows 10 Professionnel

IV.5.2 Environnement virtuel :

➢ Oracle VM Virtual Box :

C'est un logiciel libre de virtualisation publié par Oracle. Oracle VM

VirtualBox permet la création d'un ou de plusieurs ordinateurs virtuels dans lesquels s'installent d'autres systèmes d'exploitation[22]. Il fonctionne sous Windows mais aussi sur Mac, Linux …

Dans notre cas on a utilisé la version 6.0.12

IV.5.3 Environnement logiciel :

➢ Visual Studio Code :

C'est un éditeur de code open source édité par Microsoft [23]. Cet outil destiné aux développeurs supporte plusieurs dizaines de langages de programmation comme le HTML, C++, PHP, Javascript, CSS, etc.

Notre réalisation est faite sur visuel Studio code de version 1.49.2.

- **Eclipse ide java Entreprise Edition :**

Eclipse est un IDE, *Integrated Development Environment* (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation [24]. La version standard d'Eclipse inclut l'IDE de développement de Java, ainsi que ces outils de développement (JDT) qui sont des plugins [6]nécessaires pour la programmation.

- **Xampp :**

C'est un ensemble de logiciels permettant de mettre en place un serveur Web local et un serveur FTP et un serveur de messagerie électronique. Simple d'utilisation, il est à la portée d'un grand nombre de personnes puisqu'il ne demande aucune connaissance particulière [25].

- **Draw.io :**

C'est un logiciel gratuit de diagrammes en ligne[26]. Il est utilisé pour des organigrammes, des diagrammes de processus, des organigrammes, des diagrammes UML, ER et de réseau.

IV.5.4 Les langages utilisés :

- **Java :**

C'est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems[27].

- **HTML :**

(Hyper Text Mark_up Language) est le langage de base pour créer un site web, il est inspiré du XML et qui repose sur le principe de balises imbriquées[28], il est simple ; compatible ; mais limité, ainsi on fait appel à un autre langage pour le compléter qui est le CSS.

- **CSS :**

(Cascading Style Sheet) c'est un langage servant à décrire la présentation et le style d'un document HTML, en d'autre termes il sert à définir le design d'un site web[29].

- **JavaScript :**

C’est un langage de programmation, c’est une forme de code qui permet de dicter à l’ordinateur quoi faire. Le code JavaScript est lu et exécuté sur le navigateur web, donc sur la machine de l’internaute. C’est ce qu’on appelle du code côté client [30].

- **Bootstrap :**

C’est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript[31]

- **jQuery :**

C’est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML Le but de la bibliothèque étant le parcours et la modification du DOM , elle contient de nombreuses fonctionnalités ; notamment des animations, la manipulation des feuilles de style en cascade (accessibilité des classes et attributs), la gestion des évènements, etc. L'utilisation d'Ajax est facilitée et de nombreux plugins sont présents[32]

IV.6 Présentation des interfaces :

Dans cette phase nous allons montrer les différentes interfaces de notre application.

IV.6.1 L’interface « prévision d’escale » :

La figure suivante représente l’interface de prévision d’escale :

L'utilisateur cliqué sur le menu « planification » puis il sélectionne l'option « prévision d'escale», il remplit le formulaire affiché et il clique sur « enregistrer ».

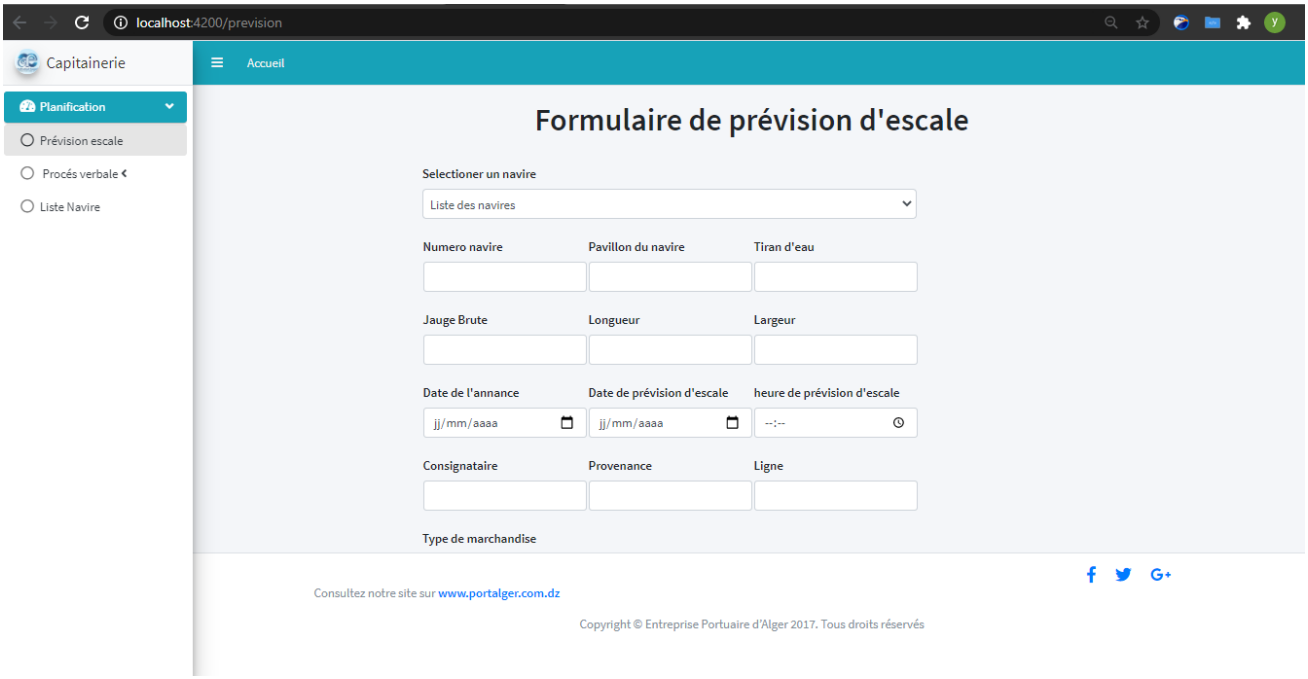


Figure 32 : Interface de prévision d'escale.

IV.6.2 L'interface procès verbal:

L'utilisateur cliqué sur le menu « planification » puis il sélectionne l'option « procès-verbal», il remplit le formulaire affiché et il clique sur « enregistrer ».

La figure ci-dessous montre l'interface proces verbale :

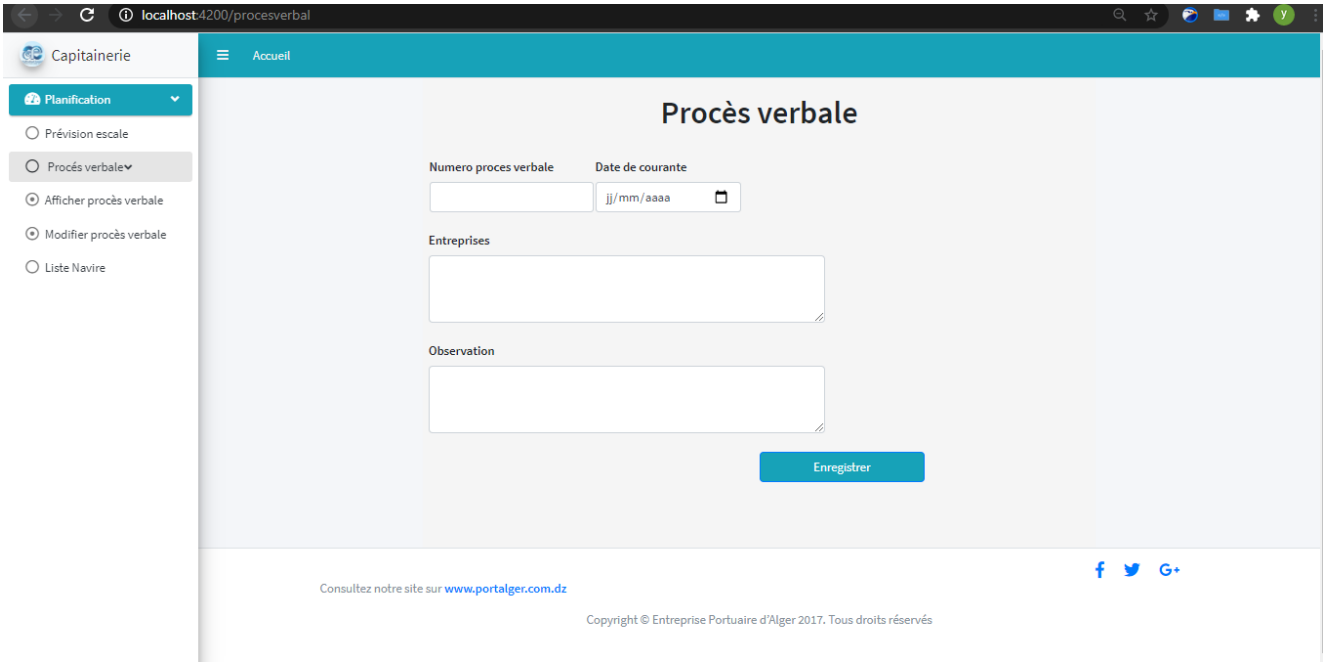


Figure 33: Interface procès verbale.

La figure ci-dessous montre l interface de consultation de la liste des navires

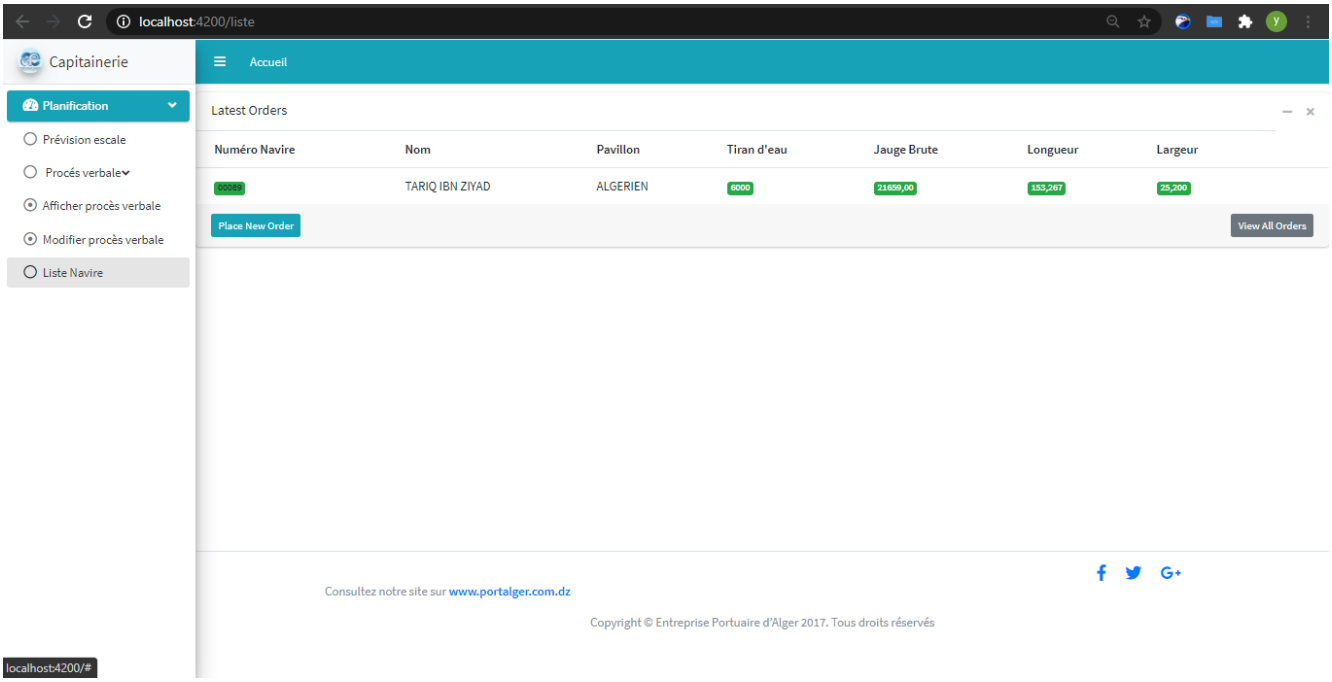


Figure 34: Interface liste navires.

IV.7 Les étapes de configurations et d'installation de docker :

Les étapes de configuration et d'installations de Docker se présentent comme suit :

- La configuration de la carte réseau de la machine virtuelle avec le mode d'accès « réseau prive hôte » pour qu'on puisse accéder à la machine de l'extérieur.
- Nous installons SSH sur la machine virtuelle pour l'utiliser dans la communication entre la machine hôte et la machine virtuelle. La figure ci-dessous présente la communication entre les deux machines.

Figure 35 : Interface Communication entre deux machines.

- Installation de docker avec la commande suivante :

Sudo dh get dokcer.sh

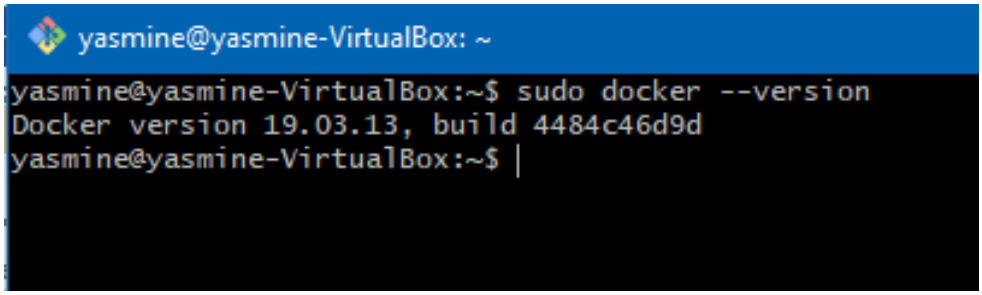


Figure 36 : Interface version Docker utilisée.

Pour pouvoir manipuler un docker à partir de la machine host nous avons besoin d'utiliser un client docker.

Pour que docker client puisse communiquer à distante docker « engine » nous exposons service docker à travers une « API rest » comme suit :

1. éditer le fichier « docker. Service » existé dans le dossier lib/systemd/system avec l'éditeur de texte « nano » :

```
8b03f1e11359: Pull complete
Digest: sha256:6b1daa9462046581ac15be20277a7c75476283f969cb3a61c8725ec38d3b01c3
Status: Downloaded newer image for nginx:latest
62cbd969482d2213147cfe0069551045725d0ef8b460540d635edcfab901f7d5
yasmine@yasmine-VirtualBox:~$  sudo nano /lib/systemd/system/docker.service
```

Figure 37 : Interface de la commande d'édition de fichier Docker.service.

2. ajouter les paramètres suivants **–H=tcp://0.0.0.0 :2375** dans « execStart »

```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H=tcp://0.0.0.0:2375 --containerd=/run/containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always

# Note that StartLimit* options were moved from "Service" to "Unit" in systemd 229.
# Both the old, and new location are accepted by systemd 229 and up, so using the old location
# to make them work for either version of systemd.
StartLimitBurst=3
```

Figure 38 : Interface de fichier Docker.service

3. Redémarrer le service docker on utilisent les commandes :

- **Systemctl daemon-reload**

  Ensuite nous exécutant la commande :

- **Sudo service docker restart**

IV.8 Les étapes de configurations et l'installation de kubernetes :

1. L'installation de l'outil kubectl sur notre machine physique pour interagir avec l'API de kubernetes Master.

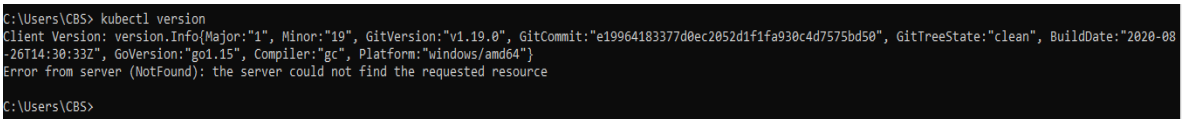


Figure 39 : Interface version kubectl.

2. L'installation de minikube sur notre machine physique avec la commande suivante : **Choco install minikube**

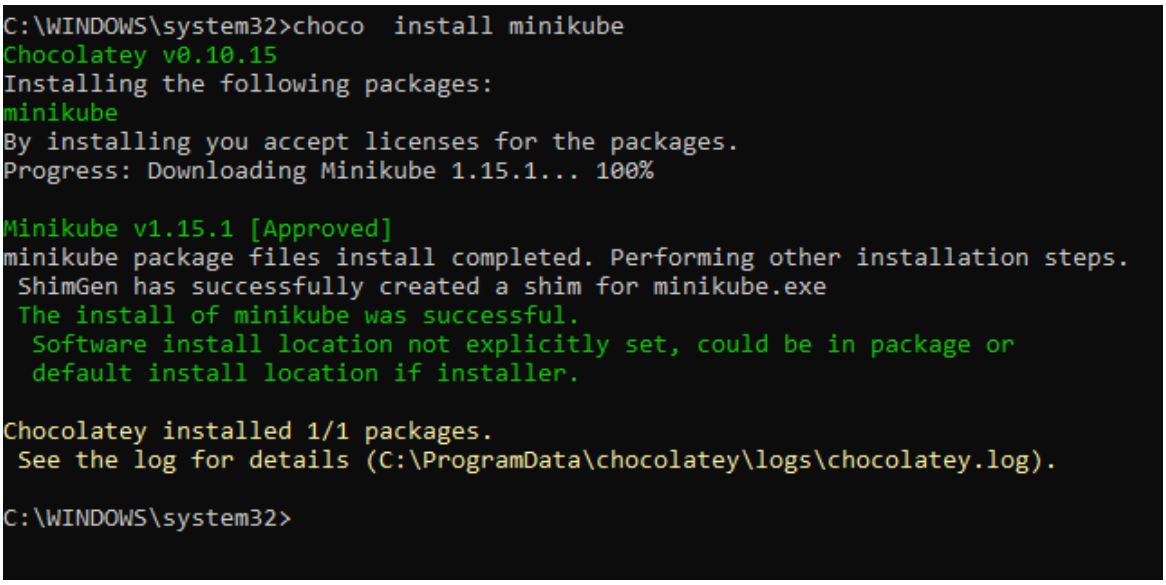


Figure 40: Interface de l'installation de minikube.

3. Démarrer minikube avec la commande : **minikube start**

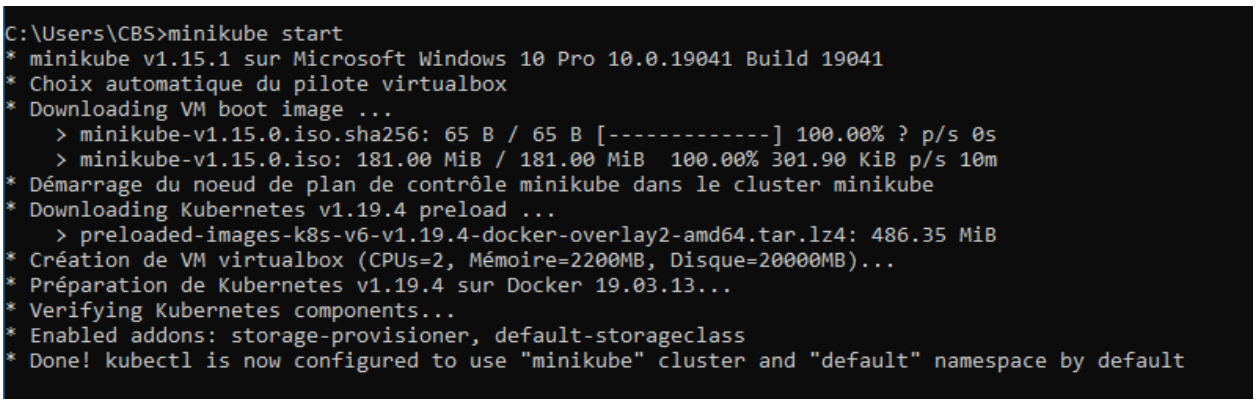


Figure 41 : Interface de démarrage de minikube.

Pour connecter docker clients à docker engine fournie par minikube en modifiant les variables d'environnement pour quelles pointes vers le host de la machine virtuelle minikube.

La commande docker-env retourne les variables d'environnement nécessaire à configurer l'environnement local pour utiliser le docker daemon à l'intérieur de minikube.

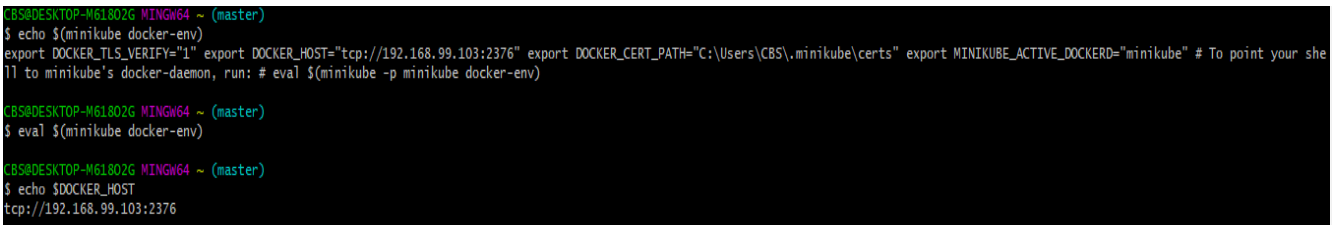


Figure 42 : Interface de communication de Docker client et Docker Engine de minikube.

IV.9 Les étapes de déploiement sur docker et kubernetes :

1. Création de docker file pour notre application (font-end) développée avec Angular.
2. faire un build pour créer l'image docker .
3. Exécuter l'image avec la commande run :

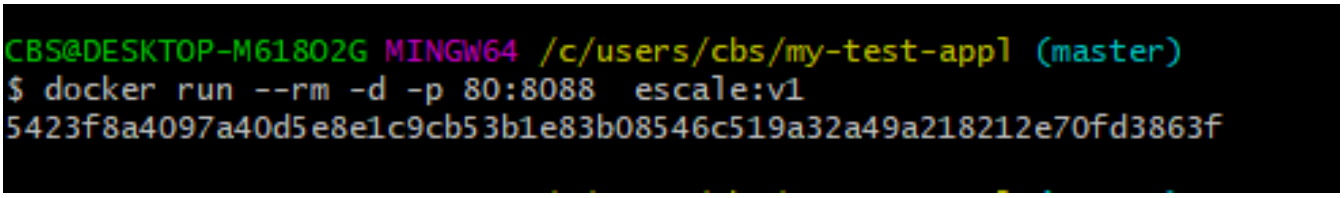


Figure 43: Interface de la commande de l'exécution de Dockerfile.

La figure suivante représente l'interface exécutée sur le port 8088 :

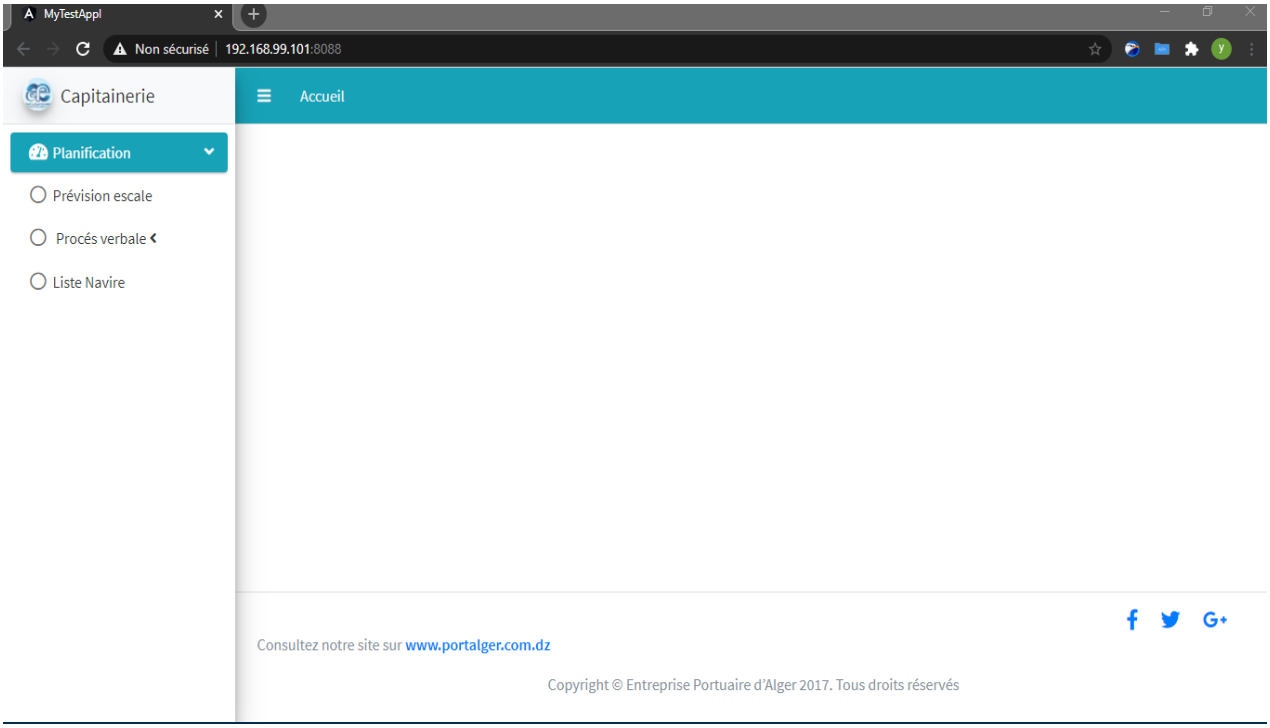


Figure 44 : Interface de la page d'accueil.

4. Création d'un fichier deployment.yml et l'exécuter avec la commande suivante :

Kubectl apply –f deployment.yaml

5. Création du fichier service .yml et l'exécuter avec la commande suivante :

Kubectl apply –f service.yaml

La figure suivante montre la création du service avec succès :

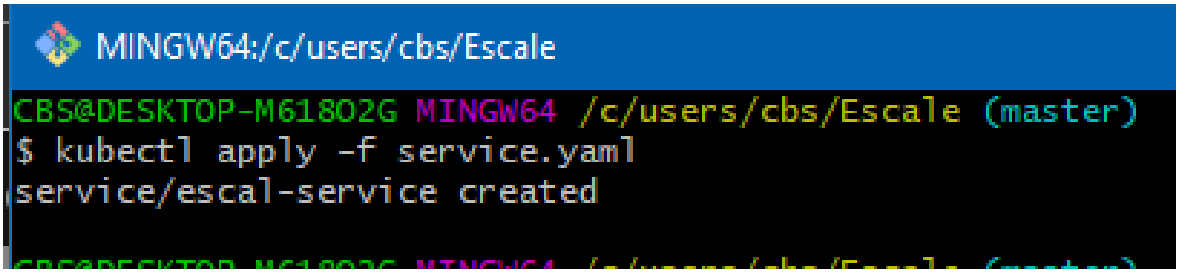


Figure 45:Interface de création de service.

Pour confirmer le déploiement de conteneur Docker on exécute la commande **:**

minikube service escal-service

```
CBS@DESKTOP-M618O2G MINGW64 /c/users/cbs/Escale (master)
$ minikube  service escal-service
|-----------|---------------|-------------|---------------------------|
| NAMESPACE |     NAME      | TARGET PORT |            URL            |
|-----------|---------------|-------------|---------------------------|
| default   | escal-service |          80 | http://192.168.99.101:32000 |
|-----------|---------------|-------------|---------------------------|
* Opening service default/escal-service in default browser...
```

Figure 46: Interface de déploiement de conteneur Docker.

La figure suivante représente l'affichage de l'interface prévision d'escale sur le port 32000

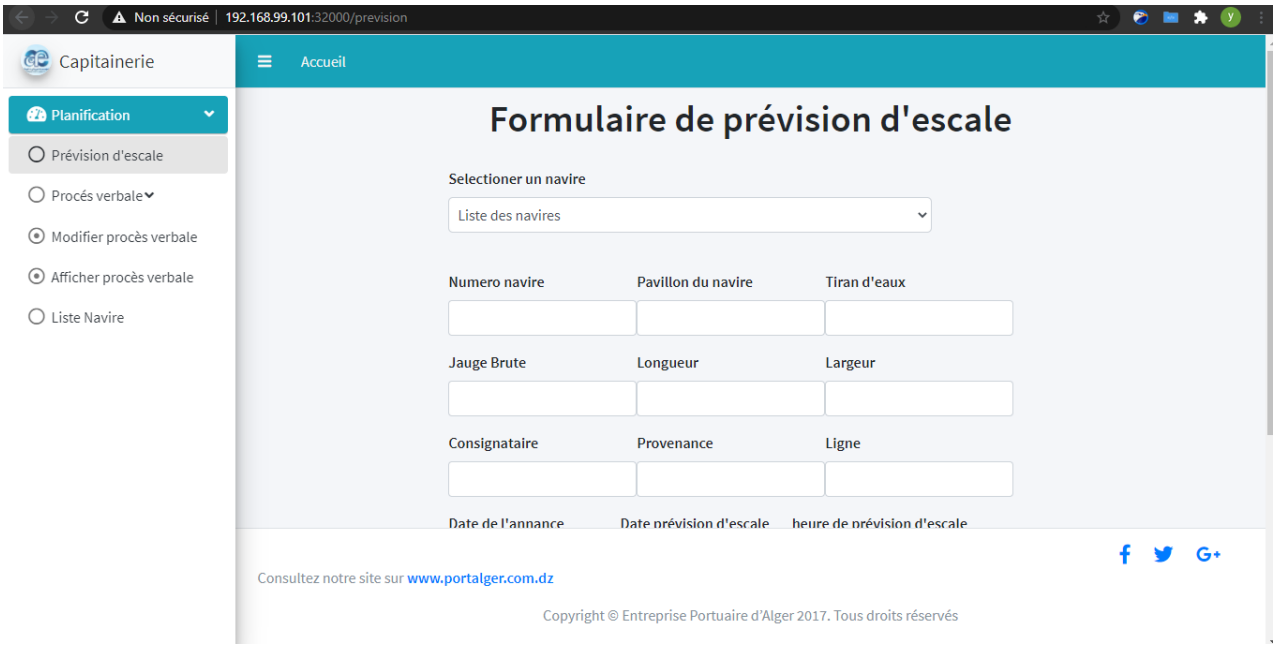


Figure 47 : Interface de procès-verbal déployer sur kubenetes.

Après l'exécution de la commande :**minikube dashboard .**

La figure suivante représente les informations sur les pods et les services de l'application (port, adresse IP ,fichiers.yaml…).

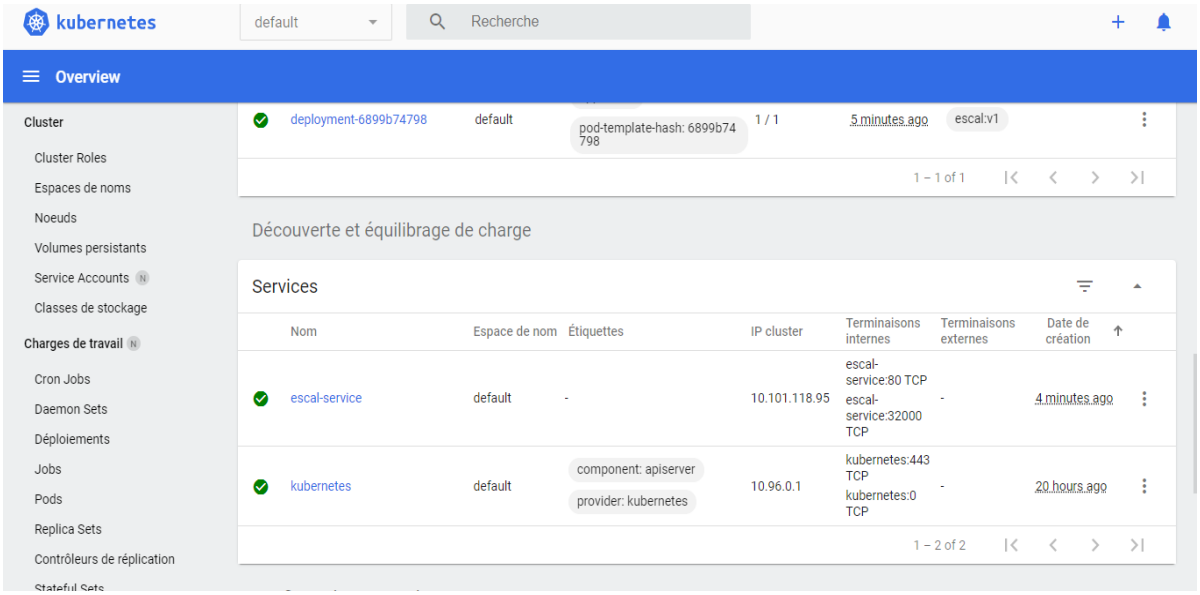


Figure 48: Interface de tableau de bord de kubernetes.

IV.10 Conclusion :

Nous avons présenté dans ce chapitre les différentes technologies, environnement et langages de programmation utilisés dans la réalisation de ce travail. Ensuite, nous avons montré la simulation de notre application, après avoir présenté les différentes étapes de configurations à suivre pour ce faire.

Conclusion générale

Conclusion générale

Adopter l'approche microservices n'est pas un sujet de prospective mais bien une réalité pour les entreprises, surtout que les géants du web comme Netflix et Amazon la trouvent efficace et fiable. En effet entreprises doivent désormais comprendre que le choix architectural est un enjeu très important qui conditionne la performance de leurs activités. Le projet intitulé « Migration en microservices »lancé par EPAL constitue un pas en avant vers une agilité de bout en bout. Son objectif vise à améliorer et à faire évoluer son organisation. Pour mener à terme ce projet, nous avons procédé par une suite d'étapes bien étudiées nous étions responsables de la mise en place de l'architecture de la solution ainsi que la conception, la réalisation et le déploiement. Nous avons pu résoudre à travers cette migration les limites exigées par le style architectural monolithique sur les grands projets. A savoir :

- Chaque microservice est déployé séparément. En cas de modification impactant le périmètre d'un microservice on le redéployait seul.
- Chaque microservice est dédié à réaliser une tâche bien spécifique, donc la durée des tests est minimale.
- Pour chaque microservice nous choisissons les technologies qui conviennent mieux à sa réalisation selon le besoin.

Afin d'améliorer notre projet nous proposants de nouvelles perspectives à savoir :

- La réalisation de tous les microservices de la capitainerie ainsi que le microservice administrateur et assurer la communication entre eux ainsi que leurs déploiements.
- Automatiser les phases de déploiement et de l'intégration continue en utilise les outils de version ING à savoir Jenkins.
- Automatiser la phase de tests.

Glossaire :

GLOSSAIRE :

A :

- ✓ **API Rest :** Application Programming Interface Gateway Representational State Transfer Une interface de programmation d'application qui fait appel à des requêtes HTTP.
- ✓ **API** : Application Programming Interface elle peut être résumée à une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

B :

- ✓ **BDD :** Base De Données

C :

- ✓ **CSS :** Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML.
- ✓ **CI :** Continuous Integration
- ✓ **CD :** Continuous Delivery (livraison continue)
- ✓ **CPU :** Control Processing Unit
- ✓ **CNCF :** Cloud Native Computing Fondation

D :

- ✓ **DDD :** Domain Driven Design
- ✓ **DevOps** : Est une approche qui permet de réunir et de collaborer les équipes de développements et d'exploitation afin de pouvoir crée et livrer des produits plus performants et plus fiables
- ✓ **DRH :** Direction Ressources Humaines
- ✓ **DFC :** Direction Finances et Comptabilités
- ✓ **DPI :** Direction Planification et Informatique

- ✓ **DER :** Direction Exploitation et Réglementation
- ✓ **DM :** Direction Manutention
- ✓ **DCL :** Direction Central Logistique
- ✓ **DAC :** Direction Acconage
- ✓ **DR :** Direction Remorquage
- ✓ **DTD :** Direction Travaux et Développement
- ✓ **DTC :** Direction Terminal à Conteneur
- ✓ **DC :** Direction Capitainerie
- ✓ **DMZ :** Demilitarized Zone est un sous-réseau séparé du réseau local et isolé de celui-ci et d'Internet (ou d'un autre réseau) par un pare-feu. Ce sous-réseau contient les machines étant susceptibles d'être accédées depuis Internet, et qui n'ont pas besoin d'accéder au réseau local.
- ✓ **DOM :** Document Objet Model

E :

- ✓ **EPAL :** Entreprise Portuaire d'*Alger*
- ✓ **EPE :** Entreprise Publique et Economique
- ✓ **EDI :** environnement de développement intégré

F :

- ✓ **FTP :** File Transfer Protocol

H :

- ✓ **HTML :** Hyper Text Mark-Up Language

I :

- ✓ **ISPS code :** International Ship and Port Facility Security (ISPS), qui en français signifie « Code international pour la sûreté des navires et des installations portuaires » il permet de détecter et d'éviter les actes illicites au sein des espaces maritimes et portuaires

- ✓ **IP :** Internet Protocol, est un numéro d'identification qui est attribué de façon permanente ou provisoire à chaque périphérique relié à un réseau informatique qui utilise l'Internet Protocol.
- ✓ **IDE :** Integrated Development Environment
- ✓ **IBM:** International Business Machines Corporation

J :

- ✓ **JSON :** (Java Script Object Notation) est un format d'échange de données
- ✓ **JDT :** Java Development Tools
- ✓ **JS :** Java Script

L :

- ✓ **LAN :** Local Area Network Il s'agit d'un ensemble d'ordinateurs appartenant à une même organisation et reliés entre eux dans une petite aire géographique

M :

- ✓ **MySQL :** est un serveur de bases de données libre. Il intègre le language SQL (Structured Query Language). Il est multi-thread, et multi-utilisateurs

O :

- ✓ **OS :** Operating System

P :

- ✓ **PHP** : Hypertext Preprocessor c'est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web.

R :

- ✓ **REST :** Representational State Transfer est un style d'architecture logicielle définissant un ensemble de contraintes à utiliser pour créer des services web
- ✓ **RAM :** Random Access Memory
- ✓ **RKT :** à prononcer "rock-it", est un container runtime créé par les équipes de CoreOS,

S :

- ✓ **SCRUM :** est une méthode agile dont le principe de base est que l'équipe avance ensemble et soit toujours prête à réorienter le projet au fur-et-à-mesure de sa progression.
- ✓ **SPA :** Société Par Actions
- ✓ **SIE :** Direction Sécurité Interne de l'Entreprise
- ✓ **SOA :** Architecture Orienté Service
- ✓ **SOAP :** Simple Object Access Protocol
- ✓ **SGBD :** Système de Gestion de Base de Données
- ✓ **SSH :** Secure Shell est un protocole de communication réseau qui permet à deux ordinateurs de communiquer.

T :

- ✓ **T.V.A :** taxe sur la valeur ajoutée

U :

- ✓ **UML :** Unified Modeling Language
- ✓ **URL :** Uniform Resource Locator est couramment appelé adresse web. Cette adresse sert à désigner une ressource présente sur le web par une suite de caractère ASCII.

V :

- ✓ **VM :** machines virtuelles

W :

- ✓ **WAN :** Wide Area Network est un réseau informatique ou un réseau de télécommunications couvrant une grande zone géographique

X :

- ✓ **XML :** Extensible Markup Language C'est simplement une façon de stocker des données qui peuvent être facilement lues par d'autres programmes.

Bibliographie

[1],lien : https://cloud.google.com/solutions/migrating-a-monolithic-app-to-microservices-gke?hl=fr

https://whatis.techtarget.com/fr/definition/architecture-monolithique

[2]lien :https://www.google.com/search?sxsrf=ALeKk0352QAH2oZnUts82j3aJh0WGw6UPQ%3A1585574559640&ei=n_KBXv_XJt3Kgwfj0qzABA&q=architecture+microservice&oq=application+monolithique+definition&gs_lcp=CgZwc3ktYWIQARgEMgQIABBHMgQIABBHMgQIABBHMgQIABBHMgQIABBHMgQIABBHMgQIABBHUABYAGCPWmgAcAF4AIABAIgBAJIBAJgBAKoBB2d3cy13aXo&sclient=psy-ab

[3], lien : https://www.talend.com/fr/resources/guide-microservices/

[4] ,lien : http://ziobrando.blogspot.com/2009/06/

[5], A. Avram,F.Maniscu Domain Driven Design vite fait.p63

[6],lien : https://www.fichier-pdf.fr/2017/02/14/rapport-pfe-salma/preview

[7], A. Avram,F.Maniscu Domain Driven Design vite fait.p67

[8], lien : https://www.slideshare.net/doniahammami/rapport-pfe-talan2018doniahammami

[9], lien : https://www.martinfowler.com/bliki/PolyglotPersistence.html

[10], lien : https://www.slideshare.net/LiliaSfaxi/eserviceschp5-microservices-et-api-management

[11],lien : https://www.lemagit.fr/conseil/Architecture-monolithique-vs-microservices-avantages-et-inconvenients

[12],lien : https://www.supinfo.com/articles/single/5676-qu-est-ce-que-architecture-microservices

[13],lien : https://www.javatpoint.com/devops-lifecycle

[14],lien : https://www.simplilearn.com/agile-vs-devops-article

[15],lien :https://www.google.com/search?q=la+m%C3%A9thode+agile+scrum&oq=la+m%C3%A9thode+agile+scrum&aqs=chrome..69i57.6473j0j7&sourceid=chrome&ie=UTF-8

[16], lien : https://www.simplilearn.com/agile-vs-devops-article

[17] : Pascal Roques, Franck Vallée, UML 2 en action de l'analyse des besoins à la conception(p51), edition Eyrolles2007

[18], lien : https://www.commentcamarche.com/contents/1138-uml-cas-d-utilisation-use-cases

[19] https://www.pack-logiciels-libres.fr/spip.php?logiciel35

[20] https://openclassrooms.com/fr/courses/4668056-construisez-des-microservices/5122425-decouvrez-le-framework-spring-boot#:~:text=Spring%20Boot%20est%20un%20framework%20qui%20permet%20de%20d%C3%A9marrer%20rapidement,utilise%20l'annotation%20%40EnableAutoConfiguration%20.

[21] https://kubernetes.io/fr/

[22]https://www.01net.com/telecharger/windows/Programmation/creation/fiches/130819.html

[23] https://fr.wikipedia.org/wiki/Visual_Studio_Code

[24] https://dept-info.labri.fr/ENSEIGNEMENT/programmation2/intro-eclipse/

[25] https://www.pack-logiciels-libres.fr/spip.php?logiciel44

[26] lien, https://www.tice-education.fr/tous-les-articles-er-ressources/articles-internet/819-draw-io-un-outil-pour-dessiner-des-diagrammes-en-ligne

[27] lien, https://1000logos.net/java-logo/

[28]lien, http://glossaire.infowebmaster.fr/html/

[29] lien, http://glossaire.infowebmaster.fr/css/

[30]lien,

https://developer.mozilla.org/fr/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[31] lien, https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[32 https://fr.wikipedia.org/wiki/JQuery