République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. MIRA de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de Fin de Cycle

En vue de l'obtention du diplôme Master 2 en informatique

Option : Génie Logiciel

Thème

Conception et réalisation d'une application Web pour le suivi des escales des navires.

"Cas d'étude : Entreprise Portuaire de Bejaia - EPB".

Présenté par : Mr. MEHDI Mounir

Devant le jury composé de :

Président	Mme HAMZA Lamia	M.C.A
Examinateur	Mr ZERARGA Loutfi	M.C.A
Encadrant	Mme AIT HACENE Souhila	M.A.A

2021/2022

Remerciements

Mes premiers remerciements s'adressent à Dieu le tout puissant qui par sa bonté et sa miséricorde m'a permis d'avoir le courage, la foi et la Volonté de mener ce travail.

Je tiens à remercier mon encadrante Mme AIT HACENE Souhila qui a été présente à tout moment de la réalisation de ce projet, et pour ses conseils et ses orientations.

Je remercie également Mr MOUOUDJ Omar, qui a accepté de m'encadrer durant le stage, sans oublie de remercie tous ses collègues qui ont était très accueillants.

Ainsi, mes remerciements aux membres du jury pour l'intérêt qu'ils ont porté à notre projet en acceptant de l'examiner.

A tous mes enseignants et les membres du département informatique, sans oublier tous ceux qui m'ont aidé à mener à bien ce projet.

Dédicaces

Ce modeste travail est dédié :

A mes chers parents qui m'ont soutenu et encouragé tout au long de mes études.

A mes frères.

A mes enseignants.

A mes amis(e).

A toutes personnes qui m'ont apporté de l'aide.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

LISTE DES ABREVIATIONS

APCS : Algerian Port Community System

BDD : Base De Données

CNAN : Compagnie Nationale Algérienne de Navigation

CPN : Conférence de Placement des Navires

CSS : Cascading Style Sheets

DC : Direction Capitainerie

EPB : Entreprise Portuaire de Bejaia

HTML : Hypertext Markup Language

JS : JavaScript

MVC : Modèle-Vue-Contrôleur

ONP : Office National des Ports

PHP : Hypertext Preprocessor

POO : Programmation Orienté Objet

S.I.P : Portail d'Information Portuaire

SO.NA.MA : Société Nationale de Manutention

SPA : Single Page Application

UML : Unified Modeling Language

UP : Unified Process

VSC : Visual Studio Code

INTRODUCTION GÉNÉRALE

Les entreprises sont désormais conscientes de l'impact d'une gestion efficace des ressources internes sur l'amélioration des performances et donc de leurs compétitivités sur le marché. Cependant, les tâches administratives deviennent de plus en plus difficiles et complexes à cause de la croissance des activités créant des flux de données massifs. Pour surmonter ces difficultés, les entreprises sont prêtes à investir dans la mise en place de systèmes intégrant les dernières technologies logicielles en vue d'améliorer leurs services, de réduire leurs coûts et d'accroître leur flexibilité et leur agilité.

L'Entreprise Portuaire de Bejaia (EPB) souhaite rivaliser avec d'autre entreprise de renommé international en modernisant son système informatique. En effet, à l'issue de nos observations, lors de notre stage à l'EPB, nous avons remarqué que certaines directions rencontrent des difficultés lors de la gestion et le traitement des données en temps réel. Notre étude de cas a été réalisée au niveau du service capitainerie qui non seulement procède un certain nombre d'applications utilisées pour la gestion des escales, mais en plus ces système n'interagissent pas entre eux et on retrouve une même fonctionnalité dans plusieurs applications à la fois.

Aussi, afin de permettre une meilleure gestion et traitement des données en relation avec les escales des navires au niveau de l'EPB et plus précisément du service capitainerie, nous proposons une application web dynamique qui regroupe toutes les fonctionnalités nécessaires pour la gestion des escales. En plus, nous avons intégré une fonction permettant l'envoi de notifications dés modification d'une information concernant l'état d'un navire (en attente, en rade, en quai, sortie) ou ses propriétés (le poids, tirant eau, etc.). Cette dernière fonctionnalité permet au personnel de l'EPB d'être tenus informé de toute modification ou changement d'information en temps réel.

Ce mémoire est divisé en quatre chapitres, structuré comme suit :

Chapitre 1 : Dans ce chapitre, nous présentons l'organisme d'accueil, la problématique et les objectifs que traite ce projet ainsi que la méthodologie de développement et de conception utilisée.

Chapitre 2 : Ce chapitre porte sur l'étude préliminaire et analyse des besoins où se fait une description globale du comportement du futur système, à travers la définition des besoins fonctionnels et non fonctionnels, ainsi que l'identification des acteurs et les différents cas d'utilisation, qui seront illustrés par des diagrammes suivis par une description textuelle.

Chapitre 3 : Ce chapitre concerne la phase de conception où nous détaillons notre solution et présentons les diagrammes de séquences et de classe ainsi que le model relationnel utilisé lors de l'implémentation.

Chapitre 4 : Dans ce dernier chapitre, nous décrivons l'environnement de développement et les choix techniques adoptés lors de la réalisation de notre projet. Nous passons par la suite à la présentation de quelques captures d'écrans des interfaces conçues.

Nous clôturons ce mémoire par la conclusion générale qui résume les principales fonctionnalités réalisées ainsi que quelques perspectives.

Chapitre I.

Présentation de l’organisme

1. Introduction

Dans ce premier chapitre, nous commençons par présenter l'organisme d'accueil, son historique, ses missions, ses activités, et sa structure. Ensuite, nous passons à l'étude de l'existant afin d'en déterminer les lacunes et proposer une solution dans le but d'assurer une meilleure gestion des escales. A la fin de ce chapitre, nous présentons la méthodologie de développement sélectionnée et adoptée pour la réalisation de notre projet.

2. Présentation de l'organisme d'accueil

Pour avoir une idée générale sur l'Entreprise Portuaire de Bejaia (EPB), nous décrivons son évolution à travers le temps, sa position géographique, ses missions ainsi que ses activités [1].

مؤسسة ميناء بجاية
ENTREPRISE PORTUAIRE DE BEJAIA

Figure 1 : Logo d'EPB [1]

2.1. Historique de l'EPB

Le décret n°82-285 du 14 Août 1982 publié dans le journal officiel n° 33 porta création de l'Entreprise Portuaire de Béjaïa ; entreprise socialiste à caractère économique ; conformément aux principes de la charte de l'organisation des entreprises, aux dispositions de l'ordonnance n° 71-74 du 16 Novembre 1971 relative à la gestion socialiste des entreprises et les textes pris pour son application à l'endroit des ports maritimes.

L'entreprise, réputée commerciale dans ses relations avec les tiers, fut régie par la législation en vigueur et soumise aux règles édictées par le susmentionné décret.

Pour accomplir ses missions, la société se substitue à l'Office National des Ports (ONP), à la Société Nationale de Manutention (SO.NA.MA) et partiellement à la Compagnie Nationale Algérienne de Navigation (CNAN).

Elle fut dotée par l'Etat, du patrimoine, des activités, des structures et des moyens détenus par l'ONP, la SO.NA.MA et de l'activité Remorquage, précédemment dévolue à la CNAN, ainsi que du personnel dédié à la gestion et aux fonctionnements de l'EPB.

En exécution des lois n° 88.01, 88.03 et 88.04 du 02 Janvier 1988 s'inscrivant dans le cadre des réformes économiques et portant sur l'autonomie des entreprises, et suivant les prescriptions des décrets n°88.101 du 16 Mai 1988, n°88.199 du 21 Juin 1988 et n°88.177 du 28 Septembre 1988, l'Entreprise Portuaire de Béjaïa ; entreprise socialiste ; est transformée en Entreprise Publique Economique, Société par Actions (EPE-SPA) depuis le 15 Février 1989, son capital social fut fixé à Dix millions (10.000.000) de dinars algérien, actuellement, il a était augmenté à 3.500.000.000 de DA.

2.2. Position géographique de l'EPB

Le port de Bejaia est situé sur la côte Est de l'Algérie, le golf de la baie de Bejaia. Il joue un rôle très important dans les transactions internationales vu sa position géographique. Il est délimité par :

- La mer méditerranéenne, à l'Est.
- La ville de Bejaia, à l'Ouest.
- L'avenue des frères Amrani et la route nationale N° 09, au Nord,
- Les jetées de fermeture et du large sur une longueur de 2.750 m, au Sud.

Figure 2 : Carte géographique de l'EPB [1]

2.3. Missions de l'EPB

L'entreprise a pour missions :

- La gestion, l'exploitation et le développement du domaine portuaire dans le but de promouvoir les échanges extérieurs du pays, et d'en assurer la sûreté et la sécurité.
- Elle est chargée des travaux d'entretien, d'aménagement, de renouvellement et de création d'infrastructures.
- L'entreprise assure également des prestations à caractère commercial, à savoir : le Remorquage, la manutention et l'aconage.

2.4. Activités de l'EPB

Les principales activités de l'entreprise sont :

- L'exploitation de l'outillage et des installations portuaires.
- L'exécution des travaux d'entretien, d'aménagement et de renouvellement de la super structure portuaire.
- L'exercice du monopole des opérations d'acconage et de manutention portuaire.
- L'exercice du monopole des opérations de remorquage, de pilotage et d'amarrage.

2.5. Structure de l'EPB

L'organigramme organisationnel actuel de l'EPB, présenté dans la figure 3, découle de la dernière forme d'organisation des ports algériens qui est basée essentiellement sur la séparation des missions relavant des prérogatives de puissance publique des activités commerciales libres à la concurrence. Ainsi, l'EPB est organisé en huit directions, classées en deux structures :

- **Direction Opérationnelles**

Cette direction est composée de structures ayant une relation directe avec les clients, et qui prennent en charge les opérations sur le terrain.

- **Direction Fonctionnelles**

Cette direction est composée de structures de soutien aux structures opérationnelles.

La figure 3 représente l'organigramme des différentes directions de l'EPB :

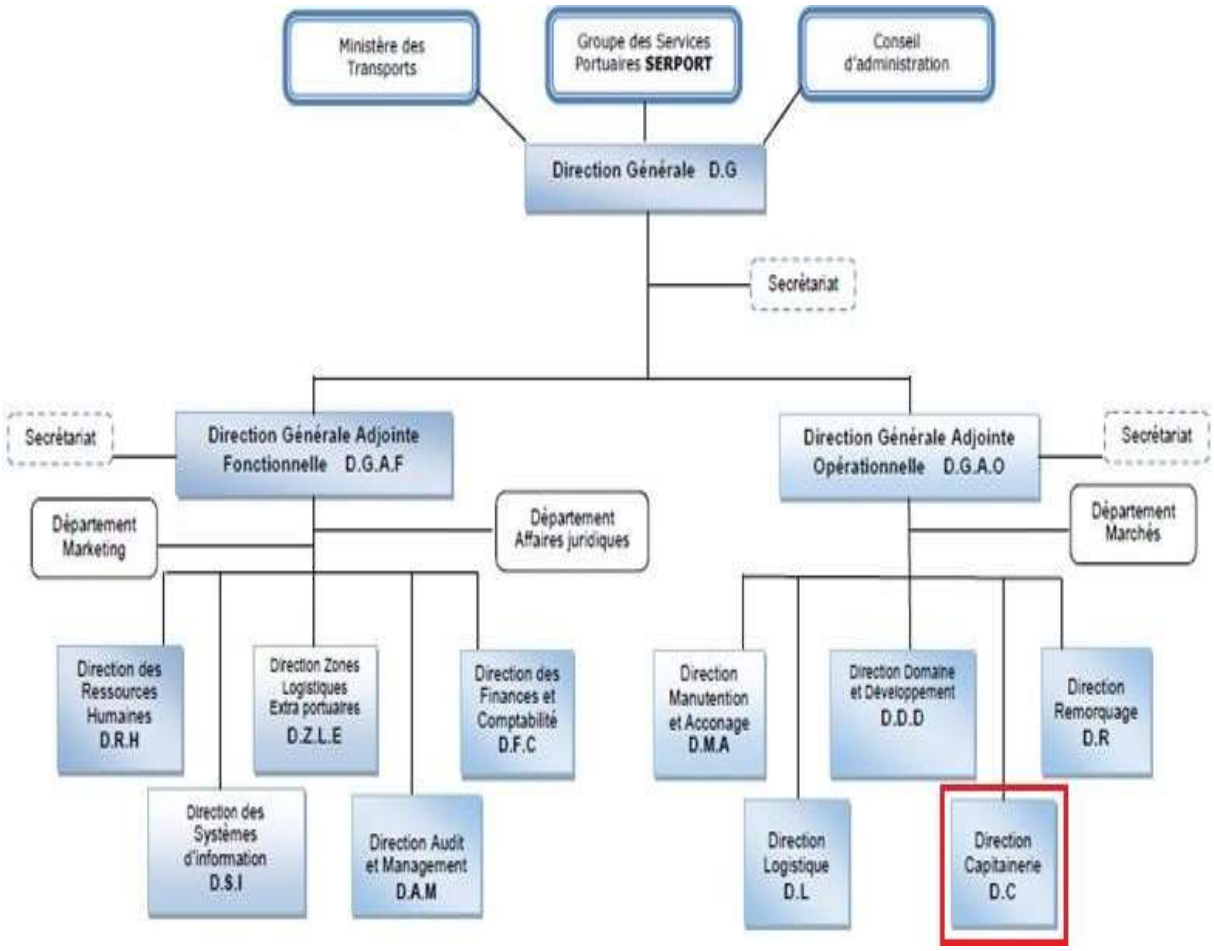

Figure 3 : L'organigramme des directions de l'EPB [1]

3. Etude de l'existant

Lors de la gestion des activités portuaire, le module escale est considéré comme étant le noyau de tous les autres modules, d'où les autres processus de l'EPB puisent les données nécessaires à leur bon fonctionnement.

Nous avons focalisé notre étude au niveau de la Direction Capitainerie (DC), où nous avons étudié les systèmes utilisés par l'EPB pour la gestion des escales, dans le but de faire ressortir les lacunes de ces systèmes et de proposer une solution pour les paliers.

Lors de notre étude, nous avons constaté l'utilisation de plusieurs applications pour la gestion des escales.

Le tableau 1 représente l'ensemble des applications utilisés actuellement par l'EPB pour la gestion des escales :

Application	Type	Technologie	Utilisateurs	Fonctionnalités
Gestion des escales	Logiciel	DELPHI	- Directeur - Gestionnaire des escales - Officier du port - Officier de radio	Ajouter ou modifier des navires : ▪ En attendus. ▪ En rade. ▪ En quai.
Suivi de la Conférence de Placement des Navires (CPN)	Logiciel	DELPHI	- Directeur - Gestionnaire des escales - Officier du port - Officier de radio	▪ Ajouter ou modifier les décisions et prévisions des CPN.
Récupération des annonces	Logiciel	DELPHI	- Directeur - Gestionnaire des Escales - Officier de radio	▪ Récupération des annonces. ▪ Corrections des données. ▪ Validation de l'annonce.
Portail d'Information Portuaire (SIP)	Site Web Local	PHP	- Communauté d'EPB	Consultations de : ▪ Les CPN. ▪ Liste des navires attendus. ▪ Liste des navires en rade. ▪ Liste des navires en quai.

Tableau 1 : Tableau des applications de gestion d'escales existantes

La CPN c'est une réunion organisée tous les jours le matin, entre l'officier du port et les personnes concernées par le navire ou la cargaison afin de planifier les manouvres et d'affecter les moyens.

4. Problématique

Nous avons constaté un certain nombre de difficultés et d'incohérences sur les systèmes existants telle que :

- L'existence de trois systèmes séparés pour la gestion d'escales, ce qui engendre une redondance et donc une lenteur.
- Le partage et l'échange d'informations avec d'autres systèmes se fait manuellement ou par des passerelles, ce qui produit la perte de temps et augmente le taux d'erreur.
- Absence d'interactions entre les systèmes ce qui provoque l'incohérence des données.
- Les décisions et les modifications des données ne sont pas transmise en temps réel.

- La base de données (BDD) utilisée est incomplète en termes de relations entre les données.
- Manque des fonctionnalités dans les systèmes pour faciliter la manipulation.

5. Solution Proposées

Dans cette partie nous allons présenter notre solution à savoir une application web dynamique pour la gestion des escales.

5.1. Objectifs

Le but de notre projet est de proposer une solution qui réponds aux exigences de l'EPB à savoir le service de la capitainerie, avec objectifs de :

- Proposition d'une application web dynamique ouvert à la communauté d'EPB.
- La mise en place d'une base de données unique.
- L'augmentation du taux d'informatisation des activités portuaires, tel que : l'état de navire, informations sur les manouvres et les moyens utiliser, etc.
- La réduction du taux d'erreurs lors du traitement des données.
- Élaboration un reporting précis et en temps réel pouvant utiliser lors de la prise de décision durant la CPN par exemple.
- Programmation des services sur mesures.

5.2. Solution « Suivi des escales »

Dans ce mémoire nous proposons une application web dynamique, dédier pour la gestion des escales utilisée par le service de capitainerie. L'application proposée permet de palier les lacunes des applications existantes et offres de nouvelles fonctionnalités tel que :

- L'envoie de notification.
- Gestion complète la CPN (ajout, modification, suppression).
- Gestion de l'état des navires (ajouter en attente, rade, en quai, mettre en sortie).
- Suivre l'état de navires.
- Consultation de tableau de bord.

6. Méthodologie de développement

L'adoption d'une méthodologie de développement est nécessaire pour garantir une bonne gestion et communication lors du développement d'un projet.

6.1. Unified Process (UP)

L'UP est un processus de développement de logiciels. Ce dernier est une trame commune des meilleures pratiques de développement. La définition d'un processus UP est constituée de plusieurs disciplines d'activité de production et de contrôle de cette production [2].

6.1.1. Caractéristiques d'UP

- **Itératif et incrémental :** le projet est découpé en itérations de courte durée qui permettent de mieux suivre l'avancement globale. A la fin de chaque itération une partie exécutable du système finale est produite, de façon incrémentale (par ajout).
- **Centré sur l'architecture :** tout système logiciel complexe doit être décomposé en partie modulaire afin d'en faciliter la maintenance et l'évolution. Cette architecture (fonctionnelle, logique, matérielle, etc.) doit être modéliser en UML.
- **Guidé par les cas d'utilisation d'UML :** le but principal d'un système informatique est de satisfaire les besoins du client. Le processus de développement sera donc accès sur l'utilisateur. Le cas d'utilisation permet d'illustrer les besoins du client en termes de fonctionnalités.
- **Piloté par les risques :** les risques majeurs du projet doivent être identifiés au plus tôt mais surtout levés le plus rapidement. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.

6.1.2. Les phases du processus unifié

- **Analyse des besoins :** L'analyse des besoins donne une vue du projet sous forme de produit fini. Cette phase porte essentiellement sur les besoins principaux (du point de vue de l'utilisateur), l'architecture générale du système, les risques majeurs, les délais et les coûts.
- **Elaboration :** L'élaboration reprend les éléments de la phase d'analyse des besoins et les précise pour arriver à une spécification détaillée de la solution à mettre en œuvre.

L'élaboration permet de préciser la plupart des cas d'utilisation, de concevoir l'architecture du système et surtout de déterminer l'architecture de référence.

- **Construction :** La construction est le moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version. C'est la phase la plus consommatrice en ressources et en efforts.
- **Transition :** Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées. Tout simplement la phase de transition permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux.

La figure 4 illustre les phases du processus unifié :

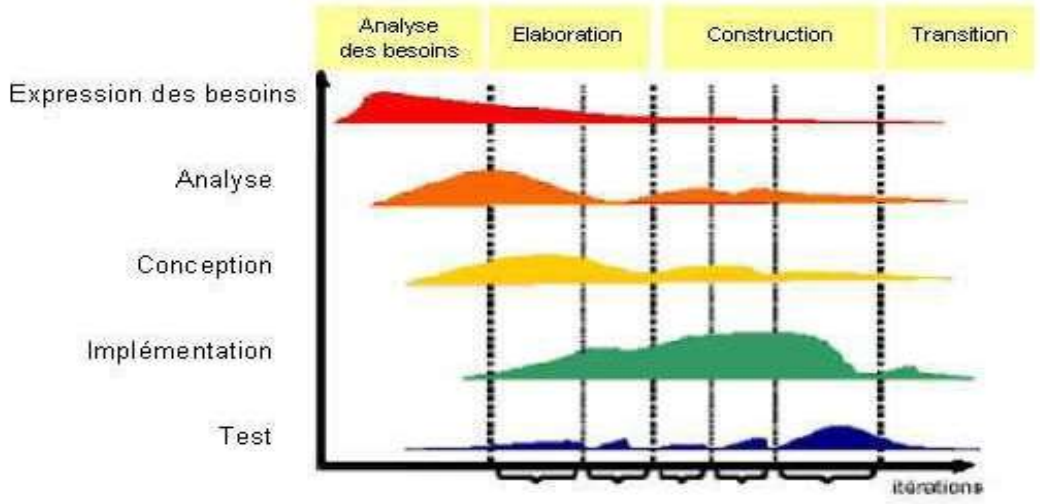

Figure 4 : Les phases du processus unifié [2]

Chaque phase dans UP est une composition d'étapes. Les étapes définit dans le processus de développement UP sont : l'expression des besoins, l'analyse, la conception, l'implémentation et le test et le déploiement.

6.2. Unified Modeling Langage (UML)

L'UML est un langage de modélisation graphique et textuel. Il est destiné à comprendre et décrire des besoins, concevoir des solutions, spécifier et documenter des systèmes, ainsi qu'à esquisser des architectures logicielles, et connecter les points de L'UML unifie à la fois

les notations et les concepts orientés objet. Il ne s'agit pas d'une simple notation graphique, car les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage [3].

6.2.1. Pourquoi UML

Dans la réalisation d'une application il est mieux de passer par la modélisation de système. Le choix d'UML nous permet de :

- Faire collaborer des participants de tous horizons autour d'un même document de synthèse.
- Obtenir une modélisation de très haut niveau indépendante des langages et des environnements.
- Faire des simulations avant de construire un système.
- Exprimer dans un seul modèle tous les aspects statiques, dynamiques, juridiques, spécifications, etc.
- Documenter un projet.
- Encadrer l'analyse.
- Faciliter la compréhension de représentations abstraites complexes.

6.2.2. Type de Diagrammes UML

UML s'articule autour de treize diagrammes, chacun d'eux étant dédié à la représentation de concept particuliers d'un système logiciel. Ces diagrammes sont repartis en deux catégories comme l'illustre la figure 5.

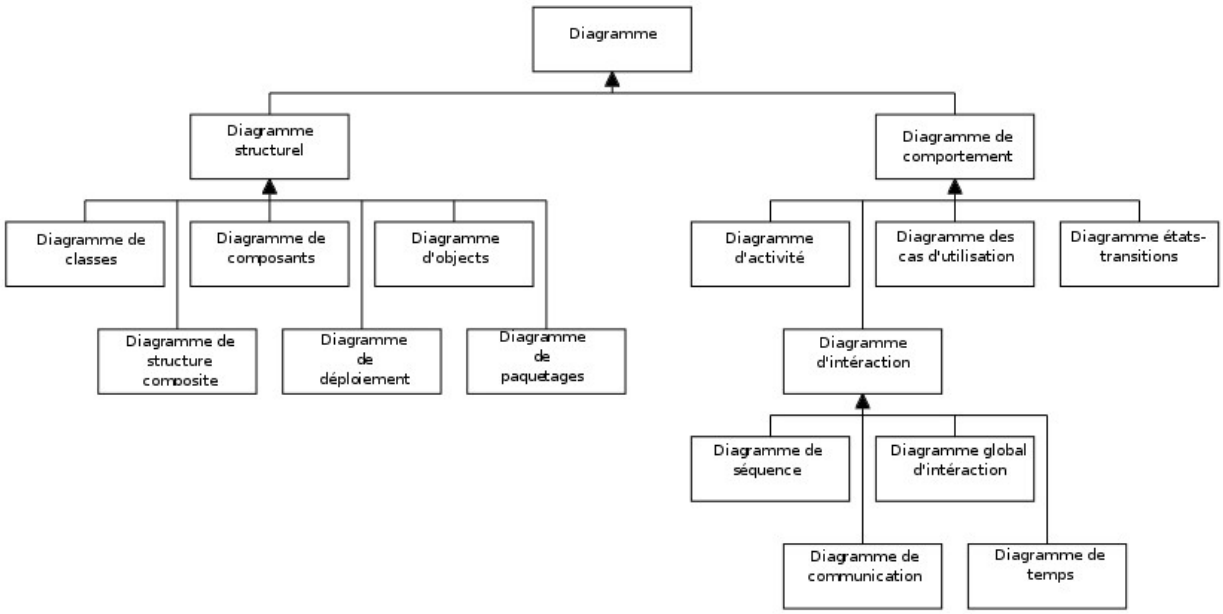

Figure 5 : Les types de diagrammes UML2 [3]

Voici la définition des diagrammes utilisé pour la modélisation et la conception de notre application :

- **Diagramme de contexte :** Permet de définir des limites d'étude. Il met le schéma en contexte en listant les acteurs ou éléments qui agissent ou interagissent avec eux.
- **Diagramme des cas d'utilisation :** destiné à représenter les besoins des utilisateurs par rapport au système. C'est l'un des diagrammes les plus structurants lors de l'analyse d'un système logiciel.
- **Diagramme de séquences :** Permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations.
- **Diagramme de classes :** représente la description statique du système où chaque classe a une partie donnée et une partie traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système.

7. Conclusion

Dans ce premier chapitre, nous avons présenté l'Entreprise Portuaire de Bejaia. Ensuite, nous avons étudié les applications utilisées actuellement par l'EPB dans le but de faire ressortir leur limites et faiblesses afin de proposée une solution sous forme d'une application regroupant toutes les fonctionnalités intervenant lors de la gestion des escales et intégrant de nouvelles fonctionnaires indispensables lors de cette gestion. Par la suite, nous avons présenté la méthodologie de développement adoptée durant le développement de notre solution. Dans le chapitre suivant, nous présentons la partie spécification et analyse des besoins.

Chapitre II.

Spécification et Analyse des besoins

1. Introduction

Dans ce chapitre, nous commençons par la première phase de développement de l'application qu'est la spécification et l'analyse des besoins. Nous identifiant les acteurs, après nous décrirons les besoins fonctionnels et non fonctionnels. Ensuite, nous passons aux diagrammes des cas d'utilisation et quelques description textuelle.

2. Identification des acteurs

Un acteur représente une entité externe qui joue un rôle et interagit directement avec le système étudié [4].

Dans notre application, nous avons distingué divers acteurs, qui sont classées en deux catégories à savoir :

2.1. Acteurs principaux

Un acteur principal est un acteur qui utilisent le système pour réaliser une ou plusieurs tâches. Nous pouvons donc considérer les acteurs principaux comme les futurs utilisateurs du l'application.

Les acteurs principaux de notre projet sont :

- **Communauté EPB :** Il s'agit de toutes les personnes qui ont accès à l'application, ils peuvent consulter les données publiques et les imprimer sans authentification.
- **Membre :** c'est toutes personnes à un compte pour s'authentifier, il a l'accès à consulter l'historique des escales des navires, consulter les notifications et de crée une.
- **Directeur :** Son rôle principale est de gérer les membres (Ajouter, modifier, désactiver), et de consulter leur historique.
- **Gestionnaire des escales :** son rôle est de s'assurer que les annonces soit bien récupérer et valider, il peut aussi modifier des données d'une escale en cas de changement d'informations.
- **Officier du Port :** Les officiers de port sont chargés de la gestion de la CPN (Ajouter, modifier, supprimer des prévisions ou mouvements).
- **Officier de Radio :** Il ajoute le navire en rade, et il exécute les prévisions et mouvements de la CPN, de plus il peut jouer le rôle de Gestionnaire à tout moment.

Dans la figure 6, nous illustrons les relations entre les acteurs principaux :

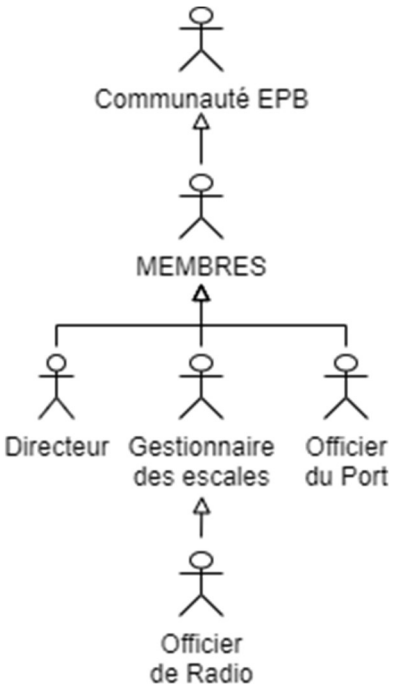

Figure 6 : Relation entre les acteurs principaux

2.2. Acteurs secondaires

Contrairement aux acteurs principaux, les acteurs secondaires n'utilisent pas le système mais le système à besoins d'eux pour fonctionner correctement. Cela est généralement un autre système avec lequel le nôtre doit échanger des informations. Notre application sollicite l'acteur secondaire suivant :

- **APCS.dz :** C'est une plateforme nationale communautaire portuaire d'échange de données numériques, qui permet de publier ou de récupérer toutes les données concernant l'annonce d'escale des navires.

3. Spécification des besoins

3.1. Besoins fonctionnels

Notre application doit satisfaire les besoins fonctionnels suivants :

- ✓ Authentification et gestion des rôles.
- ✓ Gestion d'escale : récupération d'annonce, ajout de navires en attendu, ajout de navires en rade, exécution des prévisions (Ajout de navire en quai si c'est une

prévision d'entrée ou rendre navire en état finie si c'est une prévision de sortie) et des mouvements sur le navire.

- ✓ Gestion de CPN : ajout, modification et/ou suppression des prévisions et mouvements.
- ✓ Archivage l'historique des membres.
- ✓ Intégration de système des notifications.
- ✓ Consultation des différents listes (Liste des navires attendus, liste des navires en rade, liste des navires en quai, liste des CPN, liste historique des navires).
- ✓ Pouvoir imprimer les différentes listes.

3.2. Besoins non fonctionnels

Ce sont les besoins permettant l'amélioration de la qualité des services offerts par une application. Dans notre travail nous avons considéré la liste des besoins non fonctionnel présentés ci-dessous :

- ✓ Sécurité : l'application doit être extrêmement sécurisée, et les données ne doivent pas être accessibles facilement à tous, mais plutôt au moyen d'un identifiant et d'un mot de passe attribués à l'utilisateur.
- ✓ Convivialité : l'application doit être facile à utiliser, et elle doit présenter un enchainement logique entre les interfaces.
- ✓ L'extensibilité : l'application devra être extensible c'est-à-dire qu'il serait possible d'ajouter ou de modifier des fonctionnalités.
- ✓ La performance : l'application devra être performante c'est-à-dire que le système doit réagir dans un délai minimal quel que soit l'action de l'utilisateur.
- ✓ L'ergonomie : l'application doit être intuitif.

4. Diagramme du contexte

Dans la figure 7, nous illustrons les différents acteurs qui interagissent avec nôtre système :

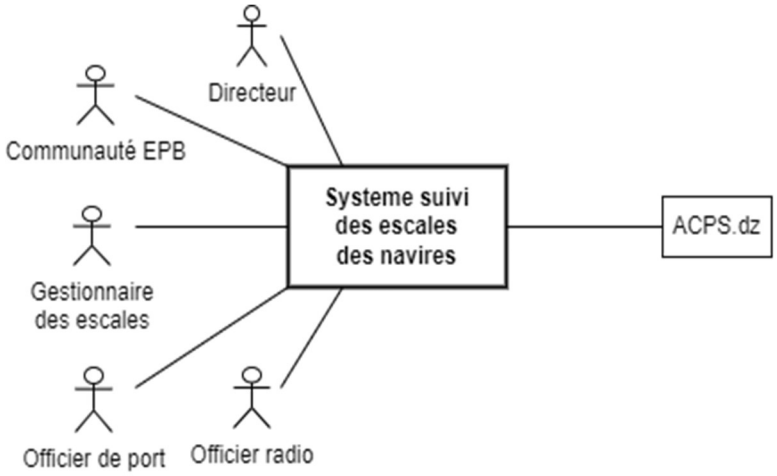

Figure 7 : Diagramme de contexte

5. Identification des cas d'utilisation

Un cas d'utilisation représente une fonctionnalité ou un service offert par le système, et il exprime les interactions entre les acteurs et le système [5].

Le tableau 2 regroupe les cas d'utilisation nécessaires pour le bon fonctionnement du système :

N°	Cas d'utilisation		Acteur
1	Authentifier		- Membre
2	Gérer les notifications	Envoyer une notification	- Membre
		ImprimerNotifications	
3	Consulter l'historique des escales	ImprimerEscales	- Membre
4	Gérer la liste des utilisateurs	Ajouter un utilisateur	- Directeur
		Modifier un utilisateur	
		ImprimerUtilisateurs	
5	Consulter historique des tâches	Chercher	- Directeur
		ImprimerHistorique	
6	Charger les nouvelles annonces	Ajouter à la liste attendu	- Gestionnaire des escales - Officier de radio
7	Gérer les informations des escales	Modifier	- Gestionnaire des escales - Officier de radio

8	Gérer l'exécution des décisions de CPN	Exécuter une décision		- Officier de radio
		Modifier une exécution		
9	Gérer les navires en rade	Ajouter à la liste en rade		- Officier de radio
10	Gérer CPN	Prévision d'entrée	Ajouter	- Officier du port
			Modifier	
			Supprimer	
		Prévision de sortie	Ajouter	
			Modifier	
			Supprimer	
		Mouvement	Ajouter	
			Modifier	
			Supprimer	
11	Afficher tableau de bord			- Communauté EPB
12	Consulter les décisions de la CPN	Chercher pas date		- Communauté EPB
		ImprimerCPN		
13	Consulter la liste des navires attendu	ImprimerAttendu		- Communauté EPB
14	Consulter la liste des navires en rade	ImprimerRade		- Communauté EPB
15	Consulter la liste des navires en quai	ImprimerQuai		- Communauté EPB

Tableau 2 : Liste des cas d'utilisation

6. Diagrammes des cas d'utilisation

Dans ce qui suit, nous présentons les diagrammes de cas d'utilisations définit lors de la modélisation de notre application.

6.1. Cas d'utilisation associés à Communauté EPB

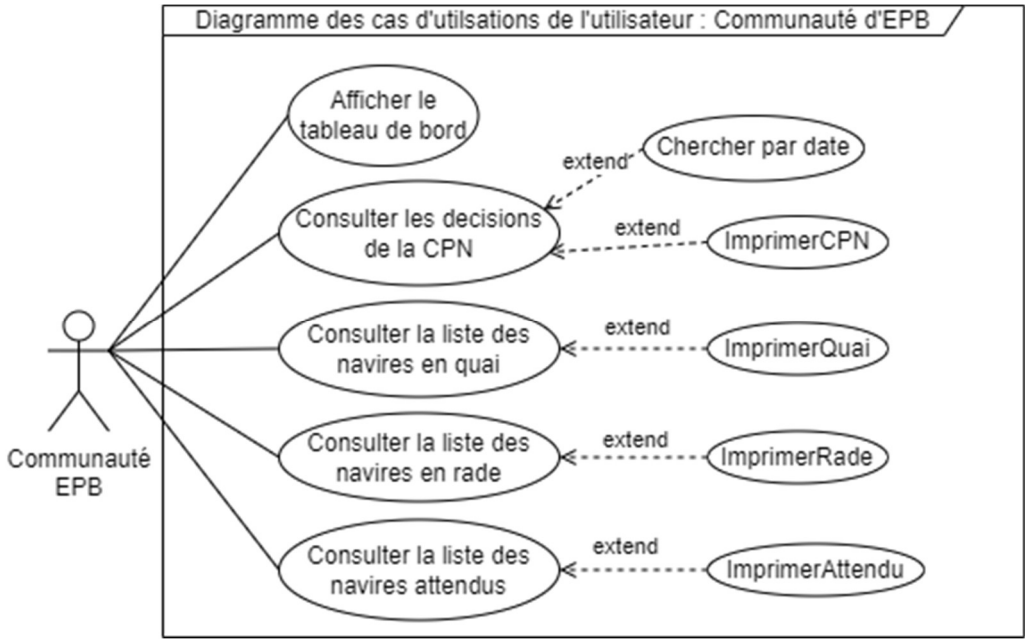

Figure 8 : Cas d'utilisation associés à Communauté EPB

Ce diagramme illustre les fonctionnalités que la communauté d'EPB peuvent réaliser sans s'authentifier.

6.2. Cas d'utilisation associés au Directeur

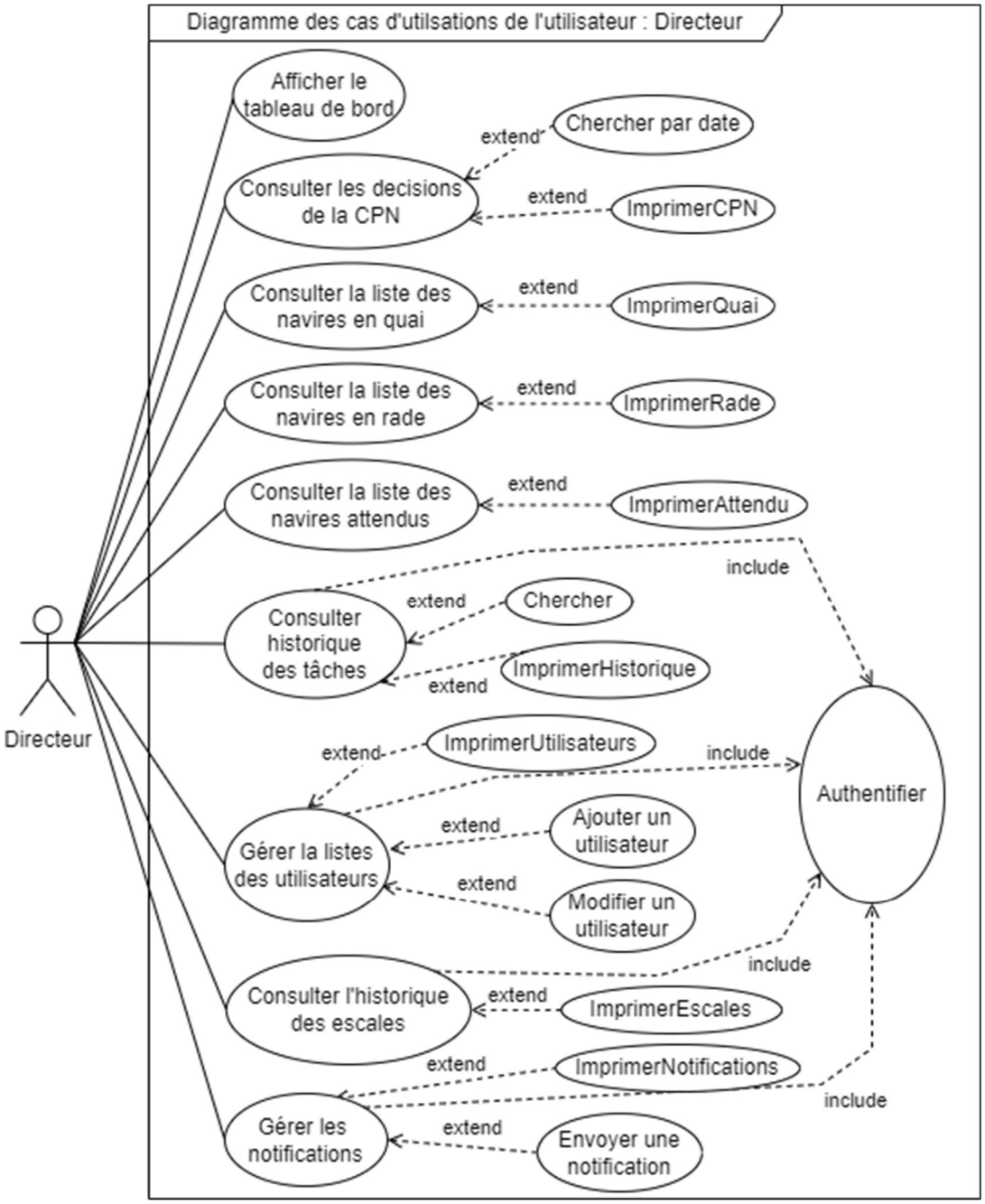

Figure 9 : Cas d'utilisation associés au Directeur

Ce diagramme illustre les cas d'utilisation associés au directeur et cela après l'authentification.

6.3. Cas d'utilisation associés au Gestionnaire des escales

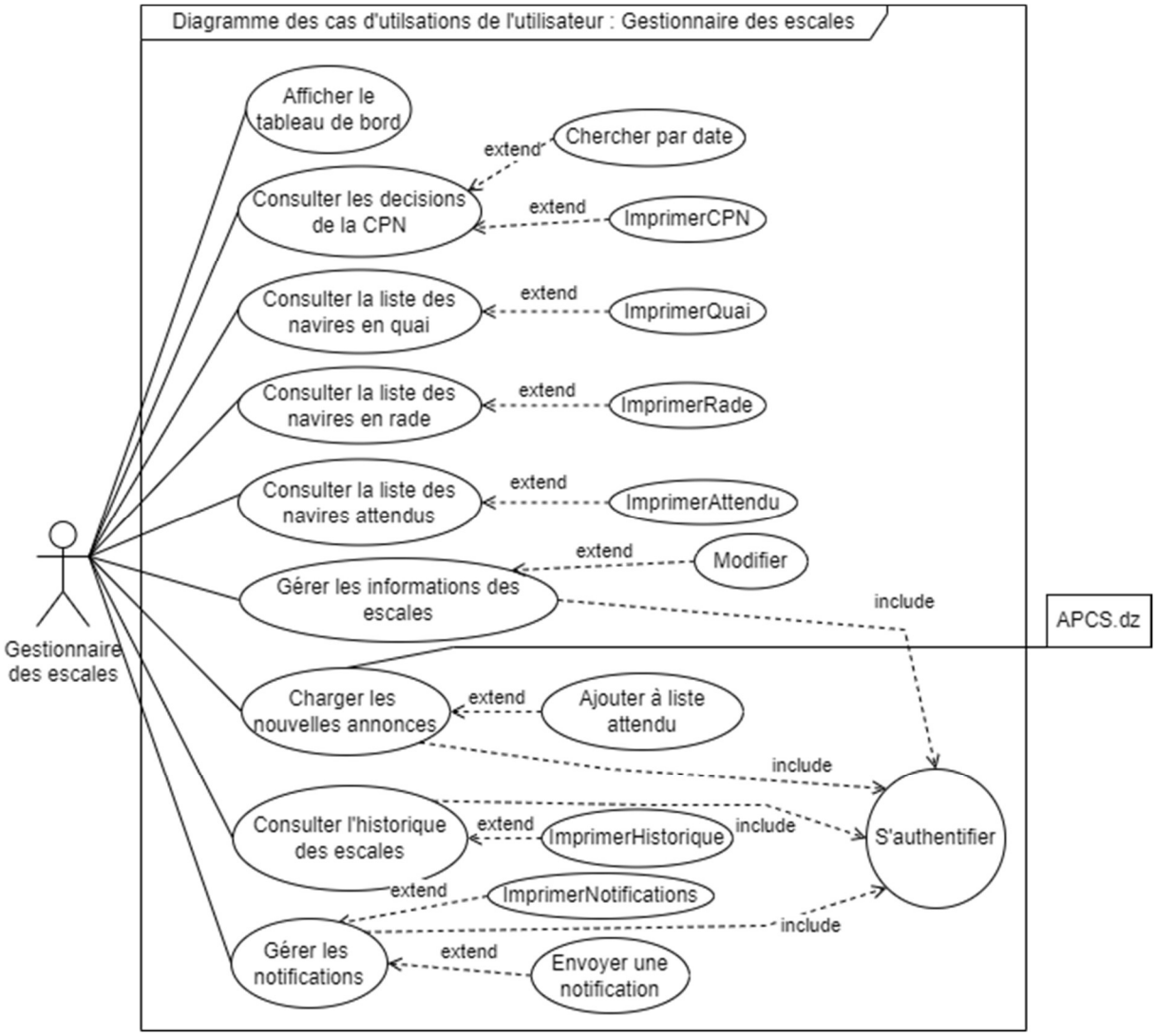

Figure 10 : Cas d'utilisation associés au Gestionnaire des escales

6.4. Cas d'utilisation associés à l'Officier du port

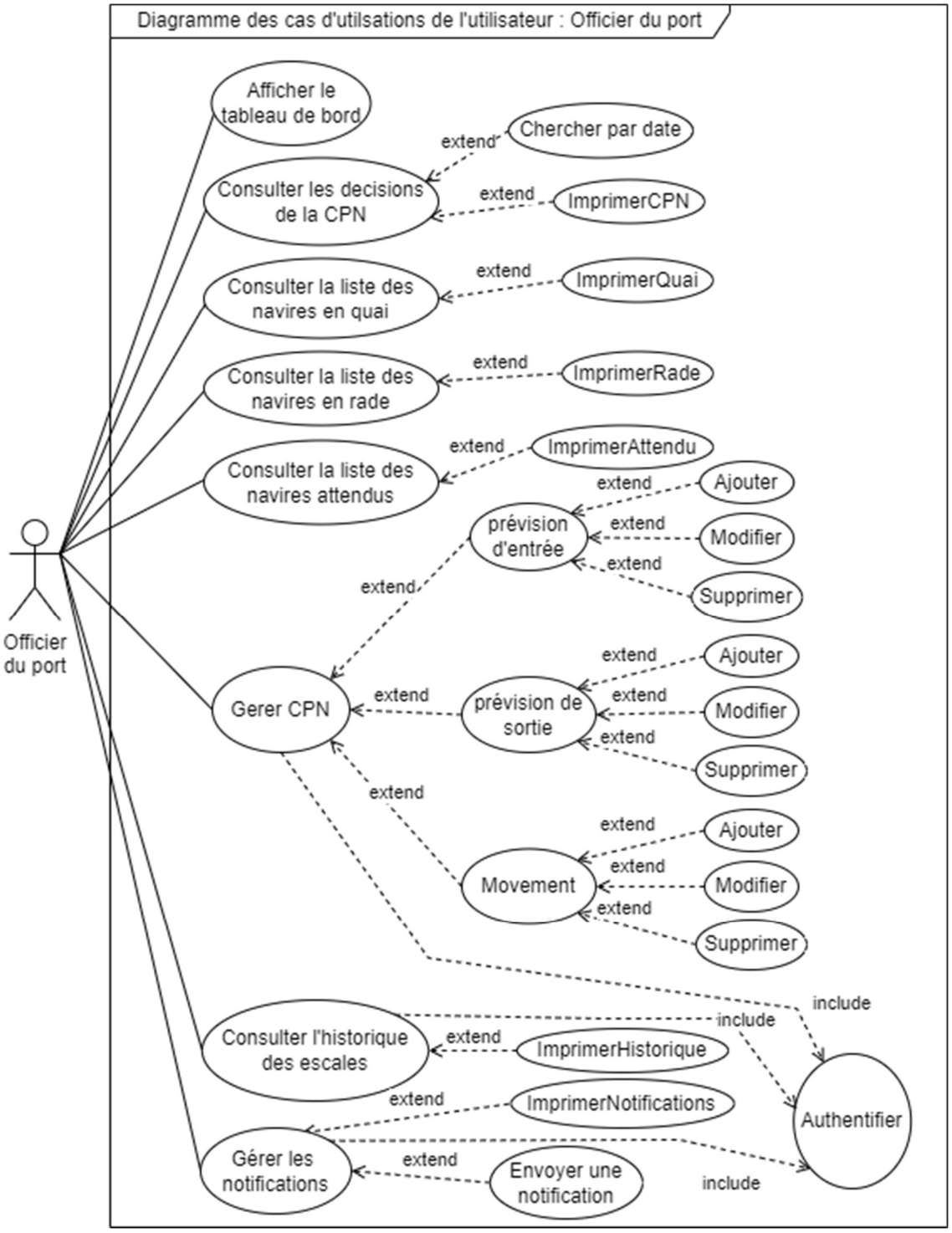

Figure 11 : Cas d'utilisation associés à l'officier de port

6.5. Cas d'utilisation associés à l'Officier de radio

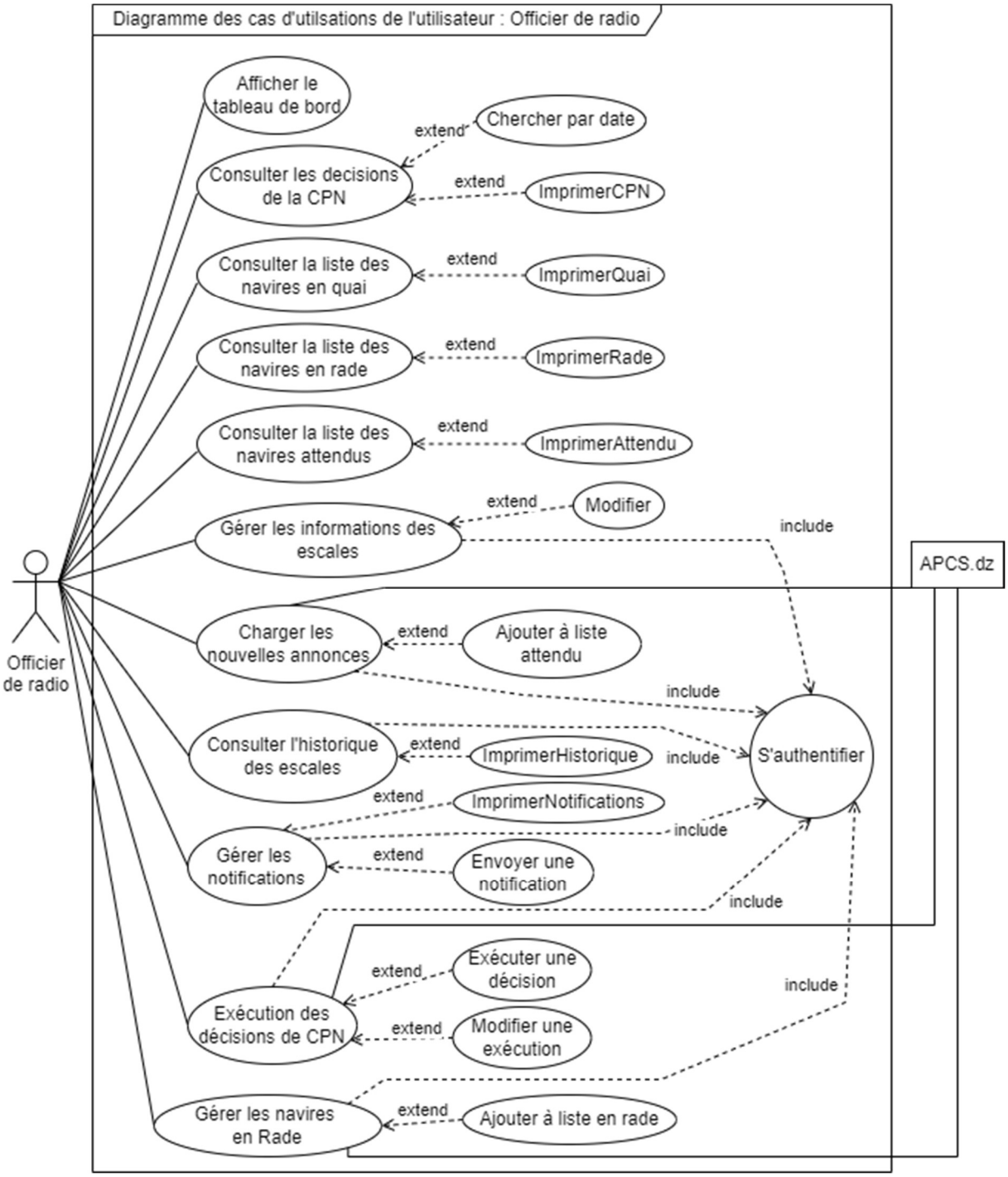

Figure 12 : Cas d'utilisation associés à l'officier de radio

6.6. Diagramme de cas d'utilisation global

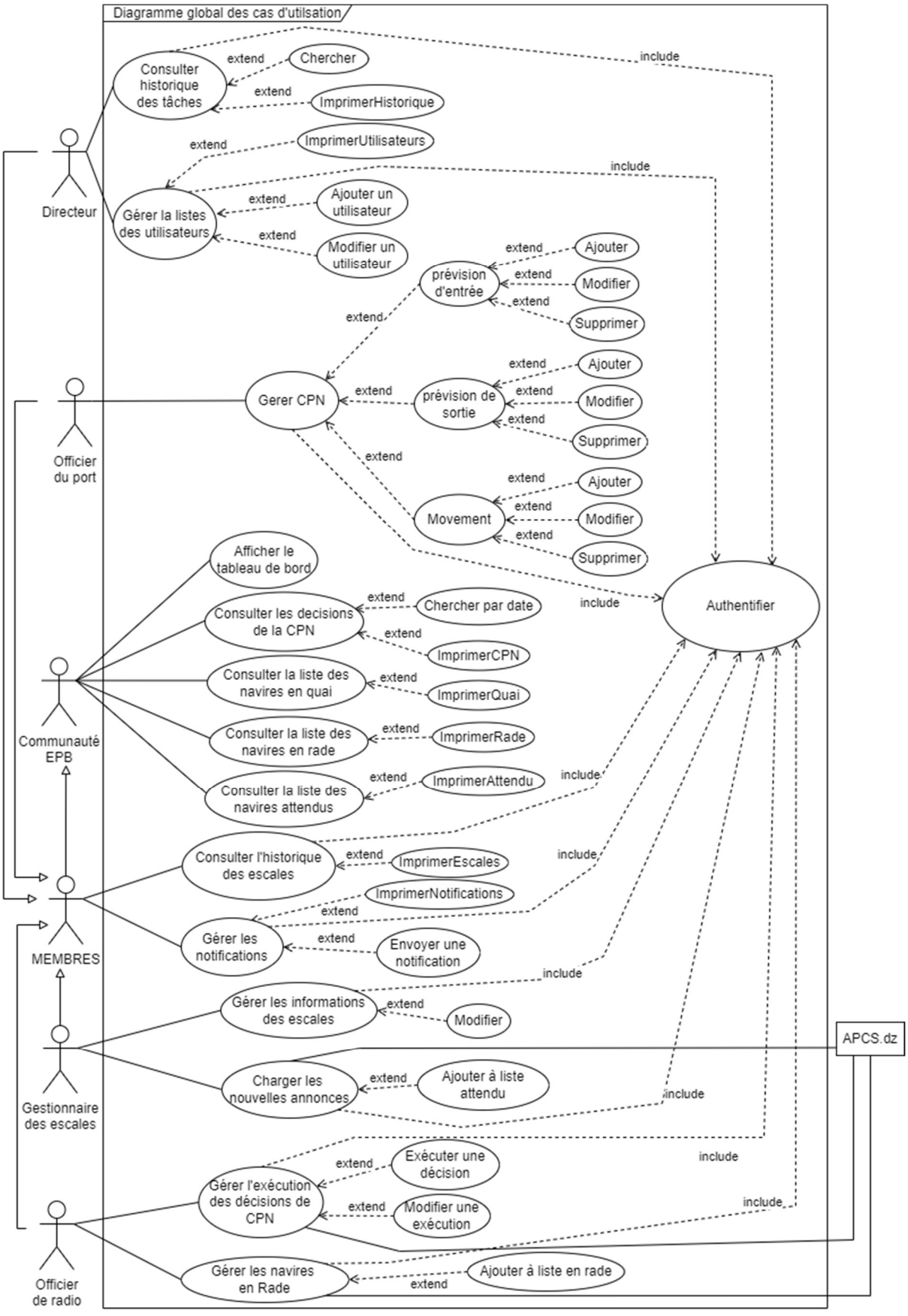

Figure 13 : Diagramme de cas d'utilisation global

7. Description textuelle des cas d'utilisation

Une autre définition d'un cas d'utilisation est qu'il s'agit d'un ensemble de scénarios de succès ou d'échec qui décrivent comment un acteur particulier utilise le système pour atteindre un objectif. Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système. Un cas d'utilisation doit avoir un début et une fin clairement identifiés. Il faut aussi préciser les variantes possibles tout en essayant d'ordonner séquentiellement les descriptions afin d'améliorer leur lisibilité [6].

7.1. Cas d'utilisation « Authentifier »

Cas d'utilisation	**Authentifier**
Objectif	Permettre aux utilisateurs d'accéder à ses privilèges dans le système
Acteurs	Directeur, Gestionnaire des escales, Officier de radio, Officier du port
Précondition	L'utilisateur doit avoir un compte.
Scénario nominal	1. Cliquer sur buton de Login. 2. Le système affiche la fenêtre d'authentification. 3. L'utilisateur introduit ses données de connexion et valide. 4. Le système vérifier l'existence du compte. 5. Le système vérifier que le compte est actif. 6. Le système confirme l'authentification.
Scénario alternatif	A1. Si le compte n'existe pas A2. Si le compte est inactif Dans les deux exceptions le système affiche un message d'erreur et reprend l'étape 2 du scénario nominal.
Post-condition	L'utilisateur est authentifié et diriger vers le tableau de bord

Tableau 3 : Description textuelle du cas d'utilisation "Authentifier"

7.2. Cas d'utilisation « Chargement des nouvelles annonces »

Cas d'utilisation	**Chargement des nouvelles annonces**
Objectif	Récupérer les annonces des navires de la plateforme APCS vers l'application
Acteurs	Gestionnaire des escales, Officier de radio
Précondition	L'acteur s'est bien authentifié
Scénario nominal	1. Cliquer sur le bouton Annonces. 2. Le système récupère les nouvelles annonces de APCS. (Nous n'avons pas l'accès à APCS.dz pour l'instant) 3. Le système récupère les annonces non valider de sa BDD. 4. Le système affiche les annonces.
Scénario alternatif	A1. Récupération de APCS échouée 1. Rediriger vers l'étape 3 du scénario nominal. A2. Récupération de la BDD échoué 1. Afficher un message d'erreur.
Post-condition	Récupération des annonces

Tableau 4 : Description textuelle du cas d'utilisation "Chargement des nouvelles annonces"

7.3. Cas d'utilisation « Ajouter à la liste attendu »

Cas d'utilisation	**Ajouter à la liste attendu**
Objectif	Valider l'annonce de navire et l'ajouter à la liste attendu
Acteurs	Gestionnaire des escales, Officier de radio
Précondition	L'acteur s'est bien authentifié et une annonce est affichée dans l'interface d'Annonces
Scénario nominal	1. Cliquer sur le bouton Valider sur une annonce afficher. 2. Le système affiche la fenêtre de validation. 3. L'utilisateur clique sur valider. 4. Le système ajoute le navire à la liste attendu. 5. Le système envoi l'état (attendu) du navire à APCS.dz. (Nous n'avons pas l'accès à APCS.dz pour l'instant) 6. Le système réaffiche l'interface des annonces.
Scénario alternatif	A1. Ajout à la liste attendu échouée 1. Rediriger vers l'étape 2 du scénario nominal. 2. Afficher un message d'erreur.
Post-condition	Navire ajouter à la liste Attendu

Tableau 5 : Description textuelle du cas d'utilisation "Ajouter à la liste attendu"

7.4. Cas d'utilisation « Valider des navires en Rade »

Cas d'utilisation	**Valider des navires en Rade**
Objectif	L'ajout de navire à la liste en Rade
Acteurs	Officier de radio
Précondition	L'acteur s'est bien authentifié
Scénario nominal	1. Sélectionner le bouton Valider en Rade. 2. Le système récupère les navires dans la liste Attendu. 3. Le système affiche les navires dans l'interface de validation. 4. Cliquer sur valider sur un navire affiché. 5. Le système affiche la fenêtre de validation. 6. Saisir les données et valider. 7. Le système ajoute le navire à la liste en Rade. 8. Le système envoi l'état (en rade) du navire à APCS.dz. (Nous n'avons pas l'accès à APCS.dz pour l'instant) 9. Le système réaffiche l'interface de validation.
Scénario alternatif	A1. Récupération des navires dans la liste Attendu échouée 1. Afficher message d'erreur. A2. Ajout à la liste en Rade échouée 1. Afficher un message d'erreur. 2. Rediriger vers l'étape 2 du scénario nominal.
Post-condition	Navire ajouter à la liste en Rade

Tableau 6 : Description textuelle du cas d'utilisation "Valider des navires en Rade"

7.5. Cas d'utilisation « Ajouter - Prévision d'entrée »

Cas d'utilisation	Ajouter - Prévision d'entrée
Objectif	L'ajout d'une prévision d'entrée d'un navire
Acteurs	Officier du port
Précondition	L'acteur s'est bien authentifié
Scénario nominal	1. Cliquer sur le bouton Ajouter dans la section Prévision d'entrée. 2. Le système affiche la fenêtre d'ajout. 3. Saisir les données et valider. 4. Le système ajoute la prévision à la BDD. 5. Le système réaffiche l'interface de Gérer CPN.
Scénario alternatif	A1. L'ajout de prévision a échouée 1. Afficher un message d'erreur. 2. Rediriger vers l'étape 5 du scénario nominal.
Post-condition	Prévision ajouter avec succès

Tableau 7 : Description textuelle du cas d'utilisation "Ajouter - Prévision d'entrée"

7.6. Cas d'utilisation « Modifier - Prévision d'entrée »

Cas d'utilisation	Modifier - Prévision d'entrée
Objectif	Modification d'une prévision d'entrée du navire
Acteurs	Officier du port
Précondition	L'acteur s'est bien authentifié et une décision non exécuter est affichée dans l'interface Gérer CPN
Scénario nominal	1. Cliquer sur le bouton Modifier sur une prévision afficher. 2. Le système affiche fenêtre de modification. 3. Modifier les données et valider. 4. Le système modifie la prévision dans la BDD. 5. Le système réaffiche l'interface Gérer CPN.
Scénario alternatif	A1. Modification de prévision échouée 1. Afficher un message d'erreur. 2. Rediriger vers l'étape 5 du scénario nominal.
Post-condition	Prévision modifier avec succès

Tableau 8 : Description textuelle du cas d'utilisation "Modifier - Prévision d'entrée"

7.7. Cas d'utilisation « Exécuter une décision »

Cas d'utilisation	**Exécuter une décision**
Objectif	Ajouter le navire à la liste des navires en quai (Entrée), ou le mettre en état Finie (Sortie) ou changer son poste en quai (mouvement).
Acteurs	Officier de radio
Précondition	L'acteur s'est bien authentifié et une décision est affichée dans l'interface d'Exécutions
Scénario nominal	1. Cliquer sur le buton Valider sur une décision afficher. 2. Le système affiche fenêtre de validation. 3. Remplir les données et valider. 4. Cliquer sur valider. 5. Le système modifie l'état du navire - Si c'est une entrée : Ajouter à la liste des navires en quai. - Si c'est une sortie : Mettre en état Finie. - Si c'est un mouvement : Changer son poste. 6. Le système envoi l'état ou le poste changer du navire à APCS.dz. (Nous n'avons pas l'accès à APCS.dz pour l'instant) 7. Le système réaffiche l'interface d'exécutions.
Scénario alternatif	A1. Modification de l'état du navire a échouée 1. Afficher un message d'erreur. 2. Rediriger vers l'étape 7 du scénario nominal.
Post-condition	Décision valider avec succès

Tableau 9 : Description textuelle du cas d'utilisation "Exécuter une décision"

8. Conclusion

Ce chapitre est consacré à la spécification et l'analyse des besoins de notre application. Nous avons identifié les acteurs et décrit les besoin fonctionnels et non fonctionnels, ensuite nous avons présenté les diagrammes de cas d'utilisation, ainsi que leurs descriptions textuelles. Le chapitre suivant sera dédié à la présentation de la conception et modélisation de notre application.

Chapitre III.

Conception

1. Introduction

La phase de conception est une phase très importante dans le cycle de développement, consiste à utiliser les informations collectées pour établir l'architecture de l'application à réaliser. Nous présentons les diagrammes de séquence, diagramme de classe, ainsi que le modèle relationnel sur lequel ce base la construction de notre base de données.

2. Diagrammes de séquence des cas d'utilisation

Le diagramme de séquence suit le diagramme de cas d'utilisation car ils sont complémentaires. Il permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets. En particulier, il montre également les objets participant aux interactions via les lignes de vie et les messages qu'ils échangent [7].

2.1. Diagramme de séquence du cas d'utilisation « Authentifier »

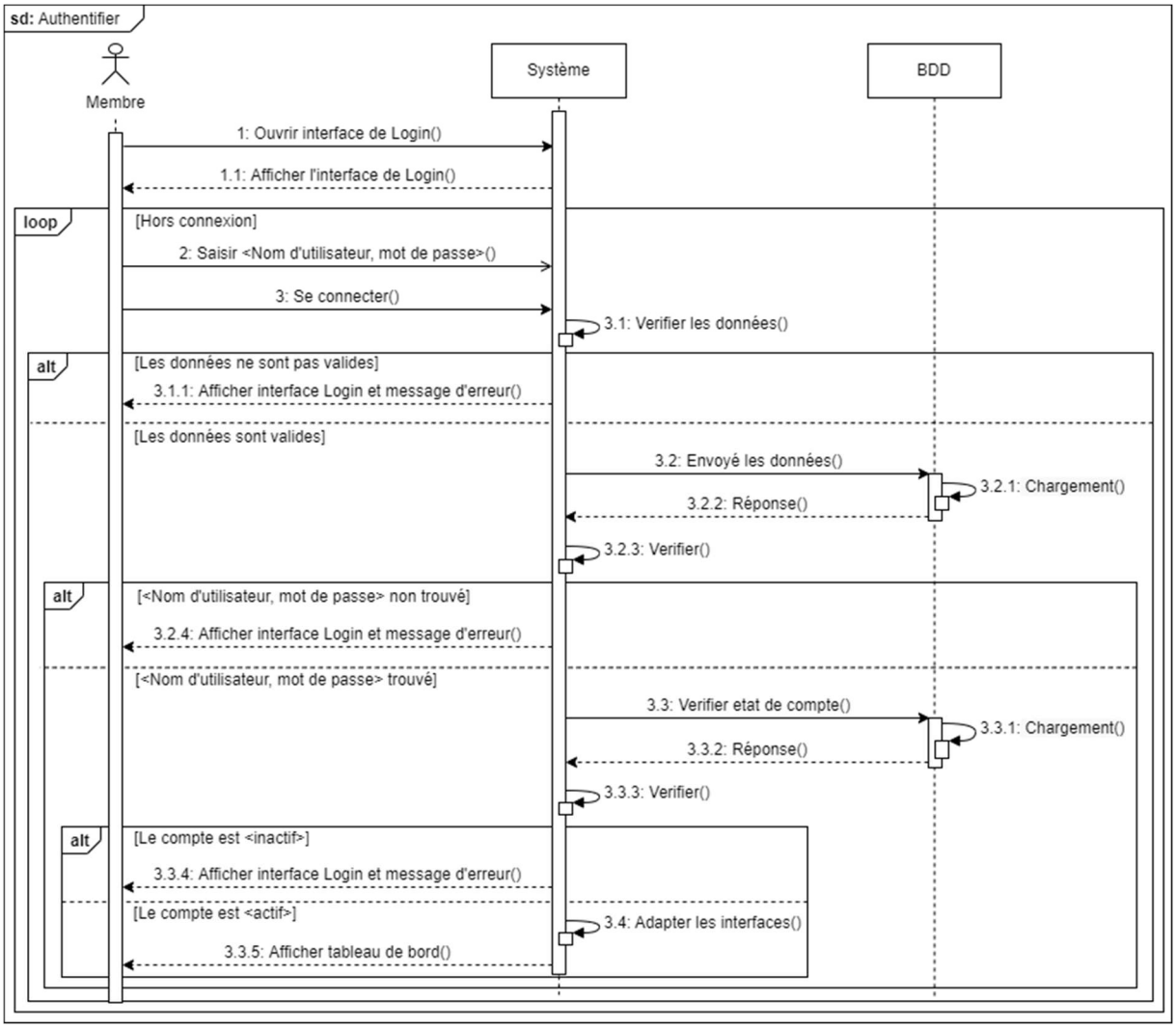

Figure 14 : Diagramme de séquence du cas d'utilisation « Authentifier »

2.2. Diagramme de séquence du cas « Charger les nouvelles annonces - Ajouter à la liste attendu »

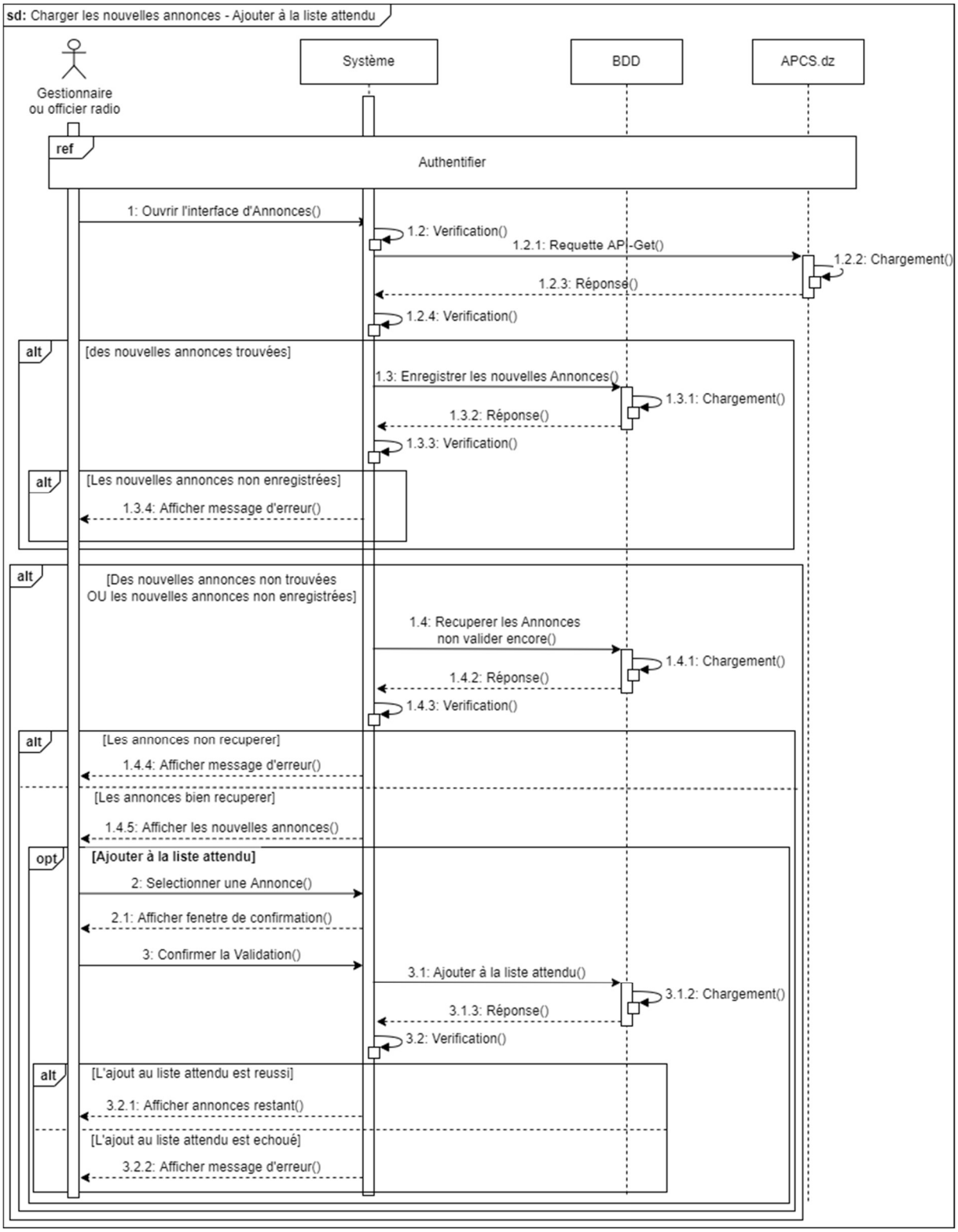

Figure 15 : Diagramme de séquence du cas « Charger les nouvelles annonces - Ajouter à la liste attendu »

2.3. Diagramme de séquence du cas « Gérer les navires en rade - Ajouter à la liste en rade »

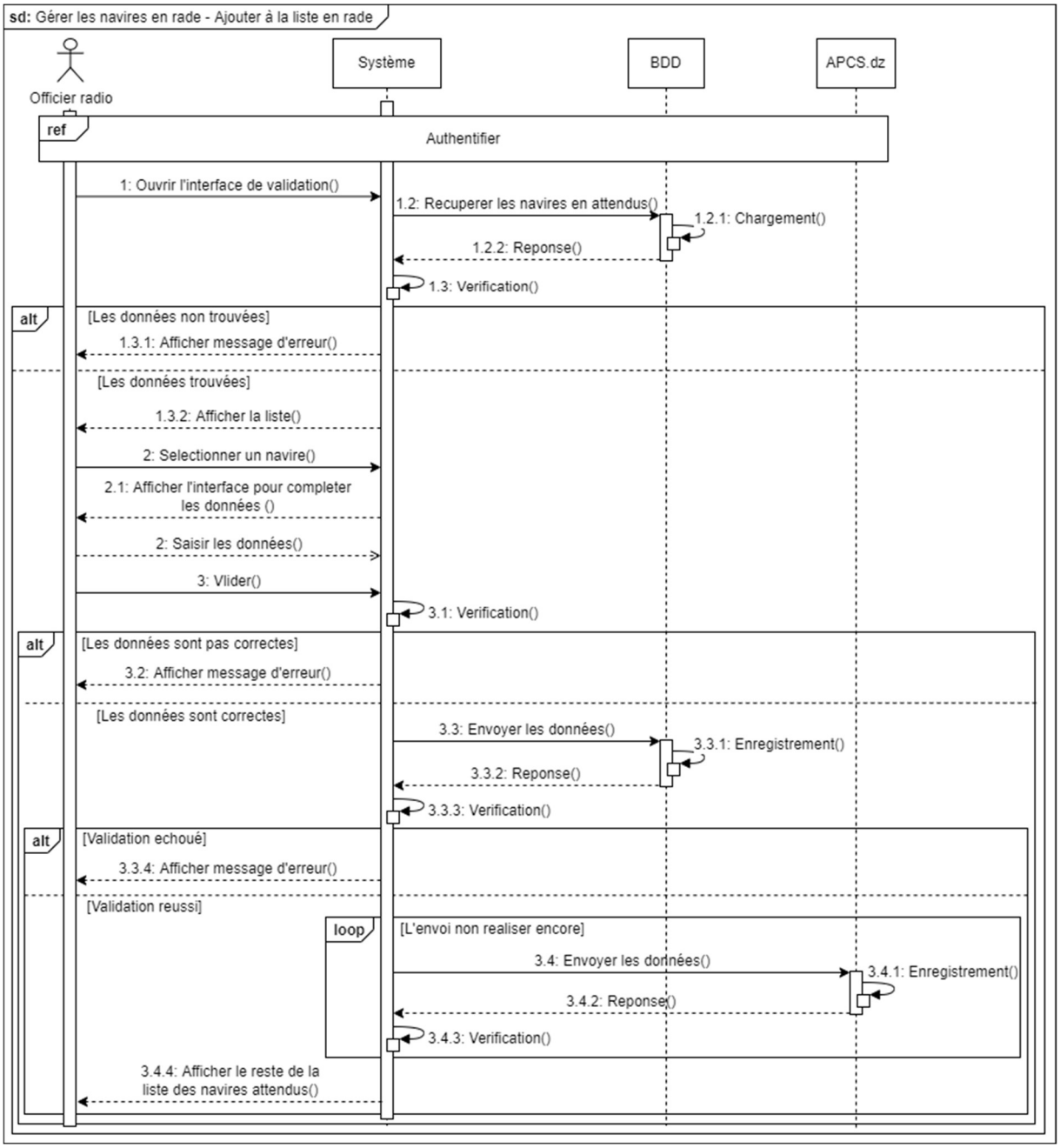

Figure 16 : Diagramme de séquence du cas « Gérer les navires en rade - Ajouter à la liste en rade »

2.4. Diagramme de séquence du cas « Prévision d'entrée »

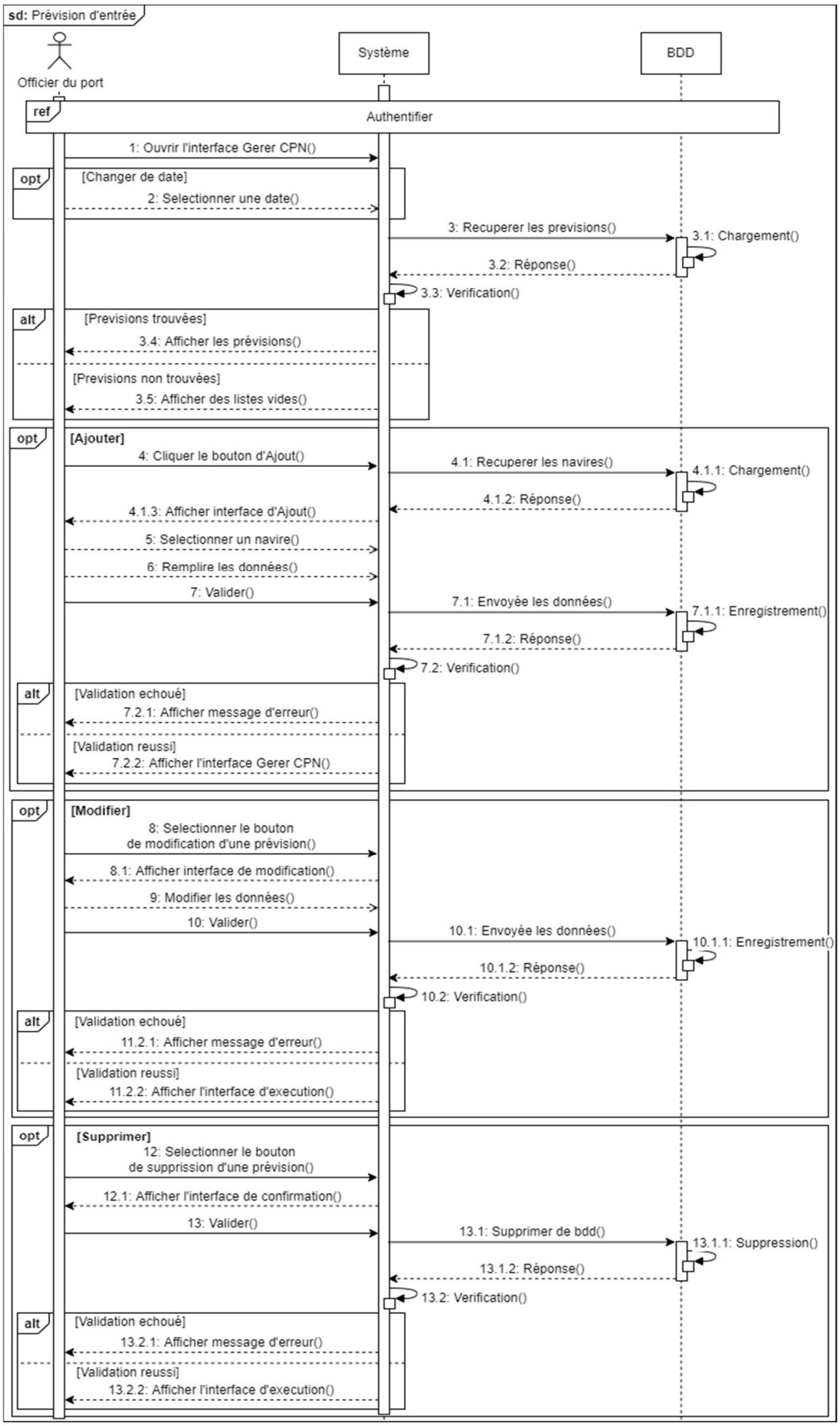

Figure 17 : Diagramme de séquence du cas « Prévision d'entrée »

2.5. Diagramme de séquence du cas « Exécuter une décision »

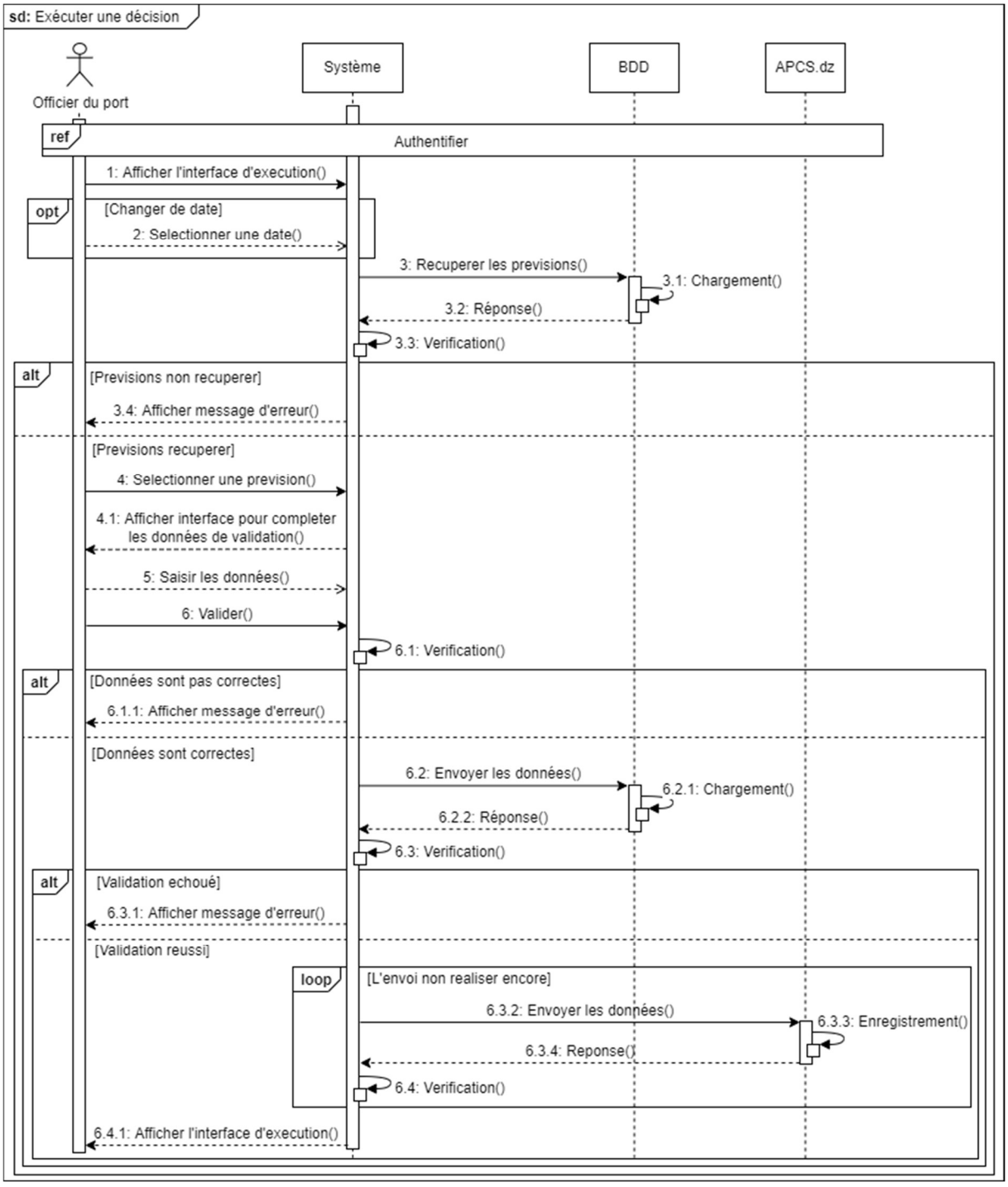

Figure 18 : Diagramme de séquence du cas « Exécuter une décision »

3. Diagramme de classe

Les diagrammes de classes expriment de manière générale la structure statique d'un système, en termes de classes et de relations entre ces dernières. La figure 19, illustre le diagramme de classe de notre application.

3.1. Présentation des classes et leurs attributs

Dans le tableau 10, nous allons décrire en détails les différentes classes schématisées.

Classe	Attribut	Type	Définition	Méthodes
Membres	id	Int	Identifiant de membre	- connectet() - deconnecter() - consulte_ historique_ navire()
	username	Varchar	Nom pour s'authentifier	
	password	Varchar	Mot de passe	
	nom	Varchar	Nom de l'utilisateur	
	prenom	Varchar	Prénom de l'utilisateur	
	dateNaissance	Date	Date de naissance de l'utilisateur	
	lieuNaissance	Varchar	Lieu de naissance de l'utilisateur	
	adresse	Varchar	Adresse de l'utilisateur	
	email	Varchar	Email de l'utilisateur	
	numero	Varchar	Numéro de téléphone de l'utilisateur	
	etat	Boolean	Etat de l'utilisateur	
	role	Varchar	Rôle de l'utilisateur	
Directeur	/	/	/	- cree_ utilisateur() - modifier_ utilisateur() - consulte_ historique_ utilisateur
Gestionnaire	/	/	/	- valider_ annonce() - modifier_ listes
Officier_port	/	/	/	- cree_cpn() - modifier_ cpn() - supprimer_ cpn()
Officier_radio	/	/	/	- valider_ executions() - modifier_ executons()

Notifications	idnotification	Int	Identifiant de notification	- envoyer() - consulter()
	a	Varchar	Nom d'utilisateur de récepteur	
	titre	Varchar	Titre de notification	
	contenus	Text	Contenue de notification	
	lu	Boolean	Etat de lecture du notification	
	cree_le	DateTime	Date et heure de création	
	vue_le	DateTime	Date et heure de lecture	
Historique_ utilisateur	idhistorique	Int	Identifiant de l'historie	- consulter (Directeur)
	realiserle	DateTime	Date et heure de tache	
	tache	Text	Detaille sur la tache	
Cpn	idCPN	Int	Identifiant de la CPN	/
	journee	Date	Jour de création de la prévision	
	heure	Time	Heure de création de la prévision	
	typeDecision	Varchar	Type de prévision	
	dateExecution	Date	Date de réalisation de la prévision	
	heureExecution	Time	Heure de réalisation de la prévision	
Sorties	details	Varchar	Consignes	/
Entrees	details	Varchar	Instructions à suivre	/
	auPoste	Int	Numéro du quai	
Movements	auPoste	Int	Numéro du quai	/
Executer	pilote	Varchar	Nom et prénom de pilote	/
	dateDebut_ Executon	Date	Date début d'exécution de prévision	
	dateFin_ Executon	Date	Date fine d'exécution de prévision	
	heureDebut_ Executon	Time	Heure début d'exécution de prévision	
	heureFin_ Executon	Time	Heure fine d'exécution de prévision	
Escales	numEscale	Varchar	Code d'escale	- modifier()
	etatEscale	Varchar	Etat d'escale (attendu, rade, qui, finie)	

Attendus	/	/	/	/
Qaui	poste	Int	Numéro du quai	/
	dateAccostage	Date	Date d'entrée en quai	
	heureAccostage	Time	Heure d'entrée en quai	
Rade	dateDHR	Date	Date de mouillage en rade	/
	heureDHR	Time	Heure de mouillage en rade	
	cordonnee	Varchar	Cordonnée de mouillage en rade	
Annonces	idAnnonce	Int	Identifiant de l'annonce	- valider()
	num_voyage	Varchar	Numéro de voyage	
	type_voyage	Varchar	Type de voyage	
	code_service	Varchar	Code de service	
	nature_escale	Varchar	Nature d'escale	
	consignataire	Varchar	Nom et prénom de consignataire	
	code_ consignataire	Varchar	Code de consignataire	
	armateur	Varchar	Armateur navire	
	nom_navire	Varchar	Nom de navire	
	type_navire	Varchar	Type de navire	
	imo_navre	Varchar	Numéro IMO de navire	
	call_sign	Varchar	Code de navire	
	mmsi	Varchar	Numéro MMSI de navire	
	largeur_navire	Float	La largeur de navire	
	longueur_navire	Float	La longeur de navire	
	pavillon	Varchar	Pays de navire	
	port_provenance	Varchar	Le port de provenance	
	port_destination	Varchar	Le port de destination	
	dernier_port_ touche	Varchar	Le port d'escale	
	eta	Datetime	Date et heure estimer de l'arrivée de navire	
	etd	Datetime	Date et heure estimer de sortie de navire	

	tirant_eau_ arrive_avant	Float	La hauteur de la partie avant immergée du navire	
	tirant_eau_ arrive_arriere	Float	La hauteur de la partie arrière immergée du navire	
	tirant_eau_ete	Float	La hauteur de la partie immergée du navire	
	marchandise	Varchar	La marchandise	
	poids	Float	Le poids du navire	
	type_ marchandise	Varchar	Type de marchandise	
	receptionnaire	Varchar	Le réceptionnaire	

Tableau 10 : Présentation des classes de l'application à réaliser

3.2. Diagramme de classe de l'application

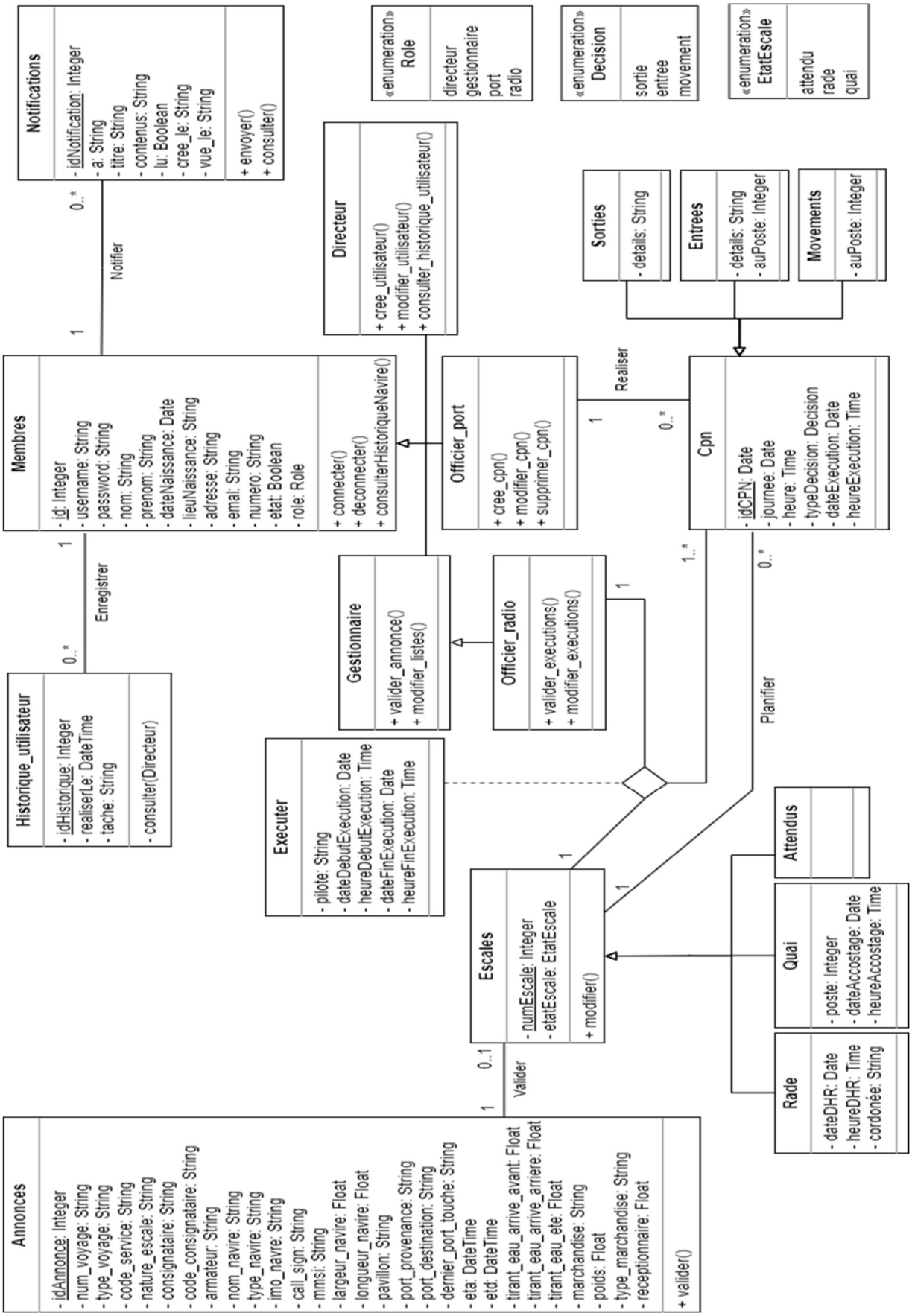

Figure 19 : Diagramme de classe

4. Présentation du modèle relationnel

En appliquant les règles de passages au diagramme de classes [8], nous obtenons le schéma relationnel suivant :

- **Utilisateurs** (id, username, password, nom, prenom, dateNaissance, lieuNaissance, adresse, email, telephone, etat, role).
- **Notifications** (idNotification, a, titre, contenus, lu, creeLe, vueLe, #idUtilisateur).
- **Historique_utilisateur** (idHistorique, realiserLe, tache, #idUtilisateur).
- **Annonces** (idAnnonce, num_voyage, type_voyage, code_service, nature_escale, consignataire, code_consignataire, Armateur, nom_navire, type_navire, imo_navire, call_sign, mmsi, largeur_navire, longueur_navire, pavillon, port_provenance, port_destination, dernier_port_touche, eta, etd, tirant_eau_arrive_avant, tirant_eau_arrive_arriere, tirant_eau_ete, marchandise, poids, type_marchandise, receptionnaire).
- **Escales** (numEscale, etatEscale, dateDHR, heureDHR, coordonnee, poste, dateAccostage, heureAccostage, #idAnnonce).
- **Cpn** (idCPN, #idNumEscale, #idUtilisateurs, journee, heure, dateExecution, heureExecution, details, auPoste, typeDecision).
- **Exécuter** (pilote, dateDebutExecution, heureDebutExecution, dateFinExecution, heureFinExecution, #idUtilisateur, #idCPN, #numEscale).

5. Conclusion

Ce chapitre nous a permis de décrire d'une manière globale et détaillée le fonctionnement du système afin d'en faciliter sa réalisation. Nous avons présenté les diagrammes de séquences, terminer par l'élaboration du diagramme de classe et le passage au modèle relationnel lui correspondant.

Nous entamons la partie implémentation qui fera l'objet du chapitre suivant.

Chapitre IV.

Réalisation

1. Introduction

Après avoir abordé la partie conception et analyse des besoins appropriée à notre projet, nous allons dans ce chapitre décrire la phase de réalisation. Ceci en spécifiant les outils, les langages et l'environnement de développement, ainsi que l'architecture de déploiement et tous les choix techniques que nous avons adoptés pour la réalisation de notre projet, nous clôturerons ce chapitre par la présentation des interfaces de notre application.

2. Outils, langages et environnements de développement

Pour mettre en place notre solution, nous avons utilisé des technologies fiables, performantes et extensibles, pour garantir la satisfaction des besoins fonctionnels ainsi que les exigences de qualité et de performance.

- **Laravel**

Il s'agit d'un Framework PHP open source entièrement développé en Programmation Orienté Objet (POO), qui respecte le principe du modèle de conception Modèle-Vue-Contrôleur(MVC). Ses sources sont hébergées sur Github. Nous avons adopté ce Framework pour le développement de la partie logique coté serveur (Back-end) de notre application car il apporte une sécurité de haut niveau, il est considéré comme l'un des meilleurs frameworks PHP, il est hautement évolutif et il dispose d'une très grande communauté.

Il regroupe différentes bibliothèques performantes, telles que le système d'authentification pour les connexions, système de vérification, système de migration de BDD, système d'autorisation et bien d'autres [9].

Figure 20 : Logo du Framework Laravel

- **Vue.JS**

C'est un Framework JavaScript open-source progressif, utilisé pour construire des interfaces utilisateur et des applications Web monopages (SPA - Single Page Application). L'un des frameworks front-end les plus populaires, il couvre la plupart des fonctionnalités communes nécessaires. Il s'appuie sur les standards HTML, CSS et JS et fournit un modèle de

programmation déclaratif et basé sur des composants qui nous aide à développer efficacement des interfaces utilisateur, qu'elles soient simples ou complexes [10].

Nous l'avons choisi pour la construction de côté client du notre application car il est très flexible, performant et adaptable progressivement.

Figure 21 : Logo du Framework VueJS

- **Vuetify**

Vuetify est un framework de composants d'interface utilisateur construit au-dessus de Vue.js. Basé sur la spécification « Material Design », nous l'utilisons car il fournit les outils dont nous avons besoin pour créer des composants et des interfaces riches et attrayantes, qui fonctionnent de manière transparente sur les smartphones et les tablettes ainsi que sur les navigateurs Web [11].

Figure 22 : Logo du Framework Vuetify

- **Node.JS**

C'est un environnement d'exécution JavaScript, basé sur le moteur JavaScript V8 de Chrome. Node.js qui est multiplateforme et open source qui exécute du code JavaScript côté serveur, en dehors du navigateur Web [12].

Figure 23 : Logo du système Node.JS

- **MySQL**

C'est un système de gestion de bases de données relationnelles, son rôle est d'enregistrer des données de manière organisée afin d'aider à les retrouver facilement [13].

Figure 24 : Logo du système MySQL

- **WampServer**

WampServer est une plate-forme de développement Web basée sur Windows pour les applications Web de type Wamp (Windows Apache MySQL, PHP), permettant d'utiliser localement le serveur Apache2, le langage de script PHP et la base de données MySQL. Il existe également PHPMyAdmin pour une gestion plus facile de la base de données [14].

Figure 25 : Logo de la plateforme WampServer

- **Git**

Git est le système de contrôle de version le plus utilisé aujourd'hui. Il permet d'enregistrer des versions de projets et de basculer entre eux et de manipuler le code de projet avec sécurité et flexibilité [15].

Figure 26 : Logo du système Git

- **GitHub**

GitHub est une plateforme open source de contrôle de version qui permet aux développeurs de stocker et de partager leur code publiquement ou en privé. Le site est aussi un espace de collaboration. Tous les utilisateurs peuvent contribuer aux projets publiés sur GitHub en suggérant des modifications. Une partie du succès de GitHub réside dans la façon dont le site a facilité ce processus. Pour éviter que les utilisateurs ne se dérangent en modifiant le programme en même temps, chacun télécharge le code sur son propre ordinateur, apporte des modifications et le publie sur GitHub après vérification.

Ce site est basé sur Git. Des espaces de discussion mis à la disposition de tous les développeurs pour discuter de leurs programmes et contributions [16].

Figure 27 : Logo du service GitHub

- **Visual Studio Code (VSC)**

Visual Studio Code est un éditeur de code source développé par Microsoft pour Windows, Linux et MacOs avec une prise en charge prête à l'emploi pour presque tous les langages de programmation populaires. Certains d'entre eux sont inclus par défaut comme JS, CSS et HTML, mais il dispose d'un riche écosystème d'extensions pour d'autres langages téléchargeables gratuitement. Il est léger et puissant lors de son utilisation. Il possède un terminal et un déboguer intégrer. Ses paramètres sont ouverts pour l'adapté aux utilisateurs [17].

Figure 28 : Logo de l'éditeur VSCode

- **Draw.io**

Il s'agit d'une application en ligne gratuite accessible à partir de navigateur et utilisée pour dessiner des diagrammes et des organigrammes. Il a une interface simple et facile à utiliser [18].

Figure 29 : Logo du logiciel Draw.io

3. Architecture et modèle de conception

- **MVC**

Le Model-View-Controller est une méthode d'organisation du développement d'applications Web permettant de séparer les différents concepts résultant de nos pages PHP. Nous trouvons Laravel pamis les frameworks qui utilisent cette architecture, comme son nom l'indique, divise l'application en trois composants logiques, qui gèrent des aspects de développement spécifiques pour rendre Laravel accessible, puissant et également fournit les outils dont nous avons besoin pour exécuter et maintenir des applications robustes [19].

- La partie Model, permet de gérer facilement et efficacement les manipulations de données et de créer automatiquement des relations entre les tables.
- La partie View, s'occupe de faire afficher ce que le Model renvoie et de recevoir toute interaction de l'utilisateur.

- La partie Controller, synchronise le Model et la View. Il traite les actions de l'utilisateur via des requêtes, demande des changements au Model, puis transmet les données à la View.
- Routing, indépendamment de l'architecture MVC, mais fait partie intégrante de tous les frameworks PHP, il permet de pointer via l'url vers des dossiers virtuels appelés routes, contrairement à l'architecture classique qui pointes vers des fichiers.

La figure ci-dessous nous décrit l'architecture MVC de Laravel :

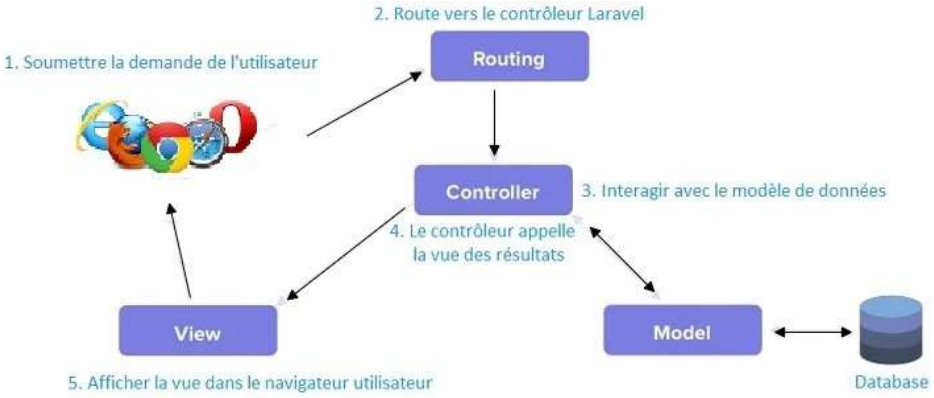

Figure 30 : Schéma de l'architecteur MVC

- **SPA**

Contrairement aux pages traditionnelles qui rechargent la page entière même pour des petites mises à jour. Une Single-page application (SPA) interagit avec l'utilisateur en réécrivant dynamiquement la page courante plutôt que de charger de nouvelles pages entières depuis un serveur. Cette stratégie a l'avantage de fournir une expérience utilisateur plus fluide en évitant les interruptions occasionnées par les chargements successifs des pages en provenance du serveur [20].

Pour notre application, nous avons adopté cette approche avec le framework Vue.JS.

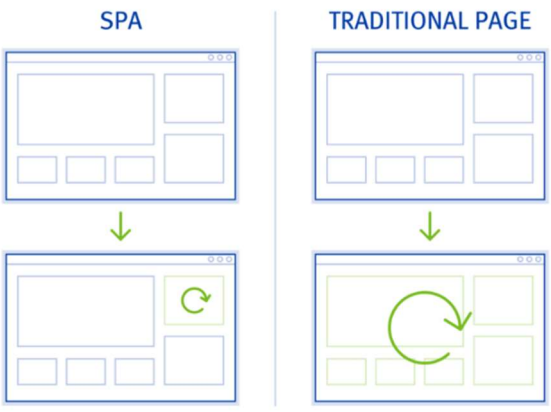

Figure 31 : Schéma de l'architecteur SPA

4. Présentation des interfaces de l'application réalisée

Dans ce qui suit, nous allons présenter quelques interfaces graphiques de notre application.

4.1. Interfaces associées au Communauté EPB

- **Tableau de bord**

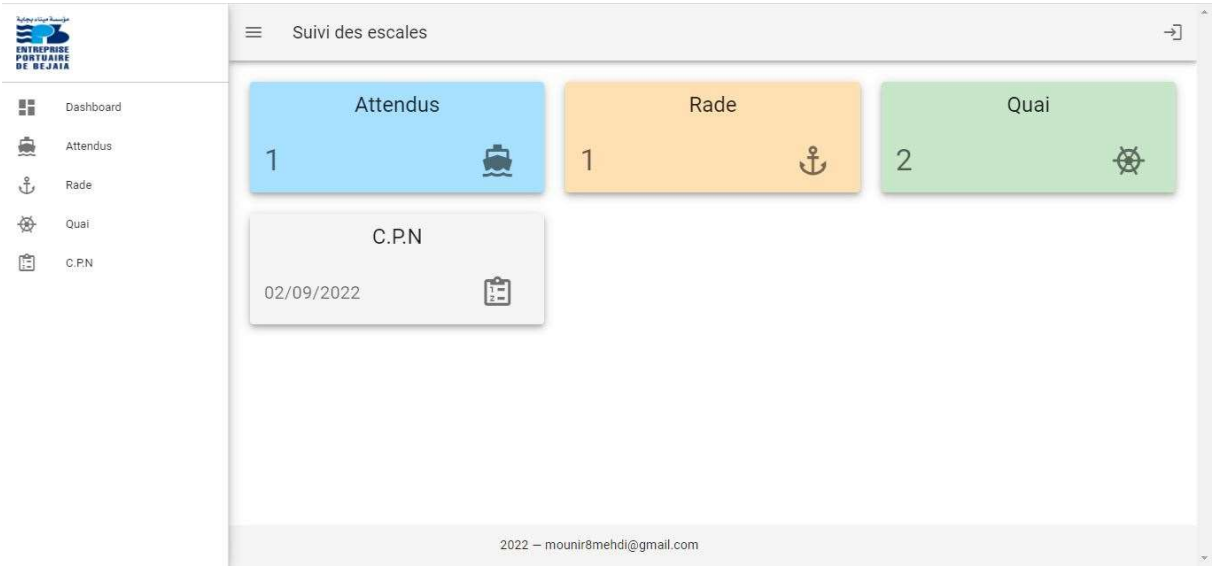

Figure 32 : Interface du tableau de bord

- **Consultation de la liste des navires attendus**

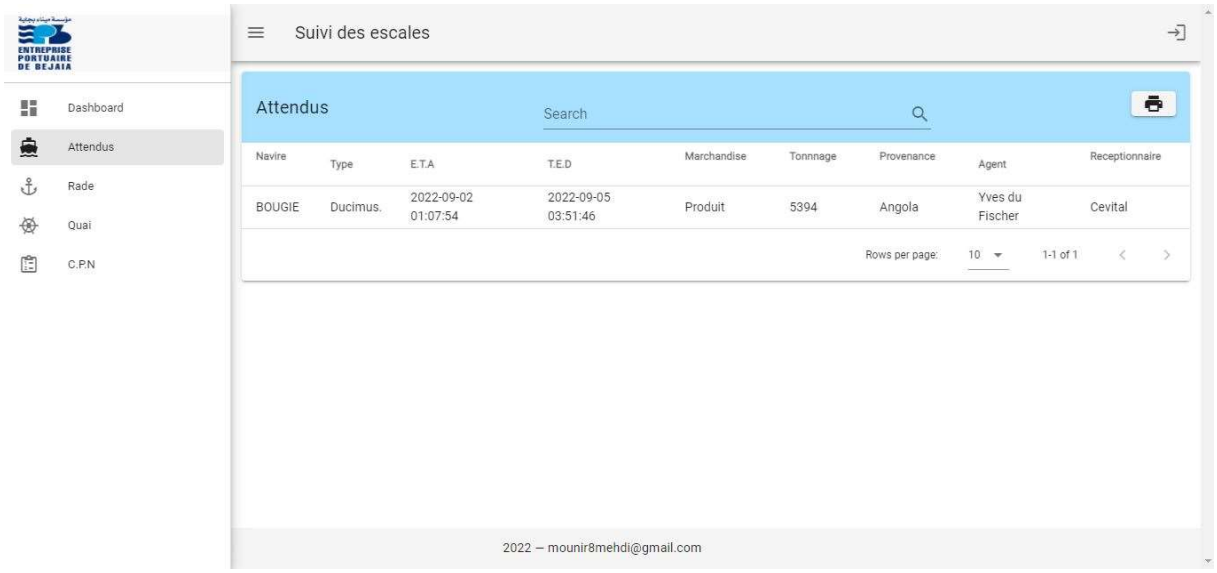

Figure 33 : Interface consultation de la liste des navires attendus

- **Consultation de la liste des navires en rade**

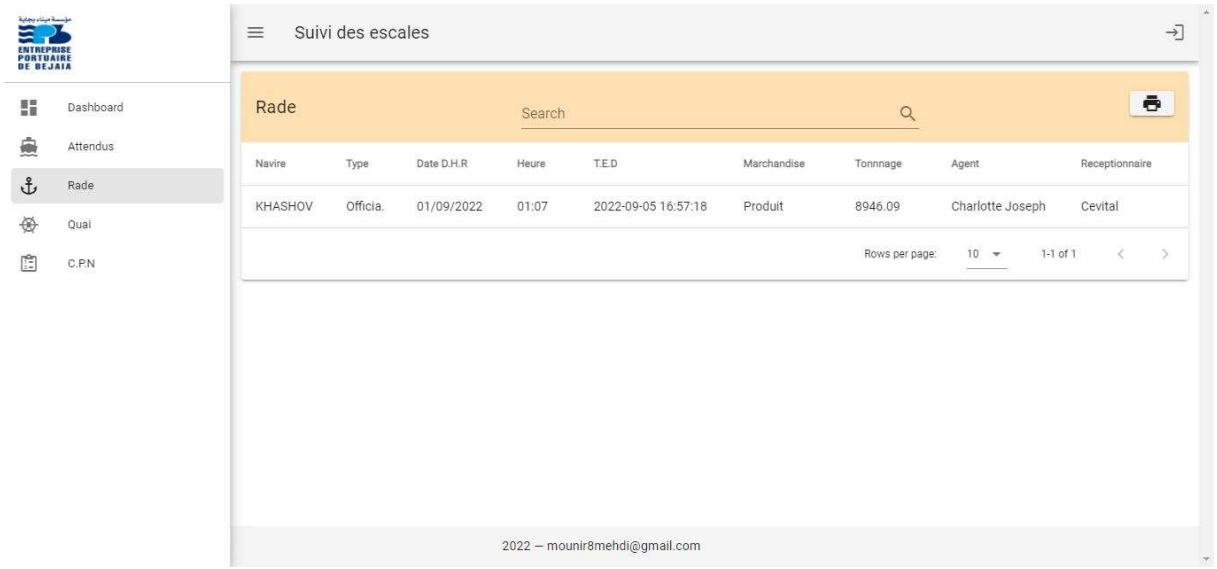

Figure 34 : Interface consultation de la liste des navires en rade

- **Consultation de la liste des navires en quai**

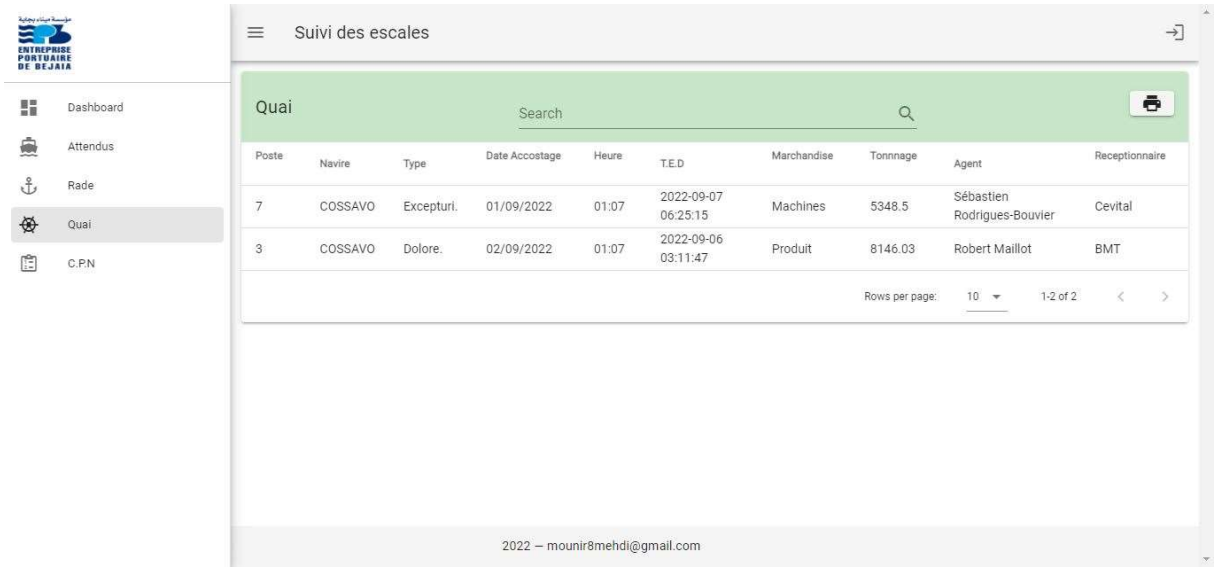

Figure 35 : Interface consultation de la liste des navires en quai

- **Consultation des CPN**

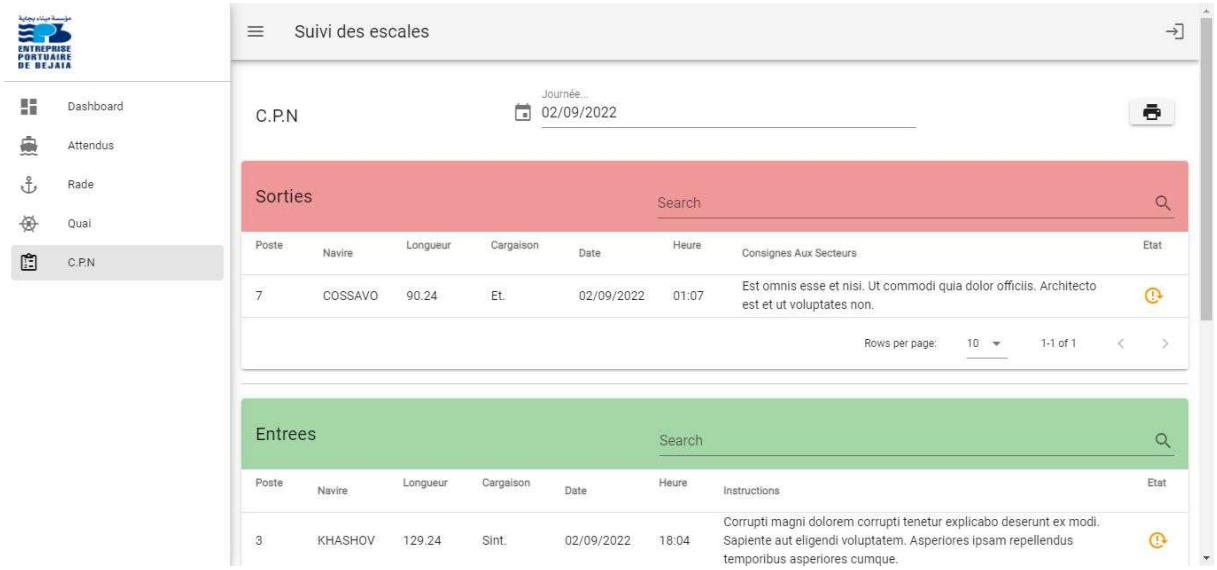

Figure 36 : Interface consultation des CPN

- **Authentification**

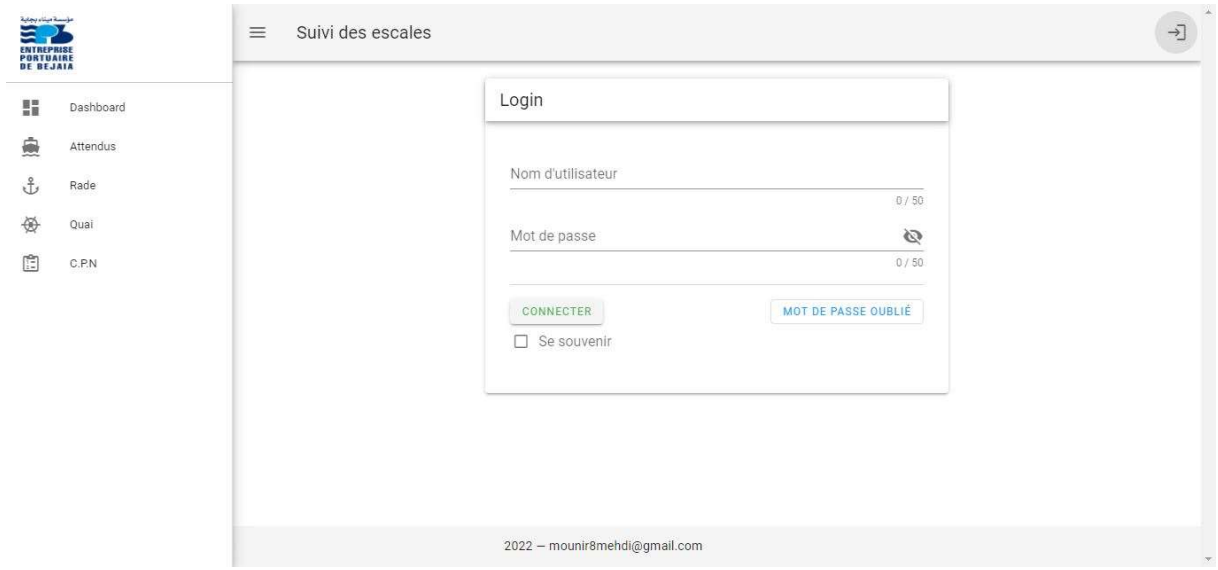

Figure 37 : Interface de connexion

4.2. Interfaces associées aux Membres

- **Profile**

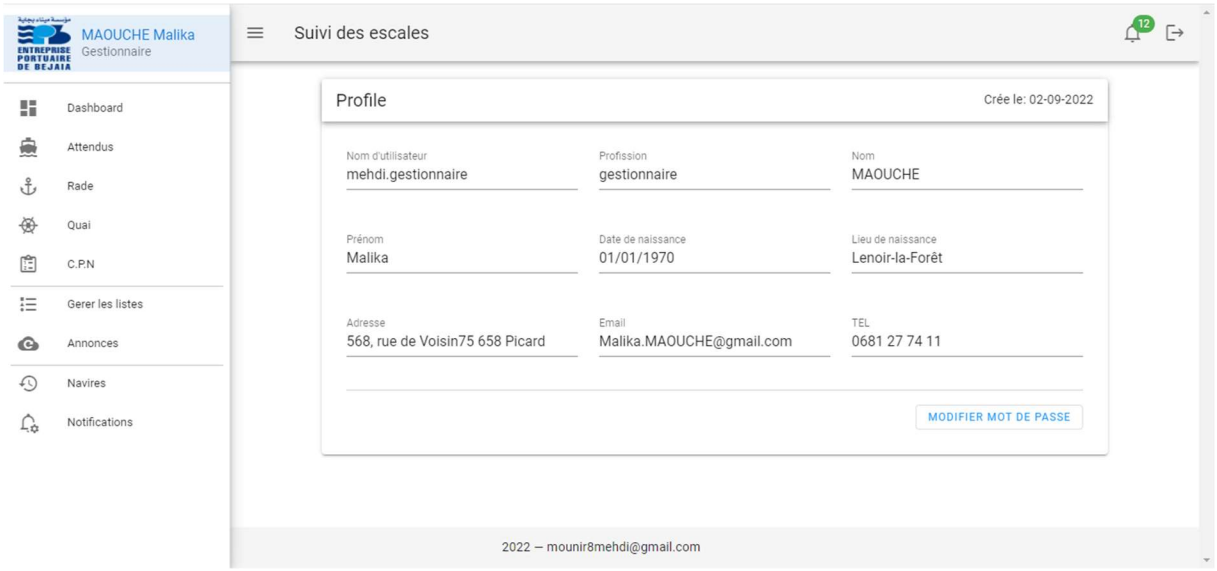

Figure 38 : Interface de profile

- **Consultation de l'historique des navires**

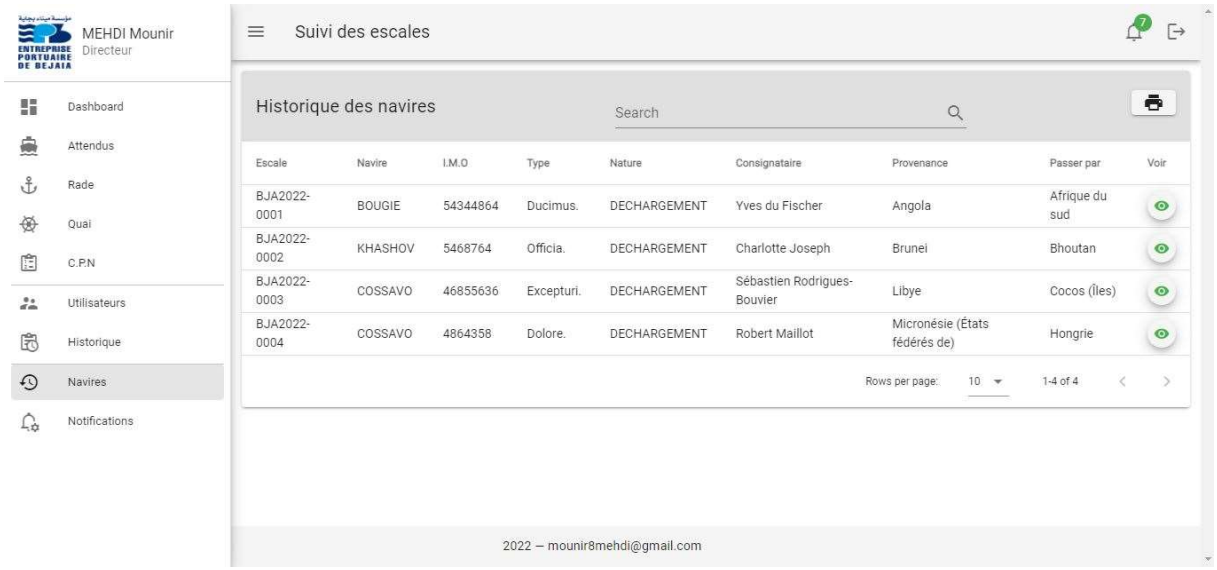

Figure 39 : Interface consultation de l'historique des navires

- **Consultation des détails de navire**

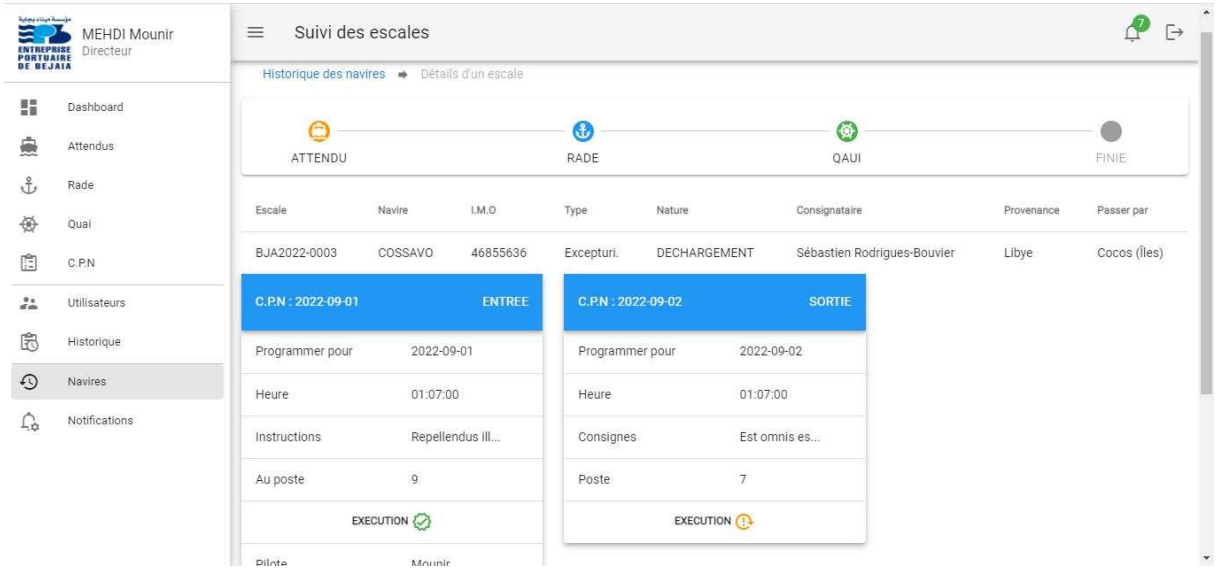

Figure 40 : Interface consultation des détails de navire

- **Consultation des notifications**

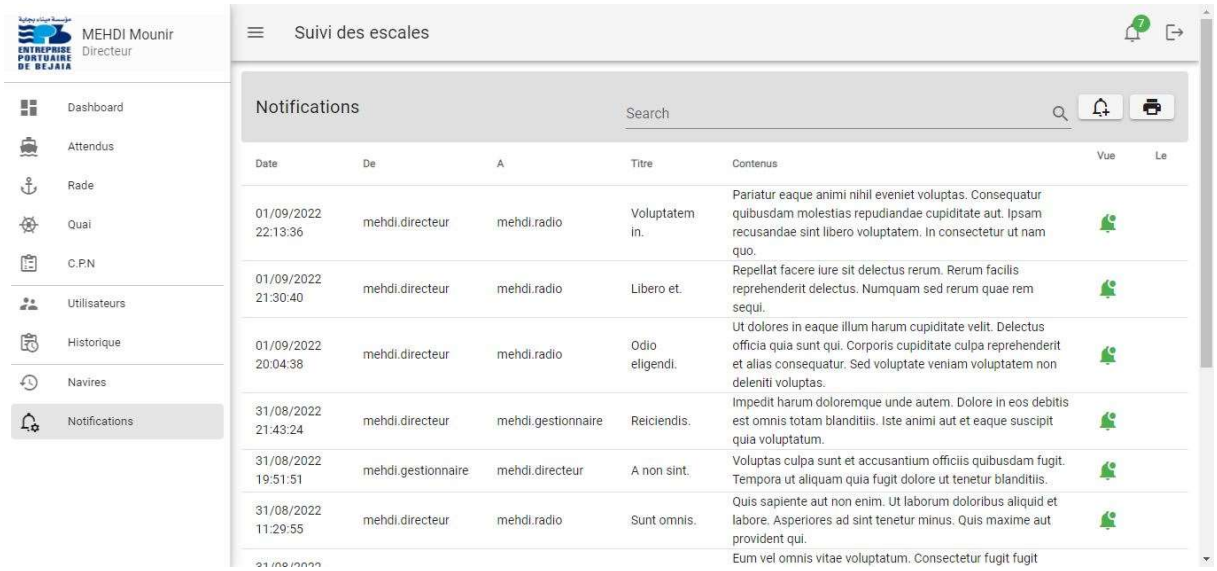

Figure 41 : Interface consultation des notifications

- **Création d'une notification**

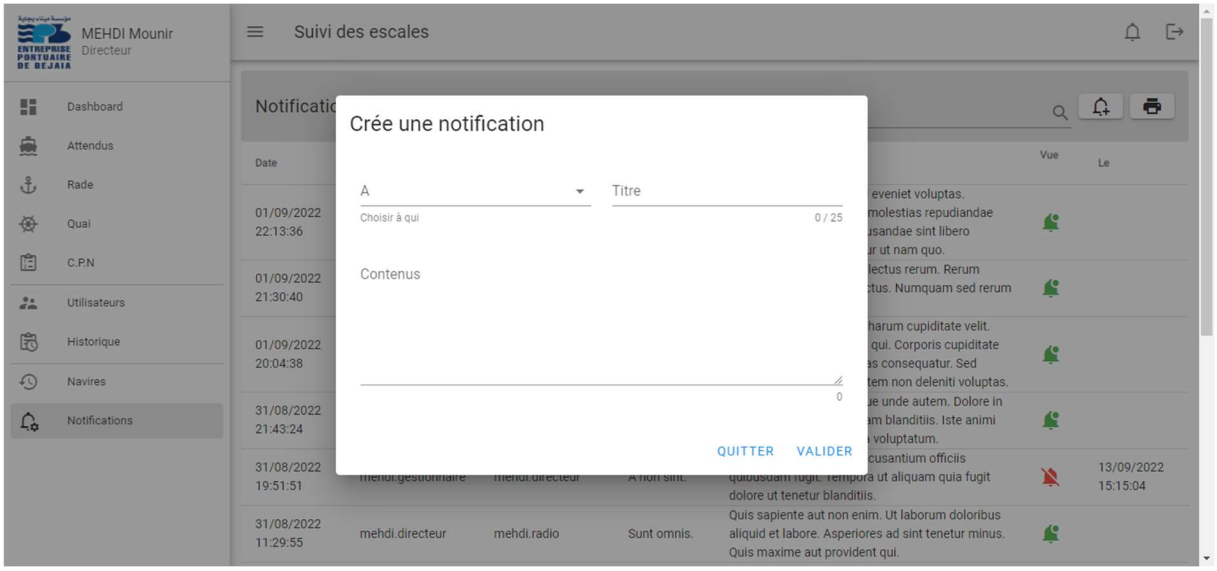

Figure 42 : Interface création d'une notification

4.3. Interfaces associées au Directeur

- **Consultation de la liste des utilisateurs**

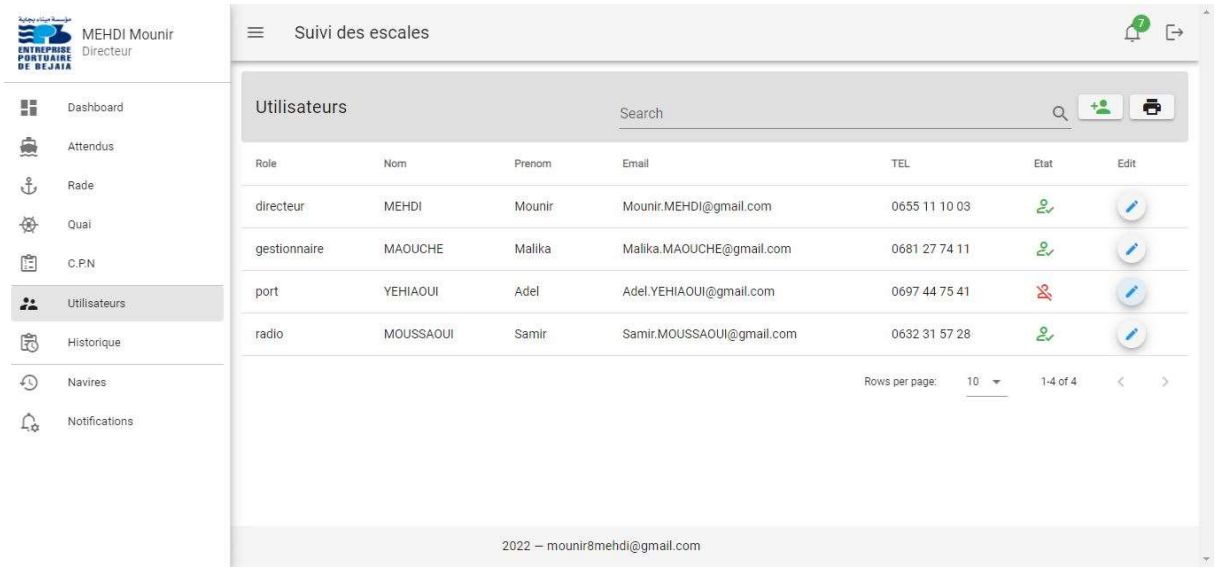

Figure 43 : Interface consultation de la liste des utilisateurs

- **Création d'un utilisateur**

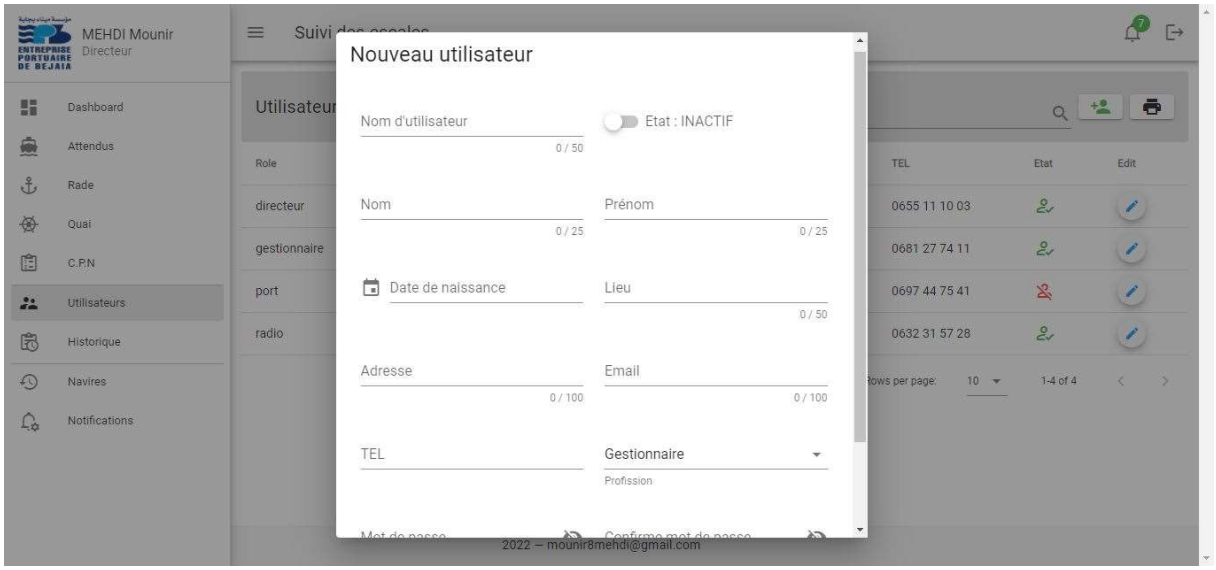

Figure 44 : Interface création d'un utilisateur

- **Consultation de l'historique des utilisateurs**

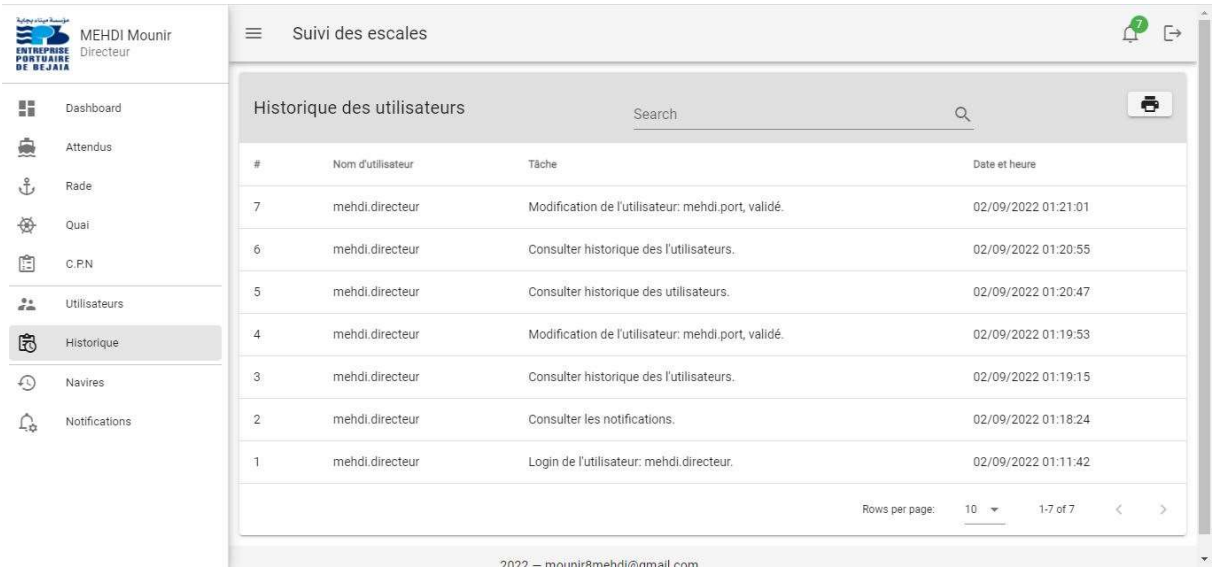

Figure 45 : Interface consultation de l'historique des utilisateurs

4.4. Interfaces associées au Gestionnaire des escales

- **Chargement des nouvelles annonces**

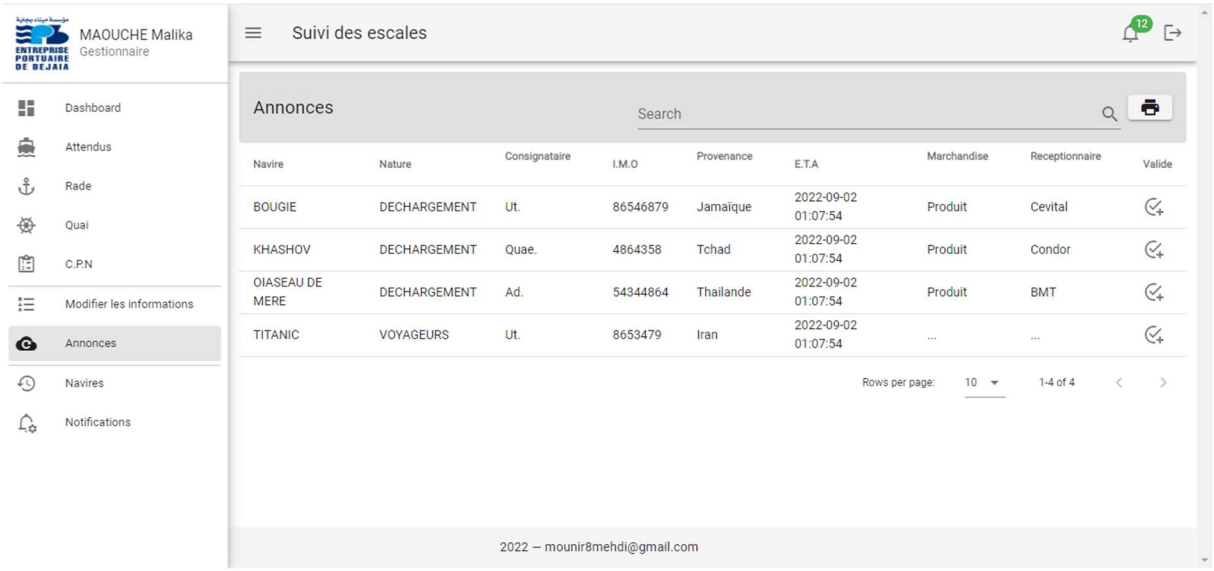

Figure 46 : Interface chargement des nouvelles annonces

- **Modifier les informations d'escales**

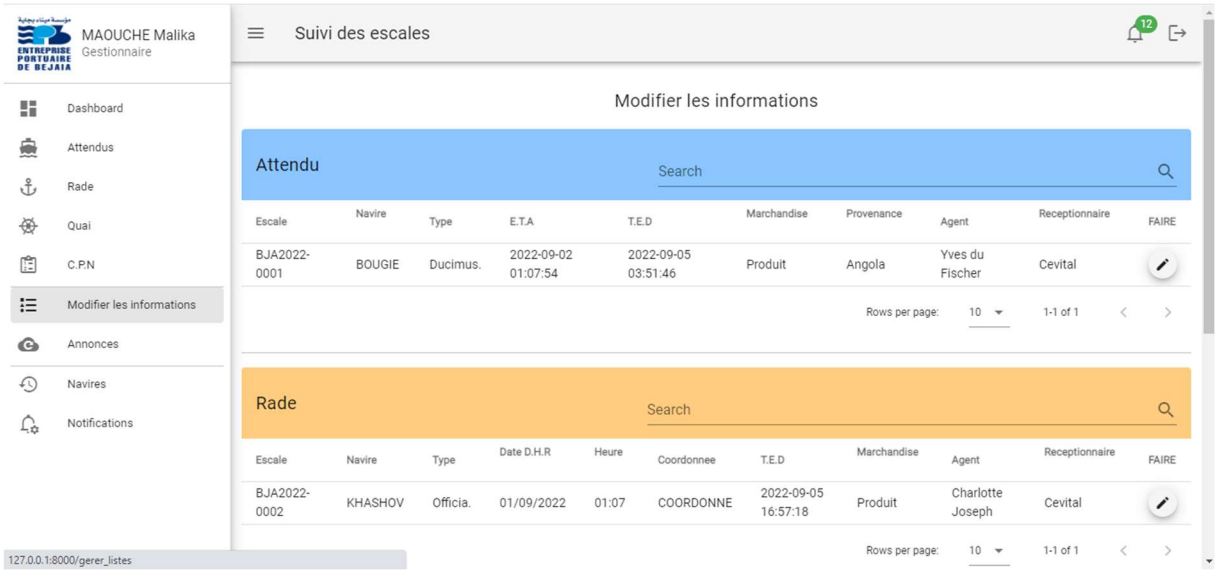

Figure 47 : Interface gestion des listes d'escales

4.5. Interfaces associées à l'Officier du port

- **Gestion de CPN**

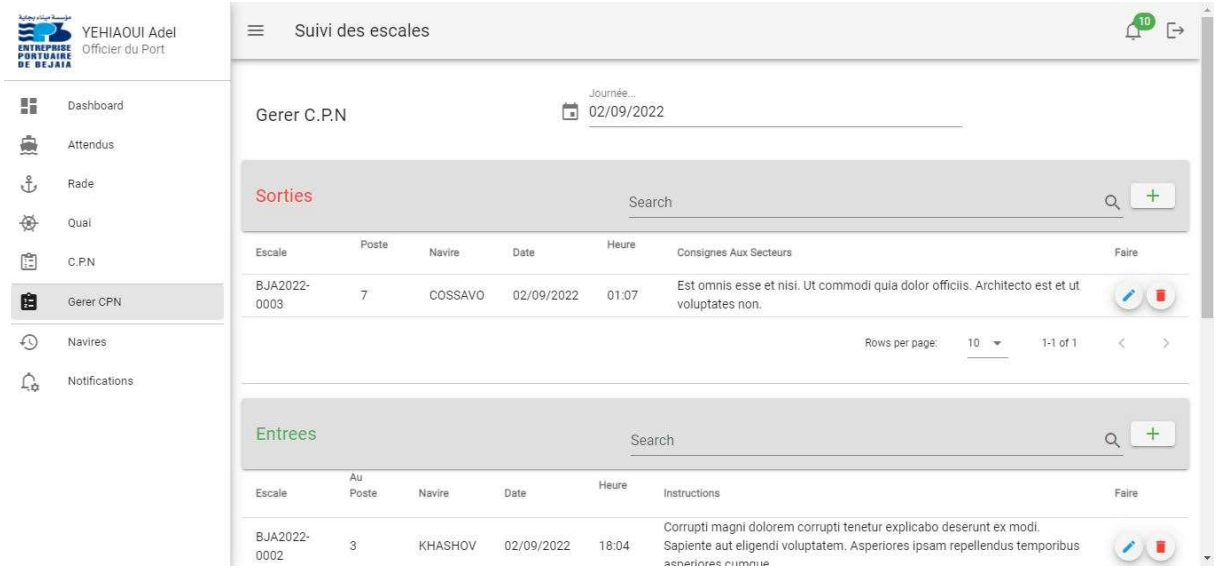

Figure 48 : Interface gestion de CPN

- **Ajout d'une nouvelle prévision**

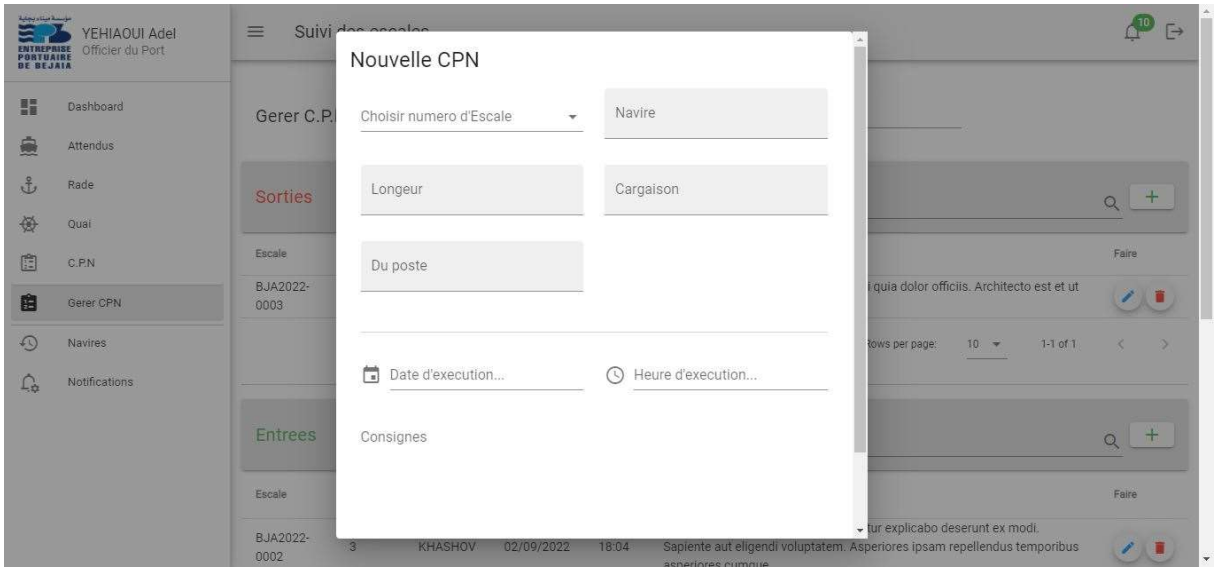

Figure 49 : Interface d'ajout d'une nouvelle prévision

4.6. Interfaces associées à l'Officier de radio

- **Validation des navires en rade**

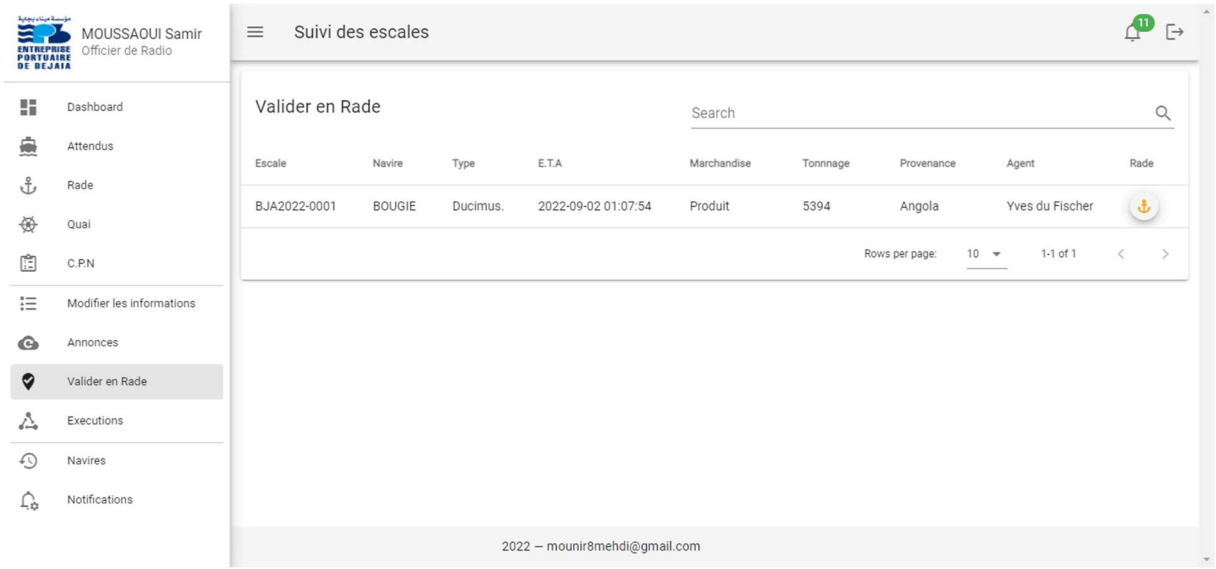

Figure 50 : Interface de validation des navires en rade

- **Valider un navire en rade**

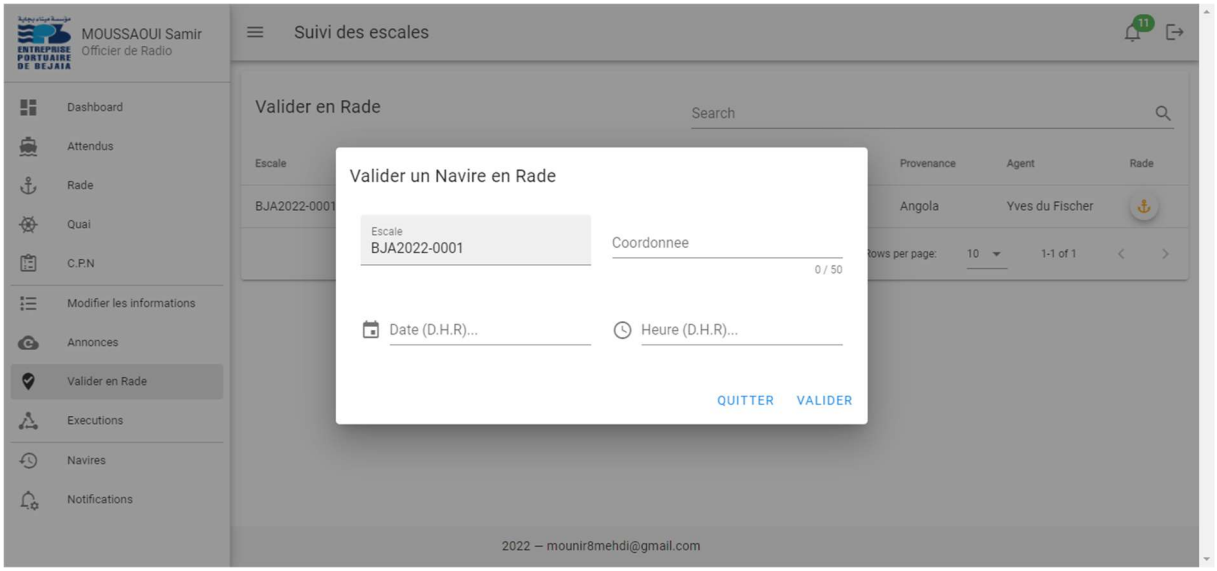

Figure 51 : Interface pour valider un navire en rade

- **Exécution des décisions de CPN**

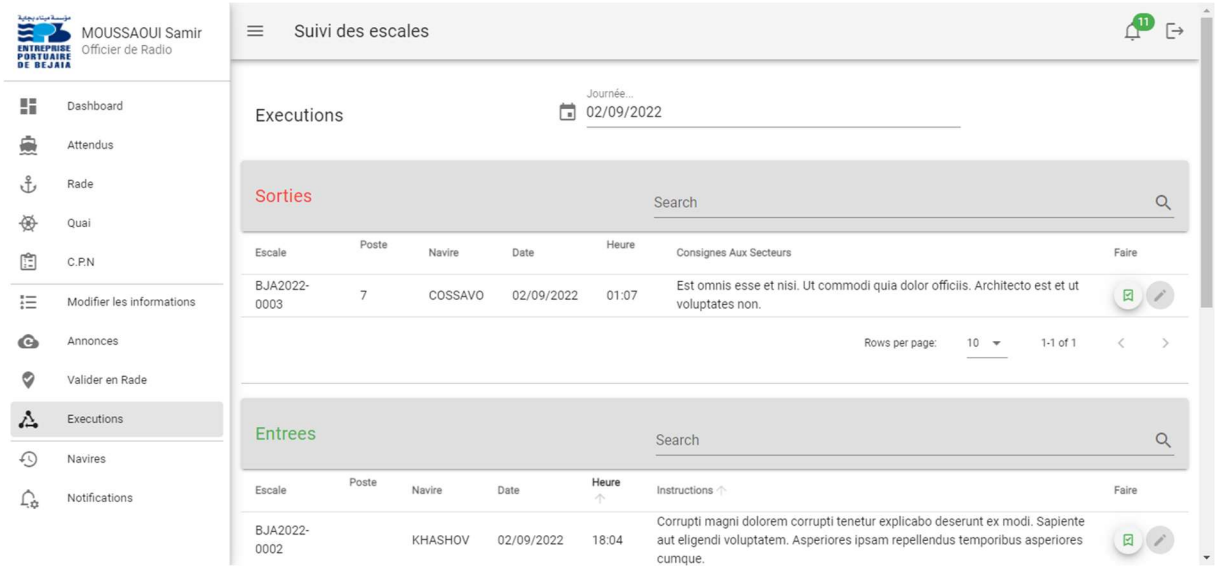

Figure 52 : Interface d'exécution des décisions de CPN

- **Validation d'une prévision**

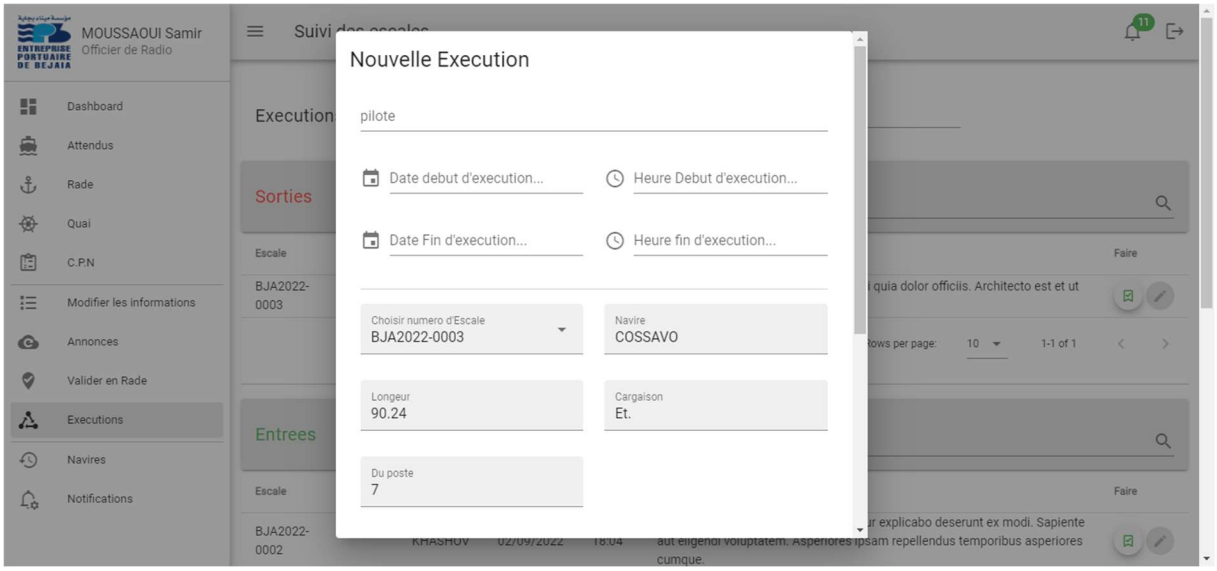

Figure 53 : Interface de validation d'une prévision

5. Conclusion

Dans ce dernier chapitre, nous avons présenté les aspects pratiques liés à la réalisation de notre application, à savoir les différents outils et langages utilisés lors du développement, ainsi que l'environnement de travail et l'architecture déployer. A la fin nous avons présenté quelques interfaces graphiques de notre application.

CONCLUSION GENERALE ET PERSPECTIVES

Durant ce projet de fin d'étude, nous avons présentés les étapes de conception et de réalisation d'une application web dynamique dédiée au suivi des escales des navires au sein de l'EPB. Ce projet nous a permis de mettre en pratique toutes les connaissances acquises au cours de notre formation universitaire.

En effet, nous avons fait une étude approfondie de l'ensemble des applications utilisées par le service capitainerie de l'EPB pour la gestion des escales. Nous avons par la suite critiqué ces systèmes et fait ressortir leur failles, faiblesses et limites dans le but de proposer une solution adéquate an accord avec l'organisme d'accueil. Notre application a été développée en utilisant un ensemble d'outil et de techniques innovantes tel-que : framework Laravel, VueJS, Vuetify.

Contrairement aux solutions existantes, l'application proposée permet non seulement d'être informer sur l'état des navires au sein du port (en attente, en rade, en quai, sortie) en temps réel, mais aussi d'alerter les membres du personnel de la capitainerie en cas de modification de toute information en relation avec les navires à savoir : le numéro du quai attribué à un navire, données relatives à un navire (le poids, tirant eau, etc.)

Comme tout projet, notre travail peut être compléter et amélioré par l'intégration de nouvelles fonctionnalités. Ainsi, nous envisageons comme perspectives, d'une part de relier l'application à la base nationale APCS.dz, définit en tant qu'acteur secondaire dans notre modélisation, à laquelle nous n'avions pas de droit d'accès mais aussi de programmer des alertes sur les smartphones des utilisateurs en cas de création d'une nouvelle notification.

En plus du service capitainerie, nous espérons élargir le champ d'utilisation de notre proposition à d'autres services de l'EPB par l'intégration de nouveaux modules permettant de prendre en charge la factorisation, la gestion des manifestes, la gestion des remorqueurs etc.

Ce projet a fait l'objet d'une expérience très intéressante, car elle nous a permis de nous familiariser avec de nouvelles notions d'une part, et d'améliorer nos connaissances et nos compétences dans le domaine de la programmation, et d'autre part avoir le sens de responsabilité et gestion de projets qui permet et facilite l'insertion dans le domaine professionnel et de développement.

BIBLIOGRAPHIE

[1] Source Entreprise Portuaire de Bejaia (EPB). [Accés le 25 4 2022].

[2] J. Ivar, B. Grady et R. James, "Le processus unifié de développement logiciel", Eyrolles, Éd., 2000.

[3] C. Morley, J. Hugues et B. Leblanc, "UML 2 pour l'analyse d'un système d'information", 4eme éd., Dunod, Éd., 2008.

[4] R. Pascal et V. Franck, "UML 2 en action: De l'analyse des besoins à la conception", 4eme éd., Eyrolles, Éd., 2007.

[5] K. Nasser, M. Dominique et P. Paré, "De Merise à UML (ancien tirage)", 1ère éd., Eyrolles, Éd., 1998.

[6] C. Alistair, "Rédiger des cas d'utilisation efficaces", Eyrolles, Éd., 2001.

[7] R. Pascal, "Uml 2.5 par la pratique", 8eme éd., Eyrolles, Éd., 2018.

[8] C. Stéphane, "Introduction au passage uml-relationnel : classes et associations", u. Formation, Éd., 2018, pp. 3-8.

[9] «Le framework PHP Laravel – la construction d'applications web pour tous,» Kinsta, 24 5 2022. [En ligne]. Available: https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-laravel/. [Accès le 1 9 2022].

[10] «Introduction | Vue.js,» Vue.js, [En ligne]. Available: https://vuejs.org/guide/introduction.html. [Accès le 4 9 2022].

[11] «Why you should be using Vuetify — Vuetify,» Vuetify, 2022. [En ligne]. Available: https://vuetifyjs.com/en/introduction/why-vuetify/#feature-guides. [Accès le 4 9 2022].

[12] «Introduction To Node.Js,» OpenJS Foundation, 2022. [En ligne]. Available: https://nodejs.dev/en/learn/. [Accès le 5 9 2022].

[13] «MySQL,» Oracle, 2022. [En ligne]. Available: https://www.mysql.com/fr/. [Accès le 7 9 2022].

[14] B. Romain, «WampServer,» 2022. [En ligne]. Available: https://www.wampserver.com/. [Accès le 7 9 2022].

[15] «Qu'est-ce que Git : devenez un pro de Git grâce à ce guide,» Bitbucket, 2022. [En ligne]. Available: https://www.atlassian.com/fr/git/tutorials/what-is-git. [Accès le 8 9 2022].

[16] «Github : tout savoir sur cette plateforme d'hébergement de code,» Groupe Publithings, 2022. [En ligne]. Available: https://www.lebigdata.fr/github-tout-savoir. [Accès le 8 9 2022].

[17] «Documentation for Visual Studio Code,» Microsoft, 2022. [En ligne]. Available: https://code.visualstudio.com/docs. [Accès le 8 9 2022].

[18] «Draw.io : un outil pour dessiner des diagrammes en ligne,» Tice Education, 2022. [En ligne]. Available: https://www.tice-education.fr/tous-les-articles-er-ressources/articles-internet/819-draw-io-un-outil-pour-dessiner-des-diagrammes-en-ligne. [Accès le 8 9 2022].

[19] H. Riadh, «Le modèle MVC de Laravel - apcpedagogie,» 2022. [En ligne]. Available: https://apcpedagogie.com/le-modele-mvc-de-laravel/. [Accès le 9 9 2022].

[20] «Qu'est-ce qu'une Application à Page Unique (Single Page App) ?,» mobiskill, 2022. [En ligne]. Available: https://mobiskill.fr/blog/conseils-emploi-tech/quest-ce-quune-single-page-application/. [Accès le 12 9 2022].

RÉSUMÉ

L’objectif de ce mémoire, est la conception et réalisation d’une application web pour le suivi des escales des navires, il est le résultat d'une étude approfondie de système existant lors du stage à l’Entreprise Portuaire de Bejaia. Nous avons utilisé le processus de développement UP, le langage de modélisation UML pour la conception, et pour la programmation nous avons choisi le framework Laravel pour le back-end qui respecte le modèle de conception MVC, et VueJS pour le front-end pour le développement avec l’architecteur SPA, et avec Vuetify pour styliser les composants, ainsi que MySQL en tant que système de gestion de base de données.

Mots-clés : Entreprise Portuaire de Bejaia, Application Web, Suivi des escales, Laravel, VueJS, UML.

ABSTRACT

The objective of this thesis is the design and production of a web application for monitoring ship calls, it is the result of an in-depth study of the existing system during the internship at the Port Company of Bejaia. We used the UP development process, the UML modeling language for the design, and for the programming we chose the Laravel framework for the back-end which respects the MVC design pattern, and VueJS for the front-end for the development with the SPA architect, and with Vuetify to style the components, as well as MySQL as a database management system.

Keywords : Bejaia Port Company, Web Application, Vessel Call Tracking, Laravel, VueJS, UML.