



MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA
RECHERCHE SCIENTIFIQUE
INSTITUT SUPÉRIEUR DES SCIENCES APPLIQUÉES ET DE
TECHNOLOGIE DE SOUSSE



RAPPORT DE PROJET DE FIN D'ÉTUDES

présenté en vue de l'obtention du

DIPLOÔME NATIONAL D'INGÉNIEUR EN INFORMATIQUE
Spécialité : Génie Logiciel

Élaboré par
Wyded Oueslati

Développement d'une application mobile pour la supervision et la gestion des missions du Robot "P-guard"

Encadrant académique :

M. Sami Achour

Co-encadrante académique :

Mme. Soumaya Bel Hadj Youssef

Encadrant professionnel :

M. Mohamed Ali Rouatbi

Président :

M. Farouk Cherif

Examinateuse :

Mme. Maha Khemaja

Réalisé au sein de



Code du projet : FI-GL22-025

Année universitaire : 2021-2022

Dédicace

Je dédie ce travail :

*À mon cher père **Fathi**, ma grande fierté, l'homme qui était et restera une source d'inspiration pour sa confiance, sa patience et son soutien tout au long de mes études.*

*À ma chère mère **Naziha**, la femme merveilleuse, symbole d'amour et de sacrifice, Qui m'a soutenu durant mes années d'études, qui m'a appris à aimer le bon comportement.*

*À mon cher Frère **Wyem** et mes chères Soeurs **Wifek** et **Wyjden** pour leur appui et leur encouragement tout au long de mes études et de mon cursus.*

*A tous les **membres de ma famille** sans aucune exception, sans lesquels je ne serais point ce que je suis aujourd'hui.*

À mes meilleurs amis, qui ont été là pour moi chaque fois que j'avais besoin, et qui ont rendu mes années universitaires si merveilleuses !

À tous ceux que j'aime...

Remerciements

A la fin de ce travail, j'adresse mes remerciements à tous ceux qui ont contribué, de près ou de loin à la réalisation de ce projet.

Ma gratitude va à toute l'équipe d'ENOVA ROBOTICS qui m'a accueilli, et plus particulièrement à **M. Mohamed Ali ROUATBI** pour son encadrement et les précieux conseils qu'il m'a apporté tout au long du stage, aux réunions qui ont rythmé les étapes de mon travail. C'est grâce à ses propositions d'amélioration et ses critiques qui ont contribué à la réalisation de ce travail.

J'exprime également ma profonde gratitude à Monsieur **Sami ACHOUR**, J'apprécie non seulement son encadrement Fructueux, ses conseils précieux, et sa disponibilité mais aussi sa bonne humeur et son aide durant mes années à l'ISSATSo. Il a été toujours là pour me guider et me soutenir.

Je profite aussi de cette occasion pour exprimer ma profonde gratitude à madame **Soumaya Bel Hadj Youssef** pour son soutien, le temps qu'elle m'a accordé, et ses conseils pour me guider tout au long de mon travail.

Mon dernier mot s'adresse aux **membres du jury** pour le temps qu'ils ont pris pour examiner mon travail, car c'est l'une des étapes les plus importantes de ma vie. Je vous remercie, ainsi que tous les professeurs de mon institution, pour l'éducation que j'ai reçue au cours des cinq dernières années.

Nomenclature

<i>2TUP</i>	Two Tracks Unified Process
<i>API</i>	Application Programming Interface
<i>COVID</i>	Coronavirus Disease
<i>GPS</i>	Global Positioning System
<i>HTML</i>	HyperText Markup Language
<i>HTTP</i>	Hypertext Transfer Protocol
<i>IDE</i>	Integrated Development Environment
<i>IHM</i>	Interface Homme-Machine
<i>iOS</i>	iPhone Operating System
<i>IP</i>	Internet Protocol
<i>JSON</i>	JavaScript Objet Notation
<i>MVC</i>	Model View Controller
<i>OPCMA</i>	Open Consulting Management
<i>OS</i>	Operating System
<i>P – Guard</i>	Pearl Guard
<i>ROS</i>	Robot Operating System
<i>RTK</i>	Real Time Kinematic
<i>RTSP</i>	Real Time Streaming Protocol
<i>SA</i>	Société Anonyme
<i>SDK</i>	Software Development Kit
<i>UGV</i>	Unmanned Ground Vehicle
<i>UML</i>	Unified Modeling Language
<i>UP</i>	Unified process
<i>VLC</i>	VideoLan Client

Table des matières

Table des figures	vi
Liste des tableaux	i
Introduction Générale	1
1 Cadre général du projet	3
1.1 Présentation de l'organisme d'accueil	4
1.1.1 Présentation	4
1.1.2 Organigramme de la société	4
1.1.3 Produits réalisés par la société	5
1.2 Présentation du projet	8
1.2.1 Cadre du projet	8
1.2.2 Etude de l'existant	9
1.2.3 Problématique	14
1.2.4 Objectifs	15
1.2.5 Solution envisagée	16
1.3 Méthodologie de gestion de projet	18
1.3.1 Méthodologie UP	18
1.3.2 Méthodologie 2TUP	19
1.3.3 Choix de la méthodologie	20
2 Etude préliminaire : Spécification des besoins fonctionnels et techniques du projet	22
2.1 Modélisation du contexte	23
2.1.1 Identification des acteurs	23
2.1.2 Identification des messages échangés	24
2.2 Branche fonctionnelle	24
2.2.1 Besoins fonctionnels	24
2.2.2 Diagramme de cas d'utilisation global	26
2.2.3 Raffinement des cas d'utilisation	27
2.2.4 Analyse	34
2.3 Branche technique	37
2.3.1 Besoins non fonctionnels	37
2.3.2 Contraintes techniques	38

2.3.3	Conception générique	40
3	Conception détaillée	43
3.1	Prototype des interfaces utilisateur	44
3.2	Architecture logique de l'application	45
3.2.1	Introduction à l'architecture MVC	45
3.2.2	Architecture GETX	46
3.3	Vue statique de l'application : Diagramme de classes	47
3.4	Vue dynamique de l'application	49
3.4.1	Diagrammes de séquence	49
3.4.2	Diagramme d'états-transitions	58
4	Réalisation	60
4.1	Environnement de développement	61
4.1.1	Environnement matériel	61
4.1.2	Environnement logiciel	61
4.2	Application développée	66
4.2.1	Les interfaces de bord :	66
4.2.2	Interface d'authentification :	67
4.2.3	Interface d'accueil :	67
4.2.4	Interface des scénarios	70
4.2.5	Interface des caméras	72
4.2.6	Interface des missions	73
4.2.7	Interface de navigation libre	75
4.2.8	Interface de paramètres	75
4.2.9	Interface de "connexion perdue"	76
4.3	Défis soulevés	77
4.4	Validation des besoins non fonctionnels	78
4.5	Tests et validation	78
Conclusion et perspectives		80
Références		83

Table des figures

1.1	Organigramme d'ENOVA ROBOTICS	5
1.2	Le robot Minilab	5
1.3	Le robot Covea	6
1.4	Le robot Oggy	6
1.5	Le robot P-Guard	7
1.6	IHM Qt	10
1.7	IHM du robot E-vigilante	12
1.8	IHM du robot UGV	13
1.9	Station de recharge du robot P-Guard	17
1.10	Schéma illustrant le fonctionnement global du système	18
1.11	Méthodologie 2TUP	20
2.1	Illustration de la phase de l'étude préliminaire	23
2.2	Diagramme de cas d'utilisation global	26
2.3	Diagramme de cas d'utilisation "Gérer mission"	27
2.4	Diagramme de cas d'utilisation "Gérer scenario"	29
2.5	Diagramme de cas d'utilisation "Gérer la navigation"	31
2.6	Diagramme de cas d'utilisation "Contrôler l'état du robot"	32
2.7	Diagramme de cas d'utilisation "Consulter les flux vidéo transmis"	33
2.8	Diagramme d'activité du cas d'utilisation "Créer mission par manette intégrée" .	34
2.9	Diagramme d'activité du cas d'utilisation "Consulter les flux vidéo transmis" .	35
2.10	Diagramme d'activité du cas d'utilisation "Créer scenario"	36
2.11	Diagramme d'activité du cas d'utilisation "Lancer scenario"	37
2.12	Schéma décrivant une session WebSocket	39
2.13	Schéma décrivant la communication via RTSP	39
2.14	Architecture physique du système	41
2.15	Diagramme de déploiement du système	42
3.1	Prototype des interfaces utilisateur de l'application mobile	44
3.2	Illustration du modèle MVC	45
3.3	Présentation de l'architecture de GETX	47
3.4	Diagramme de classe du système	48
3.5	Diagramme de séquences du cas d'utilisation "S'authentifier"	50
3.6	Diagramme de séquences du cas d'utilisation "Vérifier connexion"	51

3.7	Diagramme de séquences du cas d'utilisation "Activer l'arrêt d'urgence"	52
3.8	Diagramme de séquences du cas d'utilisation "Créer mission par manette intégrée"	54
3.9	Diagramme de séquences du cas d'utilisation "Créer mission par mappe"	55
3.10	Diagramme de séquences du cas d'utilisation "Lancer un message d'interpellation"	56
3.11	Diagramme de séquences du cas d'utilisation "Lancer un scenario"	57
3.12	Diagramme d'états-transitions du cas d'utilisation "Créer mission"	58
3.13	Diagramme d'états-transitions du cas d'utilisation "Lancer un scenario"	59
4.1	Interfaces de bord	66
4.2	Interfaces d'authentification	67
4.3	Interfaces d'accueil	68
4.4	Barre de tâches	68
4.5	menu de navigation	69
4.6	Panneau principal de l'interface d'accueil	69
4.7	Liste des messages d'interpellation	70
4.8	Interface des scénarios	70
4.9	Interface illustrant la phase du chargement d'un scénario	71
4.10	Étapes pour créer un nouveau scénario	72
4.11	Interface des caméras	73
4.12	Interface des mission	73
4.13	Les modes de création d'une mission	74
4.14	Création de missions par le mode "Map"	74
4.15	Création de missions par le mode "Joystick"	75
4.16	Interface de navigation libre du robot	75
4.17	Interface de paramètres	76
4.18	Exemple de notification reçue	76
4.19	Interface de perte de connexion	77

Liste des tableaux

1.1	Avantages et inconvénients de l'interface Qt	11
1.2	Avantages et inconvénients de l'interface du robot E-vigilante	12
1.3	Avantages et inconvénients de l'interface du robot UGV	14
2.1	Espaces des fonctionnalités offertes par l'application	25
2.2	Description textuelle du cas d'utilisation "Créer mission par manette intégrée" .	28
2.3	Description textuelle du cas d'utilisation "Ajouter scenario"	30
2.4	Description textuelle du cas d'utilisation "Activer commande par manette physique" .	31
2.5	Description textuelle du cas d'utilisation "Faire charger le robot"	32
2.6	Description textuelle du cas d'utilisation "Faire tourner la tourelle"	33

Introduction Générale

Le monde est toujours en état d'évolution prospère, une évolution rendue possible grâce à l'avancement technologique et l'innovation. L'une des facettes de ces progrès qui recèle le potentiel immense de transformer de nombreux aspects de notre vie est la robotique.

Lorsque nous pensons à la robotique, nous pourrions imaginer des automates inspirés de la science-fiction et ressemblant à des êtres-humains. Bien que ces types de machines soient encore pour la plupart encore fictifs, les robots d'aujourd'hui représentent une technologie très réelle et très utile pour les humains. Les robots construisent des choses pour nous, ils nous aident à explorer et à surveiller notre monde, et ils peuvent même nous soigner.

De nos jours, les robots atteignent une capacité et une robustesse exceptionnelle, et tant la robotique que l'intelligence artificielle entraînera des répercussions énormes sur de nombreux secteurs tels que l'industrie militaire, les soins de santé, le transport et la logistique.

La robotique s'est également étendue récemment dans le domaine de sécurité que ce soit pour les individus moraux ou bien les grands sites. Elle est par essence multidisciplinaire, traitant dès leur origine des problèmes couplant mécanique, automatique, électronique, informatique, etc.

Cette complexité induit de nombreuses exigences, notamment sur l'informatique censée gérer le fonctionnement du robot et supporter ses capacités d'action, d'adaptation, de décision, etc., c'est une « intelligence » que lui confère son contrôle. C'est-à-dire la manière dont est conçu et développé le logiciel chargé du contrôle du robot, c'est cette partie qui sera justement le cœur du présent projet.

Pour être en mesure de suivre ce marché en constante évolution, Enova Robotics, l'un des leaders de la conception et de la fabrication de robots, propose le développement d'une solution logicielle mobile pour la téléopération de son robot de sécurité, le P-Guard tout en gérant l'ensemble de données transmises par le robot en temps réel et en bénéficiant d'une administration optimale des ressources. C'est dans ce cadre que notre stage se déroule.

Le présent rapport résume le travail effectué, et qui a comme objectif de définir le contexte du projet et de décrire son sujet, son architecture, ses outils et ses méthodes utilisés, ainsi que les résultats obtenus.

Il est organisé en quatre chapitres comme suit :

- Le premier chapitre, intitulé «**Cadre général du projet**» est dédié à une présentation de la société hôte Enova Robotics, l'aperçu général du projet, la problématique du sujet, la description de la solution proposée et de la méthodologie de travail adoptée.
- Le deuxième chapitre, intitulé «**Spécification des besoins fonctionnels et techniques**

du projet» est consacré aux spécifications des besoins en définissant les acteurs de notre application, en spécifiant les besoins fonctionnels et technique à l'aide des diagrammes de cas d'utilisation et des diagrammes d'activités auxquels notre application doit répondre.

- Le troisième chapitre, intitulé «**Conception détaillée**» décrit la conception détaillée et l'architecture logicielle de l'application proposée par le biais d'un ensemble de diagrammes de séquence et d'états-transitions.
- Le quatrième chapitre, intitulé «**Réalisation**» présente tous les détails de la réalisation du projet, l'environnement et les outils de travail, ainsi que les défis soulevés lors du développement, avec des captures d'écran illustratives du travail réalisé.
- Une conclusion sera consacrée pour indiquer les différents atouts de l'application réalisée et des perspectives éventuelles de son amélioration seront données à la fin du rapport.

Chapitre **1**

Cadre général du projet

Plan

1.1	Présentation de l'organisme d'accueil	4
1.1.1	Présentation	4
1.1.2	Organigramme de la société	4
1.1.3	Produits réalisés par la société	5
1.2	Présentation du projet	8
1.2.1	Cadre du projet	8
1.2.2	Etude de l'existant	9
1.2.3	Problématique	14
1.2.4	Objectifs	15
1.2.5	Solution envisagée	16
1.3	Méthodologie de gestion de projet	18
1.3.1	Méthodologie UP	18
1.3.2	Méthodologie 2TUP	19
1.3.3	Choix de la méthodologie	20

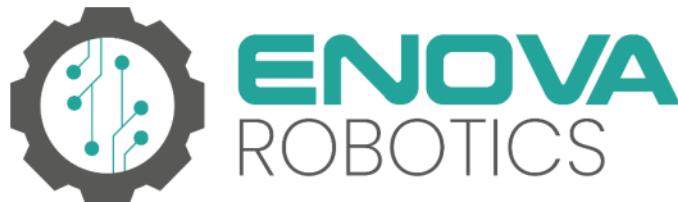
Introduction

Au début de ce rapport, nous présentons l'entreprise qui a proposé ce projet en décrivant le secteur, dans lequel elle exerce ses activités, et ses divers produits et services. Nous plaçons, par la suite, le projet dans son cadre général et nous détaillons la problématique ainsi que la solution envisagée. Nous présentons aussi l'analyse et l'étude faite sur les solutions déjà implémentées. Enfin, nous décrivons la méthodologie de travail adoptée.

1.1 Présentation de l'organisme d'accueil

1.1.1 Présentation

ENOVA ROBOTICS [1] est une société S.A totalement exportatrice, fondée en 2014, basée dans l'espace incubateur au Technopole de Novation, Sousse Tunisie. Elle fabrique et commercialise des robots intelligents dans les domaines de la sécurité, de la santé et de l'environnement. Elle représente la première société en Afrique et dans la région arabe qui fabrique sa propre marque de robots. Elle a été créée suite à une expérience de dizaine d'années dans l'enseignement et la recherche dans le domaine robotique. La réalisation de plusieurs prototypes, les essais et la volonté de développer des systèmes innovants a aidé ENOVA à acquérir l'expérience et les connaissances nécessaires pour satisfaire les besoins particuliers de ses clients en termes de design et de fonctionnalités.



1.1.2 Organigramme de la société

L'équipe d'Enova Robotics est répartie sur trois pôles :

- **Pôle commercial** : C'est un département formé d'un designer et un responsable de l'entreprise, son rôle consiste à prendre en charge l'organisation du travail, les relations avec les clients et la synchronisation de l'équipe.
- **Pôle informatique** : Il s'agit d'une équipe composée par des ingénieurs informatiques, des docteurs chercheurs en informatique, un directeur du système d'information et des doctorants qui se chargent de la programmation du robot et de ses interfaces d'utilisation.
- **Pôle mécatronique** : Ce département est formé par des ingénieurs en mécatronique et des techniciens en mécanique. Cette équipe définit des cahiers de charges et assure la conception et la fabrication des produits de l'entreprise.

La figure 1.1 représente l'organigramme de la société qui résume les différents services qu'elle comporte.

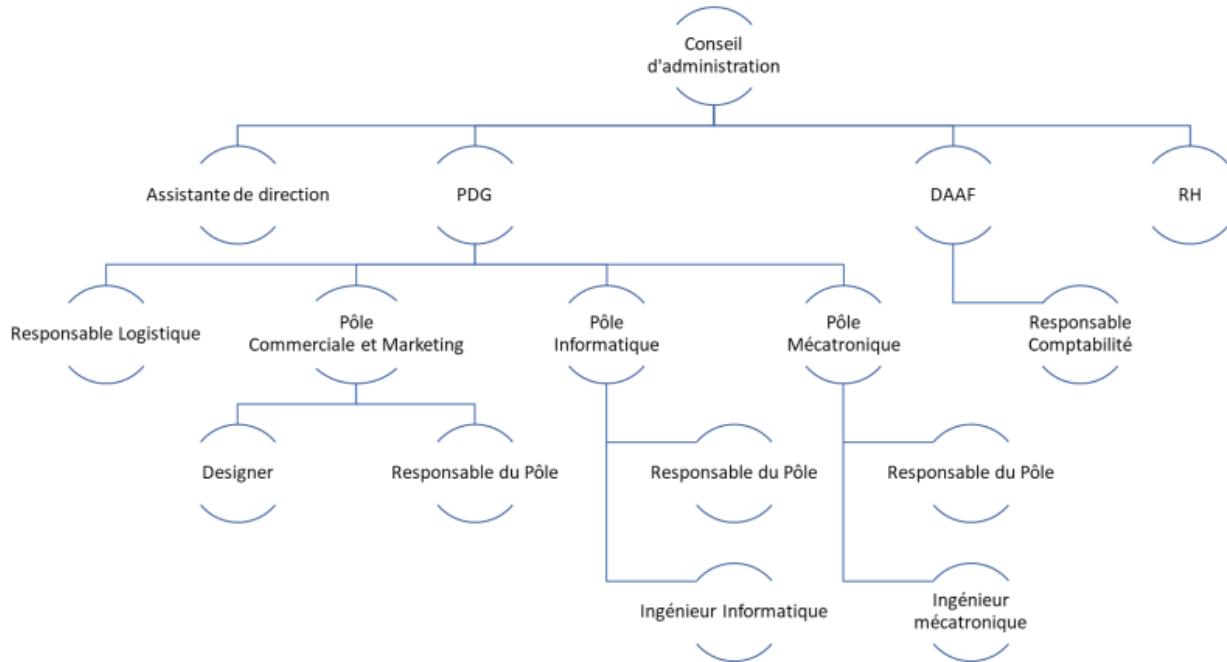


FIGURE 1.1 – Organigramme d'ENOVA ROBOTICS

1.1.3 Produits réalisés par la société

La société conçoit, fabrique et commercialise différentes gammes de robots mobiles selon leur domaine d'application. Elle développe son propre robot de marque soit pour le proposer dans le marché ou pour personnaliser des solutions pour répondre à un besoin spécifique du client. Parmi les produits de Enova, nous présentons :

Le robot Minilab : comme il se présente dans la figure 1.3, c'est un robot de taille moyenne destiné à L'éducation, il offre ainsi une expérience d'enseignement complète grâce à ses laboratoires préfabriqués et son aptitude à être simulé sur Matlab et Gazebo, il possède une architecture de contrôle open source qui est disponible sur GITHUB et elle est basée sur Robot Operating System (ROS). Il opte à offrir une meilleure expérience éducative plus compréhensive pour les étudiants[2].



FIGURE 1.2 – Le robot Minilab

Le robot Covea : Covea, présent sur la figure 1.4, est un robot de télé-présence conçu pour aider les personnes âgées à domicile. Il permet ainsi la télé-présence, le contrôle continu et la télévigilance. Il est donc un atout pour faciliter les suivis aux médecins et leur permettre d'interagir avec les personnes éloignées. Ce modèle a été déployé dans l'un des principaux hôpitaux tunisiens qui prend en charge les patients atteints de la COVID-19, afin de limiter le contact entre les soignants et les malades et d'améliorer les échanges entre les patients et les familles, ce qui a représenté une première dans le pays[2].



FIGURE 1.3 – Le robot Covea

Le robot Oggie : Oggie, présent sur la figure 1.5, est un robot de sécurité à domicile développé par Enova Robotics en partenariat avec d'autres entreprises tunisiennes : Chifco et OPCMA. C'est un robot compagnon qui protège et surveille la maison et qui a la possibilité de communiquer avec les objets connectés tout en agissant en temps réel.



FIGURE 1.4 – Le robot Oggie

Le robot Pearl Guard (PGuard) : Pearl Guard, le robot dont la partie informatique de commande constitue le sujet du présent Projet de Fin d'Etudes. Il s'agit d'un robot mobile autonome capable de naviguer sans arrêt pour plus de 8 heures. Conçu et optimisé principalement pour la surveillance des sites industriels, il permet de substituer l'acteur humain en première ligne de danger renforçant ainsi la sécurité du personnel et des installations matérielles. Ce robot de 100 kg prend la forme d'un mini véhicule comme le montre la figure 1.6. Son équipement de pointe, caméras thermiques, ordinateurs de bord, suite de communication longue portée, lui permettent d'effectuer des rondes, détecter des intrusions en toute autonomie et permet à l'utilisateur de faire des interventions à distance. Véhicule tout terrain, Pearl Guard peut poursuivre un intrus en transmettant en temps réel sa localisation précise ainsi qu'un flux vidéo et flux thermique. Efficace, robuste et autonome, Pearl Guard assure une sécurité optimale aux sites où il est déployé. Ce robot a fait beaucoup parler de lui à l'échelle nationale et internationale durant la crise de COVID-19 où il a été déployé pour patrouiller dans les quartiers de la capitale tunisienne afin de s'assurer que les gens obéissent au confinement exigé par les autorités, dans le cadre de la lutte contre le coronavirus[3][4].



FIGURE 1.5 – Le robot P-Guard

1.2 Présentation du projet

Ce projet a été réalisé au sein de la société Enova Robotics et ce, dans le cadre d'un projet de fin d'études en vue de l'obtention du diplôme national d'ingénieur en Informatique à l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse (ISSATSo) au compte de l'année académique 2021/2022.

L'objectif principal, de ce projet, consiste à implémenter une solution logicielle mobile pour la téléopération du Robot P-Guard.

Nous présentons, dans ce qui suit, le contexte général du projet. Nous exposons, en premier lieu la problématique et les raisons qui ont poussé Enova Robotics à proposer ce sujet. En second lieu, nous présentons la solution proposée afin de résoudre la problématique.

1.2.1 Cadre du projet

Dans le monde d'aujourd'hui, axé sur la technologie, les nouvelles tendances technologiques telles que la robotique et l'intelligence artificielle attirent beaucoup l'attention. Les robots envahissent de plus en plus le monde et contribuent à tous les aspects de la vie et de l'industrie. L'innovation dans la robotique touche à plusieurs domaines. Ainsi, sans nier les liens entre les différents pans de la robotique, le cœur du présent projet est la robotique mobile¹, spécifiquement, les robots de surveillance et de sécurité.

Le comportement des robots de surveillance repose sur deux types d'actions ; d'une part l'assistance des agents humains dans la surveillance pour assurer la sécurité environnementale et d'autre part, la surveillance au sens de gardiennage, c'est-à-dire la prévention des intrusions physiques ou les interventions qui requièrent un niveau de précision ou des qualités inaccessibles à l'opérateur humain.

Parmi les robots de sécurité existants les plus connus, nous citons le robot K5 [5], E-Vigilante [6], le robot UGV (unmanned Ground Vehicle) de la société américaine SMP Robotics [7].

Ces robots possèdent une autonomie de plusieurs heures pour effectuer leurs missions et leurs rondes de levée de doutes.

Techniquement parlant, un robot est composé d'un microcontrôleur ainsi que d'un ou plusieurs capteurs et actionneurs. Le développement d'un robot mobile et autonome nécessite une implantation matérielle ainsi qu'une implantation logicielle.

Un robot ne fonctionne qu'après l'exécution continue d'un programme informatique constitué d'un ensemble d'algorithmes destinés à chaque tâche souhaitée et la réception d'ordres par une application de commande.

Ça prouve que la partie informatique représente l'âme du robot et c'est cette partie qui constitue le centre d'intérêt de ce Projet de Fin d'Etudes.

Les robots, dotés d'un ensemble de cameras, suivent des missions prédéfinies pour assurer la télésurveillance en transmettant un flux vidéo et audio continu. La commande et le pilotage

1. Secteur industriel qui témoigne un développement extrêmement rapide, combine l'ingénierie, l'informatique, mais aussi les sciences cognitives et l'intelligence artificielle pour faire face à la complexité des robots mobiles.

pratique et intuitif des robots se fait à l'aide des solutions logicielles et des interfaces de contrôles utilisées par les moniteurs ou les agents de sécurité.

1.2.2 Etude de l'existant

Cette section est consacrée à une étude critique des applications existantes communiquant avec des robots de sécurité , afin d'introduire notre solution proposée.

1.2.2.1 Analyse de l'existant

Certes que la concurrence fait rage entre les différentes entreprises qui exercent dans le domaine de la robotique.

Cependant, parmi les membres de la communauté de robotique, seuls des petits programmes destinés à l'apprentissage ou à l'échange sont accessibles en ligne. Il se trouve également des IHM généralistes qui permettent à un utilisateur humain d'interagir avec un dispositif informatisé (un outil, un ordinateur).

Toutefois, pour la commande et le contrôle des robots de sécurité, il n'existe pas de solutions génériques adoptées aux différentes caractéristiques des robots existants, mais plutôt chaque solution est dédiée à un robot spécifique avec ses caractéristiques uniques.

Les IHM dédiées aux robots de sécurité spécifiquement ne sont pas, généralement, partagées au grand public, pour cette raison l'étude est faite sur la solution interne qui était utilisée pour l'interaction avec le robot P-Guard de Enova Robotics.

- **IHM du robot P-guard (Enova Roboics)**

Ce robot de 100 kg prend la forme d'un mini véhicule, comme indiqué dans la section des produits réalisés par la société, il comporte 4 cameras, 4 roues motrices actionnées par 4 moteurs électriques, qui lui permettent d'évoluer sur tous les revêtements et d'atteindre 12km/h en vitesse max.

Enova Robotics a mis en œuvre l'interface utilisateur Qt en 2015 par l'utilisation d'un Framework QT pour commander le robot.

La figure 1.6 représente l'interface utilisateur de la solution robotique P-Guard.

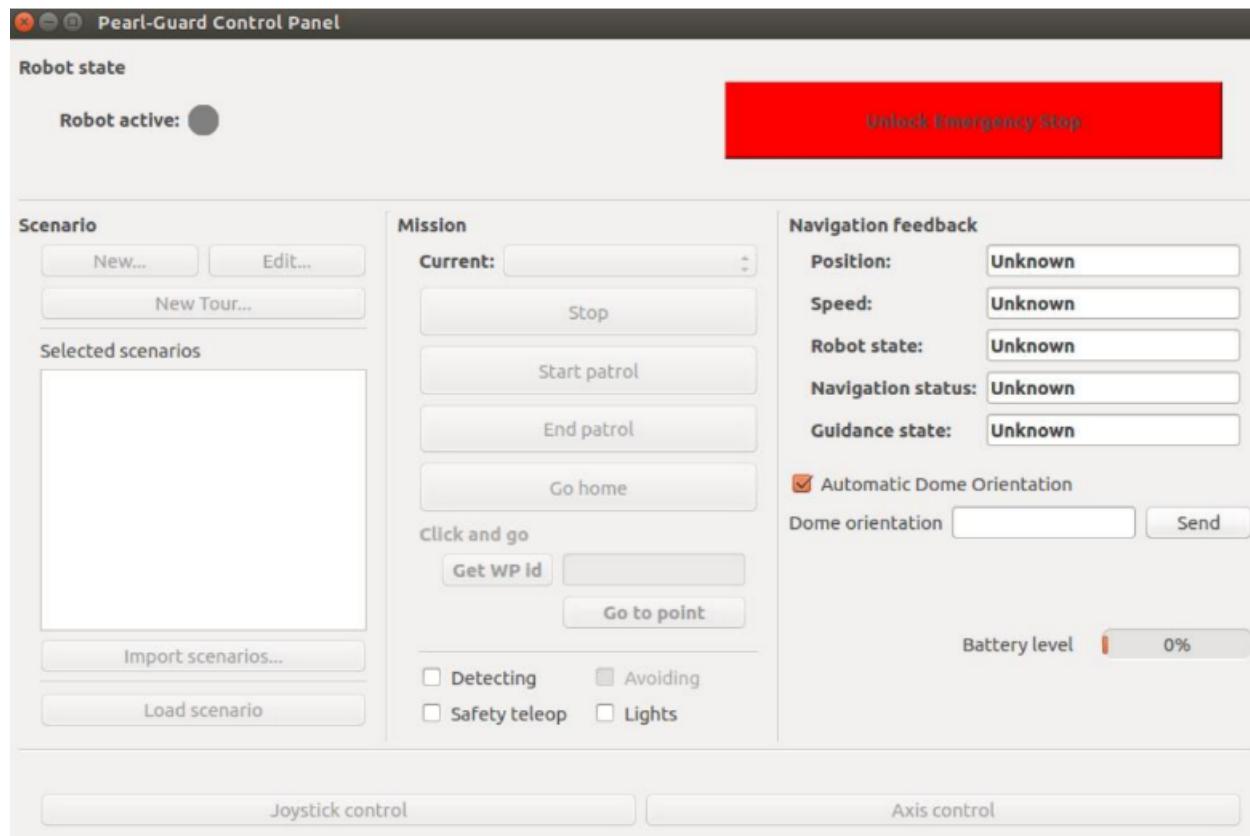


FIGURE 1.6 – IHM Qt

Cette application desktop offre un ensemble de fonctionnalités permettant d’interagir avec le robot et de gérer la majorité des actions du robot depuis le PC de contrôle.

L’agent peut gérer l’ensemble des missions et des scenarios ainsi que les périphériques connectés.

Elle présente un ensemble de panneaux de contrôle où chacun est utilisé pour réaliser des tâches spécifiques (ajouter ou modifier des missions, allumer les lampes, lancer une tournée). Elle détaille aussi les feedbacks sur l’état actuel reçus par le robot.

1.2.2.2 Critique de l’existant

D’après la solution existante étudiée, les fonctionnalités offertes semblent être réduites et ne répondent ni aux attentes d’un utilisateur final, ni aux besoins critiques demandés pour le bon fonctionnement d’un robot occupant un poste critique de sécurité.

L’application offre un ensemble de fonctionnalités et services que nous critiquons dans le Tableau 1.1 :

✓ Avantages	✗ Inconvénients
<ul style="list-style-type: none"> - Présentation de la majorité de fonctionnalités essentielles à la gestion des rondes au niveau de L'interface principale. - Valorisation des retours du robot et visualisation de son état courant. - Accès simple aux différents modes de commandes du robot. 	<ul style="list-style-type: none"> - Non-intégration du flux vidéo transmis par le robot. - Absence de géolocalisation pour le tracking en temps réel de la position du robot. - Fonctionnement sur un environnement bien déterminé (L'application n'est pas multiplateforme). - Commande et contrôle fixe du robot (Limitation de la mobilité de l'agent). - Limitation de l'exploitation du potentiel de la machine. - Fonctionnement limité aux réseaux locaux (Wifi/Ethernet). - Difficulté de maintenance : L'interface n'est pas flexible et n'est pas évolutive pour répondre aux éventuelles extensions exigées par le domaine d'utilisation.

TABLE 1.1 – Avantages et inconvénients de l'interface Qt

1.2.2.3 Solutions comparables

Il existe d'autres interfaces utilisateur pour la gestion du fonctionnement des robots de sécurité qui sont dédiées à des robots spécifiques, non compatibles avec le robot P-Guard mais offrent des fonctionnalités similaires.

- **IHM du robot E-vigilante (UBECOME)**

e-vigilante est un robot de service professionnel développé par la société française UBECOME consacrée à la supervision interne d'entrepôts et de sites industriels. Il effectue des rondes automatisées et prévient immédiatement la personne chargée de la surveillance du site lorsqu'un incident est détecté.

La figure 1.7 [6] représente l'interface utilisateur de la solution robotique e-vigilante :

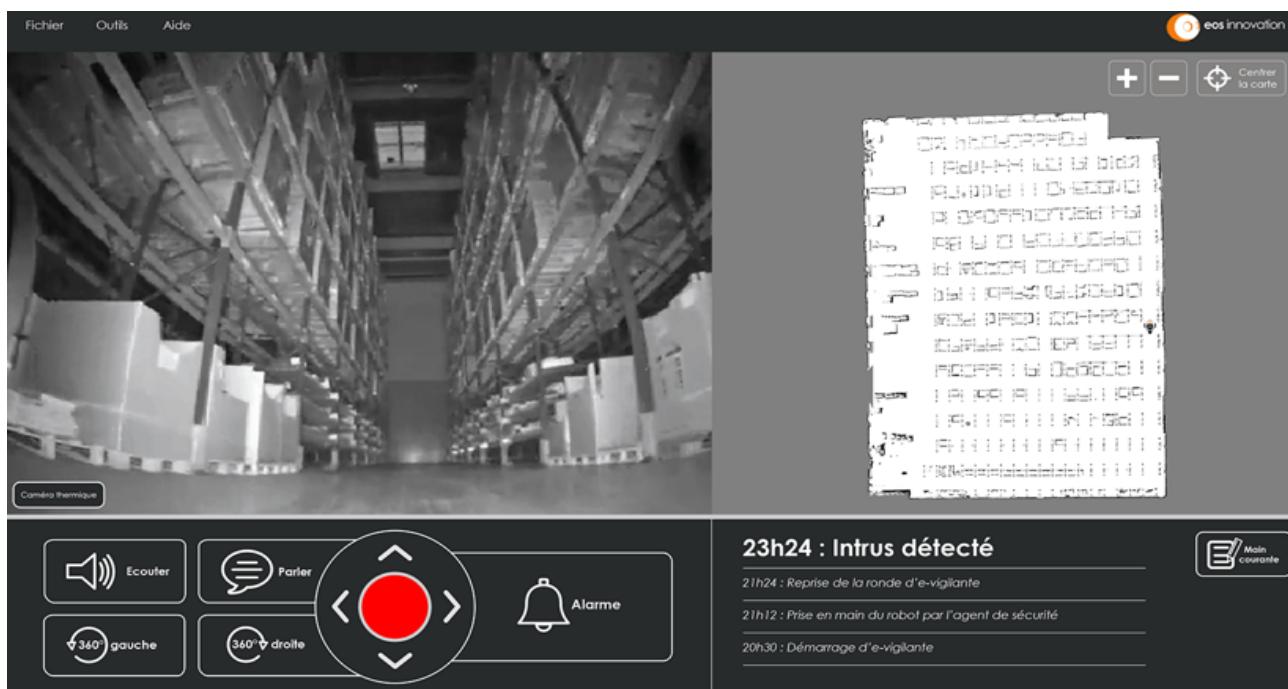


FIGURE 1.7 – IHM du robot E-vigilante

L’interface d’e-vigilante est conçue pour présenter les fonctions majeures, et les données transmises par le robot. Elle permet la gestion des rondes du robot, la visualisation du flux vidéo transmis ainsi qu’un paramétrage adapté à l’évolution du site.

L’agent de sécurité a un accès direct à l’historique des intrus détectés, et il peut commander le robot par l’interface selon les conditions environnantes.

Nous présentons à travers le Tableau 1.2 les avantages et les inconvénients de l’interface utilisateur de la solution robotique E-vigilante.

✓ Avantages	✗ Inconvénients
<ul style="list-style-type: none"> - Présentation de la majorité de fonctionnalités essentielles à la commande du robot au niveau de l’interface principale. - Simplicité d’emploi de l’application. - Visualisation effective du flux vidéo en temps réel. - Détection d’intrusion et affichage de l’historique des activités du robot. - Accès à la partie administrateur permettant de faciliter le paramétrage des rondes d’e-vigilante, et la gestion des utilisateurs 	<ul style="list-style-type: none"> - Absence de géolocalisation pour le tracking en temps réel de la position du robot. - Commande et contrôle fixe du robot (Limitation de la mobilité de l’agent). - Fonctionnement sur un environnement bien déterminé (L’application n’est pas multiplateforme). - Visualisation du flux d’une seule caméra à un instant donné

TABLE 1.2 – Avantages et inconvénients de l’interface du robot E-vigilante

- **IHM du robot UGV (SMP Robotics)**

SMP Robotics fabrique des robots de surveillance destinés à compléter l'action d'un agent de sécurité humain sur des sites industriels isolés.

La figure 1.8 [8] représente l'interface utilisateur de la solution robotique UGV :

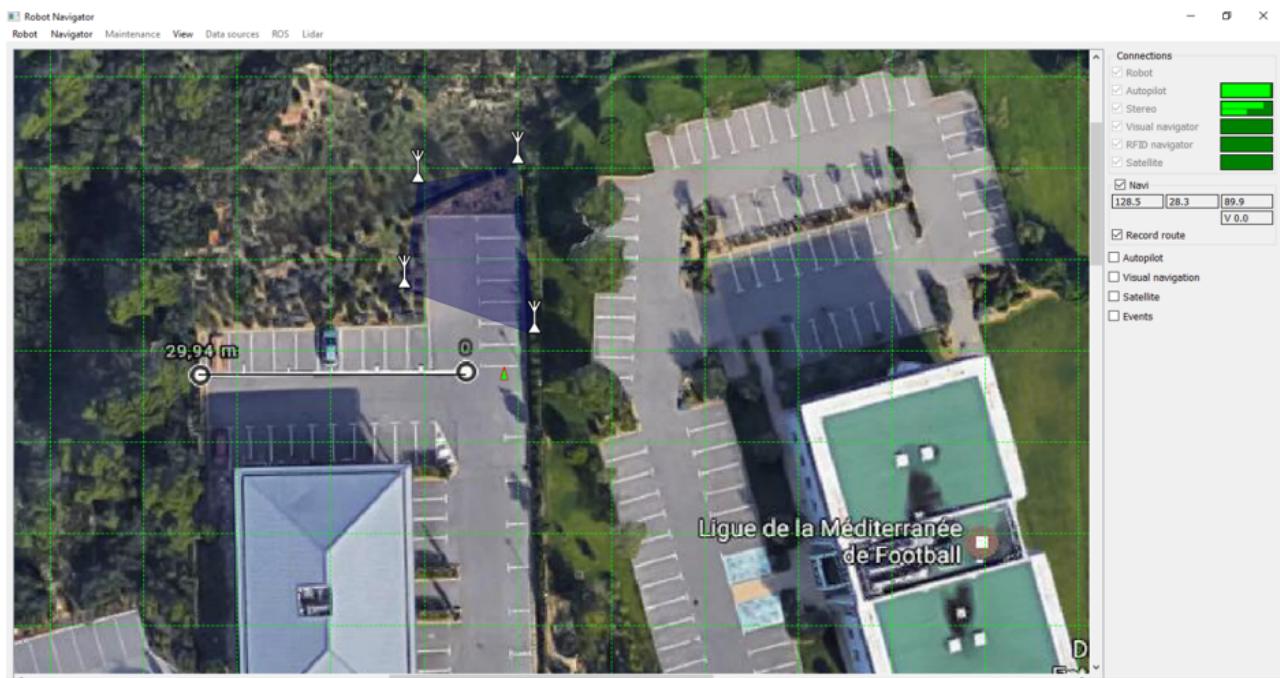


FIGURE 1.8 – IHM du robot UGV

SMP Robotics a développé sa propre IHM pour la gestion du robot UGV (unmanned Ground Vehicle). L'interface permet de se connecter à un robot, le commander et visualiser son positionnement via une carte locale qui n'a aucune corrélation avec des données de localisation globales (Ie GPS).

Elle permet également de créer un ensemble de points pour le patrolling et le lancement des tours.

Le Tableau 1.3 énumère les avantages et les inconvénients de l'IHM.

✓ Avantages	✗ Inconvénients
<ul style="list-style-type: none"> - Gestion de la majeure partie des interactions avec le robot. - Connexion au robot par adresse IP. Il suffit de configurer le logiciel pour être connecté au bon réseau. - Visualisation claire du fonctionnement des différents modules de localisation à l'aide des barres vertes. - Suivie des points de passage du robot. 	<ul style="list-style-type: none"> - Nécessité de consulter un autre logiciel pour accéder au flux vidéo. - Téléchargement et intégration manuelle de la carte de façon à pouvoir visualiser le parcours du robot. - Commande et contrôle fixe du robot (Limitation de la mobilité de l'agent). - Difficulté de maintenance : L'interface n'est pas flexible et n'est pas évolutive pour répondre aux éventuelles extensions exigées par le domaine d'utilisation.

TABLE 1.3 – Avantages et inconvénients de l'interface du robot UGV

1.2.3 Problématique

Comme indiqué précédemment, Enova Robotics propose parmi sa gamme de robots le P-Guard, un robot de sécurité extérieur conçu pour faire des patrouilles en milieu extérieur en assurant la fonction de télésurveillance tout en transmettant un flux audio et vidéo continu.

D'après l'étude de l'existant, parmi les problèmes-clés de la robotique mobile, nous traitons, la difficulté de contrôle. Le robot suit des ordres et des opérations prédéfinies. C'est à l'agent de sécurité de superviser le robot et d'envoyer en continu des commandes convenables pour l'obtention du comportement désiré. Comme le robot présente, parfois, des difficultés d'interaction avec son environnement local, l'agent a besoin d'un ensemble d'actions simples et pratiques pour commander le robot.

Le robot ne peut pas encore évaluer efficacement si une levée de doute est positive ou négative, encore moins prendre la décision d'intervenir dans le bon moment. Tous ces limites sont actuellement des sujets de recherche afin de les améliorer dans les prochaines années. C'est pour cette raison qu'un agent de sécurité devra se déplacer pour toute vérification nécessaire lors d'événements douteux ce qui demande un effort humain intensif et résulte un gaspillage d'énergie et de temps.

Les robots offrent, évidemment, une multitude d'avantages pour l'industrie, cependant, ils présentent encore des risques de mauvais fonctionnement. Ils peuvent confronter des problèmes avec leur panneau de commande, de défaillances mécaniques, de pannes de courant ou de facteurs environnementaux. A cause de cela, un agent doit être présent en face d'une IHM pour prendre des décisions convenables. Le fonctionnement du robot est donc accompli dans un cadre

de collaboration homme-machine.

En outre, le fonctionnement contenu du robot pour faire les levées de doutes ou pour les rondes de surveillance autonomes, résulte une diminution du niveau de sa batterie, ce qui nécessite un rechargeement régulier et une supervision en temps réel de son état.

Toute fonction de commande, de contrôle ou de suivi sera donc effectuée en téléopération à travers une interface distante. Dans le but d'optimiser les tâches de l'agent de sécurité, et de servir la commercialisation de la solution sur le marché de la supervision à distance et sécurité, Il faut considérer un ensemble de facteurs indispensables. Il faut favoriser principalement la mobilité et la liberté de mouvement des moniteurs. Le robot est généralement en état de mobilité, considérer le contrôle via une interface fixe limitera évidemment leurs tâches. En outre, on cherche souvent à optimiser l'expérience de l'utilisateur en enrichissant son engagement, et améliorer la praticabilité et l'accessibilité aux services et processus dont ils ont besoin en temps réel.

D'autre part, une autre contrainte qui s'impose, qui est la synchronisation lors d'un contrôle multiple du robot par plusieurs applications. L'utilisateur de l'application doit être notifié de tout changement au niveau de l'état du robot. Le système doit aussi réagir pour visualiser les actions effectuées en temps réel.

De ces fait, L'agent a besoin d'une solution graphique pratique pour assurer la tâche essentielle de surveillance et de sécurisation de sites tout en recueillant toutes les données envoyées par le robot.

1.2.4 Objectifs

L'objectif principal du projet consiste à créer une solution logicielle mobile pour interagir avec le robot dans le but de faciliter et de compléter les tâches d'un agent de sécurité et renforcer la protection des grands sites.

Efficacité opérationnelle : L'application mobile doit offrir un ensemble de fonctionnalités permettant de contrôler et commander le robot et répondre aux problèmes cités précédemment :

- Faciliter les tâches d'un agent de sécurité.
- Faire gagner du temps.
- Eviter le gaspillage d'énergie .
- Diminuer le risque d'erreur et du mauvais fonctionnement du robot.
- Fournir une administration optimale des ressources.
- Suivre en temps réel le changement d'état du robot.
- Suivre et géolocaliser le robot.

- Avoir une visualisation directe du flux vidéo.
- Avoir la possibilité d'intervenir lors de la détection d'un intrus.

Approche centrée sur l'utilisateur : L'application doit assurer une flexibilité et une facilité d'utilisation à travers des interfaces ergonomiques préservant la meilleure expérience utilisateur.

1.2.5 Solution envisagée

Vu les différentes contraintes liées à la problématique, que nous avons détaillé dans la section précédente, et dans le cadre d'une démarche d'amélioration continue des produits proposés, Enova Robotics combine un esprit de réflexion orienté solution avec un flux de travail intensif pour développer une solution logicielle mobile pour la téléopération du robot P-Guard.

La solution proposée sera déployée sur une tablette et vendue aux clients, accompagnant le robot entier comme l'outil de commande principal.

La surveillance par des robots ne tend pas à remplacer l'agent de sécurité, mais plutôt à l'aider à renforcer la sécurité des grands sites et compléter ses actions dans un esprit de collaboration entre l'humain et la technologie.

L'application mobile consiste en une solution plus pratique pour la commande du robot à travers des dispositifs connectés à l'interface et le contrôle de ses différentes fonctionnalités ainsi que le suivi en temps réel de son état.

L'interface est destinée aux agents de sécurité jouant le rôle de superviseur des robots. Ces derniers effectuent des patrouilles dans un environnement extérieur et transmettent à l'application un flux vidéo, des alertes dans certains cas, et des rapports sur l'état courant. Cette solution présente un intérêt non négligeable, car elle permet de limiter les coûts d'envoi et de mouvement d'agents de sécurité pour faire des levées de doute, notamment pour des cas de fausses alertes.

L'application est une IHM qui permet aux utilisateurs d'être au courant, en temps réel, avec tout changement dans l'état du robot : niveau du chargement de la batterie, qualité de la connexion, retour sur la navigation, l'état de la correction RTK², erreur au niveau du fonctionnement, ainsi que la distance séparant le robot de la station de recharge.

Non seulement ce que précède, mais de plus, l'agent a accès au flux des caméras et du micro embarqués sur le robot P-Guard. Lors de la détection d'un intrus à travers les cameras surveillées, l'agent a la possibilité de télé-interpeller l'individu en lançant un message d'interpellation ou tout autre son pré-enregistré.

D'autre part, le P-Guard possède une capacité de téléopération sécurisée. Concernant sa navigation autonome, il suffit de spécifier un ensemble de scénarios de surveillance optimaux et personnalisés selon le site où le robot effectue ses patrouilles. Un scénario est un ensemble

2. Le RTK est le dispositif permettant de transmettre en temps réel les données de corrections d'une base d'observation aux GPS mobiles.

de missions spécifiques. Pour chaque mission ou chemin préférable, il est possible de modifier la vitesse.

Après avoir chargé un scénario spécifique, le robot effectue ses rondes d'une manière autonome.

L'agent de sécurité dispose de deux modes de gestion des missions. L'ajout de l'ensemble de points formant la mission, peut se faire soit par la manette connectée logiquement au robot ou manuellement en plaçant les points désirés sur la mappe. La ronde est ensuite enregistrée et pourra être réalisée autant de fois que souhaité de manière fiable et répétable. Tout mouvement réalisé par le robot sera visualisé sur le Map intégré dans l'interface donc elle offrira aussi un suivi en temps réel de la localisation du robot.

L'application offrira aussi la possibilité de gérer la diminution du niveau de la batterie. En effet Lors de la détection d'un niveau faible, le système fera les calculs nécessaires pour indiquer ensuite à l'utilisateur le niveau de la batterie ainsi que la distance séparant le robot de sa base de recharge. L'agent aura la possibilité d'interrompre l'exécution de la ronde est de renvoyer le robot vers la station.



FIGURE 1.9 – Station de recharge du robot P-Guard

Finalement, Pour satisfaire la contrainte de synchronisation lors d'une manipulation multiple du robot, l'application sera responsable de notifier l'utilisateur des changements au niveau de l'état du robot (par exemple lors de l'enregistrement d'une nouvelle mission) pour être au courant des tâches exécutées par le robot. De plus l'agent disposera d'une liste sauvegardant l'historique d'états du robot. Le système, affichera aussi les actions effectuées en temps réel (par exemple l'action de tourner la tourelle va être visualisée en direct sur toute instance connectée de l'application).

Pour l'agent, l'application mobile est l'outil idéal pour commander le robot, elle assure sa liberté de mouvement et lui permet d'effectuer ses tâches de la manière la plus pratique et la plus efficace en gérant le robot et le commandant de proche ou de loin.

On peut penser à concevoir une application web ou plutôt desktop pour toute gestion des

fonctionnalités du robot mais il est de plus en plus demandé de penser avant tout MOBILE ! de synthétiser les informations à afficher et de sélectionner les fonctionnalités les plus adaptées pour le mobile tout en donnant un intérêt majeur à l'ergonomie (Une application de téléopération mobile, apporte une expérience nouvelle et améliore l'accessibilité des utilisateurs.) Cette solution assurera une optimisation notable des tâches de l'agent et servira la commercialisation du robot sur le marché.

La figure 1.10 résume le fonctionnement du système global.



FIGURE 1.10 – Schéma illustrant le fonctionnement global du système

1.3 Méthodologie de gestion de projet

Le choix de la méthodologie de travail en informatique est une étape essentielle pour assurer une bonne performance de développement en termes de qualité et de productivité. Plusieurs méthodologies existent chacune avec des faveurs et des difficultés différents selon la taille et la complexité du système.

Cette diversité ne permet pas de définir un seul processus universel. Les créateurs **d'UML** ont travaillé à unifier les meilleures pratiques de développement orienté objet pour donner le **processus unifié**.

1.3.1 Méthodologie UP

Le Processus unifié ou UP est une méthode de développement générique qui peut être facilement adaptée aux contextes de projets et de systèmes différents. Ce processus permet d'éclaircir qui fait quoi, et permet à chacun de savoir quelle est sa place dans le processus de production du logiciel. Il intègre toutes les activités de conception et de réalisation au sein de cycles de développement.

UP est un processus qui se caractérise par une démarche itérative et incrémentale, pilotée par les cas d'utilisation, et centrée sur l'architecture et les modèles UML[9].

- **Une démarche itérative et incrémentale :** Le logiciel évolue suivant des incréments, et chaque phase de la démarche comprend des itérations. Une itération est donc un cycle de développement des incréments depuis le recueil des besoins jusqu'à l'implantation et aux tests.
- **Une démarche guidée par des cas d'utilisation :** L'objectif principal d'un système logiciel est de rendre service à ses utilisateurs. Les besoins des utilisateurs permettent de définir des cas d'utilisation. A partir de ces cas d'utilisation, une série de modèles UML peuvent être créés, il faut par conséquent bien comprendre les désirs et les besoins des futurs utilisateurs.
- **Une démarche centrée sur l'architecture :** L'architecture d'un système logiciel peut être décrite comme les différentes vues du système qui doit être construit. Cette architecture doit prévoir la réalisation de tous les cas d'utilisation.

1.3.2 Méthodologie 2TUP

Comme nous avons désigné, les processus unifiés constituent une trame commune pour intégrer les meilleures pratiques de développement. Un processus UP est itératif et incrémental, conduit par les exigences des utilisateurs, piloté par les risques et orienté composants.

Le processus 2TUP se situe dans cette lignée, en insistant sur la séparation initiale des aspects fonctionnels et techniques. « 2 Tracks » signifie littéralement que le processus suit 2 chemins. Il s'agit des chemins « fonctionnel » et « technique ».

Les deux branches d'étude fusionnent ensuite pour la conception du système, ce qui donne la forme d'un processus de développement en Y[10].

La branche fonctionnelle comporte :

- La capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs.
- L'analyse, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.

La branche technique comporte :

- La capture des besoins techniques, qui recense toutes les contraintes et les choix non-fonctionnels. Les outils sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionné généralement par des prérequis d'architecture technique.
- Les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels.

La branche de conception et d'implémentation comporte :

- La conception détaillée, qui étudie ensuite comment réaliser chaque composant.

- l'étape de codage, qui produit ces composants.
- l'étape de recette, qui consiste enfin à valider les fonctions du système développé.

La figure 1.11 illustre le processus de la méthodologie 2TUP.

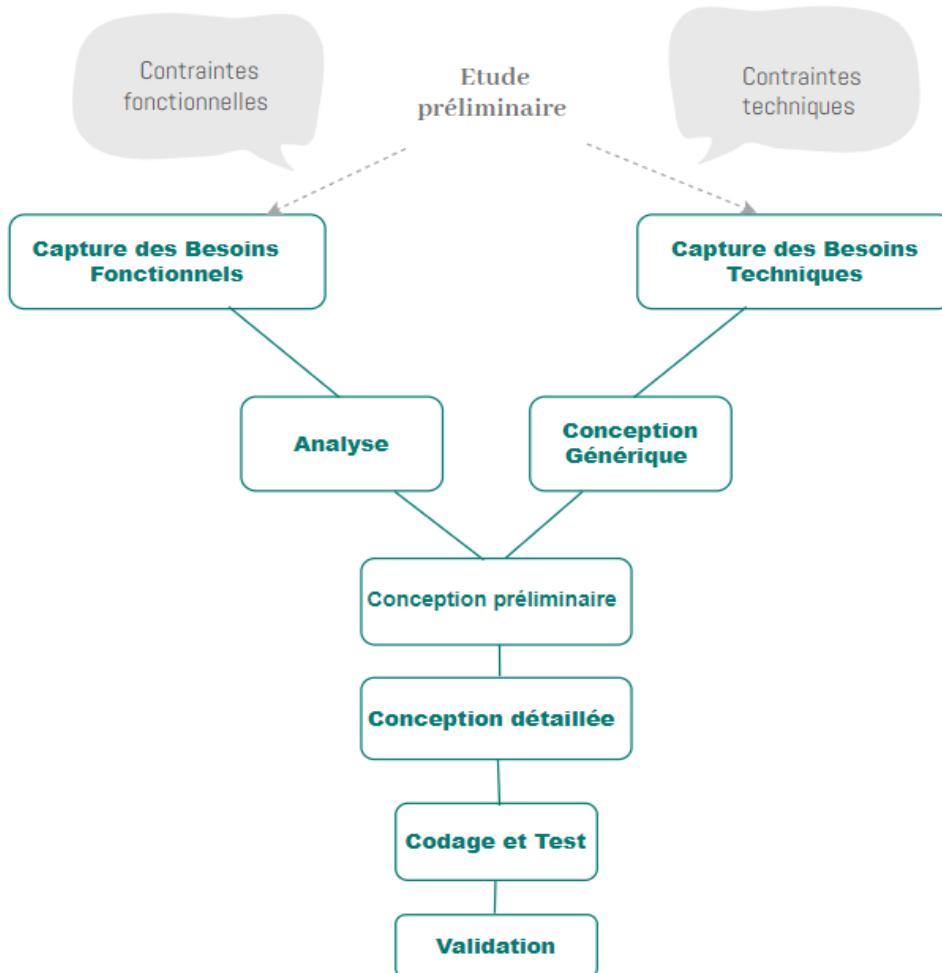


FIGURE 1.11 – Méthodologie 2TUP

1.3.3 Choix de la méthodologie

Le projet à réaliser dans le cadre de ce stage nécessite l'adéquation entre les besoins en termes de fonctionnalités chez son utilisateur et les exigences techniques. Pour cette raison, le choix d'une méthode de gestion de projet telle que le 2TUP est justifié par plusieurs raisons :

- Le projet fait abstraction dans son démarrage sur la logique métier pour se concentrer sur l'étude des scénarios et des fonctionnalités que peut apporter l'application lors de son interaction avec le robot et ses périphériques connectés.
- Le projet représente une certaine complexité au niveau technique d'où la nécessité de prévoir une phase pour étudier toutes les exigences techniques qu'il faut prendre en considération.
- Le système intégré dans le robot conditionne les choix techniques faits pour l'implémentation de l'application mobile, ce qui exige la prise en compte de contraintes d'intégration avec l'existant.

- A un moment donné, une liaison entre le modèle fonctionnel et le modèle technique doit être faite pour vérifier l'adaptation de l'architecture technique avec les besoins de l'utilisateur.

Se basant sur ce qui précède, la méthode 2TUP est la plus adéquate pour le contexte de ce projet.

Conclusion

Dans ce chapitre, nous avons présenté l'entreprise au sein de laquelle notre stage a été effectué. Ensuite, nous avons dévoilé le contexte du sujet de notre projet, les problèmes abordés, ainsi que la solution proposée. Finalement, nous avons précisé la méthodologie à adopter pour réaliser ce travail.

Prenant en considération les objectifs indiqués, le chapitre suivant portera sur l'identification des acteurs, et la spécification des besoins fonctionnels et non fonctionnels auxquels la solution proposée doit répondre.

Chapitre 2

Etude préliminaire : Spécification des besoins fonctionnels et techniques du projet

Plan

2.1	Modélisation du contexte	23
2.1.1	Identification des acteurs	23
2.1.2	Identification des messages échangés	24
2.2	Branche fonctionnelle	24
2.2.1	Besoins fonctionnels	24
2.2.2	Diagramme de cas d'utilisation global	26
2.2.3	Raffinement des cas d'utilisation	27
2.2.4	Analyse	34
2.3	Branche technique	37
2.3.1	Besoins non fonctionnels	37
2.3.2	Contraintes techniques	38
2.3.3	Conception générique	40

Introduction

Le deuxième chapitre est consacré à présenter le projet d'une manière formelle. Nous entamons dans ce qui suit, la présentation et la spécification des besoins fonctionnels et techniques. En respectant les principes de la méthodologie 2TUP, nous commençons par la première étape du cycle du vie du projet qui est la modélisation du contexte en identifiant les acteurs qui vont interagir avec le système et les messages échangés prévus. Par la suite, nous présenterons les deux chemins que le processus suit dans le but de détailler les contraintes fonctionnelles et techniques du projet.

La figure 2.1 illustre le résultat donné par l'étude préliminaire.

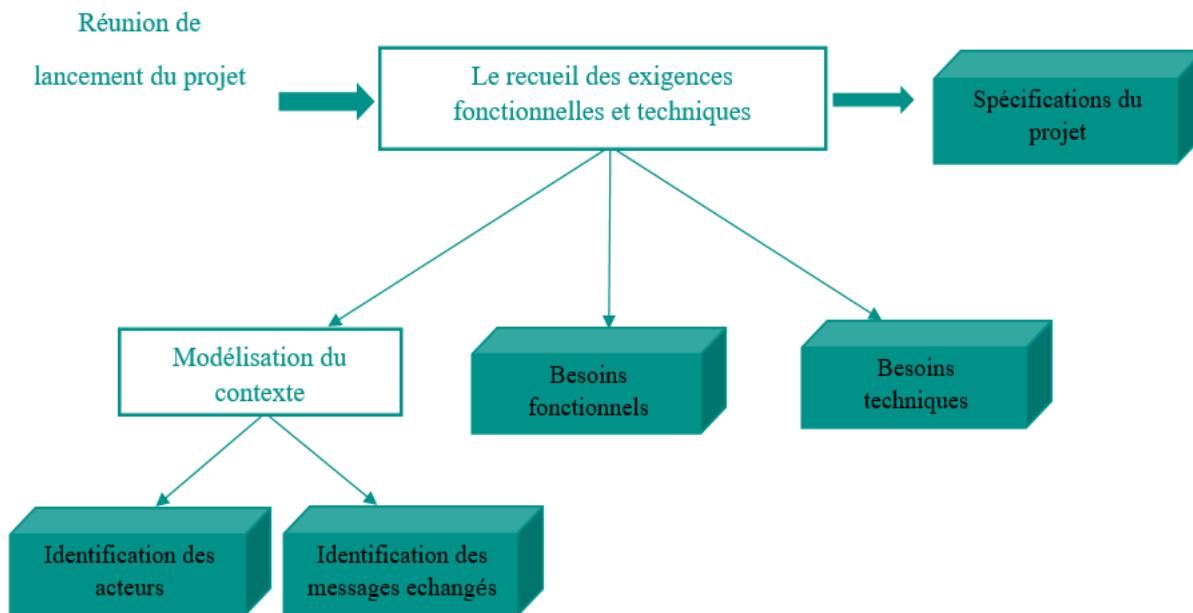


FIGURE 2.1 – Illustration de la phase de l'étude préliminaire

2.1 Modélisation du contexte

La modélisation du contexte est la toute première étape du processus 2TUP. Il s'agit principalement de définir les diagrammes de contextes : le diagramme de contexte statique qui représente les acteurs du système et le diagramme de contexte dynamique qui représente les messages échangés entre l'application et le reste d'éléments du système d'information.

2.1.1 Identification des acteurs

Dans cette section, nous présentons les principaux acteurs intervenant dans le fonctionnement du système. Un acteur est toute entité jouant un rôle actif, qui interagit avec l'application. Le système qui représente l'application mobile interagissant directement avec le robot est dédié à un acteur principal.

- **Agent de sécurité** : représente le superviseur du robot effectuant les patrouilles extérieurs et l'utilisateur primordial de l'application mobile. Il gère les scénarios et les missions et tout échange d'information avec le robot ou les périphériques connectés.

2.1.2 Identification des messages échangés

Dans cette section, nous énumérons les messages généraux envoyés et reçus par le système. Un message représente une communication unidirectionnelle transportant des informations dans le but de déclencher une activité dans le récepteur.

L'application émet les messages suivants :

- Liste des points formant une mission.
- Création et modification d'un scénario.
- Actions pour changer l'état du robot (allumer la lampe, faire tourner la tourelle).
- Requêtes pour guider et faire déplacer le robot.

L'application reçoit les messages suivants :

- Information sur l'état initial du robot.
- Position courante du robot.
- Flux vidéo et audio en temps réel.
- Changement instantané au niveau de l'état du robot (niveau de batterie).
- Messages de succès ou d'échec des requêtes envoyées par l'application.

2.2 Branche fonctionnelle

La capture des besoins fonctionnels consiste à identifier et détailler les différents cas d'utilisation obtenus à partir du diagramme de contexte dynamique (les messages échangés).

2.2.1 Besoins fonctionnels

Les besoins fonctionnels représentent les actions et les fonctionnalités qu'un système doit être capable à effectuer. Ce qui suit résume les besoins fonctionnels du système qui doivent satisfaire les objectifs détaillés précédemment. Pour mieux les éclaircir nous avons choisi de segmenter l'application en trois espaces :

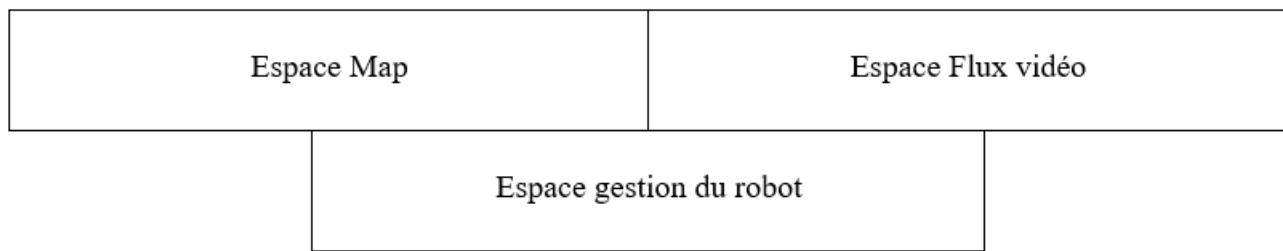


TABLE 2.1 – Espaces des fonctionnalités offertes par l’application

Espace Map

L’application permet à l’agent de sécurité de :

- Suivre la géolocalisation du robot.
- Spécifier des points GPS directement sur la carte.

Espace Flux vidéo

L’agent est capable de :

- Visualiser les flux vidéo.
- Avoir une vue plus détaillée d’une caméra spécifique.
- Contrôler le zoom de la caméra.
- Contrôler la rotation de la tourelle entre [0 ° ..360 °].

Espace gestion du robot

La gestion du robot comporte le traitement de l’ensemble d’information reçu par le robot en temps réel ainsi que l’ensemble d’action effectué pour changer son état.

- Traiter l’état de la connexion du robot.
- Visualiser les données (le niveau de la batterie, l’état de la correction RTK, la liste des missions et des scenarios).
- Afficher les messages d’alerte.
- Activer l’arrêt d’urgence.
- Agir sur les lampes et les périphériques du robot.
- Guider et diriger le robot d’un point à un autre.

L’agent de sécurité est aussi capable de gérer les missions du robot. Pour le faire, il dispose de **deux modes différents :**

- Ajouter des missions à travers la manette logique intégrée
- Ajouter des missions en spécifiant des points GPS directement sur la map.

De plus, il est possible de

- Gérer les scénarios : à partir de la liste des missions enregistrées, l'agent peut créer et modifier des scénarios.
- Lancer des scénarios : Pour activer le fonctionnement autonome du robot l'agent lance un scénario spécifique.

2.2.2 Diagramme de cas d'utilisation global

Les diagrammes de cas d'utilisation décrivent généralement le comportement du système en spécifiant l'ensemble d'actions que le système peut effectuer en collaboration avec un ou plusieurs utilisateurs externes du système (acteurs). Chaque cas d'utilisation est un scénario qui doit fournir un résultat observable et précieux aux acteurs sans se soucier de la manière dont le système exécutera réellement la fonction décrite.

La figure 2.2 représente le diagramme de cas d'utilisation général qui comporte les interactions entre notre système et ses acteurs.

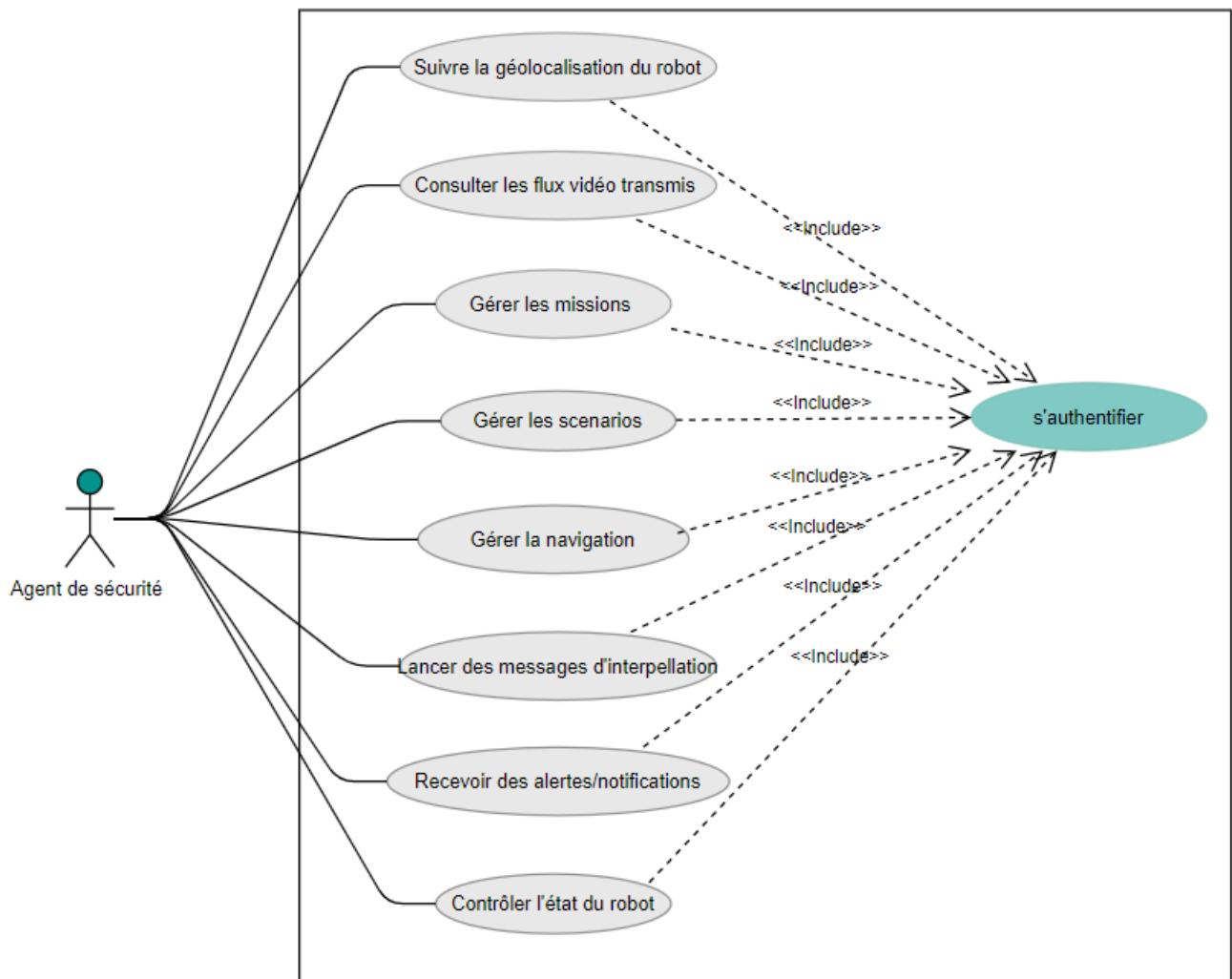


FIGURE 2.2 – Diagramme de cas d'utilisation global

2.2.3 Raffinement des cas d'utilisation

Dans cette section, nous présentons le détail des principaux cas d'utilisation.

2.2.3.1 Raffinement du cas d'utilisation "Gérer mission"

Notons qu'une mission est une suite de points géographiques enregistrée dans le système du robot, représentant l'unité de l'itinéraire parcouru lors des patrouilles autonomes. L'agent peut ajouter et supprimer des missions.

La figure 2.3 représente le diagramme de cas d'utilisation raffiné "Gérer mission".

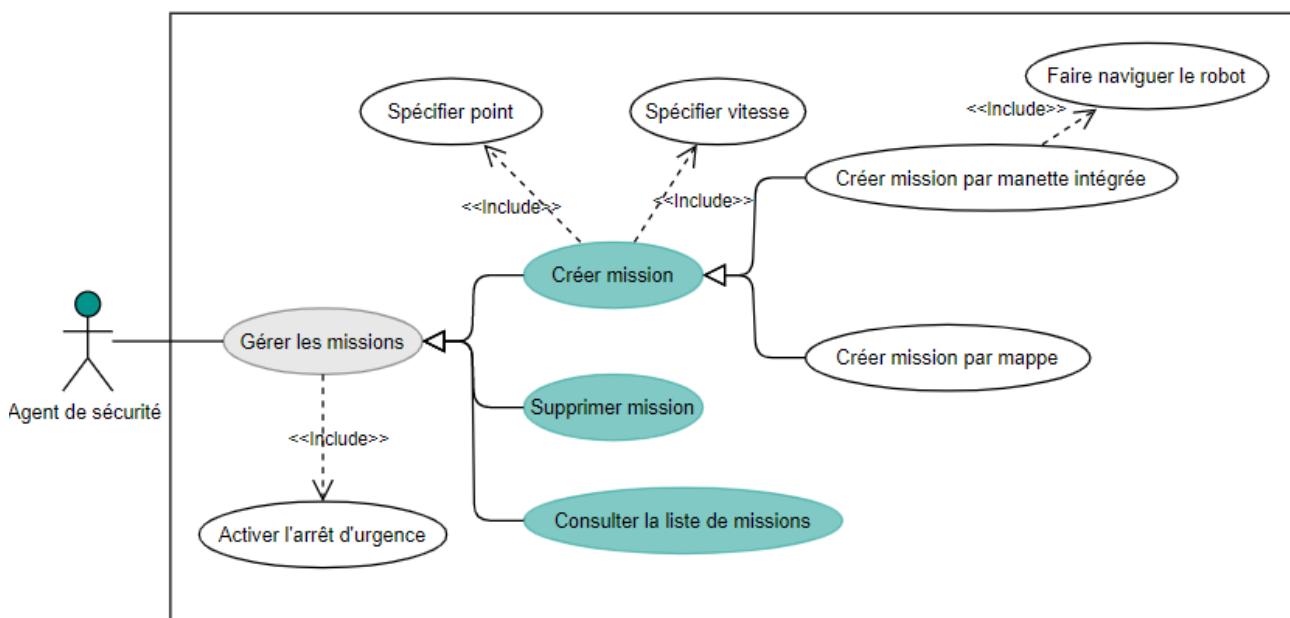


FIGURE 2.3 – Diagramme de cas d'utilisation "Gérer mission"

La création d'une nouvelle mission consiste à spécifier l'ensemble de points géographiques traversés par le robot lors des patrouilles effectuées. Il est possible de créer la mission par deux modes différents :

- Mode « Joy » : L'agent compte sur la manette intégrée dans l'application pour guider le robot vers les points désirés, spécifier la vitesse du robot à chaque point, et les sélectionner.
- Mode « Map » : L'agent sélectionne la liste de points en appuyant sur la carte affichant la localisation du robot et l'espace environnant.

Le tableau 2.2 indique les tâches à exécuter par l'utilisateur pour ajouter une mission par la manette intégrée dans l'application.

Titre	Créer mission par manette intégrée
Acteur	Agent de sécurité
Préconditions	Le système fonctionne, l'agent doit être authentifié et il doit activer l'arrêt d'urgence du robot (Robot en état « Waiting »).
Postconditions	La mission doit être ajoutée et enregistrée dans le robot.
Scénario nominal	<ol style="list-style-type: none"> 1. L'agent lance l'application et s'authentifie. 2. Il choisit l'item « Mission » du menu. 3. L'application affiche la liste des missions existantes. 4. L'agent active l'arrêt d'urgence. 5. Il clique sur le bouton «Add» puis spécifie un nom pour la mission. 6. L'agent spécifie le mode d'ajout par manette intégrée. 7. L'application affiche la localisation actuelle du robot ainsi que la manette pour contrôler le robot. 8. L'agent guide le robot vers les points désirés. 9. L'agent valide les points 10. L'agent définit la vitesse de passage par chaque point. 11. Il valide la création du mission.
Scénario d'exception	<p>11.a. L'agent annule l'opération de création de la mission.</p>

TABLE 2.2 – Description textuelle du cas d'utilisation "Créer mission par manette intégrée"

2.2.3.2 Raffinement du cas d'utilisation "Gérer Scenario"

Un scenario est un ensemble de missions répété infiniment ou un nombre fini de fois.

L'agent peut créer, ou supprimer un scenario. Il peut aussi lancer un scenario c'est-à-dire le spécifier et indiquer au robot de l'effectuer en toute autonomie.

La figure 2.4 représente le diagramme de cas d'utilisation raffiné "Gérer scenario".

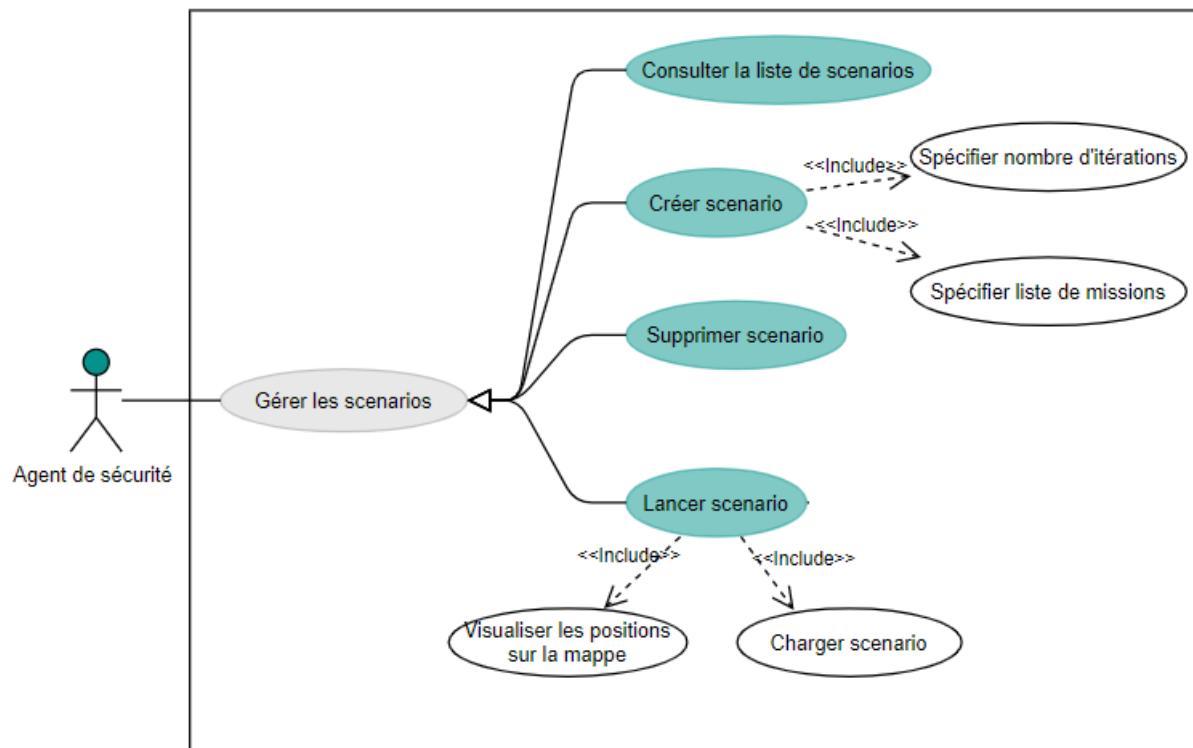


FIGURE 2.4 – Diagramme de cas d'utilisation "Gérer scenario"

Le tableau 2.3 indique les tâches à exécuter par l'utilisateur pour ajouter un scenario.

Titre	Créer scenario
Acteur	Agent de sécurité
Préconditions	Le système fonctionne, l'agent doit être authentifié.
Postconditions	Le scenario doit être ajouté, enregistré dans le robot et prêt à être lancé.
Scénario nominal	<ol style="list-style-type: none"> 1. L'agent lance l'application et s'authentifie. 2. Il choisit l'item « Scenario » du menu. 3. L'application affiche la liste des scénarios existants. 4. L'agent clique sur le bouton « Add ». 5. L'application affiche le pop-up des options du nouveau scenario. 6. L'agent spécifie un nom pour le scenario, le nombre d'itérations, et l'ordre d'exécution des missions puis clique sur le bouton « New Scenario ». 7. L'application affiche toute la liste des missions enregistrées dans le robot. 8. L'agent sélectionne les missions désirées et spécifie la pause après l'exécution de chaque mission. 9. L'application affiche un pop-up de confirmation de l'opération. 10. L'agent valide la création du scenario.
Scénario d'exception	<ol style="list-style-type: none"> 11.a. L'agent annule l'opération de création du scenario.

TABLE 2.3 – Description textuelle du cas d'utilisation "Ajouter scenario"

2.2.3.3 Raffinement du cas d'utilisation "Gérer la navigation"

L'agent peut gérer la navigation du robot et le guider d'un endroit à un autre de deux méthodes différentes. Il dispose, d'une part, d'une manette physique séparée de l'appareil de commande, qu'il peut utiliser après activer son fonctionnement à travers un bouton spécifique de l'application. D'autre part, il peut assurer le déplacement du robot en utilisant directement la manette logique intégrée dans l'application mobile.

La figure 2.5 représente le diagramme de cas d'utilisation raffiné "Gérer la navigation".

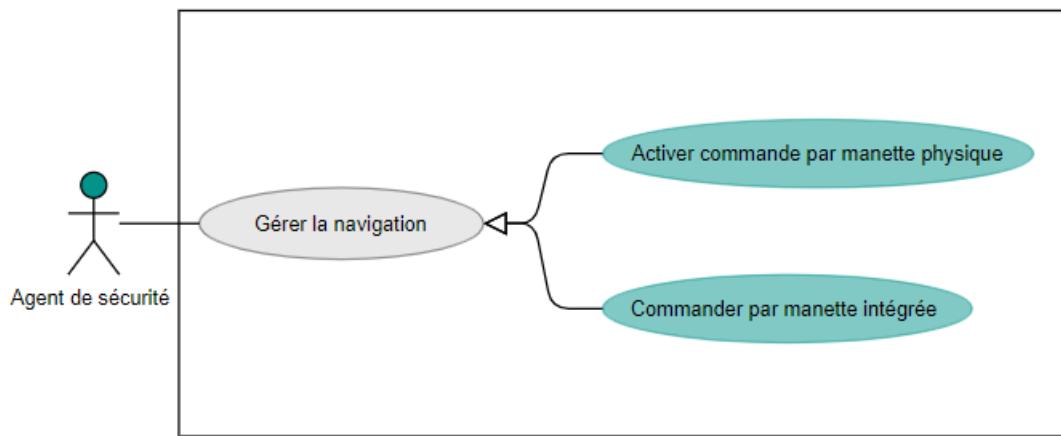


FIGURE 2.5 – Diagramme de cas d'utilisation "Gérer la navigation"

Le tableau 2.4 indique les tâches à exécuter par l'utilisateur pour activer la téléopération par manette physique.

Titre	Activer commande par manette physique
Acteur	Agent de sécurité
Préconditions	Le système fonctionne, l'agent doit être authentifié. La téléopération par manette logique est activée.
Postconditions	La téléopération par manette physique est activée.
Scénario nominal	<ol style="list-style-type: none"> 1. L'agent lance l'application et s'authentifie. 2. Dans l'interface initiale de l'application, l'agent dispose d'un bouton « Teleop » qui indique le mode de téléopération courant. 3. L'agent clique sur le bouton pour donner la main à la manette physique. 4. L'application envoie une commande pour activer le mode désiré.

TABLE 2.4 – Description textuelle du cas d'utilisation "Activer commande par manette physique"

2.2.3.4 Raffinement du cas d'utilisation "Contrôler l'état du robot"

La figure 2.6 représente le diagramme de cas d'utilisation raffiné "Contrôler l'état du robot".

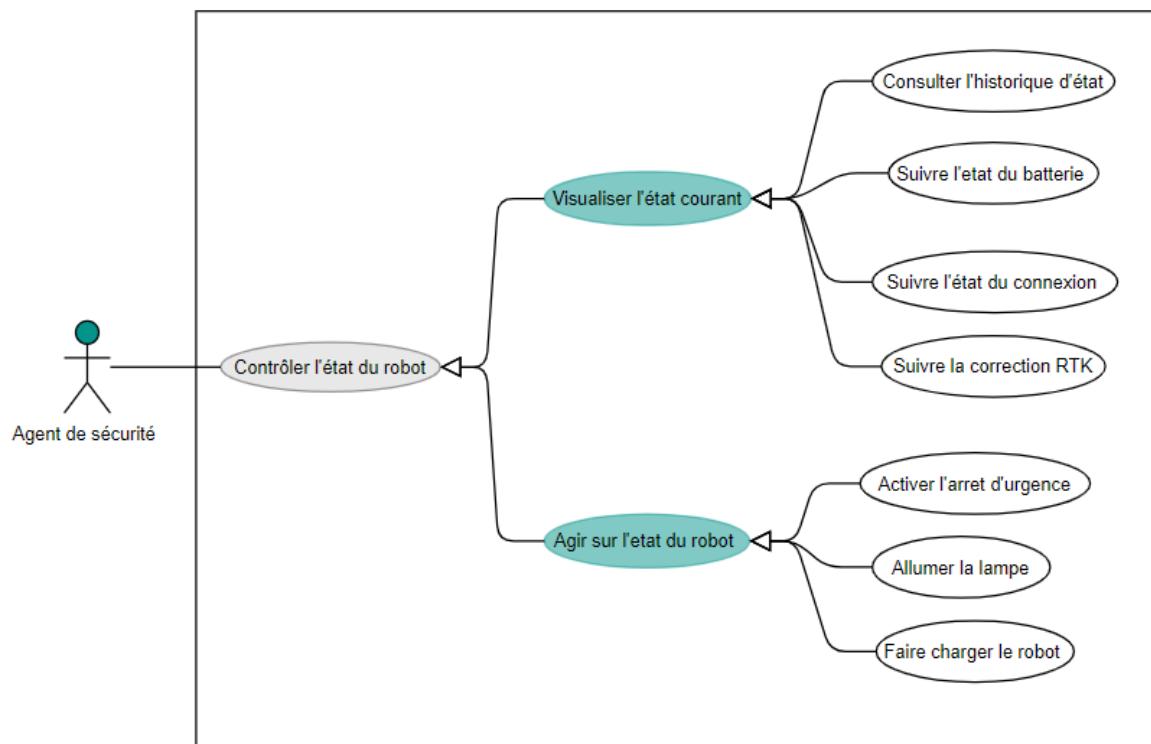


FIGURE 2.6 – Diagramme de cas d'utilisation "Contrôler l'état du robot"

Le tableau 2.5 indique les tâches à exécuter par l'utilisateur pour faire charger le robot au besoin.

Titre	Faire charger le robot
Acteur	Agent de sécurité
Préconditions	Le système fonctionne, l'agent doit être authentifié. Le robot doit être en exécution d'un scenario contenant une mission de recharge.
Postconditions	Le robot se dirige vers la station de recharge.
Scénario nominal	<ol style="list-style-type: none"> 1. L'agent lance l'application et s'authentifie. 2. L'application détecte un niveau faible de la batterie du robot. 3. Elle affiche un message d'alerte indiquant le niveau de la batterie ainsi que la distance séparant le robot de sa station de recharge. 4. L'agent clique sur l'item « Settings » du menu. 5. L'agent lance un ordre au robot pour interrompre son fonctionnement et se diriger vers sa station de recharge.

TABLE 2.5 – Description textuelle du cas d'utilisation "Faire charger le robot"

2.2.3.5 Raffinement du cas d'utilisation "Consulter les flux vidéo transmis"

La figure 2.7 représente le diagramme de cas d'utilisation raffiné "Consulter les flux vidéo transmis".

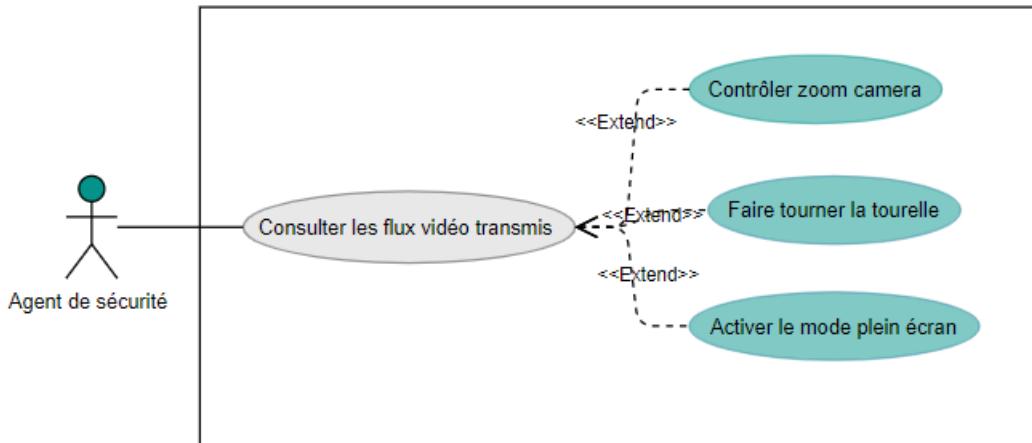


FIGURE 2.7 – Diagramme de cas d'utilisation "Consulter les flux vidéo transmis"

Le tableau 2.6 indique les tâches à exécuter par l'utilisateur pour faire tourner la tourelle du robot.

Titre	Faire tourner la tourelle
Acteur	Agent de sécurité
Préconditions	Le système fonctionne, l'agent doit être authentifié.
Postconditions	L'orientation de la tourelle du robot est changée.
Scénario nominal	<ol style="list-style-type: none"> 1. L'agent lance l'application et s'authentifie. 2. Il choisit l'item « Camera » du menu. 3. L'application affiche les vidéos transmises par le robot ainsi qu'un panneau de contrôle des caméras et de la tourelle. 4. L'agent glisse le slider pour indiquer le nouvel angle. 5. L'application envoie une commande pour changer l'orientation de la tourelle. 6. L'agent visualise le nouvel angle de la tourelle à travers le flux vidéo affiché.

TABLE 2.6 – Description textuelle du cas d'utilisation "Faire tourner la tourelle"

2.2.4 Analyse

Cette phase marque le début de l'analyse détaillée du système à réaliser. Elle va permettre d'illustrer les concepts dynamiques d'UML.

Nous allons présenter les opérations des processus et leurs conséquences sur les objets définis dans la partie statique de la description des besoins fonctionnels. Nous allons utiliser le diagramme d'activités pour décrire le déroulement des cas d'utilisation.

2.2.4.1 Diagramme d'activité : "Créer mission par manette intégrée"

Le diagramme 2.8 décrit le flux d'activités du cas d'utilisation "Créer mission par manette intégrée".

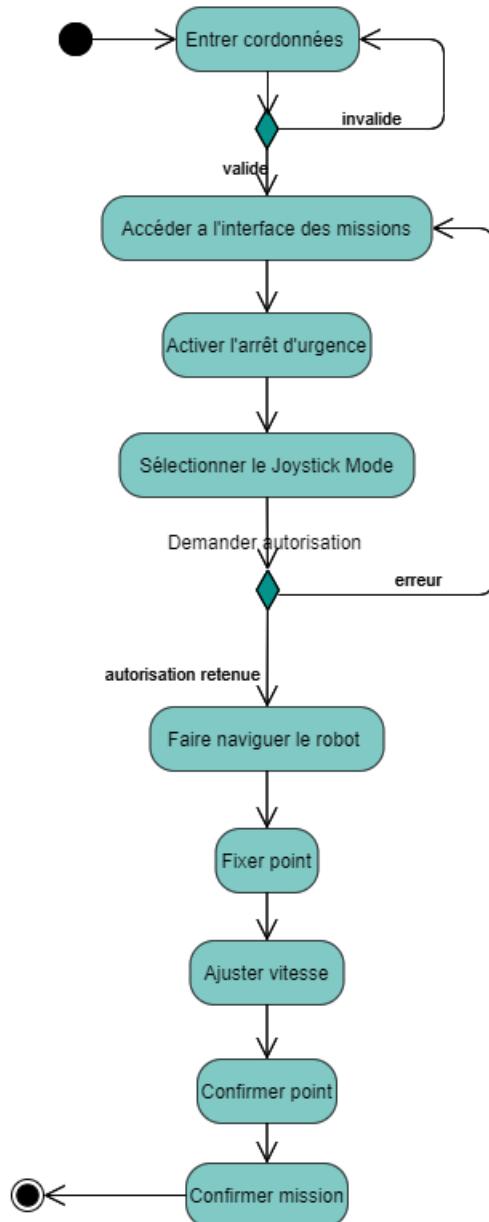


FIGURE 2.8 – Diagramme d'activité du cas d'utilisation "Créer mission par manette intégrée"

2.2.4.2 Diagramme d'activité : "Consulter les flux vidéo transmis"

L'application offre à l'agent de sécurité la possibilité de consulter les flux vidéo et faire la surveillance des sites à distance. A travers l'interface qui visualise les flux, l'agent peut contrôler la tourelle du robot et le zoom de la caméra.

Le diagramme 2.9 décrit le flux d'activités du cas d'utilisation "Consulter les flux vidéo transmis".

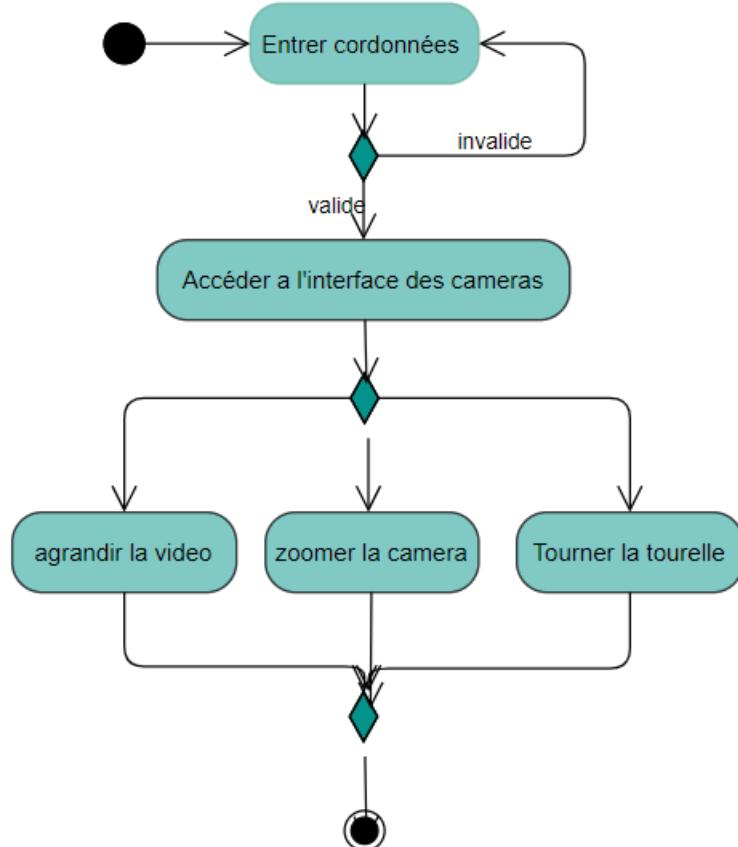


FIGURE 2.9 – Diagramme d'activité du cas d'utilisation "Consulter les flux vidéo transmis"

2.2.4.3 Diagramme d'activité : "Créer scenario"

Le diagramme 2.10 décrit le flux d'activités du cas d'utilisation "Créer scenario".

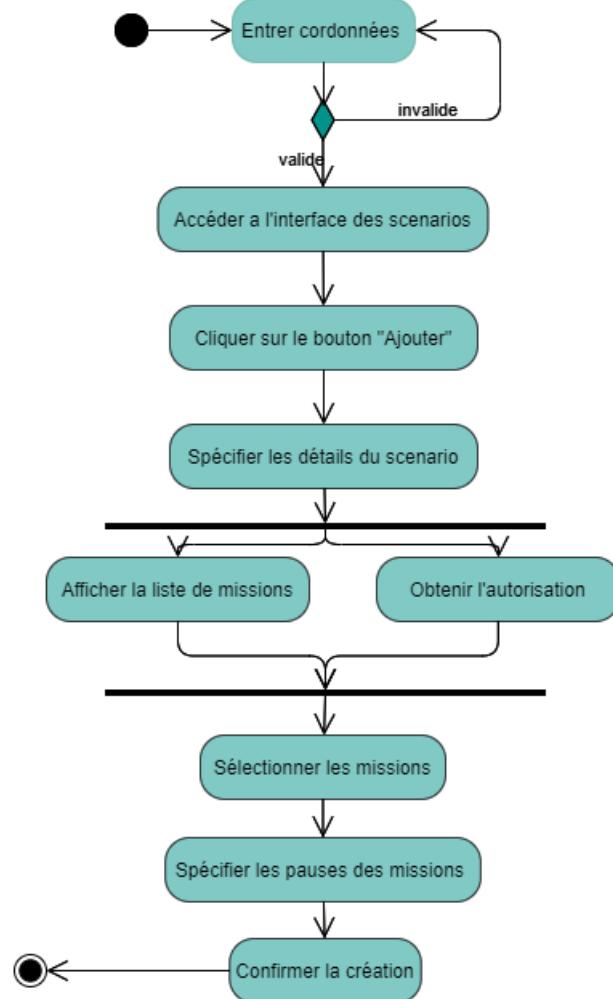


FIGURE 2.10 – Diagramme d’activité du cas d’utilisation "Créer scenario"

2.2.4.4 Diagramme d’activité : "Lancer scenario"

L’une des fonctionnalités de base offertes par l’application est le lancement d’un scenario. Elle consiste à spécifier un itinéraire, le charger dans le robot, puis indiquer à ce dernier de commencer les rondes en toute autonomie. Le robot peut patrouiller une nuit entière en effectuant le scenario indiqué par l’agent.

Pour accomplir cette fonctionnalité, l’agent doit suivre un flux d’actions bien déterminé décrit par le diagramme d’activités ci-dessous.

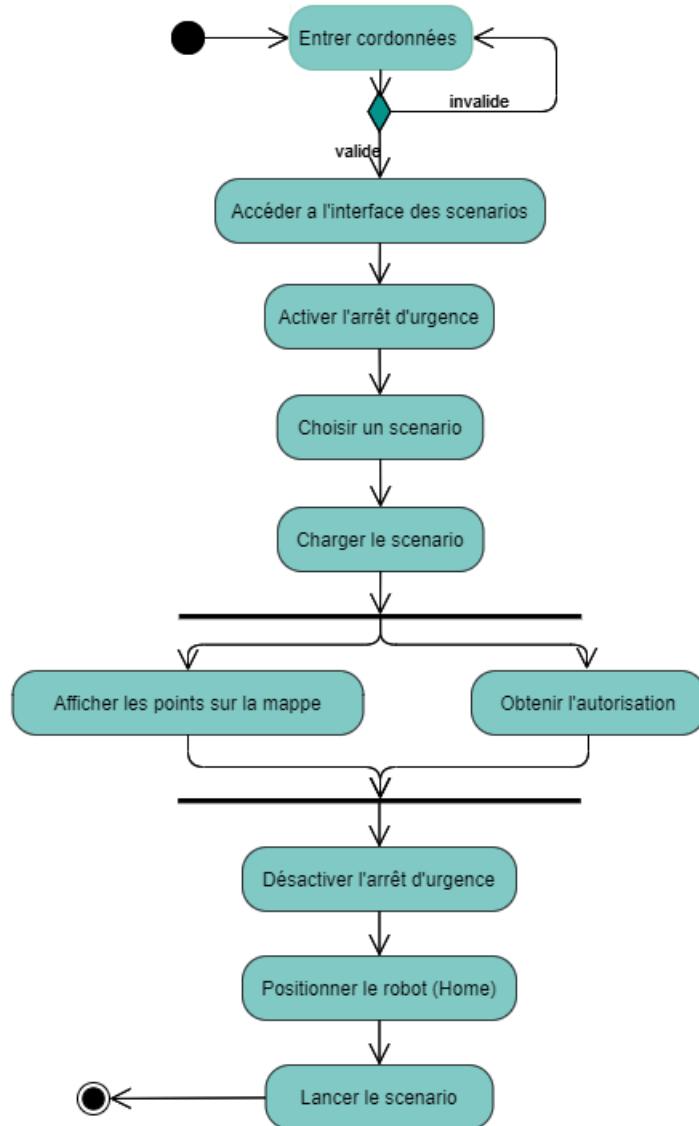


FIGURE 2.11 – Diagramme d'activité du cas d'utilisation "Lancer scenario"

2.3 Branche technique

Dans cette section, nous présenterons la deuxième branche du processus 2TUP. En effet, cette branche recense toutes les contraintes à respecter pour réaliser le système, les choix non-fonctionnels ainsi que les besoins opérationnels, indépendamment des besoins fonctionnels du système.

2.3.1 Besoins non fonctionnels

Les besoins non fonctionnels représentent les caractéristiques du système. Ils concernent les contraintes à prendre en considération pour mettre en place une solution adéquate.

La définition et la mise en place de ces besoins est essentiel comme l'application va être utilisée par des utilisateurs non experts en informatique et comme le robot prend en charge une tâche critique qui est la sécurité des sites.

- **L'ergonomie :** Les interfaces doivent être intuitives et faciles à comprendre ainsi qu'attrayantes et agréables à l'œil. L'autre chose à prendre en compte en matière d'ergonomie, est la concision des données. L'IHM donne à l'utilisateur ce qu'il cherche rapidement et facilement.
- **La disponibilité :** Comme il s'agit d'une application connectée à un robot souvent en marche et en fonctionnement, qui transmet des données et des flux en temps réel, les services doivent être disponibles en permanence pour les utilisateurs, couvrant un espace géographique entendu.
- **Fiabilité :** Le système doit assurer un fonctionnement fiable avec une bonne capacité de rétablissement et une courte durée de récupération.
- **Évolutivité :** Les fonctionnalités offertes par le P-Guard peuvent évoluer et être étendues donc l'application doit supporter l'ajout d'autres modules pour garantir la souplesse.
- **Compatibilité :** Le système doit être compatible avec différents systèmes d'exploitation et doit offrir une réactivité qui renforce la compatibilité avec le matériel qui peut être un smartphone ou une tablette.
- **Sécurité :** Le système doit impérativement garantir l'intégrité et la confidentialité des données transmises et de toutes opération effectuée. Les données utilisateurs doivent être aussi sécurisées, en effet, la création de comptes utilisateur se fait lors de la configuration initiale après la vente du robot et de l'application de commande.

2.3.2 Contraintes techniques

2.3.2.1 Le WebSocket

Une fois terminé, le recueil des besoins fonctionnels, la première contrainte technique a étudiée est la communication à travers le Web socket. Le système du robot s'appuie sur le web socket comme le moyen principal de communication pour transmettre les données en temps réel.

La communication entre un client et un serveur se fait en général, sur Internet, par le biais d'une connexion http.

En HTTP, pour consulter un site Web, le client doit d'abord envoyer une requête au serveur. Celui-ci peut ensuite répondre et transmettre le contenu désiré. Il s'agit d'un simple modèle de requête et de réponse, qui finit par occasionner un délai important entre la requête et la réponse.

L'utilisation d'un Web Socket permet la communication dynamique, en temps réel.

La connexion entre le client et le serveur s'établit grâce à la phase de handshake du protocole WebSocket. Dans ce cas, le client envoie toutes les identifications nécessaires à l'échange de données au serveur. Une fois établi, le canal de communication reste semi-ouvert. Le serveur peut s'activer de lui-même et transmettre au client toutes les informations sans que le client ne les demande.

La figure 2.12[11] décrit une session de communication par le biais du Web Socket.

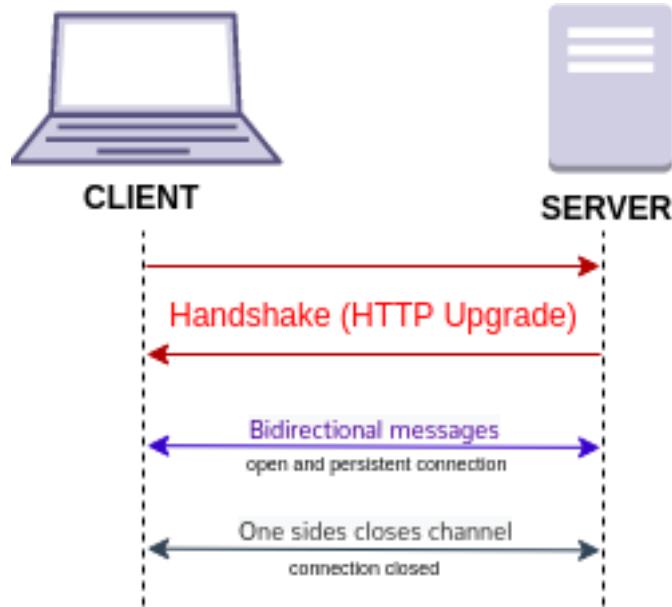


FIGURE 2.12 – Schéma décrivant une session WebSocket

2.3.2.2 Le protocole RTSP

Une autre spécification technique est la transmission du flux vidéos en temps réel par le biais du protocole RTSP.

Le RTSP (Real Time Streaming Protocol) a été conçu pour envoyer de l'audio ou de la vidéo, en direct, d'un appareil à un autre, il permet aux systèmes de streaming de communiquer, en permettant un accès à plusieurs fonctionnalités comme la position temporelle ou encore la « lecture » et la « pause » de la vidéo.

RTSP agit comme une voie pour le transport des données vidéo du point A (caméra) au point B (VLC Player/ou RTSP viewers).

En ce sens, une caméra IP est capable de diffuser des vidéos en direct sur des lecteurs multimédias compatibles RTSP tels que le lecteur multimédia VLC [12].

La figure 2.13[13] représente une communication par le RTSP.

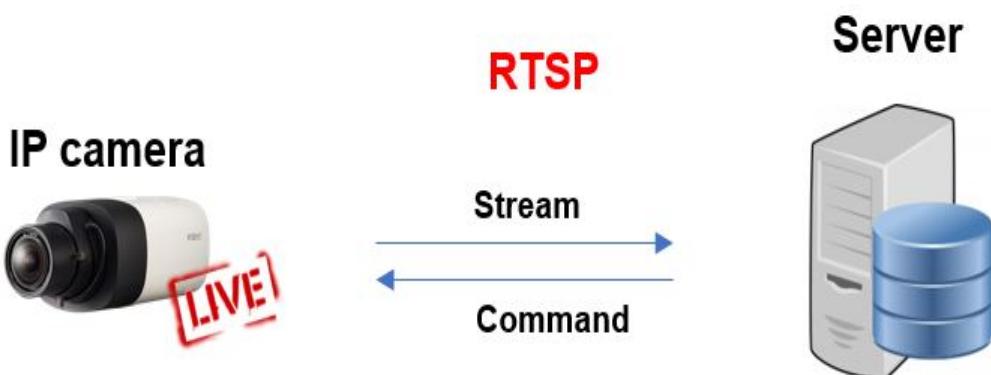


FIGURE 2.13 – Schéma décrivant la communication via RTSP

2.3.2.3 Le système ROS

ROS (Robot Operating System), un système important intégré dans le robot P-Guard, qui est un méta-système d'exploitation standardisé qui apporte différentes fonctionnalités pour la robotique.

- L'abstraction de la couche « matériel » (hardware).
- Le contrôle des périphériques de bas niveau du robot.
- La mise en œuvre simplifiée de fonctionnalités couramment utilisées.
- La transmission de messages entre processus.

Un programme ROS est formé d'un ensemble d'éléments appelés 'Les Nodes', chaque nœud réagit sur un driver d'un périphérique connecté pour exécuter une tache ou une action précise (Ex : piloter les moteurs, faire tourner la tourelle).

2.3.3 Conception générique

La branche technique permet principalement de définir les composants nécessaires à la construction de l'architecture technique. Dans cette section, nous allons alors présenter et définir ces composants, ainsi que le diagramme de déploiement du système.

2.3.3.1 Architecture technique

L'architecture présentée sur la figure 2.14 est déployée sur deux supports physiquement séparés : sur **l'appareil de commande** (Tablette ou Smartphone) et sur **le robot**. La communication entre ces deux éléments est basée sur une connexion via le réseau internet : réseau mobile 4G et une autre via réseau Wifi.

Le premier module est implanté sur l'appareil de commande. Il s'agit d'une application mobile hybride, contenant un ensemble de dépendances techniques, qui représente le client dans ce système.

Le deuxième module est déployé sur le robot, il est composé de deux sous modules principales. Aucune communication entre ces deux derniers n'est nécessaire étant donné que chacun joue un rôle bien spécifique.

Les deux modules communiquent avec l'appareil de commande via le réseau Internet. Ils sont organisés de la manière suivante :

- Le premier représente **le serveur d'authentification** qui fournit un service de validation des informations entrées par l'agent de sécurité.
L'application (le client) demande de se connecter au serveur d'authentification déployé sur le robot en envoyant une **requête http**.

Après le processus d'authentification, le serveur renvoie un Jeton qui sera utilisé dans la prochaine communication avec le robot.

- Le deuxième représente **un serveur de données** qui couvre la majorité de la communication avec l'application cliente. La communication à ce niveau se fait par le biais du

web socket.

Une fois le jeton est récupéré du serveur d'authentification, l'application doit l'inclure dans sa demande de connexion au web socket. Grâce aux caractéristiques du web socket, le robot va communiquer en permanence des informations sur son état ainsi que sa localisation sous la forme de **fichiers JSON**.

La transmission des flux vidéo se fait à travers un canal totalement différent, c'est en effet par le biais du **protocole RTSP** qui permet d'extraire le flux en direct de la caméra du robot à partir de son **adresse IP** et de la visualiser à travers un plugin VLC.

Un autre outil important pour le développement d'une application qui répond aux besoins fonctionnels précisés auparavant est le **SDK de Mapbox Maps**. Il s'agit d'un ensemble d'outils open-source permettant de créer des applications de cartographie. Le SDK offre une flexibilité pour le style visuel et la personnalisation [14].

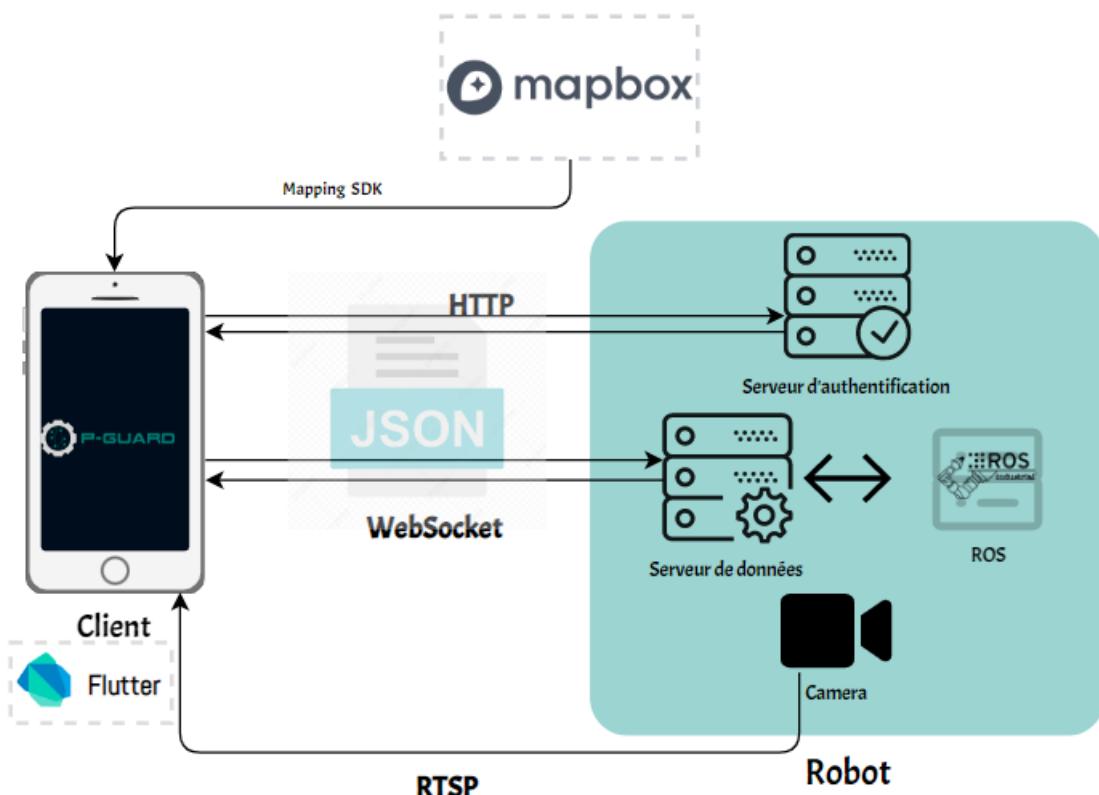


FIGURE 2.14 – Architecture physique du système

2.3.3.2 Diagramme de déploiement

Le diagramme de déploiement modélise les ressources physiques de la solution, c'est-à-dire il décrit les noeuds sur lesquels fonctionnera le système. Chaque noeud peut contenir plusieurs composants.

Le schéma 2.15 décrit le diagramme de déploiement du système.

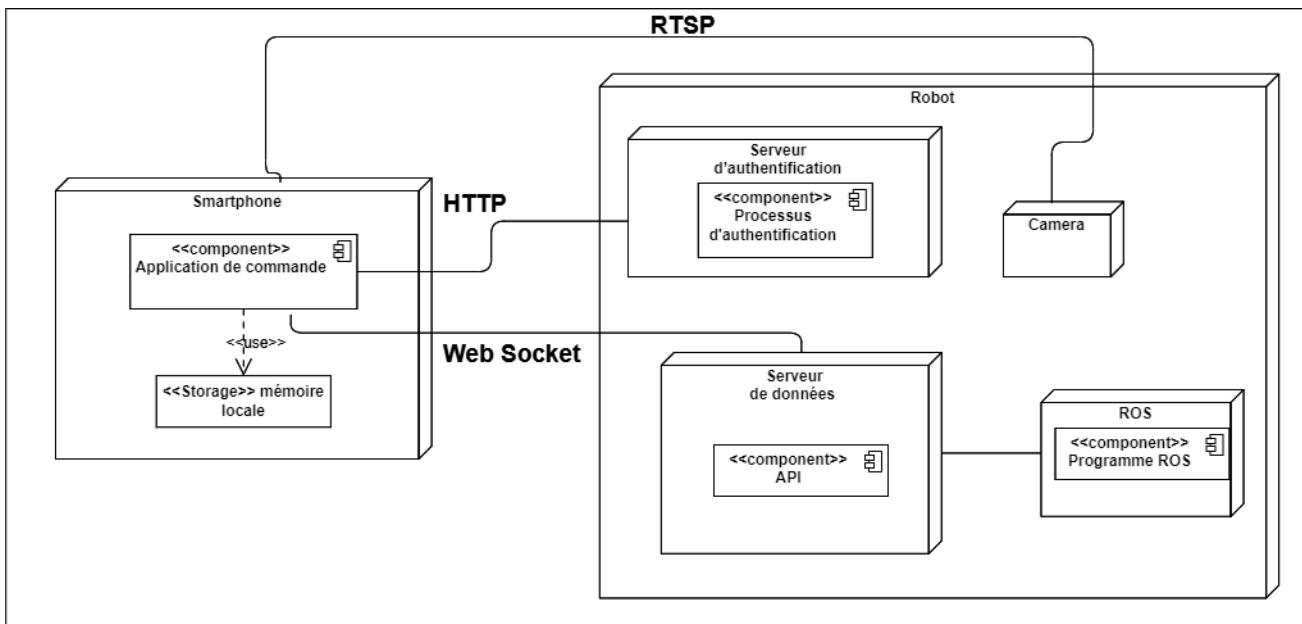


FIGURE 2.15 – Diagramme de déploiement du système

Conclusion

Tout au long de ce chapitre, nous avons élaboré les premières étapes de la méthodologie 2TUP, nous avons commencé par présenter la modélisation du contexte, puis la branche fonctionnelle ainsi que son analyse détaillée. Finalement, nous avons définit les spécifications techniques et la conception générique.

Ce chapitre a permis de définir les besoins fonctionnels et techniques de l'application, ce qui mène à entamer la phase de la conception détaillée pour assurer une bonne mise en œuvre d'un système fonctionnel répondant aux besoins cités.

Chapitre 3

Conception détaillée

Plan

3.1	Prototype des interfaces utilisateur	44
3.2	Architecture logique de l'application	45
3.2.1	Introduction à l'architecture MVC	45
3.2.2	Architecture GETX	46
3.3	Vue statique de l'application : Diagramme de classes	47
3.4	Vue dynamique de l'application	49
3.4.1	Diagrammes de séquence	49
3.4.2	Diagramme d'états-transitions	58

Introduction

Après avoir décortiqué les besoins fonctionnels et non fonctionnels et les principales fonctionnalités et contraintes techniques du système, nous entamons dans cette partie l'étude conceptuelle détaillée de l'application en décrivant son architecture logique et sa modélisation interne à travers le patron d'architecture utilisé, le diagramme de classes, ainsi qu'un ensemble de diagramme UML décrivant l'interaction internes des composants de l'application.

Cette étape est très critique dans le cycle de développement de notre application puisqu'elle permet satisfaire les besoins dégagés dans la phase spécification d'une part, et prépare à l'implémentation d'autre part.

3.1 Prototype des interfaces utilisateur

C'est une technique qui consiste à préparer les interfaces graphiques de l'application à l'aide d'un outil de conception de prototype afin de mesurer le degré de satisfaction du client à propos la compréhension du projet. L'interaction qui se produit entre l'utilisateur final et le développeur, comme résultat de la discussion de ces interfaces, les permet d'ajuster et d'implémenter les besoins exacts.

La Figure 3.1 montre le prototype d'interfaces utilisateur de l'application implémenté par Adobe XD.

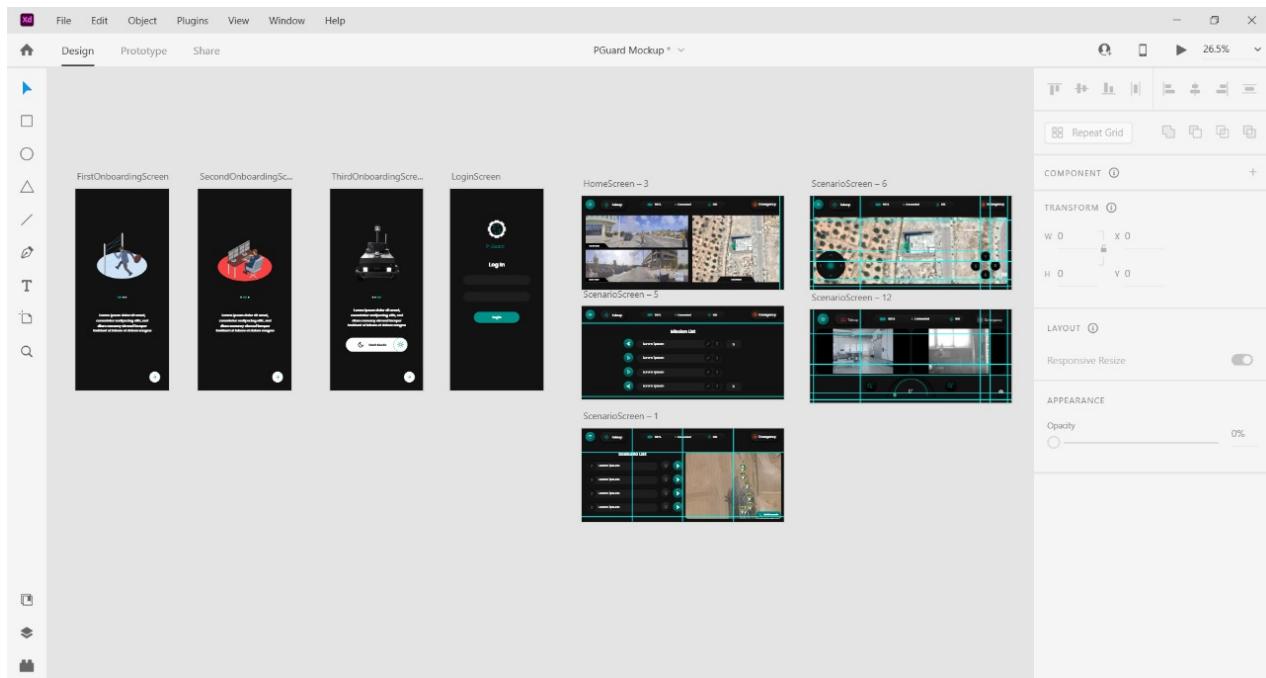


FIGURE 3.1 – Prototype des interfaces utilisateur de l'application mobile

La conception des interfaces d'une application mobile doit être extrêmement précise car nous devons implémenter toutes les fonctionnalités dont nous avons besoin dans un petit écran d'une manière ergonomique qui facilite l'utilisation de l'application. Comme notre application mobile possède de nombreuses fonctionnalités on a dû concevoir les interfaces utilisateur avant de commencer la phase de développement pour s'assurer que tous les services existent convenablement.

3.2 Architecture logique de l'application

Dans cette section, nous allons présenter le patron d'architecture qui sera utilisé dans le développement de notre projet.

3.2.1 Introduction à l'architecture MVC

Notre application va pratiquement suivre les caractéristiques du patron d'architecture, fréquemment utilisé, Le patron **MVC**.

C'est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au niveau de leur architecture.

Ce paradigme regroupe les fonctions nécessaires en trois catégories : **un modèle**, **une vue**, et **un contrôleur**.

La figure 3.2 illustre le modèle MVC lors de son utilisation dans notre application.

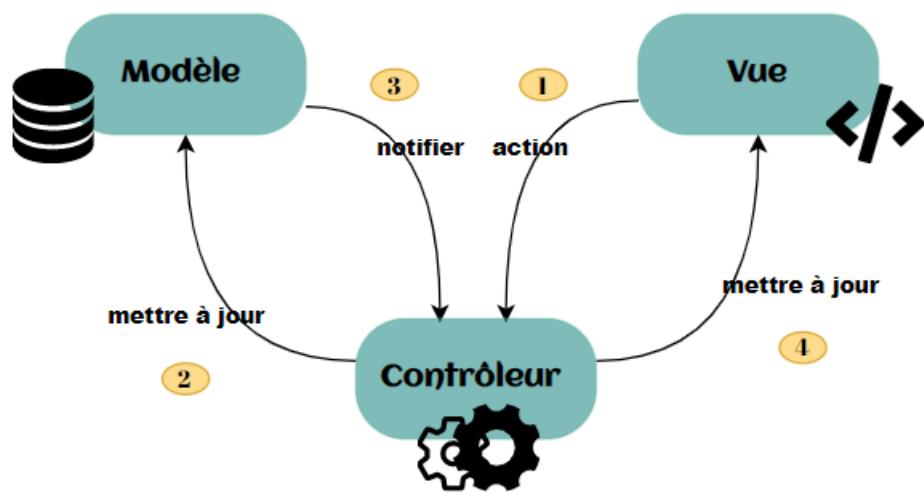


FIGURE 3.2 – Illustration du modèle MVC

Le patron MVC permet de bien organiser l'application. Il va nous aider à savoir quels fichiers créer pour quelle fonctionnalité, et surtout à définir leurs rôles. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts :

- **Modèle** : Cette partie gère les données de notre application. Son rôle est de récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur.
 - Parmi les modèles implémentés dans notre application, nous citons «RobotInfo» «UserInfo» «ScenarioInfo»
- **Vue** : Cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher en retour donc c'est un moyen d'affichage des objets dans une application. Elle permet à l'utilisateur également de modifier les données.
 - Parmi les vues implémentées dans notre application, nous citons «HomeScreen» «MissionScreen» «CameraScreen».
- **Contrôleur** : Cette partie agit comme une interface entre le modèle et la vue, pour traiter toute la logique métier et les requêtes entrantes, manipuler les données à l'aide du composant Modèle et interagir avec les Vues pour rendre le résultat final. Le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le contenu à afficher à la vue.
 - Parmi les contrôleurs qui seront implémentés dans notre application, nous citons «WebSocketController» «MenuController» «HomeOptionsController».

Il faut retenir que le contrôleur est le chef d'orchestre dans notre application : c'est lui qui reçoit les requêtes de l'agent de sécurité et qui contacte d'autres fichiers (le modèle et la vue) pour échanger les informations avec eux.

3.2.2 Architecture GetX

GetX est une solution extra-légère et puissante pour Flutter. Il combine une gestion d'état hautes performances, une injection de dépendance intelligente et une gestion du navigation rapide et pratique.

GetX impose une organisation architecturale pour le projet bien spécifique. En effet, il permet le découplage total de la vue, de la logique de présentation, de la logique métier, et de la navigation.

Il permet de savoir où trouver chaque fonctionnalité de notre application, avec un code propre par défaut. En plus de faciliter la maintenance. GetX est un moyen pratique et évolutif de création d'applications hautes performances qui suit le modèle MVC. La figure 3.3 [15] représente l'architecture de GETX.

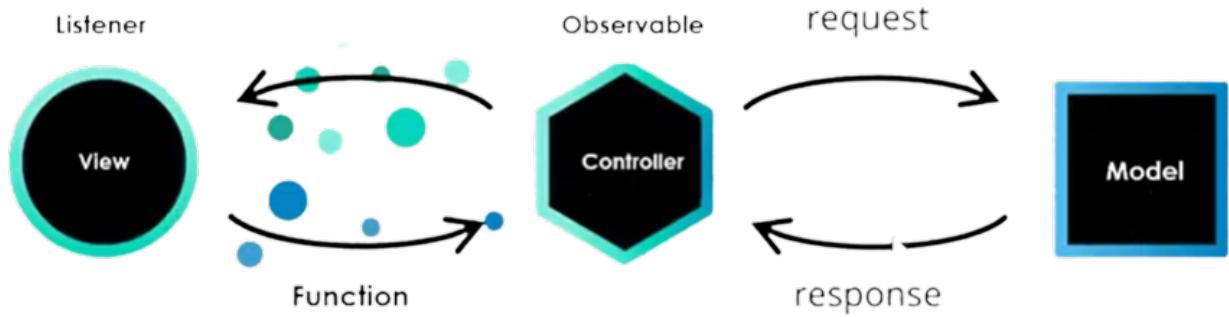


FIGURE 3.3 – Présentation de l'architecture de GETX

- GetX présente le contrôleur comme un objet observable, qui observe tout changement dans l'état des variables ou précisément au niveau des modèles.
- Les vues jouent le rôle d'écouteurs qui attendent les décisions et les traitements du contrôleur pour mettre à jour l'affiche visualisé par l'utilisateur.

3.3 Vue statique de l'application : Diagramme de classes

Le diagramme de classe est l'un des diagrammes UML les plus répandus. Il permet de décrire la structure interne du système pour mieux expliquer son logique métier.

Les différentes classes persistantes dans notre application sont :

- **Classe Robot** : C'est l'entité qui encapsule les données relatives au robot.
- **Classe Utilisateur** : C'est l'entité qui encapsule les données relatives à un utilisateur de l'application c'est-à-dire un agent de sécurité.
- **Classe Navigation** : C'est l'entité qui encapsule les données relatives à la méthode de navigation du robot (autonome en lançant un scenario spécifique ou par les manettes).
- **Classe Scénario** : Afin d'assurer la navigation autonome, nous devons créer et lancer des scénarios. Cette classe englobe toutes les données nécessaires d'un scénario.
- **Classe Mission** : Un scénario est formé d'un ensemble de missions. Les missions permettent à un robot de définir la trajectoire qui va être suivie. Une mission est représentée par un ensemble de points et de segments.
- **Classe station de recharge** : C'est l'entité qui encapsule les données relatives à la station de recharge du robot.
- **Classe Localisation** : C'est l'entité qui encapsule les données relatives à un point sur la mappe (latitude et longitude) décrivant la position du robot à un moment donné.
- **Classe Message** : C'est l'entité qui encapsule les données relatives à un message envoyé par le robot à l'application de commande.

La figure 3.4 illustre les principales classes de notre système.

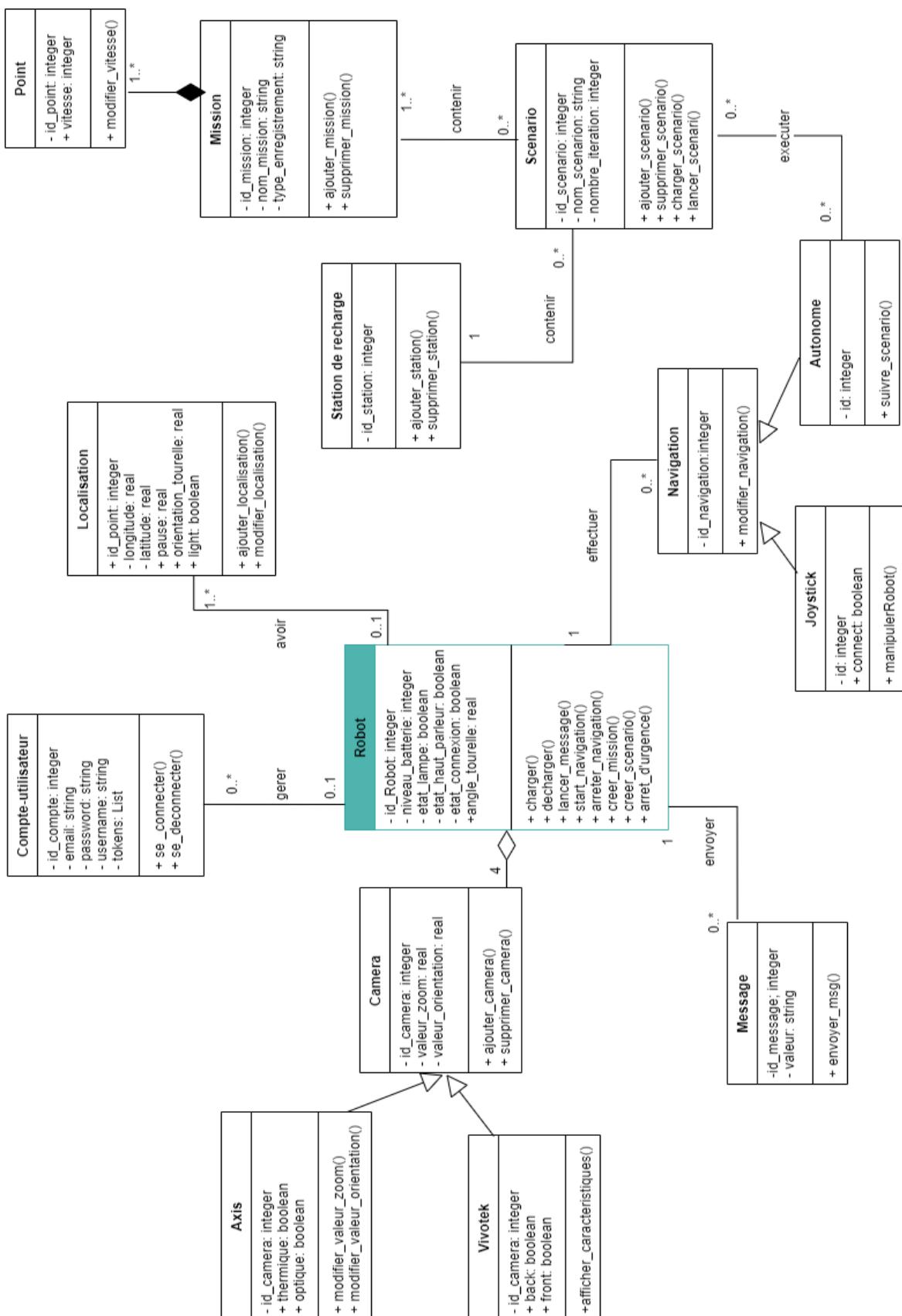


FIGURE 3.4 – Diagramme de classe du système

3.4 Vue dynamique de l'application

La modélisation dynamique d'un système montre son comportement, les interactions de ses objets et leur évolution dans le temps en mettant l'accent sur la chronologie des évènements. Pour présenter l'interaction interne détaillée des objets de notre application ainsi que les transformations d'états possibles, nous allons utiliser les diagrammes de séquence et d'état-transition.

3.4.1 Diagrammes de séquence

3.4.1.1 Concept du scénario

Nous avons décrit précédemment qu'un cas d'utilisation représente un ensemble de scénarios. Lors de l'étape de détermination des besoins fonctionnels, un scénario illustre une séquence d'interactions entre le système et ses acteurs. Nous allons maintenant remplacer le système par une collaboration entre les objets qui le forme.

Pour décrire les scénarios, nous allons utiliser des diagrammes de séquence qui mettent en évidence la chronologie des messages échangés.

3.4.1.2 Diagramme de séquences détaillé du cas d'utilisation : "S'authentifier"

Pour accéder à l'application "P-Guard", l'utilisateur doit s'authentifier afin de pouvoir bénéficier des services fournis.

Tout d'abord, l'utilisateur doit remplir le formulaire d'authentification qui contient son e-mail et son mot de passe, récupérés par le client lors de la première configuration du système, puis l'application vérifiera son compte dans le serveur d'application.

Le serveur envoie en retour un jeton que l'application stocke localement et utilise pour la communication Web socket avec le robot.

Le diagramme 3.5 décrit les scénarios possibles du cas d'utilisation "S'authentifier".

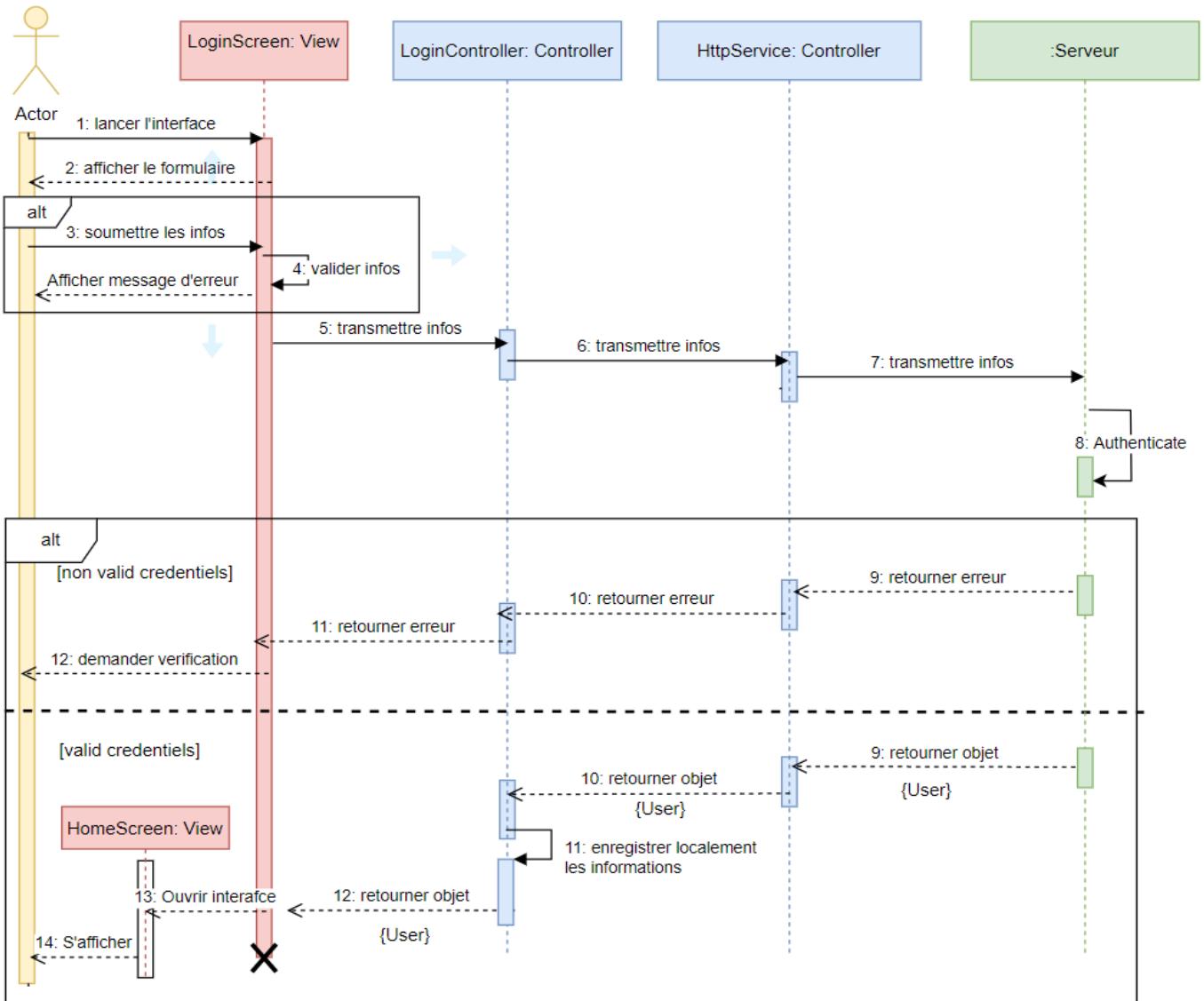


FIGURE 3.5 – Diagramme de séquences du cas d'utilisation "S'authentifier"

3.4.1.3 Diagramme de séquences détaillé du cas d'utilisation : "Vérifier connexion"

L'application doit détecter toute perte de connexion avec le robot et empêcher l'utilisateur d'effectuer des actions dans ce cas, en lui informant des évènements survenus.

La vérification de la présence d'informations transmises par le robot indique l'état de connexion. L'application demande la connexion au robot et reste en écoute de toute information transmise. Dans le cas de réussite de connexion, Le robot envoie en permanence des indications sur son état et sa localisation. L'application sauvegarde l'état et affiche les interfaces en transmettant les données nécessaires. Dans le cas contraire, c'est-à-dire dans le cas de manque de données transmises, l'application affiche l'interface de perte de connexion.

Le diagramme 3.6 décrit les scénarios possibles du cas d'utilisation "Vérifier connexion".

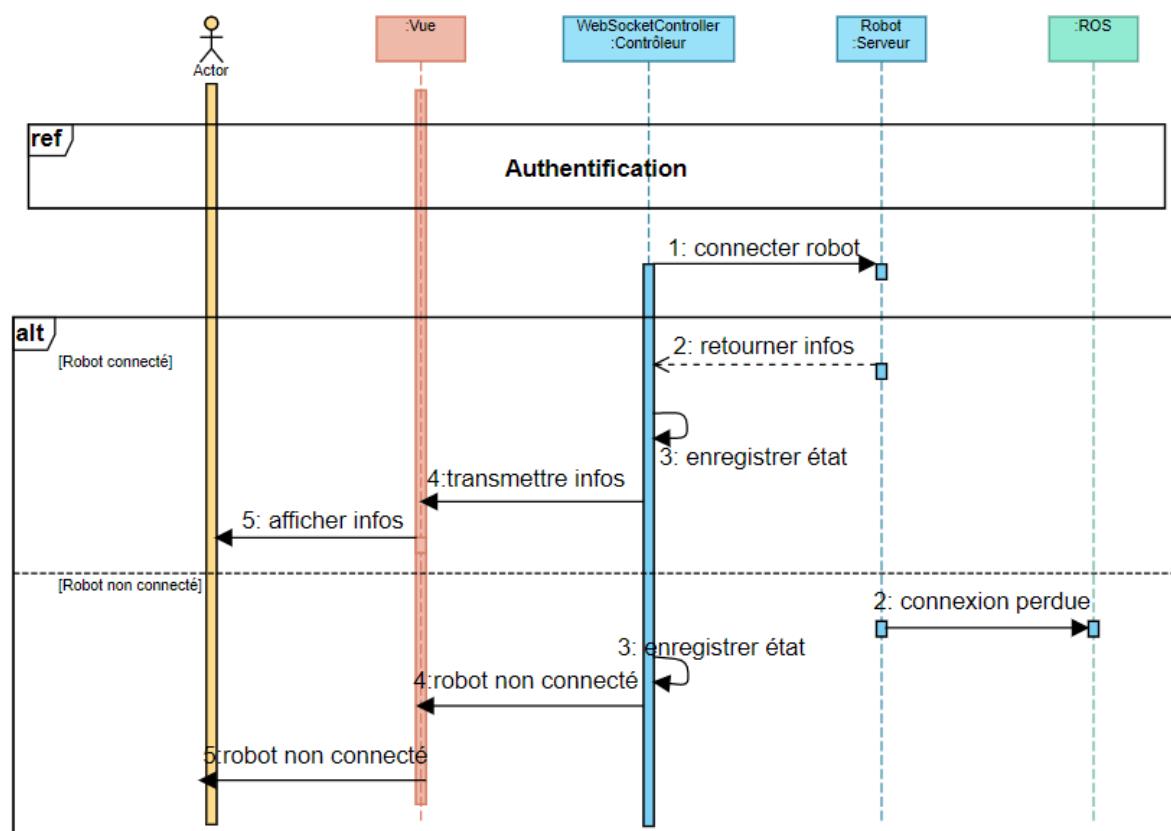


FIGURE 3.6 – Diagramme de séquences du cas d'utilisation "Vérifier connexion"

3.4.1.4 Diagramme de séquences détaillé du cas d'utilisation : "Activer l'arrêt d'urgence"

En cas de besoin, L'agent de sécurité a recours à activer l'arrêt d'urgence du robot pour terminer son fonctionnement immédiatement. Ce cas d'utilisation est aussi nécessaire pour la réalisation d'autres actions comme la création d'une nouvelle mission ou le lancement d'un scenario. L'agent appuie sur le bouton « Emergency Stop », l'application demande l'action du système ROS du robot, et indique la confirmation en changeant la couleur du bouton appuyé.

Le diagramme 3.7 décrit les scénarios possibles du cas d'utilisation "Activer l'arrêt d'urgence".

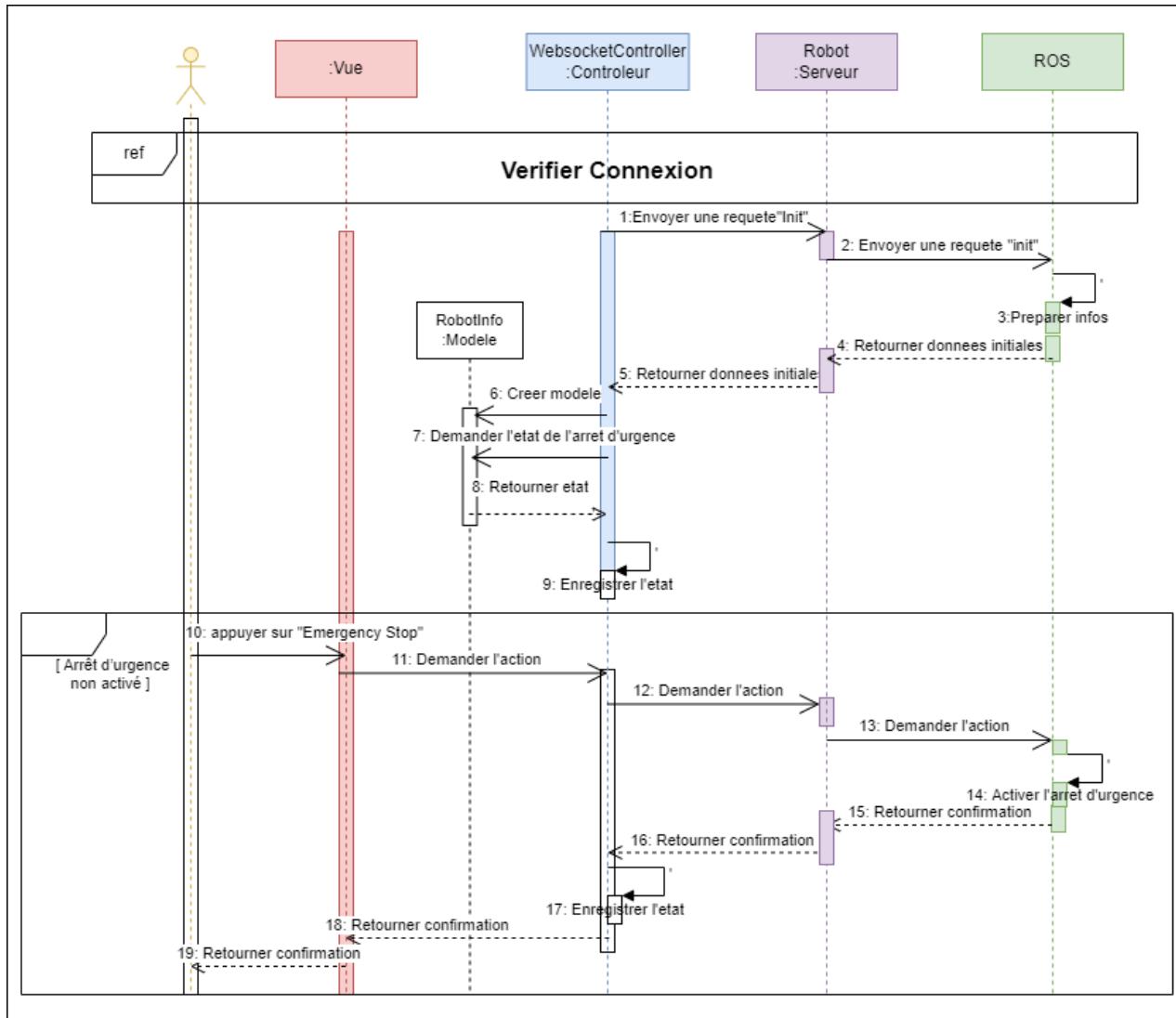


FIGURE 3.7 – Diagramme de séquences du cas d'utilisation "Activer l'arrêt d'urgence"

3.4.1.5 Diagrammes de séquences détaillés du cas d'utilisation : "Créer mission"

- Le diagramme 3.8 décrit les scénarios possibles du cas d'utilisation "Créer mission" par le mode « Joy » dont l'agent peut, au premier lieu choisir un nom pour la mission et le mode de sa création. L'application le dirige vers une interface où il dispose d'une manette avec quatre sens principaux et quatre boutons offrant des fonctionnalités différentes. Il oriente la manette pour déplacer le robot vers le premier point désiré, sélectionne ce point par le premier bouton, spécifie sa vitesse à l'aide du deuxième, et confirme son choix par le troisième bouton, il répète ces actions jusqu'à terminer la confirmation de l'ensemble de points formant la mission. Finalement l'agent doit confirmer la création de la mission en appuyant sur le quatrième bouton du joystick.
- Le diagramme 3.9 décrit les scénarios possibles du cas d'utilisation "Créer mission" par le mode « Map ». Après spécifier le mode de création, l'application lance l'interface correspondante qui affiche une mappe visualisant la localisation du robot P-Guard et l'espace environnant. L'agent commence la création de la mission en tapant sur la position désirée. Une boîte de dialogue apparaît pour spécifier la vitesse de passage de cette position. L'agent répète ces actions jusqu'à terminer tous les points de la mission puis confirme les actions effectuées.

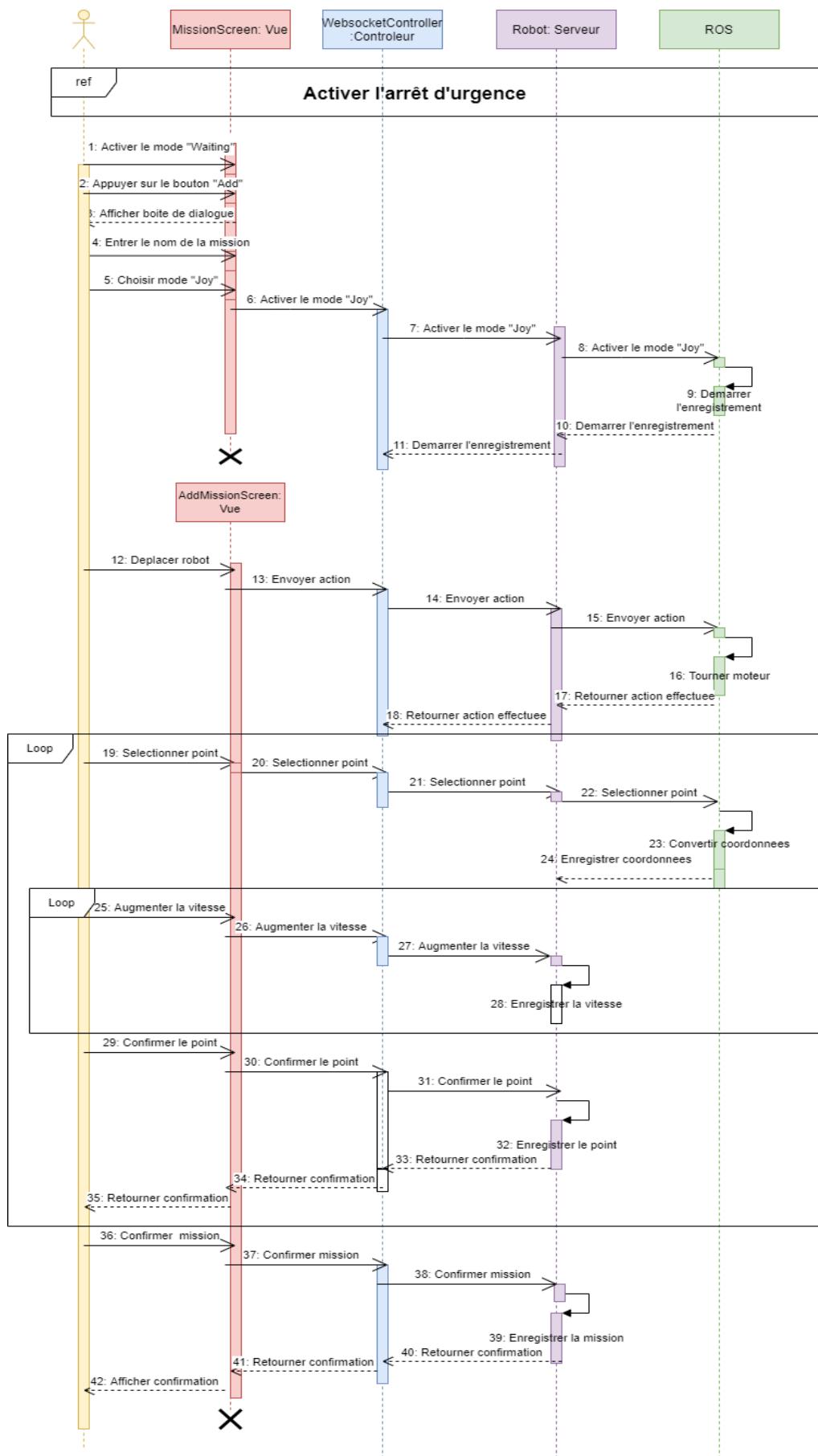


FIGURE 3.8 – Diagramme de séquences du cas d'utilisation "Créer mission par manette intégrée"

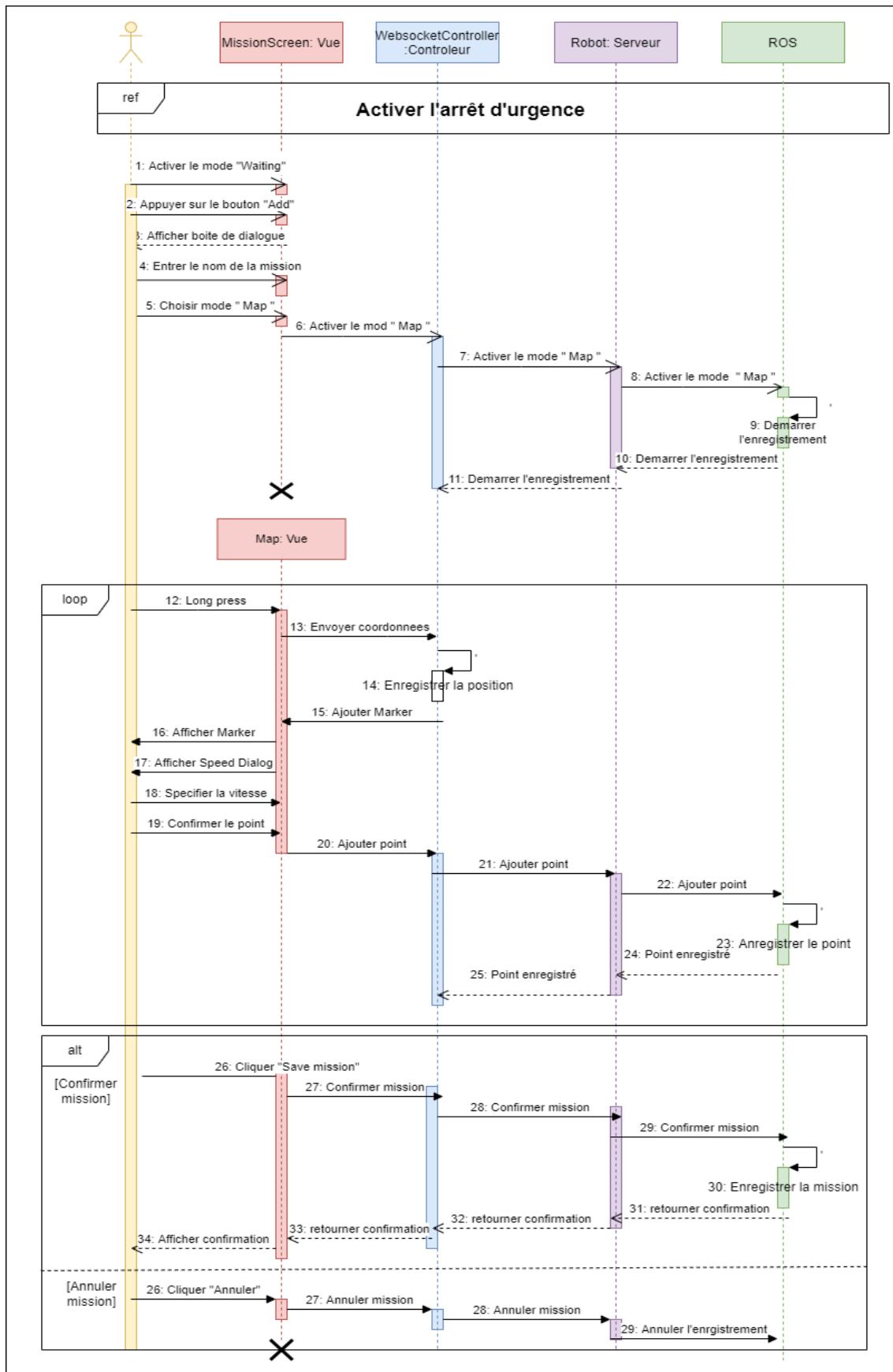


FIGURE 3.9 – Diagramme de séquences du cas d'utilisation "Créer mission par mappe"

3.4.1.6 Diagramme de séquences détaillé du cas d'utilisation : "Lancer un message d'interpellation"

Dans certains cas, ou l'agent remarque des personnes ou des objets douteux, il peut lancer des messages vocaux, dits d'interpellation, à travers le robot pour prévenir les personnes à l'extérieur et les forcer à quitter l'espace privé. Pour l'opérer, l'agent clique sur le bouton «Launch», l'application affiche la liste des messages d'interpellation enregistrés, par la suite il sélectionne un message précis pour le lancer. L'Id du message sera donc envoyé au robot qui reconnaît le message et le lance immédiatement.

Le diagramme 3.10 décrit les scénarios possibles du cas d'utilisation "Lancer un message d'interpellation".

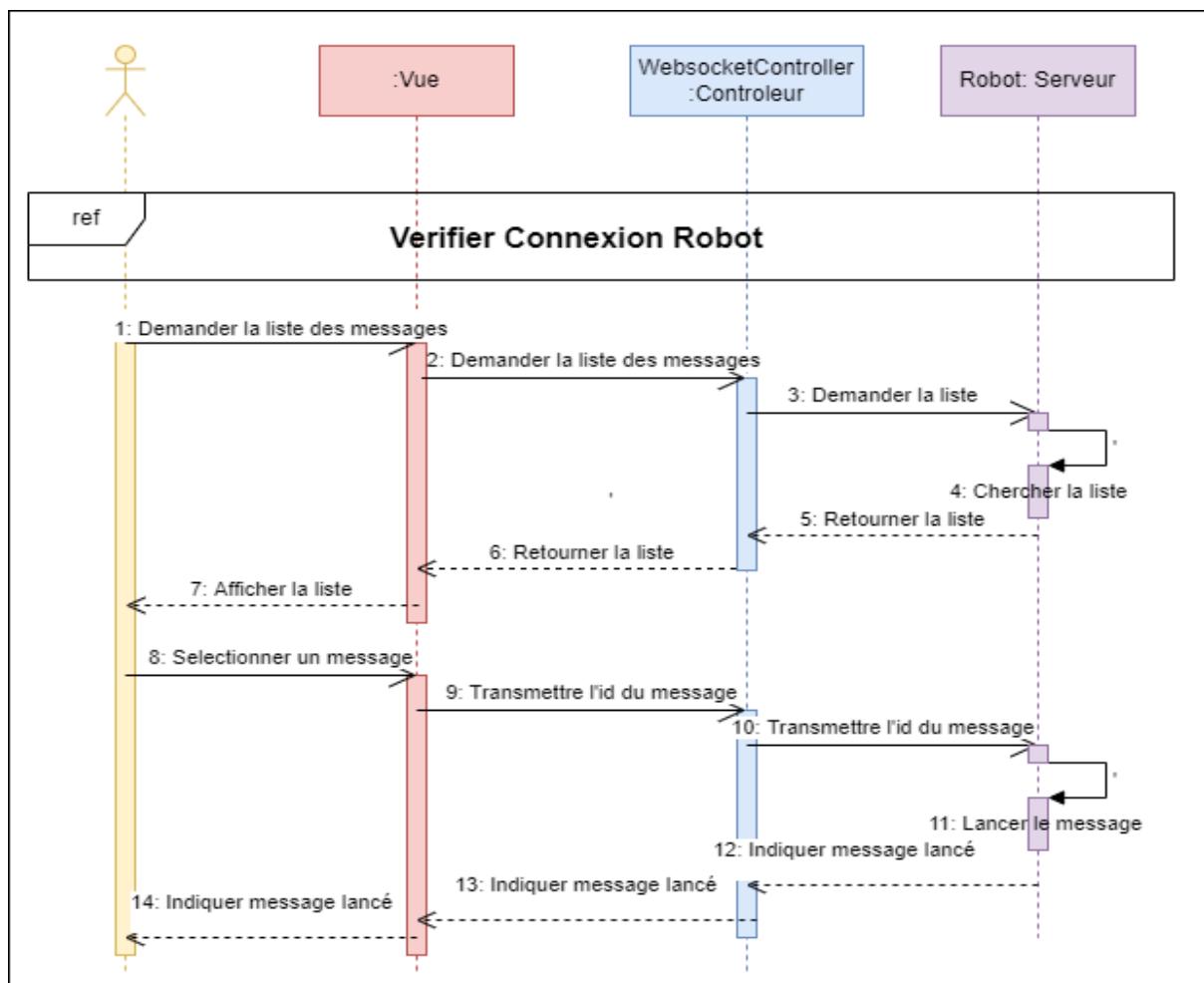


FIGURE 3.10 – Diagramme de séquences du cas d'utilisation "Lancer un message d'interpellation"

3.4.1.7 Diagramme de séquences détaillé du cas d'utilisation : "Lancer un scenario"

Pour lancer l'exécution d'un scenario spécifique, l'agent commence par activer l'arrêt d'urgence pour interrompre tout fonctionnement, puis charge le scenario dans le robot. L'application affiche un ensemble de marqueurs sur la mappe indiquant l'ensemble de positions à parcourir. Par la suite, l'agent désactive l'arrêt d'urgence et lance le scenario. L'application positionne le robot dans un point appelé « Home» et lance le fonctionnement automatique du robot.

Le diagramme 3.11 décrit les scénarios possibles du cas d'utilisation "Lancer un scenario".

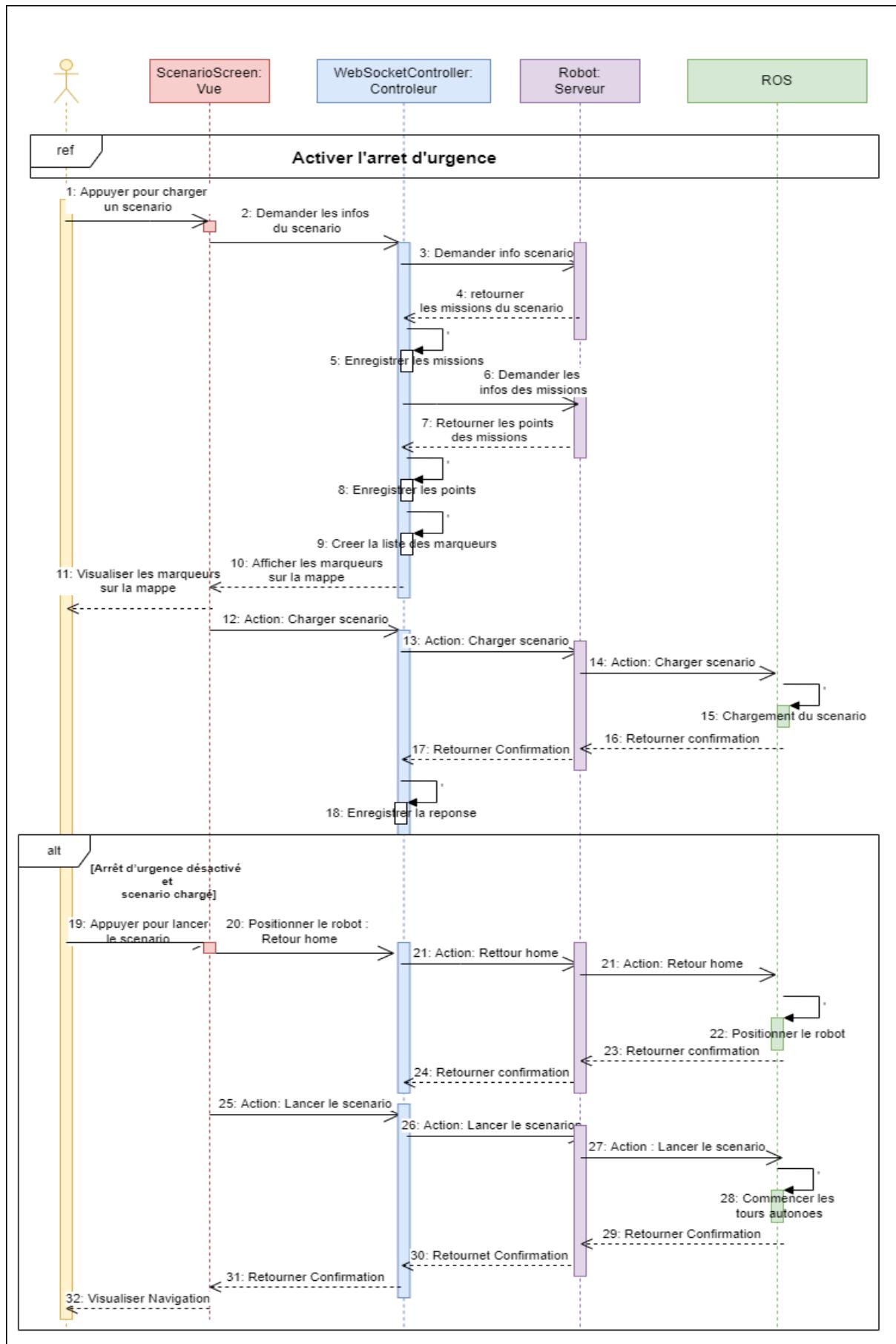


FIGURE 3.11 – Diagramme de séquences du cas d'utilisation "Lancer un scenario"

3.4.2 Diagramme d'états-transitions

Les diagrammes d'état-transition décrivent le comportement intérieur d'un objet en utilisant un automate d'états finis. Ils présentent les séquences possibles d'états et d'actions qu'une instance de classe peut traiter au cours de son cycle de vie en réaction à des événements discrets.

Lors de son fonctionnement et exécution des tâches, le robot P-Guard, passe par un ensemble d'états différents nécessaire à la réalisation des actions de l'agent. L'application mobile doit visualiser ces états et guider l'agent pour les respecter lors du contrôle du robot.

Nous allons modéliser les transitions de l'état du robot par le diagramme d'états-transitions.

3.4.2.1 Diagramme d'états-transitions du cas d'utilisation "Créer mission"

Lors de la création d'une nouvelle mission, comme nous avons indiqué précédemment, l'agent doit activer l'arrêt d'urgence du robot, passer en état Waiting, puis commencer l'enregistrement de la mission.

La figure 3.12 présente le diagramme d'états-transitions du cas d'utilisation "Créer mission".

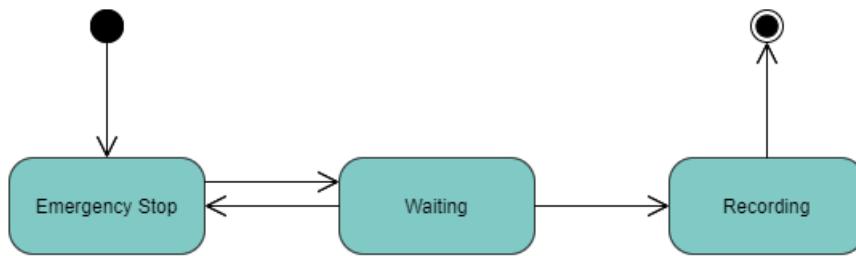


FIGURE 3.12 – Diagramme d'états-transitions du cas d'utilisation "Créer mission"

3.4.2.2 Diagramme d'états-transitions du cas d'utilisation "Lancer un scenario"

Le processus du lancement d'un scenario nécessite le changement de l'état du robot en arrêt d'urgence pour charger le scenario dans le robot. Par la suite, l'agent doit désactiver cet arrêt pour pouvoir lancer le fonctionnement autonome et l'exécution du scenario.

La figure 3.13 présente le diagramme d'états-transitions du cas d'utilisation "Lancer un scenario".

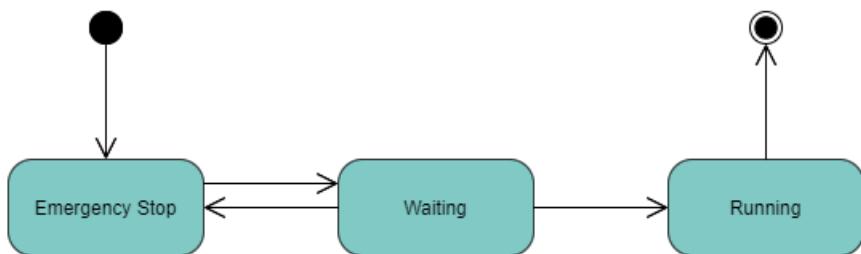


FIGURE 3.13 – Diagramme d'états-transitions du cas d'utilisation "Lancer un scenario"

Conclusion

Dans ce chapitre nous avons présenté l'une des plus importantes phases du processus du développement de notre projet. Dans ce cadre, nous avons présenté la conception détaillée afin de donner un aperçu approfondi de l'application. Nous avons précisé l'architecture logique et le patron utilisé et nous avons décrit aussi la vue dynamique du système à travers un ensemble de diagrammes de séquences et de diagrammes d'états-transitions.

Le chapitre suivant portera sur la phase réalisation du projet, qui sera illustrée par des imprimés écran des différentes interfaces et par une description de l'environnement de travail et des outils utilisés.

Chapitre 4

Réalisation

Plan

4.1	Environnement de développement	61
4.1.1	Environnement matériel	61
4.1.2	Environnement logiciel	61
4.2	Application développée	66
4.2.1	Les interfaces de bord :	66
4.2.2	Interface d'authentification :	67
4.2.3	Interface d'accueil :	67
4.2.4	Interface des scénarios	70
4.2.5	Interface des caméras	72
4.2.6	Interface des missions	73
4.2.7	Interface de navigation libre	75
4.2.8	Interface de paramètres	75
4.2.9	Interface de "connexion perdue"	76
4.3	Défis soulevés	77
4.4	Validation des besoins non fonctionnels	78
4.5	Tests et validation	78

Introduction

Après avoir défini toutes les exigences nécessaires pour la réalisation du système et la satisfaction des objectifs, nous pouvons commencer à construire notre solution logicielle qui s'appelle "P-Guard" comme le robot lui-même.

Nous présentons dans ce dernier chapitre, l'environnement de développement de cette solution, les choix technologiques, le langage adopté, les défis soulevés ainsi que la réalisation.

4.1 Environnement de développement

Cette section présente l'environnement matériel mis à la disposition du projet et l'environnement logiciel utilisé pour le développement.

4.1.1 Environnement matériel

Au cours des différentes étapes de notre projet, à savoir la documentation, et l'implémentation du code, nous avons disposé de :

- Un ordinateur personnel dont la configuration est la suivante :
 - **Marque** : Acer.
 - **Processeur** : Intel Core i5-7300HQ.
 - **RAM** : 8 Go.
 - **Disque dur** : 128 Go SSD.
 - **Système d'exploitation** : Windows 10.
- Un Smartphone ayant les caractéristiques suivantes :
 - **Marque** : Oppo A9.
 - **Processeur** : Snapdragon 665.
 - **RAM** : 8 Go.
 - **Version du système** : Android 9.0.
- Une tablette de test ayant les caractéristiques suivantes :
 - **Marque** : SAMSUNG Galaxy TAB A8.
 - **Processeur** : Octa-Core.
 - **RAM** : 4 Go .
 - **Version du système** : Android 11.

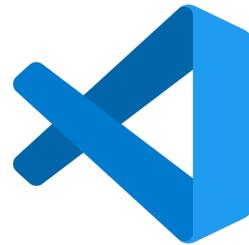
4.1.2 Environnement logiciel

Tout au long de la phase de développement, nous avons utilisé les outils logiciels et les langages de développement suivants :

4.1.2.1 Outils logiciels

Durant l'implémentation de notre solution nous avons utilisé les logiciels suivants :

- **Visual Studio Code** Visual Studio Code (VSC par la suite) est un éditeur de code open-source, gratuit et multi-plateforme, développé par Microsoft, VSC est développé avec Electron. Principalement conçu pour le développement d'application avec JavaScript, TypeScript et dart, l'éditeur peut s'adapter à d'autres types de langages grâce à un système d'extension bien fourni[16].



- **Android Studio** : Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et il utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, macOS, Chrome OS et Linux [17].



- **Mapbox** : Mapbox est une plate-forme de données de localisation pour les applications mobiles et Web. Elle offre des services de cartographie d'extérieur. Les cartes sont simples à intégrer dans les applications. [14].



- **Postman** : Postman est une application qui permet de lancer des appels d'API et de les tester sur le navigateur web. Il dispose d'un environnement graphique complet et clair pour gérer l'ensemble des interactions avec les API[18].



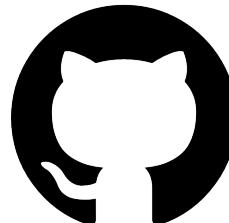
- **Lottie** : Lottie est un format ouvert d'animation vectorielle basé sur JSON créé à l'origine par la branche de design de Airbnb avec son plugin d'export BodyMovin, pour les animations produites sous After Effects. Il est à l'origine produit pour pouvoir être utilisé avec Android, iOS et React Native[19].



- **Swagger** : Swagger est un logiciel open source qui offre des outils permettant de générer la documentation pour son API Web. Il offre également une interface permettant d'explorer et de tester les différentes méthodes offertes par le service[20].



- **Github** : GitHub est une plate-forme d'hébergement de code pour le contrôle de version et la collaboration. Il permet de travailler ensemble avec d'autres sur des projets logiciels[21].



- **Slack** : Slack est une plateforme de communication collaborative propriétaire ainsi qu'un logiciel de gestion de projets. C'est une application de messagerie utilisée par les entreprises qui connecte les gens aux informations dont ils ont besoin. En réunissant les gens pour travailler comme une équipe unifiée, Slack transforme la façon dont les organisations communiquent[22].



- **Visual Paradigm** : Visual Paradigm est un logiciel de dessin permettant aux programmeurs de mettre en place des diagrammes UML ainsi que la création rapide des dessins techniques et commerciaux, des schémas réseaux, Charts, des circuits électroniques, des maquettages d'interfaces et des organigrammes en ligne gratuitement[23].



- **Draw.io** : Draw.io est un logiciel de dessin graphique multiplateforme gratuit et open source développé en HTML5 et JavaScript. Son interface peut être utilisée pour créer des diagrammes tels que des organigrammes, des structures filaires, des diagrammes UML, des organigrammes et des diagrammes de réseau[24].



- **Adobe XD** : Adobe XD est présenté comme l'outil idéal pour faciliter la conception d'une interface utilisateur ergonomique et réaliser une déclinaison flexible sur les différents terminaux mobiles : tablettes, iphone, Smartphone, etc[25].



4.1.2.2 Technologies utilisées

- **Flutter** : Flutter est un framework de développement d'applications multiplateforme, conçu par Google, dont la première version a été publiée sous forme de projet open source à la fin de l'année 2018.

Nous avons choisi Flutter pour plusieurs raisons :

- **Large soutien communautaire** : Flutter est développé par le géant de la technologie - Google. Ainsi, Google soutient constamment les développeurs de Flutter avec des mises à jour fréquentes et des correctifs de problèmes. De plus, lors de l'utilisation de Flutter, nous pouvons constater une augmentation significative des performances par rapport à des technologies similaires. La communauté de développement d'applications Flutter n'a cessé de croître.
Depuis sa première version, Flutter compte désormais 81 200 étoiles sur Github. En raison de la communauté amicale des développeurs.
- **Similaire au développement natives d'applications** : Les rendus logiciels de Flutter utilisent un moteur graphique interne appelé Skia. Ce logiciel permet un développement rapide et bien optimisé par rapport à la plupart des autres frameworks d'applications mobiles.
Une application Flutter sera indifférenciable d'une application native. Parce que Flutter ne s'appuie sur aucune représentation ou interprétation de code intermédiaire.
- **Écriture de code plus rapide** : En générale, les développeurs iOS et Android doivent écrire du code, puis attendre qu'il soit compilé et chargé sur l'appareil avant de voir les modifications. Mais, avec le rechargement à chaud de Flutter, ils peuvent vérifier les effets immédiatement ou sans délai.
- **Au-delà des applications mobiles** : En 2019, Google a annoncé Flutter Octopus, qui permet le débogage sur plusieurs plates-formes simultanément. À partir d'une seule base de code, les développeurs peuvent créer une solution qui fonctionnera non seulement sur mobile, sur le Web, mais également sur Mac.
- **Rapport coût-efficacité** : L'optimisation des coûts est un objectif essentiel pour toute entreprise. Lors du développement d'applications natives, les propriétaires d'entreprise doivent payer deux fois : pour la version iOS, et pour la version Android. Ils doivent partager le budget avec deux équipes distinctes. Avec la fonctionnalité Flutter, ils n'ont besoin que d'une seule équipe de développement Flutter pour développer et gérer les applications [26].



- **Dart** : Dart est un langage de programmation optimisé pour les applications sur plusieurs plates-formes. Il est développé par Google et utilisé pour créer des applications mobiles,

de bureau, et Web [27].



- **Latex** : Latex est un système de composition de haute qualité ; il comprend des fonctionnalités designé pour la production de documentation technique et scientifique [28].

LATEX

4.2 Application développée

Dans cette section, nous énumérons les différentes fonctionnalités de notre application en présentant les interfaces graphiques réalisées :

4.2.1 Les interfaces de bord :

Les interfaces de bord présentées dans la figure 4.1 apparaissent une seule fois au cours du processus d'utilisation de l'application (lors de la toute première utilisation) pour la présenter et pour introduire le robot de sécurité ainsi que certaines informations.

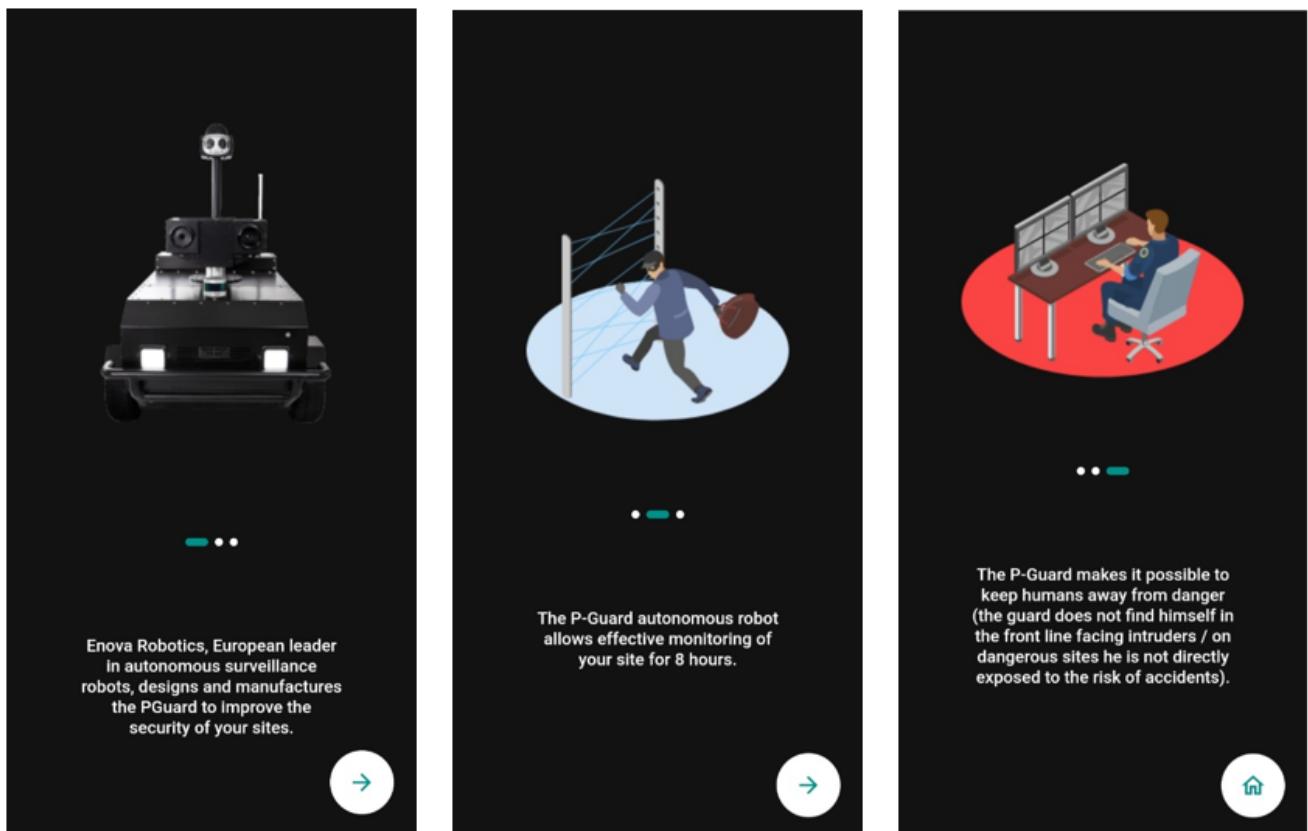


FIGURE 4.1 – Interfaces de bord

4.2.2 Interface d'authentification :

L'utilisateur doit tout d'abord s'authentifier afin de profiter des fonctionnalités de l'application. Pour ce faire, il doit saisir son identifiant (son email) et son mot de passe.

La figure 4.2 montre l'interface d'authentification avec la boîte d'alerte qui apparaît dans le cas de coordonnées incorrectes.

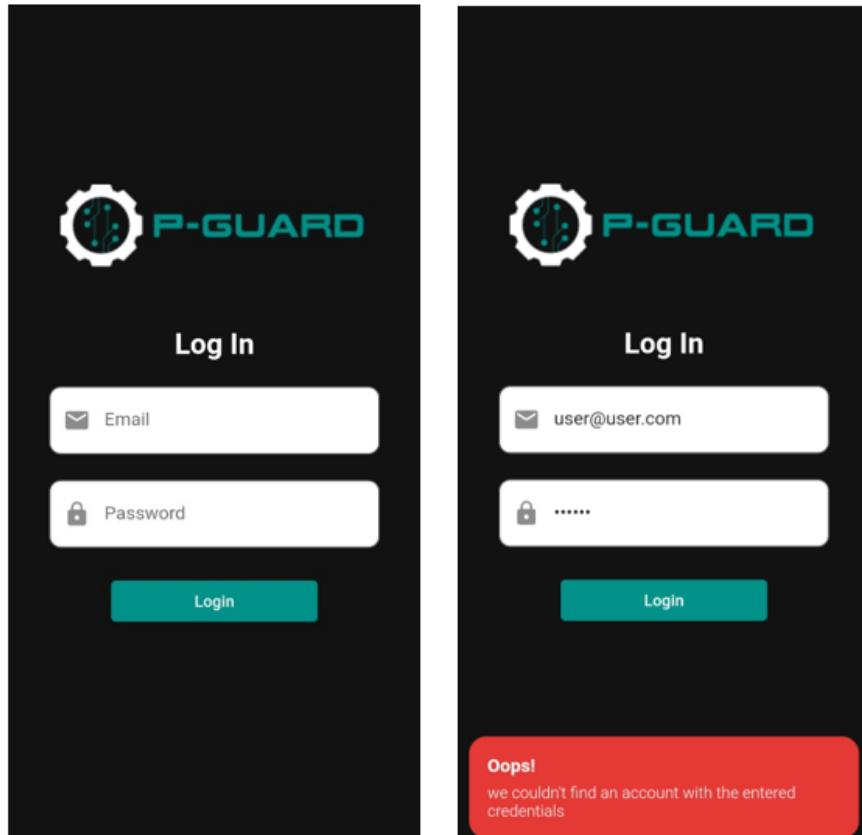


FIGURE 4.2 – Interfaces d'authentification

4.2.3 Interface d'accueil :

La figure 4.3 représente l'interface d'accueil de l'application qui se compose de trois parties principales, une barre de tâches, un menu de navigation et un panneau principal.

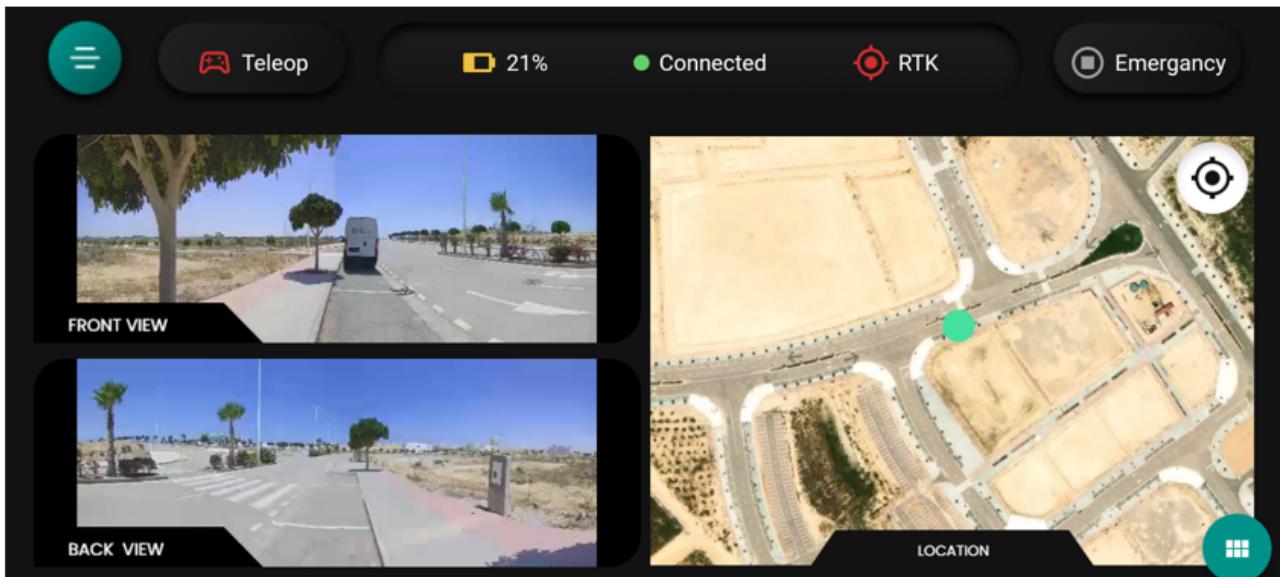


FIGURE 4.3 – Interfaces d'accueil

4.2.3.1 Barre de tâches

La barre de tâches représente un ensemble d'informations concernant l'état courant du robot tel que le niveau de sa batterie, l'état de correction GPS : RTK (si elle est activée ou pas) et l'état de connexion du robot.

La barre contient aussi deux boutons. Le premier à mentionner est le bouton de l'arrêt d'urgence. L'agent de sécurité active l'emergency stop du robot dans plusieurs situations. Parmi eux, la création d'une nouvelle mission et le chargement d'un scenario spécifique dans le robot.

Le deuxième bouton est le « Teleop » qui consiste à donner la main à la manette physique pour contrôler la navigation du robot.

La figure 4.4 représente la barre de tâches de l'application.

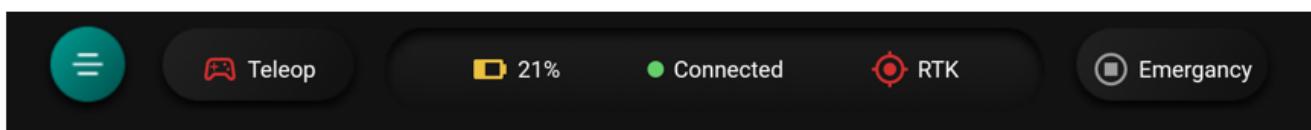


FIGURE 4.4 – Barre de tâches

4.2.3.2 Menu de navigation

La figure 4.5 représente le menu de navigation entre les interfaces de l'application.

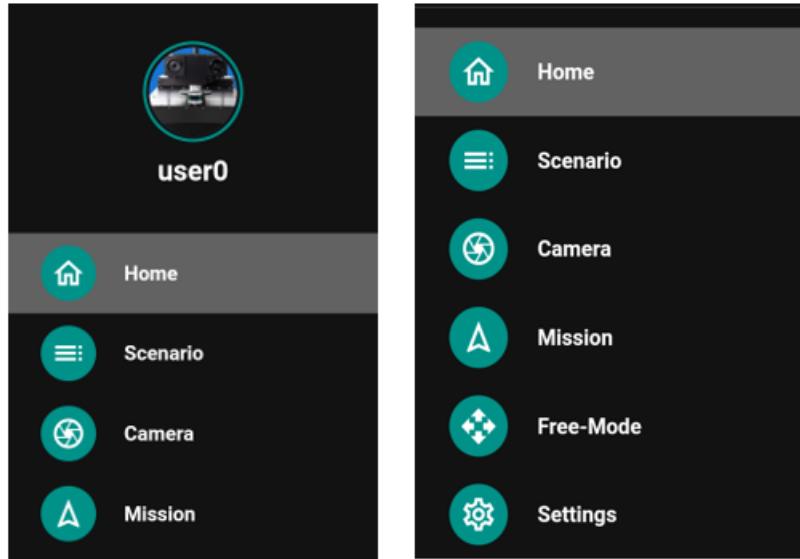


FIGURE 4.5 – menu de navigation

4.2.3.3 Panneau principal de l'interface d'accueil

L'interface d'accueil affiche les flux vidéo transmis par le robot en temps réel, ainsi que sa position sur la mappe. Elle offre la possibilité de contrôler l'état du robot (allumer la lampe), ou de lancer un message d'interpellation à travers le haut-parleur du robot dans le cas de détection d'un intrus à partir des vidéos visualisés.

La figure 4.6 représente le panneau principal de l'interface d'accueil.



FIGURE 4.6 – Panneau principal de l'interface d'accueil

Pour lancer un message d'interpellation, l'agent dispose d'une liste de sons pré-enregistrés qui peuvent être émis dans différents cas d'urgence.

La figure 4.7 représente la liste des messages d'interpellation.

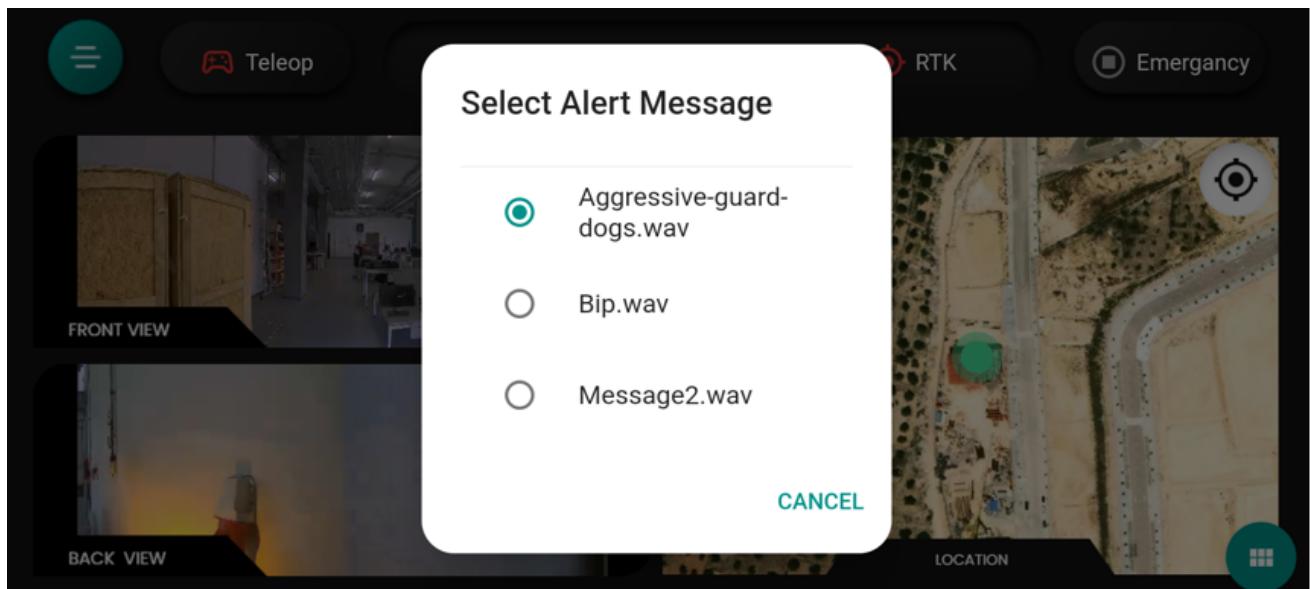


FIGURE 4.7 – Liste des messages d’interpellation

4.2.4 Interface des scénarios

La figure 4.8 montre l'espace principal du gestion des scénarios.

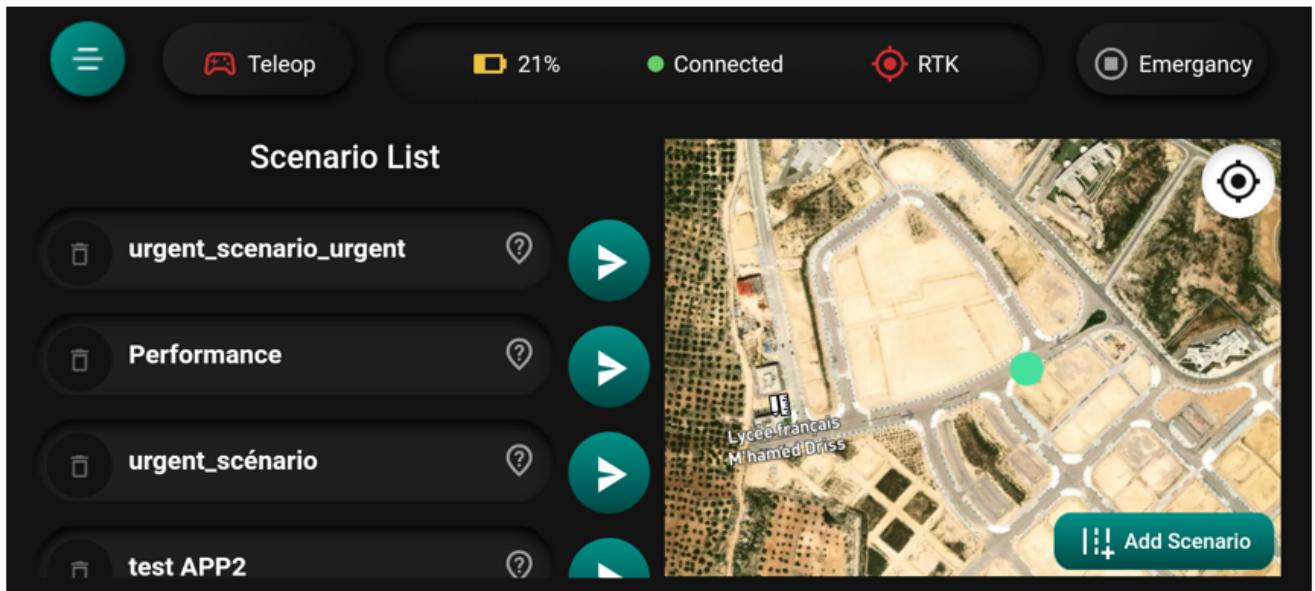


FIGURE 4.8 – Interface des scénarios

L’interface présente l’ensemble de scenarios enregistrés dans le robot. L’agent de sécurité peut lancer l’un de ces scenarios pour qu’il soit exécuté en autonomie par le robot. Il y a une étape importante qui précède le lancement, qui est le chargement du scenario correspondant dans le robot.

Après activer l’arrêt d’urgence l’agent peut charger le scenario en cliquant sur l’icône de localisation qui sera activé.

La figure 4.9 montre la phase de chargement d'un scenario dans le robot qui fait afficher sur la mappe l'ensemble de point à parcourir par le robot pour effectuer ses rondes de surveillance.

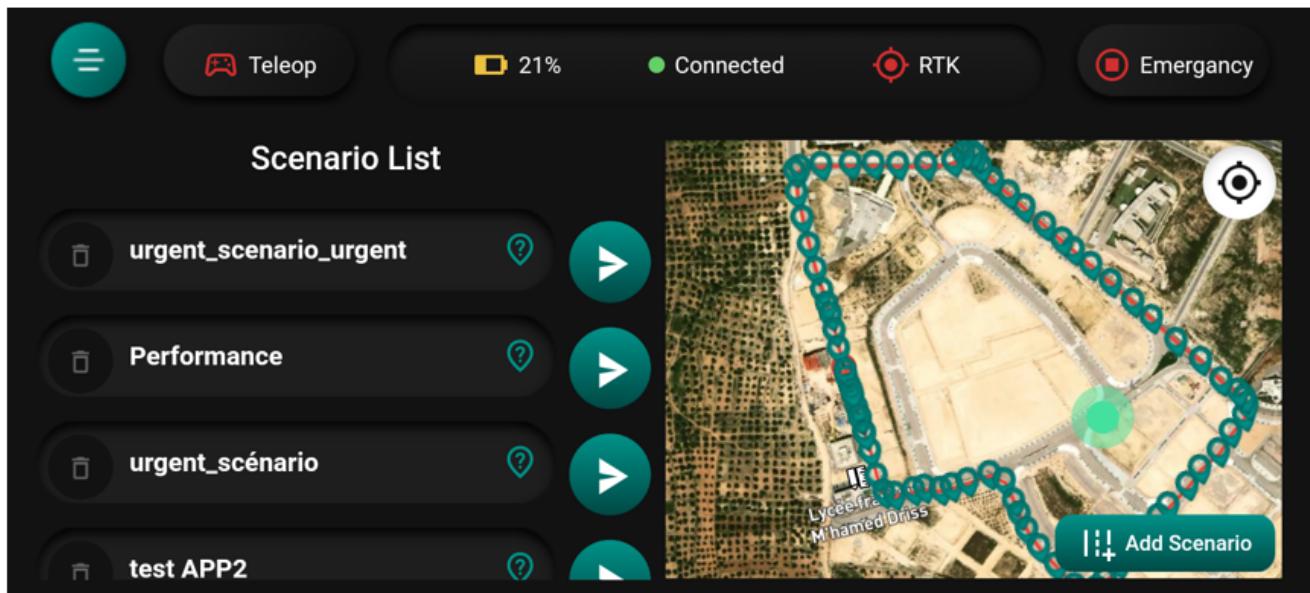


FIGURE 4.9 – Interface illustrant la phase du chargement d'un scénario

L'agent de sécurité peut ajouter un nouveau scénario en spécifiant ses options, la liste de missions qu'il contient, et l'arrêt après chaque mission exécutée.

La figure 4.10 représente les étapes suivies pour créer un nouveau scénario.

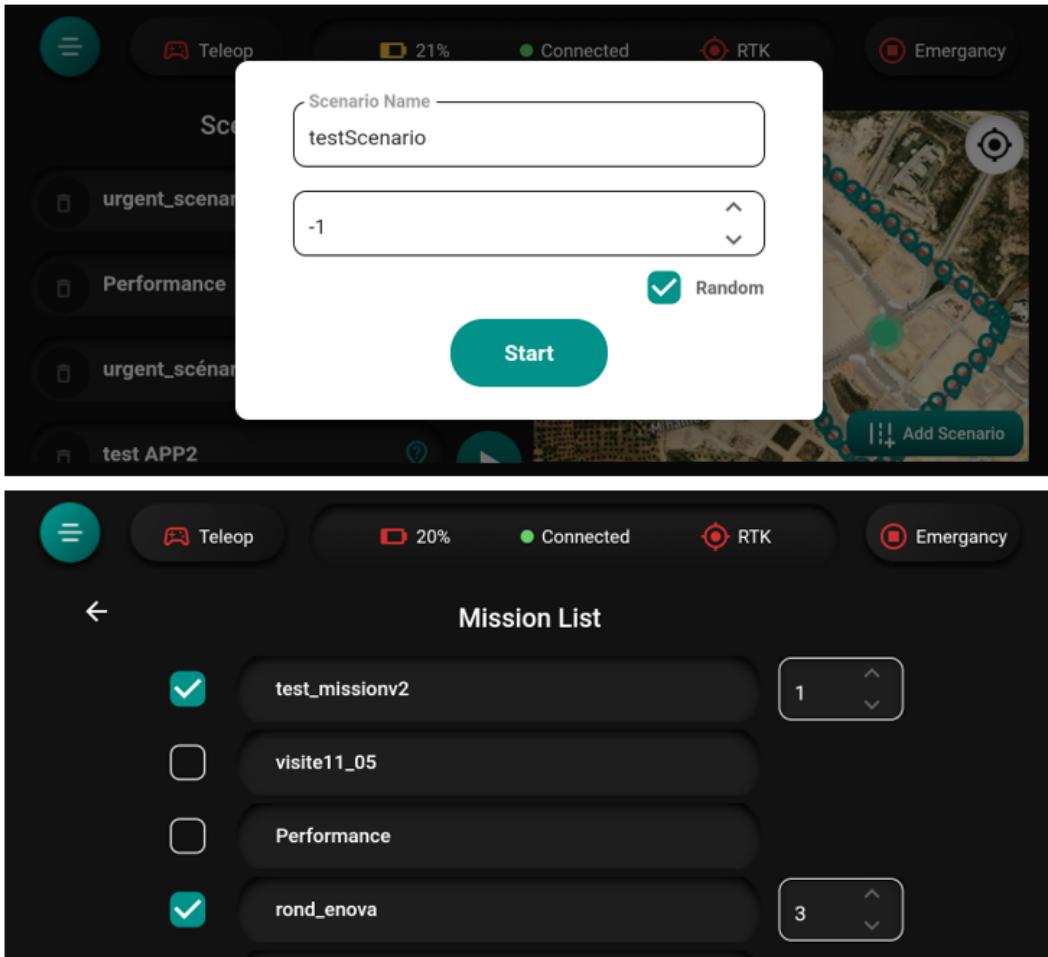


FIGURE 4.10 – Étapes pour créer un nouveau scénario

4.2.5 Interface des caméras

D'autre part, l'application offre un ensemble de fonctionnalités permettant à l'agent de gérer les caméras du robot. Ce dernier possède une caméra optique et une autre thermique. Les flux sont affichés dans cet espace du caméra. L'agent peut zoomer et dézoomer pour une vue plus précise et peut faire tourner la tourelle (360 °) pour surveiller tous les cotes.

La figure 4.11 représente l'espace de gestion du caméra.

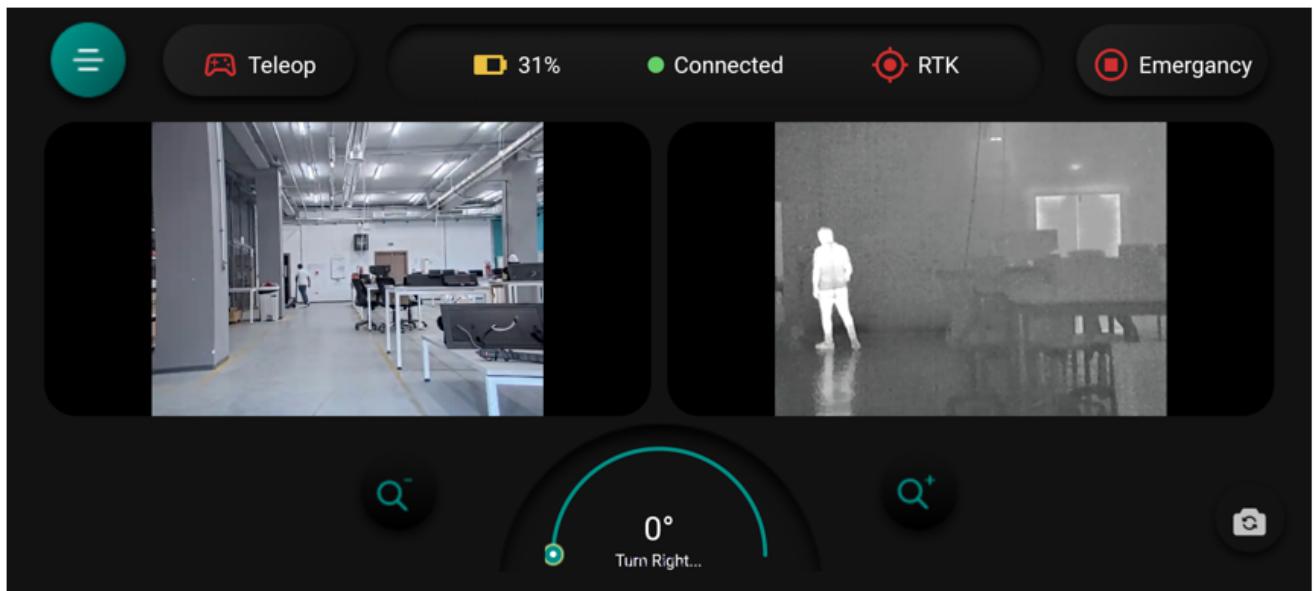


FIGURE 4.11 – Interface des caméras

4.2.6 Interface des missions

La figure 4.12 représente la liste des missions enregistrées dans le robot.

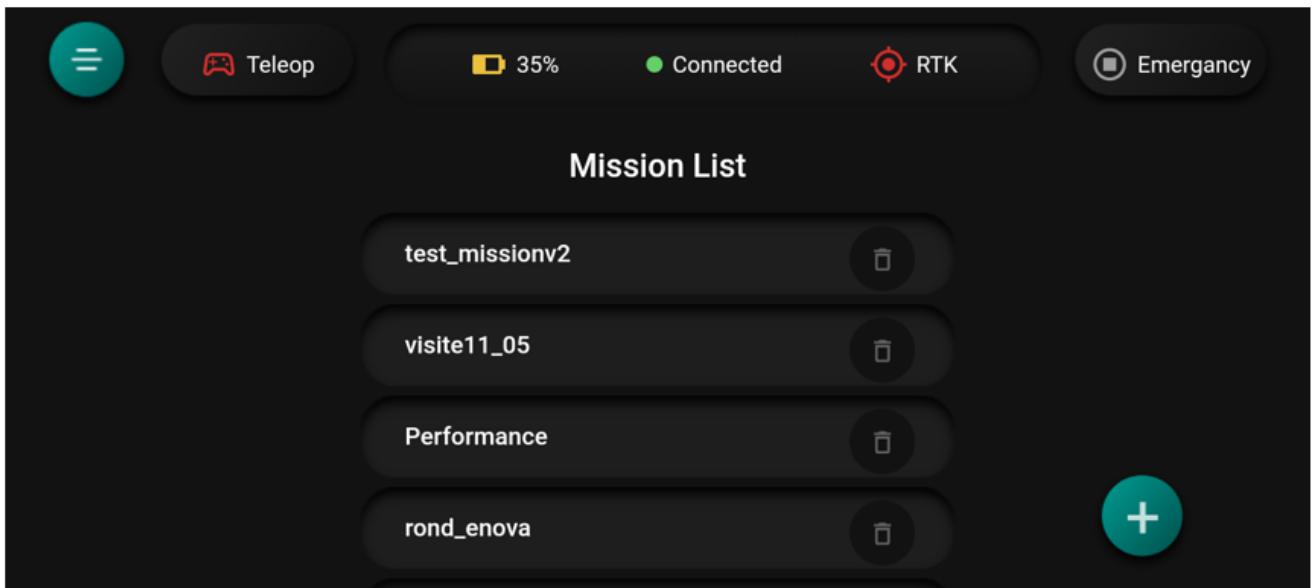


FIGURE 4.12 – Interface des mission

En cliquant sur le bouton « + » pour ajouter une mission, l'application propose les deux modes de création, le mode « Map » et le mode « Joystick ».

La figure 4.13 représente la boîte de dialogue affichée pour choisir le mode de création de la mission.

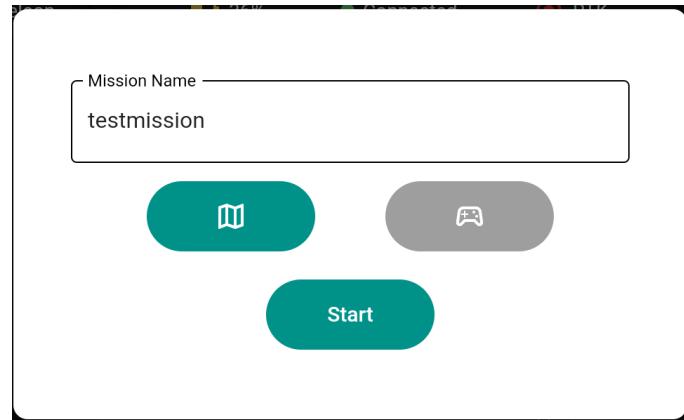


FIGURE 4.13 – Les modes de création d'une mission

Le premier mode consiste à spécifier l'ensemble de points de la ronde en appuyant sur la mappe et en identifiant la vitesse de passage par chaque point dans la boîte de dialogue qui apparaît. La figure 4.14 représente la première méthode de création d'une mission.

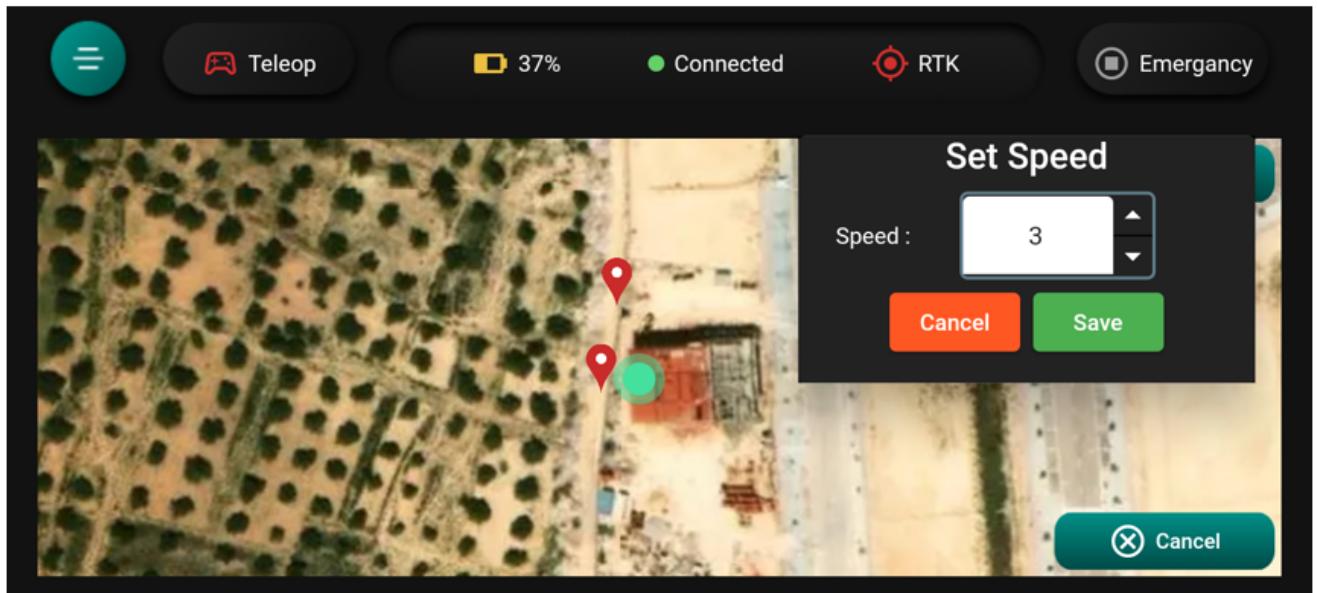


FIGURE 4.14 – Création de missions par le mode "Map"

Le deuxième mode consiste à faire naviguer le robot par la manette intégrée jusqu'à atteindre le point désiré puis spécifier la vitesse de passage à l'aide des boutons du joystick. L'agent doit répéter cette action jusqu'à terminer la liste de points.

La figure 4.15 représente la deuxième méthode de création d'une mission.

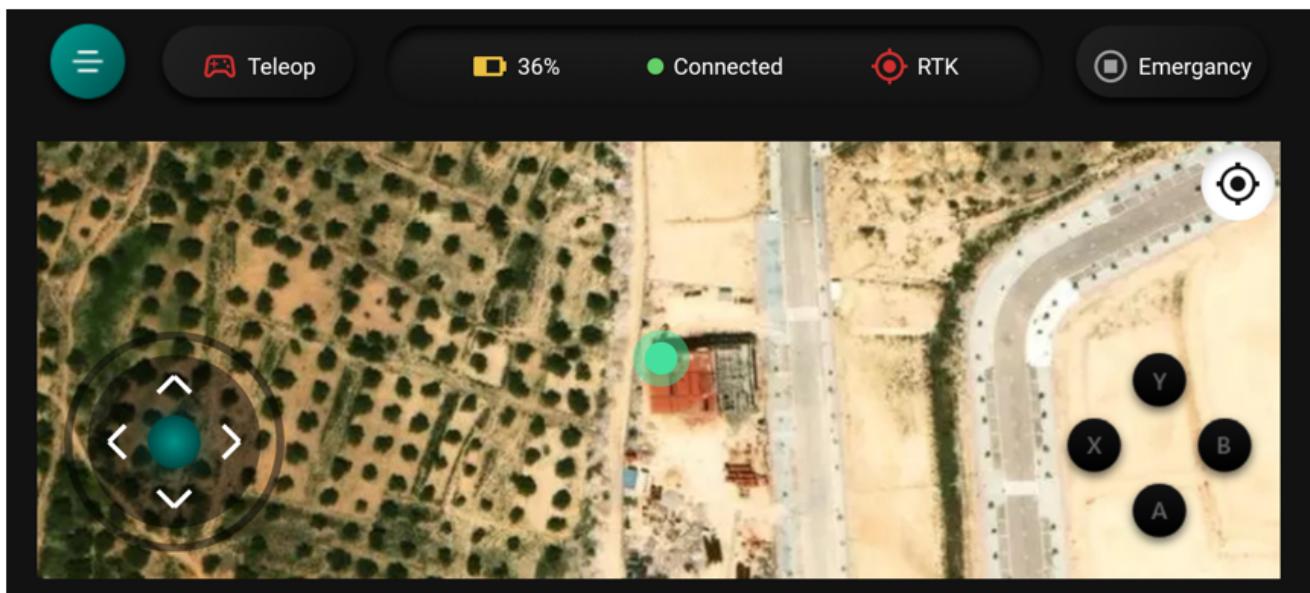


FIGURE 4.15 – Crédit de missions par le mode "Joystick"

4.2.7 Interface de navigation libre

Pour, simplement, faire déplacer le robot d'un point à un autre en visualisant la caméra frontale, l'agent dispose d'un espace dit « Free Mode » présenté par la figure 4.16.

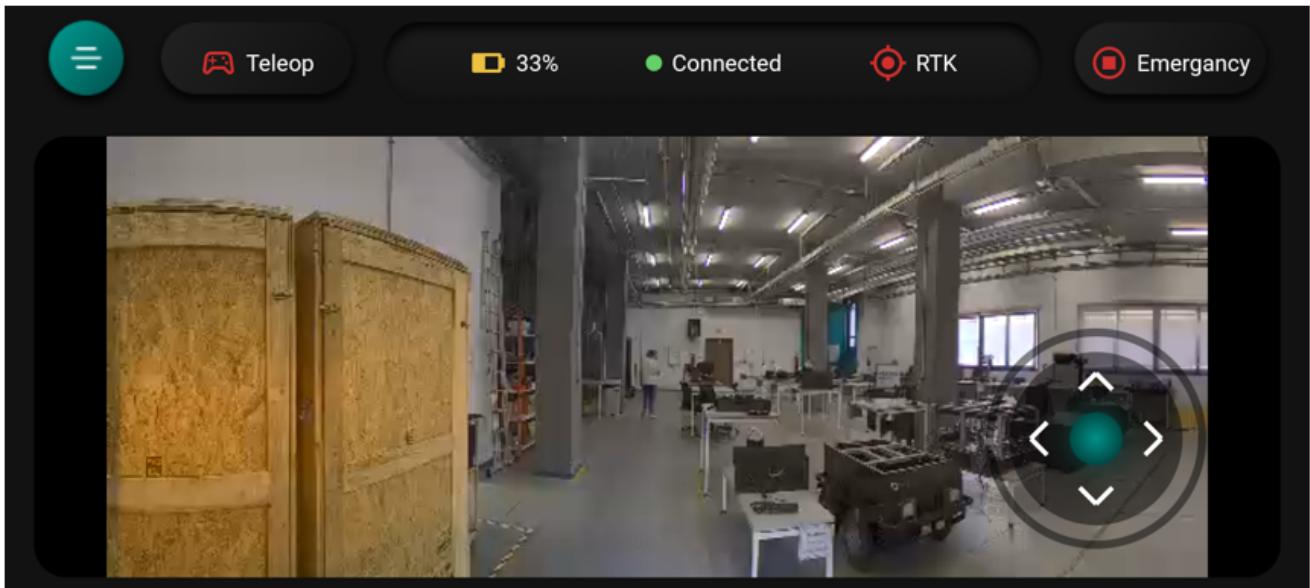


FIGURE 4.16 – Interface de navigation libre du robot

4.2.8 Interface de paramètres

L'interface de paramètres offre un ensemble de fonctionnalités importantes. En fait, l'agent peut consulter l'historique d'états du robot, les arrêts d'urgence déclenchés ainsi que les missions enregistrées.

En plus, il peut activer la réception d'alertes pour se notifier dans les cas d'urgence comme un

niveau bas de la batterie. Dans ce cas, l'agent peut interrompre le fonctionnement du robot et le renvoyer vers sa base de recharge.

La figure 4.17 représente l'interface des paramètres de l'application.

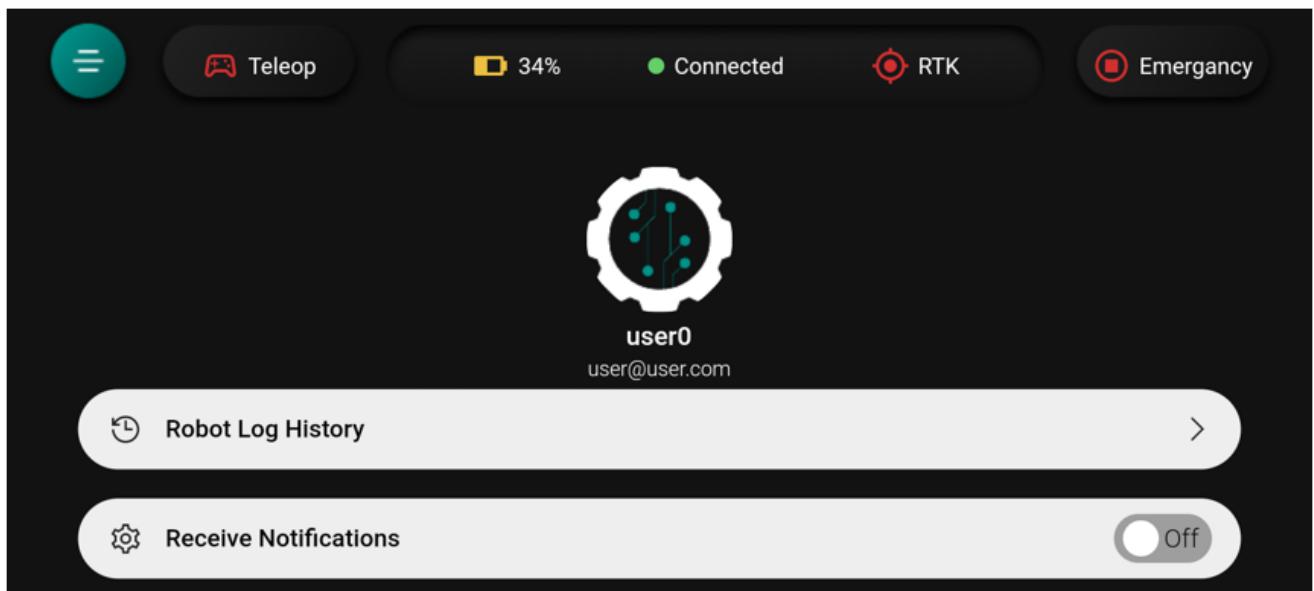


FIGURE 4.17 – Interface de paramètres

La figure 4.18 représente un exemple de notification affichée à l'agent de sécurité.

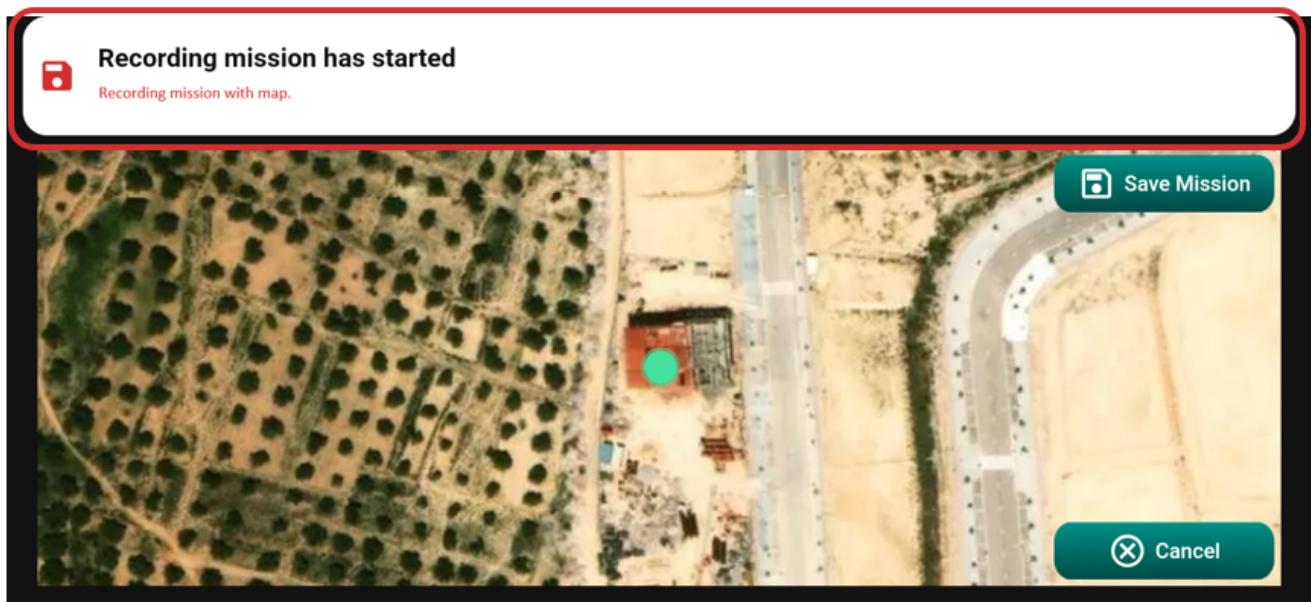


FIGURE 4.18 – Exemple de notification reçue

4.2.9 Interface de "connexion perdue"

Lors de la détection d'une perte de connexion entre l'application et le robot, une interface spécifique s'affiche, présentée dans la figure 4.19, indiquant à l'utilisateur l'erreur déclenché. Le système essaye immédiatement de reconnecter au robot pour récupérer de nouveau les données

transmises.

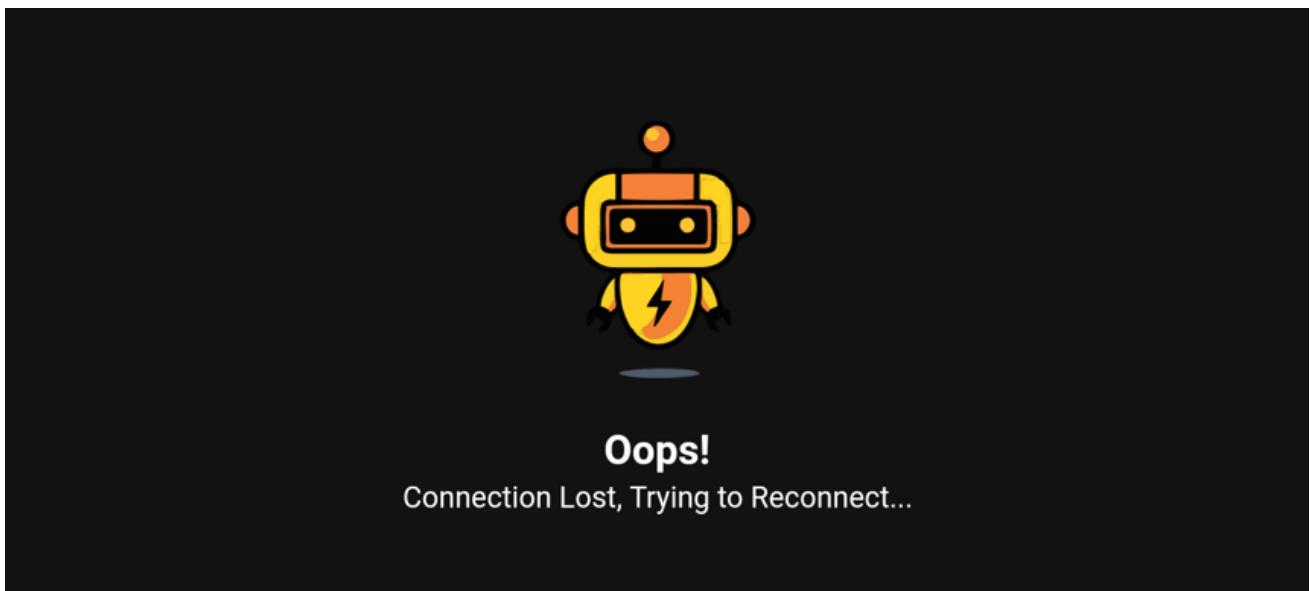


FIGURE 4.19 – Interface de perte de connexion

4.3 Défis soulevés

Lors du développement de notre application, tout au long de la durée du stage, plusieurs contraintes nous ont considérablement retardés :

- Pour des raisons de sécurité et de prévention de problèmes, nous n'avons pas eu accès au système réel du robot qu'après deux mois de travail, au début de la période de stage, l'application s'est connectée à un système alternatif qui simule le fonctionnement du robot. Le retour de ce dernier système, est le plus souvent des valeurs nulles et inexactes. Par conséquent, lors de la communication avec le robot, on a été obligé d'adapter de nouveau la plupart des fonctionnalités développées ce qui demandait un effort et un temps considérable.
- Le robot de test est utilisé par plusieurs équipes de spécialités différentes, (comme l'informatique, le mécanique, et l'intelligence artificielle) ce qui nous oblige, en cas de besoins, d'interrompre nos tests et suspendre les tâches pendant quelques heures.
- Pour ce qui concerne la technologie utilisée, un des défis rencontrés est relié à la WebSocket utilisé. En effet, la forme la plus courante du Stream de la WebSocket ne peut être écoutée qu'une seule fois à la fois. Si nous essayons d'ajouter plusieurs écouteurs, qui est le cas de notre application contenant plusieurs pages, cela lèvera toujours une exception.
Pour éviter cette erreur, nous avons utilisé un contrôleur spécifique dit StreamController pour convertir le flux en diffusion à l'aide de la méthode Broadcast() qu'il offre.
- Le système intégré dans le robot contient des modules qui supportent des clients spécifiques. Prenant l'exemple du module responsable de la transmission des flux vidéo qui est

développé de manière à pouvoir communiquer avec des clients JavaScript mais pas Dart ou autres, ce qui pose un problème d'incompatibilité.

4.4 Validation des besoins non fonctionnels

Dans un projet informatique, l'insatisfaction des exigences non fonctionnelles peut causer des problèmes lors de la mise en production, en situation réelle. Au mieux, ces défauts sont corrigés lors des tests effectués sur l'application.

Les besoins non fonctionnels identifiés lors de la phase de l'étude préliminaire :

- Pour créer une application facile à comprendre et agréable à l'œil, nous avons passé par une étape de conception des interfaces par le logiciel Adobe XD et nous avons effectué les changements nécessaires tout au long du périodes de stage pour assurer une bonne expérience utilisateur.

Le choix du mode sombre pour notre application est expliqué par plusieurs raisons, premièrement, comme les agents de sécurité vont passer un temps considérable face aux interfaces pour surveiller les flux vidéo transmis nous avons choisi de favoriser le dark mode pour réduire la fatigue des yeux. De plus ça va réduire la consommation d'énergie et donc économiser la batterie.

- Pour assurer la fiabilité et la disponibilité des services offerts, nous avons implémenté une méthode qui est responsable de se reconnecter au robot dès la détection d'un problème de connexion.
- L'architecture logique utilisée pour le développement de notre application est basée sur la séparation du logique métier et des composants affichés à l'utilisateur. L'application est composée d'un ensemble de contrôleurs, chacun responsable de la réalisation d'une fonctionnalité spécifique, ce qui facilite tout changement ou ajout de nouveaux modules.
- La technologie utilisée qui est Flutter, est un SDK permettant le développement d'applications mobiles multiplateformes. À l'aide d'une seule base de code indépendante de la plate-forme, Flutter permet de créer des applications hautes performances avec des interfaces utilisateur fonctionnelles pour différents systèmes d'exploitation : principalement Android et IOS. Les dimensions et les tailles des composants graphiques ne sont pas constantes mais plutôt relatives à la taille de l'écran du matériel utilisé.
- Pour assurer l'intégrité des données de l'application, la création des comptes utilisateur n'est pas accessible à tout le monde, les agents utilisent plutôt des comptes préconfigurés par les techniciens selon la demande du client.

4.5 Tests et validation

Après le cycle du développement, une phase de test s'avère nécessaire. Des scénarios bien déterminés ont été mis en place pour tester les différentes fonctionnalités de la plateforme et sa

compatibilité avec le robot et les besoins définis précédemment. La phase de test est effectuée au sein des locaux de Enova Robotics. En outre, une scène de test a été préparée où le robot est chargé de différents scénarios. Durant les tests, quelques anomalies ont fait surface. Après diagnostique et suivi ces anomalies ont été résolues et corrigées.

Conclusion

Dans ce chapitre, nous avons détaillé la spécification technique de notre logiciel à travers la présentation des environnements matériels et logiciels. Nous avons aussi décrit les fonctionnalités réalisées de notre application, illustrées par des imprimés écran, les défis soulevés lors du développement de l'application, ainsi que la validation des besoins non fonctionnels fixés dans la phase de l'étude préliminaire.

Nous clôturons notre rapport par la conclusion générale et les perspectives de notre projet.

Conclusion et perspectives

Ce projet s'inscrit dans le cadre de l'obtention du diplôme national d'ingénieur en informatique de l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse, réalisé au sein de la société Enova Robotics. Il a porté sur la conception et le développement d'une application mobile permettant le pilotage et le contrôle du robot de sécurité, le P-Guard, et jouant le rôle de l'outil principal de commande à distance du robot.

Le robot ne fonctionne proprement qu'après l'exécution d'un ensemble d'ordres lancés par l'agent de sécurité à travers le système que nous avons développé.

L'application traite de la manière la plus efficace et robuste l'ensemble de données transmises par le robot. Elle permet une visualisation claire du flux vidéo des quatre cameras du robot (optiques et thermiques) ainsi qu'un suivi en temps réel de sa localisation. L'agent de sécurité peut contrôler directement l'état du robot, gérer les scénarios et les missions qu'il doit exécuter, recevoir des alertes et lancer des messages d'interpellation en cas de besoin. C'est grâce à ces actions qu'on assure un renforcement de la sécurité des sites surveillés.

Pour réaliser ce travail, nous avons d'abord établi une étude préliminaire du projet pour spécifier ses besoins fonctionnels et techniques, qui ont été utilisés par la suite pour établir la conception détaillée du système ainsi que son implémentation.

Le stage a été une excellente occasion pour sortir du cadre théorique et atteindre de nouveaux objectifs. Il convient de souligner un point important qui se manifeste dans le fait que cette expérience m'a permis d'améliorer mes connaissances techniques, de comprendre et apprendre à maîtriser une variété de technologies que nous avons utilisé au cours de ce travail, et surtout d'acquérir de nouvelles connaissances dans le domaine de la robotique mobile. De plus, il m'a fallu apprendre à bien gérer mon temps afin d'atteindre le maximum du travail prévu et fournir une application appropriée dans les délais imposés. Ce stage m'a aussi permis de me familiariser avec la vie professionnelle et notamment au travail en équipe ce qui ne peut me pousser qu'à exprimer ma satisfaction envers mes relations avec tout le personnel de l'entreprise qui m'ont beaucoup aidé durant toute la période du stage.

Pour conclure, bien que ce travail satisfasse les objectifs initialement fixés, notre application réalisée pourrait faire l'objet d'amélioration.

On peut s'étendre pour améliorer la méthode de transmission du flux vidéo. Dans le domaine de diffusion en temps réel il existe une grande diversité de protocoles et de techniques. Le plus courant est l'utilisation du protocole RTSP pour réaliser le streaming, mais avec un grand nombre de cameras partageant une même bande passante, cette technique peut montrer une certaine défaillance. La solution optimale sera la création d'un serveur WebRTC et l'implémentation d'une connexion peer-to-peer offrant un service performant de streaming.

D'autre part, on peut penser à la possibilité de gérer un ensemble de robot en même temps, faire donc adapter les interfaces et ajouter les fonctionnalités convenables. Comme le client peut avoir plus qu'un robot fonctionnant à la fois, on peut simplifier les taches, en favorisant la gestion des robots à travers une même application.

L'ajout d'un mode « light » pour l'application représente une autre amélioration possible dans le but d'optimiser le paramétrage.

Références

- [1] Enova robotics. <https://www.enovarobotics.eu/fr/>. [consulté le 01/02/2022].
- [2] Enova products. <https://www.enovarobotics.eu/other-products/>. [consulté le 01/02/2022].
- [3] Bbc news. <https://www.bbc.com/afrique/region-52173171>, Mai 2020. [consulté le 03/02/2022].
- [4] Tv5 monde. <https://information.tv5monde.com/afrique/la-tunisie-se-lance-dans-les-nouvelles-technologies-pour-contrer-le-coronavirus-358938>, Mai 2020. [consulté le 03/02/2022].
- [5] Le robot k5. <https://www.knightscope.com/k5/>. [consulté le 04/02/2022].
- [6] Le robot e-vigilante. <https://www.u-become.com/e-vigilante/>. [consulté le 04/02/2022].
- [7] Le robot ugv de smp robotics. https://smprobotics.com/products_autonomous_ugv/. [consulté le 04/02/2022].
- [8] Interface utilisateur de smp. https://www.ensta-bretagne.fr/jaulin/rapport_pfe_el_abdalaoui.pdf. [consulté le 04/02/2022].
- [9] Precessus up. <https://www.lucas-uzan.fr/cest-quoi-le-processus-unifie/>. [consulté le 06/04/2022].
- [10] Methodologie 2tup. <https://www.lucas-uzan.fr/cest-quoi-le-processus-unifie/>. [consulté le 06/04/2022].
- [11] Communication via web socket. <https://fr.wikipedia.org/wiki/WebSocket>. [consulté le 06/02/2022].
- [12] Communication via le protocole rtsp. <https://www.cctvcameraworld.com/what-is-rtsp/>. [consulté le 04/03/2022].
- [13] Explication du communication rtsp. <https://www.dvraid.com/dvr-general/what-is-rtsp-protocol-in-cctv-systems/>. [consulté le 04/03/2022].
- [14] Mapbox. <https://www.mapbox.com/>. [consulté le 26/04/2022].
- [15] Architecture getx. <https://www.youtube.com/watch?v=iSZpl9n4wdU/>. [consulté le 08/05/2022].
- [16] Visual studio code. <https://blog.webnet.fr/visual-studio-code/>. [consulté le 02/06/2022].
- [17] Android studio. https://fr.wikipedia.org/wiki/Android_Studio. [consulté le

02/06/2022].

- [18] Postman. <https://blog.webnet.fr/presentation-de-postman-outil-multifonction-pour-api-web/>. [consulté le 02/06/2022].
- [19] Lottie animation. <https://lottiefiles.com/>. [consulté le 05/06/2022].
- [20] Swagger. <https://swagger.io/>. [consulté le 05/06/2022].
- [21] Github. <https://fr.wikipedia.org/wiki/GitHub>. [consulté le 05/06/2022].
- [22] Slack. <https://slack.com/>. [consulté le 05/06/2022].
- [23] Visual paradigm. <http://igm.univ-mlv.fr/~dr/XPOSE2010/visualparadigm/diagrammes.htm>. [consulté le 05/06/2022].
- [24] Draw.io. <https://drawio-app.com/>. [consulté le 05/06/2022].
- [25] Adobe xd. <https://www.adobe.com/products/xd.html>. [consulté le 05/06/2022].
- [26] Flutter. <https://flutter.dev/>. [consulté le 03/02/2022].
- [27] Dart. <https://dart.dev/>. [consulté le 03/02/2022].
- [28] Latex. <https://www.latex-project.org/>. [consulté le 05/06/2022].

الملخص

هذا المستند عباره عن ملخص عملي في مشروع التخرج داخل الشركة اينوفا روبيتك. الغرض من هذا المشروع هو انشاء تطبيقة محمولة تمكن من التحكم عن بعد في الروبوت الامني بيكارد وتضمن التتحقق في الوقت الفعلى من حالة الروبوت ومعطياته عبر الكاميرات ومحموعه من اجهزه الاستشعار المدججة . يهدف التطبيق الى تسهيل مهام حارس الامن كونه المراقب الاول للروبوت وتمكنه بنقرات بسيطة من ادارة المهام والسيناريوهات التي يقوم بها الروبوت باستقلاليه كاملة. قمنا بانجاز هذا المشروع من خلال توفير كل الجهد اللازمه لتلبية الاحتياجات الوظيفية والتقنية للعميل في اطار اكتساب معارف ومعلومات جيدة في ما يتعلق بالتقنيات الحديثة مثل فلاتر

الكلمات الرئيسية : تطوير الويب، التحكم عن بعد، روبوت امني، فلاتر، مقبس الويب، النهج دوتيب

Résumé

Ce document est une synthèse de mon travail dans le cadre du projet de fin d'études réalisé au sein de la société *Enova Robotics*. Le but fondamental de ce projet, est de créer une application mobile servant la téléopération du robot de sécurité, le P-Guard, en assurant la vérification en temps réel de son état via des caméras et un ensemble de capteurs intégrés. L'application vise à faciliter les tâches d'un agent de sécurité, comme étant le moniteur du robot, et lui permettre, par de simples clicks, de gérer les missions et les scénarios exécutés par le robot en toute autonomie. Nous avons réalisé ce projet en fournissant tout l'effort nécessaire pour répondre aux besoins fonctionnels et techniques du client, en essayant d'acquérir une bonne maîtrise des technologies récentes telles que Flutter.

Mots-clés : Développement mobile, Téléopération, Robot de sécurité, Flutter, Websocket, méthodologie 2TUP.

Abstract

This document is a summary of my work as part of the end-of-studies project which was developed within the company *Enova Robotics*. The fundamental purpose of this project is to create a mobile application serving the teleoperation of the security robot, P-Guard, and ensuring real-time verification of its status via cameras and a set of integrated sensors. The application aims to facilitate the tasks of a security guard, as being the robot monitor, and to allow him, through simple clicks, to manage the missions and scenarios performed by the robot in full autonomy. We have achieved this project by providing all the necessary effort to meet the functional and technical needs of the client, trying to acquire a good knowledge of recent technologies such as Flutter.

Keywords : Mobile development, Teleoperation, Security robot, Flutter, Websocket, 2TUP methodology.

