Dédicaces

Je dédie ce travail à :

Monsieur , Monsieur pour m'avoir encadré et fait de leurs mieux afin de m'aider.

etc.

Remerciements

Je remercie

Je suis reconnaissant

J'exprime ma gratitude

Table des matières

Introduction générale **1**

1 Introduction Générale **2**

1.1 Introduction . . . 3

1.2 Présentation du cadre du Projet . . . 3

1.2.1 Présentation de l'organisme d'accueil . . . 3

1.2.2 Présentation générale du projet . . . 3

1.3 Étude de l'existant . . . 4

1.3.1 Description de l'existant . . . 4

1.3.2 Critique de l'existant . . . 5

1.3.3 Solution proposée . . . 5

1.4 Étude du projet . . . 5

1.4.1 Contexte du projet . . . 5

1.4.2 Objectifs de l'application . . . 6

1.5 Conclusion . . . 7

2 conception **8**

2.1 Introduction . . . 9

2.2 Identification des Acteurs . . . 9

2.3 Spécifications des besoins . . . 9

2.3.1 Les besoins fonctionnels . . . 9

2.3.2 Les besoins non fonctionnels . . . 10

2.4 Diagrammes de cas d'utilisation . . . 12

2.4.1 Diagramme de cas d'utilisation global . . . 12

2.4.2 Diagramme de cas d'utilisation de gestion des cultures . . . 14

2.4.3 Diagramme de cas d'utilisation de système d'irrigation . . . 15

2.4.4 Diagramme de cas d'utilisation de système de détection de maladie avec IA . . . 16

2.4.5 Diagramme de cas d'utilisation de marché agricole . . . 17

2.5 Diagramme de classe . . . 18

2.6 Diagrammes de séquences . . . 25
2.6.1 Diagramme de séquence globale . . . 26
2.6.2 Diagramme de séquence de l'authentification d'utilisateur . . . 27
2.6.3 Diagramme de séquence de gestion des cultures . . . 28
2.6.4 Diagramme de séquence de système d'irrigation manuelle . . . 29
2.6.5 Diagramme de séquence de système d'irrigation automatique . . . 30
2.6.6 Diagramme de séquence de consultation des données IoT . . . 31
2.6.7 Diagramme de séquence de Détection des maladies des cultures avec IA . . . 32
2.6.8 Diagramme de séquence de système de marché agricole . . . 33
2.6.9 Diagramme de séquence de l'Agribot . . . 35
2.7 Conclusion . . . 36

3 Réalisation **37**

3.1 Introduction . . . 38
3.2 Environnement de travail . . . 38
3.2.1 Environnement logiciel . . . 38
3.2.2 Environnement technique . . . 40
3.2.3 Environnement matériel . . . 42
3.2.4 Tableau Comparatif des Choix Technologiques . . . 44
3.3 Réalisation du montage . . . 46
3.3.1 Réalisation du montage de chaque composante . . . 46
3.4 Réalisation de robot agricole . . . 50
3.4.1 Conception et fabrication . . . 50
3.4.2 Composants matériels . . . 51
3.4.3 Système de contrôle à distance . . . 53
3.4.4 Vision et détection par intelligence artificielle . . . 53
3.5 Choix du modèle d'intelligence artificielle : YOLOv11 . . . 53
3.5.1 Raisons du choix de YOLOv11 . . . 53
3.5.2 Performances du modèle . . . 54
3.5.3 Comparaison avec d'autres modèles . . . 54
3.6 Présentation de quelques interfaces de l'application . . . 55
3.6.1 Logo de l'application . . . 55

3.6.2 Interface de choix de type d'utilisateur 55
3.6.3 Interface de marché agricole pour les clients 56
3.6.4 Interface d'authentification pour l'agriculteur 56
3.6.5 Interface de l'Écran d'accueil 57
3.6.6 Interface de la gestion des cultures 57
3.6.7 Interface du contrôle de l'irrigation manuel 58
3.6.8 Interface de l'irrigation automatique 58
3.6.9 Interface de la météo 59
3.6.10 Interface du dashboard 59
3.6.11 Interface de l'ajout d'un produit au marché 60
3.6.12 Interface de la gestion financière 60
3.6.13 Interface de contrôle de robot 61
3.6.14 Interface de la détection des maladies 61
3.6.15 Interface de l'assistant virtuel « AgriBot » 62
3.7 Conclusion 62

Table des figures

1.1 Logo de Digi Growing 3
1.2 Logo de l'application FarmLogs 4
1.3 logo de l'application AgriWebb 5

2.1 Diagramme de cas d'utilisation global 13
2.2 Diagramme de cas d'utilisation de gestion des cultures 15
2.3 Diagramme de cas d'utilisation de système d'irrigation 16
2.4 Diagramme de cas d'utilisation de système de détection de maladie avec IA 17
2.5 Diagramme de cas d'utilisation de marché agricole 18
2.6 Diagramme de classe 19
2.7 Diagramme de séquence globale 26
2.8 Diagramme de séquence de l'authentification d'utilisateur 27
2.9 Diagramme de séquence de la gestion des cultures 28
2.10 Diagramme de séquence de système d'irrigation manuel 29
2.11 Diagramme de séquence de système d'irrigation automatique 30
2.12 diagramme de séquence de consultation des données IoT 31
2.13 Diagramme de séquence de Détection des maladies des cultures avec IA 32
2.14 Diagramme de séquence de système de marché agricole 34
2.15 Diagramme de séquence de l'Agribot 35

3.1 Logo de Staruml 38
3.2 Logo de l'Android Studio 38
3.3 Logo de Visual Studio Code 39
3.4 Logo de L'Arduino IDE 39
3.5 Logo de Postman 39
3.6 Logo de Creality Slicer 40
3.7 Logo de Flutter 40
3.8 Logo de Dart 40
3.9 Logo de Python 41
3.10 Logo de SQL 41

3.11 Logo de Latex 41
3.12 Logo de Json 41
3.13 Carte Raspberry Pi 42
3.14 Carte ESP32 42
3.15 Capteur DHT22 42
3.16 Capteur d'humidité du sol 43
3.17 Capteur de niveau d'eau 43
3.18 Capteur de flamme 43
3.19 Capteur de pH du sol 43
3.20 Pompe à eau 44
3.21 Câblage du capteur DHT22 47
3.22 Câblage du capteur de l'humidité du sol 48
3.23 Câblage du capteur de ph 48
3.24 Câblage du capteur de niveau d'eau 49
3.25 Câblage de la pompe d'eau 49
3.26 Image de robot agricole 50
3.27 Modèle 3D de robot agricole 50
3.28 l'impression 3D du robot agricole 51
3.29 La précision de modèle IA 54
3.30 logo de l'application 55
3.31 Interface de choix de type d'utilisateur 55
3.32 Interface de marché agricole pour les clients 56
3.33 Interface d'authentification pour l'agriculteur 56
3.34 Interface de l'Écran d'accueil 57
3.35 Interface de la gestion des cultures 57
3.36 Interface du contrôle manuel de l'irrigation 58
3.37 Interface de l'irrigation automatique 58
3.38 Interface de la météo 59
3.39 Interface du dashboard 59
3.40 Interface de l'ajout d'un produit au marché 60
3.41 Interface de la gestion financière 60

3.42 Interface de contrôle de robot agricole 61
3.43 Interface de la détection des maladies 61
3.44 Interface de l'AgriBot 62

Liste des tableaux

2.1 Acteurs et leurs fonctionnalités 10
2.2 les besoins non fonctionnels 11

3.1 Comparaison des technologies mobiles selon différents critères 44
3.2 Comparaison de python avec autres langages 45
3.3 Comparaison entre Raspberry Pi et ESP32-CAM 45
3.4 Comparaison entre ESP32 et Arduino UNO 46
3.5 Détails des connexions entre les capteurs/actionneurs et l'ESP32 47
3.6 Tableau de branchement de robot agricole 52
3.7 Tableau comparatif de YOLOv11 avec d'autres modèles 54

Liste des abréviations

- **IA** = **I**ntelligence **A**rtificielle
- **IOT** = **I**nternet **O**f **T**hings
- **Json** = **J**avaScript **O**bject **N**otation
- **RGPD** = **R**èglement **G**énéral de **P**rotection des **D**onnées
- **SQL** = **S**tructured **Q**uery **L**anguage

Introduction générale

L'agriculture est l'un des principaux piliers de l'économie mondiale. Elle assure la sécurité alimentaire de la population tout en fournissant des emplois et des revenus. Cependant, ce secteur vital est confronté à un certain nombre de défis, notamment les effets du changement climatique.

La gestion inefficace des ressources naturelles telles que l'eau, la baisse de productivité des terres arables et le manque d'outils technologiques appropriés à grande et à petite échelle constituent autant de défis à relever. Face à ces problèmes, il est devenu nécessaire de repenser les méthodes agricoles traditionnelles et d'intégrer des solutions technologiques modernes pour promouvoir une agriculture plus durable, plus efficace et plus intelligente.

C'est dans ce contexte que s'inscrit notre projet de fin d'études, qui vise à concevoir et à développer une application mobile de gestion agricole intelligente " Agrinova", combinant des capteurs IoT, des services météorologiques avancés et des technologies d'intelligence artificielle et de robotiques.

Cette solution vise à fournir aux agriculteurs des outils précis et automatisés pour surveiller leur environnement, optimiser l'irrigation, anticiper les conditions climatiques et recevoir des recommandations personnalisées, le tout dans une interface conviviale.

Ce rapport est composé de trois chapitres. Le premier, "Contexte générale", présente l'organisme d'accueil, une étude de l'existant, les solutions proposées et l'étude de notre projet réalisé. Le deuxième chapitre, "Conception", décrit les besoins fonctionnels et non fonctionnels, ainsi que les principaux diagrammes de conception (cas d'utilisation, classes, séquences). Enfin, le troisième chapitre, "Réalisation", expose les outils utilisés, le fonctionnement du système et l'application finale avec ses interfaces, et on termine ce rapport par une conclusion générale.

Chapitre 1

Introduction Générale

Plan

1 **Introduction** 3

2 **Présentation du cadre du Projet** 3

3 **Étude de l'existant** 4

4 **Étude du projet** 5

5 **Conclusion** 7

1.1 Introduction

Dans ce chapitre, nous allons présenter l'organisme d'accueil Digi Growing, au sein duquel nous avons effectué notre stage de projet de fin d'études (PFE), ainsi que ses activités et services. Nous décrirons ensuite l'étude de l'existant que nous avons menée pour identifier les besoins spécifiques et les solutions à apporter, suivie de la présentation du projet réalisé, en exposant sa problématique, son contexte et ses objectifs.

1.2 Présentation du cadre du Projet

1.2.1 Présentation de l'organisme d'accueil

1.2.1.1 Présentation de Digi Growing

Digi Growing est une entreprise jeune et dynamique, fondée en janvier 2024, spécialisée dans divers domaines de la technologie et de l'innovation. Sa mission est de répondre aux besoins croissants du marché numérique en offrant des solutions innovantes et personnalisées dans les domaines de l'Internet des Objets (IoT) et des systèmes embarqués, du développement web et mobile, de la formation spécialisée, ainsi que de la création et de la gestion de médias.

Figure 1.1 : Logo de Digi Growing

1.2.2 Présentation générale du projet

Agrinova est une application mobile qui améliore la gestion des fermes. Elle utilise des capteurs IoT et des prévisions météo pour optimiser l'irrigation via l'IA, tandis qu'un robot détecte les maladies des plantes et un chatbot offre des conseils agricoles.

1.2.2.1 Problématique

Le secteur agricole est confronté à des défis majeurs tels que le changement climatique, la gestion inefficace des ressources, l'apparition de maladies des plantes et le manque d'outils numériques adaptés. Malgré les avancées technologiques, de nombreux agriculteurs n'ont pas accès à des solutions intelligentes pour gérer efficacement leurs cultures. Cette situation limite la productivité et la rentabilité des exploitations, soulignant la nécessité d'intégrer des technologies avancées pour une agriculture plus durable et efficace.

1.3 Étude de l'existant

Cette partie consiste à analyser les solutions existantes, de faire un état de l'art afin de pouvoir dégager les besoins du projet, et de les prendre en considération lors de la conception et la réalisation de notre travail.

1.3.1 Description de l'existant

Actuellement, plusieurs solutions numériques existent pour répondre aux besoins de l'agriculture intelligente. On retrouve des applications telles que FarmLogs ou AgriWebb qui se concentrent sur l'optimisation de l'irrigation et la surveillance environnementale via des capteurs IoT. Par ailleurs, des robots agricoles comme ceux développés par John Deere ou Blue River Technology sont conçus pour surveiller les cultures et détecter les maladies, mais leurs fonctionnalités restent souvent limitées à un usage spécifique et impliquent des coûts élevés.

Figure 1.2 : Logo de l'application FarmLogs

Figure 1.3 : logo de l'application AgriWebb

1.3.2 Critique de l'existant

Les solutions actuelles sont généralement fragmentées, difficiles à intégrer et souvent complexes à utiliser pour les agriculteurs. Elles manquent d'automatisation avancée et ne permettent pas une interconnexion fluide entre la surveillance environnementale, la détection des maladies et la gestion commerciale. Par exemple, certaines applications se limitent à la gestion de l'irrigation ou du suivi environnemental, tandis que les robots spécialisés, bien que performants, demeurent inaccessibles en raison de leur coût élevé. Cette situation complique la gestion globale des exploitations et empêche une prise de décision optimale par les agriculteurs.

1.3.3 Solution proposée

La solution proposée, Agrinova, vise à regrouper ces fonctionnalités dans une seule application mobile. En intégrant des capteurs IoT, des services météorologiques, des modèles d'intelligence artificielle et un robot agricole intelligent, Agrinova permet d'optimiser l'irrigation, d'automatiser les tâches, de détecter précocement les maladies et de faciliter la gestion commerciale. Cette approche intégrée offre une alternative accessible et performante, répondant de manière globale aux défis de l'agriculture moderne.

1.4 Étude du projet

1.4.1 Contexte du projet

Dans le cadre de notre projet de fin d'études à l'Institut Supérieur des Sciences Appliquées et Technologies de Mahdia (ISSATMA), nous avons développé une application mobile nommée "Agrinova" pour la gestion intelligente des fermes agricoles.

L'application intègre des données en temps réel provenant de capteurs IoT (température, humidité, etc.) et des informations météorologiques via l'API Open-Météo.

Grâce à un modèle d'intelligence artificielle (IA), l'application analyse ces données et propose des recommandations pour optimiser l'irrigation en fonction des conditions climatiques et des cultures.

En complément, le projet inclut un robot agricole capable de détecter les maladies des plantes à l'aide de l'IA et de fournir des solutions adaptées.

L'application offre également des fonctionnalités de gestion financière, permettant aux agriculteurs de suivre leurs dépenses, leurs revenus, d'analyser l'évolution financière à travers des graphiques et d'évaluer la rentabilité des cultures, facilitant ainsi une gestion optimisée de l'exploitation.

De plus, un système d'échange commercial est intégré, permettant aux agriculteurs de publier leurs produits à vendre et aux clients d'acheter directement via l'application, facilitant la mise en relation directe entre eux.

1.4.2 Objectifs de l'application

L'application "Agrinova" permet de faciliter la gestion des fermes agricoles en utilisant des technologies intelligentes. Voici les principaux objectifs :

1) Gestion des cultures : Les agriculteurs peuvent enregistrer, modifier et supprimer leurs cultures. Cela permet une meilleure organisation du travail agricole et une traçabilité claire des activités.

2) Optimisation de l'irrigation : L'application utilise des capteurs IoT et l'IA pour surveiller les conditions climatiques et offrir des recommandations personnalisées. Cela permet de réduire la consommation d'eau, d'améliorer l'efficacité de l'irrigation et de maximiser la productivité des cultures.

3) Automatisation des tâches agricoles : Grâce aux capteurs et aux prévisions météorologiques, l'application automatise l'irrigation et le suivi des cultures, permettant une prise de décision autonome pour le but de réduire les erreurs humaines et d'optimiser l'efficacité des tâches agricoles.

4) Détection des maladies des plantes : Un robot équipé d'IA détecte les maladies et propose des solutions adaptées pour maintenir la santé des cultures.

5) Alertes climatiques : L'application envoie des alertes pour prévenir les agriculteurs des conditions climatiques importantes et des risques.

6) Suivi des finances agricoles : L'application aide à suivre les dépenses, les revenus et les performances des cultures pour une gestion financière optimisée.

7) Assistance virtuelle : Les agriculteurs peuvent interagir avec le chatbot "Agribot" pour

obtenir des réponses rapides et des conseils utiles afin de mieux gérer leurs cultures et résoudre leurs problèmes agricoles.

8) Échange commercial : Cela permet de faciliter les échanges entre agriculteurs et clients en permettant aux agriculteurs de publier leurs produits à vendre, et aux clients de les acheter directement via l'application.

1.5 Conclusion

Ce premier chapitre a d'abord été consacré aux présentations du cadre général du projet ainsi qu'à l'organisme d'accueil, puis les problématiques du secteur agricole notamment à travers un état de l'art et une critique de l'existant pour identifier les solutions les mieux adaptées à ces problématiques, avant de présenter les solutions proposées par "AgriNova" pour y répondre. Dans le prochain chapitre, on abordera l'étude des besoins fonctionnels et non fonctionnels de laquelle découleront les différents diagrammes de conception.

Chapitre 2

CONCEPTION

Plan

1 **Introduction** 9

2 **Identification des Acteurs** 9

3 **Spécifications des besoins** 9

4 **Diagrammes de cas d'utilisation** 12

5 **Diagramme de classe** 18

6 **Diagrammes de séquences** 25

7 **Conclusion** 36

2.1 Introduction

Dans ce chapitre, on va détailler les différents éléments de conception, en effet les besoins fonctionnels et non fonctionnels, les acteurs de notre projet pour les implémenter dans des diagrammes de cas d'utilisations qui décrire une vue générale de projet, le diagramme de classe qui présente les classes et les interfaces du système et les diagrammes de séquence pour détailler les interactions possibles

2.2 Identification des Acteurs

· Agriculteur : Utilisateur principal qui gère ses cultures via l'application.

· Client : Personne qui consulte et achète les produits agricoles publiés.

· Robot Agricole : Engin mobile capturant une vidéo en temps réel pour la détection des maladies.

· IA Détection Maladie : Module qui analyse les images pour identifier les maladies.

· IA Irrigation : Système recommandant des stratégies d'irrigation adaptées.

· Agribot : Assistant virtuel fournissant conseils et recommandations agricoles.

2.3 Spécifications des besoins

La spécification des besoins est une étape cruciale qui comprend les besoins fonctionnels décrivant les fonctionnalités du système et les besoins non fonctionnels présentant les contraintes et exigences externes et internes du système.

2.3.1 Les besoins fonctionnels

Les besoins fonctionnels détaillent les fonctionnalités attendues par différents acteurs du système.

Je vais les présenter sous forme de tableau pour une meilleure lisibilité.

Acteur	Fonctionnalités
Agriculteur	- S'authentifier au système - Gérer les cultures (ajout, modification, suppression) - Gérer les produits - Contrôler manuellement l'irrigation - Consulter les données IoT - Suivre les revenus et dépenses - Contrôler le robot agricole - Consulter l'Agribot pour des conseils
Client	- Consulter le catalogue du marché agricole - Acheter des produits
Robot Agricole	- Capturer des vidéos en temps réel - Se déplacer dans les cultures - Transmettre les vidéos à l'IA
IA Détection Maladie	- Analyser les images reçues - Détecter les maladies - Identifier le type - Proposer des solutions
IA Irrigation	- Analyser les données des capteurs - Vérifier les prévisions météo - Recommander l'irrigation adaptée
Agribot	- Fournir des conseils et recommandations agricoles

Tableau 2.1 : Acteurs et leurs fonctionnalités

2.3.2 Les besoins non fonctionnels

Les besoins non fonctionnels définissent les critères de performance, de sécurité et de qualité que le système doit respecter pour assurer la qualité, la fiabilité et la durabilité du système.

Catégorie	Description
Sécurité	- Protection des données utilisateurs - Authentification sécurisée - Protection contre les attaques
Performance	- Temps de réponse rapide - Support de nombreux utilisateurs - Optimisation des ressources
Disponibilité	- Application toujours accessible - Sauvegarde automatique - Maintenance planifiée
Scalabilité	- Support de la croissance - Architecture modulaire - Gestion de la charge
Maintenabilité	- Code bien organisé - Documentation claire - Tests réguliers
Interopérabilité	- Compatible tous navigateurs - Fonctionne sur mobile - Intègre les API
Accessibilité	- Facile à utiliser - Interface claire
Expérience Utilisateur	- Interface intuitive - Rapide et fluide - Messages clairs
Conformité	- Respect du Règlement Général sur la Protection des Données (RGPD) - Protection des données - Traçabilité des actions
Portabilité	- Fonctionne partout - Support multi-langues - Adapté à tous écrans

Tableau 2.2 : les besoins non fonctionnels

2.4 Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation représente les cas d'utilisation, les acteurs et leurs relations pour modéliser les besoins des utilisateurs et identifier les fonctionnalités et limites du système.

2.4.1 Diagramme de cas d'utilisation global

La figure 4 représente le diagramme de cas d'utilisation global du projet.

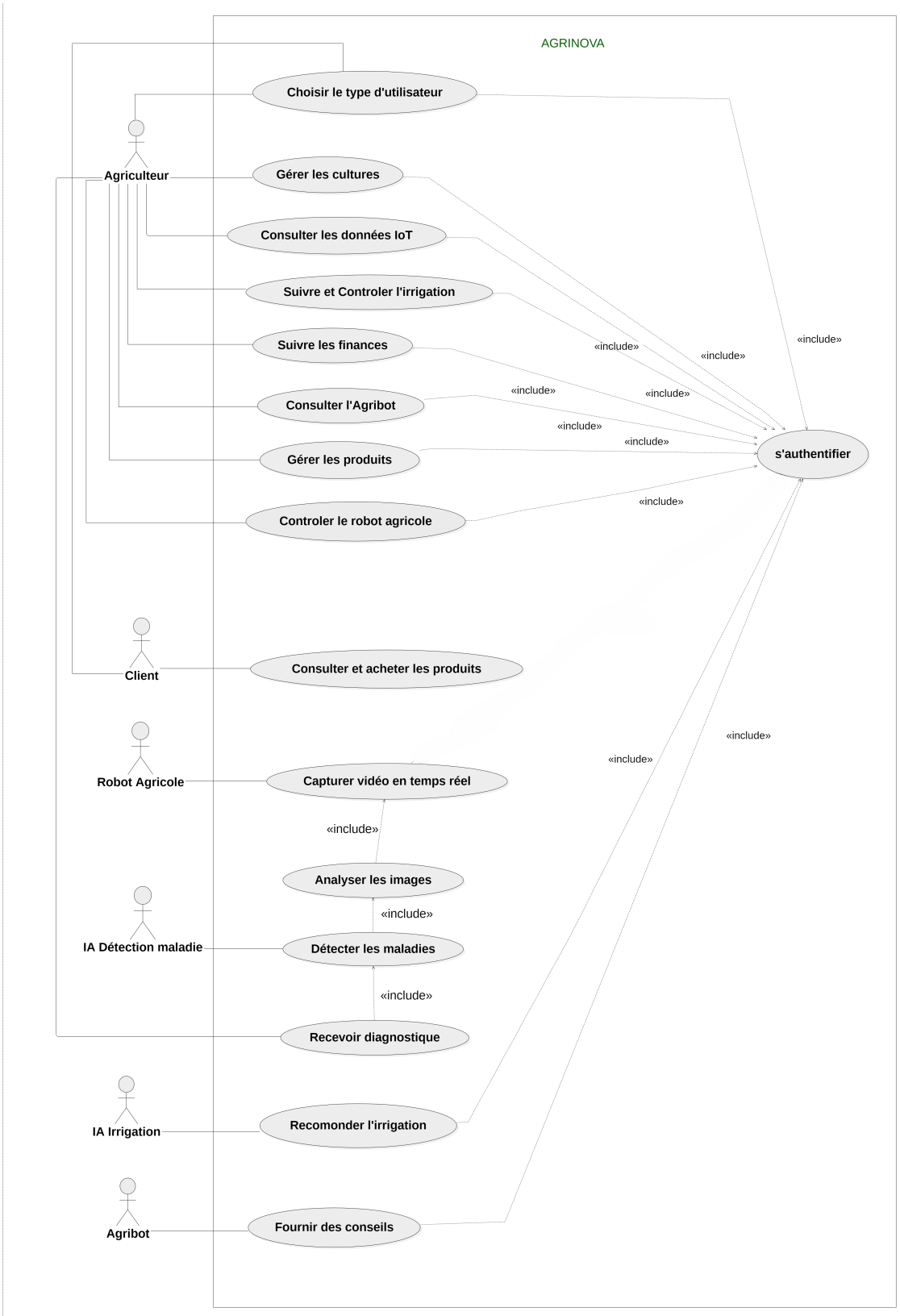

Figure 2.1 : Diagramme de cas d'utilisation global

Agriculteur (acteur principal) :

- Choisir le type d'utilisateur
- Gérer les cultures
- Consulter les données IoT
- Suivre et Contrôler l'irrigation
- Suivre les finances
- Consulter l'Agribot
- Gérer les produits
- Contrôler le robot agricole

Client :

- Consulter et acheter les produits

Robot Agricole :

- Capturer vidéo en temps réel

IA Détection maladie :

- Analyser les images
- Détecter les maladies
- Fournir des solutions

IA Irrigation :

- Recommander l'irrigation

Agribot :

- Fournir des conseils

2.4.2 Diagramme de cas d'utilisation de gestion des cultures

Dans ce diagramme de gestion des cultures, l'agriculteur peut effectuer trois opérations principales : ajouter une nouvelle culture, modifier les informations d'une culture existante, et supprimer une culture du système.

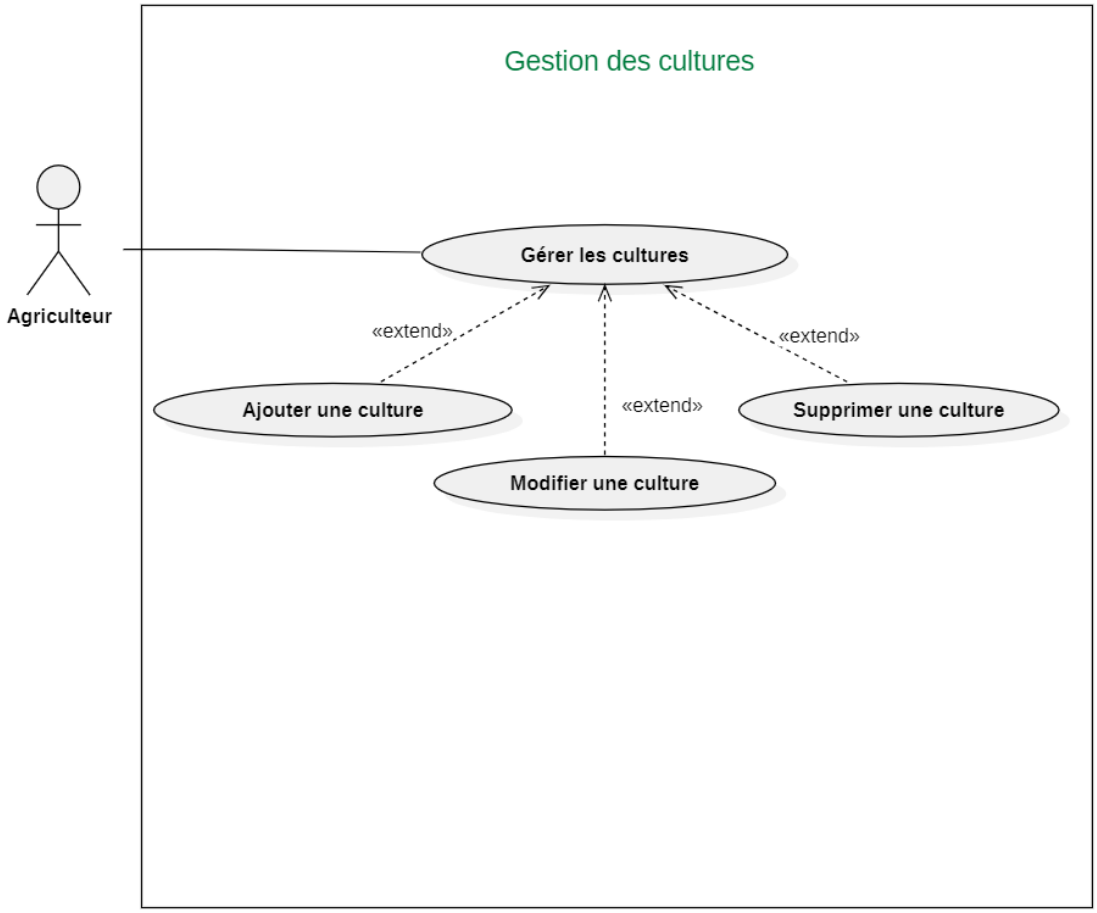

Figure 2.2 : Diagramme de cas d'utilisation de gestion des cultures

2.4.3 Diagramme de cas d'utilisation de système d'irrigation

Ce diagramme présente qu'il existe deux types d'irrigation :

· Manuelle : l'agriculteur contrôle l'irrigation manuellement

· Automatique : Le système d'IA irrigation recommande automatiquement l'irrigation après avoir analysé les données IoT et vérifié les prévisions météo.

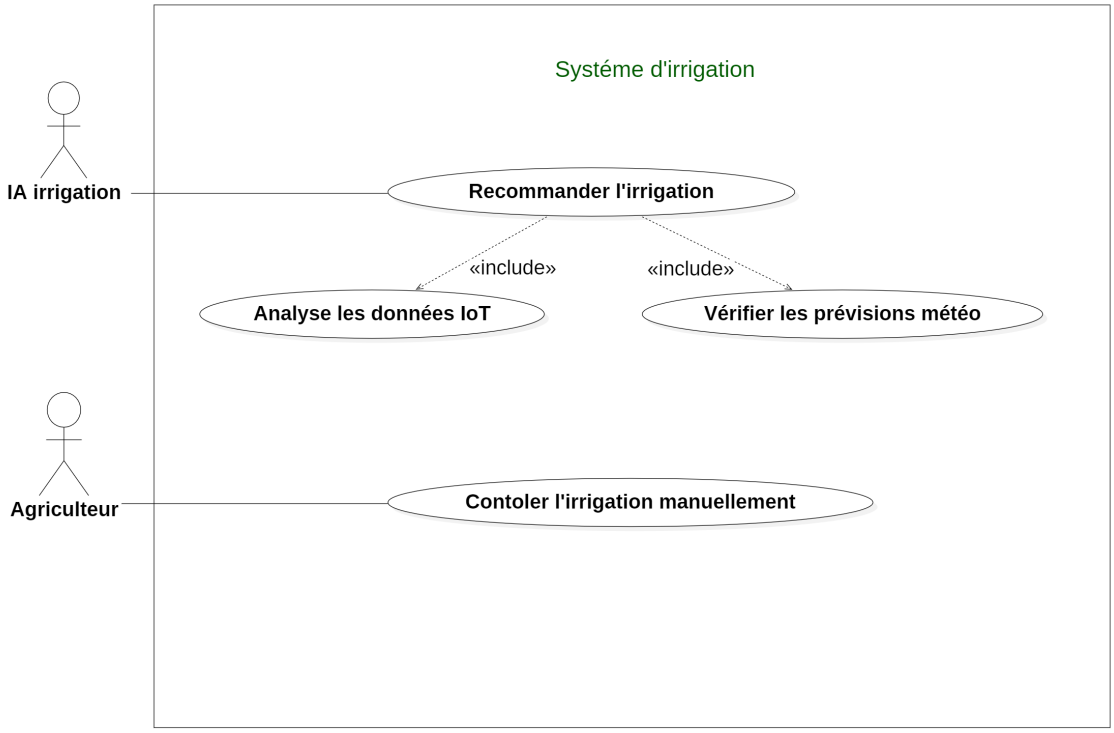

Figure 2.3 : Diagramme de cas d'utilisation de système d'irrigation

2.4.4 Diagramme de cas d'utilisation de système de détection de maladie avec IA

Dans ce diagramme, le système de détection de maladie fonctionne en chaîne : le Robot Agricole capture les vidéos en temps réel, puis l'IA Détection maladie analyse ces images, détecte les maladies éventuelles et propose des solutions adaptées.

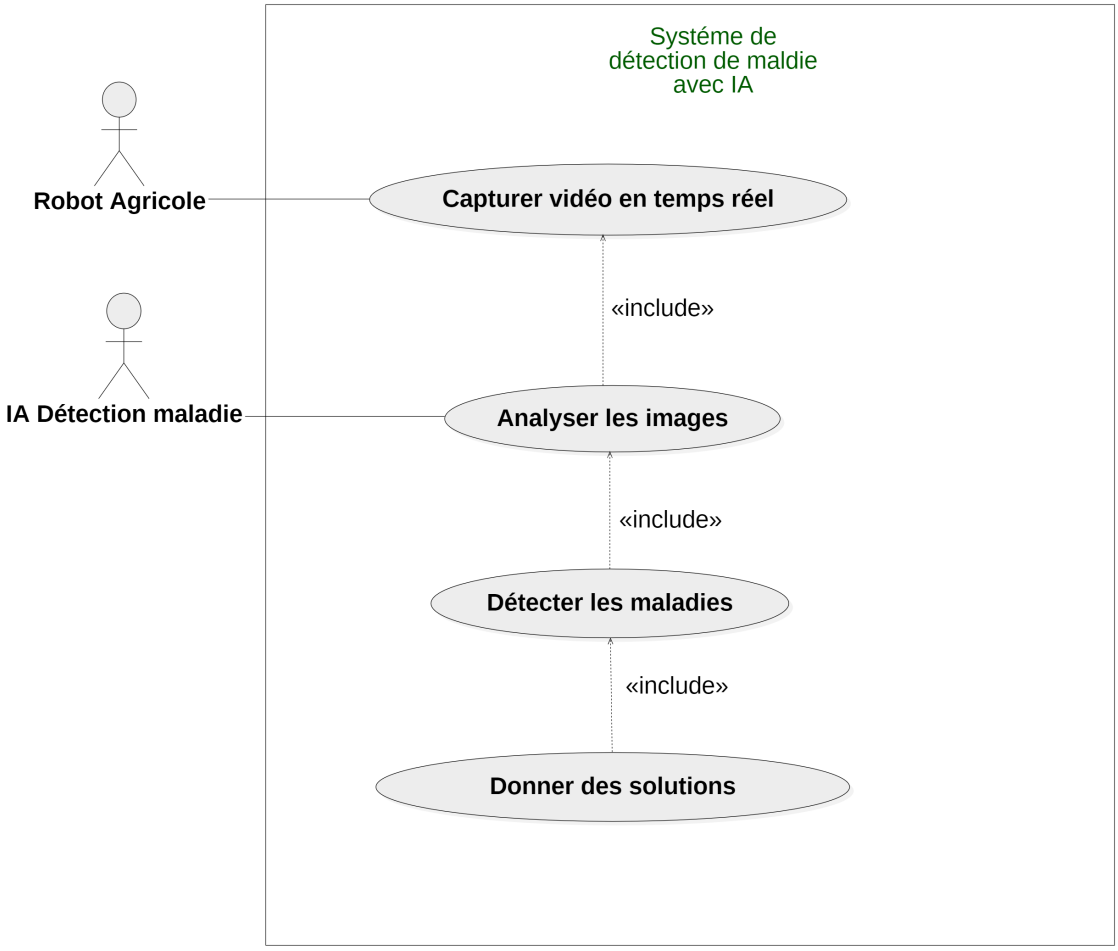

Figure 2.4 : Diagramme de cas d'utilisation de système de détection de maladie avec IA

2.4.5 Diagramme de cas d'utilisation de marché agricole

Dans ce diagramme du Marché Agricole, deux acteurs interagissent : l'Agriculteur qui peut gérer ses produits (ajouter, modifier, supprimer) et le Client qui peut consulter et acheter les produits disponibles et publier par l'agriculteur.

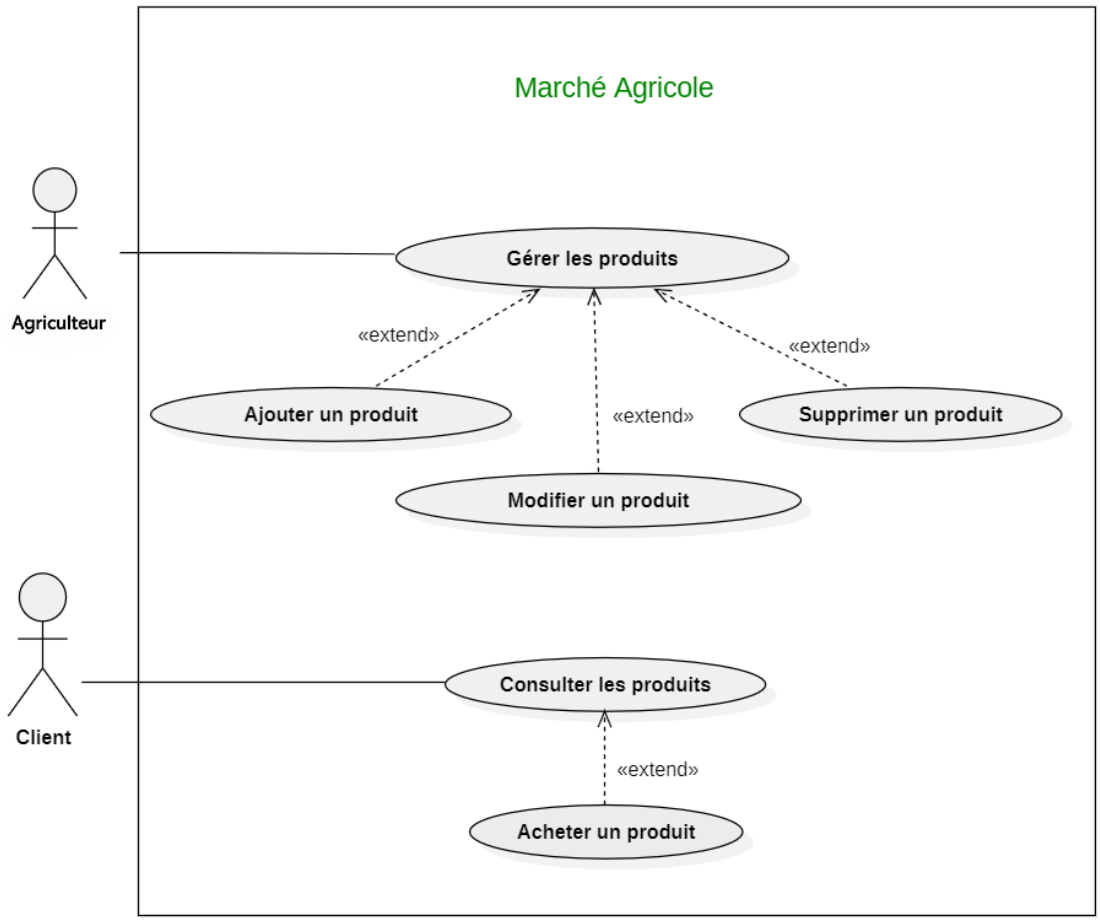

Figure 2.5 : Diagramme de cas d'utilisation de marché agricole

2.5 Diagramme de classe

Pour présenter une vue statique du système, on utilise le diagramme de classe, en effet Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisations.

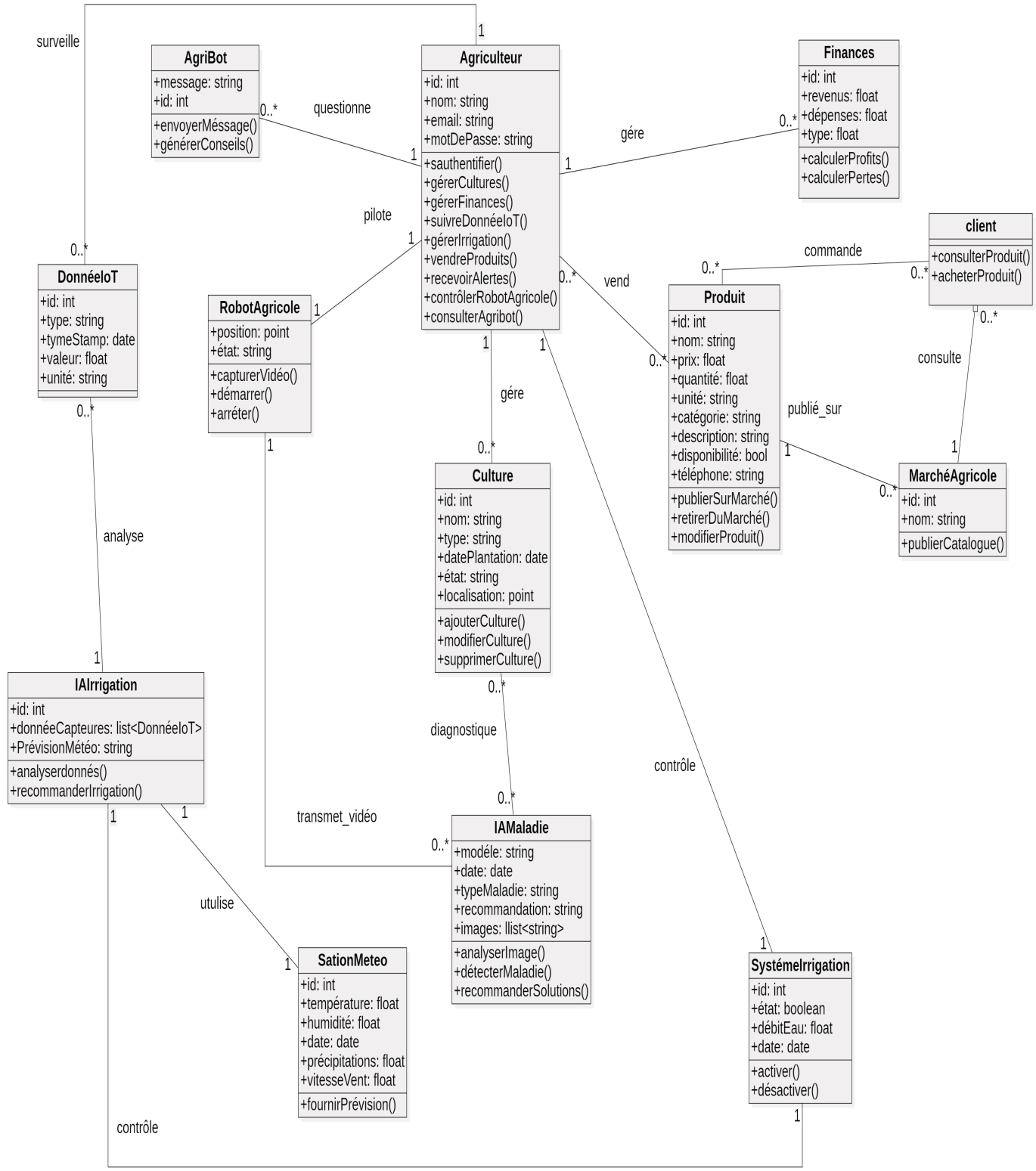

Figure 2.6 : Diagramme de classe

Voici une description détaillée de chaque classe.

Classe Agriculteur :

Cette classe représente l'utilisateur principal du système, qui est l'agriculteur responsable de la gestion globale de la ferme.

Attributs :

— Id : Un entier unique qui identifie l'agriculteur.

— Nom : La chaîne de caractères représentant le nom complet de l'agriculteur.

— Email : La chaîne de caractères utilisés pour l'adresse électronique de l'agriculteur (pour la connexion et la communication).

— MotDePasse : La chaîne cryptée assurant la sécurisation de l'accès au système.

Méthodes :

— S'authentifier () : Permet à l'agriculteur de se connecter au système.

— GererCultures () : Permet de créer, modifier ou supprimer des cultures.

— GererFinances () : Accède à la gestion financière de la ferme (suivi des revenus, dépenses, etc.).

— SuivreDonneeIoT () : Permet de consulter en temps réel les données collectées par les capteurs IoT.

— GererIrrigation () : Active ou désactive et supervise le système d'irrigation.

— VendreProduits () : Gère la mise en vente des produits agricoles.

— RecevoirAlertes () : Reçoit des notifications (alertes sur l'état des cultures, conditions météo, etc.).

— ControlerRobotAgricole () : Interagit avec le robot agricole pour des tâches spécifiques (ex. capturer des vidéos).

— ConsulterAgribot () : Permet d'accéder à l'assistant virtuel (AgriBot) pour obtenir des conseils.

Classe AgriBot : Cette classe représente l'assistant virtuel qui génère des conseils et aide l'agriculteur en communiquant des informations pertinentes.

Attributs :

— Id : Un entier unique identifiant l'assistant AgriBot.

— Message : Une chaîne de caractères contenant le message ou le conseil généré.

Méthodes :

— EnvoyerMessage () : Envoie un message ou une notification à l'agriculteur.

— GenererConseils () : Génère des recommandations personnalisées en fonction des données du système.

Classe DonneeIoT : Cette classe représente les données mesurées par les capteurs IoT déployés sur le terrain.

Attributs :

— Id : Un entier unique identifiant la donnée.

— Type : La chaîne de caractères indiquant le type de mesure (ex. température, humidité, luminosité).

— TimeStamp : La date et l'heure auxquelles la mesure a été effectuée.

— Valeur : La valeur mesurée (de type float) par le capteur.

— Unité : La chaîne de caractères désignant l'unité de mesure (ex. °C,

Classe RobotAgricole : Cette classe représente le robot agricole, qui peut capturer des vidéos en temps réelle sur le terrain.

Attributs :

— Position : Un objet de type Point indiquant la position actuelle du robot sur le terrain.

— Etat : Une chaîne indiquant l'état du robot (par exemple, "en marche", "arrêté").

Méthodes :

— CapturerVideo () : Lance la capture vidéo pour surveiller le terrain.

— Demarrer () : Démarre l'exécution du robot.

— Arreter () : Arrête les opérations du robot.

Classe Culture : Cette classe représente une culture ou une plantation présente dans la ferme.

Attributs :

— Id : Un entier unique pour identifier la culture.

— Nom : Une chaîne désignant le nom de la culture (ex. tomates, blé).

— Type : La catégorie ou le type de la culture (légume, céréale, fruit).

— DatePlantation : La date de plantation de la culture.

— Etat : Une chaîne indiquant l'état de la culture (par exemple, "en croissance", "récolté").

— Localisation : Un objet de type Point représentant l'emplacement de la culture dans la ferme.

Méthodes :

— AjouterCulture () : Ajoute une nouvelle culture au système.

— ModifierCulture () : Modifie les informations d'une culture existante.

— SupprimerCulture () : Supprime une culture du système.

Classe IAMaladie : Cette classe représente le module d'intelligence artificielle dédié au diagnostic des maladies des cultures.

Attributs :

— Modele : Une chaîne décrivant le modèle d'IA utilisé pour l'analyse.

— Date : La date à laquelle le diagnostic a été effectué ou le modèle mis à jour.

— TypeMaladie : La chaîne indiquant le type de maladie détectée.

— Recommandation : Une chaîne fournissant les recommandations (traitement, engrais, pesticide).

— Images : Une liste de chaînes contenant les références ou chemins vers les images utilisées pour l'analyse.

Méthodes :

— AnalyserImage () : Analyse l'image importée et détecte d'éventuelles maladies.

— DetecterMaladie () : Implémente la logique de détection en se basant sur le modèle d'IA.

— RecommanderSolutions () : Propose des solutions et traitements appropriés en fonction du diagnostic.

Classe IAIrrigation : Cette classe représente le module d'intelligence artificielle qui analyse les données et préconise l'irrigation adaptée.

Attributs :

— Id : Un entier unique identifiant le module IAIrrigation.

— DonneesCapteurs : Une liste d'objets de type DonneeIoT regroupant les données actuelles des capteurs.

— PrevisionMeteo : Une chaîne qui contient les prévisions météorologiques récupérées via une API.

Méthodes :

— AnalyserDonnes () : Analyse les données des capteurs pour déterminer les besoins en eau.

— RecommanderIrrigation () : Génère des recommandations sur l'activation ou la modulation de l'irrigation.

Classe StationMeteo : Cette classe représente le module de collecte des données météorologiques essentielles pour l'agriculture.

Attributs :

— Id : Un entier unique identifiant l'enregistrement météo.

— Temperature : Une valeur float représentant la température.

— Humidite : Un float exprimant le pourcentage d'humidité de l'air.

— Date : La date et l'heure de l'enregistrement.

— Precipitations : Un float indiquant la quantité de précipitations (en mm).

— VitesseVent : Un float indiquant la vitesse du vent (en km/h ou m/s).

Méthodes :

— FournirPrevision () : Récupère et renvoie les données météo ou les prévisions.

Classe SystemeIrrigation : Cette classe représente le système d'irrigation qui gère l'arrosage automatique ou manuel des cultures.

Attributs :

— Id : Un entier identifiant unique pour le système.

— Etat : Un booléen indiquant si le système est activé (true) ou désactivé (false).

— DebitEau : Un float représentant le débit d'eau utilisé lors d'un cycle d'irrigation.

— Date : La date et l'heure du dernier cycle d'irrigation.

Méthodes :

— Activer () : Active le système d'irrigation.

— Desactiver () : Désactive le système d'irrigation.

Classe Produit : Cette classe représente un produit agricole mis en vente.

Attributs :

— Id : Un entier unique pour identifier le produit.

— Nom : Nom du produit.

— Prix : Float indiquant le prix unitaire.

— Quantite : Float indiquant la quantité disponible.

— Unite : Chaîne désignant l'unité de mesure (ex. kg, litre).

— Categorie : Catégorie ou type de produit (ex. légume, fruit).

— Description : Description détaillée du produit.

— Disponibilite : Booléen indiquant si le produit est actuellement disponible.

— Telephone : Numéro de téléphone de contact pour la vente.

Méthodes :

— PublierSurMarche () : Met le produit en vente sur le marché.

— RetirerDuMarche () : Retire le produit de la vente.

— ModifierProduit () : Modifie les informations du produit.

Classe MarcheAgricole : Cette classe représente la plateforme de vente externe destinée à la commercialisation des produits agricoles.

Attributs :

— Id : Un entier unique pour identifier le catalogue de la plateforme.

— Nom : Nom de la plateforme de vente ou du marché agricol.

Méthodes :

— PublierCatalogue () : Publie le catalogue de produits sur la plateforme.

Classe Client : Cette classe représente un client potentiel qui consulte et achète des produits sur le marché.

Méthodes :

— ConsulterProduit () : Permet au client de visualiser les produits disponibles.

— AcheterProduit () : Permet au client d'effectuer l'achat d'un produit.

Classe Finances : Cette classe représente la gestion financière de la ferme, permettant de suivre les revenus et les dépenses afin de calculer la rentabilité.

Attributs :

— Id : Un entier unique identifiant le rapport financier.

— Revenus : Float indiquant le total des revenus générés.

— Depenses : Float représentant le total des dépenses engagées.

— Type : (Ce champ peut représenter une catégorisation supplémentaire, par exemple, type de dépense ou de revenu, si nécessaire.)

Méthodes :

— CalculerProfits () : Calcule le profit net (revenus - dépenses).

— CalculerPertes () : Calcule les pertes, le cas échéant.

2.6 Diagrammes de séquences

Le diagramme de séquence est une représentation qui décrit le comportement dynamique de système pour modéliser les interactions entre les objets dans un cas d'utilisation unique.

2.6.1 Diagramme de séquence globale

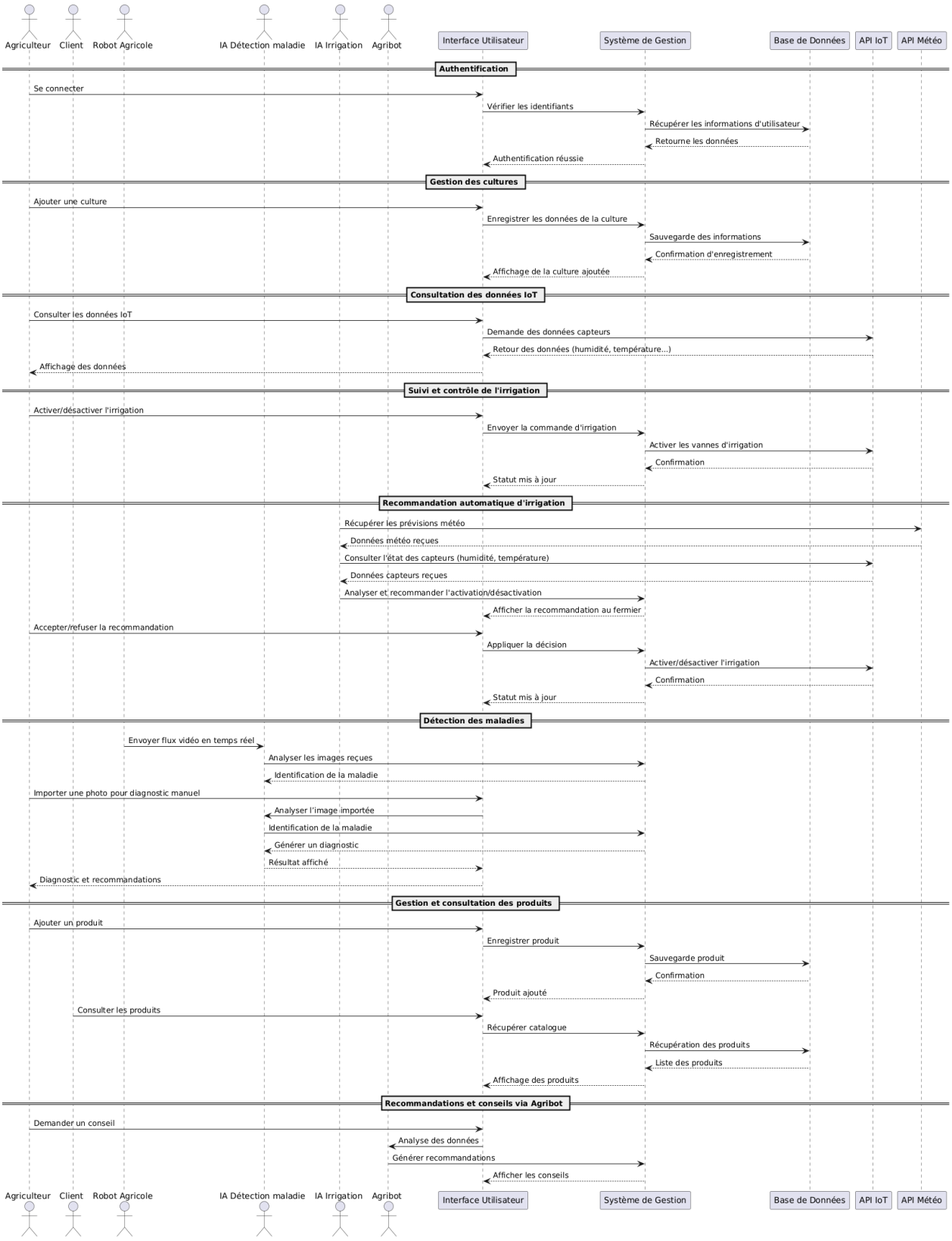

Figure 2.7 : Diagramme de séquence globale

2.6.2 Diagramme de séquence de l'authentification d'utilisateur

Ce diagramme de séquence illustre le processus de connexion sécurisée au système, avec vérification des identifiants et récupération des informations utilisateur.

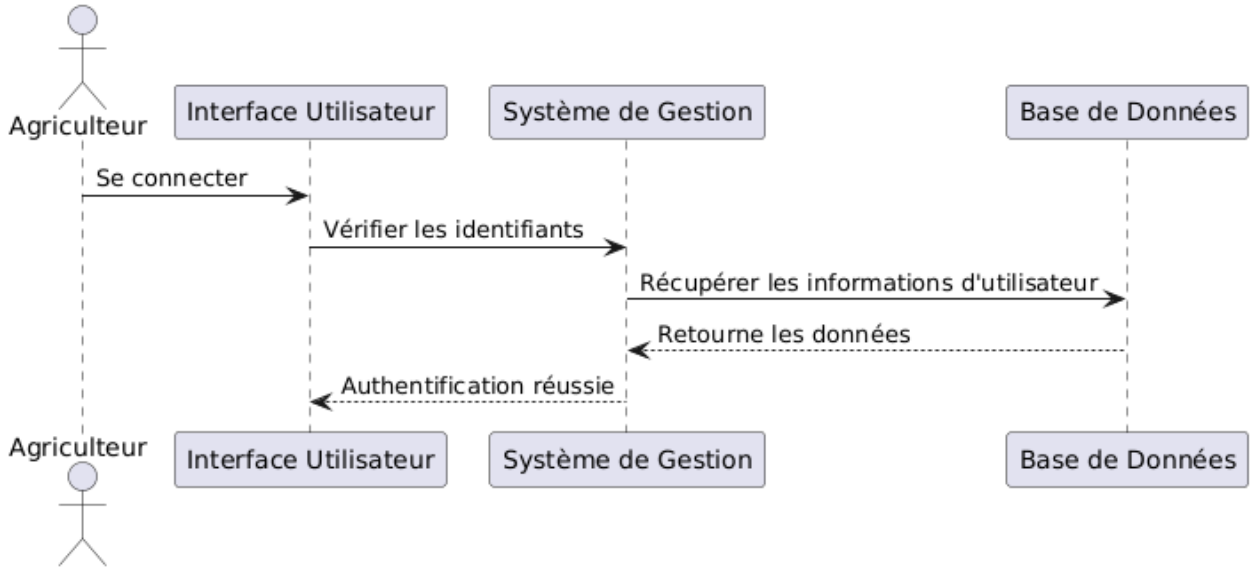

Figure 2.8 : Diagramme de séquence de l'authentification d'utilisateur

Titre du cas : Authentification de l'utilisateur

Objectif : Permettre à l'agriculteur de se connecter de manière sécurisée au système AgriNova.

Résumé : cas d'utilisation décrit le processus d'authentification d'un utilisateur, de la saisie des identifiants jusqu'à la validation de l'accès au système.

Acteur principal :

- Agriculteur

Acteurs secondaires :

- Interface Utilisateur
- Système de Gestion
- Base de Données

Préconditions :

- L'utilisateur dispose d'un compte valide
- Le système d'authentification est fonctionnel
- La base de données est accessible
- L'interface de connexion est disponible

2.6.3 Diagramme de séquence de gestion des cultures

Ce diagramme de séquence montre l'interaction entre l'agriculteur et les différents composants du système pour ajouter une nouvelle culture avec succès.

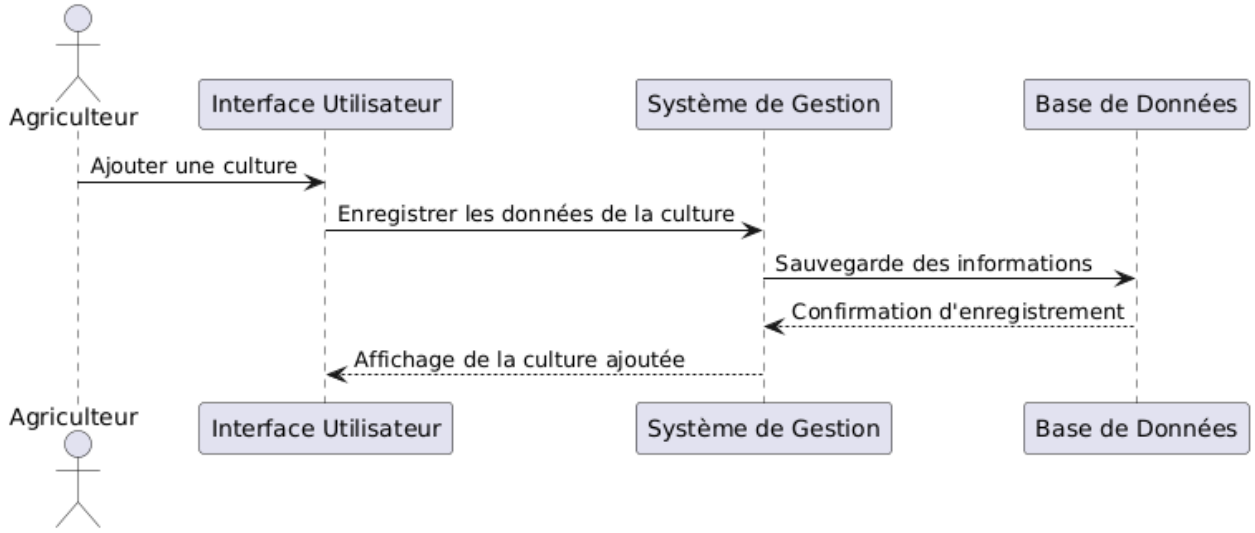

Figure 2.9 : Diagramme de séquence de la gestion des cultures

Titre du cas : Ajouter une culture

Objectif : Permettre à l'agriculteur d'enregistrer une nouvelle culture dans le système AgriNova.

Résumé : Ce cas d'utilisation décrit le processus d'ajout d'une nouvelle culture, depuis la saisie des informations par l'agriculteur jusqu'à la confirmation de l'enregistrement dans la base de données.

Acteur principal :

- Agriculteur

Acteurs secondaires :

- Interface Utilisateur
- Système de Gestion
- Base de Données

Préconditions :

- L'agriculteur est authentifié dans le système
- L'agriculteur a accès au module de gestion des cultures
- L'interface utilisateur est fonctionnelle
- La base de données est accessible

2.6.4 Diagramme de séquence de système d'irrigation manuelle

Ce diagramme de séquence montre l'interaction entre l'agriculteur et le système pour contrôler manuellement l'irrigation, avec confirmation de l'action effectuée.

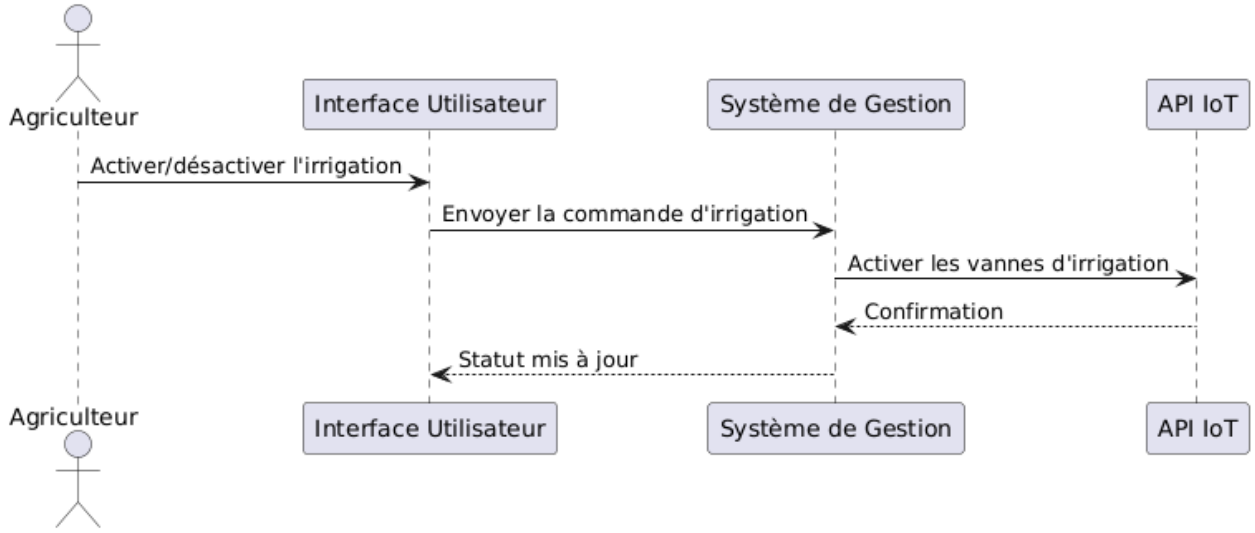

Figure 2.10 : Diagramme de séquence de système d'irrigation manuel

Titre du cas : Contrôle manuel de l'irrigation

Objectif : Permettre à l'agriculteur de contrôler manuellement le système d'irrigation (activation/ désactivation).

Résumé : Ce cas d'utilisation décrit le processus de contrôle manuel du système d'irrigation, depuis la commande de l'agriculteur jusqu'à l'activation effective des vannes d'irrigation.

Acteur principal :

- Agriculteur
- Acteurs secondaires :
- Interface Utilisateur
- Système de Gestion
- API IoT (vannes d'irrigation)

Préconditions :

- L'agriculteur est authentifié dans le système
- Le système d'irrigation est opérationnel
- Les vannes d'irrigation sont connectées
- L'API IoT est accessible et fonctionnelle

2.6.5 Diagramme de séquence de système d'irrigation automatique

Ce diagramme de séquence illustre le processus intelligent de recommandation d'irrigation, combinant données météorologiques, données des capteurs, analyse IA, et validation humaine pour une gestion optimale de l'irrigation .

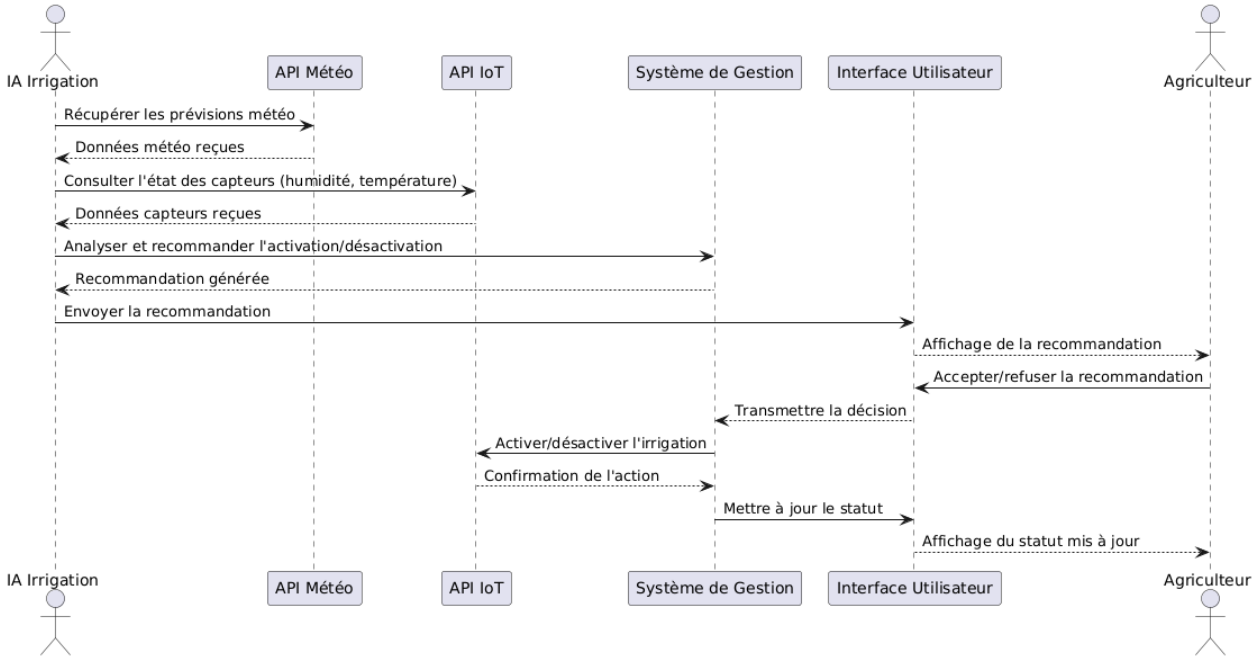

Figure 2.11 : Diagramme de séquence de système d'irrigation automatique

Titre du cas : Recommandation automatique d'irrigation

Objectif : Fournir des recommandations automatiques d'irrigation basées sur l'analyse des données météorologiques et des capteurs.

Résumé : Ce cas d'utilisation décrit le processus complet de génération et de gestion des recommandations d'irrigation, depuis la collecte des données jusqu'à l'action finale sur le système d'irrigation.

Acteur principal :

- IA Irrigation
- Agriculteur

Acteurs secondaires :

- API Météo
- API IoT
- Système de Gestion

- Interface Utilisateur

Préconditions :

- Le système IA d'irrigation est opérationnel
- Les APIs (Météo et IoT) sont accessibles
- Les capteurs sont fonctionnels
- L'agriculteur est connecté au système
- Le système d'irrigation est en état de marche

2.6.6 Diagramme de séquence de consultation des données IoT

Ce diagramme de séquence montre le processus simple mais essentiel de consultation des données des capteurs, permettant à l'agriculteur de surveiller les conditions environnementales de ses cultures.

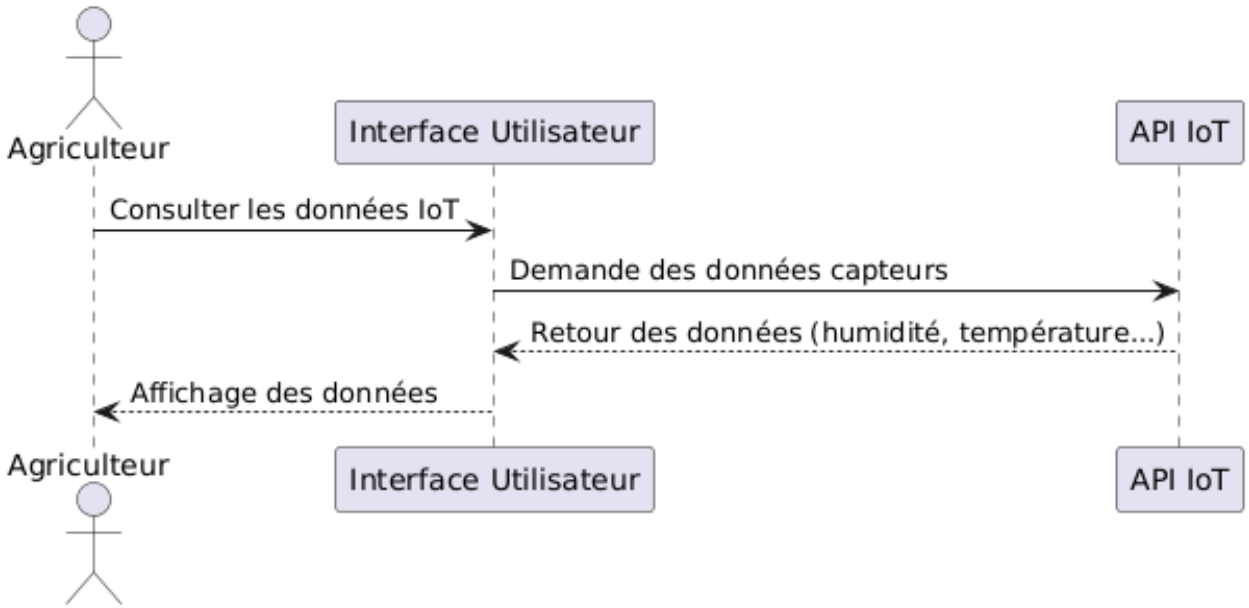

Figure 2.12 : diagramme de séquence de consultation des données IoT

Titre du cas : Consultation des données IoT

Objectif : Permettre à l'agriculteur de visualiser les données en temps réel des capteurs IoT (humidité, température, etc.).

Résumé : Ce cas d'utilisation décrit le processus de consultation des données des capteurs IoT, depuis la demande de l'agriculteur jusqu'à l'affichage des informations.

Acteur principal :

- Agriculteur

- Acteurs secondaires :

- Interface Utilisateur

- API IoT

Préconditions :

- L'agriculteur est authentifié dans le système

- Les capteurs IoT sont fonctionnels

- L'API IoT est accessible

- La connexion réseau est active

2.6.7 Diagramme de séquence de Détection des maladies des cultures avec IA

Ce diagramme de séquence illustre le double processus de détection des maladies, combinant la surveillance automatique par robot et la possibilité d'analyse manuelle par l'agriculteur, pour une détection complète et flexible des problèmes phytosanitaires.

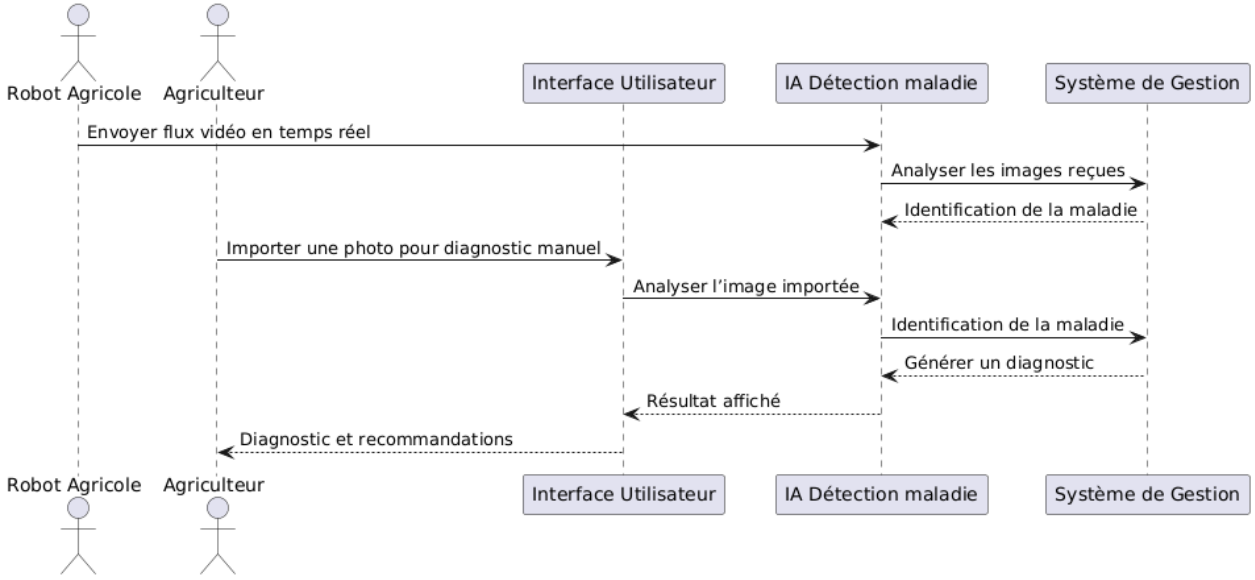

Figure 2.13 : Diagramme de séquence de Détection des maladies des cultures avec IA

Titre du cas :

Détection des maladies des cultures

Objectif : Détecter et diagnostiquer les maladies des cultures à travers l'analyse d'images en temps réel et manuelles.

Résumé : Ce cas d'utilisation décrit deux processus de détection de maladies :

1. Analyse automatique via le flux vidéo du robot agricole

2. Analyse manuelle via des photos importées par l'agriculteur

Acteurs principaux :

- Robot Agricole
- Agriculteur

Acteurs secondaires :

- Interface Utilisateur
- IA Détection maladie
- Système de Gestion

Préconditions :

- Le robot agricole est opérationnel (pour le flux vidéo)
- L'IA de détection est fonctionnelle
- L'agriculteur est authentifié
- La qualité des images est suffisante pour l'analyse
- Le système de gestion est accessible

2.6.8 Diagramme de séquence de système de marché agricole

Ce diagramme de séquence illustre l'interaction entre les vendeurs (agriculteurs) et les acheteurs (clients) dans le système de marché agricole, montrant le flux complet depuis l'ajout d'un produit jusqu'à sa consultation par les clients potentiels.

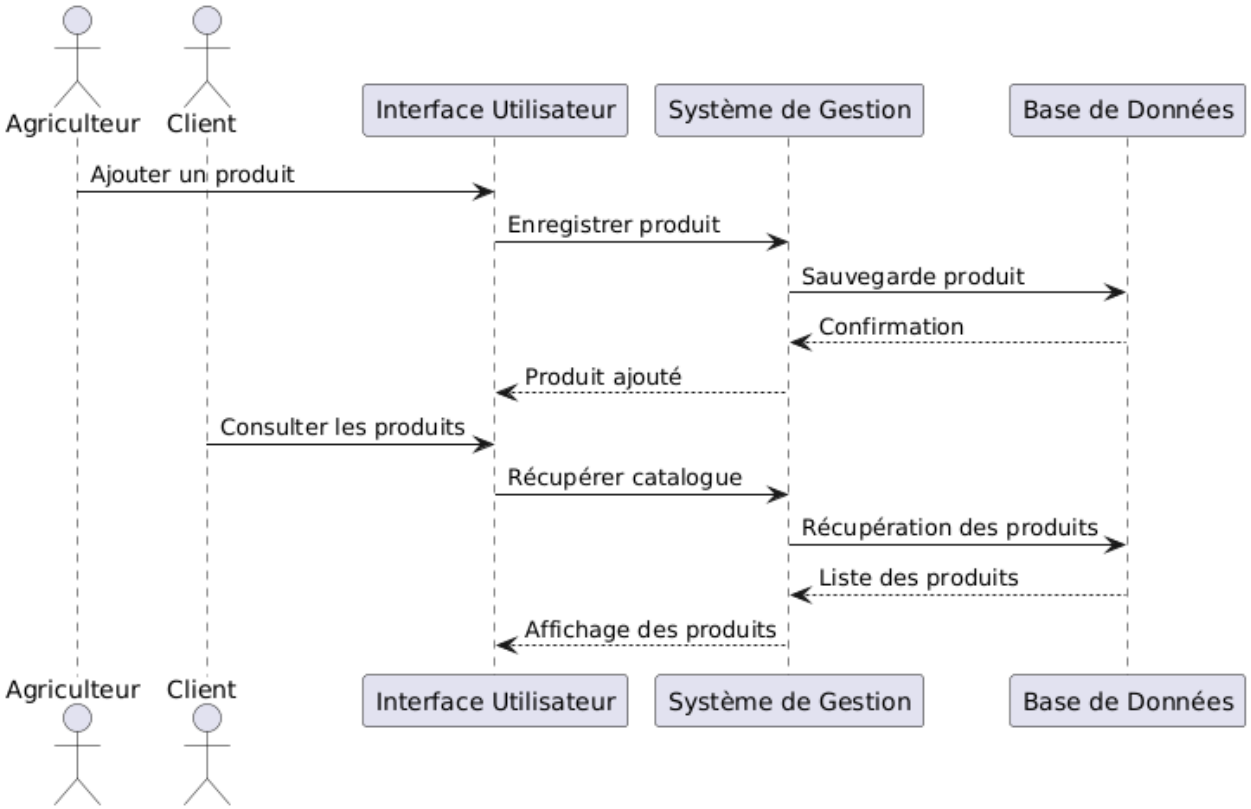

Figure 2.14 : Diagramme de séquence de système de marché agricole

Titre du cas : Gestion et consultation des produits

Objectif : Permettre à l'agriculteur d'ajouter des produits et au client de consulter le catalogue des produits disponibles.

Résumé : Ce cas d'utilisation décrit deux processus complémentaires :

1. L'ajout de produits par l'agriculteur.

2. La consultation du catalogue par le client

Acteurs principaux :

- Agriculteur

- Client

Acteurs secondaires :

- Interface Utilisateur

- Système de Gestion

- Base de Données

Préconditions :

- L'agriculteur est authentifié pour ajouter des produits
- Le client a accès au système de consultation
- La base de données est opérationnelle
- Le système de gestion des produits est fonctionnel

2.6.9 Diagramme de séquence de l'Agribot

Ce diagramme de séquence montre le processus d'interaction entre l'agriculteur et l'assistant virtuel Agribot, illustrant comment le système analyse les données et génère des conseils personnalisés pour optimiser la gestion agricole.

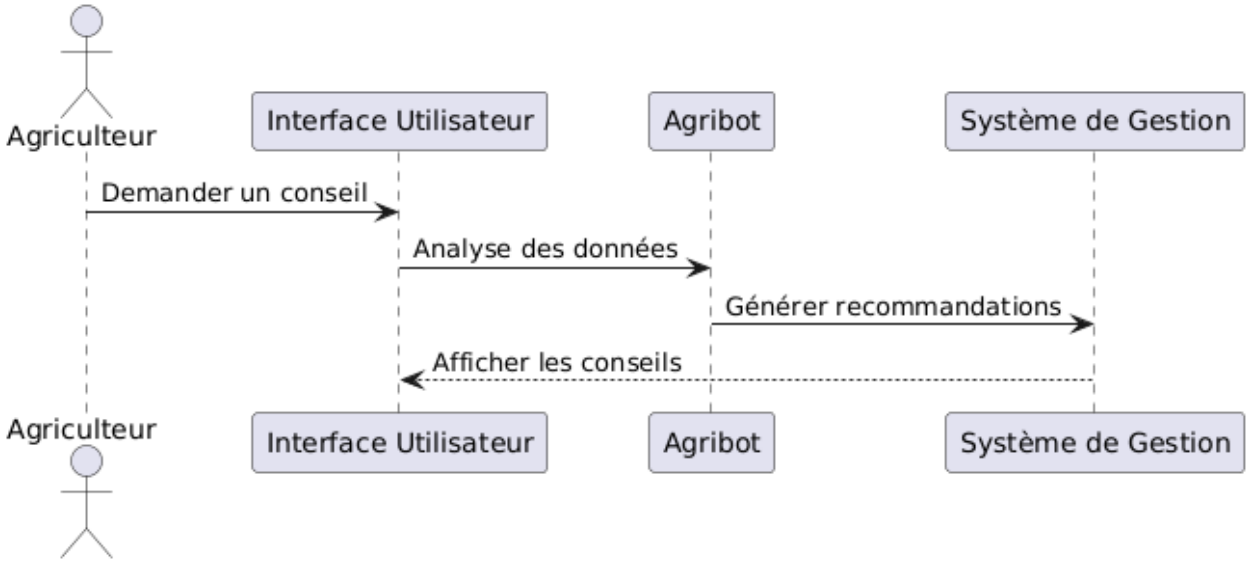

Figure 2.15 : Diagramme de séquence de l'Agribot

Titre du cas : Consultation de l'Agribot

Objectif : Permettre à l'agriculteur d'obtenir des conseils personnalisés via l'assistant virtuel Agribot.

Résumé : Ce cas d'utilisation décrit le processus de demande et d'obtention de conseils agricoles automatisés via l'Agribot, depuis la requête de l'agriculteur jusqu'à l'affichage des recommandations.

Acteurs principaux :

- Agriculteur
- Agribot

Acteurs secondaires :

- Interface Utilisateur

- Système de Gestion

Préconditions :

- L'agriculteur est authentifié dans le système
- L'Agribot est opérationnel
- Le système de gestion contient des données suffisantes
- L'interface utilisateur est fonctionnelle

2.7 Conclusion

En conclusion de ce chapitre, nous avons présenté la conception de projet en effet, les besoins fonctionnels et non fonctionnels, les diagrammes de cas d'utilisations pour montrer la vue globale du projet, le digramme de classe pour exposer une vue statique et les diagrammes de la séquence qui décrivent une vue dynamique de projet.

Chapitre 3

Réalisation

Plan

1 **Introduction** **38**

2 **Environnement de travail** **38**

3 **Réalisation du montage** **46**

4 **Réalisation de robot agricole** **50**

5 **Choix du modèle d'intelligence artificielle : YOLOv11** **53**

6 **Présentation de quelques interfaces de l'application** **55**

7 **Conclusion** **62**

3.1 Introduction

Dans ce chapitre, nous allons présenter l'environnement matériel et logiciel utilisé pour le développement de la solution proposée, tout en expliquant nos choix techniques concernant les langages de programmation et les outils utilisés. Enfin, nous donnerons une vue d'ensemble des interfaces de l'application, accompagnée d'une description du fonctionnement global du système.

3.2 Environnement de travail

3.2.1 Environnement logiciel

- StarUML

StarUML est un outil de modélisation UML qui permet de créer des diagrammes pour concevoir des systèmes logiciels. Il prend en charge des diagrammes UML 2.x et est compatible avec plusieurs plateformes (Windows, macOS, Linux). [1]

Figure 3.1 : Logo de Staruml

- Android Studio

Android Studio est un environnement de développement intégré (IDE) officiel pour créer des applications Android. Il offre des outils puissants pour le codage, la conception et le débogage. [2]

Figure 3.2 : Logo de l'Android Studio

- Visual Studio Code

Visual Studio Code (VS Code) est un éditeur de code léger et open-source, offrant des fonctionnalités

puissantes comme le débogage, la gestion de versions et l'intégration avec divers langages de programmation. [3]

Figure 3.3 : Logo de Visual Studio Code

- Arduino IDE

L'Arduino IDE est un environnement de développement intégré utilisé pour programmer les cartes Arduino. Il permet d'écrire, de compiler et de télécharger des programmes (sketches) vers les cartes Arduino via une interface simple et accessible. [4]

Figure 3.4 : Logo de L'Arduino IDE

- Postman

Postman est un outil utilisé pour tester et développer des APIs, permettant d'envoyer des requêtes HTTP et d'analyser les réponses. [5]

Figure 3.5 : Logo de Postman

- Creality Slicer

Creality Slicer est un logiciel de découpe 3D basé sur Cura, utilisé pour préparer les modèles à imprimer sur les imprimantes Creality en générant du G-code. [6] [7]

Figure 3.6 : Logo de Creality Slicer

3.2.2 Environnement technique

- Flutter

Flutter est un framework open-source de Google pour créer des applications mobiles, web et de bureau avec un seul code base. [8]

Figure 3.7 : Logo de Flutter

- Dart

Dart est un langage de programmation optimisé pour les applications front-end, principalement utilisé avec Flutter pour créer des interfaces utilisateur mobiles. [9]

Figure 3.8 : Logo de Dart

- Python

Python est un langage de programmation polyvalent, facile à apprendre, utilisé pour le développement web, l'analyse de données, l'intelligence artificielle et plus encore. [10]

Figure 3.9 : Logo de Python

- SQL

SQL (Structured Query Language) est un langage utilisé pour gérer et interroger des bases de données relationnelles. [11]

Figure 3.10 : Logo de SQL

- LaTeX

LaTeX est un langage de composition de documents, utilisé pour créer des documents scientifiques ou techniques avec une mise en page soignée, notamment pour les formules mathématiques . [21]

Figure 3.11 : Logo de Latex

- Json

JSON (JavaScript Object Notation) est un format de données léger, facile à lire et à écrire, couramment utilisé pour échanger des données entre serveurs et clients.[12]

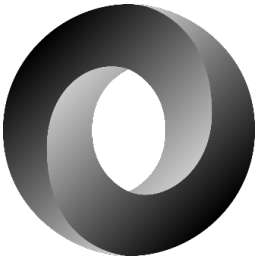

Figure 3.12 : Logo de Json

3.2.3 Environnement matériel

- Raspberry Pi

Le Raspberry Pi est un ordinateur monocarte à faible coût et de petite taille, conçu pour l'enseignement de l'informatique et largement utilisé dans des domaines tels que la domotique, la robotique et l'IoT. [13]

Figure 3.13 : Carte Raspberry Pi

- ESP32

L'ESP32 est un microcontrôleur économique et économe en énergie, intégrant des fonctionnalités Wi-Fi et Bluetooth, idéal pour les applications IoT. [14]

Figure 3.14 : Carte ESP32

- Capteur DHT22

Le DHT22 est un capteur numérique de température et d'humidité, offrant une haute précision et une large plage de mesure. [15]

Figure 3.15 : Capteur DHT22

- Capteur d'humidité du sol

Ce capteur mesure la teneur en eau du sol, essentielle pour l'irrigation agricole et la surveillance environnementale. [16]

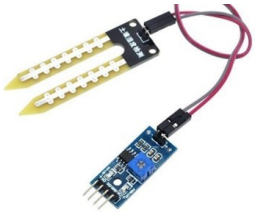

Figure 3.16 : Capteur d'humidité du sol

- Capteur de niveau d'eau

Un capteur de niveau d'eau détecte la hauteur du liquide dans un récipient, convertissant cette information en un signal électrique. [17]

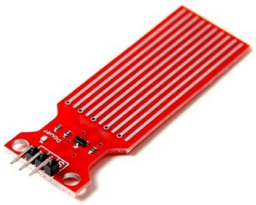

Figure 3.17 : Capteur de niveau d'eau

- Capteur de flamme

Ce capteur détecte la présence d'une flamme dans des systèmes à gaz, assurant la sécurité en interrompant l'alimentation en gaz en cas d'absence de flamme. [18]

Figure 3.18 : Capteur de flamme

- Capteur de pH du sol

Un capteur de pH du sol mesure l'acidité ou l'alcalinité du sol en détectant la concentration d'ions hydrogène, fournissant des données précieuses pour l'agriculture. [19]

Figure 3.19 : Capteur de pH du sol

- Pompe à eau

Une pompe à eau est un dispositif mécanique conçu pour déplacer l'eau d'un endroit à un autre en créant une différence de pression. Elle est utilisée dans divers domaines tels que l'agriculture. [20]

Figure 3.20 : Pompe à eau

3.2.4 Tableau Comparatif des Choix Technologiques

- Flutter vs Autres Frameworks Mobile

Critère	**Flutter**	**React Native**	**Native Android**	**Native iOS**
Performance	Compilation native, performances optimales	Performances moyennes à cause du bridge JavaScript	Performances natives optimales	Performances natives optimales
Développement	Hot Reload instantané, développement rapide	Rechargement plus lent, développement moyen	Développement plus lent, compilation nécessaire	Développement plus lent, compilation nécessaire
Cross-platform	Un seul code pour Android et iOS	Un seul code mais avec limitations	Code spécifique Android uniquement	Code spécifique iOS uniquement
UI/UX	Widgets personnalisables, design cohérent	Dépendance des composants natifs, design variable	Design natif Android	Design natif iOS
Communauté	Communauté en forte croissance	Grande communauté établie	Très grande communauté	Très grande communauté

Tableau 3.1 : Comparaison des technologies mobiles selon différents critères

Raison du choix de flutter : Meilleur compromis entre performance et rapidité de développement

• Python vs Autres Langages pour l'IA

Critère	**Python**	**C++**	**Java**	**JavaScript**
Bibliothèques IA	Bibliothèques complètes (TensorFlow, OpenCV...)	Bibliothèques limitées	Peu de bibliothèques spécialisées	Peu de bibliothèques spécialisées
Facilité d'utilisation	Syntaxe claire et intuitive	Syntaxe complexe	Syntaxe verbeuse	Syntaxe flexible
Performance	Bonnes performances avec optimisations	Performances optimales	Bonnes performances	Performances limitées
Communauté IA	Très grande communauté active en IA	Communauté moyenne en IA	Petite communauté en IA	Petite communauté en IA
Intégration	Intégration facile avec Flask et autres frameworks	Intégration plus complexe	Intégration moyenne	Intégration moyenne

Tableau 3.2 : Comparaison de python avec autres langages

Raison du choix de python : Meilleur écosystème pour l'IA et le traitement d'images

• Raspberry Pi vs ESP32-CAM

Critère	**Raspberry Pi**	**ESP32-CAM**
Puissance CPU	Processeur quad-core puissant	Processeur single-core limité
Mémoire	4GB+ de RAM, stockage extensible	512KB de RAM, stockage limité
Traitement IA	Capable d'exécuter YOLOv8 et traitement d'images complexe	Trop limité pour le traitement IA
Caméra	Support caméra HD avec autofocus, qualité professionnelle	Qualité d'image limitée, pas d'autofocus
Connectivité	WiFi, Ethernet, Bluetooth, USB	WiFi uniquement
Prix	Plus cher mais justifié par les capacités	Moins cher mais capacités limitées

Tableau 3.3 : Comparaison entre Raspberry Pi et ESP32-CAM

Raison du choix de Raspberry Pi : Nécessaire pour le traitement IA lourd et la qualité d'image

• ESP32 vs Arduino UNO

Critère	**ESP32**	**Arduino UNO**
Puissance CPU	Processeur dual-core à 240MHz	Processeur single-core à 16MHz
Mémoire	520KB de SRAM, 4MB de flash	2KB de SRAM, 32KB de flash
WiFi	Module WiFi intégré	Nécessite module externe
Bluetooth	Module Bluetooth intégré	Nécessite module externe
GPIO	34 pins GPIO, plus de flexibilité	14 pins GPIO, plus limité
Prix	Prix similaire pour plus de fonctionnalités	Prix similaire pour moins de fonctionnalités

Tableau 3.4 : Comparaison entre ESP32 et Arduino UNO

Raison du choix de ESP32 : Connectivité sans fil intégrée et puissance de traitement supérieure

3.2.4.1 Impact sur le Projet

Performance Globale :

- Traitement IA efficace
- Interface utilisateur fluide
- Communication rapide

Développement

- Temps de développement réduit
- Maintenance simplifiée
- Documentation abondante

Coût

- Équilibre performance/prix
- Solutions open-source
- Matériel accessible

3.3 Réalisation du montage

3.3.1 Réalisation du montage de chaque composante

Dans cette partie, on va présenter les différents montages de notre projet.

Capteur	**Port Capteur**	**Port ESP32**
DHT22	VCC GND DATA	3.3V ou 5V GND GPIO 14
Humidité du sol	VCC GND AO	3.3V ou 5V GND GPIO 32
Capteur de flamme	VCC GND DO	3.3V ou 5V GND GPIO 27
Capteur de pH (v1.1)	VCC GND AO	5V GND GPIO 34
Niveau d'eau	VCC GND AO	3.3V ou 5V GND GPIO 35
Pompe	VCC GND Signal	5V GND GPIO 26

Tableau 3.5 : Détails des connexions entre les capteurs/actionneurs et l'ESP32

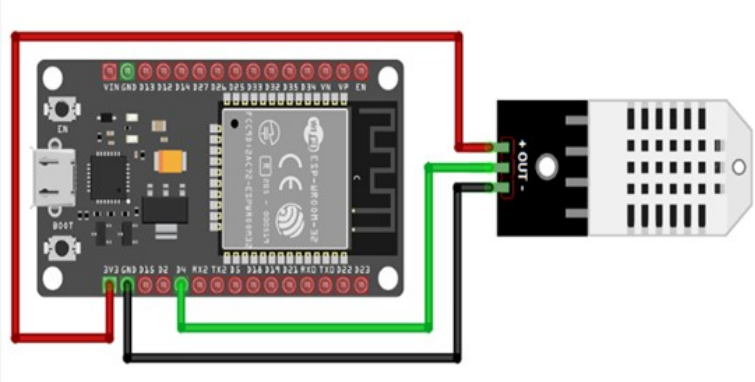

Figure 3.21 : Câblage du capteur DHT22

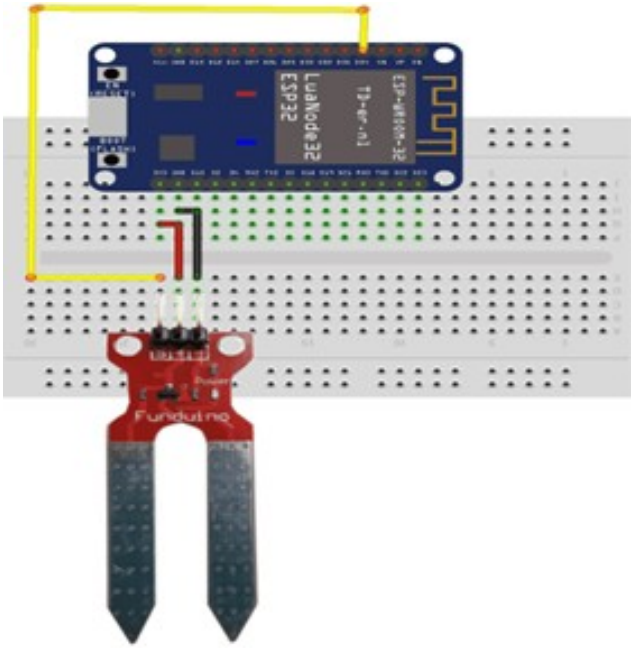

Figure 3.22 : Câblage du capteur de l'humidité du sol

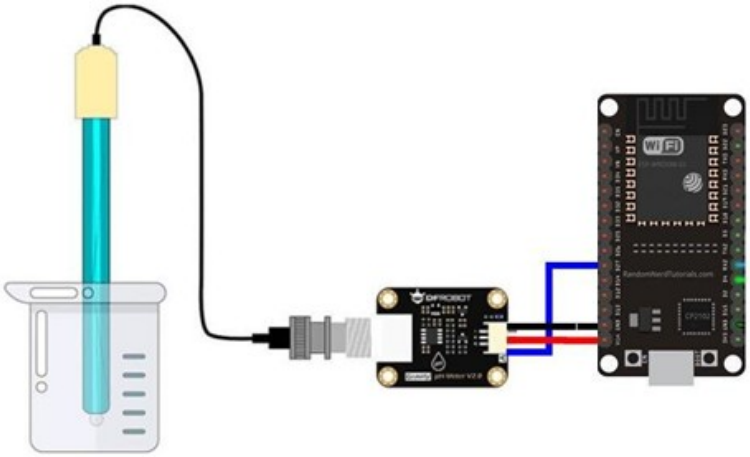

Figure 3.23 : Câblage du capteur de ph

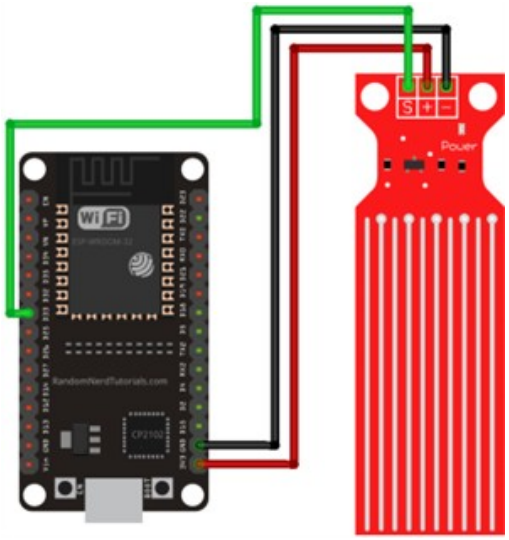

Figure 3.24 : Câblage du capteur de niveau d'eau

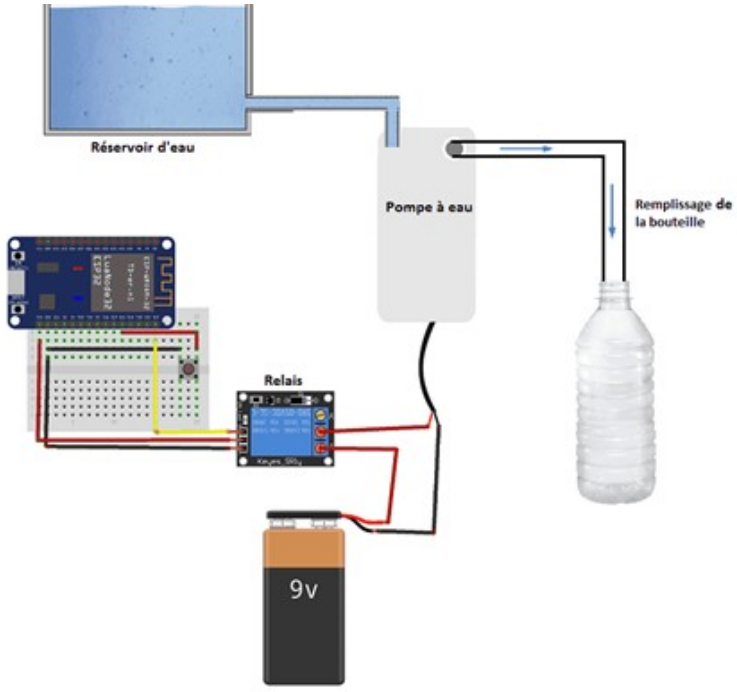

Figure 3.25 : Câblage de la pompe d'eau

3.4 Réalisation de robot agricole

Dans cette partie on va aborder la construction de robot agricole capable d'assister dans la surveillance et l'analyse de l'état des cultures. Ce robot est conçu pour être contrôlé à distance via l'application "AgriNova" avec une interface intuitive basée sur des boutons de contrôles et un joystick.

Figure 3.26 : Image de robot agricole

3.4.1 Conception et fabrication

La structure du robot a été modélisée en 3D à l'aide du logiciel Creality Slicer. Cette étape nous a permis de concevoir un design adapté aux exigences agricoles : stabilité, mobilité sur sol irrégulier, et espace pour intégrer les composants électroniques. Une fois le modèle validé, nous avons imprimé les différentes parties avec une imprimante 3D, puis procédé à l'assemblage physique.

Voici quelques photos de modèles 3D de robots :

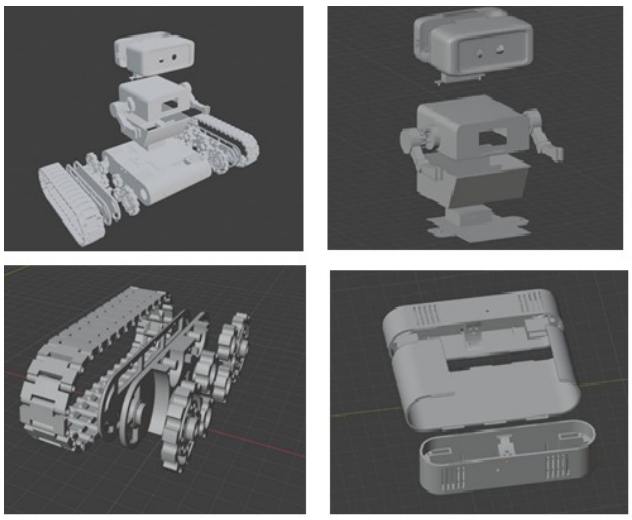

Figure 3.27 : Modèle 3D de robot agricole

Voici une photo réelle prise lors de l'impression 3D :

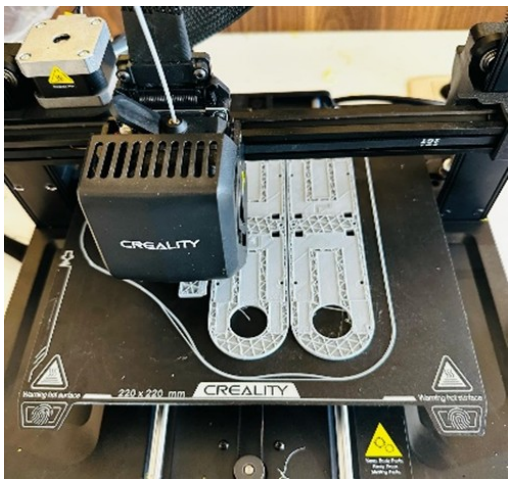

Figure 3.28 : l'impression 3D du robot agricole

3.4.2 Composants matériels

Le robot intègre les éléments suivants :

· Module L298N : utilisé pour le contrôle des moteurs à courant continu.

· Deux moteurs DC (12V) : pour le déplacement du robot.

· Batterie Li-ion 3.7V et support pour piles : pour l'alimentation électrique.

· Caméra Raspberry Pi : positionnée à l'avant pour la capture de vidéos en temps réel.

Voici le tableau de branchement de robot agricole :

Composant	**Broche/Port**	**Connecté à**
L298N ENA	Active moteur A (PWM)	D13 (ESP32)
L298N IN1	Direction moteur A	D12 (ESP32)
L298N IN2	Direction moteur A	D14 (ESP32)
L298N IN3	Direction moteur B	D27 (ESP32)
L298N IN4	Direction moteur B	D26 (ESP32)
L298N ENB	Active moteur B (PWM)	D25 (ESP32)
L298N OUT1/OUT2	Vers moteur DC A	Fils du moteur gauche
L298N OUT3/OUT4	Vers moteur DC B	Fils du moteur droit
L298N VCC	Alimentation moteurs	+ des batteries Li-ion
L298N GND	Masse batterie	- de la batterie + GND ESP32
ESP32 GND	Masse	Connecté à GND du L298N
ESP32 USB	Alimentation ESP32	Via câble USB

Tableau 3.6 : Tableau de branchement de robot agricole

3.4.3 Système de contrôle à distance

Le robot est piloté via L'application mobile "AgriNova", utilisant des WebSockets pour permettre une communication rapide et continue avec l'unité embarquée. L'interface permet à l'utilisateur de contrôler la direction du robot à l'aide d'un joystick virtuel et de boutons.

3.4.4 Vision et détection par intelligence artificielle

Les vidéos transmises par la caméra du robot sont analysées en temps réel afin de détecter d'éventuelles maladies affectant les feuilles des plantes. Cette détection repose sur un modèle d'intelligence artificielle "YOLOv11", entraîné sur une base de données d'images de feuilles saines et malades. Le modèle est capable d'identifier visuellement divers symptômes comme les taches, les décolorations ou les déformations.

Une fois une anomalie détectée, le système ne se contente pas de signaler le problème : il propose également des solutions adaptées en fonction du type de maladie reconnu.

Ce système d'analyse intelligent permet ainsi d'aider les agriculteurs à prendre des décisions rapides et précises, tout en réduisant les pertes liées aux maladies végétales.

3.5 Choix du modèle d'intelligence artificielle : YOLOv11

Pour la détection des maladies sur les feuilles des plantes à partir des images capturées par le robot, nous avons opté pour le modèle YOLOv11 (You Only Look Once, version 8), proposé par Ultralytics. Ce modèle de type CNN (Convolutional Neural Network) est spécialisé dans la détection d'objets en temps réel et s'est révélé particulièrement efficace pour les applications agricoles nécessitant rapidité et précision.

3.5.1 Raisons du choix de YOLOv11

Plusieurs critères ont guidé notre décision :

· Vitesse d'inférence élevée, adaptée à l'analyse en temps réel sur un robot mobile.

· Taille du modèle optimisée, facilitant l'exécution même sur des plateformes embarquées.

· Excellente précision de détection, même pour des objets de petite taille comme les lésions sur les feuilles.

· Simplicité d'implémentation grâce à l'API Ultralytics et la compatibilité avec PyTorch.

3.5.2 Performances du modèle

Nous avons entraîné le modèle YOLOv11 sur un dataset d'images de feuilles saines et malades. L'évaluation sur un ensemble de test contenant 5033 images a donné les résultats suivants :

· Précision : 95.6 %

· Rappel : 92.9 %

· mAP@0.5 : 97.3 %

· mAP@0.5 :0.95 : 83.0 %

Ces résultats confirment l'excellente capacité du modèle à localiser et classifier les signes de maladies végétales.

3.5.3 Comparaison avec d'autres modèles

Nous avons également comparé YOLOv11 à d'autres modèles populaires :

Modèle	**mAP@0.5**	**Vitesse (ms/image)**	**Points forts**
YOLOv11	97.3%	6.7 ms	Précis, rapide, léger
YOLOv5	95.1%	8.3 ms	Bonne précision, mais plus lent
SSD MobileNet	88.2%	12.4 ms	Léger, mais moins précis
Faster R-CNN	91.5%	67.8 ms	Très précis, mais trop lent

Tableau 3.7 : Tableau comparatif de YOLOv11 avec d'autres modèles

YOLOv11 s'est imposé comme le meilleur compromis entre performance, précision et rapidité pour notre cas d'usage en environnement agricole.

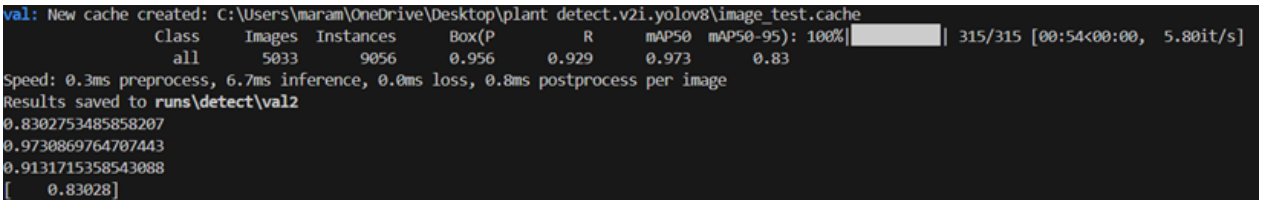

Figure 3.29 : La précision de modèle IA

3.6 Présentation de quelques interfaces de l'application

3.6.1 Logo de l'application

Figure 3.30 : logo de l'application

3.6.2 Interface de choix de type d'utilisateur

Figure 3.31 : Interface de choix de type d'utilisateur

Si c'est un client, il accède uniquement à l'interface du marché, où il peut consulter et acheter les produits disponibles

3.6.3 Interface de marché agricole pour les clients

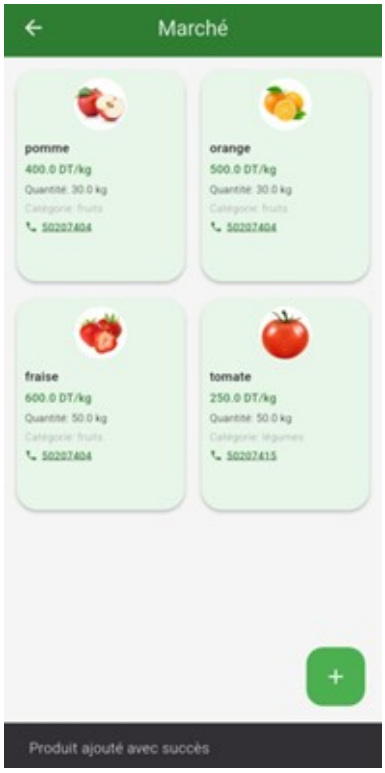

Figure 3.32 : Interface de marché agricole pour les clients

3.6.4 Interface d'authentification pour l'agriculteur

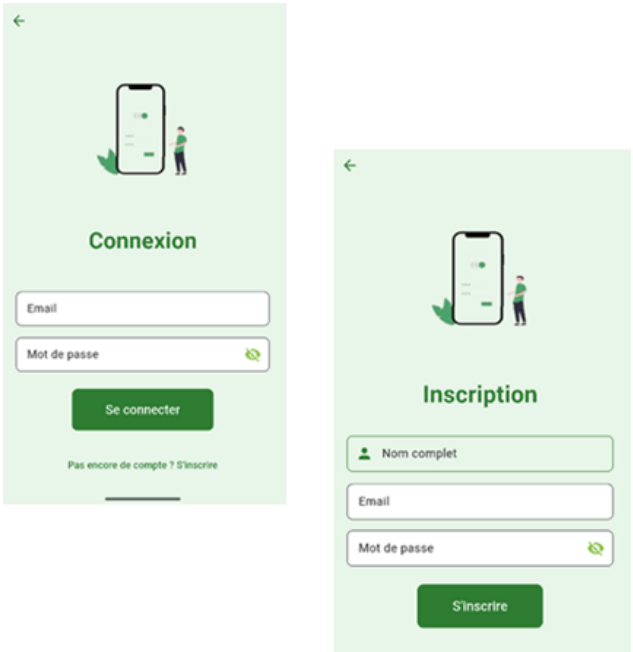

Figure 3.33 : Interface d'authentification pour l'agriculteur

3.6.5 Interface de l'Écran d'accueil

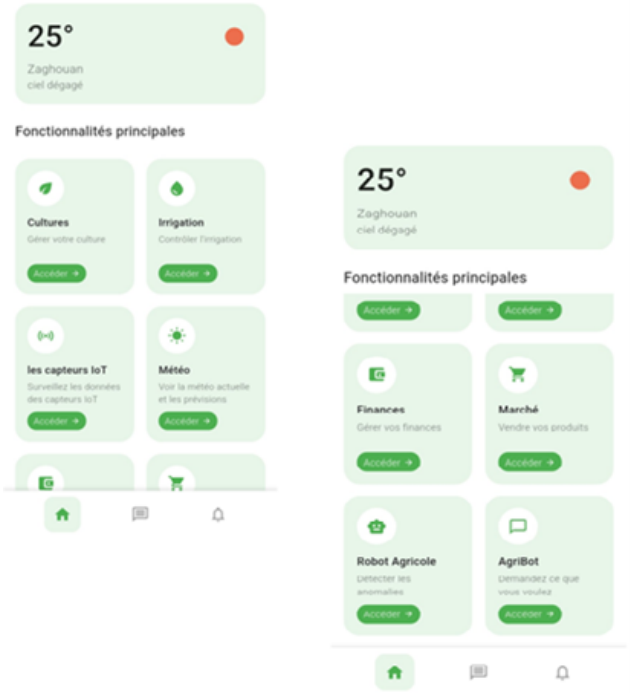

Figure 3.34 : Interface de l'Écran d'accueil

3.6.6 Interface de la gestion des cultures

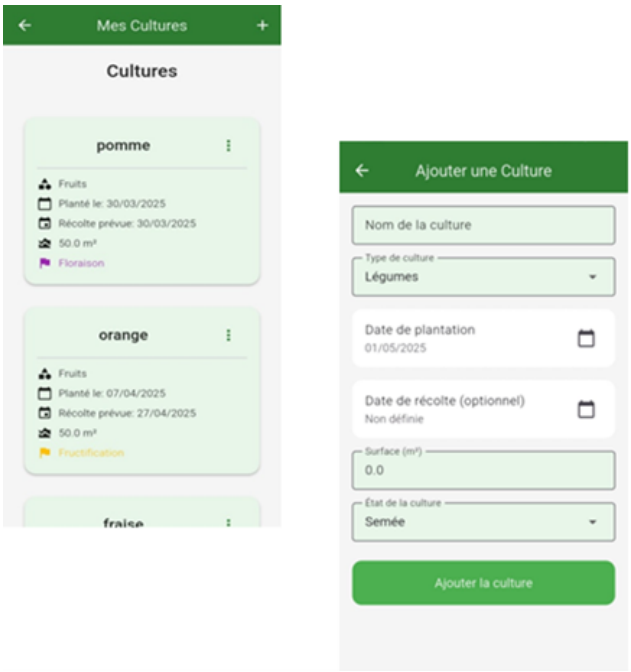

Figure 3.35 : Interface de la gestion des cultures

3.6.7 Interface du contrôle de l'irrigation manuel

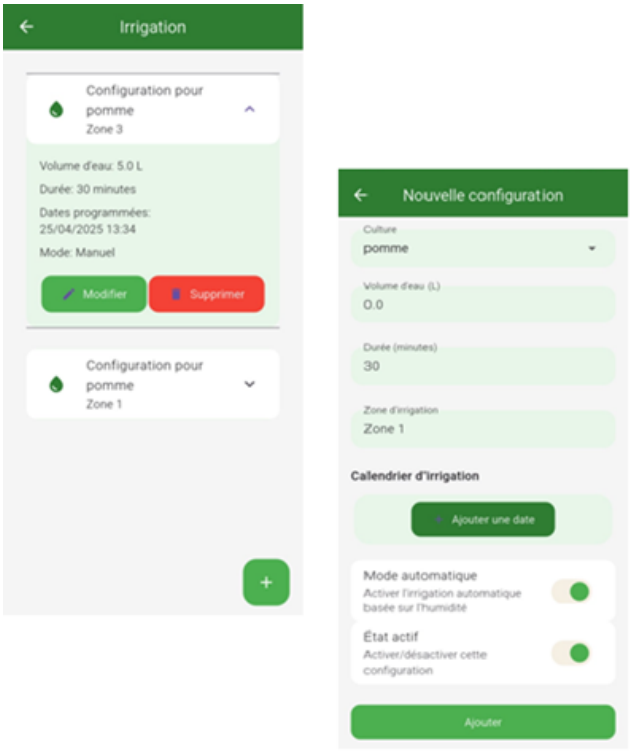

Figure 3.36 : Interface du contrôle manuel de l'irrigation

3.6.8 Interface de l'irrigation automatique

Figure 3.37 : Interface de l'irrigation automatique

3.6.9 Interface de la météo

Figure 3.38 : Interface de la météo

3.6.10 Interface du dashboard

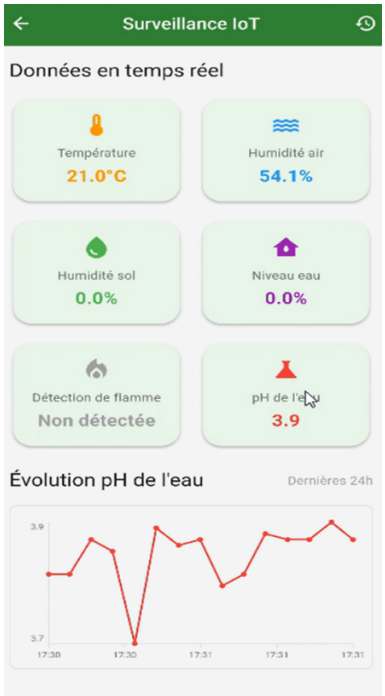

Figure 3.39 : Interface du dashboard

3.6.11 Interface de l'ajout d'un produit au marché

Figure 3.40 : Interface de l'ajout d'un produit au marché

3.6.12 Interface de la gestion financière

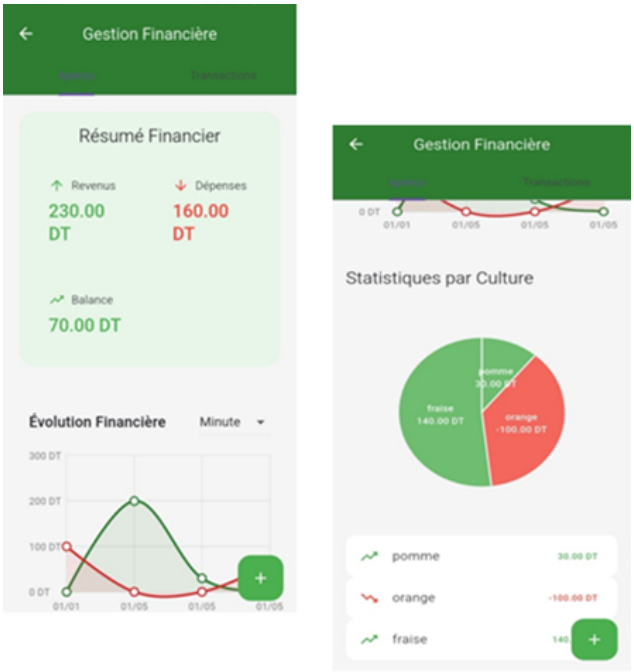

Figure 3.41 : Interface de la gestion financière

3.6.13 Interface de contrôle de robot

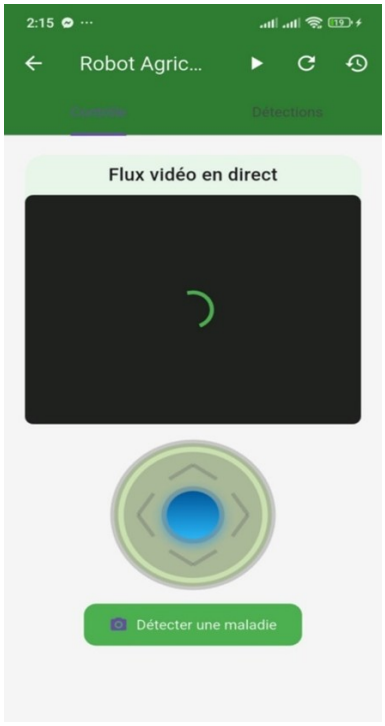

Figure 3.42 : Interface de contrôle de robot agricole

3.6.14 Interface de la détection des maladies

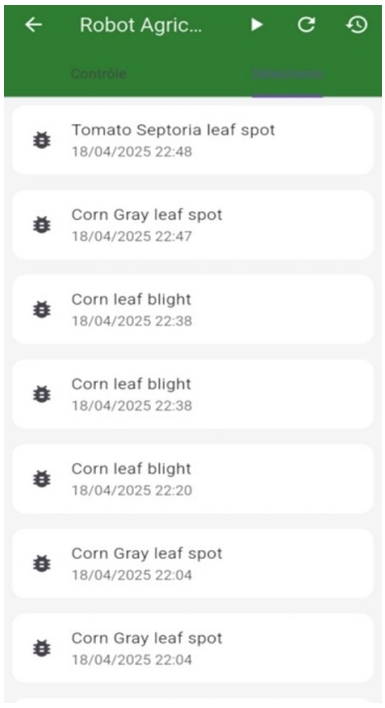

Figure 3.43 : Interface de la détection des maladies

3.6.15 Interface de l'assistant virtuel « AgriBot »

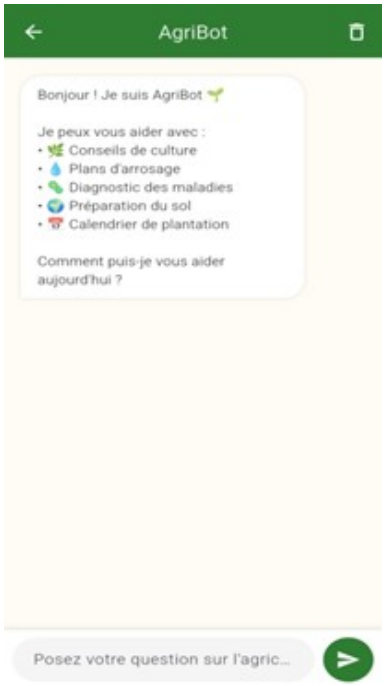

Figure 3.44 : Interface de l'AgriBot

3.7 Conclusion

Ce chapitre a permis de présenter les environnements matériels, logiciels et techniques utilisés, de détailler le montage des différents composants du système, de comparer les choix technologiques, d'expliquer la réalisation du robot agricole ainsi que le choix du modèle d'intelligence artificielle, et enfin d'illustrer les interfaces principales de l'application développée.

Conclusion générale

Aujourd'hui, l'intelligence artificielle (IA), l'Internet des Objets (IoT) et le développement mobile jouent un rôle clé dans la modernisation de nombreux secteurs, notamment l'agriculture. Ces technologies permettent de créer des solutions intelligentes, automatisées et accessibles, capables de répondre aux besoins croissants de performance, de durabilité et de précision dans la gestion des exploitations agricoles.

Dans ce cadre, notre projet vise à intégrer ces outils pour concevoir une application mobile innovante facilitant la gestion de la ferme, un robot agricole capable de détecter certaines maladies, ainsi que des capteurs IoT destinés à collecter des données environnementales telles que l'humidité, la température ou encore la qualité du sol. Ces données sont ensuite analysées afin de permettre une meilleure prise de décision, notamment pour optimiser l'irrigation grâce à un système intelligent, évitant ainsi le gaspillage d'eau. L'ensemble contribue à améliorer l'efficacité des interventions sur le terrain et à promouvoir une agriculture plus durable et connectée.

Tout au long de ce travail, nous avons mobilisé des compétences en systèmes embarqués, en développement mobile avec Flutter, en communication entre dispositifs IoT, ainsi qu'en conception logicielle. Le système réalisé représente une application concrète de nos acquis universitaires et un pas vers une agriculture plus innovante et connectée.

Enfin, afin de mieux répondre aux besoins réels des utilisateurs finaux, nous envisageons des améliorations futures, notamment l'ajout de nouvelles fonctionnalités, l'optimisation des performances du système, et la prise en charge multilingue de l'application, y compris l'intégration de l'arabe tunisien **"derja"**, pour faciliter l'utilisation et la compréhension par les agriculteurs locaux.