

RÉPUBLIQUE TUNISIENNE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ TUNIS EL MANAR



FACULTÉ DES SCIENCES DE TUNIS

DÉPARTEMENT DES SCIENCES DE L'INFORMATIQUE



MISE EN PLACE
D'UN SYSTÈME D'AIDE À LA
DÉCISION EN IRRIGATION



Présenté par :

M. Hamdi MOHAMED ALAA
M. Bennouri IHEB

Encadrants :

Pr. Fathallah KARIM
Pr. Ben Salem KAMEL

Remerciements

En premier lieu, Louange à ALLAH, le tout-puissant, pour l'éclaircissement de notre savoir. Nous remercions et nous témoignons toute notre reconnaissance à notre encadrant côté université Monsieur Ben Salem Kamel, enseignant à la Faculté des Sciences de Tunis, pour son encadrement, sa disponibilité et ses conseils précieux.

Nous souhaitons exprimer nos sincères gratitude particulièrement envers notre encadrant côté société Monsieur Fathalah Karim, pour son soutien, ses conseils judicieux et ses critiques constructives qui nous ont dirigé tout au long de ce stage et qui nous ont aidé pour bien mener à ce projet.

Je souhaite également remercier les membres du jury pour avoir bien voulu évaluer notre travail. Nous remercions ainsi et enfin tous ceux qui ont contribué à la réalisation de ce travail.

Un salut spécial à nos chers parents, nos frère, nos amis, nos proches et nos collègues pour nous avoir bien soutenu et aidé tout au long du projet.

Tableau des matiers

I	Cadre du projet et état de l'art	10
1	Irrigation de précision	10
1.1	Estimation des besoins d'irrigation	10
1.2	Pilotage de l'irrigation	16
2	Réseau de capteurs sans fils (rcsf)	17
2.1	Historique	17
2.2	Architecture du réseau	18
2.2.A	Topologie du réseau	18
2.2.B	Anatomie du noeud	19
2.3	Utilisation des RCSF dans l'irrigation	19
2.4	RCSF et l'internet	20
3	Problématique et démarche de travail	20
3.1	Motivation	20
3.2	Existant et sa limitation	20
3.3	Objectif	21
3.4	Méthodologie de travail	21
4	Conclusion	22
II	Spécification des besoins fonctionnels et non fonctionnels	23
1	Spécification des besoins fonctionnels	23
1.1	Diagramme de cas d'utilisation	23
1.2	Diagramme de cas d'utilisation Général	24
1.3	Description des diagrammes de cas d'utilisations	25
1.3.A	Authentification	25
1.3.B	Gestion des utilisateurs	26
1.3.C	Gestion des projets	27
1.3.D	Gestion des sols	28
1.3.E	Gestion des plantes	29
1.3.F	Gestion des noeuds	30
1.3.G	Gestion des données climatiques	31
2	Spécification des besoins non fonctionnels	32
2.1	Python	32
2.2	Django	33
2.3	PostgreSQL	34
2.4	Google Map	35
2.5	Llibelium	36

2.5.A	La carte Wasp mote	36
2.5.B	Le module Xbee	38
2.5.C	Shield Wasp mote de capteurs d'agriculture	38
2.5.D	Gateway Meshlium	39
3	Conclusion	40
III	Conception	41
1	Choix du langage de conception	41
2	Conception détaillée	41
2.1	Diagrammes de séquence	41
2.1.A	Client : Inscription	42
2.1.B	Client ou administrateur : Authentification	43
2.1.C	Client : Choix du type de projet	44
2.1.D	Client : Création du projet de type web service climatique	44
2.1.E	Client : Création du projet de type nœud(s) capteur(s) dans une parcelle	46
2.1.F	Client : Création du projet de type nœud(s) capteur(s) dans in pot	47
2.1.G	Administrateur : Création du projet	48
2.1.H	Administrateur : Gestion des (clients - plantes - sols - projets - polygones - nœuds - climat journalier)	49
2.2	Diagramme de classe	50
IV	Réalisation du projet	51
1	Architecture mise en place	51
1.1	Architecture générale	51
1.2	Architecture MTV	52
2	Environnement Matériel	53
3	Environnement logiciel	54
3.1	L'installation de l'environnement Django 1.10	54
3.2	Création et configuration du serveur PostgreSQL 9.6 et PostGIS	56
3.3	Création de la base de données	56
3.4	Intégration des services de OpenWeather	57
3.5	Configuration et intégration de réseau capteur	57
3.5.A	Environnement de programmation	57
3.5.B	Configuration et tests du Meshlium	58
3.5.C	Organigramme du fonctionnement du carte waspmote	60
4	Démonstration de la réalisation de l'application	61
4.1	Interfaces : Authentification	61
4.2	Interface : Inscription	63
4.3	Interface : Liste des projets	64
4.4	Interfaces : Création de projet	65
4.4.A	Interface : Choix de type de projet	65
4.4.B	Interface : Sélectionnement des parcelles	66
4.4.C	Interface : Choix de type de plante	67
4.4.D	Interface : Choix de type de sol	68
4.4.E	Interface : Positionnement des nœuds	69

4.5	Interface : projet	70
4.6	Interfaces : Administrateur	71
4.6.A	Interface : Les tableaux	71
4.6.B	Interface : Gestion des données	72
4.7	Interfaces : Les consoles des serveurs	74
4.7.A	Interface : Serveur Django	74
4.7.B	Interface : Serveur rempir_journal	75
4.7.C	Interface : Serveur planning_owm	76
5	Conclusion	76
A	Partie du code de la carte Wasp mote	80

Liste des figures

1.1	Le stade de développement du plante	13
1.2	Classification des textures du sol	15
1.3	Les systèmes d'irrigation par aspersion	16
1.4	Disposition d'irrigation goutte à goutte	17
1.5	Evolution des capteurs	18
1.6	Diagramme de GANTT	21
2.1	Diagramme de cas d'utilisation Général	24
2.2	Diagramme des cas d'utilisations - Authentification	25
2.3	Diagramme des cas d'utilisations - Gestion des utilisateurs	26
2.4	Diagramme des cas d'utilisations - Gestion des projets	27
2.5	Diagramme des cas d'utilisations - Gestion des sols	28
2.6	Diagramme des cas d'utilisations - Gestion des plantes	29
2.7	Diagramme des cas d'utilisations - Gestion des noeuds	30
2.8	Diagramme des cas d'utilisations - Gestion des données climatiques	31
2.9	Icône - Python	32
2.10	Classement des langages de programmation	32
2.11	Icône - Django	33
2.12	Icône - PostgreSQL	34
2.13	Icône - Google Map	35
2.14	La carte Wasp mote	37
2.15	Composant xbee	38
2.16	les différents capteurs climatiques	39
2.17	Fonctionnement du Meshlium	40
3.1	Description des diagrammes de séquences - Inscription	42
3.2	Description des diagrammes de séquences - Authentification	43
3.3	Description des diagrammes de séquences - Choix du type de projet	44
3.4	Description des diagrammes de séquences - Création du projet de type web service climatique	45
3.5	Description des diagrammes de séquences - Création du projet de type noeud(s) capteur(s) dans une parcelle	46
3.6	Description des diagrammes de séquences - Création du projet de type noeud(s) capteur(s) dans in pot	47
3.7	Description des diagrammes de séquences - Création du projet	48
3.8	Description des diagrammes de séquences - Gestion de l'application	49
3.9	Diagramme de classe	50

4.1	Architecture générale de l'application	52
4.2	Architecture MTV	53
4.3	Capture d'écran de l'installation du l'environnement Django	55
4.4	Icône - Environement de développement Wasp mote	57
4.5	Interface - Connexion sur le routeur Meshlium	58
4.6	Interface - Architecture adoptée : ZigBee GPRS AP	58
4.7	Statu - connecté et une adresse ip est affecté	59
4.8	Fonctionnement du carte waspmote	60
4.9	Connexion utilisateur	61
4.10	Connexion administrateur	62
4.11	Inscription de l'utilisateur	63
4.12	Interface du liste des projets	64
4.13	Interface de création de projet - Choix du type de projet	65
4.14	Interface de création de projet - Slectionner les parcelles	66
4.15	Interface de création de projet - Choix de type de plante	67
4.16	Interface de création de projet - Choix de type de sol	68
4.17	Interface de création de projet - Positionner les nœuds dans la parcelle	69
4.18	Interface du projet	70
4.19	Interface d'administrateur	71
4.20	Exemple d'interface de gestion des données dans le tableau Utilisateurs	72
4.21	Exemple d'interface permet d'ajouter un nouveau utilisateur dans le tableau Utilisateurs	73
4.22	Exemple d'interface permet de modifier les valeurs du utilisateur hamdi	73
4.23	Interface du console du serveur principale	74
4.24	Interface du console du serveur remplir_journal	75
4.25	Interface du console du serveur planning_owm	76

Liste des tableaux

2.1	Description des cas d'utilisations - Authentification	25
2.2	Description des cas d'utilisations - Gestion des utilisateurs	26
2.3	Description des cas d'utilisations - Gestion des projets	27
2.4	Description des cas d'utilisations - Gestion des sols	28
2.5	Description des cas d'utilisations - Gestion des plantes	29
2.6	Description des cas d'utilisations - Gestion des noeuds	30
2.7	Description des cas d'utilisations - Gestion des données climatiques	31

Introduction générale

Les agriculteurs d'aujourd'hui font face à un ensemble de défis : une demande mondiale croissante de nourriture, un climat changeant et une quantité limitée d'eau . Avec l'évolution de l'Internet des objets (IOT) qui s'est transformé en un concept initial en solutions matérielles et logicielles réelles, on voit émerger aujourd'hui de nombreuses applications logicielles dans le secteur agricole qui permettent une rationalisation de l'exploitation et une augmentation de la productivité.

En Tunisie, l'utilisation de l'IOT pour l'irrigation est devenue indisponible, pour cela nous cherchons des meilleures solutions afin d'améliorer la productivité et la qualité des produits agricoles avec le moindre gaspillage d'eau possible. Dans ce contexte, nous allons concevoir et mettre en place un système qui calcule le besoin réelle des plantes en irrigation (irrigation de précision ou irrigation intelligente).

Ce rapport présente une synthèse de notre projet de fin d'études pour couronner notre formation des Sciences de l'Informatique au sein de la Faculté des Sciences de Tunis. Ce rapport sera divisé en quatre chapitres.

Le premier chapitre intitulé "Cadre du projet et état de l'art" est consacré à la présentation de la notion d'irrigation intelligente ou de précision ainsi qu'une partie explicative du standard de calcul des besoins des plantes et les conditions pour réussir ces calculs.

Le deuxième chapitre intitulé "Analyse des besoins fonctionnels et non fonctionnels" comme son titre l'indique, est consacré à l'élaboration du cahier de charges et les choix techniques de notre projet.

Le troisième chapitre, "Conception", est dédié à la présentation des différents éléments conceptionnels du projet.

Dans le dernier chapitre, intitulé "Réalisation", nous exposons l'environnement matériel et logiciel. Ensuite, nous présentons les différentes étapes d'implémentation de notre solution accompagnées des tests et des captures d'écran de notre application.

Chapitre I

Cadre du projet et état de l'art

Ce travail est réalisé dans le cadre de notre projet de fin d'études, le travail est venu à la suite d'une proposition faite de la part de l'entrepris(..)qui est une petite startup qui vient de se lancer à la ville Kébili et dont l'un de ses objectifs cibles est de réaliser des applications à base de l'informatique et de l'internet des objets afin d'améliorer l'utilisation en eau d'irrigation dans les oasis de palmier dattier.

L'objectif de ce chapitre est de définir les concepts de base en relation avec notre application ainsi que les objectifs à réaliser dans ce projet.

1 Irrigation de précision

1.1 Estimation des besoins d'irrigation

L'irrigation de précision est un nouveau concept mondial dans le domaine d'irrigation qui implique l'application exacte et précise de l'eau pour répondre aux besoins spécifiques et particuliers de chaque plante.

Plusieurs techniques ont été développées dans ce but, parmi lesquelles on peut citer la goutte à goutte au moyen de gouteurs et la micro-aspersion au moyen de diffuseurs et, de façon plus marginale, l'irrigation gravitaire. La consommation de l'eau est un élément clé pour la croissance de la plante. Il est évident donc que satisfaire ces besoins en eau est à la base d'une production optimale des champs agricole.

En effet les molécules d'eau sont consommées lors du processus de photosynthèse dans les feuilles de la plantes. Les facteurs affectant l'évapotranspiration d'une culture agricole sont groupé en trois catégories :

1. les facteurs climatiques affectant l'évapotranspiration est la radiation solaire, la température de l'air, l'humidité de l'air et la vitesse du vent.

2. Les facteurs liés à la cultures : le type de la culture, la variété dans l'espèce et la phase de développement influe énormément sur l'évapotranspiration d'une culture qui ne manque pas d'eau. En effet la forme des feuilles, la hauteur des culture, le degré de verdure etc. influe sur l'interception de la plante de l'énergie solaire ainsi que le pouvoir évaporisateur du vent.
3. Les méthodes de gestion et les conditions environnementales : des facteurs telle que la salinité, la fertilité du sol, la couverture de la surface, les maladies, la pollution de l'air tout ces facteurs permettent une détérioration dans la croissance de la plante. Ceci s'exprime évidement par une limitation dans l'évapotranspiration de la plante.

La modélisation de ces paramètres permettent une estimation efficace de l'évapotranspiration réelles par la plante. Pour l'estimation de l'évapotranspiration quel que soit le climat, la FAO (Food and Agriculture Organization) à mis en place un standard international qui s'appelle le modèle Fao 56.

Ce modèle est actuellement le plus utilisé par les agronomes du monde entier afin d'estimer d'une manière précise les besoin en eau d'une culture agricole. Ce dernier permet d'estimer l'évapotranspiration par heure, par journée, décadaire et mensuelle. Cette estimation se fait en fonction des données climatiques, des caractéristiques de la plante et en fonction des aspects liée à la gestion de la culture. En pratique ce modèle peut être exprimé par l'équation suivante :

$$ET = Kc \times ET_0$$

Où :

ET : évapotranspiration réelle,

Kc : coefficient lié à la plante elle-même, sa forme, son niveau d'évolution, etc.

ET_0 : évapotranspiration de référence basée sur les paramètres climatiques.

Le premier terme dans le modèle d'évapotranspiration FAO 56 est l'évapotranspiration de référence ET_0 . L'équation de Penmann Monteith a été retenue comme l'unique standard pour l'estimation de la valeur de l'évapotranspiration de référence ET_0 .

Le modèle de Penmann Monteith permet d'estimer ET₀ en fonction de quatre paramètres climatiques : La température de l'air, l'humidité de l'air, l'énergie fournie par le soleil et la vitesse du vent. Cette relation est définie par l'équation ci-dessus pour l'évapotranspiration journalière :

$$ET_0 = \frac{0.408 \times \Delta \times (R_n - G) + \gamma \times \frac{900}{T+273} \times u_2 \times (e_s - e_a)}{\Delta + \gamma \times (1 + 0.34 \times u_2)}$$

Où :

ET_0 : évapotranspiration de référence (mm/h),

R_n : rayonnement net (W.m⁻²),

G : flux de chaleur du sol (W.m⁻²),

T : température moyenne de l'air (°C),

Δ : pente de la courbe de la tension de vapeur saturante (kPa.°C⁻¹),

γ : constante psychométrique (kPa.°C⁻¹),

e_s : tension de vapeur à la température t (kPa),

e_a : tension de vapeur saturante (kPa),

u_2 : vitesse moyenne du vent à 2 m (m.s⁻¹).

Dans le cas où quelques paramètres sont indisponibles, le modèle Fao 56 propose un ensemble d'équation permettant de rapprocher leurs valeurs en fonction d'autres paramètres.

Le deuxième terme dans le modèle d'évapotranspiration FAO 56 est le coefficient culture K_c. Ce coefficient permet de modéliser les caractéristiques qui différencient un type de culture particulière par rapport à celle de la culture de référence (on considère comme culture de référence le gazon). La principale caractéristique est le pouvoir évaporateur de l'eau relié aux nombres de stomates dans la plante et à leur caractéristique.

Pour une culture particulière ceci dépend de la phase de croissance de la plante. Le modèle FAO 56 définit quatre stades de croissance d'une culture :

- Stade initial
- Stade de développement
- Stade mi-saison
- Stade fin saison

Le coefficient cultural varie principalement suivant le type de culture et son stade de développement (voir la figure suivante).

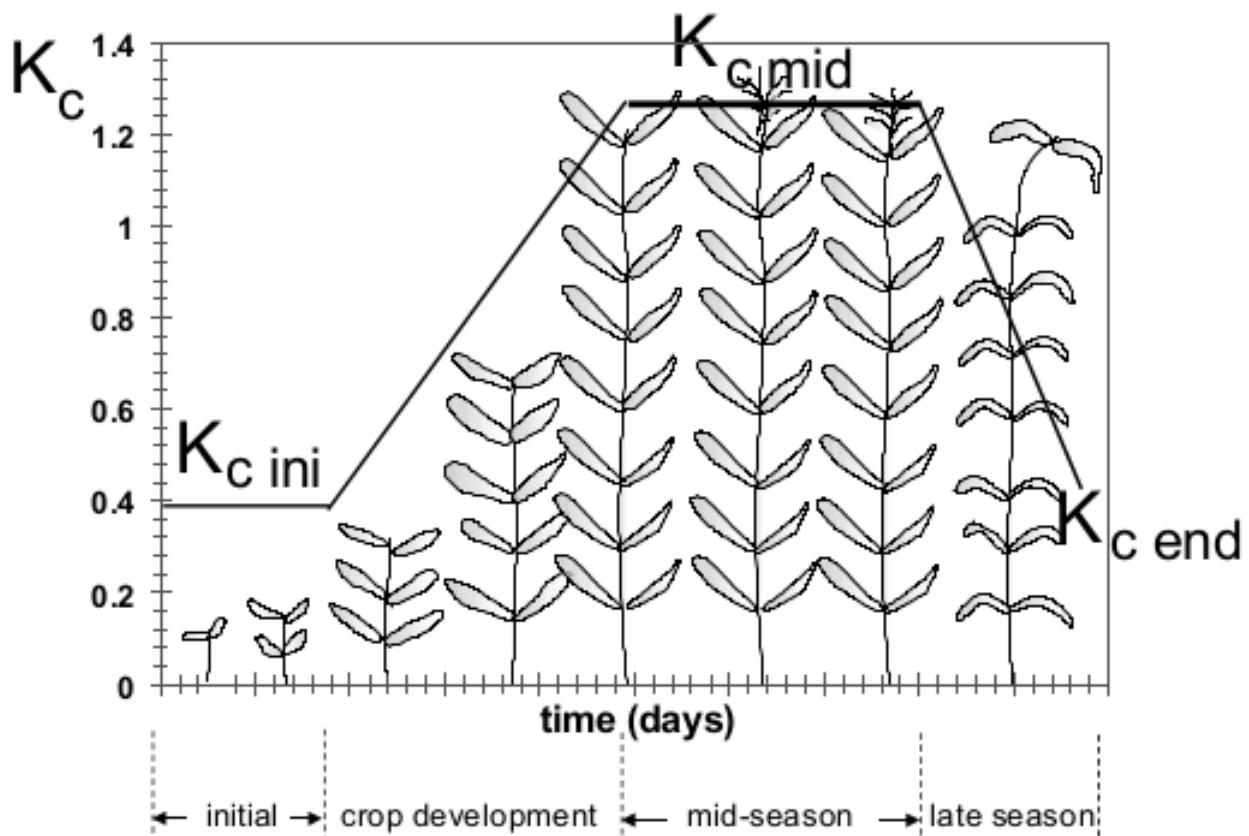


FIGURE 1.1 – Le stade de développement du plante

La variation temporelle du coefficient cultural indique que :

- Il est minimum au début du développement de la culture,
- Il augmente avec la croissance du végétal et l'augmentation du couvert végétal,
- Il passe par un maximum au moment du stade adulte pendant la phase de reproduction (floraison et formation du grain),
- Il décroît enfin.

La quantité d'eau nécessaire peut être estimée en appliquant l'équation suivante :

$$BP = BE - P - HS$$

Où :

BP : la quantité qu'on doit apporter pour la plante,

BE : les besoins de la plante après l'évapotranspiration,

P : pluie efficace (mm),

HS : les réserves d'eaux exploitables par la plante déjà disponible dans le sol.

Afin de pouvoir estimer exactement l'apport nécessaire en eau d'irrigation il faut pouvoir estimer les réserves déjà disponibles dans le sol. Parmi les quantités d'eau emmagasiné dans le sol une partie seulement est dite la réserve utile (RU). Le RU est par définition la quantité totale d'eau du sol utilisable par une culture. Elle dépend de la nature du sol mais aussi de la profondeur du sol colonisée par les racines et de la charge en cailloux.

La Réserve Utile dépend de la texture du sol et s'exprime en mm d'eau par mètre de profondeur de sol. La texture d'un sol correspond à la répartition des minéraux indépendamment de la nature et de la composition de ces minéraux. La texture du sol est divisée en trois types :

- sables : $> 50 \mu\text{m}$
- limons : de $50 \mu\text{m}$ à $2 \mu\text{m}$
- argiles : $< 2 \mu\text{m}$

Cette classification est représentée à l'aide d'un triangle, appelé triangle des textures (voir figure suivante).

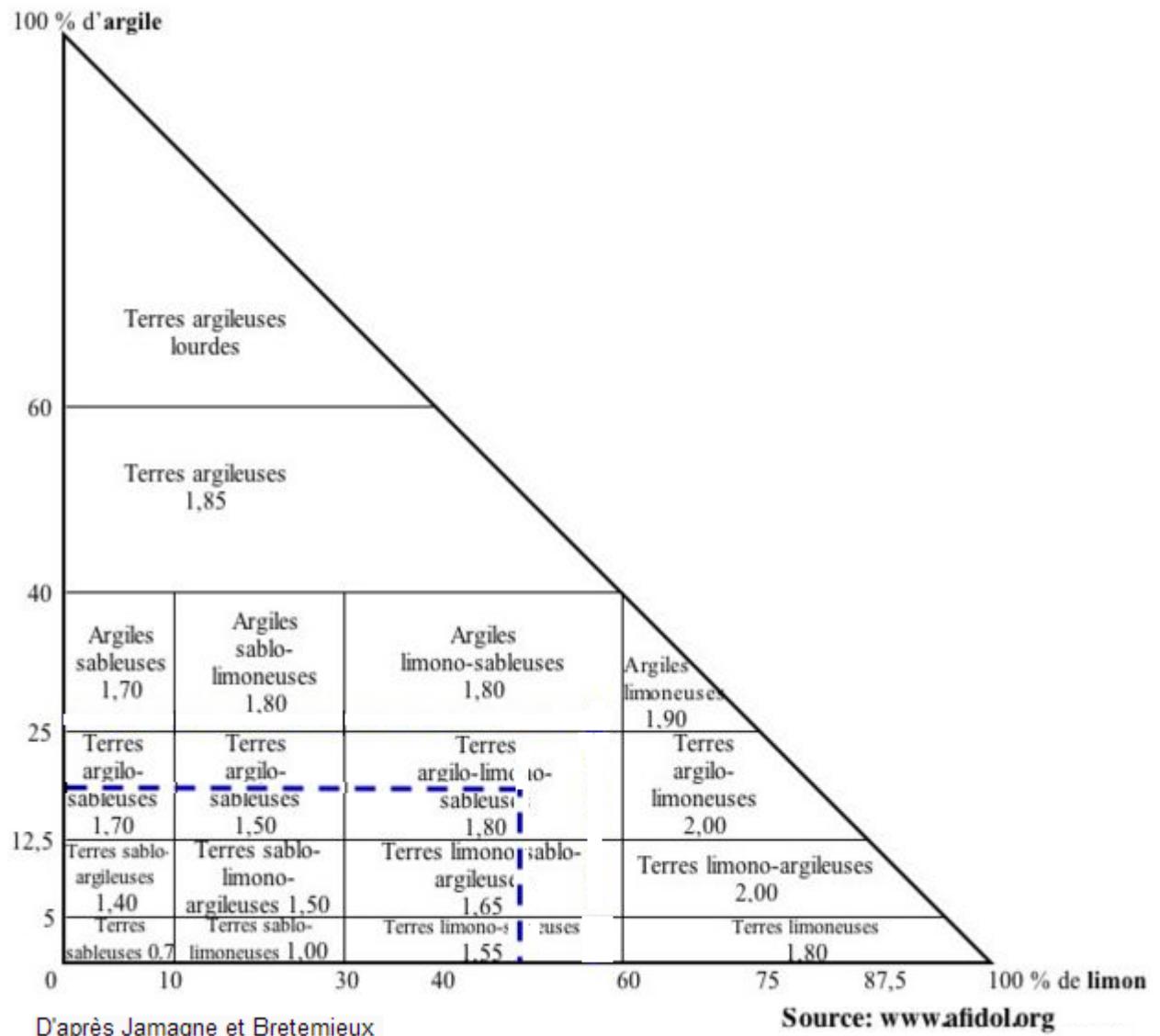


FIGURE 1.2 – Classification des textures du sol

Il est possible de regrouper les textures en quatre classes fondamentales, qui permettent de définir les principales propriétés du sol :

- texture sableuse : sol bien aéré, facile à travailler, pauvre en réserve d'eau, pauvre en éléments nutritifs, faible capacité d'échange anionique et cationique.
- texture limoneuse : l'excès de limon et l'insuffisance d'argile peuvent provoquer la formation d'une structure massive, accompagnée de mauvaises propriétés physiques. Cette tendance est corrigée par une teneur suffisante en humus et calcium.

- texture argileuse : sol chimiquement riche, mais à piètres propriétés physiques ; milieu imperméable et mal aéré, formant obstacle à la pénétration des racines ; travail du sol difficile, en raison de la forte plasticité (état humide), ou de la compacité (sol sec). Une bonne structure favorisée par l'humification corrige en partie ces propriétés défavorables.
- texture équilibrée : elle correspond à l'optimum, dans la mesure où elle présente la plupart des qualités de trois types précédents, sans en avoir les défauts.

1.2 Pilotage de l'irrigation

Le pilotage de l'irrigation, appelé encore conduite ou programmation des arrosages (irrigation scheduling en anglais), consiste à répondre à 3 questions essentielles : quand irriguer, quelle dose prévoir, comment l'apporter ? auxquelles il faut pouvoir répondre avec une précision maximale pour garantir un potentiel de production quantitatif et qualitatif à moindre coût en évitant tout apport d'eau intitule et nuisible.

Le pilotage de l'irrigation aura donc pour but, d'éviter le stress hydrique sur l'ensemble du cycle de la culture afin d'avoir une production maximale sans contrainte d'alimentation en eau, et l'optimisation de l'utilisation de l'eau quand les ressources sont limitées. Les modes d'irrigation proposés :

- L'irrigation par aspersion qui est une technique d'irrigation par laquelle l'eau est apporté aux plantes sous la forme d'une pluie artificielle.



FIGURE 1.3 – Les systèmes d'irrigation par aspersion

- L'irrigation goutte à goutte qui présente des dispositifs apportant l'eau de façon ponctuelle à des faibles débits (2 à 12 litres/h).

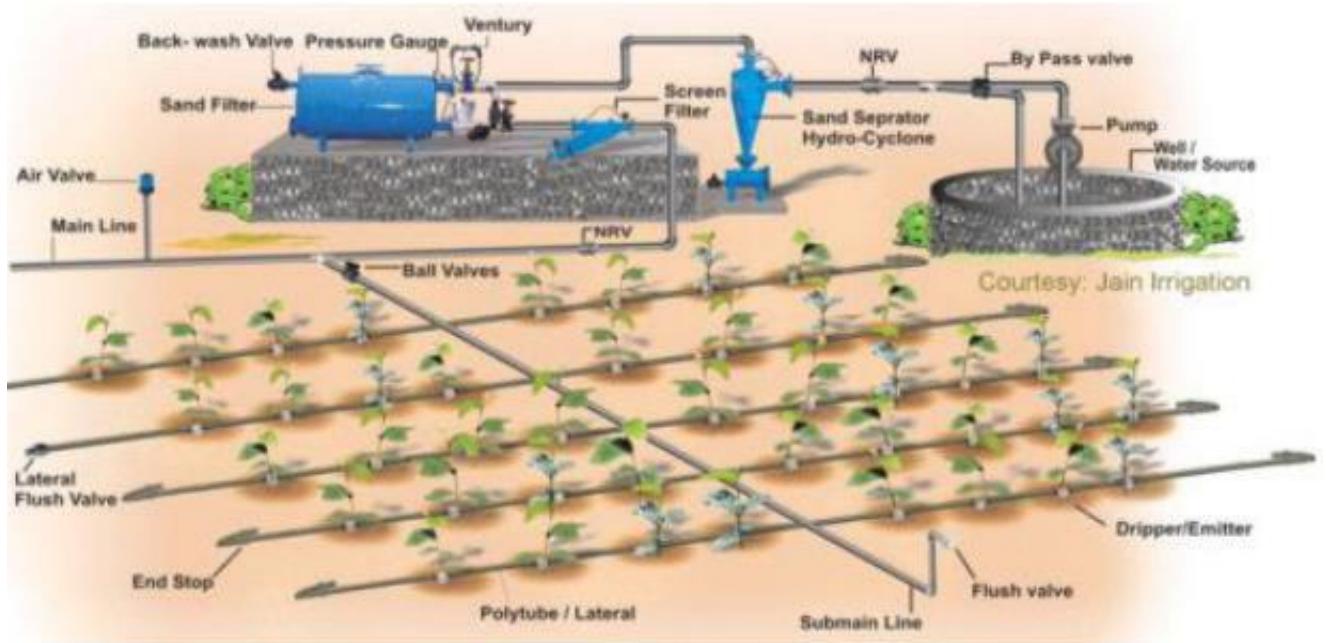


FIGURE 1.4 – Disposition d'irrigation goutte à goutte

Nous dépendons ici sur les électrovannes comme moyens d'arrosage qui représentent des robinets commandés électriquement par le programmeur.

2 Réseau de capteurs sans fils (rcsf)

2.1 Historique

A partir des années 90, l'utilisation des RCSF (réseau de capteurs sans fil) connaît un grand essor dans des domaines variés tel que le secteur militaire, la médecine, l'environnement, etc. L'un des domaines d'application prometteurs des RCSF est l'agriculture de précision où la technologie des RCSF offre un support important qui permettra la gestion précise des ressources, l'optimisation de l'irrigation, etc.

Un réseau de capteurs sans fil est un réseau composé d'un ensemble de nœuds (capteurs). Ces derniers sont des objets électroniques de taille extrêmement réduite avec des ressources très limitées, autonomes, capable de traiter des informations et de les transmettre, via les ondes radio, à une autre entité (capteurs, unité de traitements...) sur une distance limitée à quelques mètres.



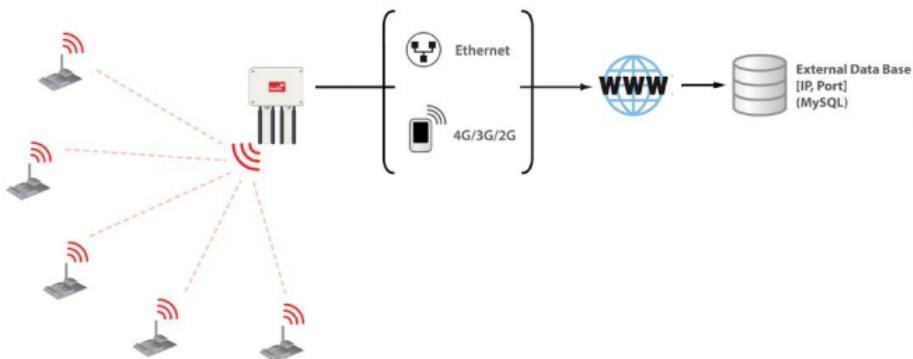
FIGURE 1.5 – Evolution des capteurs

2.2 Architecture du réseau

Un système composé de stations de capteurs répartis à travers le champ, d'une station de contrôle de l'irrigation, et une station de base. Les stations de champ capteurs surveillent les conditions sur le terrain de l'humidité du sol, la température du sol et la température de l'air, tandis qu'une station météorologique proche surveille l'information micro-météorologique sur le terrain, c'est à dire, la température de l'air, l'humidité relative, précipitations, vitesse du vent, direction du vent, et le rayonnement solaire. Toutes les données de champ sensorielles sont transmises sans fil à la station de base. La station de base traite les données de champ sensorielles grâce à un programme de décision convivial et envoie des commandes de contrôle à la station de contrôle de l'irrigation, la station de base se nourrit des signaux de commande à la station de contrôle de l'irrigation pour fonctionner site spécifiquement chaque groupe d'arrosage individuel d'appliquer une profondeur d'eau donnée.

2.2.A Topologie du réseau

L'architecture de réseau adoptée est l'architecture en étoile qui est composé d'un coordinateur du réseau de capteurs sans fil et d'une pluralité de nœuds capteurs sans fil dont tous les nœuds envoient et reçoivent seulement des données avec la station de base. Cette topologie est simple et elle demande une faible consommation d'énergie.



2.2.B Anatomie du noeud

Un noeud capteur contient quatre unités de base : l’unité de captage, l’unité de traitement, l’unité de transmission, et l’unité de contrôle d’énergie.

- l’unité de traitement : c’est l’unité principale du capteur. Elle est composée d’un microcontrôleur et d’une mémoire, son rôle est de contrôler le bon fonctionnement des autres unités. Cette unité est chargée aussi d’exécuter les protocoles de communications qui permettent de faire collaborer un capteur avec d’autres capteurs.
- l’unité de captage : est composée d’un dispositif qui va obtenir des mesures analogiques sur les paramètres environnementaux et d’un convertisseur Analogique/Numérique qui transformer les signaux analogiques en numériques afin de pouvoir être traiter par l’unité de traitement.
- l’unité de transmission : est responsable de toutes les émissions et réceptions de données sur un medium sans fil.
- l’unité d’alimentation : c’est la batterie qui, n’est généralement ni rechargeable ni remplaçable. L’énergie limitée des capteurs représente la contrainte principale lors de conception de protocoles pour les RCSF. Les unités d’énergie peuvent être supportées par des photopiles qui permettent de convertir l’énergie lumineuse en courant électrique.

2.3 Utilisation des RCSF dans l’irrigation

Ces capteurs nous donnent les informations nécessaires pour estimer les besoins de la plante et nous permet de contrôler la quantité d’eau dans le sol et de surveiller les quantités d’eau irrigués par les électrovannes.

2.4 RCSF et l'internet

C'est le domaine IOT (internet des objets), où nous sommes entraînés à transmettre les flux de données du réseau de capteurs sur le réseau internet afin d'avoir une application de supervision à distance et qui permettent de contrôler les composants de champs agricoles.

3 Problématique et démarche de travail

3.1 Motivation

Dans ces années en Tunisie on connaît une période de sécheresse due au mauvais usage des ressources hydriques, d'un assainissement déficient et surtout au phénomène du changement climatique.

La Tunisie reçoit des précipitations qui balancent entre 1500 mm dans le nord et moins de 50 mm dans le sud, soit une moyenne annuelle de 36 109 m³. Ce volume peut atteindre 90 109 m³/an, pendant année fortement pluvieuse et se limite à 11 109 m³, pendant année de sécheresse.

Les $\frac{3}{4}$ du territoire de la Tunisie sont semi-arides ou arides, car localisés entre la mer méditerranée et le Sahara. En plus de la variabilité du climat méditerranéen, l'influence de cette aridité fait de l'eau une ressource rarissime ayant une répartition temporelle irrégulière et spatiale inégale.

Donc un nouveau besoin a été né dont développer des applications dans les domaines agricoles pour permettre une consommation optimale de l'eau. Dans ce cadre on se propose de développer une application informatique qui utilise les technologies de l'internet des objets afin de participer à résoudre ce problème.

3.2 Existant et sa limitation

Ici, nous allons lister les applications existantes et les besoins qu'elles couvrent :

- l'Institut National des Grandes Cultures a mis au point une application mobile gratuite nommée IRREY. Cette application permet aux agriculteurs de maîtriser le processus d'irrigation pour optimiser l'utilisation de l'eau en fonction de la nature de la culture , du sol et du climat .
- Aussi nous avons l'application "MABIA-Region" introduite par l'Institut National Agronomique de Tunisie. Ce logiciel facilite le management de l'irrigation de l'eau.

Ces applications font l'estimation sur des données historiques donc manque de temps réel. Ils sont peu développés par rapport à l'étranger et ces solutions sont coûteuses.

3.3 Objectif

L'objectif est de développer un système web accessible à distance qui permet une utilisation facile et gratuite pour les fermiers dans l'estimation exacte de leurs besoins en eaux d'irrigation. Les fonctions permises aux utilisateurs sont :

- Sélectionner une parcelle
- Sélectionner un nœud sur la parcelle
- Examiner les besoins d'eaux estimés pour la parcelle et les conditions climatiques toute au long du jour
- Suivant un planning spécifique, être notifié par un email

3.4 Méthodologie de travail

- Diagramme de GANTT : En vue de modéliser les différentes étapes constituant le projet, nous avons préparé le diagramme de GANTT (voir figure suivante).

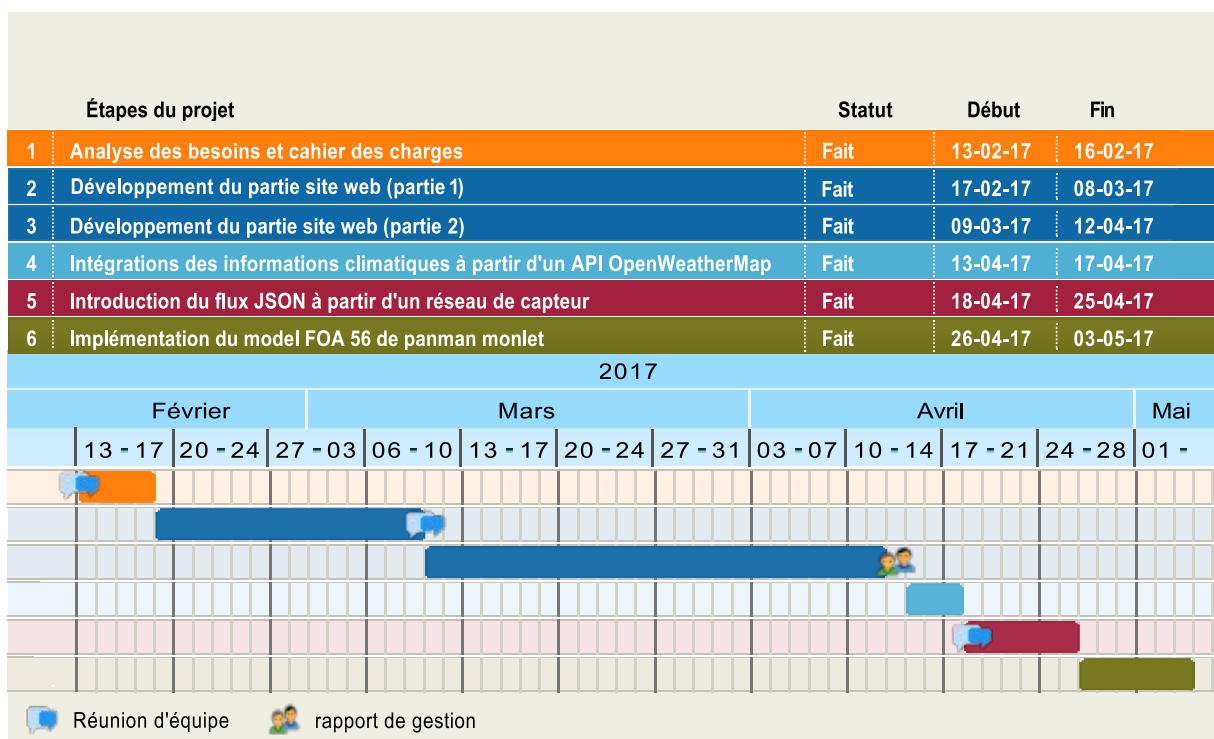


FIGURE 1.6 – Diagramme de GANTT

4 Conclusion

Dans ce chapitre on a défini les contextes et les concepts de base en relation avec les réseaux de capteurs utilisé avec l’irrigation intelligente. Dans le chapitre suivant, nous allons concentrer plus sur les besoins fonctionnels et non fonctionnels de notre application.

Chapitre II

Spécification des besoins fonctionnels et non fonctionnels

Dans ce chapitre, Nous allons identifier les fonctionnalités que doit fournir notre application, ainsi que les différents cas d'utilisation de notre projet.

1 Spécification des besoins fonctionnels

La maîtrise d'ouvrage et les utilisateurs ne sont pas des informaticiens. Il leur faut donc un moyen simple d'exprimer leurs besoins. C'est précisément le rôle des diagrammes de cas d'utilisation qui permettent de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Il s'agit donc de la première étape UML d'analyse d'un système.

1.1 Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique.

1.2 Diagramme de cas d'utilisation Général

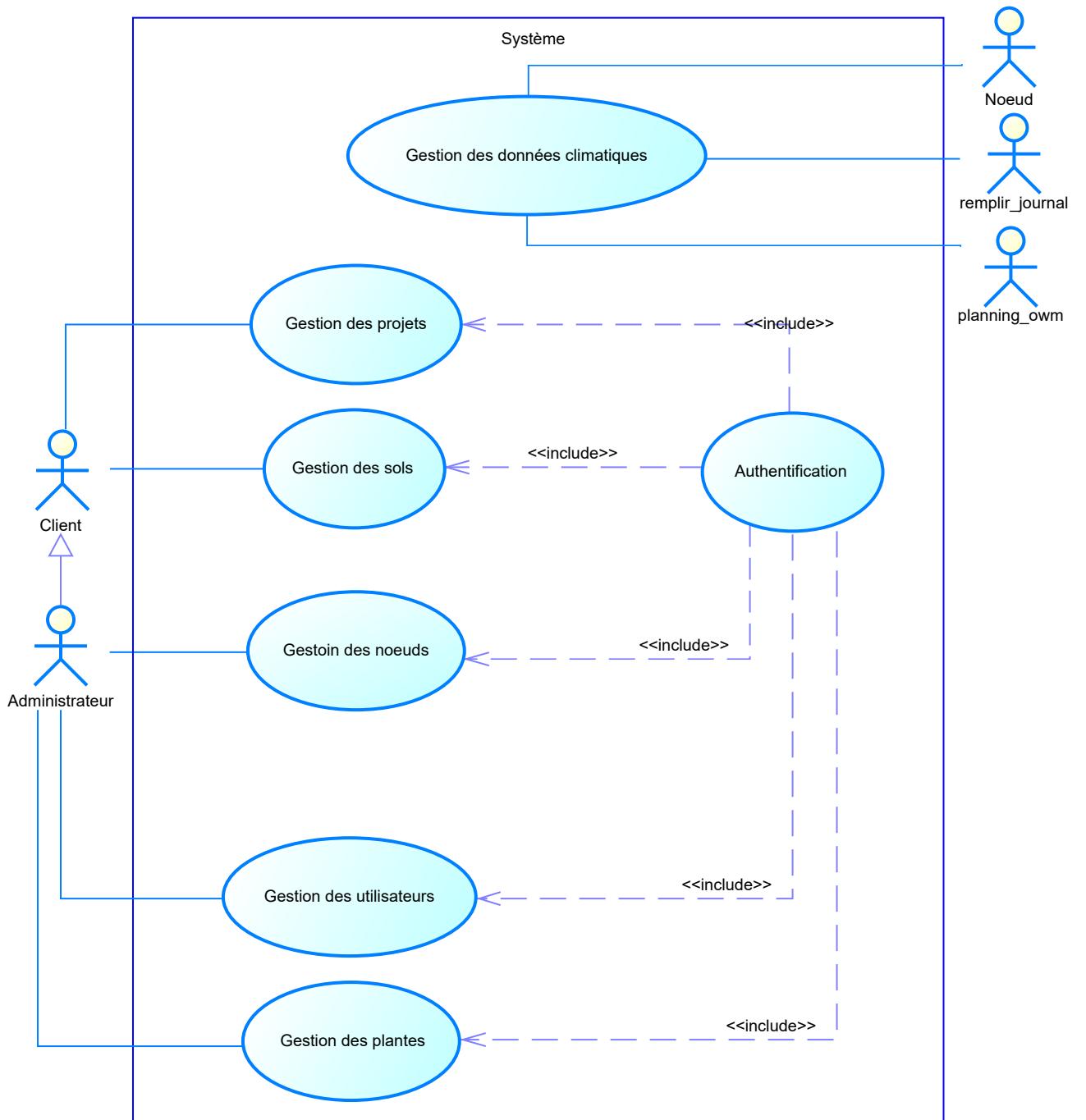


FIGURE 2.1 – Diagramme de cas d'utilisation Général

1.3 Description des diagrammes de cas d'utilisations

1.3.A Authentification

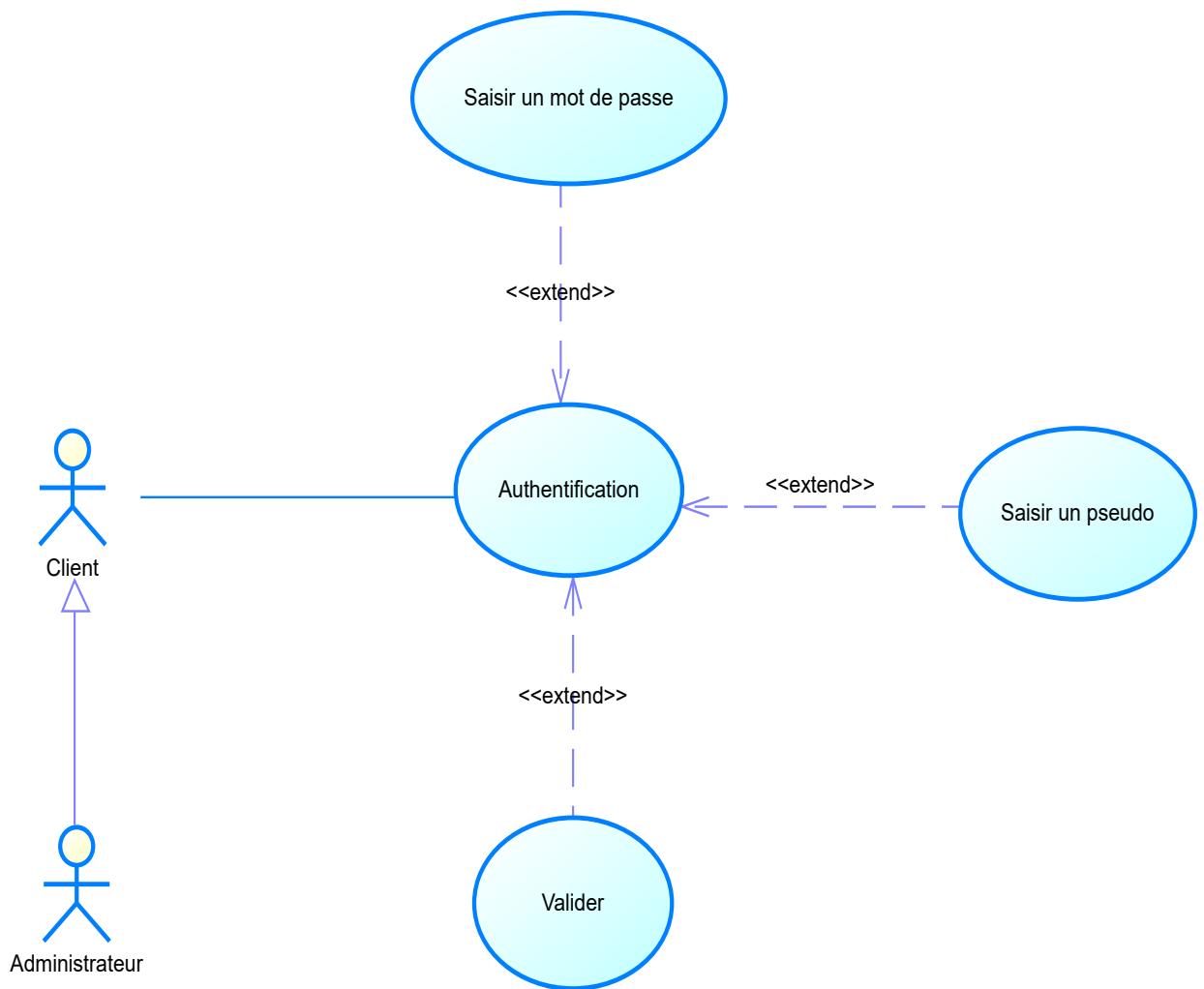


FIGURE 2.2 – Diagramme des cas d'utilisations - Authentification

Nom de cas	Authentification
Acteur	Utilisateur, Administrateur.
Pré condition	-
Poste condition	Authentification réussite.
Scénario principale	L'utilisateur doit entrer son pseudo et mot de passe pour avoir accès à la fonctionnalité du site.

TABLEAU 2.1 – Description des cas d'utilisations - Authentification

1.3.B Gestion des utilisateurs

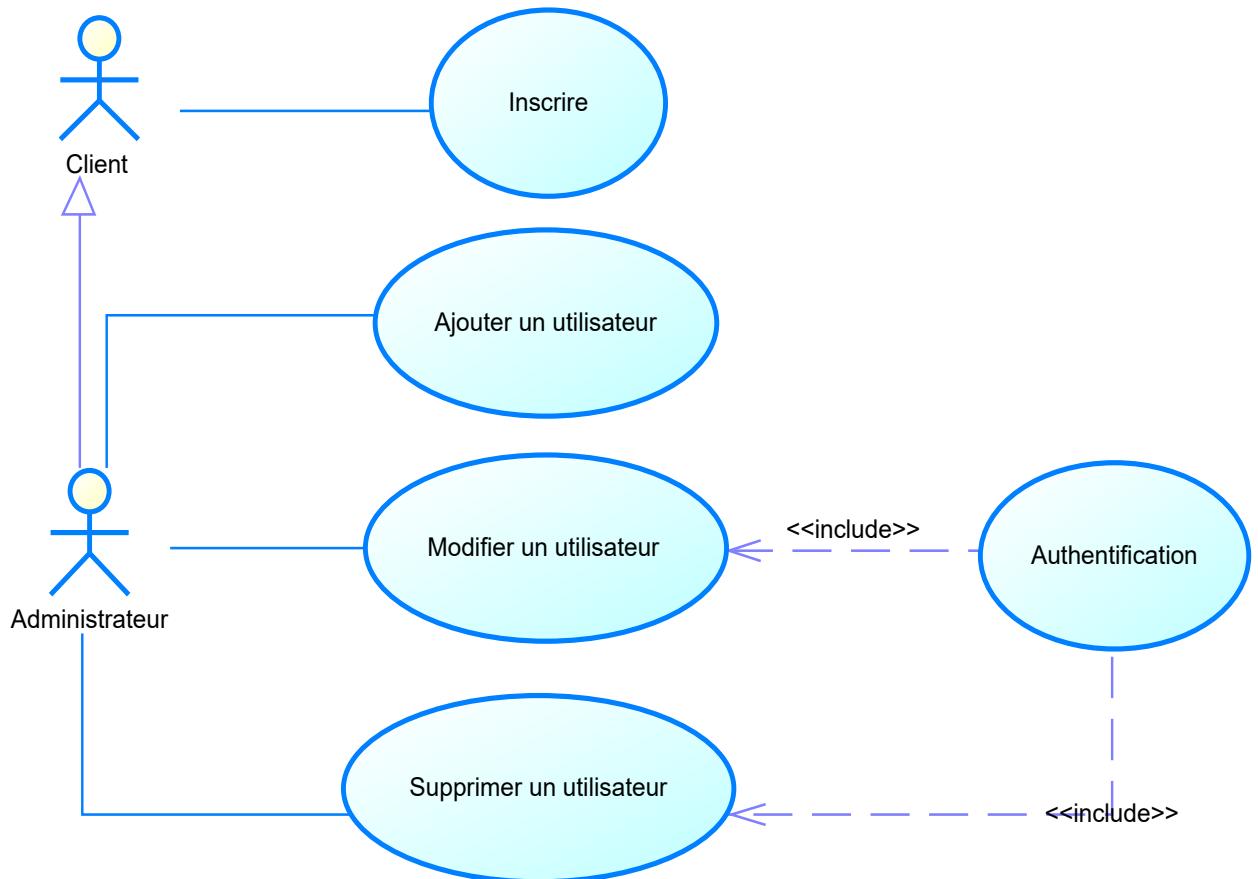


FIGURE 2.3 – Diagramme des cas d'utilisations - Gestion des utilisateurs

Nom de cas	Gestion des utilisateurs
Acteur	Utilisateur, Administateur.
Pré condition	Authentification réussite.
Poste condition	Le système permet d'ajouter, modifier et supprimer un utilisateur.
Scénario principale	Dans cette partie, le client doit saisir ses données personnelles pour pouvoir accéder au site. Par contre l'administrateur a une interface qui permet d'ajouter d'autre utilisateur, les modifier et les supprimer à partir de leurs ID.

TABLEAU 2.2 – Description des cas d'utilisations - Gestion des utilisateurs

1.3.C Gestion des projets

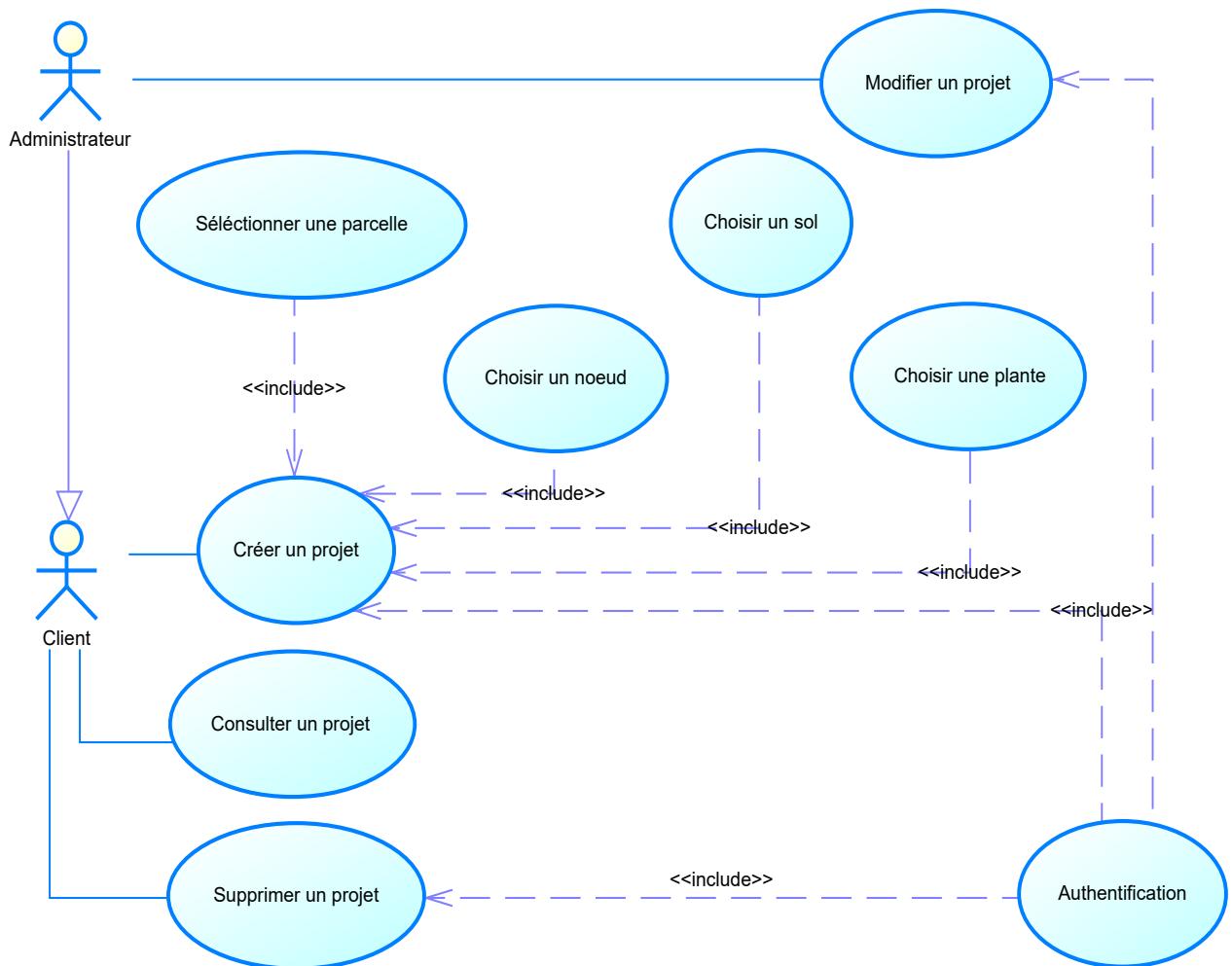


FIGURE 2.4 – Diagramme des cas d'utilisations - Gestion des projets

Nom de cas	Gestion des projets
Acteur	Utilisateur, Administrateur.
Pré condition	Authentification réussite.
Poste condition	Le système permet de créer, modifier et supprimer un projet.
Scénario principale	Dans cette partie, le client doit saisir les données du projet et sélectionner la parcelle de son terrain.

TABLEAU 2.3 – Description des cas d'utilisations - Gestion des projets

1.3.D Gestion des sols

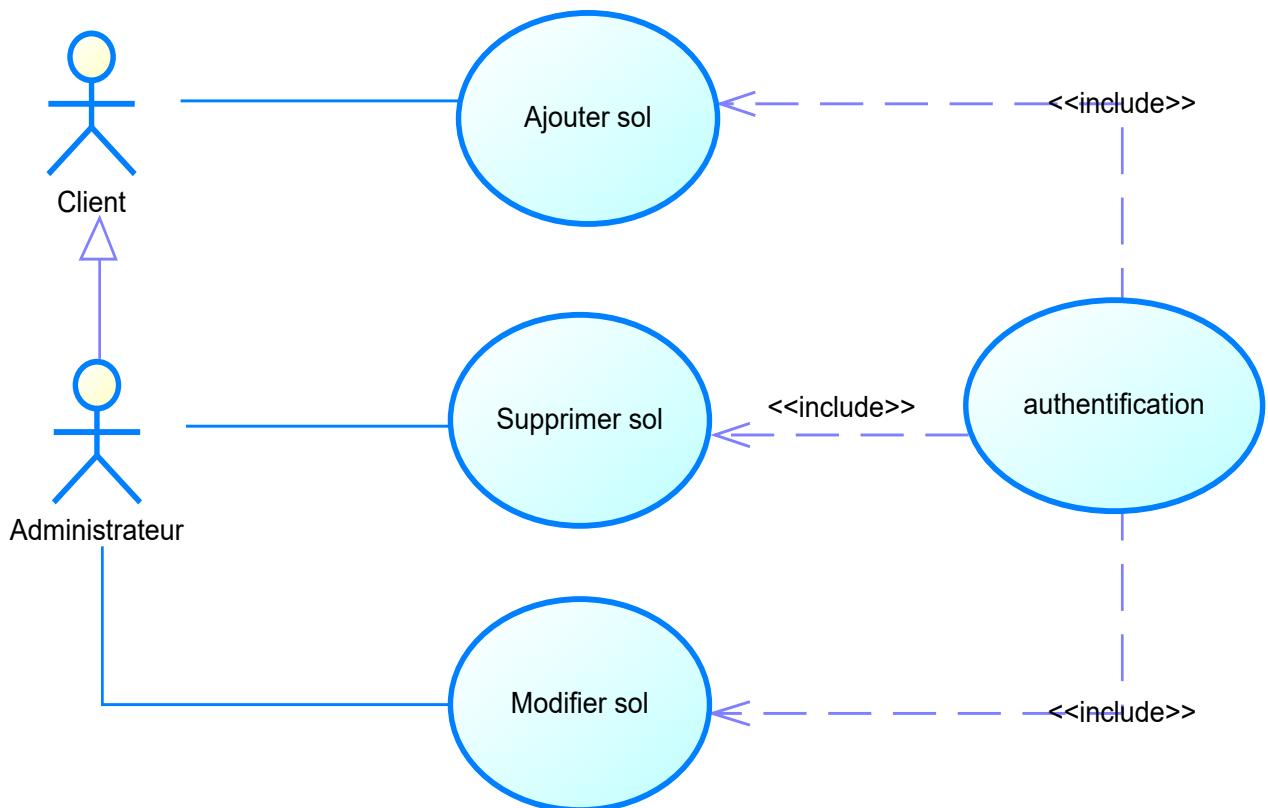


FIGURE 2.5 – Diagramme des cas d'utilisations - Gestion des sols

Nom de cas	Gestion des sols
Acteur	Utilisateur, Administrateur.
Pré condition	Authentification réussite.
Poste condition	Le système permet d'ajouter, modifier et supprimer un sol.
Scénario principale	Dans cette partie, l'utilisateur doit saisir les données du sol de sa parcelle pour pouvoir accéder au formulaire suivant. Par contre l'administrateur peut modifier ou supprimer un sol en utilisant son ID.

TABLEAU 2.4 – Description des cas d'utilisations - Gestion des sols

1.3.E Gestion des plantes

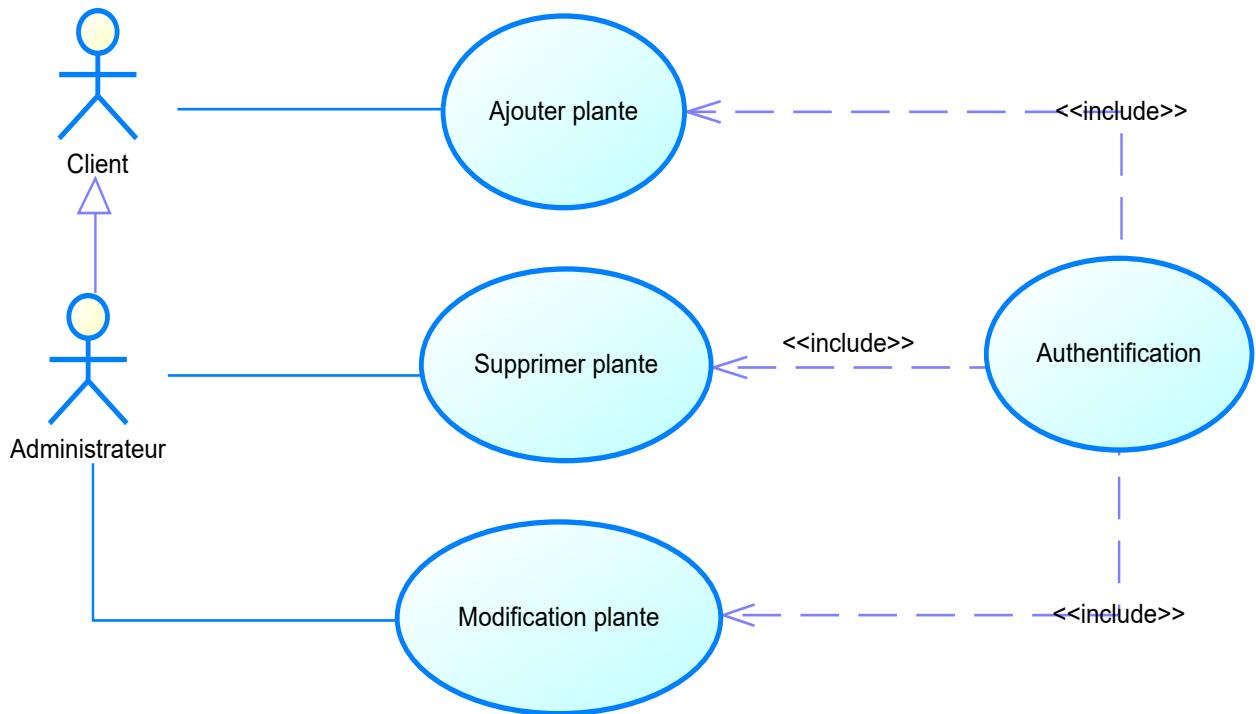


FIGURE 2.6 – Diagramme des cas d'utilisations - Gestion des plantes

Nom de cas	Gestion des plantes
Acteur	Utilisateur, Administrateur.
Pré condition	Authentification réussite.
Poste condition	Le système permet d'ajouter, modifier et supprimer une plante.
Scénario principale	Dans cette partie, l'utilisateur doit saisir les données du plante pour pouvoir accéder au formulaire suivant. Par contre l'administrateur peut modifier ou supprimer une plante en utilisant sa propre ID.

TABLEAU 2.5 – Description des cas d'utilisations - Gestion des plantes

1.3.F Gestion des nœuds

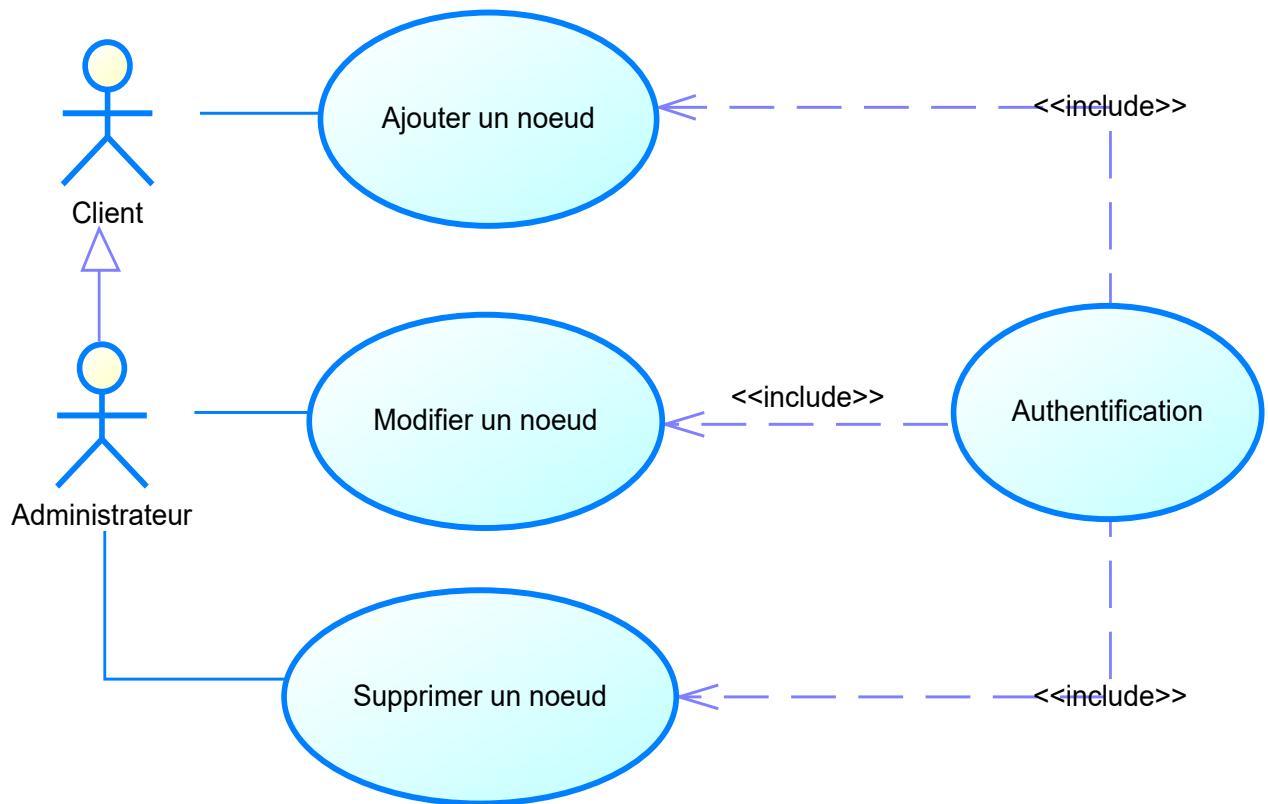


FIGURE 2.7 – Diagramme des cas d'utilisations - Gestion des noeuds

Nom de cas	Gestion des nœuds
Acteur	Utilisateur, Administrateur.
Pré condition	Authentification réussite.
Poste condition	Le système permet de créer, modifier et supprimer un nœud.
Scénario principale	Dans cette partie, l'utilisateur doit saisir les données géographiques des nœuds ainsi que leurs types qu'il va utiliser pour son projet pour valider la réalisations du projet. Par contre l'administrateur a le droit de modifier ou supprimer un nœud grâce à son ID.

TABLEAU 2.6 – Description des cas d'utilisations - Gestion des noeuds

1.3.G Gestion des données climatiques

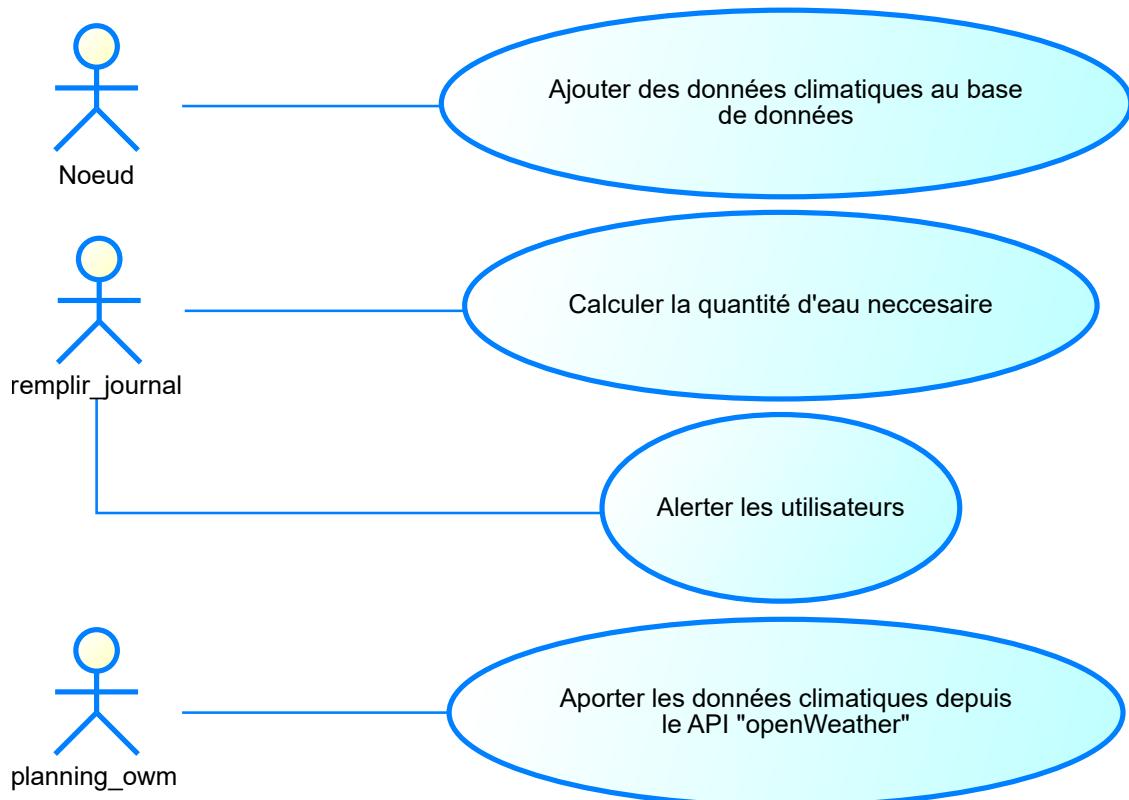


FIGURE 2.8 – Diagramme des cas d'utilisations - Gestion des données climatiques

Nom de cas	Gestion des données climatiques
Acteur	Noeud, remplir_journal, planning_owm.
Pré condition	-
Poste condition	-
Scénario principale	Dans cette partie, le serveur va vérifier les données climatiques prévenant des noeuds puis les enregistrer dans sa base de données. Chaque jour, avant le coucher du soleil, le serveur va traiter les données climatiques journaliers ensuite calculer la quantité nécessaire d'eau en (l/m^2), finalement alerter les abonnées en envoyant un mail.

TABLEAU 2.7 – Description des cas d'utilisations - Gestion des données climatiques

2 Spécification des besoins non fonctionnels

2.1 Python

Python 2.7 est un langage de programmation objet, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions.

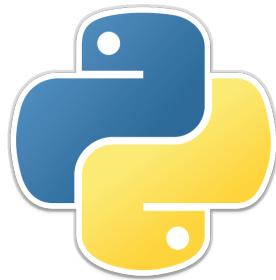


FIGURE 2.9 – Icône - Python

Pourquoi python :

C'est la langage la plus propriété pour tous ce qui est communication avec les noeuds capteurs. Elle dispose les bibliothèques les plus riches parmi les autre langages de programmation dans la communications avec le réseau capteur et internet objet, etc. Ce langage de programmation est open source.

Voici une figure qui représente le classement du langage python parmi les autres langages de programmation :



FIGURE 2.10 – Classement des langages de programmation

2.2 Django

Django est un framework de développement web écrit en Python, il est simple d'accès, rapide et il fournit les outils nécessaires à sécuriser notre application, à gérer la structure de nos modèles (base de données). Sa capacité impressionnante est de s'adapter rapidement et flexiblement aux demandes de trafic les plus lourdes. Il prend la sécurité au sérieux et nous aide à éviter de nombreuses erreurs de sécurité courantes comme "Cross site request forgery (CSRF) protection" ou empêcher un utilisateur malveillant d'exécuter des actions en utilisant les informations d'identification d'un autre utilisateur sans la connaissance ou le consentement de l'utilisateur.



FIGURE 2.11 – Icône - Django

Avantages :

C'est un framework idéal pour réaliser un projet collaboratif avec la moins sécurité nécessaire et qui permet de gérer les bases des données spatiales en utilisant le module GeoDjango. Il y a une plus grande liberté dans la gestion du code et en plus un système d'administration y est déjà pré-construite. Ce framework est simple à déboguer.

Le module GeoDjango a pour but de constituer un cadre applicatif Web géographique de classe mondiale, il vise à faciliter au maximum la création d'application WebGIS et l'exploitation du potentiel de données géolocalisées.

Inconvénients :

La documentation de Django est très détaillé, qu'elle prend du temps pour la étudier complètement.

2.3 PostgreSQL

PostgreSQL est un système gratuit de gestion de base de données relationnelle et objet (SGBDRO) robuste et puissant, aux fonctionnalités riches et avancées. Il peut stocker plus de types de données que les types simples traditionnels (entiers, caractères, etc).

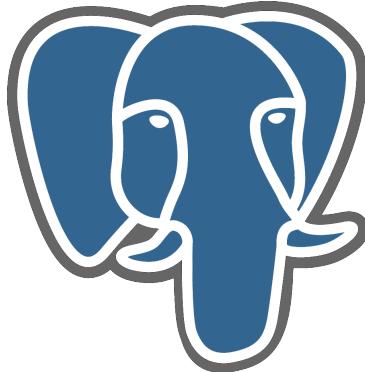


FIGURE 2.12 – Icône - PostgreSQL

PostGIS version 2.3 :

C'est le module spatial qui confère à PostgreSQL le statut de système de gestion de base de données relationnelle (SGDBR) spatial.

pgAdmin version 2 :

Cette interface d'utilisateur est un outil d'administration graphique pour PostgreSQL.

Avantages :

PostgreSQL permet une meilleure prise en charge des données volumineuses par rapport au mySQL. Il permet la gestion des relations entre données géographiques et données attributaires, la gestion des accès concurrentiels et la gestion fine possible des droits d'accès.

Il possède la possibilité de disposer des ‘vues’ (tableaux virtuelles résultantes d'une requête SQL de type SELECT). Grâce au lien fort avec QGIS, ses fonctions spatiales sont plus étendues que sous QGIS natif.

QGIS : QGIS est un Système d'Information Géographique (SIG) convivial distribué sous licence publique générale GNU. C'est un projet officiel de la fondation Open Source Geospatial (OSGeo). Il est compatible avec Linux, Unix, Mac OS X, Windows et Android et intègre de nombreux formats vecteur, raster, base de données et fonctionnalités

Inconvénients :

Le déploiement d'un système de gestion de base de données est un projet de services et donc doit être planifié (charge de travail, etc) et géré comme tel (implication d'un nombre varié d'acteurs). Ce qui fait naissance à un charge de travail supplémentaire pour l'exploitation. Le PostgreSQL mobilise les compétences 'systèmes' pour l'installation et le paramétrage et oblige de maintenir et de gérer les imports(exports vers les fichiers 'à plat' qui sont plus faciles à véhiculer pour les échanges avec les partenaires. Le projet de migration des données et de reprise de l'existant (en particulier des 'projets' QGIS) en cas de décision de migration d'un patrimoine existant sous forme de fichiers 'à plat'. En cas de changement de version majeure de PostgreSQL, les opérations de migration sont obligatoire.

2.4 Google Map

Google Maps est un service gratuit de cartographie en ligne. Le service a été créé par Google. C'est un service qui permet, à partir de l'échelle d'un pays, de zoomer jusqu'à l'échelle d'une rue. Des prises de vue fixes montrant les détails de certaines rues sont également accessibles grâce à une passerelle vers Google Street View.



FIGURE 2.13 – Icône - Google Map

API : Cet api permet de localiser tout type de données sur une carte (routière, satellite, mixte) à partir de son adresse postale. Cet api s'avère très utile pour proposer aux internautes une vision globale et géographique de données. Les résultats sur la carte apparaissent sous la forme d'un picto (petite icône) cliquable. En cliquant sur cet icône, une fenêtre s'ouvre dans laquelle se trouve toutes les informations concernant la donnée géolocalisée (membre, restaurant, hôtel...). Il est également possible de calculer un itinéraire (piéton ou voiture) depuis un point de départ (par exemple, votre résidence) jusqu'à un point d'arrivée (votre recherche).

2.5 Libelium

C'est un plateforme équipe avec un grand nombre de nœuds qui sont des micro-capteurs capables de récolter et de transmettre des données environnementales d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils peuvent être aléatoirement dispersés dans une zone géographique, appelée "champ de captage" correspondant au terrain d'intérêt pour le phénomène capté.

2.5.A La carte Wasp mote

La technologie Wasp mote est une carte électronique constituée d'un microcontrôleur où elle porte un ensemble de modules matériels qui collecte des données à partir d'une variété de capteurs et qui transmettent des informations à une plate-forme Cloud, par le biais de différents technologies (UMTS / GPRS / WiFi / Zigbee / Ethernet).

les caractéristiques de la carte waspmote sont :

- Microcontrôleur : ATmega1281
- Fréquence : 8MHz
- SRAM : 8KB
- EEPROM : 4KB
- FLASH : 128KB
- SD Carte : 2GB
- Poids : 20gr
- Dimensions : 73.5 x 51 x 13 mm
- Intervalle de la température : [-20°C, +65°C]

Wasp mote est basé sur une architecture modulaire. L'idée est d'intégrer uniquement les modules nécessaires dans chaque dispositif. Ces modules peuvent être modifiés et étendus en fonction des besoins.

Les modules disponibles pour intégration dans Wasp mote sont classés en :

- Modules de ZigBee / Bluetooth (2,4GHz, 868MHz, 900MHz).
- Module GSM / GPRS (quadribande : 850MHz / 900MHz / 1800MHz / 1900MHz).
- Module UMTS (quadribande EGSM : 850MHz / 900MHz / 1800MHz / 1900MHz).
- Module WiFi (Redpine RS9110-N-4-24-02).
- Module PoE (Power over Ethernet).
- Modules de capteurs (Expansion Board).

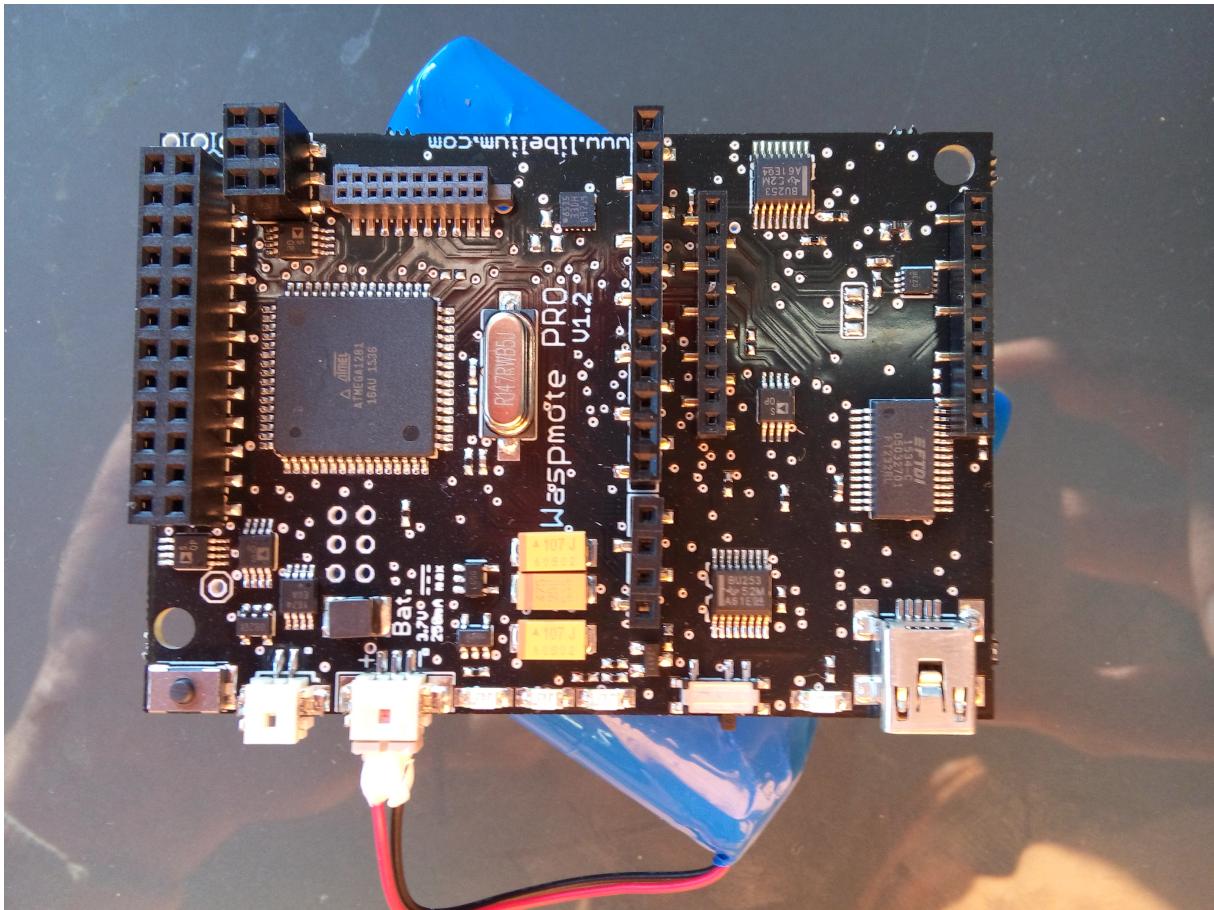


FIGURE 2.14 – La carte Waspmote

C'est quoi un microcontrôleur :

C'est un circuit intégré qui rassemble les éléments essentiels d'une carte waspmote : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties.

Les caractéristiques d'un microcontrôleur :

Les microcontrôleurs se caractérisent par une densité plus haute d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

2.5.B Le module Xbee

Les modules XBee sont des circuits de communication sans-fil utilisant les protocoles IEEE 802.15.4 pour la couche 2 du modèle OSI et ZigBee pour la couche 3.

Le protocole IEEE 802.15.4 permet la communication entre les modules alors que le protocole ZigBee crée la hiérarchie du réseau et configure d'autres paramètres comme l'association, l'authentification, l'encodage, le routage ou encore les services de la couche d'application autrement appelés les clusters.

Le 802.15.4 est un protocole à basse consommation (Lazy). Il utilise des cycles courts, ce qui permet au transmetteur d'être mis en veille la plupart du temps.



FIGURE 2.15 – Composant xbee

2.5.C Shield WaspMote de capteurs d'agriculture

La carte waspmote d'agriculture permet de surveiller plusieurs paramètres environnementaux impliquant une large bande d'applications comme l'analyse de développement croissant et l'observation du météo. Pour cela, cette carte a été fourni avec des capteurs pour la température, l'humidité, l'humidité du sol, rayonnement solaire, vitesse du vent, pression atmosphérique et plein d'autres, cette carte permet au utilisateur d'analyser les paramètres les plus importants dans les applications agriculture et météo.



FIGURE 2.16 – les différents capteurs climatiques

2.5.D Gateway Meshlium

Meshlium est un routeur Linux qui fonctionne comme la passerelle des Waspmove Sensor Networks.

Il peut contenir 6 radios interfaces différentes : WiFi 2.4GHz, WiFi 5GHz, 3G / GPRS, Bluetooth, XBee et LoRa.

De plus, ce Meshlium peut également intégrer un GPS module pour applications mobiles et véhiculaires et être solaire et alimenté par batterie.

Ces caractéristiques avec une boîte aluminium IP65 permettent à Meshlium d'être placé n'importe où en extérieur.

Meshlium peut fonctionner comme suit :

- un routeur XBee / LoRa to Ethernet pour les nœuds Waspmove.
- un routeur XBee / LoRa vers 3G / GPRS pour les nœuds Waspmove.
- un point d'accès WiFi.
- un nœud WiFi Mesh (double bande 2,4 GHz-5GHz).
- un routeur WiFi vers 3G / GPRS.

- un scanner et un analyseur Bluetooth.
- un GPS-3G / GPRS en temps réel tracker.
- un scanner de téléphones intelligents (détecte les appareils iPhone et Android).

Meshlium est une station Linux complète qui offre des différents services, environnements de programmation et systèmes de stockage.

Les clients peuvent se connecter à Meshlium via wifi avec leurs pc et/ou leurs smartphones pour contrôler l'interface et pour accéder aux données du capteur.

Pour prendre l'accès à l'internet, Meshlium utilise ; soit une connection Ethernet soit une connection 3G/GPRS.

Quand Meshlium sera connecté à un hub ou un switcheur, il va prendre automatiquement une IP en utilisant DHCP.

Ce modèle peut recevoir et stocker les données depuis Waspmotés avec GPRS , 3G ou Wifi et les envoyer via protocole HTTP.

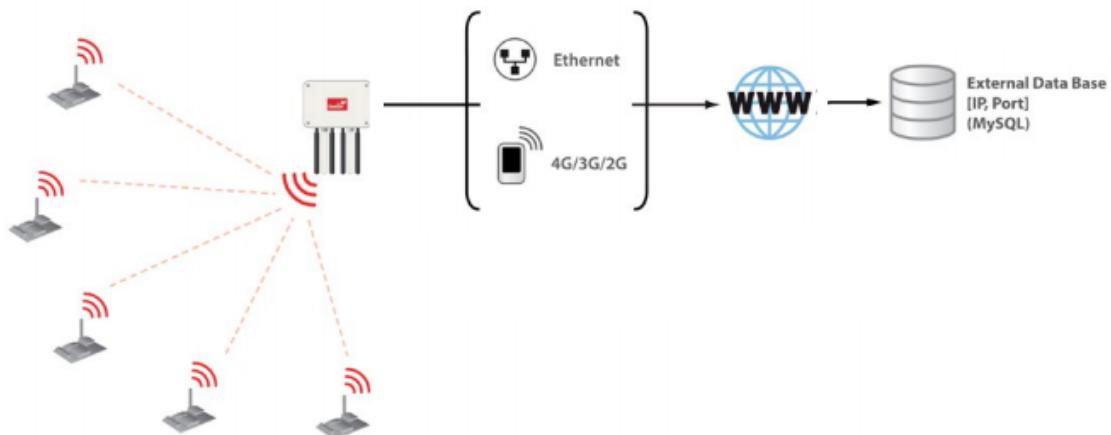


FIGURE 2.17 – Fonctionnement du Meshlium

3 Conclusion

Dans ce chapitre, on a spécifié les besoins fonctionnels et non fonctionnels de notre projet.

Chapitre III

Conception

En se basant sur les informations collectées dans le chapitre précédent, nous avons eu une vision plus concrète des besoins de notre application. Dans ce chapitre nous abordons la partie conception du projet, dans laquelle, nous détaillons les différents éléments de conception, à savoir les diagrammes de séquences, et le diagrammes de classe.

1 Choix du langage de conception

Notre application est orientée objet, le processus de développement suggéré est l'UML. L'UML (Unified Modeling Language) est un langage de modélisation graphique pour visualiser la conception d'un système. Le choix a été fait sur la présentation des diagrammes de séquence, des diagrammes de cas d'utilisation (voir chapitre II, section 1) et le diagramme de classe.

2 Conception détaillée

2.1 Diagrammes de séquence

Le diagramme de séquence décrit l'aspect dynamique du système. Il modélise les interactions entre les objets ou entre utilisateur et objet, en mettant l'accent sur la chronologie des messages échangés. Dans ce qui suit, nous allons dresser les diagrammes séquences de chaque acteur.

2.1.A Client : Inscription

C'est le premier scénario qui se déroule lors du déclenchement de l'application. Selon la figure 3.1, le client est invité à saisir ses informations personnelles. Le système vérifie la disponibilité du pseudo et du mot de passe, pour donner ensuite accès aux fonctionnalités de l'application.

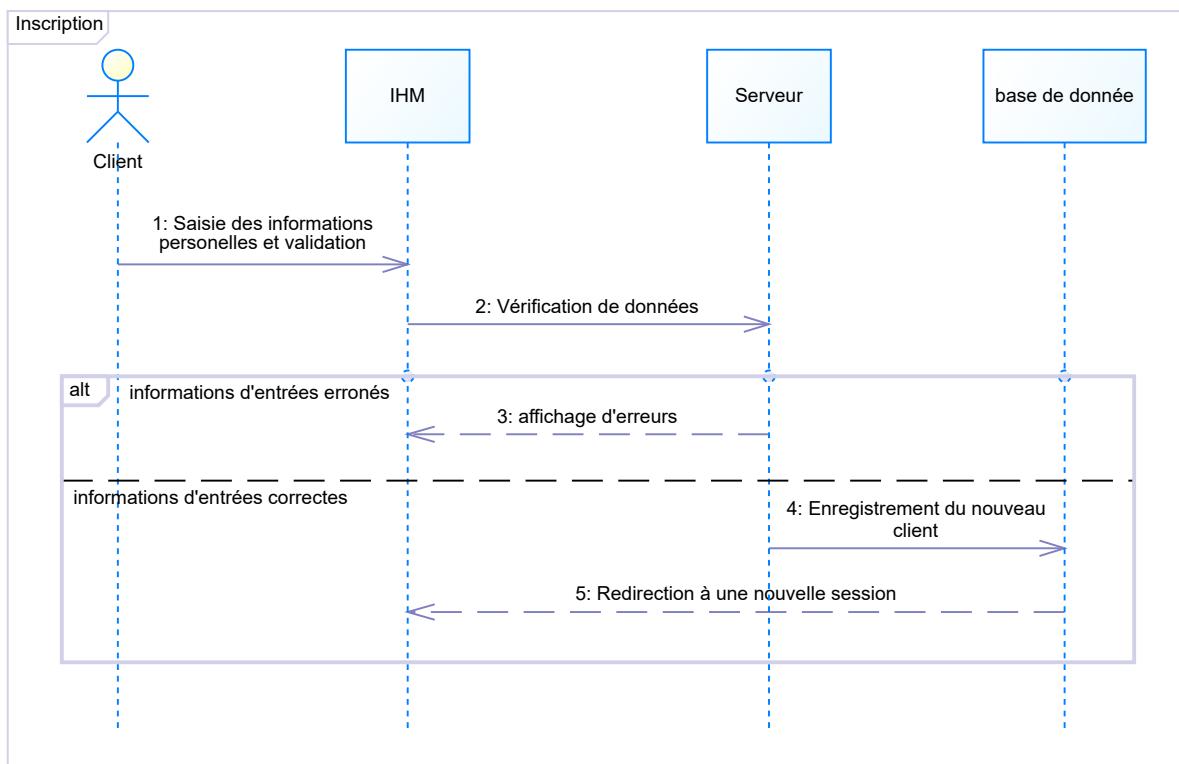


FIGURE 3.1 – Description des diagrammes de séquences - Inscription

2.1.B Client ou administrateur : Authentification

Selon la figure 3.2, le système invite l'utilisateur à saisir les paramètres d'accès (pseudo et mot de passe). Il essaie ensuite de vérifier l'unicité de ses paramètres. Lorsque la vérification est terminée sans erreurs, l'utilisateur sera redirectionné vers la page qui contient tous ses projets d'irrigation.

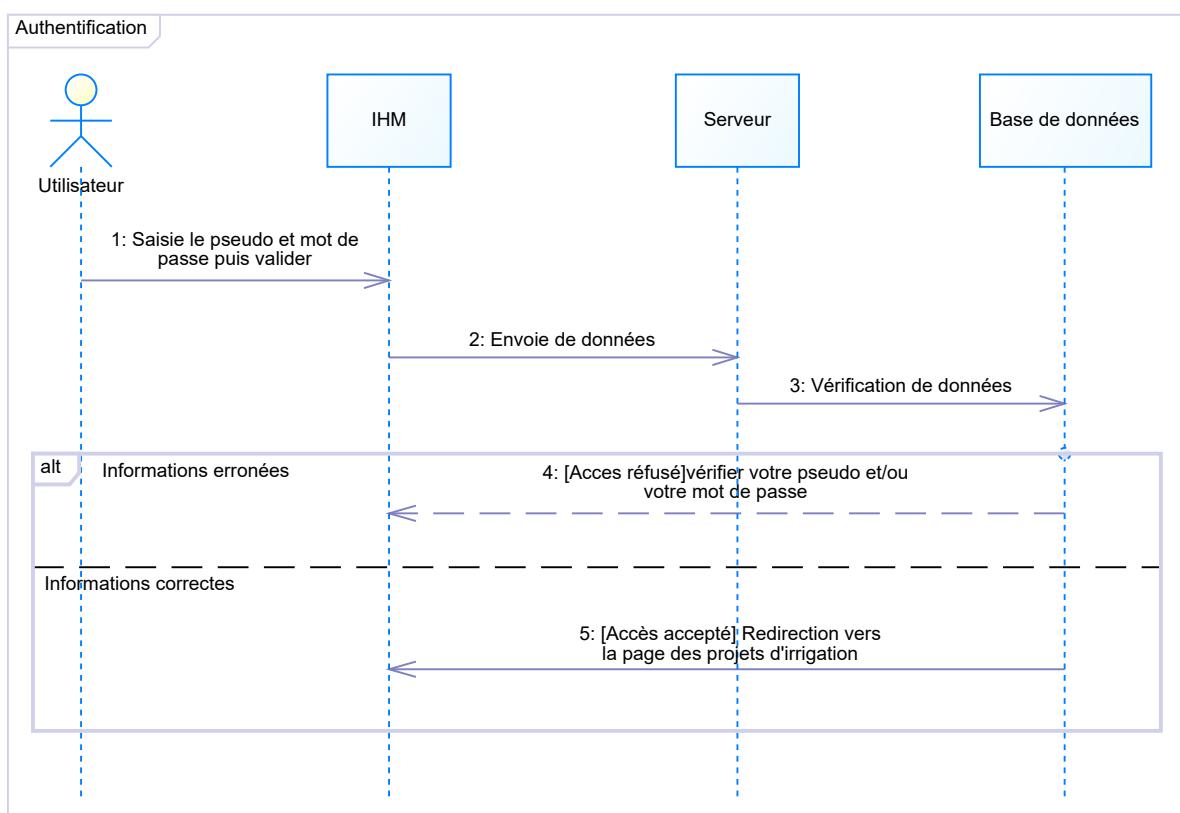


FIGURE 3.2 – Description des diagrammes de séquences - Authentification

2.1.C Client : Choix du type de projet

Dans le menu du Création d'un projet, l'utilisateur se trouve devant le choix entre 3 types de projet (estimation avec service web climatique ou avec réseau de capteurs sans fils).

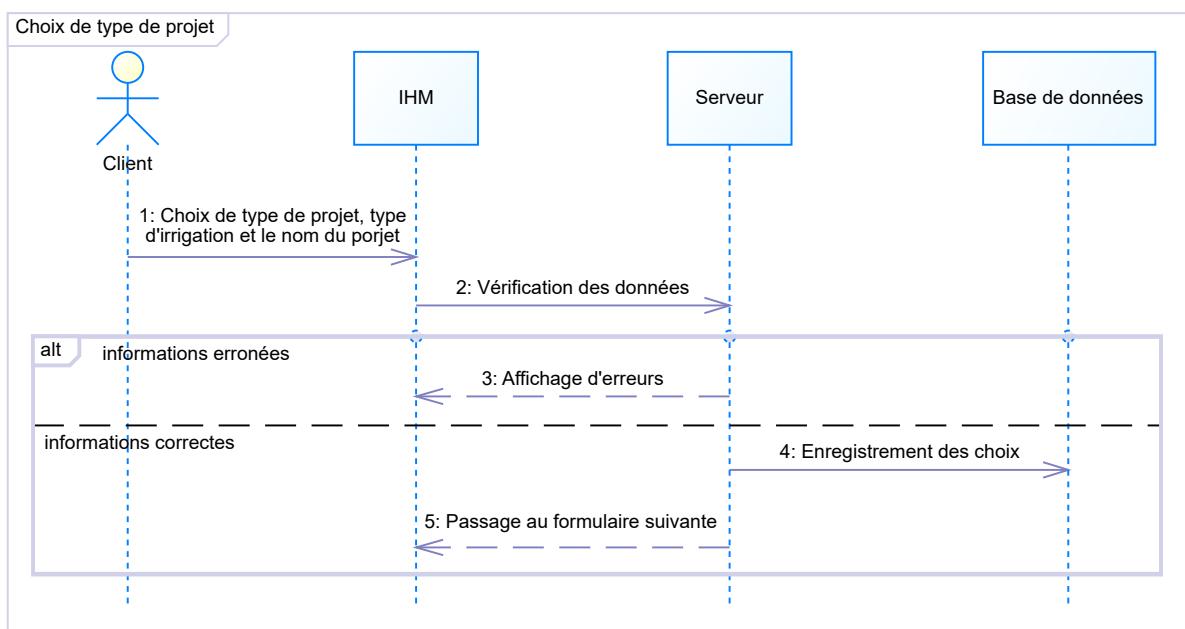


FIGURE 3.3 – Description des diagrammes de séquences - Choix du type de projet

2.1.D Client : Crédit du projet de type web service climatique

Après la confirmation du choix de type de projet, l'utilisateur dessine ou importe un polygone puis crée ou sélectionne un sol. Enfin, il finit par créer ou sélectionner une plante. Ce scénario est décrit dans la figure 3.4.

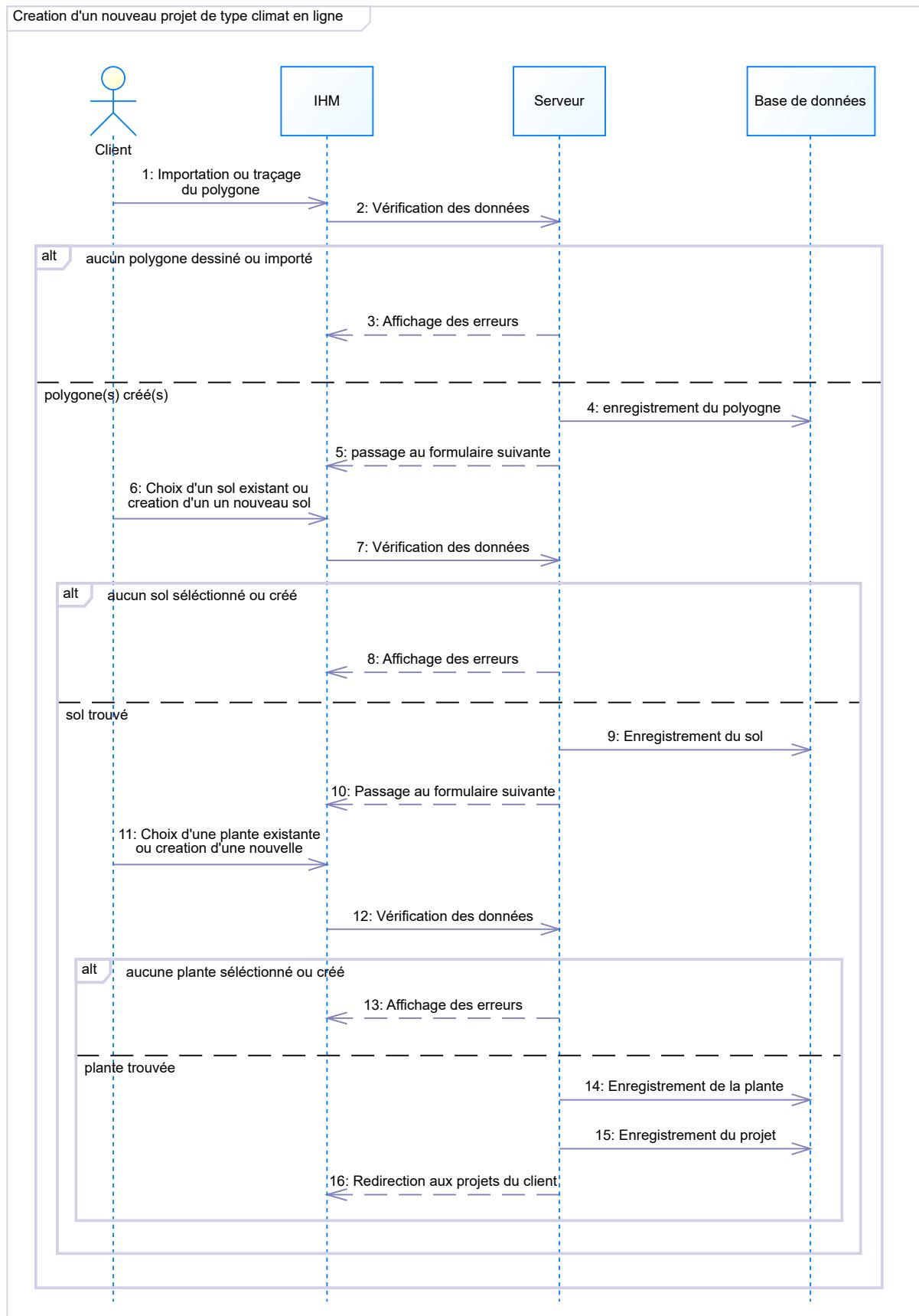


FIGURE 3.4 – Description des diagrammes de séquences - Création du projet de type web service climatique

2.1.E Client : Création du projet de type nœud(s) capteur(s) dans une parcelle

Après la confirmation du choix de type de projet, l'utilisateur dessine ou importe un polygone puis crée ou sélectionne un sol. Ensuite, il crée ou sélectionne une plante puis il pointe ses capteurs sur sa parcelle. Ce scénario est décrit dans la figure 3.5.

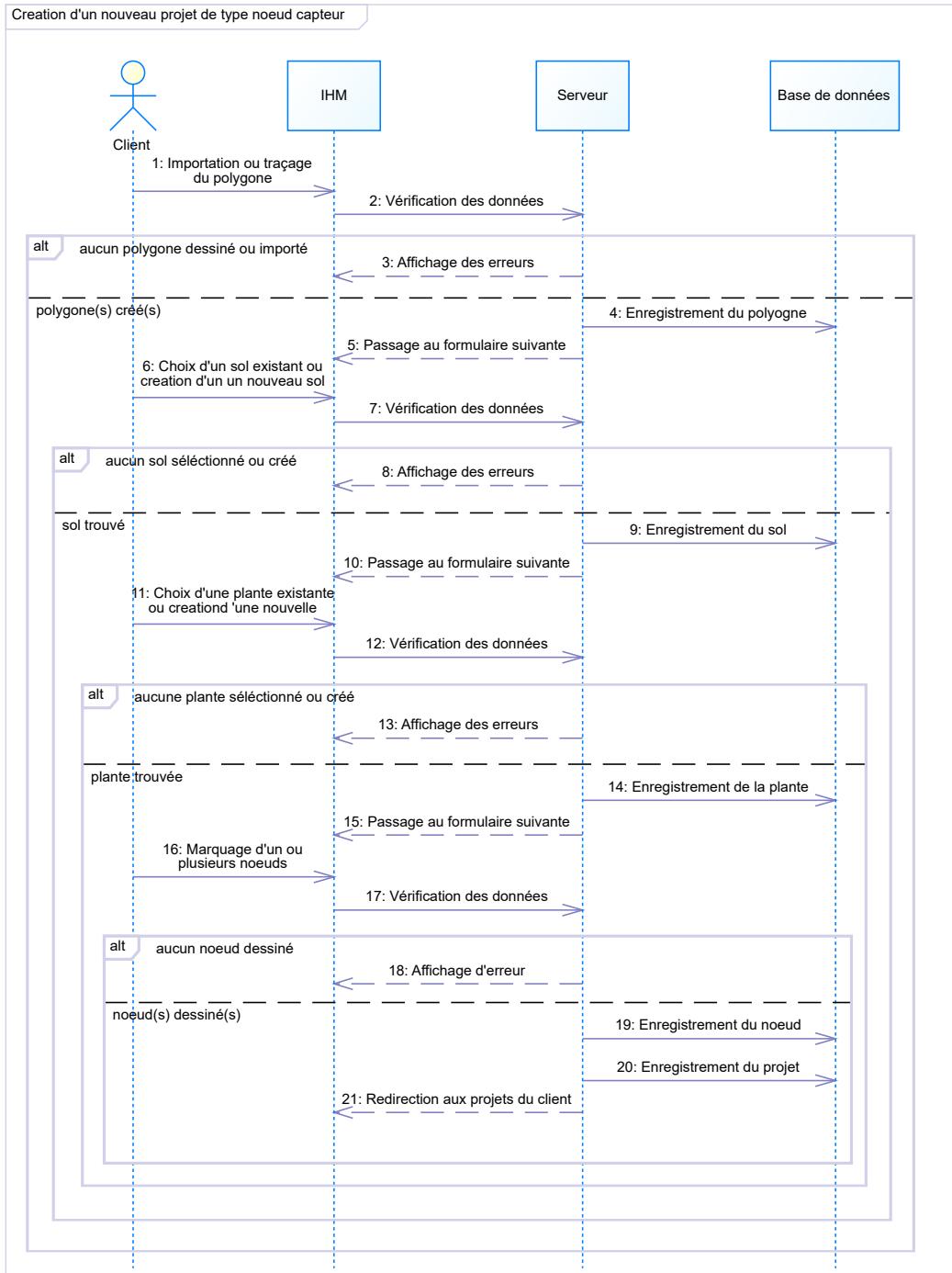


FIGURE 3.5 – Description des diagrammes de séquences - Création du projet de type nœud(s) capteur(s) dans une parcelle

2.1.F Client : Création du projet de type nœud(s) capteur(s) dans in pot

Après la confirmation du choix du type de projet, l'utilisateur dessine ou importe un polygone puis crée ou sélectionne un sol. Ensuite, il crée ou sélectionne une plante, enfin, il pointe son capteur sur une map pour indiquer son pot. Ce scénario est décrit dans la figure 3.6.

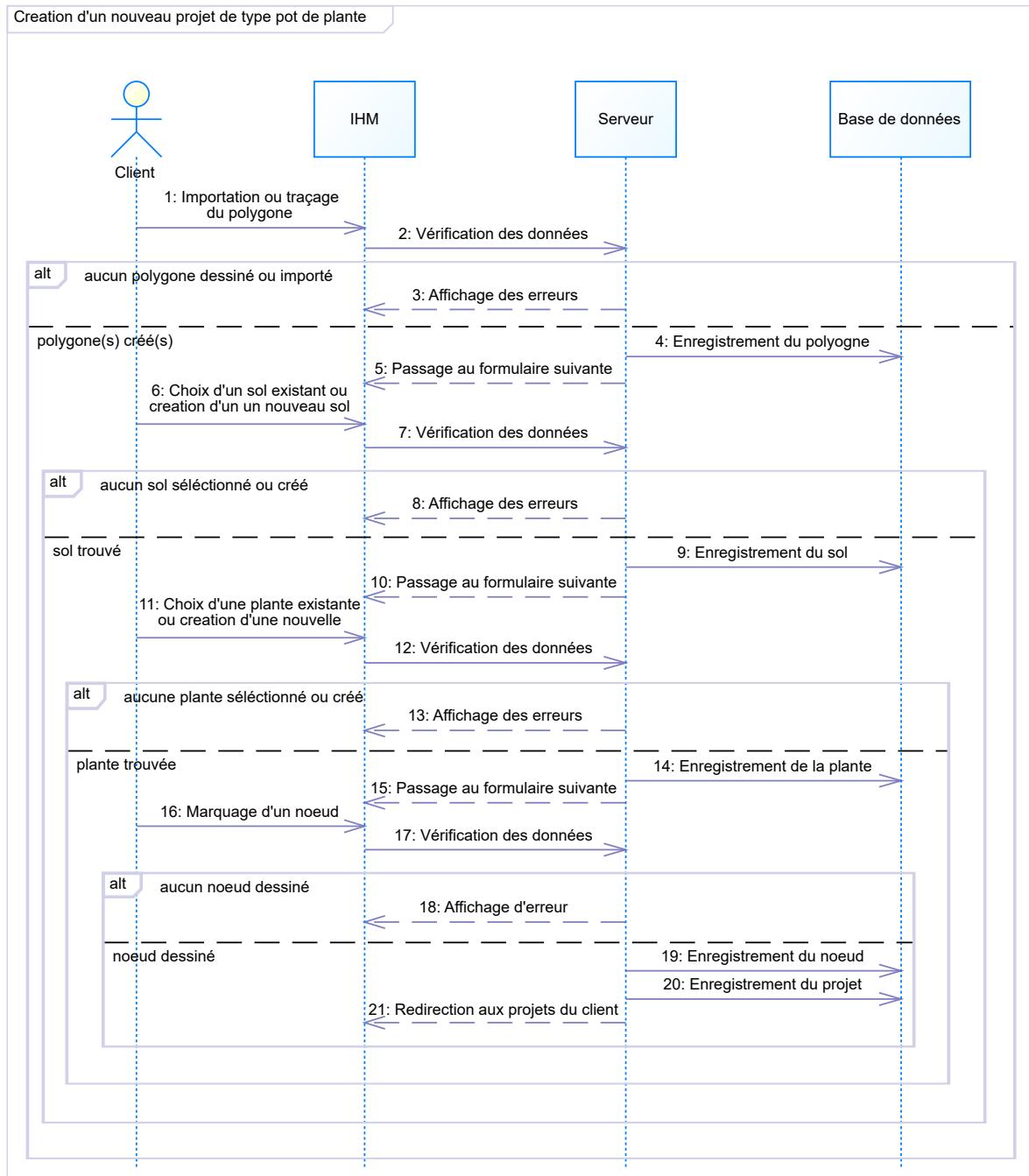


FIGURE 3.6 – Description des diagrammes de séquences - Création du projet de type nœud(s) capteur(s) dans in pot

2.1.G Administrateur : Création du projet

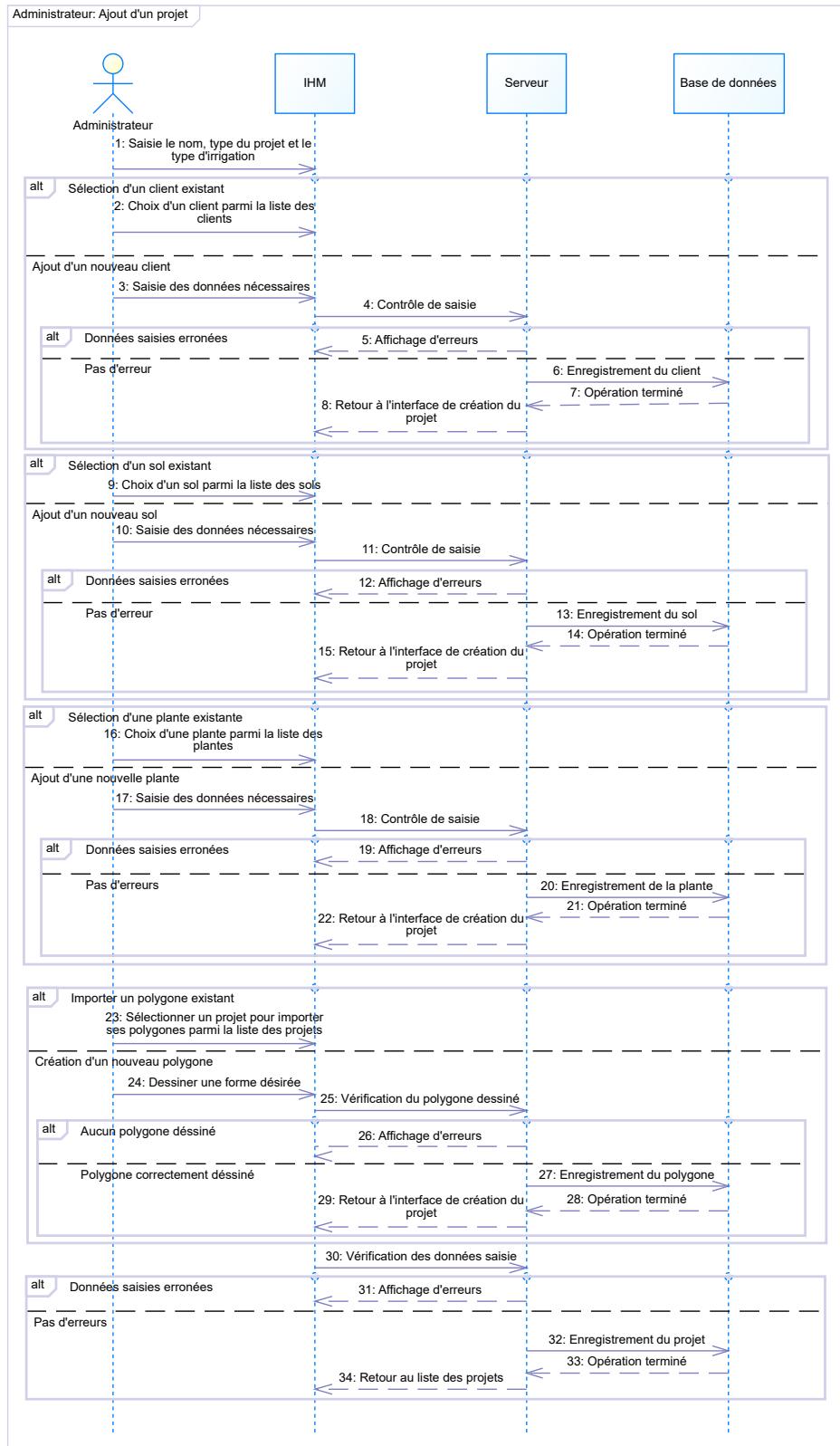


FIGURE 3.7 – Description des diagrammes de séquences - Création du projet

2.1.H Administrateur : Gestion des (clients - plantes - sols - projets - polygones - noeuds - climat journalier)

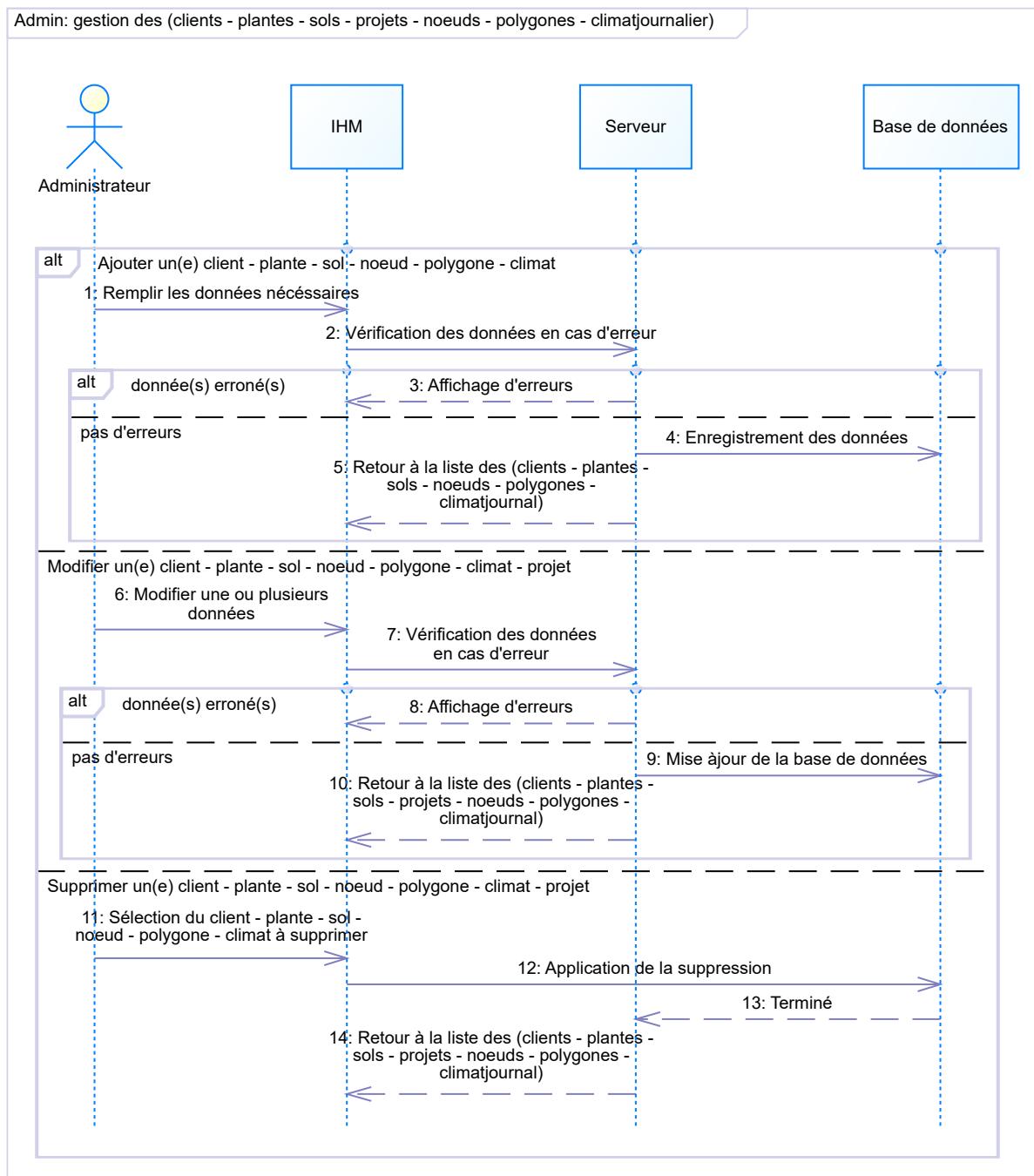


FIGURE 3.8 – Description des diagrammes de séquences - Gestion de l'application

2.2 Diagramme de classe

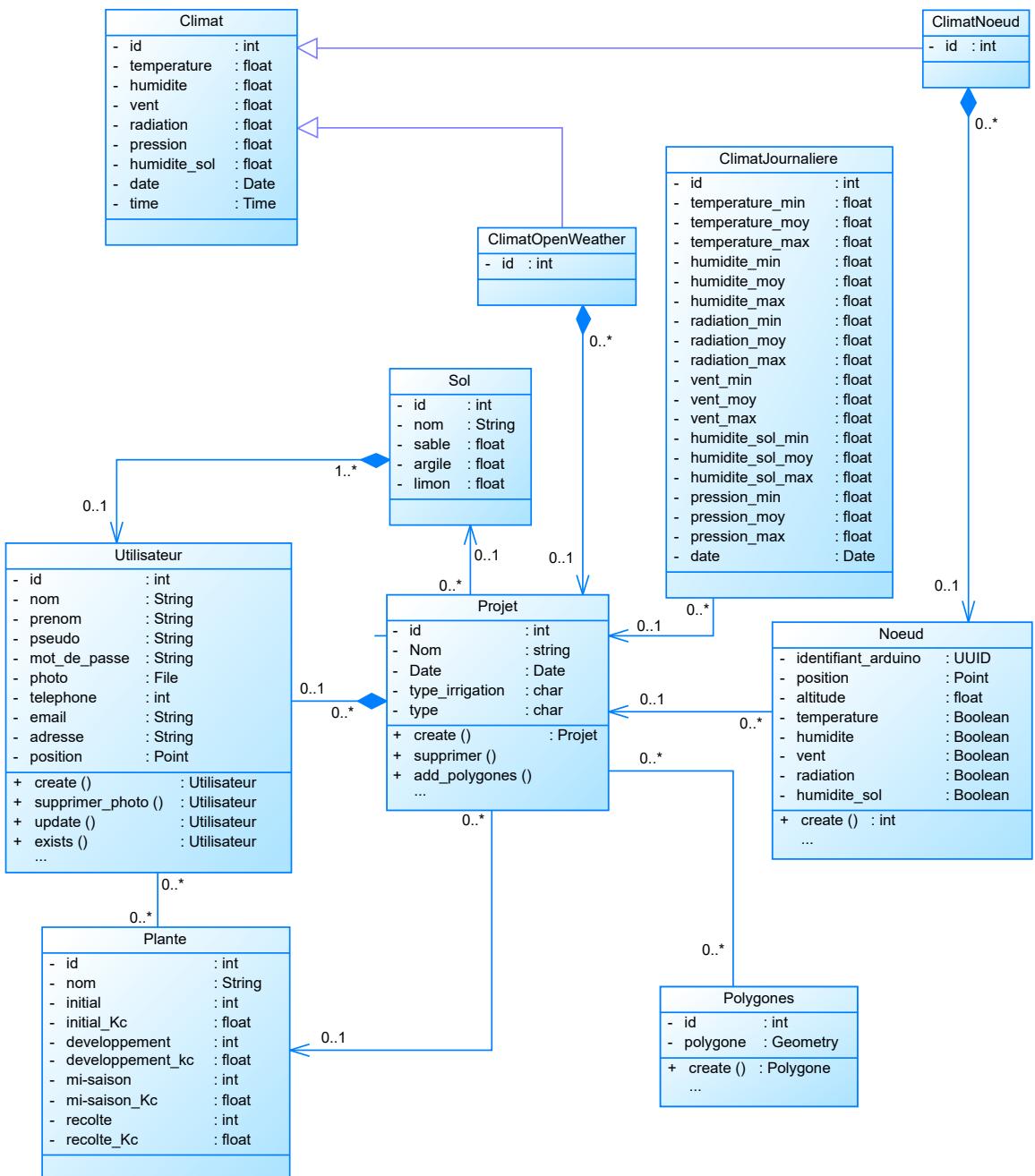


FIGURE 3.9 – Diagramme de classe

Chapitre IV

Réalisation du projet

La réalisation est la dernière phase de la construction d'une application. C'est au niveau de laquelle que notre projet sera implémenté réellement. L'interaction de l'environnement logiciel et matériel nous permet d'avoir une application fonctionnelle.

En fait, nous allons en premier lieu parler de l'architecture matérielle mise en place ainsi les outils matériels et logiciels utilisés lors du développement. Ensuite, nous présentons les interfaces de l'application.

1 Architecture mise en place

1.1 Architecture générale

L'architecture mise en place contient une partie embarquée sur le réseau des capteurs, partie de base de données, partie serveur et partie client web.

Pour la partie embarquée sur le réseau des capteurs, on propose dans notre application que les données de l'environnement seront collectées soit par un nœud de capteur : le capteur envoie les données périodiquement sous forme d'un flux JSON à notre web service afin de les enregistrer dans la base de données de l'application. Soit par un web service météo : On récupère les données par l'envoi d'une requête au web service météo contenant les coordonnées géographiques du lieu.

La partie de base de données consiste à un serveur qui gère les données des utilisateurs ainsi que leurs projets. La partie serveur qui contrôle et traite les requêtes reçues par les clients et les nœuds. La figure 4.1 donnera une meilleure compréhension de l'architecture général.

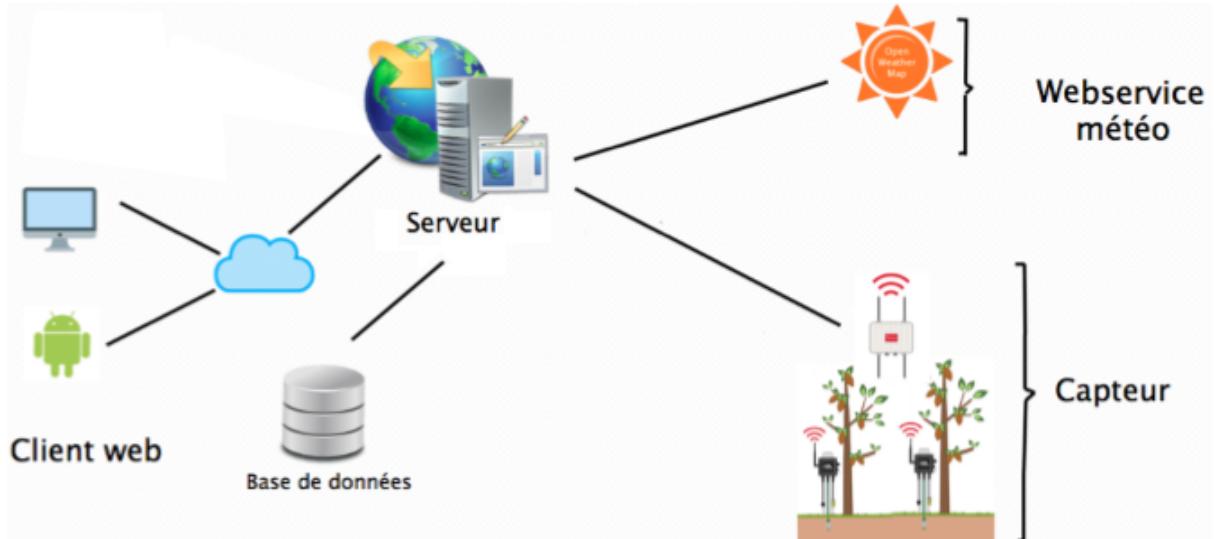


FIGURE 4.1 – Architecture générale de l'application

1.2 Architecture MTV

Django est un système MTV ; modèle, template et vue. Dans notre interprétation du MTV, la vue décrit les données qui sont présentées à l'utilisateur. Ce n'est pas nécessairement comment les données se présentent, mais quelles données sont présentées. La vue décrit quelles données l'utilisateur voit, pas comment il les voit. C'est une distinction subtile.

Donc, dans notre cas, une vue est la fonction de rappel Python pour une URL donnée, car la fonction de rappel détermine quelles données sont présentées. De plus, il est sensé de séparer le contenu de la présentation ; c'est là que les gabarits (templates) entrent en jeu.

Dans Django, une vue décrit quelles données sont présentées, mais elle délègue normalement la suite à un gabarit, qui décrit comment les données sont présentées. Dans le cas de Django, il s'agit probablement du système en lui-même : la machinerie qui envoie une requête à la vue appropriée, selon la configuration d'URL de Django. La figure 4.2 donnera une meilleure compréhension de l'architecture MTV.

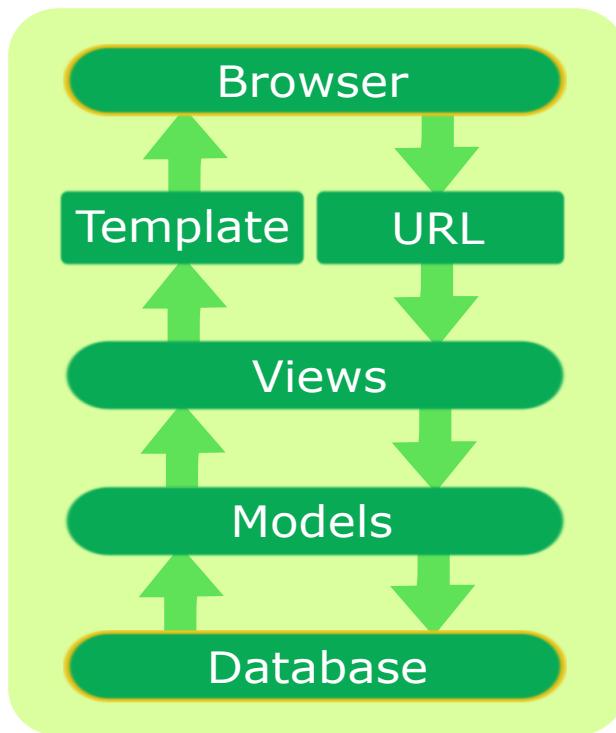


FIGURE 4.2 – Architecture MTV

2 Environnement Matériel

Pour réaliser ce projet, nous avons utilisé deux ordinateurs avec ces caractéristiques suivantes :

Caractéristiques 1 : Inspiron-n5010

- Processeur : Processeur Intel Core(TM) i5 CPU M480 @ 2.62GHz 2.66GHz
- Disque dur : 500 Go
- RAM : 4 Go
- Système d'exploitation : Ubuntu 16.1 LTS

Caractéristiques 2 : Asus X550 G

- Processeur Intel Core(TM) i7-4710HQ CPU @ 2.5GHz 2.5GHz
- Disque dur : 1 To
- RAM : 8 Go
- Système d'exploitation : Ubuntu 14.4 LTS

3 Environnement logiciel

Dans notre projet, nous avons utilisé les versions des logiciels suivantes :

- Python : version 2.7
- Django : version 1.10
- PostgreSQL : version 9.6

Pour pouvoir les installé sans aucun problème, il faudra suivre les étapes nécessaires de chaque logiciel qui se trouvent dans les titres suivants.

3.1 L'installation de l'environnement Django 1.10

La manière la plus flexible pour installer Django sur n'importe quel système est à partir de l'outil virtualenv. Cet outil va permettre de créer des environnements Python virtuels qui permettent d'installer n'importe quel paquet Python sans affecter le reste du système. Cela permet de sélectionner des paquets Python par projet, indépendamment des conflits avec les autres exigences du projet.

Alors, il faut commencer par installer 'pip' à partir des dépôts Ubuntu puis actualisez les paquètes de Python avant de commencer en exécutant cette commande :

```
sudo apt-get update
```

Nous avons utilisé la langage Python 2.7, il faut alors taper la commande suivante pour installer pip :

```
sudo apt-get install python-pip
```

Une fois que pip est installé, procédez par installer la paquête virtualenv :

```
sudo pip install virtualenv
```

Après avoir installer l'environnement virtuel, créez un environnement virtuel dans le répertoire du projet en tapant :

```
virtualenv nom_de_l_environnement_virtuel
```

Cela installera une version autonome de Python, ainsi que pip, dans une structure de répertoire isolée dans votre répertoire de projet. Pour installer des paquets dans l'environnement isolé, il faut l'activer en tapant :

```
source nom_de_l_environnement_virtuel/bin/activate
```

Après avoir configuré l'environnement virtuel, procédez par installer Django en exécutant cette commande :

```
pip install django
```

Pour vérifier si Django est installé avec succès, il faut exécuter la commande suivante :

```
django-admin --version
```

La figure ci-dessous correspondant à l'installation de Django, Elle indique que l'installation est effectuée avec succès.

```
django.contrib.flatpages.tests.middleware: module references __file__
django.contrib.admin.bin.compress: module references __file__
django.contrib.sitemaps.tests.http: module references __file__
django.contrib.gis.geometry.test_data: module references __file__
django.contrib.gis.tests.geogapp.tests: module references __file__
django.contrib.gis.tests.geo3d.tests: module references __file__
django.contrib.gis.tests.layermap.tests: module references __file__
django.contrib.formtools.tests.__init__: module references __file__
django.contrib.formtools.tests.wizard.wizardtests.tests: module references __file__
django.contrib.formtools.tests.wizard.namedwizardtests.tests: module references __file__
django.contrib.admindocs.views: module references __file__
django.db.utils: module references __file__
django.db.models.loading: module references __file__
django.core.management.__init__: module references __file__
django.core.management.__init__: module references __path__
django.core.management.templates: module references __path__
django.core.management.sql: module references __file__
django.core.management.commands.loaddata: module references __file__
django.core.management.commands.loaddata: module references __path__
django.core.management.commands.makemessages: module references __file__
django.views.i18n: module references __file__
django.utils.version: module references __file__
django.utils.module_loading: module references __path__
django.utils.autoreload: module references __file__
django.utils.unittest.loader: module references __file__
django.utils.unittest.collector: module references __file__
django.utils.translation.trans_real: module references __file__
django.test._doctest: module references __file__
django.test._doctest: module MAY be using inspect.getsourcefile
Adding Django 1.5.4 to easy-install.pth file
Installing django-admin.py script to /home/benn/testings/myenv/bin

Installed /home/benn/testings/nyenv/lib/python2.7/site-packages/Django-1.5.4-py2.7.egg
Processing dependencies for django
Finished processing dependencies for django
(myenv)benn@benn-Inspiron-N5010:~/testings$
```

FIGURE 4.3 – Capture d'écran de l'installation du l'environnement Django

3.2 Crédation et configuration du serveur PostgreSQL 9.6 et PostGIS

Ubuntu comprend PostgreSQL par défaut. Pour installer PostgreSQL et PostGIS sur Ubuntu, il faut exécuter les commandes suivant :

```
sudo apt-get install postgresql-9.6
sudo apt-get install postgresql-9.6-postgis-2.3 postgresql-contrib-9.6
sudo apt-get install postgis
```

La première commande permet d'installer le serveur PostgreSQL version 9.6, La deuxième commande installe le fameux PostGIS version 2.3 qui est compatible avec la version du serveur PostgreSQL choisi. la dernière commande est optionnelle, Elle installe PGAdmin4 ; C'est une interface qui facilite la gestion des données dans le serveur PostgreSQL pour l'utilisateur, mais dans notre projet, on a utilisé des commandes pour créer et configurer notre base de données.

Aucune configuration supplémentaire a été réalisé après avoir installer les paquets nécessaire du serveur, on a utilisé la configuration par défaut.

3.3 Crédation de la base de données

Quand l'installation de PostgreSQL et PostGIS est terminée, il faut ouvrir la console de PostgreSQL puis créer la base de données. Mais avant de commencer la création des tableaux, il faudra transformer la base de données en une base qui peut gérer les données spatial, Cette transformation est réalisée avec l'exécution de la dernière commande permis ces commandes :

```
sudo -u postgres psql
CREATE DATABASE nom_de_la_base_de_données ;
\connect nom_de_la_base_de_données ;
CREATE EXTENSION postgis ;
```

Si l'utilisateur veut ajouter des tableaux dans une base de données existante, il doit exécuter la même commande qui permet d'ouvrir la console de PostgreSQL en ajoutant le nom de la base de données en tant que paramètre, comme indique l'exemple suivant :

```
sudo -u postgres psql nom_de_la_base_de_données
```

3.4 Intégration des services de OpenWeather

Le service météorologique OpenWeatherMap est un service gratuit qui est basé sur la plate-forme 'VANE Geospatial Data Science' pour la collecte, le traitement et la distribution d'informations sur notre planète grâce à des outils et des API faciles à utiliser. Les services de l'API de OpenWeather sont garantie seulement pour les utilisateurs qui possède un clé d'identification. Pour avoir l'accès à ce clé, une inscription dans le site officiel de OpenWeather est obligatoire.

pyOWM : pyOWM est une bibliothèque client Python pour l'OpenWeatherMap (OWM) web API qui nécessite un clé unique donné par OpenWeather. Elle permet une consommation rapide et facile des données météorologiques OWM des applications Python via un modèle d'objet simple et de manière respectueuse de l'homme. PyOWM fonctionne sur Python 2.7 et Python 3.2+, et s'intègre avec les modèles Django 1.10+.

Pour l'installer, il suffit d'exécuter la commande suivante dans le terminal du système :

```
pip install pyowm
```

3.5 Configuration et intégration de réseau capteur

3.5.A Environnement de programmation

Wasmote-IDE est un compilateur open source pour la plateforme arduino. Il est utilisé pour écrire le code et le télécharger à Wasmote. Il est également utilisé pour surveiller la sortie série et pour le débogage.



FIGURE 4.4 – Icône - Environnement de développement Wasmote

3.5.B Configuration et tests du Meshlium

Le gateway Meshlium fournit le système Meshlium Manager comme un outil simple pour configurer la communication. l'accès sur la passerelle se fait sur l'adresse IP 10.10.10.1.

- Se connecter au routeur Meshlium



FIGURE 4.5 – Interface - Connexion sur le routeur Meshlium

- Choisir l'architecture à adopter

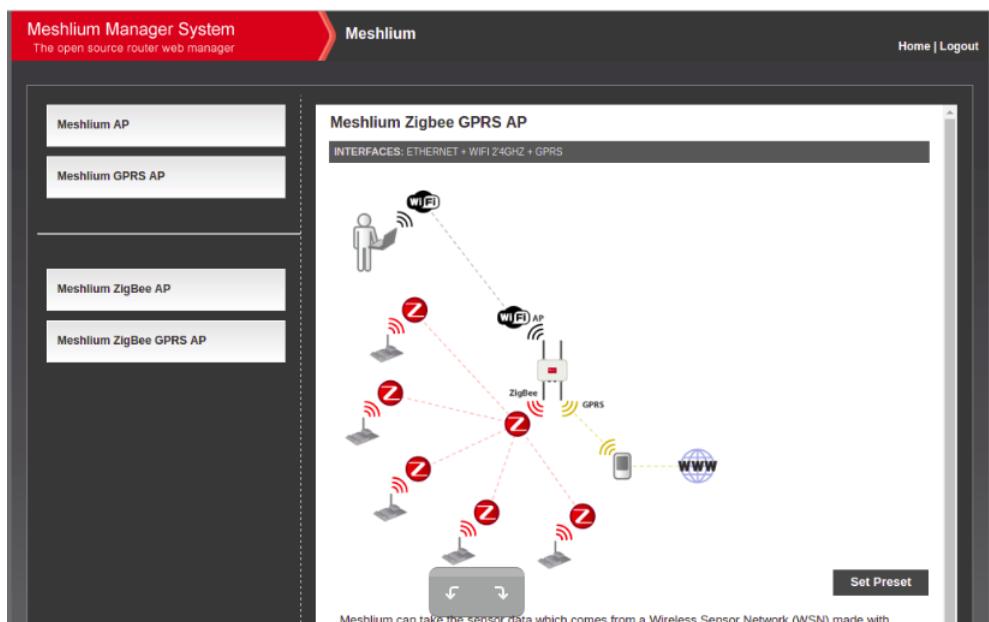


FIGURE 4.6 – Interface - Architecture adoptée : ZigBee GPRS AP

- Configuration des paramètres mobiles : On définit les paramètres mobiles nécessaires pour se connecter au Meshlium (Nom d'utilisateur, Mot de passe, Numéro de téléphone, etc). Après avoir rempli tout les champs, on sauvegarde et on essaie de

se connecter à l'opérateur pour obtenir une adresse IP valide. Une fois la connexion est établie, la passerelle par défaut de la machine est changée afin que tout les clients connectés via Wifi atteindront l'Internet via GPRS. Voir la figure suivante :

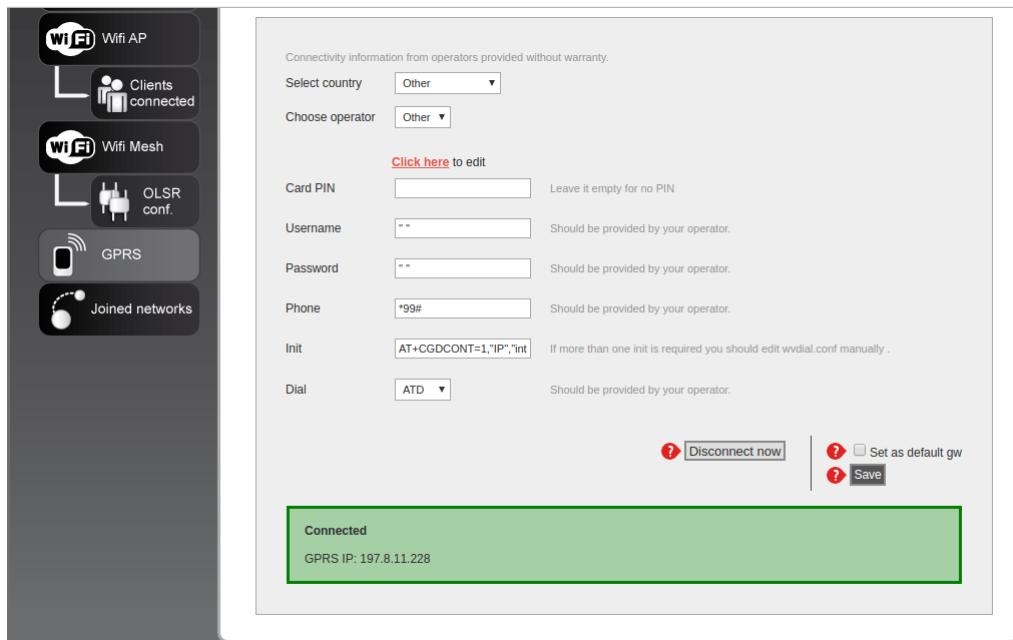


FIGURE 4.7 – Statu - connecté et une adresse ip est affecté



3.5.C Organigramme du fonctionnement du carte waspmote

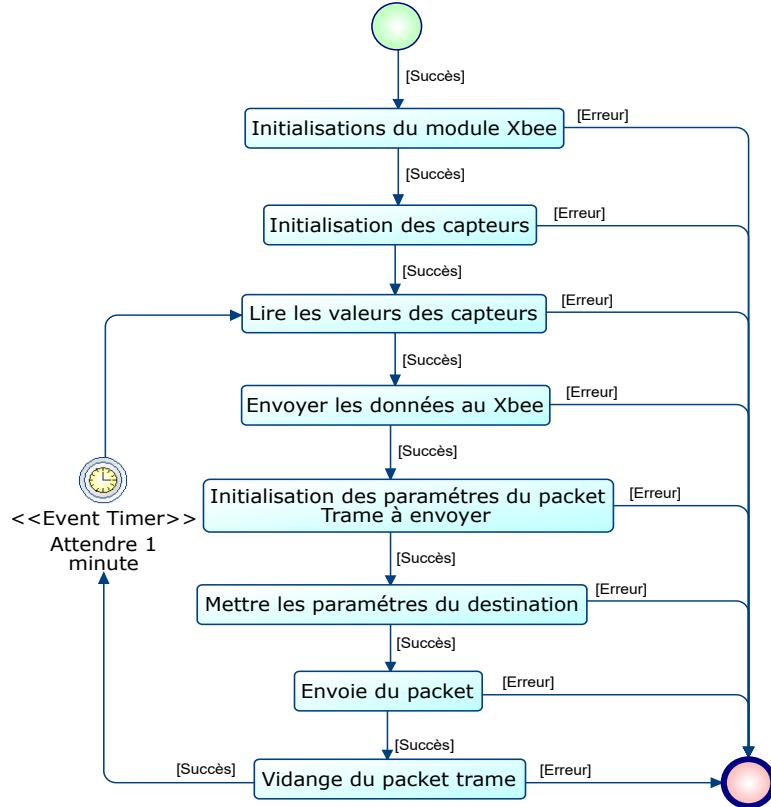


FIGURE 4.8 – Fonctionnement du carte waspmote

Lorsque la carte est activé, Elle exécute au méthode 'setup' qui va faire une initialisation au module de communication intégré Xbee et les capteurs intégrés dans la carte pour déterminer s'ils sont prés à s'exécuter. Ensuite, elle fera appel à la méthode la méthode 'loop' qui va être exécutée en un boucle infini. Cette dernière va capter les valeurs climatiques mesurée par les capteurs et les envoyés vers le serveur principal (serveur Django) pour être traitée puis sauvegardée dans la base de données. Mais avant de les envoyés, ces données climatiques vont être structurée sur un format bien définit. C'est pourquoi la carte va initialiser la paquet en ajoutant l'adresse du source et l'adresse du serveur en tant qu'une adresse de destinataire avant de l'envoyer. Après l'envioe des données, la carte va entrer dans un mode repos avant le prochain traitement pour conserver sa énergie.

4 Démonstration de la réalisation de l'application

Cette partie sera consacrée à la présentation du réalisation de l'application au cours de notre période de stage. Nous allons ainsi présenter toute les interfaces de notre application.

4.1 Interfaces : Authentification

Dans notre projet, On a deux interfaces différents. La premier est l'interface de l'utilisateur et la deuxième est l'interface de l'administrateur.

The screenshot shows a user login form titled "Connexion". The form has two input fields: "PSEUDO:" containing "Pseudo" and "MOT DE PASSE:" containing "Mot de pass". Below the inputs is a large green "Connexion" button. At the bottom of the form is a link "Annuler".

FIGURE 4.9 – Connexion utilisateur

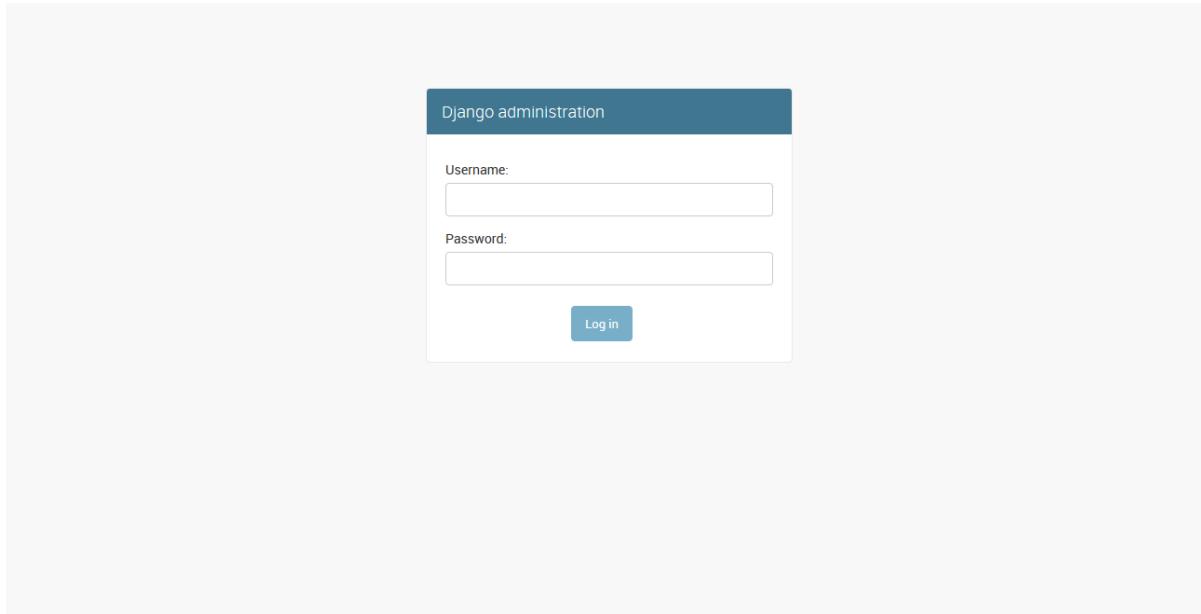


FIGURE 4.10 – Connexion administrateur

Fonctionnement : D'après les deux figures (figure 4.9 et figure 4.10), il suffit de remplir les champs (pseudo et mot de passe) puis valider en cliquant sur le bouton de validation. Le serveur va vérifier les données, si elles ne sont pas valides, une message d'erreur sera affiché au client ou administrateur pour indiquer qu'il a saisi des fausses valeurs.

4.2 Interface : Inscription

Cette interface est associé au clients non abonnées.

The screenshot shows a registration form titled "Inscription". The form fields are as follows:

- NOM:** Nom (text input field)
- PRENOM:** Prenom (text input field)
- PSEUDO:** Pseudo (text input field)
- MOT DE PASSE:** Mot de passe (text input field)
- CONFIRMER LE MOT DE PASSE:** Resaisir le mot de passe (text input field)
- ADRESSE:** (text input field)

FIGURE 4.11 – Inscription de l'utilisateur

Fonctionnement : Dans cette partie (figure 4.11), le client doit remplir les champs du formulaire avec ses données personnelles (nom, prénom, adresse, etc) puis les valider en appliquant sur le bouton de validation.

si les données sont valide, le serveur dirigera l'utilisateur vers l'interface où il peut créer un nouveau projet (figure 4.12), sinon il renvoi des messages d'erreur pour indiquer la quelle entre les valeurs saisie sont invalides.

4.3 Interface : Liste des projets

Cette interface contient tous les projets créés par le client.

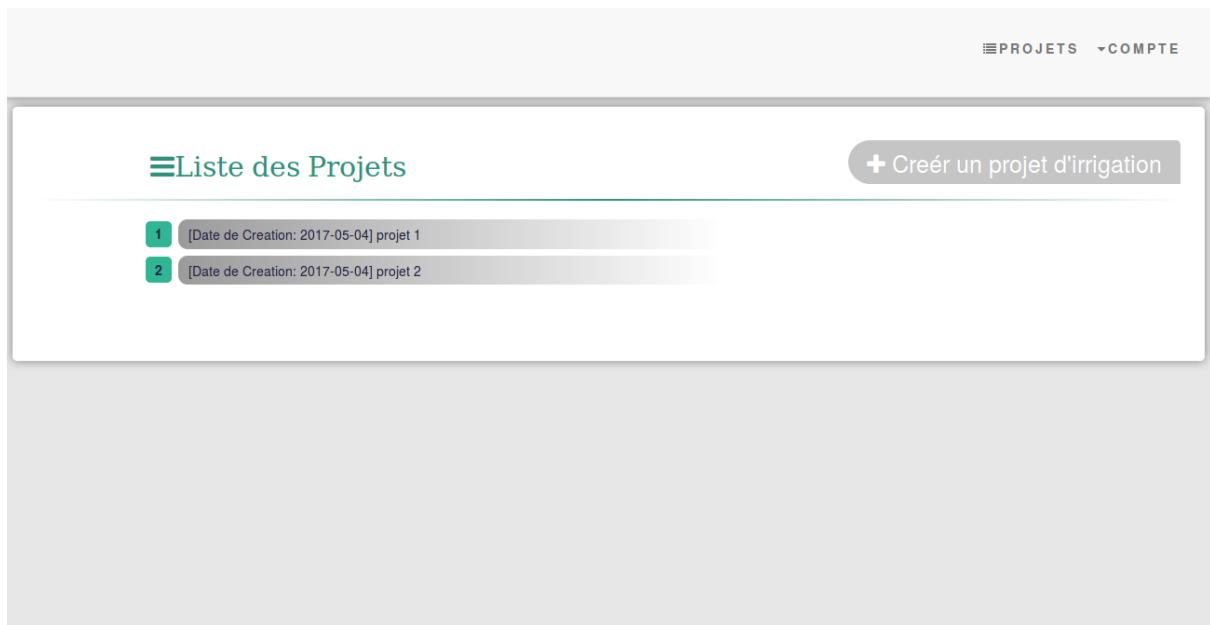


FIGURE 4.12 – Interface du liste des projets

Fonctionnement : Dans cette partie (figure 4.12), le client peut créer un nouveau projet d'irrigation (figure 4.13) en cliquant sur le bouton "Créer un projet d'irrigation" ou superviser un projet en cliquant sur son nom, ce qui va le mener vers une interface (Exemple : figure 4.18) qui contient les données climatiques journalières prévenant depuis les nœuds ou le API "OpenWeather".

4.4 Interfaces : Création de projet

Ces interfaces vont guider les clients à créer un nouveau projet d'irrigation qui satisfait leur besoins.

4.4.A Interface : Choix de type de projet

The screenshot shows a web-based application interface for creating a new irrigation project. At the top right, there are navigation links for 'PROJETS' and 'COMPTE'. The main title is '+ Creation d'un projet'. Below it, there's a field labeled 'Nom du projet:' with a placeholder 'Nom'. Underneath, a section titled 'L'objectif de ce projet est:' contains three radio button options: 'Superviser une surface en utilisant les fonctionnalités de openweather.', 'Superviser une surface en utilisant mes propres noeuds.', and 'Superviser une plante (Présence d'un noeud au moins est obligatoire!)'. A dropdown menu for 'Type d'irrigation:' is set to 'Irrigation par aspersion'. At the bottom right is a green button labeled 'Formulaire Suivante'.

FIGURE 4.13 – Interface de création de projet - Choix du type de projet

Fonctionnement : Dans cette partie (figure 4.13), le client doit saisir le nom du projet, type d'irrigation puis le type du projet.

Il existe trois types de projet :

Parcelle : Ce type existe pour les clients qui veulent créer un projet pour superviser le développement de leur plantes avec des noeuds.

Plante : Ce type existe pour les clients qui veulent créer un projet pour superviser le développement d'une seule plante avec des noeuds.

OpenWeather : Ce type existe pour les clients qui veulent créer un projet pour superviser le développement des plantes avec les fonctionnalités de l'API "OpenWeather".

4.4.B Interface : Sélectionnement des parcelles

Cette figure 4.14 sera visible seulement pour les clients qui ont choisi le type de projet d'irrigation 'Parcelle', dans le cas des deux autres choix, le serveur dirige le client vers l'interface suivante (figure 4.15) sans passer par la figure au dessous.

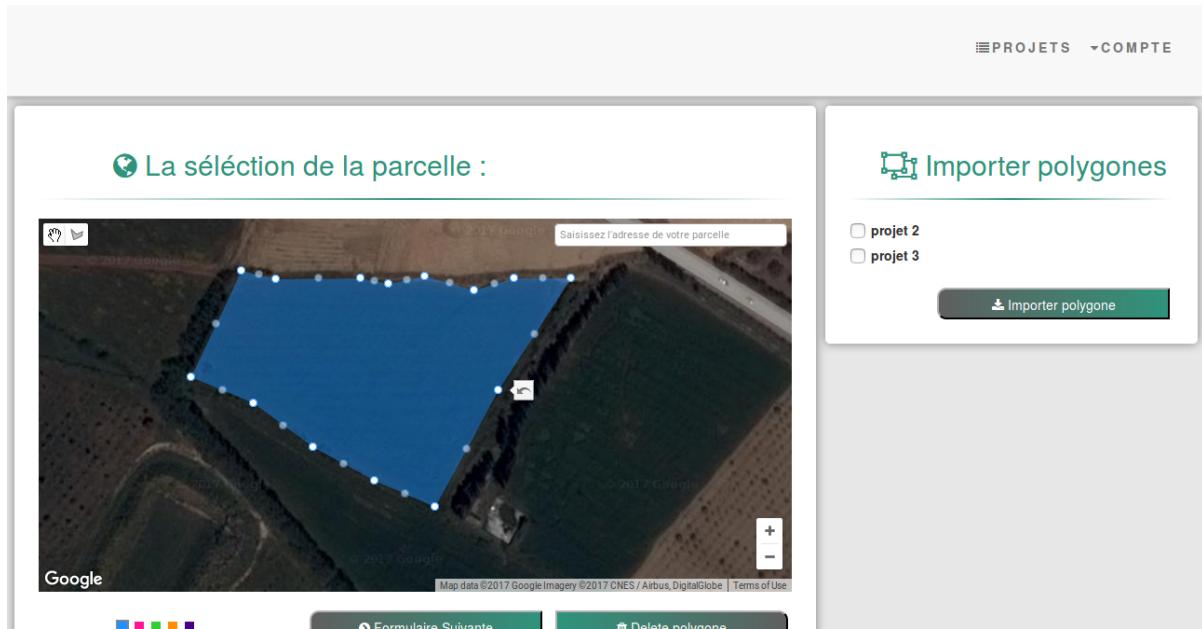


FIGURE 4.14 – Interface de création de projet - Sélectionner les parcelles

Fonctionnement : Dans cette partie (figure 4.14), le client doit graver sa parcelle dans le serveur en utilisant les fonctionnalités de Google Map.

4.4.C Interface : Choix de type de plante

Nom du plante	Phase initiale		Phase développement		Phase mi-saison		Phase récolte	
	Periode	Coeficient cultural	Periode	Coeficient cultural	Periode	Coeficient cultural	Periode	Coeficient cultural
Mon Plante 1	50/J	2.0	30/J	5.0	35/J	7.0	20/J	3.0
Tomate	15/J	0.32	30/J	0.32	40/J	1.2	40/J	1.2

FIGURE 4.15 – Interface de création de projet - Choix de type de plante

Fonctionnement : Dans cette partie (figure 4.15), le client sélectionnera la plante qu'il va planter parmi les différents choix possibles. Si la plante désirée n'existe pas, Il peut créer une nouvelle plante avec les coefficient culturaux désiré en cliquant sur le bouton "ajouter une plante".

Lors de la prochaine création d'un nouveau projet, les plantes créées par leur respectivement clients seront visible seulement pour eux.

4.4.D Interface : Choix de type de sol

The screenshot shows a user interface for creating a project, specifically for selecting soil types. At the top right, there are navigation links: 'PROJETS' and 'COMPTE'. The main title is 'Creation d'un projet: Choix du sol'. Below the title is a table with four columns: 'Nom du sol', 'Pourcentage du sable', 'Pourcentage d'argile', and 'Pourcentage du limon'. A single row is present in the table, showing 'sable' in the first column and '62.0', '19.0', and '19.0' respectively in the other three columns. At the bottom of the form are two buttons: 'Formulaire Suivante' (Next Form) and '+ Ajouter un sol' (Add a soil type).

Nom du sol	Pourcentage du sable	Pourcentage d'argile	Pourcentage du limon
sable	62.0	19.0	19.0

FIGURE 4.16 – Interface de création de projet - Choix de type de sol

Fonctionnement : Dans cette partie (figure 4.16), le client sélectionne le type de sol de sa parcelle parmi les différents choix sinon il peut ajouter un nouveau type en cliquant sur le bouton "ajouter un sol".

Les types de sols créés respectivement par leurs utilisateurs seront seulement visible par eux.

4.4.E Interface : Positionnement des nœuds

Cette figure 4.17 sera seulement visible pour les clients qui ont choisi les deux types de projet d'irrigation 'Parcelle' et 'nœud', dans le cas du choix restant, le serveur confirmera la création du projet et il dirigera le client vers l'interface (figure 4.12) qui contient la liste des projets créés.

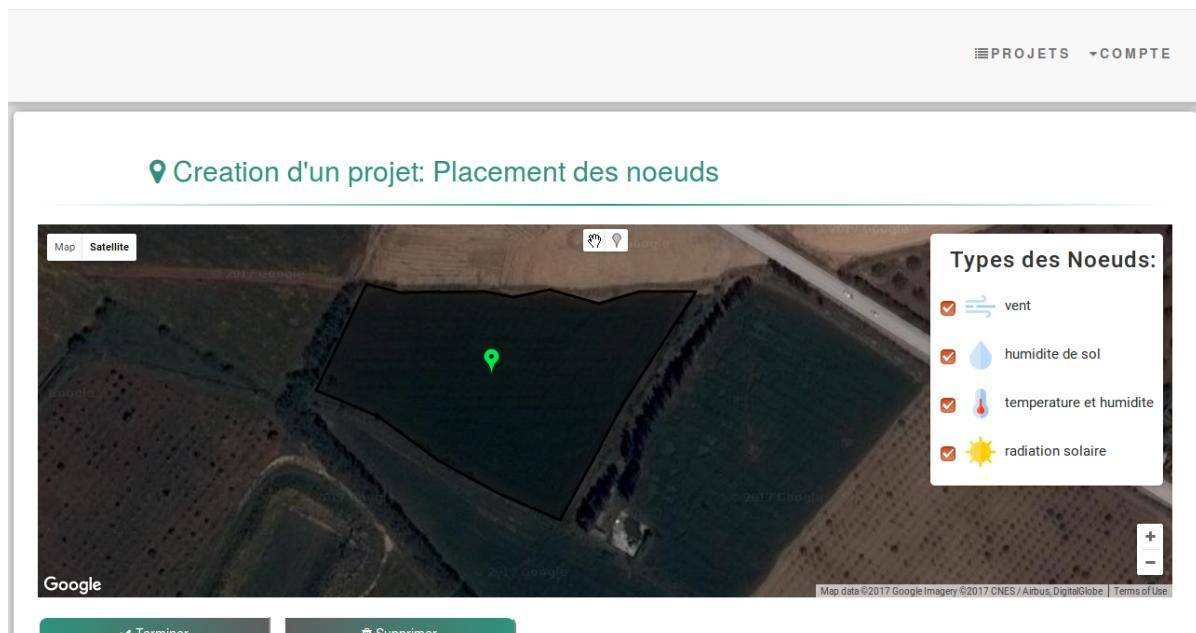


FIGURE 4.17 – Interface de création de projet - Positionner les nœuds dans la parcelle

Fonctionnement : Dans cette partie (figure 4.17), le client doit mettre la position de chaque nœud ainsi que leur fonctionnalité (type des capteurs) dans google map.

4.5 Interface : projet

Ce genre d'interface est consacrée dans le but de permettre aux clients de superviser le développement des plantes et de suivre les données climatiques prévenant depuis les noeuds ou depuis l'API "OpenWeather".



FIGURE 4.18 – Interface du projet

Fonctionnement : Dans cette partie (figure 4.18), s'il y a eu des problèmes avec les nœuds, le serveur affichera des messages d'erreurs au dessus des graphes.
pour supprimer le projet, il suffit de cliquer sur le bouton "Suprrimer le projet".

4.6 Interfaces : Administrateur

Cet ensemble des interfaces permet au administrateur de gérer la base de données

4.6.A Interface : Les tableaux

APPLICATION	
Climat journalières	+ Add ⚡ Change
Noeuds	+ Add ⚡ Change
Plantes	+ Add ⚡ Change
Polygones	+ Add ⚡ Change
Projets	+ Add ⚡ Change
Sols	+ Add ⚡ Change
Utilisateurs	+ Add ⚡ Change

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add ⚡ Change
Users	+ Add ⚡ Change

Recent actions

My actions

- ⚡ [Date de Creation: 2017-05-04] projet 2 Projet
- ✖ [Date de Creation: 2017-05-04] Pomme Projet
- ✖ PROJET INCOMPLET: benouri Projet
- ✖ PROJET INCOMPLET: azerazerazer Projet
- ✖ PROJET INCOMPLET: sol test Projet
- ✖ PROJET INCOMPLET: testsetset Projet
- ✖ PROJET INCOMPLET: ok Projet
- ✖ PROJET INCOMPLET: okokoqkscqsc Projet
- ✖ [Date de Creation: 2017-05-07] plante test Projet

FIGURE 4.19 – Interface d'administrateur

Fonctionnement : Cette interface contient tout les tableaux qui existent dans la base de données et dont l'administrateur a l'autorisation de les modifier. Les deux tableaux ("Groups" et "Users") sont des tableaux qui permettent l'administrateur de créer/ajouter un groupe/utilisateur en tant qu'administrateur et leurs affecter des droits spécifiques (comme le droit de supprimer des données dans un certain tableau depuis la base de données). Elle permet à l'administrateur de voir les action de modification, ajout ou suppression récemment effectué dans la base de données.

4.6.B Interface : Gestion des données

ID	NOM	PRENOM	PSEUDO	MOT DE PASSE	ADRESSE	POSITION	TELEPHONE	EMAIL
2	hamdi	med alaa	mrfahrenheit	bbbb	Rue d'Alger, Marseille, France	SRID=4326:POINT (5.39006289999975 43.289336)	25059460	hamdi.med.alaa@gmail.com
1	bennouri	iheb	bennouri	123	El Omrane supérieur, Tunis, Tunisia	SRID=4326:POINT (10.124073 36.8305331)	55612969	bennouri.iheb@gmail.com

FIGURE 4.20 – Exemple d’interface de gestion des données dans le tableau Utilisateurs

Fonctionnement : Cette interface est consacrée pour visualiser les données sauvegardées dans un tableau comme dans la figure si-dessus (figure 4.20). Elle permet aussi de filtrer les valeur par une valeur donnée saisie par l’administrateur, et d’effectuer des action (ajout, modification et suppression) sur le tableau sélectionné.

Django administration

WELCOME, BENNOURI [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Home · Application · Utilisateurs · Add utilisateur

Add utilisateur

Nom:	<input type="text"/>
Prenom:	<input type="text"/>
Pseudo:	<input type="text"/>
Mot de passe:	<input type="text"/>
Photo:	<input type="button" value="Browse..."/> No file selected.
Telephone:	<input type="text"/>
Email:	<input type="text"/>
Adresse:	<input type="text"/>
Position:	

FIGURE 4.21 – Exemple d’interface permet d’ajouter un nouveau utilisateur dans le tableau Utilisateurs

Django administration

WELCOME, BENNOURI [VIEW SITE / CHANGE PASSWORD / LOG OUT](#)

Home · Application · Utilisateurs · Utilisateur: hamdi med alaa, Pseudo: mrfairenheit

Change utilisateur

HISTORY

Nom:	hamdi
Prenom:	med alaa
Pseudo:	mrfairenheit
Mot de passe:	bbbb
Photo:	Currently: application/mrfairenheit.jpg Change: <input type="button" value="Browse..."/> No file selected.
Telephone:	25059460
Email:	hamdi.med.alaa@gmail.com
Adresse:	Rue d'Alger, Marseille, France
Position:	

FIGURE 4.22 – Exemple d’interface permet de modifier les valeurs du utilisateur hamdi

Fonctionnement du site d'administrateur : La figure 4.19 est consacrée pour que l'administrateur puisse faire la gestion des données, Elle contient tout les tableaux existant dans la base de données.

Pour que l'administrateur puisse exécuté une action (ajout, modification ou suppression), Il doit sélectionner un tableau parmi eux, ce qui va le prendre vers une autre interface [exemple : figure 4.20] qui affichera les données existant. Depuis ce dernier l'administrateur peut :

- Ajouter un nouveau ligne dans le tableau en cliquant sur le bouton 'ajouter', ce dernier dirigera l'administrateur vers une interface spécialisée [exemple : figure 4.21].
- Supprimer une ligne existante en choisissant l'action de suppression (existe dans la liste qui est positionnée au dessus du tableau).
- Modifier une ligne en cliquant sur l'ID de la ligne désirée, ce qui va transmettre l'administrateur vers une autre interface [exemple : figure 4.22].

4.7 Interfaces : Les consoles des serveurs

4.7.A Interface : Serveur Django

```
benouri@benouri-Inspiron-N5010:~/Documents/serveur$ python manage.py runserver 192.168.1.5:8000 --insecure
Performing system checks...

System check identified no issues (0 silenced).
May 29, 2017 - 09:18:14
Django version 1.10.6, using settings 'serveur.settings'
Starting development server at http://192.168.1.5:8000/
Quit the server with CONTROL-C.
MIDDLEWARE [Contrôleur]: <function _get_response at 0x7f271a338c0>
MIDDLEWARE [Contrôleur]: <function _get_response at 0x7f26f9205938>
[29/May/2017 09:18:38] "GET / HTTP/1.1" 302 0
REQUEST -> /application/
[29/May/2017 09:18:39] "GET /application/ HTTP/1.1" 200 4535
[29/May/2017 09:18:39] "GET /static/css/bootstrap.min.css HTTP/1.1" 200 122540
[29/May/2017 09:18:39] "GET /static/css/navigation.css HTTP/1.1" 200 564
[29/May/2017 09:18:39] "GET /static/css/fonts.googleapis.com.Playball.css HTTP/1.1" 200 740
[29/May/2017 09:18:39] "GET /static/js/jquery.min.js HTTP/1.1" 200 268382
[29/May/2017 09:18:39] "GET /static/css/accueil.css HTTP/1.1" 200 6268
[29/May/2017 09:18:39] "GET /static/css/fonts.googleapis.com.Raleway.css HTTP/1.1" 200 6685
[29/May/2017 09:18:39] "GET /static/js/modernizr.custom.js HTTP/1.1" 200 15246
[29/May/2017 09:18:39] "GET /static/js/main.js HTTP/1.1" 200 1125
[29/May/2017 09:18:39] "GET /static/images/background.jpg HTTP/1.1" 200 680341
[29/May/2017 09:18:39] "GET /static/images/drops.png HTTP/1.1" 200 861
REQUEST -> /application/inscription/
[29/May/2017 09:18:56] "GET /application/inscription/ HTTP/1.1" 200 3601
[29/May/2017 09:18:56] "GET /static/css/fonts.css HTTP/1.1" 200 6685
[29/May/2017 09:18:56] "GET /static/css/fonts.playball.css HTTP/1.1" 200 740
[29/May/2017 09:18:56] "GET /static/css/base.css HTTP/1.1" 200 4604
[29/May/2017 09:18:56] "GET /static/js/bootstrap.min.js HTTP/1.1" 200 35951
[29/May/2017 09:18:56] "GET /static/css/formulaire.css HTTP/1.1" 200 1833
[29/May/2017 09:18:56] "GET /static/images/drops.png HTTP/1.1" 200 801
REQUEST -> /application/inscription/
[29/May/2017 09:18:56] "GET /application/inscription/ HTTP/1.1" 200 3601
[29/May/2017 09:18:56] "GET /static/css/fonts.css HTTP/1.1" 200 6685
[29/May/2017 09:18:56] "GET /static/css/fonts.playball.css HTTP/1.1" 200 740
[29/May/2017 09:18:56] "GET /static/css/base.css HTTP/1.1" 200 4604
[29/May/2017 09:18:56] "GET /static/js/bootstrap.min.js HTTP/1.1" 200 35951
[29/May/2017 09:18:56] "GET /static/css/formulaire.css HTTP/1.1" 200 1833
```

FIGURE 4.23 – Interface du console du serveur principale

Fonctionnement : Cette console va afficher toutes les requêtes, réponses et redirections exécutés par le serveur Django.

4.7.B Interface : Serveur rempir_journal

```

benouri@benouri-Inspiron-N5010:~/Documents/serveur$ python manage.py shell
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> execfile('rempli_journal.py')
Traceback (most recent call last):
  File "<console>", line 1, in <module>
IOError: [Errno 2] No such file or directory: 'rempli_journal.py'
>>> execfile('remplir_journal.py')
Remplir journal: on
Traitement en cours...
    > Traitement des noeuds:
        > Traitement des donnees de [noeud: d0cd103e-f8ad-4fdb-aee5-3a5a04f7585d, projet: projet 1]
        >>> Les donnees ont etait traiter.
        > Traitement des donnees de [noeud: 2e900195-d78f-468c-9543-16584d331910, projet: projet 1]
        >>> Les donnees ont etait traiter.
        > Traitement des donnees de [noeud: 4450a6ce-1ddb-4884-8f37-1abb4bde8a6c, projet: projet 1]
        >>> Les donnees ont etait traiter.
    >>> NOEUDS: Terminer avec succes
    > Traitement des donnees de OpenWeather:
        > Traitement des donnees du projet [projet 2]
        >>> Traitement est terminer avec succes.
    >>> OPENWEATHER: Terminer avec succes
    > Envoie des Message pour alerter les abonnees...
        > [nom: projet 2]: Message'd>alert est envoier.
    >>> ENVOIE: Terminer avec succes
> JOURNAL: mise a jour est terminer avec succes

```

FIGURE 4.24 – Interface du console du serveur remplir_journal

Fonctionnement : Chaque soirée, ce serveur va calculer les quantités d'eau nécessaires pour chaque projet puis les envoyer dans un mail vers leur utilisateur correspondant. Cette console affichera le résultat de l'exécution des traitements pour indiquer s'il y a eu une erreur.

4.7.C Interface : Serveur planning_owm

```
benouri@benouri-Inspiron-N5010:~/Documents/serveur$ python manage.py shell
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> execfile('planning_owm.py')
Planning owm: on
> Traitement en cours...[1 projet va etre mis a jour]
> [nom: "projet 2"] est a jour.
> Traitement termine avec succes.
> Traitement en cours...[1 projet va etre mis a jour]
> [nom: "projet 2"] est a jour.
> Traitement termine avec succes.
```

FIGURE 4.25 – Interface du console du serveur planning_owm

Fonctionnement : Après chaque petite période, ce serveur va sauvegarder les données climatiques de l'instant de l'exécution pour chaque projet qui n'utilise pas les pièces Arduino. Cette console affichera le résultat de l'exécution réalisée pour indiquer s'il y a eu une erreur.

5 Conclusion

Dans ce chapitre de notre rapport nous avons présenté tout les outils et moyens de réalisation de notre application ainsi que les interfaces principaux du l'application.

Conclusion générale et perspectives

Notre projet a été réalisé en vue de l'obtention du diplôme de licence fondamentale en informatique de la Faculté des Sciences de Tunis. Ce projet consiste en la mise en place d'un système d'aide à la décision en irrigation à base des réseaux de capteurs sans fil.

La problématique de notre projet consiste à l'application exacte et précise de l'eau pour répondre aux besoins spécifiques et particuliers de chaque plante qui est devenue un enjeu capital dans le domaine agricole. En fait, l'irrigation de précision réduit énormément la perte d'eaux pour éviter une crise des ressources hydriques.

Notre sujet couvre la conception, le développement et le déploiement d'une application web qui estime les besoins réelles de la plante en eau.

Pour cela, nous avons mené une étude du concept d'irrigation de précision pour entamer notre projet. Certes, plusieurs difficultés se sont présentées face à nous et nous ont compliquées la tâche vu l'aspect technique du projet et le manque de documentation, mais grâce à notre motivation et à la mise en collaboration de nos différentes expériences, tous nos travaux ont été gratifiés de succès.

Après avoir choisi une approche pour la suivre durant le développement de notre solution, nous avons commencé par préparer l'environnement de travail nécessaire. Ensuite, nous avons passé à la phase spécification et conception et le développement du logique métier. Enfin nous avons terminé par la phase de réalisation.

Réaliser en milieu professionnel, ce projet nous a permis de confronter plusieurs contraintes que ce soit celles de temps, de matériels, de technologies et de concertations. Ce travail s'est avéré particulièrement formateur de point de vue technique.

Nous avons découvert le monde d'irrigation d'eaux, les réseaux de capteurs sans fil et toutes les connaissances informatiques en relation avec ces domaines. En effet, nous avons fondé nos bases concernant le langage Python, le framework Django et le concept des bases de données spatiales. Nous y avons trouvé un grand intérêt car ce sont des connaissances qui ne nous sont pas toutes enseignées à l'université et qui viennent compléter notre formation.

Sur le plan personnel, ce stage nous a été vraiment d'un grand apport sur le plan humain, il nous a offert l'opportunité de nous intégrer dans l'environnement de l'entreprise et d'améliorer nos capacités de communication, d'adaptation à la vie professionnelle et au travail en équipe, ce qui sera un atout dans un future proche.

Le projet est totalement accompli avec tous ses fonctionnalités. Nous pouvons aussi enrichir ce projet au niveau graphique et ergonomique. Enfin, nous espérons que le travail que nous avons effectué soit à la hauteur.

Bibliographie et Netographie

- [1] Django. Documentation officiel sur l'utilisation de django. <https://docs.djangoproject.com/en/1.11/>. Accessed : 2017-02-13.
- [2] Django. Télécharger django. <https://www.djangoproject.com/download/>. Accessed : 2017-02-13.
- [3] Django. Tutoriel sur l'utilisation de django. <https://tutorial.djangogirls.org/fr/>. Accessed : 2017-02-13.
- [4] Richard G.ALLEN, Luis S.PEREIRA, Dirk RAES, and Martin SMITH. Crop evapo-transpiration, 2006.
- [5] Google Map. Tutoriel sur l'utilisation du api de javascript de google map. <https://developers.google.com/maps/documentation/javascript/?hl=fr>. Accessed : 2017-02-13.
- [6] PostGIS. Télécharger le module postgis pour postgresql. <http://postgis.net/install/>. Accessed : 2017-02-13.
- [7] PostgreSQL. Télécharger postgresql. <https://www.postgresql.org/download/>. Accessed : 2017-02-13.
- [8] Mark Richards. Pyeto, 2015.
- [9] Ali Shali and Mohamed Jabloun. *MABIA-Region*. Institut National Agronomique de Tunisie, 2009.

Annexe A

Partie du code de la carte Wasp mote

```
/*Cette méthode va être exécuter en boucle infini*/
void loop() {
    /*Lire les valeurs des capteurs*/
    pression = SensorAgr.readValue(SENS_AGR_PRESSURE);
    temperature = SensorAgr.readValue(SENS_AGR_TEMPERATURE);
    moisture = SensorAgr.readValue(SENS_AGR_WATERMARK_1);
    luminosite = SensorAgr.readValue(SENS_AGR_TEMPERATURE);
    /*Envoyer les données au Xbee*/
    XBee.println(pression);
    XBee.println(temperature);
    XBee.println(moisture);
    XBee.println(luminosite);
    sprintf(aux," Noeud :4 temperature :%d pression :%d moisture :%d luminosite :%d%c%c",
    (int)SensorAgr.readValue(SENS_AGR_TEMPERATURE),
    (int)SensorAgr.readValue(SENS_AGR_PRESSURE),
    (int)SensorAgr.readValue(SENS_AGR_WATERMARK_1),
    (int)SensorAgr.readValue(SENS_AGR_LEAF_WETNESS), 'r', 'n');
    /*Initialisation des paramètres du packet Trame 'a envoyer*/
    paq_sent = (packetXBee) malloc(1, sizeof(packetXBee));
    paq_sent->mode = BROADCAST;
    paq_sent->MY known = 0 ;
    paq_sent->packetID = 0x52;
```

```
paq_sent->opt = 0 ;  
xbee802.hops = 0 ;  
xbee802.setOriginParams(paq_sent, MAC_TYPE) ;  
/*Mettre les param`etres du destination*/  
xbee802.setDestinationParams(paq_sent, direccion, aux, MAC_TYPE, DATA_ABSOLUTE) ;  
/*Envoie du packet*/  
xbee802.sendXBee(paq_sent) ;  
/*Vidange du packet trame*/  
free(paq_sent) ;  
paq_sent = NULL ;  
/*Mettre la pi`ece en attente pour la prochain ex`ecution dans le but d'`economiser  
l'`energie*/  
delay(10000) ;  
}
```

Résumé :

Ce travail est réalisé dans le cadre de notre projet de fin d'études dont l'objectif est de réaliser des applications à base de l'informatique et de l'internet des objets afin d'améliorer l'utilisation en eau d'irrigation dans les oasis de palmier dattier.

Mots clés : Irrigation, RCSF

Abstract :

The training course has the objective to create an application named IrriSmart, it contains many services like : management of weather datas, calculating the Et0 of each project then send the result to it's respective user.

Keywords : Irrigation, RCSF